EasyChair Preprint

№ 413

Visual Odometry Based on Convolutional Neural Networks for Large-Scale Scenes

Xuyang Meng, Chunxiao Fan and Yue Ming

August 9, 2018

Visual Odometry Based on Convolutional Neural Networks for Large-Scale Scenes

[1]Xuyang Meng, [1]Chunxiao Fan and [1]Yue Ming
[1]Beijing University of Posts and Telecommunications, Beijing, China, 100876

ABSTRACT

The main task of visual odometry (VO) is to measure camera motion and image depth, which is the basis of 3D reconstruction and the front-end of simultaneous localization and mapping (SLAM). However, most of the existing methods have low accuracy or require advanced sensors. In order to predict camera pose and image depth at the same time with high accuracy from image sequences captured by regular camera, we train a novel framework, named PD-Net, and it is based on a convolutional neural network (CNN). There are two main modules: one is pose estimator which is able to estimate the 6-DoF camera pose, the other is depth estimator computing the depth of its view. The keys of our proposed framework are that PD-Net comprises some shared convolutional layers and is divided into two branches to estimate camera motion and image depth, respectively. Experiments on KITTI and TUM datasets show that our proposed method can extract meaningful depth estimation and successfully estimate frame-to-frame camera rotations and translations in large scenes even texture-less. It outperforms previous methods in terms of accuracy and robustness.

Keywords: VO, 3D reconstruction, SLAM, CNN, pose estimation, depth prediction.

1. INTRODUCTION

With the rapid development of virtual reality (VR) and augmented reality (AR) technology, 3D reconstruction and simultaneous localization and mapping (SLAM) have attracted more and more attention. As the front-end of SLAM, visual odometry (VO) has been extensively studied. VO derives the relative motion of the camera and the depth of the scene through a series of continuous visual information acquired by a sensor mounted on a mobile robot or an automated guided vehicle (AGV) [1]. Compared with the traditional wheel odometry, VO has no error caused by wheel slip, and is suitable for non-planar motion environments and non-wheeled mobile robots [2]. More than, VO can capture environment information for 3D reconstruction, location, navigation, target tracking, etc. and it has broad application prospects.

In order to improve precision and reduce processing time, visual odometry detects sparse features firstly and then perform inter-frame estimation by matching these interesting points and descriptors, such as scale invariant feature transform-based (SIFT-based) SLAM [3], and oriented FAST and rotated BRIFE -based (ORB-based) SLAM [4]. SIFT and ORB are popular among visual SLAM because of their robust and effectiveness. But these man-made features have many limitations. On the one hand, designing some features to optimally represent image information is still a problem to be solved in computer vision [5]. On the other hand, there are many challenges in texture-less, motion blur, light-changing, dynamic objects, and so on [5]. Therefore, we adopt layering features based on deep learning to perform visual odometry. Our main contribution is to train a convolution neural network (CNN), named PD-Net, to jointly predict camera pose and images depth from some successive images with some shared and unassisted layers. The input to our pipeline is several consecutive image sequences. The network estimates the depth of each image and the camera pose between adjacent images simultaneously. This method is different from traditional structure from motion (SfM) pipeline which computes the structure and motion from two images. The illustration of our proposed framework is shown in Fig.1.

The rest of this paper is organized as follows. Section 2 summaries related work about classical methods calculating camera motion and image depth. Section 3 describes the architecture of our proposed framework. Section 4 presents our experimental evaluations and results. Eventually, conclusions are drawn in Section 5.

