Pattern Recognition 102 (2020) 107187

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

PoseConvGRU: A Monocular Approach for Visual Ego-motion Estimation by Learning

Guangyao Zhai[a,1], Liang Liu[a,1], Linjian Zhang[a,b], Yong Liu[a,*], Yunliang Jiang[c]

[a] Institute of Cyber-Systems and Control, Zhejiang University, China
[b] Fuxi AI Lab, Netease Inc., China
[c] School of Information Engineering, Huzhou University, China

ARTICLE INFO

Article history:
Received 2 June 2019
Revised 15 October 2019
Accepted 24 December 2019
Available online 21 January 2020

Keywords:
Ego-motion
Pose estimation
Deep learning
Recurrent Convolutional Neural Networks
Data augmentation

ABSTRACT

Visual ego-motion estimation is one of the longstanding problems which estimates the movement of cameras from images. Learning based ego-motion estimation methods have seen an increasing attention since its desirable properties of robustness to image noise and camera calibration independence. In this work, we propose a data-driven approach of learning based visual ego-motion estimation for a monocular camera. We use an end-to-end learning approach in allowing the model to learn a map from input image pairs to the corresponding ego-motion, which is parameterized as 6-DoF transformation matrix. We introduce a two-module Long-term Recurrent Convolutional Neural Networks called PoseConvGRU. The feature-encoding module encodes the short-term motion feature in an image pair, while the memory-propagating module captures the long-term motion feature in the consecutive image pairs. The visual memory is implemented with convolutional gated recurrent units, which allows propagating information over time. At each time step, two consecutive RGB images are stacked together to form a 6-channel tensor for feature-encoding module to learn how to extract motion information and estimate poses. The sequence of output maps is then passed through the memory-propagating module to generate the relative transformation pose of each image pair. In addition, we have designed a series of data augmentation methods to avoid the overfitting problem and improve the performance of the model when facing challengeable scenarios such as high-speed or reverse driving. We evaluate the performance of our proposed approach on the KITTI Visual Odometry benchmark and Malaga 2013 Dataset. The experiments show a competitive performance of the proposed method to the state-of-the-art monocular geometric and learning methods and encourage further exploration of learning-based methods for the purpose of estimating camera ego-motion even though geometrical methods demonstrate promising results.

1. Introduction

The capability of self-localization during the movement for intelligent vehicles is especially important. The method, which can estimate the position of the vehicle by integrating data of various sensors, is called odometry. With the development of computer vision technology, more and more visual sensors are used for vehicle positioning and motion estimation. We refer to the problem of obtaining camera poses through vision as Visual Odometry (VO) [1] or visual ego-motion [2]. The visual sensor not only provides rich sensory information but also has the advantages of low cost and tiny size. The mainstream visual ego-motion methods mainly estimate camera poses based on the geometrical characteristics of the objects in the images, so the images must contain a large number of stable texture features. Once these methods proceed in the scene with some obstacles or on a foggy day without other sensors (IMU, laser, etc.), the accuracy will be severely decreased. Since the camera can capture rich information of the scene with a modest hardware cost and can be more reliable than other sensors in many instances, localization and ego-motion estimation techniques based on visual methods still have an immense potential for both research and applications.

Recently, researchers in various fields have paid much attention to the deep learning study [3–5]. Developed to present days, those deep learning approaches represented by convolutional neural networks play significant roles in the field of computer vision [6,7]. These deep neural networks are more effective in extracting image features and finding potential patterns than traditional methods. Therefore, some related researchers consider applying deep

* Corresponding author.
E-mail address: yongliu@iipc.zju.edu.cn (Y. Liu).
[1] Contribute equally to this work.

https://doi.org/10.1016/j.patcog.2019.107187
0031-3203/

learning in the field of visual ego-motion research, which is using the deep neural network directly to learn the geometric relationship through images and then to realize the end-to-end pose estimation. This end-to-end process can completely eliminate the steps of feature extraction, feature matching, camera calibration, and graph optimization in the traditional methods, and directly obtains the camera poses according to the input images.

This paper mainly focuses on the problem of camera relative pose estimation by deep learning, only considering the situation of monocular VO. We introduce a novel Long-term Recurrent Convolutional Neural Networks, containing two modules, called PoseConvGRU. The feature-encoding module extracts the short-term motion feature in an image pair, while the memory-propagating module captures the long-term motion feature in the consecutive image pairs to estimate camera poses. We use a regularization loss term on the sequence of adjacent image pairs to improve the accuracy of the estimation of camera poses and preserve temporal consistency. We perform experiments on the KITTI Visual Odometry benchmark [8] and Malaga 2013 Dataset [9]. The main contributions are as follows:

- We propose a novel framework named PoseConvGRU, a monocular approach of visual ego-motion estimation, which is data-driven and fully trainable.
- We design a series of data augmentation methods for avoiding the overfitting problem and improve the performance of the model when proceeding through challengeable scenarios to the greatest extent such as high-speed or reverse situation.
- Our proposed neural network does not matter with the optical flow or other flow-like subspace, unlike other learning-based ego-motion estimation algorithms, which need to spend plenty of time to calculate the pre-processed dense optical flow before training the neural network [10] or use a pre-trained network to estimate the optical flow with additional calculation costs [11,12].
- Our method shows a competitive performance to state-of-the-art monocular geometric and learning methods, encouraging further research of learning-based methods.

2. Related work

2.1. Progress of visual odometry in geometric research

Matthies et al. proposed to implement indoor robotic navigation through visual input. The leading research at that time was based on the NASA Mars Exploration Program [13]. The real foundation for the VO problem is a real-time visual odometry designed by Nister et al. [1], which builds its implementary framework.

Based on this framework, the solution to the VO problem can be further divided into two sorts of methods: feature-point methods and direct methods:

Feature-point methods mainly extract the feature points in adjacent frame images, extracting the geometric relationship of the feature points by using multi-view geometry to estimate the relative camera poses, such as LIBVISO2 [14], ORB-SLAM [15]. However, these methods are time-consuming when extracting features, and only concern about the extracted feature points while ignoring the abundant information of other pixels in images, so that features extracted from images are not sufficient to restore visual ego-motion if the image texture information captured by the camera is severely scarce, and typically these methods will not work properly.

Direct methods based on the assumption of intensity value invariant can adapt more scenes. The apparent difference between the direct methods and the feature-point methods is that the former ones use the luminance information of the pixels in the images to estimate the visual ego-motion directly rather than calculating the descriptors and the key points. These methods avoid the prolonged calculation time of features and the problems caused by the lack of features. According to the number of pixels used, the direct methods can be further divided into three types: sparse ones, dense ones, and semi-dense ones. Open-sourced projects using direct methods such as SVO [16], LSD-SLAM [17], DSO [18] have gradually become essential parts of the visual ego-motion algorithm.

2.2. Progress of visual odometry in deep learning

Roberts et al. [19] proposed to study visual ego-motion problems by a traditional learning method. They map sparse optical flow to velocity state by using n separate k-Nearest-Neighbors, but this method did not achieve the camera 3-DoF position estimation and receives a substantial error additionally. The first use of deep learning to study visual ego-motion problems is a method introduced by Kishore et al. [20]. Their method bases on the two-staged operation, using stereo images to estimate depth firstly and designing two separate convolutional neural networks for learning discrete velocity and direction angle. They subtly transform the pose regression problems into classification problems, but the reliability of the obtained result is modest since the massive error has been generated in the process of discretizing the direction angle and velocity. The first method for end-to-end estimation of camera 6-DoF pose based on convolutional neural networks is PoseNet proposed by Kendall et al. [21]. The neural network framework of this method was modified from GoogLeNet [22]. Due to PoseNet's inaccurate estimation for handling scenes with some obstacles, Kendall et al. [23] proposed a Bayesian Convolutional Neural Network to make the regression of the camera 6-DoF poses. The advantage of using a Bayesian Convolutional Neural Network is that it can measure uncertainties of the camera's poses and use these uncertainties to estimate the localization error and determine whether the test images are repeated. Mohanty et al. proposed DeepVO [24]. The CNN part of this method is based on AlexNet [25]. It inputs two adjacent RGB images and directly estimates the relative pose between the two images in an end-to-end way. For scenes that have not appeared in the training set before, the estimation results are unsatisfactory. This work attempts to regard the FAST features of the images as additional input information, but it cannot fundamentally solve the scenario migration problem. On the basis of utilizing RGB images, there is a kind of method of using optical flow to help obtain ego-motion. Costante et al. [10] proposed two CNN structures to estimate the frame-to-frame poses. Since this method requires the input of optical flow images, pre-processing of optical flow will cost more calculation resources, and the accuracy of optical flow calculation has a great influence on the visual ego-motion estimation, so this sort of method is difficult to be widely used.

Ronald et al. proposed VINet [26]. It not only integrates IMU information into the deep neural network to study visual ego-motion problems but also applies sequence learning to consider the pose relationship among multiple frames of images. This paper provides a novel approach to Visual Inertial Odometry (VIO) field, and the combination of CNN and LSTM for sequence learning has contributed dramatically to subsequent research. The authors above also propose VidLoc [27], using CNN and LSTM to estimate the global poses of consecutive frames, which achieves a significant improvement over PoseNet [21]. In the same year, another author of the paper published a new version of DeepVO [28], and further proposed ESP-VO [29]. DeepVO uses the image sequences as input. Firstly, the image features are extracted by CNN, put into the RNN to learn the geometry relationship among successive frames, and then the relative poses of multi-frame are directly output. The relative transformation poses between the images are positive compared to all the previous research work. Iyer et al. [30]

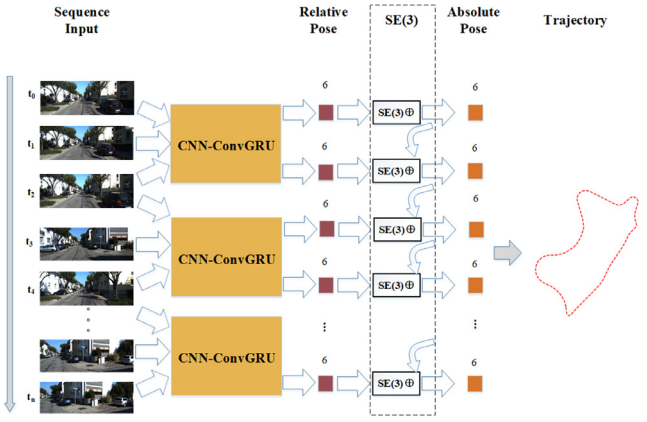

Fig. 1. Our proposed end-to-end framework PoseConvGRU can estimate visual ego-motion by extracting geometrical feature among adjacent monocular RGB images. We can draw the trajectory after obtaining all the absolute poses.

achieves an unsupervised paradigm for deep visual ego-motion learning.