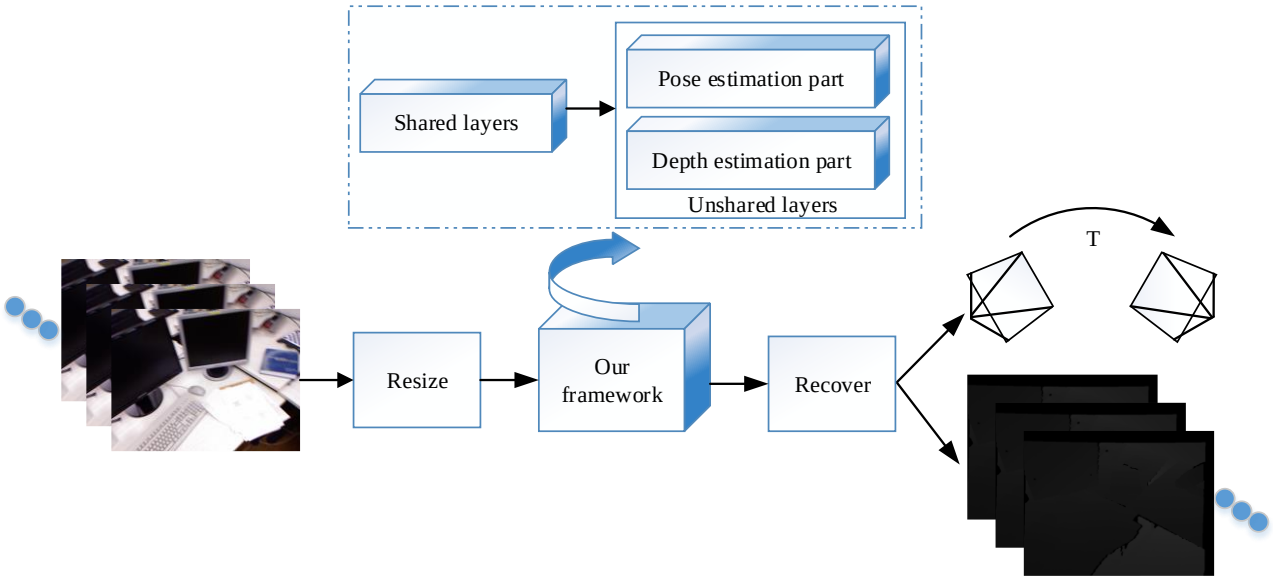

Figure 1. Illustration of our proposed framework. The input to our framework is some image sequences and outputs are camera motion and image depth. Images are resized to fit our framework and recovered before output.

2. RELATED WORK

Related work is elaborated in three aspects: visual odometry, pose estimation based on CNN and depth estimation based on CNN.

2.1 Visual Odometry

Visual odometry has been popular with large-scale scenes in the past decades. VO calculates correlation transformation from image I_k to I_{k-1}, and then integrates all transformations to recover camera's motion trajectory. As we all know, camera pose forms a rigid transformation $\boldsymbol{T}_{k,k-1} \in \boldsymbol{R}^{4\times4}$ at time k and k-1, and

$$\boldsymbol{T}_{k,k-1} = \begin{bmatrix} \boldsymbol{R}_{k,k-1} & \boldsymbol{t}_{k,k-1} \\ \boldsymbol{0}^{\boldsymbol{T}} & 1 \end{bmatrix} \tag{1}$$

where $\boldsymbol{R}_{k,k-1}$ is a rotation matric between images I_k and I_{k-1}, $\boldsymbol{t}_{k,k-1}$ is a translation matric.

We can classify visual odometry into features-based method [4, 6, 7], direct method [8, 9, 10], and semi-direct method [12, 13] from implementations. Features-based VO has long been considered the mainstream method. Nister et al [6] firstly carried out the work related to real-time monocular large scene VO. But his work resulted in much error matching due to noise and outliers. Klein et al [7] have proposed a framework named PTAM, which was an early successful VO based on features. Although its performance was not perfect, they provided a complete generic framework consisting of front-end and back-end for tracking and mapping, respectively. Besides, ORB-SLAM2 [13] was also based on PTAM, which was the first pipeline introducing nonlinear optimization. However, ORB-SLAM2 was suitable for small scenes and lacked global relocation leading to poor practicability. Apart from these, features-based method cannot be applied in texture-less scenes and it often increases algorithm complexity when we want to enhance robustness. Therefore, direct method based on the assumption of pixel gray invariance has developed rapidly in recent years. Engel et al [9] proposed a large-scale direct real-time location and mapping algorithm, LSD-SLAM, in order to construct a semi-dense 3D environment map. It was capable of building a large-scale 3D map while estimating the high-precision camera pose. Direct sparse odometry (DSO) [10] was also put forward by Engle et al. It far exceeded previous methods likely ORB-SLAM and LSD-SLAM in terms of robustness, accuracy, and effectiveness. But direct method is inapplicable when camera has very little movement, which relying entirely on gradient search. To further ensure optimality and consistency, Forster et al [11] employed semi-direct odometry (SVO). SVO depended on feature consistency and obtained the camera pose through direct method. It avoided feature matching and outliers processing, greatly shortening the calculation time. Later, Forster et al [12] proved that SVO could be extended to multi-camera systems, which tracking edges, and also supporting multiple cameras.

2.2 Pose Estimation Based on CNN

Camera pose estimation is the process of determining camera motion by analyzing the multi-view geometry between these associated pictures. Compared with traditional inter-frame prediction based on features or direct methods, deep learning-based methods don't require feature extraction and matching, even not more complex geometric operations. Konda et al [14] proposed an end-to-end deep neural network to predict camera translation and orientation. It extracted visual motion and inter-frame information using a multiplicative interaction framework, which was called unsupervised synchrony auto-encoder. The algorithm was faster than traditional methods, but it couldn't achieve the accuracy of mainstreams. Costante et al [15] applied the Brox algorithm to extract the dense optical flow characteristics used as input of the CNN network to learn the optimal feature representation. It was robust in terms of motion blur and illumination changes. However, the proposed algorithm relied on training data and produced large errors when the inter-frame speed of the image was too fast. Handa et al [16] designed an end-to-end network by extending spatial transform network to regress the classical computer vision method. They constructed geometric vision with neural network (GVNN) including global transformation and pixel transformation kernel M estimator to estimate camera pose based on RGB-D data. This approach ensured that the loss function was as close as to 0 when converging, and the missing pixels could be properly processed. Li et al [17] put forward a VGG-based CNN framework UnDeepVO, which used spatial and temporal geometric constraint relationships. They trained stereo image pairs and tested continuous monocular sequences in an unsupervised manner to estimation camera's 6-DoF pose. In a word, Pose estimation based on deep learning is more intuitive and concise.

2.3 Depth Estimation Based on CNN

Accurate depth information occupies an important role in 3D reconstruction. These classic SfM methods usually take stereo matching algorithm and triangulation principle to calculate the disparity and measure the scene depth. However its scope and accuracy have certain limitation. Depth estimation based on CNN is popular with researchers along with the development of deep learning. Tateno et al [18] put forward a propagation network, CNN-SLAM, extending ResNet-50 to a full convolution network. They minimized the soft-max layer and entropy loss function using back-propagation and stochastic gradient descent (SGD). Their method combined the pose predicted by CNN with the measurements obtained in SLAM to calculate camera motion. Godard et al [19] carried out a CNN network named monodepth using parasitic geometric constraints to generate disparity map. Their network learned to perform a single image depth estimation with good performance and robustness. Benjamin et al [20] proposed the DeMoN pipeline, which introduced multiple stacked encoder-decoder networks to predict image depth, surface normal, and optical flow simultaneously. Its core part was the iterative network aiming to improve forecasting capabilities. Sudheendra et al [21] set up a geometric neural network SfM-Net for video motion and depth estimation. They decomposed frame-to-frame pixel motion and object depth to predict rigid objects motion and depth. Their method performed comparably with classical SLAM and previous CNN-based methods.

3. OUR PROPOSED FRAMEWORK

In this section, we first give an overview of our proposed framework, then we illustrate the pose estimation module and depth estimation module in details.

3.1 Overview

An overview of our proposed framework is shown in Fig. 2. Our framework takes some consequent monocular images as input, and regresses both scaled 6-DoF camera pose and image depth, simultaneously. The proposed network consists of several shared convolutional layers and splits into two modules predicting camera motion and image depth, respectively. The kernel size is 3 for all conv layers except for the first two conv where the kernel size are 7 and 5, respectively. The stride of each conv layer is 2 and all conv layers are followed by ReLU apart from the last layer where no nonlinear activation is applied. We have also applied max pooling layer after each conv layer to prevent over fitting.

3.2 Pose Estimation Module

Pose estimation module is a CNN-based architecture to calculate the 6-DoF transformation between two images. The inputs to the module are two image sequences (along the color channels), and output is the relative pose between adjacent images. There are two groups of fully connected layers following the last pooling layer to estimate 3D translation and 3D orientation for each motion. The 3D translation t is expressed in Cartesian coordinates. We normalize the translation to resolve the scale ambiguity so that $\| t \|=1$. The rotation metric is redundant that describes the orientation with 9 quantities

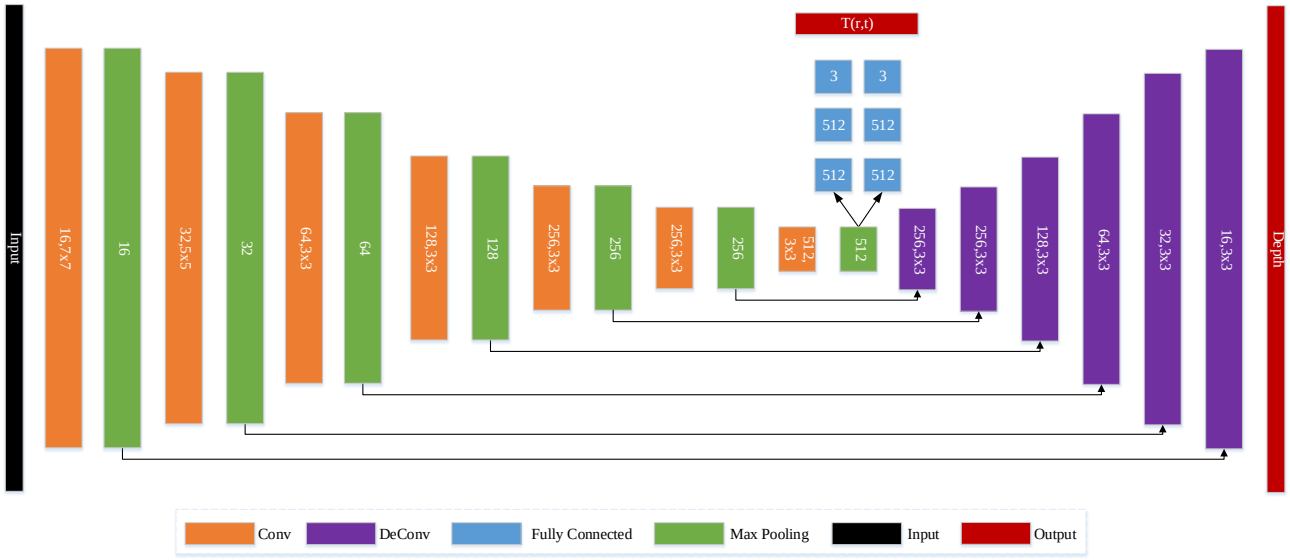

Figure 2. Architecture of our proposed framework. Our framework has some shared layers and splits into two parts to estimate camera motion and image depth, respectively. Details are described in Section 3.1.