The existing visual ego-motion estimation methods based on CNN are far less effective than the geometry-based methods. However, most methods based on the geometric model cannot benefit from continuously increased datasets. In addition, some traditional visual methods cannot proceed with challenging scenes (like Malaga dataset) since they estimate poses by detecting the motion of the feature points or pixels in images, which are susceptible to the surrounding environment and dynamic objects, while deep learning methods learn to extract intrinsic correlation of interframes or videos by completing pose-regression missions, hopefully avoiding problems caused in traditional methods.

3. Methodology

The visual ego-motion problems are quite different from those of classification, tracking:

Firstly, visual ego-motion estimation based on deep learning is a regression problem. It is not possible to accurately obtain the relative pose of two adjacent frames by simply identifying or detecting the objects in the images;

Secondly, visual odometry problem needs to process two images at the same time, and it is especially related to the order of the images as the relative poses between the two images can be numerically reverse from each other based on their respective references, so that we can obtain two different results.

Therefore, we can not merely use the popular neural network frameworks such as VGGNet [31] or DenseNet [32] to solve this estimation problem, but should adopt the structure that can learn the geometric features of the images. The overall framework is shown in Fig. 1.

3.1. The structure of PoseConvGRU

Feature-encoding module: In order to use the effective CNN structure to learn the geometric relationship from two adjacent images automatically, our approach leverages the network structure proposed by Dosovitskiy et al. - FLowNetSimple [33] but ignores the decoder part in the network, only focusing on the front convolution encoder. In DeepVO, the output feature maps of Conv6-1 are directly input into subsequent modules, which not only hugely increases network parameters and magnifies the storage of GPU, but also makes the training complexity of the network expanded. To alleviate this problem, we add a layer of Max-pooling behind the Conv6-1 layer to reduce dimensions of the feature maps.

Table 1
CNN parameters. We can see the size of kernels decreases more as the depth of the network going deeper and the size of the feature maps decreases further.

Layer	Kernel size	Stride	Weights	Tensor size
Input	-	-	-	1280 × 384 × 6
Conv1	7 × 7	2	6 × 64	640 × 192 × 64
Conv2	5 × 5	2	64 × 128	320 × 96 × 128
Conv3	5 × 5	2	128 × 256	160 × 48 × 256
Conv3_1	3 × 3	1	256 × 256	160 × 48 × 256
Conv4	3 × 3	2	256 × 512	80 × 24 × 512
Conv4_1	3 × 3	1	512 × 512	80 × 24 × 512
Conv5	3 × 3	2	512 × 512	40 × 12 × 512
Conv5_1	3 × 3	1	512 × 512	40 × 12 × 512
Conv6	3 × 3	2	512 × 1024	20 × 6 × 1024
Conv6_1	3 × 3	1	1024 × 1024	20 × 6 × 1024
Max-pooling	2 × 2	2	–	10 × 3 × 1024

The parameters of the CNN are shown in Table 1. As shown in Fig. 2, the convolutional neural network contains a total of 10 convolutional layers, and each layer is followed by a nonlinear activation function - Rectified linear unit (ReLU). The number of convolution kernels increases gradually as the depth of the network expands so that more feature maps can be obtained, which can represent more abstract features, and the decreasing of the feature map size means that the CNN is paying more attention to large-scale and significant features. The size of the convolution kernel is also gradually reduced from 7 × 7 to 5 × 5 and finally to 3 × 3 for capturing local features. The input of CNN is the original continuous multi-frame RGB images, resized to 1280 × 384. Assuming that sequence's length is n + 1, we can obtain n sets of image pairs when adjacent two frames are combined in a sliding window. These image pairs are respectively subjected to 10 convolutional layers and the last Max-pooling layer to obtain feature maps of 10 × 3 × 1024 size. For multiple pairs of images generated by the same time series, we refer to the structure of the Siamese network [34], using different branches to deal with similar problems, but will keep the CNN parameters weight-shared in the same time series, i.e., all the images of a sequence perform the feature extraction through the same CNN layer. We do not perform any pre-processing operation such as random clipping and rotating to change the geometric relationship of the objects in the images so that the original information of the images can be used for accurate pose estimation.

Memory-propagating module: The memory module builds a "visual memory" in the video clip, i.e., a long-term joint visual representation of all the clip frames to generate the transformation pose of each pair since it allows the neural network to automatically learn the intrinsic relationship among successive poses, module structure shown in Fig. 3. We use a stacked Convolutional Gated Recurrent Units (ConvGRU) [35] as our memory-propagating module, the mathematical expression shown in the Eq. (3.1) [36]:

$$\begin{aligned} z_t &= \sigma(W_{hz} * h_{t-1} + W_{xz} * x_t + b_z) \\ r_t &= \sigma(W_{hr} * h_{t-1} + W_{xr} * x_t + b_r) \\ \hat{h}_t &= \Phi(W_h * (r_t \odot h_{t-1}) + W_x * x_t + b) \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t \end{aligned} \tag{1}$$

Here, r_t and z_t are the output of gates. h_t and h_{t-1} is the hidden state and the former of last moment. $\odot$ means Hadamard Product with $*$ standing for convolution. In our case, x_t is the input of our memory module on behalf of the feature map output from the encoding module. The reason why we take advantage of ConvGRU to construct our memory module can be divided into two aspects. On the one hand, ConvGRU can remember the states of historical moments firstly, such as the geometric relationship coming from

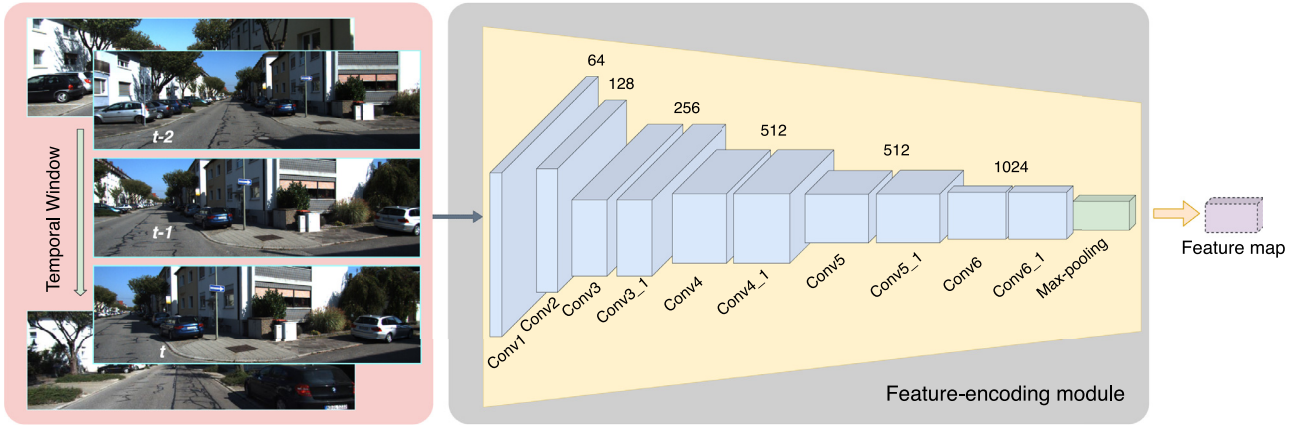

Fig. 2. Feature-encoding module. We map RGB images temporally into this module to get output feature maps for estimating ego-motion further.

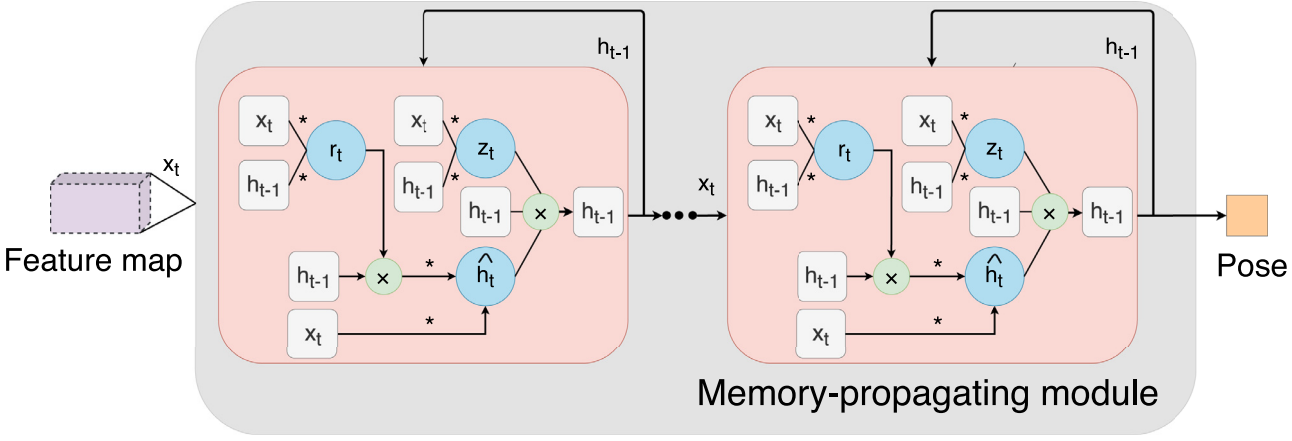

Fig. 3. Memory-propagating module. We pass feature maps obtained from the feature-encoding module into the stacked ConvGRU to propagate the long-term memory from video clips for camera poses regression.

the previous frames of images, and then estimate the pose of the current moment utilizing the geometric constraint within multiple frames; on the other hand, we choose ConvGRU rather than ConvLSTM as our memory module since it is shown that GRU has similar performance to LSTM but with reduced number of gates thus fewer parameters [37]. The image sequence is extracted by our feature-encoding module to obtain multiple $10 \times 3 \times 1024$ tensors propagated into the stacked ConvGRU. In order to further improve the presentation capability and dynamic characteristics of the whole framework, 3 cells of ConvGRU are used in practice. The end output, used for pose regression, will be a 6-dimensional pose vector, which respectively represents the relative poses (Δx, Δy, Δz, $\Delta\psi$, $\Delta\chi$, $\Delta\phi$) between adjacent two images. Finally, we transform obtained pose vectors to SE(3) and calculate the absolute ego-motion.