and Euler angle has a notable Gimbal Lock problem. Thus, we use an angle axis representation, $\boldsymbol{r} = \theta \boldsymbol{v}$, with angle θ and axis $\boldsymbol{v}$. We can convert Axis-Angle to rotation matrix by Rodrigues's Formula as shown in (2) if we need,

$$\boldsymbol{R} = \cos\theta \boldsymbol{I} + (1-\cos\theta)\boldsymbol{v}\boldsymbol{v}^T + \sin\theta \boldsymbol{v}^\wedge \tag{2}$$

where $\boldsymbol{R}$ is rotation matrix, $\boldsymbol{I}$ is identity matrix, and ^ represents a conversion from the vector to its skew-symmetric.

Pose Losses. For some loss functions of orientation and translation, we use the (non-squared) $\boldsymbol{L_2}$ norm to penalize deviations from the respective ground truths $\hat{\boldsymbol{r}}$ and $\hat{\boldsymbol{t}}$ using stochastic gradient descent. We assume that objects in the scene are all rigid bodies, so we adopt a minimal parameterization of the camera pose with 3 parameters for orientation $\boldsymbol{r}$ and translation $\boldsymbol{t}$ each. These losses for pose vectors are

$$\mathcal{L}_{orientation} = ||\boldsymbol{r} - \hat{\boldsymbol{r}}||_2 \tag{3}$$

$$\mathcal{L}_{translation} = ||\boldsymbol{t} - \hat{\boldsymbol{t}}||_2 \tag{4}$$

The translation ground truth is always normalized so that $||\hat{\boldsymbol{t}}||_2 = 1$, while the magnitude of $\hat{\boldsymbol{r}}$ indicates the angle of the rotation.

3.3 Depth Estimation Module

The depth estimation module is mainly based on an encoder-decoder structure to predict image depth. The input to the module is the same with pose estimation module, and the output is each view depth. Different from other depth estimation method [19] producing disparity images, the depth estimation module is designed to directly predict depth images. The entire network is easier to converge, when we train the network in this way. Considering that the positioning uncertainty of points increasing as the distance increases, and the inverse depth can represent points at infinity, the depth estimation module predicts the inverse depth z rather than the depth d, where

$$z = \frac{1}{d} \tag{5}$$

Our proposed framework estimates a scaling factor s to match the unit translation, so that we can use s to get the final depth value sz.

Depth Loss. We introduce depth loss and inverse depth z to our output. We use $\boldsymbol{L_1}$ loss directly on the inverse depth values for depth output:

$$\mathcal{L}_{depth} = \sum_{i,j} |sz(i,j) - \hat{z}(i,j)| \tag{6}$$

with ground truth $\hat{z}$. Note that we apply the predicted scale s to the predicted value z.

In summary, the final loss function of our system combines the previous losses together. Our final objective becomes

$$\mathcal{L}_{final} = \lambda_o \mathcal{L}_{orientation} + \lambda_t \mathcal{L}_{translation} + \lambda_d \mathcal{L}_{depth} \tag{7}$$

where λ_o, λ_t and λ_d are the weighting of orientation loss, translation loss, and depth loss, respectively.

4. EXPERIMENTS

In this section, we evaluate the performance of our proposed network, and compare with prior approaches on camera pose and image depth estimation. We mainly use the KITTI dataset [23] and TUM dataset [24] for experiments.

4.1 Implement Details

The network training is based on the publicly available TensorFlow framework [22]. During training, we used batch normalization for all the layers except for the output layer. We trained our network with Adam optimizer with parameter β_1 = 0.9 and β_2 = 0.99. Using eight cores of 3.4 GHZ Inter Core i7-3770 and an NVIDIA TITAN X GPU, we trained our proposed framework with a base learning rate of 0.001, reduced by 90% every 100 epochs. The training typically converges after about 200K iterations with a batch size of 32 and the mini-batch size of 8. All the trainings are performed with images sequence obtained from KITTI and TUM datasets.

4.2 Error Metrics

For evaluating the camera pose estimation, we use the standard evaluation method provided along with KITTI dataset [23]. So that, we adopt the average translational root-mean-square error (RMSE) drift t_{rel} (%) and average rotational RMSE drift r_{rel} (°/100m) on the length of 100m-800m.