3.2. Loss function for PoseConvGRU

Visual ego-motion estimation problem can be regarded as a conditional probability problem: Given an image sequence $X = (X_1, X_2, ..., X_{n+1})$, we aim to calculate the appearance probability of the poses $Y = (Y_1, Y_2, ..., Y_n)$ between two adjacent images in this series.

$$p(Y|X) = p(Y_1, Y_2, \ldots, Y_n|X_1, X_2, \ldots, X_{n+1}) \tag{2}$$

The problem needed to be solved here is how to decide the optimal network parameters w^* to maximize the above probability.

$$w^* = \underset{w}{\operatorname{argmax}} p(X|Y; w) \tag{3}$$

So for M sequences, Mean Squared Error (MSE) is used as the error evaluation function, and the loss function that needs to be optimized finally can be obtained as

$$w^* = \underset{w}{\operatorname{argmax}} \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \left\| P_{ij} - \hat{P}_{ij} \right\|_2^2 + \beta \left\| \Phi_{ij} - \hat{\Phi}_{ij} \right\|_2^2 \tag{4}$$

$(\hat{P}_{ij}, \hat{\Phi}_{ij})$ represents the position and orientation ground truth of the image at the jth moment in the ith sequence relative to the image at the next moment in the sequence while β is a scale factor used to maintain the balance between the position error and the orientation error. $\| \cdot \|_2$ represents l2 normalization.

3.3. Mirror-like constraints through data augmentation

We further add some constraints into the network by processing data augmentation along the training step, aiming to perform better in tests.

Data preparation: We take the sequences 00, 01, 02, 08 and 09 in KITTI's experiment as adopted in [28] and sequences 01, 04, 06, 07, 08, 10 and 11 in Malaga's experiment as adopted in [12] respectively for training, which can not satisfy realistic requirements such as high-velocity, velocity-variable, or even reverse driving situations, so it is necessary to augment data for facing challenges above. Our data augmentation is performed on the fly. We augment the training data by randomly skipping frames to simulate the first and second challenge, which results in a better performance. Randomly temporal flipping of the sequences is also performed to alleviate influences caused by the vehicle-reversing challenge.

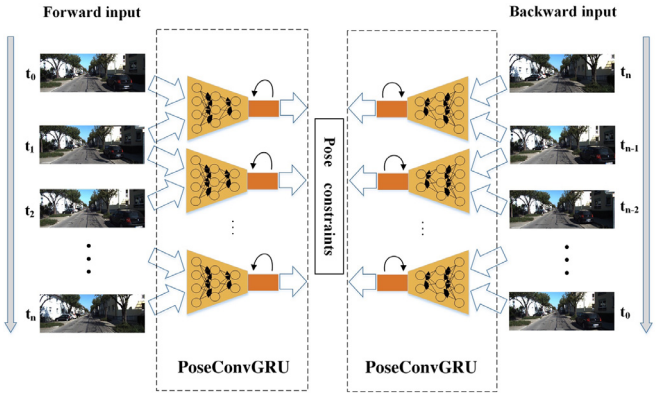

Fig. 4. Constraints in practice. The structure of whole framework presents a mirror-like symmetric construction. All CNNs in this model are weight-shared.

Advanced mirror-like constraints: PoseConvGRU-cons increases the accuracy of camera relative ego-motion estimation by adding some advanced constraints. Fig. 4 shows the framework and specific implementation.

The left half is the same as PoseConvGRU. Image sequences pass through the feature-encoding module and memory-propagating module to obtain outputs which represent the relative poses between two adjacent frames. The right half and the left half are entirely symmetrical, except that the input of image sequences is in the reverse order. The loss function based on MSE of all output poses expresses as

$$Loss = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left\|P_{1ij}-\hat{P}_{1ij}\right\|_2^2+\beta_1\left\|\Phi_{1ij}-\hat{\Phi}_{1ij}\right\|_2^2 + \left\|P_{2ij}-\hat{P}_{2ij}\right\|_2^2+\beta_2\left\|\Phi_{2ij}-\hat{\Phi}_{2ij}\right\|_2^2 \quad (5)$$

$(\hat{P}_{1ij},\ \hat{\Phi}_{1ij})$ represents the position and orientation ground truth of the image at the jth moment of the forward input in the ith sequence relative to the image at the next moment in the sequence while $(\hat{P}_{2ij},\ \hat{\Phi}_{2ij})$ represents the position and orientation ground truth of the image at the jth moment of the backward input in the ith sequence relative to the image at the previous moment in the original sequence. β_1 and β_2 separately represent the scale factors of the position error and the orientation error in the forward sequence input and the backward sequence input. $\|\cdot\|_2$ represents l2-loss we use.

4. Experiment

In this section, we validate our proposed framework on the KITTI Visual Odometry / SLAM Evaluation Dataset [8] and Malaga 2013 Dataset [9]. In the KITTI dataset, only the first 11 sequences have the ground truth data (sequences 00–10), so quantitative experiments can be performed in these 11 scenarios to compare the advantages and disadvantages of the various methods. In terms of the Malaga dataset, there is no extremely precise pose ground truth only raw GPS position nodes without orientational information, so we follow the rule as in [12], using the poses produced from stereo ORB-SLAM(-S) [15] as our regression targets.

We manually split the training set, validation set and test set for each dataset. Then we train models using the training set along with the validation set to settle down hyper-parameters. Finally, we use the test set to evaluate the model.

4.1. Training and evaluating protocols on PoseConvGRU

For the KITTI's experiment, we use the same data splits as Wang et al. used in the paper [24], where sequences 00, 01, 02, 08 and 09 used as the training and validation ("trainval") set; the remaining 6 scenes (sequences 03, 04, 05, 06, 07 and 10) used as the test set. With regard to the Malaga's experiment, we adopt the split way in [12], sequences 01, 04, 06, 07, 08, 10 and 11 used as the "trainval" set with 02, 03 and 09 used as the test set.

Table 2
The components of split dataset.

KITTI		Malaga	
Dataset	Sequence	Dataset	Sequence
Trainval	00 01 02 08 09	Trainval	01 04 06 07 08 10 11
Test	03 04 05 06 07 10	Test	02 03 09

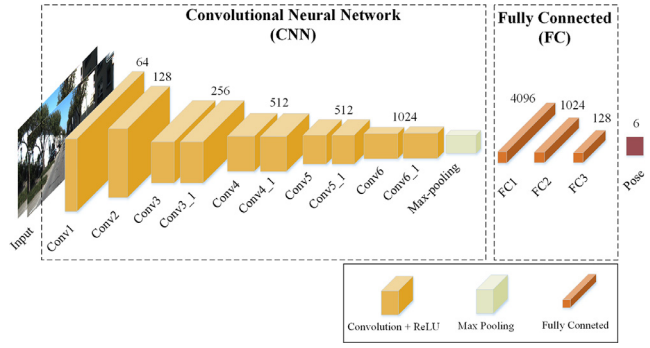

Fig. 5. The structure of Ours-onlyCNN. We leverage the FlowNetSimple structure, adding an extra max-pooling layer to estimate poses directly. Details can be seen in 4.3.

The validation set is randomly selected from the training set, following the principle of sampling without replacement. The specific constituent is shown in Table 2. A key issue here is how to generate sample sequences of images. In the experiment, we randomly select an image frame as the starting frame, and then successively take several frames to form a sequence of length T_1. In order to simplify the data training process, a fixed-length sequence is used, and the starting frames of two adjacent sample sequences are also selected across multiple frames, thereby avoiding excessive data overlap between samples. We can sample sequences every other frame to augment data. The key rule of sampling is to ensure that there are enough identical scenes contained in both images. If the camera moves too fast through some frames, data augmentation must be handled carefully.

Training details: The entire network was built on the deep learning framework PyTorch with the well-known Adam [38] optimizer. The initial learning rate is 10^{-4}. As the number of training epoch increases, the learning rate will be appropriately reduced to ensure that the optimization function is close to the optimal solution. We use two NVIDIA GeForce RTX2080 GPUs for acceleration. The batch size is set to 32. It takes about 0.15 h to train an epoch (all training data is trained for one time). After the end of an epoch, all training samples will be disordered to ensure that the training loss curve will drop smoothly. It takes about 20 hours for one entire experiment to achieve a modest loss error.

In general, training the network combining these two modules is difficult to converge. In order to shorten the convergence time, the feature-extracting module is pre-trained firstly. As shown in our ablation experiments part 4.3, we train CNN and FC layers from scratch to estimate camera poses (Recurrent Neural Network excluded), structure shown in Fig. 5. The Xavier Initialization method is used for setting weights in network parameters during training [39], and the bias is zero-initialized.

After pre-training process, we directly uses the parameters of the trained model in CNN as the initial parameters in the feature-

extracting module of PoseConvGRU and train the whole framework, that is, the fine-tuning operation.

For each batch, we set the initial state of the ConvGRU to 0. The reason can be divided into two sides. On the one hand, the image sequences we use are not selected according to the principle of sampling without replacement. So without shuffling the samples, the adjacent two batches will contain the same imagery frames, if the state here needs to pass through the two batches, we must ensure that the last frame of the first batch is the same as the first frame of the second batch, which inevitably limits the diversity of the network's samples. On the other hand, this sampling method is equivalent to dividing a scene into a number of irrelevant segmented series (including repeated images). So the regularity of each sample can be learned separately. Setting the initial state of ConvGRU to 0 means that each sample was learned from the same state, and experiments prove that it is indeed feasible. There are some related proposed methods learning the initial state, which is set as a network parameter when training Recurrent Neural Network, but this is not necessary for our case.

Additional constraints illustrations: For PoseConvGRU with advanced constraints (PoseConvGRU-cons), it is similar to the original one. The difference is that PoseConvGRU-cons inputs the image sequences to the framework in both front and reverse direction. When T_1 and T_2 are kept constant, the batch size in PoseConvGRU-cons is half of that in the original PoseConvGRU. Other parameters still remain unchanged during training. It takes about 0.5 h to train an epoch. One entire experiment was trained for about 50 hours.

Evaluation metrics: We evaluate our approaches on both dataset using *translation / rotation errors for subsequences* and *translation / rotation errors for different speeds*, which are most commonly used evaluation metrics on the KITTI VO/SLAM dataset. In KITTI's experiment, the subsequences generally take eight kinds of lengths: 100 m, 200 m, ..., 800 m, and as for the counterpart of Malaga, the subsequences take the first three lengths: 100 m, 200 m, 300 m. When calculating the error, we should sample the sequence of the same length from the entire trajectory first, and then calculate the relative poses of the camera on this sequence. Finally, we compare them with the ground truth to derive translation and rotation error respectively, computing the average error of all the sample sequences as the average error of the current sequence. Finally we traverse the subsequences of different lengths to obtain the average error of various subsequences. Another evaluation metric relies on the speed of the mobile platform. We take an average of 7 values in KITTI's experiment and 3 values in Malaga's experiment from the lowest speed to the highest speed, calculating the error between the estimated poses and ground truth at different speeds.

4.2. Runtime analysis

We evaluate the time consumption of our PoseConvGRU framework along with other optical-flow-based methods to stand for our contribution of low calculation cost. We test our model on a common GPU server, which is similar to most of deep learning based methods [10–12]. More specifically, we use one Intel(R) Core(TM) i7-8700 CPU and one NVIDIA GeForce RTX2080 GPU to carry out the network inference, results shown in Table 3.

PoseConvGRU is the fastest method among these learning methods as it needs no preprocess calculation time which other optical-flow-based methods must consume.

4.3. Ablation and comparison experiment

Firstly, we design ablation experiments along with other branches to verify each module's effectiveness and then conduct several comparison experiments with other learning and geometric approaches to reveal and verify our performance.

Table 3
Time-consuming information for each system. *Preprocess Calculation* includes optical flow calculation and RGB conversion, in particular. The best result is highlighted.

	Preprocess Calculation [s/frame]	Network Inference [s/frame]	Total Execution [s/frame]
SVR-VO[40]	15.022	0.112	15.134
P-CNN[10]	15.022	0.041	15.063
Flowdometry[11]	0.271	0.362	0.633
LS-VO[12]	0.012	0.003	0.015
ST-VO[12]	0.012	0.002	0.014
Ours	–	0.011	**0.011**

• The data except ours are derived from [11] and [12].

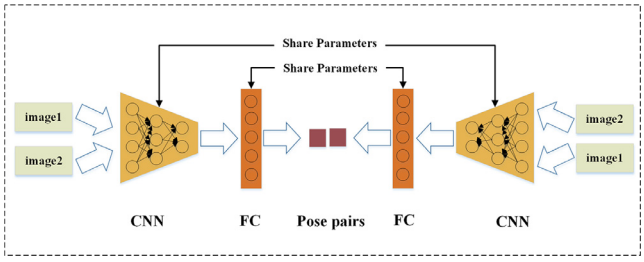

Fig. 6. The structure of Ours-onlyCNN-cons. We do the same constraints as done in 3.3 part to help perform ablation experiment.

We pre-train CNN and some FC layers jointly, adding constraints to estimate visual ego-motion directly (constrained structure performed in Fig. 6), then we remove FC layers remaining CNN along with pre-trained parameters as our initial feature-encoding module in PoseConvGRU. Next, we carry out fine-tuning with stacked ConvGRU to achieve our whole framework. As a result, we derive four branches, named Ours-onlyCNN,[2] Ours-onlyCNN-cons,[3] Ours,[4] Ours-cons[5] from entire training process.

Effectiveness on epoch in pre-training process: Taking Ours-onlyCNN-cons on KITTI's experiment as an example, the whole training process lasts for about 100 epochs, and the models with epoch = 15, 55, 75, 100 were taken out for testing. The trajectory curves of the training set are shown in Fig. 7. The trajectory curves of the test set are shown in Fig. 8, and the curves of the evaluation metrics on the test set are shown in Fig. 9.

Fig. 7 shows that at the initial stage of training, since the translation and rotation error are both substantial and gradually accumulate from the start of sequences, the trajectories of epoch-15 get more and more away from the real ones. As the number of *epoch* increases, the trajectories of training set and the ground truth get closer and when *epoch* reaches to 75, the trajectory curves of training set almost perfectly coincides with ground truth on the sequences 00, 02 and 09, which is also consistent with the continuous decline of training error. When the number of iterations is further increased, it is found that the trajectories of epoch-100 on sequences 02 and 08 deviate far from the counterparts. In fact, the training error at this moment is less than that of the previous ones. The reason for having this phenomenon is that the loss function describes only the MSE of the poses between two adjacent frames. When the poses of some frames diverge from the ground truth, the trajectories still drift far away even if the poses of other frames are estimated accurately. If we continue to train, the training error will be further reduced, and the trajectories will be closer to the

[2] Our framework without memory module and constraints.
[3] Our framework without memory module only.
[4] Our framework, PoseConvGRU itself.
[5] PoseConvGRU with constraints, named PoseConvGRU-cons.

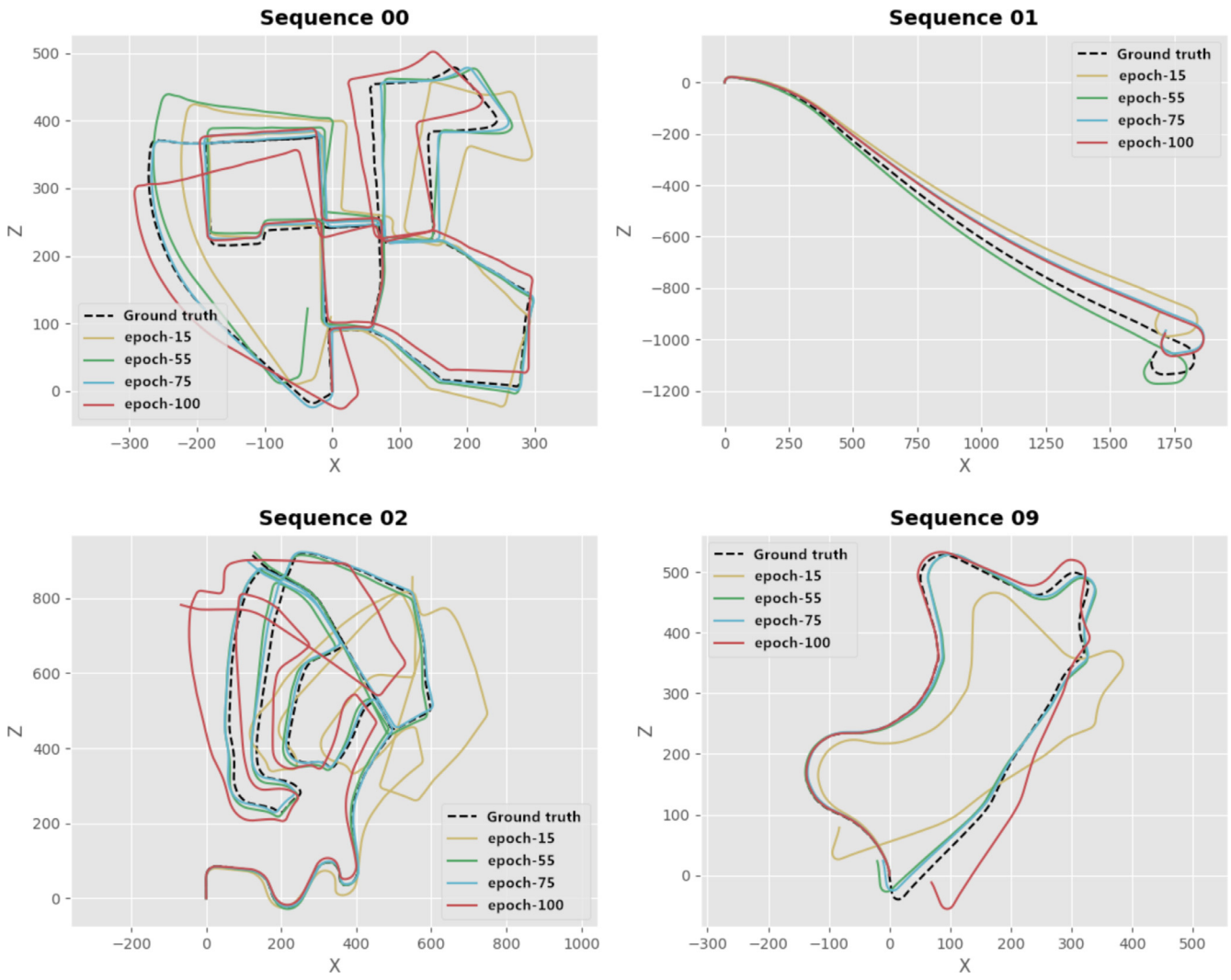

Fig. 7. Trajectories under different *epoch* (training set), represented by different colors. The labels of two axes are in KITTI's style, which sets **forward** and **right** direction of the moving vehicle as **Z** and **X** direction, respectively. The Y-axis represents vertical direction, but it is not supposed to be shown in the figure as data on this direction drifts severely in KITTI. The grid sizes are adjusted to fit each trajectory.

ground truth, but the variation will be more modest. However, due to limited time and the convergence performance for the training set does not represent the generalization ability of the model, it is also necessary to consider the presentation on the test set.