For fair comparison with state-of-art depth prediction methods, we evaluate our method using the error measures reported in [25]:

RMS: $$\sqrt{\frac{1}{T}\sum_{i \in T} \left\| d_i - d_i^{gt} \right\|^2} \tag{8}$$

log RMS: $$\sqrt{\frac{1}{T}\sum_{i \in T} \left\| \log(d_i) - \log(d_i^{gt}) \right\|^2} \tag{9}$$

abs. relative: $$\frac{1}{T}\sum_{i \in T} \frac{d_i - d_i^{gt}}{d_i^{gt}} \tag{10}$$

sq. relative: $$\frac{1}{T}\sum_{i \in T} \frac{\left\| d_i - d_i^{gt} \right\|^2}{d_i^{gt}} \tag{11}$$

accuracies: $$\%\ \text{of}\ d_i\ s.t.\ max\left(\frac{d_i}{d_i^{gt}}, \frac{d_i^{gt}}{d_i}\right) = \delta < \text{thr.} \tag{12}$$

4.3 Evaluation and Results

KITTI dataset [23] comprises several outdoor scenes captured using a stereo camera mounted on a moving vehicle. Some samples of KITTI dataset are shown in Fig. 3. A total of 21 sequences are provided of which 11 sequences are with ground truth trajectories for training and 10 sequences for evaluation without ground truth. The ground truth information is provided in term of a 3×4 transformation matric which projects i-th coordinate system into 0-th coordinate system. The dataset contains 61 scenes with a typical image being 1242×376 pixels in size. Before entering our framework, we resize these images to 512×256 and recover them before output in order to fit the network.

Figure 3. Some samples of KITTI dataset. There are 21 sequences on different outdoor scenes taken by a stereo camera fixed at a car. And (a) - (f) are some pictures of sequence 1,5,11,14,15,18, respectively.

We use sequences 00-08 to train our network and these remaining two sequences 09, 10 and sequences 11-20 whose ground truths aren't available to test. After implementing we can get their trajectories among these sequences. Trajectories of VISO2-M [26] and SVO [12] are also provided in Fig. 4. There are four trajectories, sequence 11, 12, 14 and 15, are shown in the Fig. 4. From the picture, we can find that our proposed framework outperforms VISO2-M and can be comparable with SVO. Our estimation trajectories are very close to the ground truth. In addition to qualitative comparisons, the detailed quantitative evaluation results about orientation and translation are listed in Table 1. From the table, we can see that our method achieves good pose estimation performance compared to VISO2-M and SVO. Even more, our predictions are nearly twice as accurate as VISO2-M and comparable to SVO. We also perform depth estimation on KITTI dataset and its results are shown in Fig. 5. We can estimate image depth in many scenes, however, the accuracy is not satisfactory when the light changes a lot in different situations. As we have previously assumed that objects in the scenes