Fig. 8 shows the trajectory curves for different sequences in the test set. It can be clearly seen from sequences 03 and 05 that the estimated trajectories become closer to the ground truth as the number of *epoch* increases, indicating that the framework designed in this paper can learn the motion relationship and feature information between two adjacent frames. Even if many scenes are not seen before, our framework can also estimate camera ego-motion well and do not have the overfitting problem. An important reason is that we uses tricks such as dropout when training the network. For sequence 04, the testing results under different *epoch* are almost the same. It can be seen that the camera basically moves straight forward, and the rotation change is slight, so the translation estimation is very sensitive in terms of this scenario. For sequences 06 and 07, the testing results of the models saved by epoch-55, epoch-75 and epoch-100 basically have no obvious performance improvement, illustrating that the network's learning ability has reached the bottleneck, and even if the number of *epoch* is increased, the performance will not be enhanced significantly. For sequence 10, the estimated trajectories achieve better appearances on epoch-55 and epoch-75 than epoch-100, implying that the model may have the overfitting problem for this scenario. In order to further analyze the performance of the models, it is also necessary to compare with each other on the quantitative evaluation metrics.

The four graphs in Fig. 9 represent the error evaluation metrics on the test set: the upper left and upper right graphs show the trends of translation and rotation error of subsequences under different length respectively. The lower left and lower right graphs show the variations of translation and rotation error at different speeds individually. It can be seen that for subsequences of the same length, as the number of training *epoch* increases, the translation error and the rotation error significantly decline, but there is basically no enormous improvement after epoch-75. For the ego-motion estimation at the same speed, the translation error of the later training (epoch-75) is less obvious than it of the early training (epoch-15), especially when the speed exceeds 40 km/h, the effect of epoch-75 is even worse than that of epoch-55. One big reason is that the speed distribution in the scenes of the training set is different from that of the test set. Therefore, if there is a lack of image pairs with higher speed in the training set, the framework will not accurately estimate the high-speed translation in the test set. To alleviate this problem, we can try to expand the samples

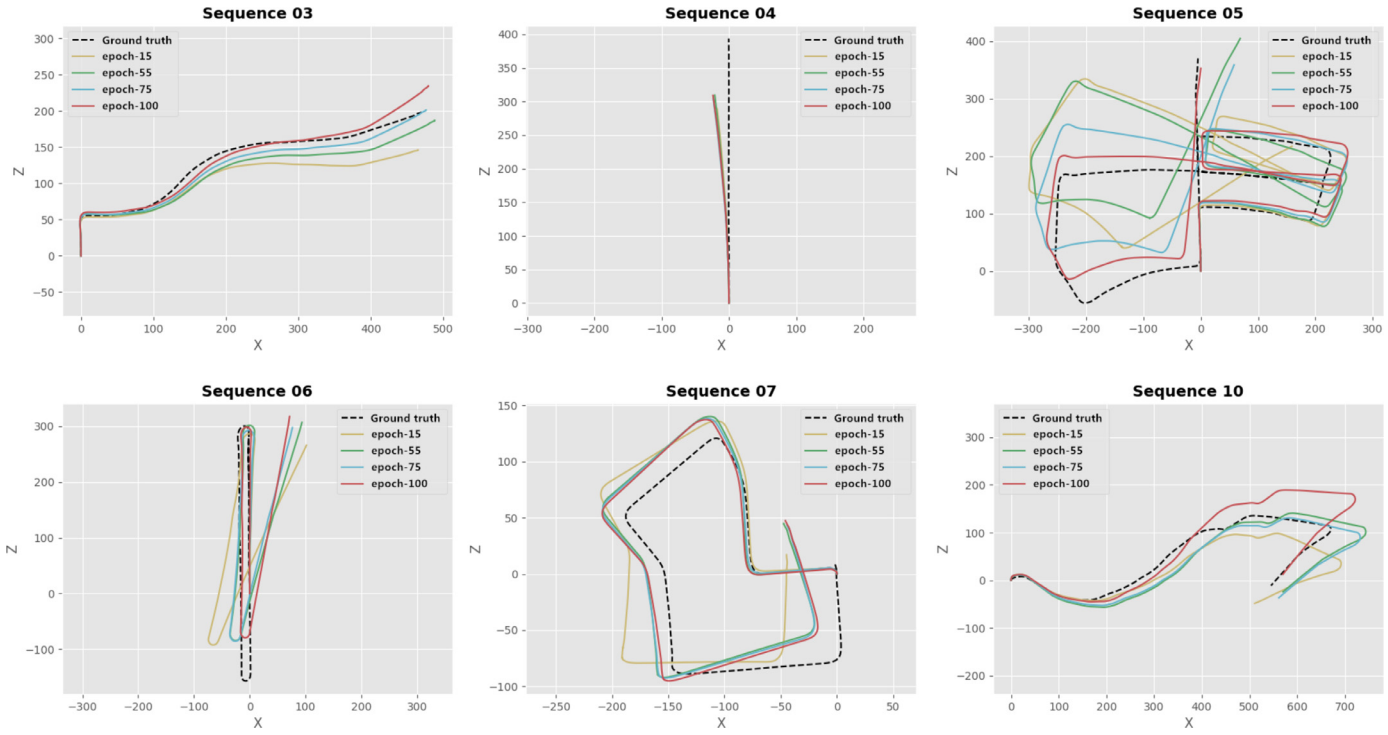

Fig. 8. Trajectories under different *epoch* (test set), represented by different colors. The labels of two axes are in KITTI's style, which sets **forward** and **right** direction of the moving vehicle as **Z** and **X** direction, respectively. The Y-axis represents vertical direction, but it is not supposed to be shown in the figure as data on this direction drifts severely in KITTI. The grid sizes are adjusted to fit each trajectory.

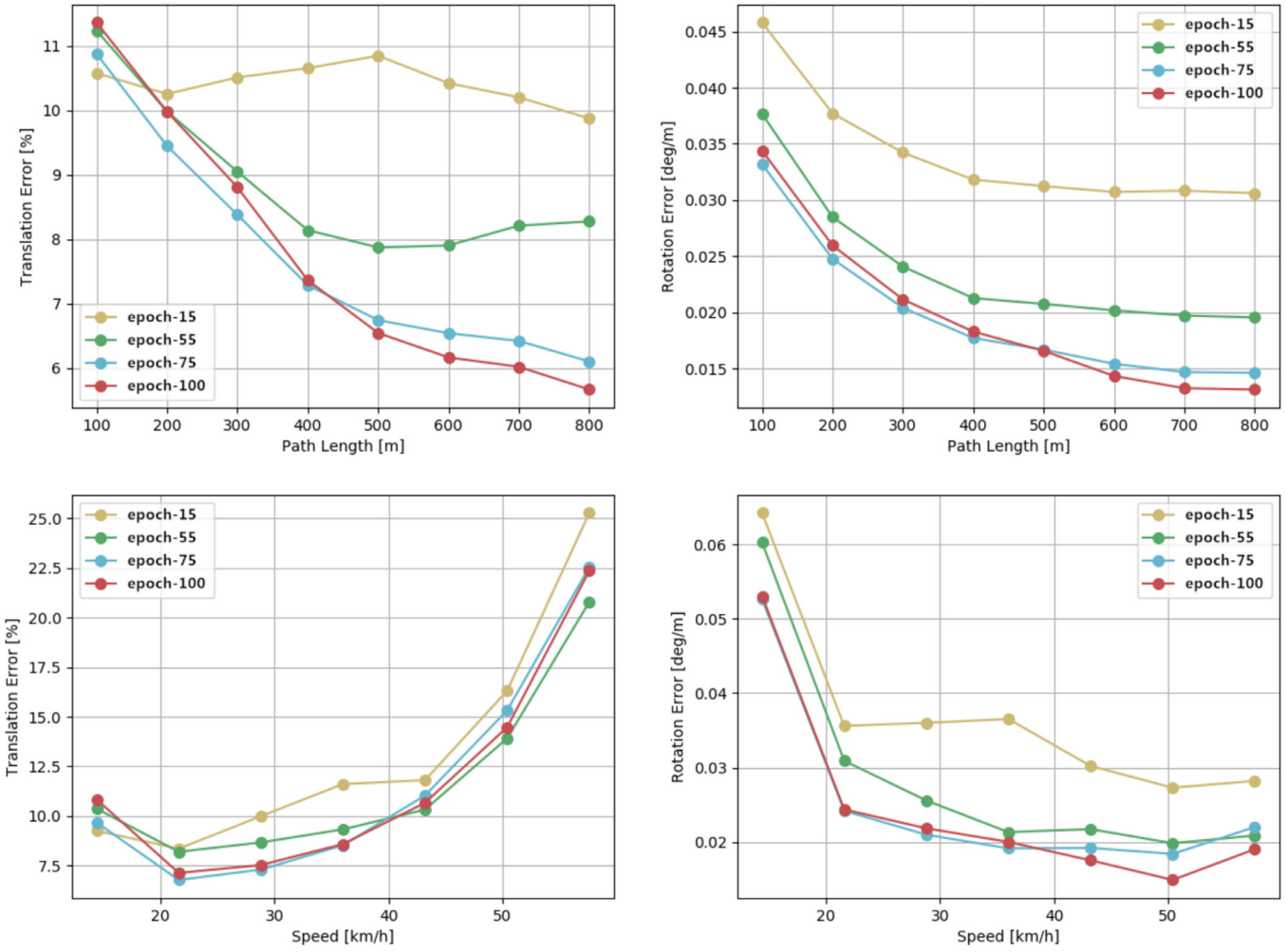

Fig. 9. Comparison results (test set) under different *epoch*.