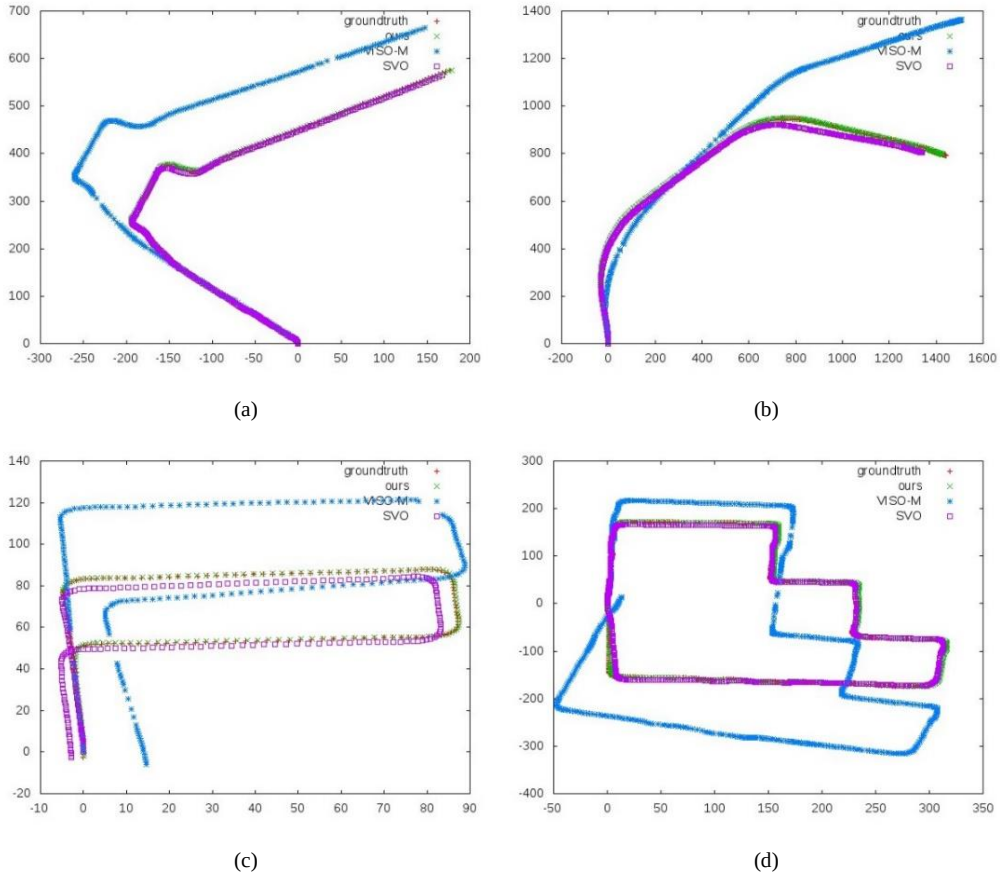

Figure 4. Trajectories of KITTI dataset with our method, VISO2-M, and SVO, where (a) - (d) represent sequence 11,12,14,15, respectively. We can found that our estimations are the closet to ground truth and outperform to VISO2-M and SVO.

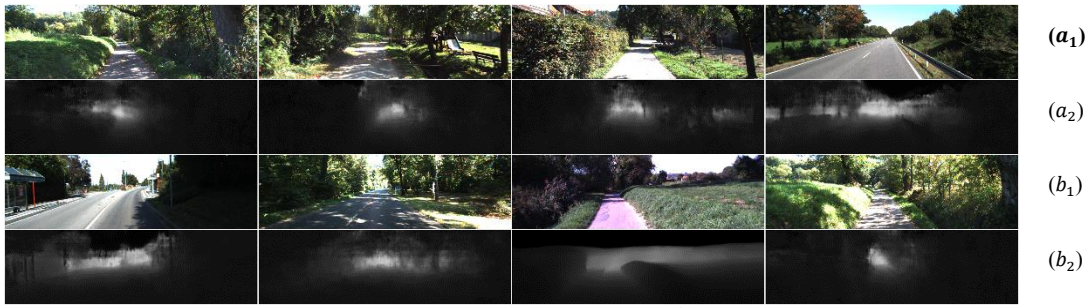

Figure 5. Our depth estimations on KITTI dataset. (a_1) and (b_1) are some raw RGB images of KITTI dataset. (a_2) and (b_2) are our depth estimations corresponded to (a_1) and (b_1), respectively.

Table 1. Pose estimation results of our proposed framework with other methods on KITTI dataset

Seq.	ours		VISO-M		SVO	
	$t_{rel}(\%)$	$r_{rel}(°/100\text{m})$	$t_{rel}(\%)$	$r_{rel}(°/100\text{m})$	$t_{rel}(\%)$	$r_{rel}(°/100\text{m})$
00	**6.63**	**2.27**	18.30	2.72	7.04	2.46
01	**6.12**	**2.09**	21.61	4.73	8.72	2.69
02	4.73	1.51	4.53	1.36	**3.70**	**1.14**
05	**5.40**	**1.87**	20.04	4.68	7.53	2.52
06	5.93	1.94	19.37	3.25	**5.86**	**1.21**
07	**6.73**	**2.54**	23.92	4.90	9.66	2.82
08	**4.93**	**1.67**	24.32	5.69	6.69	2.41
Avg.	**5.78**	**1.98**	18.87	3.90	7.03	2.18

are rigid bodies, we get some wrong estimations on non-rigid objects, likely leaves, grass, pedestrian. However, one can see that our results sometimes preserve the depth boundaries of trees better. For future work, it would be interesting to see if incorporating photometric consistency error into our framework could further improve our results.

TUM dataset [24] contains color and depth images of the Microsoft Kinect along the ground truth. Some samples of TUM dataset are shown in Fig. 6. Data is recorded at full frame rate (30Hz) and sensor resolution with 640×480. The ground truth trajectory of TUM is obtained from a high-precision motion capture system with eight cameras. In addition, TUM presents an evaluation criterion and some comparison tools, so that it is very suitable for SLAM research. The same as KITTI dataset, we resize these images to 512×256 before inputting into our framework, and recover them before output.

We use fr1/floor, fr1/desk, fr2/desk, and fr3/office to train and fr1/room, fr1/desk2, and fr3/structure_texture_far to test. The comparison in terms of absolute trajectory error between our method and other methods, LSD [9], LSD-BS [9], and ORB [4], is listed in Table 2, where Seq.1 is fr1/room, Seq.2 is fr1/desk2, and Seq.3 is fr3/structure_texture_far. From this table, we can know that our proposed network outperforms to other methods on camera trajectory estimation. Although our result is poor than LSD-BS on Seq.2, the mean absolute trajectory error is better than it. Apart from camera pose estimation on TUM dataset, the estimation of image depth is shown in Fig. 7. We have predicted almost objects among scenes and it can distinguish between different objects though our results have a little edge error. Due to the small light changes in indoor scenes, depth estimation results on TUM dataset are better than that on KITTI dataset. The calculation of mean error metric and mean accuracy metric is written in Table 3. Our method achieves the lowest RMS and highest accuracy, and outperforms LSD and ORB on these measurements. In addition, the network is comparable to LSD-BS in log RMS and Abs. relative error.