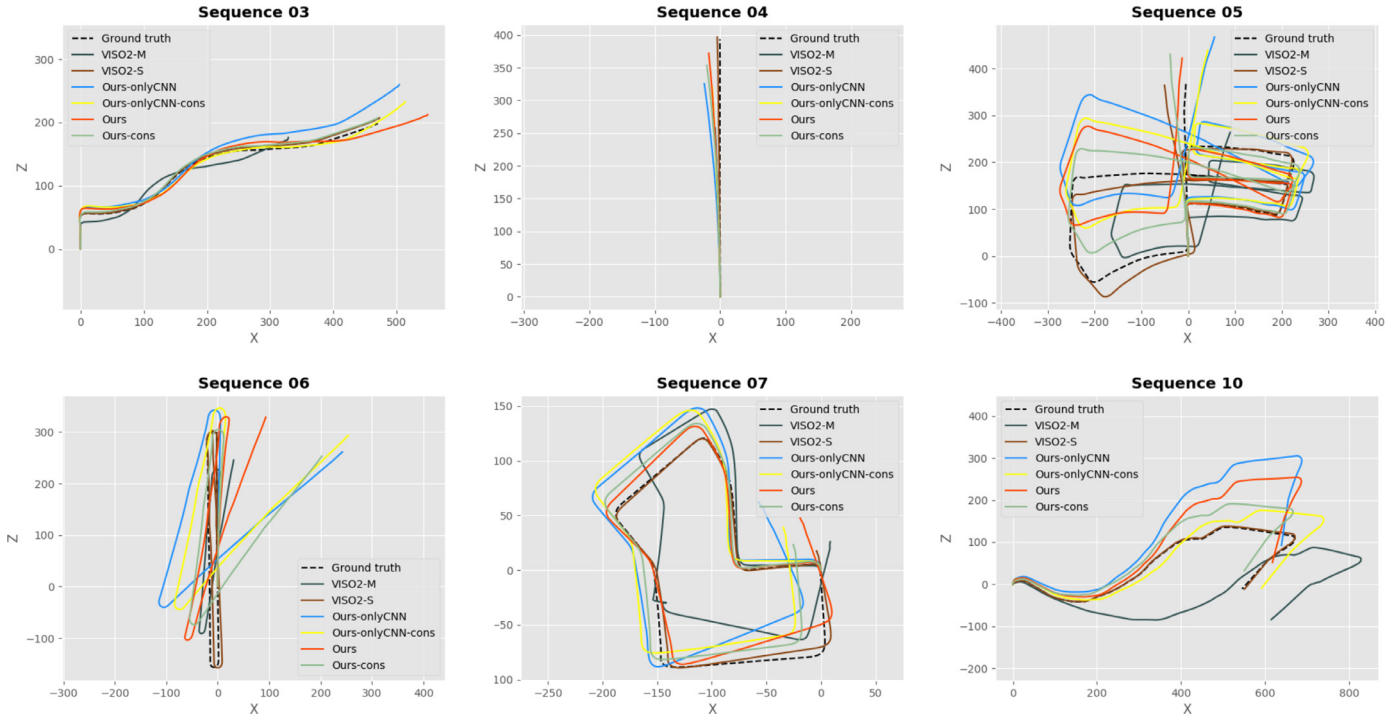

Fig. 10. Trajectories of our methods and LIBVISO2 on the Sequence 03, 04, 05, 06, 07, 10 (test set), exhibited in different colors (best viewed with zoom-in). "Ours" represents PoseConvGRU and "Ours-onlyCNN" means the version without memory module. The remaining two methods are counterparts with constraints. The labels of two axes are in KITTI's style, which sets **forward** and **right** direction of the moving vehicle as **Z** and **X** direction, respectively. It shows that our methods perform better than VISO-M, and methods with RNN-based modules outperform ones without them in many scenes.

(sampling in every few frames) in the training set, thus improving the generalization ability of the model.

Quantitative experiment result: We adopt the same training procedure and hyper-parameters on experiments of both datasets. For PoseConvGRU, we set $\beta = 0.9$ and use network parameters iterated 90 epochs for testing. For PoseConvGRU-cons, we set $\beta_1 = 0.9$, $\beta_2 = 0.999$ and use network parameters iterated 80 epochs for testing.

1) KITTI dataset: The trajectories on the KITTI's test set are shown in Fig. 10, and the evaluation metrics are shown in Fig. 11. The experiment totally compared six methods, including two baseline methods VISO2-M and VISO2-S with Ours-onlyCNN, Ours-onlyCNN-cons, Ours and Ours-cons. As can be seen from Fig. 10, the six methods can slightly estimate the shape of the trajectory accurately, but the details are various from each other.

It is difficult to conclude the advantages and disadvantages of various methods from a certain situation. For example, PoseConvGRU-cons fits the real trajectory better than the onlyCNN on the sequence 03, but is slightly inferior to the onlyCNN-cons on the sequence 05.

Fig. 11 shows the error of the six methods on different evaluation metrics, which can objectively reflect the performance of various methods. When we evaluate translation error and rotation error for subsequences, we have noticed that both errors descend as path length rises. The phenomena are caused by the error metric Eq. (4.3) that we have used in KITTI's definition.

$$E_{trans} = \frac{\left\| R_{pre}^{-1}(t_{gt} - t_{pre}) \right\|_2}{L_{path}}$$
$$E_{rot} = \frac{arccos((trace(R_{pre}^{-1} R_{gt}) - 1)/2)}{L_{path}} \quad (6)$$

The error curves of translation E_{trans} and rotation E_{rot} will descend since the growth rate of subsequences L_{path} is much higher than the absolute translation and rotation error. We discover that the rotation error declines on high velocities, which is different to the translation. This phenomenon may be caused by the fact that the vehicle-recording KITTI dataset tends to go straight on high speeds while rotate when slowing down. Moving straight, which is with few changes on rotation, can be easily learned by the framework in terms of modeling orientation.

PoseConvGRU is basically better than onlyCNN on the evaluation of four kinds of errors, since PoseConvGRU is initialized by the parameters of onlyCNN, and associates ConvGRU for multi-frame constraints. ConvGRU does improve the accuracy of visual ego-motion estimates. PoseConvGRU-cons is superior to PoseConvGRU in these evaluation metrics, indicating that the performance of the network can be improved by adding these advanced constraints. However, the effect of PoseConvGRU-cons is comparable to that of onlyCNN-cons. The main reason we believe is that the sample size is far from enough as both PoseConvGRU-cons and onlyCNN-cons is to double the training data. Table 4 compares PoseConvGRU-cons and PoseConvGRU with the other five methods, including both learning and geometric ones numerically. Our method is slightly superior to CL-VO on *rot* but inferior on *trans*. The training data in our paper is limited, so we can only obtain different samples by combining the existing images. This does not essentially introduce new sample images, so even though we constrain our framework with multi-frame through ConvGRU, it is impossible to improve the generalization ability of the network further.

The performance of the four deep learning methods proposed in this paper is better than the VISO2-M, while they are slightly inferior to the VISO2-S. Our future work consider taking the stereo RGB image into account to achieve better effect.

2) Malaga dataset: This dataset is quite different from KITTI, since it contains more sparse buildings and more extensive sky within a great number of images, making it enough challengeable for evaluating visual ego-motion estimation performance due to ORB-SLAM can not proceed successfully on all sequences. We use

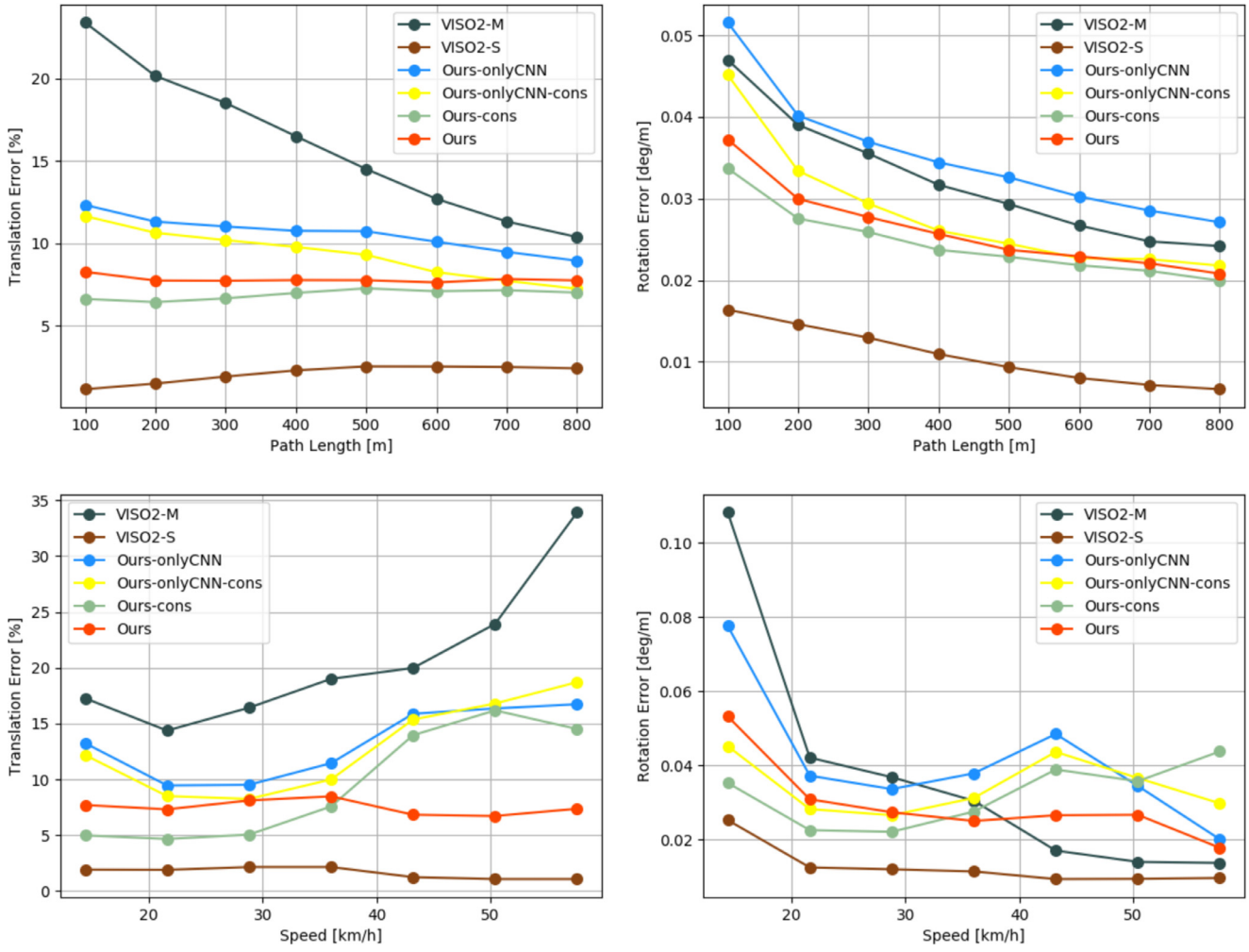

Fig. 11. Comparison results of different methods on the sequences 03, 04, 05, 06, 07, 10 (test set). "Ours" represents PoseConvGRU and "Ours-onlyCNN" means the version without memory module. The remaining two methods are counterparts with constraints.