We can draw that our proposed network can effectively predict camera motion and image depth, and it is comparable even outperforms to these state-of-art methods.

Figure 6. Some samples of TUM dataset. These sequences are captured by Microsoft Kinect in indoor scenes, where (a) – (c) are some images of fr1/desk, fr1/room, and fr2/desk, respectively.

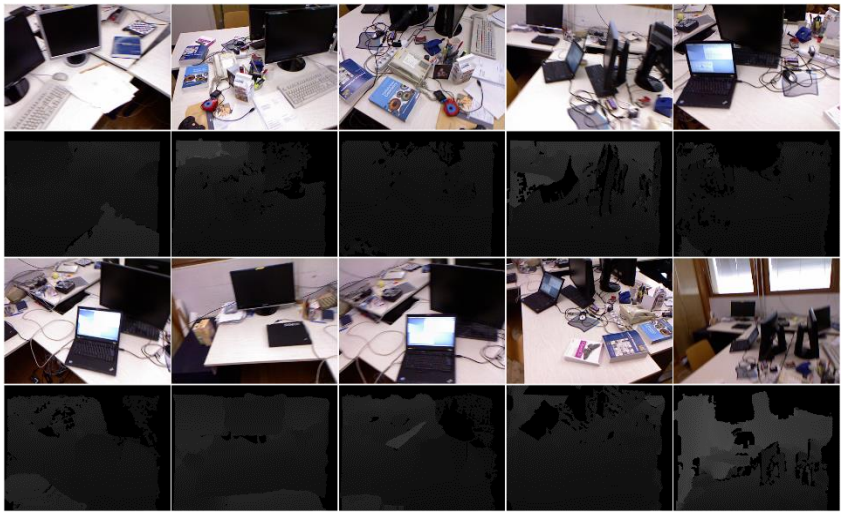

Figure 7. Our depth estimations on TUM dataset. (a_1) and (b_1) are some raw RGB images of TUM dataset. (a_2) and (b_2) are our depth estimations corresponded to (a_1) and (b_1), respectively.

Table 2. Comparison in terms of Absolute Trajectory Error on TUM dataset

	ours	**LSD [10]**	**LSD-BS [10]**	**ORB [4]**
Seq.1	**0.960**	1.627	1.634	1.213
Seq.2	0.328	0.576	**0.253**	0.532
Seq.3	**0.201**	0.726	0.307	0.649
Avg.	**0.496**	0.976	0.698	0.798

Table 3. Performance of the depth estimation between our method and other methods on TUM dataset

Method	**Error Metric (Avg.)**				**Accuracy Metric (Avg.)**		
	RMS	log RMS	Abs. rel.	Sq. rel.	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
ours	**5.372**	**0.293**	**0.185**	**1.190**	**0.807**	**0.946**	**0.953**
LSD	5.731	0.364	0.343	1.545	0.601	0.853	0.920
LSD-BS	5.458	0.301	0.192	1.284	0.713	0.890	0.947
ORB	5.602	0.330	0.204	1.407	0.634	0.871	0.937

5. CONCLUSION

In this paper, we propose a framework, named PD-Net, based on CNN for camera pose and image depth estimation. Different from previous CNN-based network, which predicted camera pose only or view depth only, our framework can calculate camera motion and image depth simultaneously. The network has some shared layers and is divided into two modules to estimate camera trajectory and image depth, respectively. Our proposed framework is trained on some consequent image sequences based on TensorFlow. We verify our method in two different benchmark datasets, KITTI and TUM. Our experimental results show that the proposed framework can complete the task to do visual odometry and it is comparable even outperforms to the state-of-art methods.

Our network is a supervised pipeline and relies on the ground truth. In the future, we will improve our approach so that it no longer requires marked data and can use an unsupervised method to train.

REFERENCES

[1] T. Takafumi, H. Uchiyama, and S. Ikeda. "Visual SLAM algorithms: a survey from 2010 to 2016." IPSJ Transactions on Computer Vision and Applications **9**(16),1-11 (2017).

[2] F. Friedrich and D. Scaramuzza. "Visual odometry: Part ii: Matching, robustness, optimization, and applications." IEEE Robotics & Automation Magazine **19**(2),78-90 (2012).

[3] A. J. Davison. "Real-time simultaneous localisation and mapping with a single camera." IEEE International Conference on Computer Vision, 1043-1050 (2003).

[4] R. Mur-Artal, J. M. Montiel, and J. D. Tardos. "ORB-SLAM: a versatile and accurate monocular SLAM system." IEEE Transactions on Robotics **31**(5),1147-1163 (2015).

[5] F. P. Jorge, J. Ruiz-Ascencio, and J. M. Rendón-Mancha. "Visual simultaneous localization and mapping: a survey." Artificial Intelligence Review **43**(1),55-81 (2015).

[6] D. Nister, O. Naroditsky, and J. Bergen. "Visual odometry." IEEE Computer Society Conference on Computer Vision and Pattern Recognition **1**,1 (2004).

[7] G. Klein and D. Murray. "Parallel tracking and mapping for small AR workspaces." IEEE and ACM International Symposium on Mixed and Augmented Reality, 225-234 (2007).

[8] R. A. Newcombe, J. L. Steven, and J. D. Andrew. "DTAM: Dense tracking and mapping in real-time." IEEE International Conference on Computer Vision, 2320-2327 (2011).

[9] J. Engel, S. Thomas, and C. Daniel. "LSD-SLAM: Large-scale direct monocular SLAM." European Conference on Computer Vision. Springer, Cham, 834-849 (2014).

[10] J. Engel, K. Vladlen, and C. Daniel. "Direct sparse odometry." IEEE Transactions on Pattern Analysis and Machine Intelligence **4**(1),1-14 (2017).

[11] C. Forster, P. Matia, and S. Davide. "SVO: Fast semi-direct monocular visual odometry." IEEE International Conference on Robotics and Automation, 15-22 (2014).

[12] C. Forster, Z. Zhang, M. Gassner, et al. "Svo: Semidirect visual odometry for monocular and multicamera systems." IEEE Transactions on Robotics **33**(2),249-265 (2017).

[13] R. Mur-Artal and J. D. Tardós. "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras." IEEE Transactions on Robotics **33**(5),1255-1262 (2017).

[14] K. R. Konda and R. Memisevic. "Learning Visual Odometry with a Convolutional Network." International Conference on Computer Vision Theory and Applications **1**,486-490 (2015).

[15] G. Costante, M. Mancini, P. Valigi, et al.. "Exploring Representation Learning With CNNs for Frame-to-Frame Ego-Motion Estimation." IEEE Robotics and Automation Letters **1**(1),18-25 (2016).

[16] A. Handa, M. Bloesch, V. Patraucean V, et al. "gvnn: Neural network library for geometric computer vision." European Conference on Computer Vision. Springer, Cham, 67-82 (2016).

[17] R. Li, W. Sen, L. Zhiqiang, et al. "UnDeepVO: Monocular visual odometry through unsupervised deep learning." IEEE International Conference on Robotics and Automation, 1937-1942 (2018).

[18] K. Tateno, F. Tombari, I. Laina, et al. "CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction." IEEE Computer Society Conference on Computer Vision and Pattern Recognition **2**,1 (2017).

[19] C. Godard, O. M. Aodha, and G. J. Brostow. "Unsupervised monocular depth estimation with left-right consistency." IEEE Computer Society Conference on Computer Vision and Pattern Recognition **2**,7 (2017).

[20] U. Benjamin, S. Ricco, C. Schmid, et al. "Demon: Depth and motion network for learning monocular stereo." IEEE Computer Society Conference on Computer Vision and Pattern Recognition **5**,6 (2017).

[21] V. Sudheendra, S. Ricco, C. Schmid, et al. "Sfm-net: Learning of structure and motion from video." arXiv preprint arXiv:1704.07804 (2017).

[22] M. Abadi, P. Barham, J. Chen, et al. "Tensorflow: a system for large-scale machine learning." USENIX Symposium on Operating Systems Design and Implementation **16**,265-283 (2016).

[23] A. Geiger, P. Lenz, C. Stiller, et al. "Vision meets robotics: The KITTI dataset." The International Journal of Robotics Research **32**(11),1231-1237 (2013).

[24] J. Sturm, N. Engelhard, F. Endres, et al. "A benchmark for the evaluation of RGB-D SLAM systems." IEEE/RSJ International Conference on Intelligent Robots and Systems, 573-580 (2012).

[25] F. Liu, C. Shen, G. Lin, et al. "Learning Depth from Single Monocular Images Using Deep Convolutional Neural Fields." IEEE Transactions on Pattern Analysis and Machine Intelligence **38**(10),2024-2039 (2016).

[26] A. Geiger, Z. Julius, and S. Christoph. "Stereoscan: Dense 3d reconstruction in real-time." Intelligent Vehicles Symposium, 963-986 (2011).

Authors' background

***This form helps us to understand your paper better, the form itself will not be published. You can delete it when you prepare the final paper if it's accepted.**

Your Name	**Title***	**Research Field**	**Personal website**
Xuyang Meng	Phd Candidate	Computer Vision and 3D Reconstruction	
Chunxiao Fan	Full Professor	Big Data Analytics, Computer vision, and 3D Reconstruction	
Yue Ming	Associate Professor	Computer Vision, Pattern Recognition, and 3D Reconstruction	

***Title can be chosen from: master student, Phd candidate, assistant professor, lecture, senior lecture, associate professor, full professor.**