Table 4
The quantitative results on KITTI dataset. CL-VO and DeepVO are learning methods. The best results are highlighted and the second-best ones are underlined.

Avg	Monocular						Stereo
	PoseConvGRU -cons (Ours)	PoseConvGRU (Ours)	CL-VO [41]	DeepVO [28]	ORB-SLAM [15]	VISO2-M [14]	VISO2-S [14]
rot(°)	**0.0247**	0.0268	0.0267	0.0351	0.3553	0.0386	0.0044
trans(%)	7.56	7.69	**7.37**	10.17	30.01	16.24	2.92

• *rot*: average frame-to-frame relative rotational drift(°/m) on 100 m–800 m. • *trans*: average frame-to-frame relative translational drift(%) on 100 m–800 m. • The data of DeepVO and ORB-SLAM are derived from [41].

the split way of training and test adopted in [12]. Table 5 compares PoseConvGRU-cons with monocular learning methods LS-VO, ST-VO, and monocular geometric method ORB-SLAM(-M). Our method is superior to LS-VO on *trans* but inferior to ORB-SLAM-M on *rot*.

Qualitative experiment result:

1) KITTI dataset: Quantitative experiments only use the first 11 sequences with ground truth to carry out training and testing in the KITTI's experiment, which can quantitatively analyze the error. Considering that the latter 11 scenes do not provide ground truth, we can still use the trained model to test in these scenes and visually evaluate the performance of our method. In this paper, all the images of sequences 00–11 are used as the training set, with a total of 23,190 image pairs, taking a tiny part as the validation set, and then we adopt the method in the quantitative ex-

Table 5
The quantitative results on Malaga dataset. LS-VO and ST-VO predict optical flow to achieve performance, while we use RGB images directly. The best results are highlighted and the second-best ones are underlined.

Avg	Methods			
	PoseConvGRU -cons (Ours)	LS-VO [12]	ST-VO [12]	ORB-SLAM-M [15]
rot(°)	0.0316	0.0690	0.1241	**0.0156**
trans(%)	**12.70**	15.56	23.20	86.60

• *rot*: weighted average frame-to-frame relative rotational drift(°/m) on 100 m - 300 m. • *trans*: weighted average frame-to-frame relative translational drift(%) on 100 m - 300 m. • The data of ORB-SLAM-M are derived from [12].

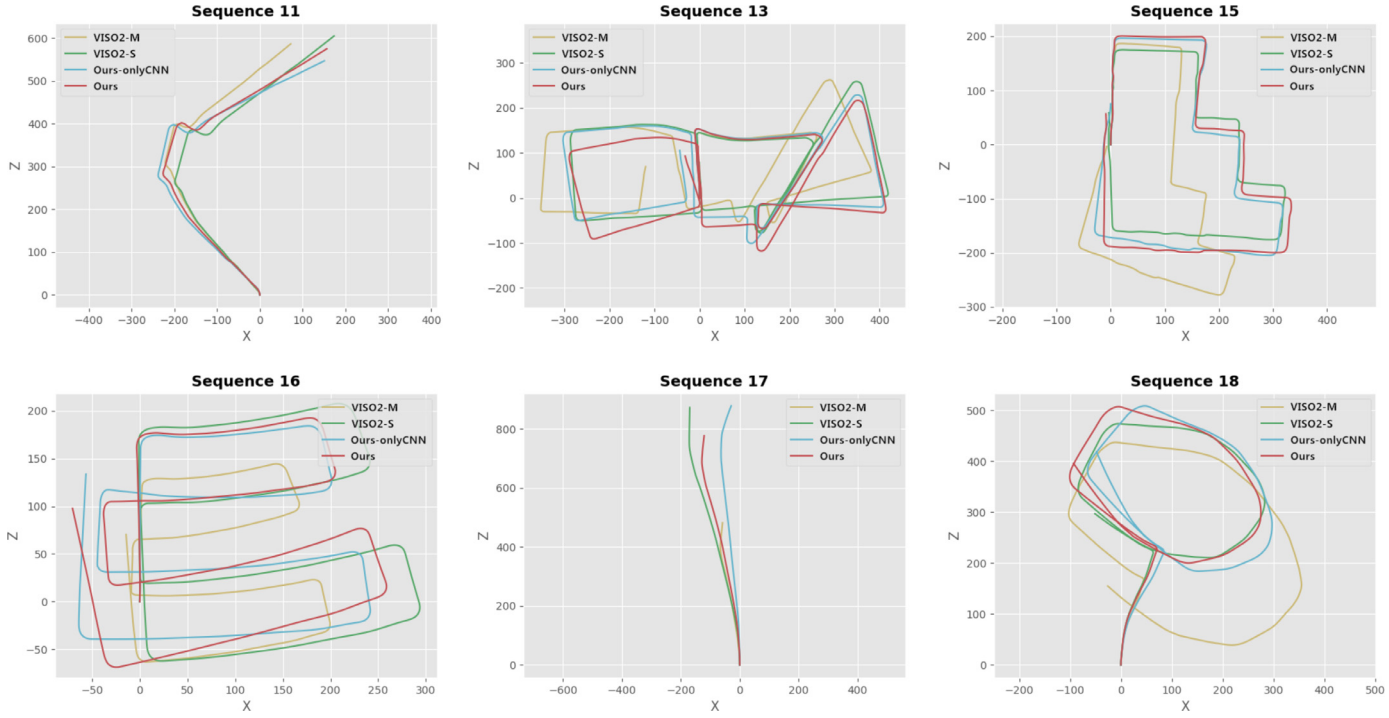

Fig. 12. These six sequences do not have ground truth, so we proceed on the qualitative experiments. "Ours" represents PoseConvGRU and "Ours-onlyCNN" means the version without memory module. The remaining two methods are counterparts with constraints. The labels of two axes are in KITTI's style, which sets **forward** and **right** direction of the moving vehicle as **Z** and **X** direction, respectively. For the best view, we only show methods without constraints.

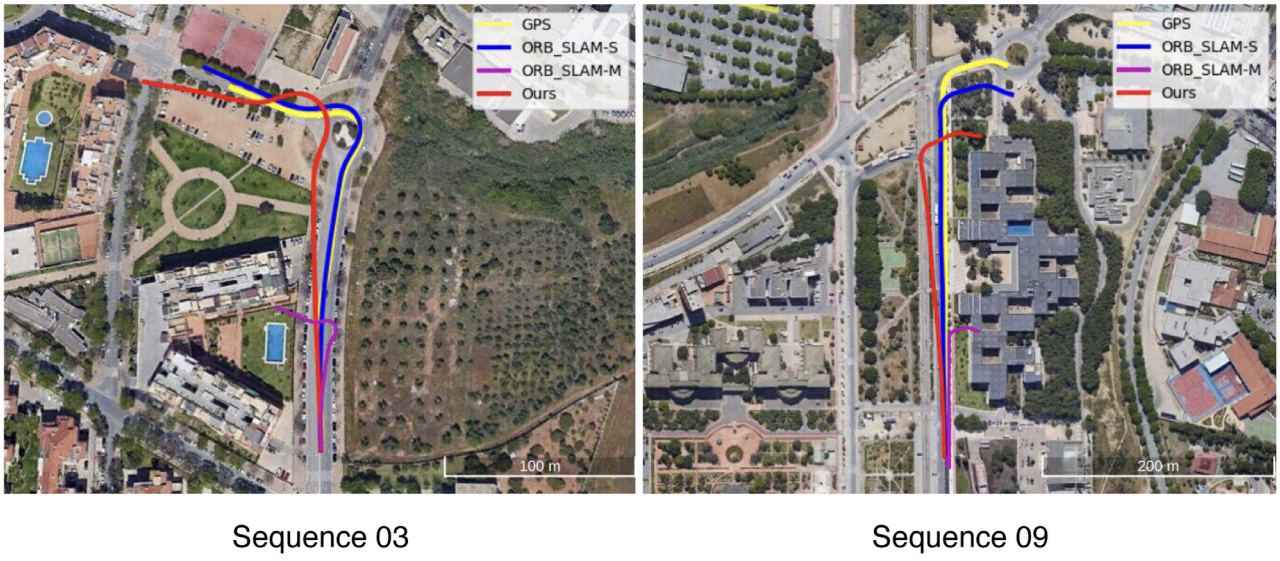

Fig. 13. Qualitative tests on Malaga dataset exhibited on Google Earth. The blue and purple trajectories express stereo and monocular ORB-SLAM respectively, with the yellow one represents GPS(ground truth). Our performance(PoseConvGRU-cons) shows in red (best viewed with zoom-in).

periment to train it. When the model converges to a certain extent, the training set and the validation set are combined for fine-tuning until the training error is basically not reduced. The model at this time is taken as the final model for the qualitative experiment and tested on the rest sequences 11–21. For avoiding messy curves intertwined and giving a better viewing experience, we only show the performance of PoseConvGRU and onlyCNN.

The partial results are shown in Fig. 12, which also include the testing results of two baseline methods VISO2-M and VISO2-S. The range of these 11 scenes is from 100 m $\times$ 100 m (such as sequence 14) to 2000 m $\times$ 5000 m (such as sequence 21), and is quite different with the training set. From the perspective of vehicle speed, the top speed of the first 11 scenes is basically around 60 km/h, while in the latter 11 scenes, the running speed is even as high as 90 km/h. When the camera frame rate is fixed at 10 fps, the movement between two adjacent images will become intense, and the lack of such samples in the training set will affect the generalization ability of the deep learning algorithm.

Because of the lack of real trajectory data, we use the results of the VISO2-S method as a reference. In sequences 11, 13, 15, 16, 17 and 18, the performance of PoseConvGRU is better than those of onlyCNN and VISO2-M, and the trajectories of onlyCNN are also more accurate than VISO2-M. In most scenarios, the trajectories of VISO2-M have severe deviations in scale compared to the other

three methods. This shows that the scale problem of monocular geometric methods is a big obstacle in the field of pose estimation, and deep learning methods can basically overcome this difficulty.

2) Malaga dataset: Since the Malaga dataset does not have ground truth, quantitative evaluation can only be conducted with poses calculated by ORB-SLAM-S reckoned as regression targets. However, Frequent GPS data is available, we still can perform qualitative comparison. Fig. 13 depicts the testing results on Malaga dataset sequences 03 and 09 (ORB-SLAM-M failed on 02), superimposed on Google Earth. As we can see from Fig. 13, PoseConvGRU-cons(Ours) predictions are close to GPS and ORB-SLAM-S in those three sequences. It is significantly better than ORB-SLAM-M because our monocular approach is scale-aware, although it suffers from drift. This experiment further shows that our approach can proceed with some scenarios successfully which classical methods (like ORB-SLAM-M) can not.

5. Conclusion

We propose a novel data-driven Long-term Recurrent Convolutional Neural Networks (PoseConvGRU) encoding geometrical features in images to gauge camera poses, which is completely end-to-end. Our proposed neural network is more real-time and less calculation-consuming, unlike other learning-based ego-motion estimation algorithms, which need to spend plenty of time to calculate the pre-processed dense optical flow before training the neural network or use a pre-trained network to estimate the optical flow with additional calculation costs. The main idea is to use CNN to extract the geometric relationship features of two adjacent images in the image sequences along with data augmentation, then pass the feature maps through a stacked ConvGRU module for feature learning, and finally achieve the regression of the relative poses among consecutive multi-frame images. The performance of our approach is better than VISO2-M and ORB-SLAM-M, showing a competitive performance to state-of-the-art learning methods. In the future, we plan to focus on the stereo study of end-to-end visual ego-motion estimation, since some significant information like scale can be directly obtained from stereo images.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work is supported by the National Natural Science Foundation of China under Grant U1509210, 61836015.

References

[1] D. Nistér, O. Naroditsky, J. Bergen, Visual odometry, in: Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on, 1, IEEE, 2004. I–I
[2] Y.-S. Chen, L.-G. Liou, Y.-P. Hung, C.-S. Fuh, Three-dimensional ego-motion estimation from motion fields observed with multiple cameras, Pattern Recognit. 34 (8) (2001) 1573–1583.
[3] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, et al., Recent advances in convolutional neural networks, Pattern Recognit. 77 (2018) 354–377.
[4] G. Yao, T. Lei, J. Zhong, A review of convolutional-neural-network-based action recognition, Pattern Recognit Lett 118 (2019) 14–22.
[5] S.M. Erfani, S. Rajasegarar, S. Karunasekera, C. Leckie, High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning, Pattern Recognit. 58 (2016) 121–134.
[6] H. Yang, C. Yuan, B. Li, Y. Du, J. Xing, W. Hu, S.J. Maybank, Asymmetric 3d convolutional neural networks for action recognition, Pattern Recognit. 85 (2019) 1–12.
[7] X. Bu, Y. Wu, Z. Gao, Y. Jia, Deep convolutional network with locality and sparsity constraints for texture classification, Pattern Recognit. 91 (2019) 34–46.
[8] A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? The kitti vision benchmark suite, in: Conference on Computer Vision and Pattern Recognition (CVPR), 2012.
[9] J.-L. Blanco-Claraco, F.-Á. Moreno-Dueñas, J. González-Jiménez, The málaga urban dataset: high-rate stereo and lidar in a realistic urban scenario, Int. J. Rob. Res. 33 (2) (2014) 207–214.
[10] G. Costante, M. Mancini, P. Valigi, T.A. Ciarfuglia, Exploring representation learning with cnns for frame-to-frame ego-motion estimation, IEEE Rob. Autom. Lett. 1 (1) (2015) 18–25.
[11] 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), IEEE, 2017, pp. 624–631.
[12] G. Costante, T.A. Ciarfuglia, Ls-vo: learning dense optical subspace for robust visual odometry estimation, IEEE Rob. Autom. Lett. 3 (3) (2018) 1735–1742.
[13] H.P. Moravec, Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover., Technical Report, Stanford Univ CA Dept of Computer Science, 1980.
[14] A. Geiger, J. Ziegler, C. Stiller, Stereoscan: Dense 3d reconstruction in real-time, in: Intelligent Vehicles Symposium (IV), 2011 IEEE, Ieee, 2011, pp. 963–968.
[15] R. Mur-Artal, J.M.M. Montiel, J.D. Tardos, Orb-slam: a versatile and accurate monocular slam system, IEEE Trans. Rob. 31 (5) (2015) 1147–1163.
[16] C. Forster, M. Pizzoli, D. Scaramuzza, Svo: Fast semi-direct monocular visual odometry, in: Robotics and Automation (ICRA), 2014 IEEE International Conference on, IEEE, 2014, pp. 15–22.
[17] J. Engel, T. Schöps, D. Cremers, Lsd-slam: Large-scale direct monocular slam, in: European Conference on Computer Vision, Springer, 2014, pp. 834–849.
[18] J. Engel, V. Koltun, D. Cremers, Direct sparse odometry, IEEE Trans. Pattern Anal. Mach. Intell. 40 (3) (2017) 611–625.
[19] R. Roberts, H. Nguyen, N. Krishnamurthi, T. Balch, Memory-based learning for visual odometry, in: Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, IEEE, 2008, pp. 47–52.
[20] K.R. Konda, R. Memisevic, Learning visual odometry with a convolutional network., in: VISAPP (1), 2015, pp. 486–490.
[21] A. Kendall, M. Grimes, R. Cipolla, Posenet: a convolutional network for real-time 6-dof camera relocalization, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 2938–2946.
[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.
[23] A. Kendall, R. Cipolla, Modelling uncertainty in deep learning for camera relocalization, in: Robotics and Automation (ICRA), 2016 IEEE International Conference on, IEEE, 2016, pp. 4762–4769.
[24] V. Mohanty, S. Agrawal, S. Datta, A. Ghosh, V.D. Sharma, D. Chakravarty, Deepvo: a deep learning approach for monocular visual odometry. arXiv:1611.06069.
[25] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.
[26] R. Clark, S. Wang, H. Wen, A. Markham, N. Trigoni, Vinet: visual-inertial odometry as a sequence-to-sequence learning problem., in: AAAI, 2017, pp. 3995–4001.
[27] R. Clark, S. Wang, A. Markham, N. Trigoni, H. Wen, Vidloc: 6-dof video-clip relocalization. arXiv:1702.06521.
[28] S. Wang, R. Clark, H. Wen, N. Trigoni, Deepvo: towards end-to-end visual odometry with deep recurrent convolutional neural networks, in: Robotics and Automation (ICRA), 2017 IEEE International Conference on, IEEE, 2017, pp. 2043–2050.
[29] S. Wang, R. Clark, H. Wen, N. Trigoni, End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks, Int. J. Rob. Res. 37 (4–5) (2018) 513–542.
[30] G. Iyer, J. Krishna Murthy, G. Gupta, M. Krishna, L. Paull, Geometric consistency for self-supervised end-to-end visual odometry, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018, pp. 267–275.
[31] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556.
[32] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, K. Keutzer, Densenet: implementing efficient convnet descriptor pyramids. arXiv:1404.1869.
[33] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, T. Brox, Flownet: learning optical flow with convolutional networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 2758–2766.
[34] G. Koch, Siamese neural networks for one-shot image recognition, 2015.
[35] N. Ballas, L. Yao, C. Pal, A. Courville, Delving deeper into convolutional networks for learning video representations. arXiv:1511.06432.
[36] M. Siam, S. Valipour, M. Jagersand, N. Ray, Convolutional gated recurrent networks for video segmentation, in: 2017 IEEE International Conference on Image Processing (ICIP), IEEE, 2017, pp. 3090–3094.
[37] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv:1412.3555.
[38] D. Kingma, J. Ba, Adam: a method for stochastic optimization. arXiv:1412.6980.
[39] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.

[40] T.A. Ciarfuglia, G. Costante, P. Valigi, E. Ricci, Evaluation of non-geometric methods for visual odometry, Rob. Auton. Syst. 62 (12) (2014) 1717–1730.
[41] M. Saputra, P. Gusmão, S. Wang, A. Markham, N. Trigoni, Learning monocular visual odometry through geometry-aware curriculum learning, in: 2019 International Conference on Robotics and Automation (ICRA), IEEE, 2019, pp. 3549–3555.

Guangyao Zhai received his B.Eng. degree in Automation school from Northwestern Polytechnical University in 2018. He is currently a M.D. candidate of the institute of Cyber Systems and Control, Department of Control Science and Engineering, Zhejiang University. His latest research interests include robotics perception, SLAM and geometric learning.

Liang Liu received his B.Eng. degree in communication engineering from Zhejiang University of Technology in 2016. He is currently a Ph.D. candidate of the institute of Cyber Systems and Control, Department of Control Science and Engineering, Zhejiang University. His latest research interests include computer vision, action recognition and geometric learning.

Linjian Zhang received his B.Eng. in Automation from Zhejiang University in 2015. He is currently a researcher in Fuxi AI Lab, Netease. His latest research interests include computer vision and Natural Language Processing.

Yong Liu received his B.S. degree in computer science and engineering from Zhejiang University in 2001, and the Ph.D. degree in computer science from Zhejiang University in 2007. He is currently a professor in the institute of Cyber Systems and Control, Department of Control Science and Engineering, Zhejiang University. He has published more than 30 research papers in machine learning, computer vision, information fusion, robotics. His latest research interests include machine learning, robotics vision, information processing and granular computing. He is the corresponding author of this paper.

Yunliang Jiang received the Ph.D. degree in Computer Science and Technology from Zhejiang University in 2006. He is a Professor in School of Information Engineering, Hu zhou University. His research interests include Intelligent Information Processing and Geographic Information Systems.