Frame-to-Frame Visual Odometry Estimation Network With Error Relaxation Method

SANGWON HWANG\textsuperscript{1}, (Student Member, IEEE), MYEONGAH CHO\textsuperscript{1}, (Student Member, IEEE), YUSEOK BAN\textsuperscript{2}, (Member, IEEE), AND KYUNGGAE LEE\textsuperscript{3}, (Member, IEEE)

\textsuperscript{1}School of Electrical and Electronic Engineering, Yonsei University, Seoul 03722, South Korea
\textsuperscript{2}School of Electronics Engineering, Chungbuk National University, Cheongju 28644, South Korea
\textsuperscript{3}School of Artificial Intelligence, Yong In University, Yongin 17092, South Korea
Corresponding author: Kyungjae Lee (kjlee@yongin.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) funded by the Korean Government (MSIT) under Grant 2020R1G1A1102041, in part by the Korea Institute for Advancement of Technology (KIAT) funded by the Korean Government (MOTIE-HRD Program for Industrial Innovation) under Grant P0020536, and in part by the National Research Foundation of Korea (NRF) funded by the Korean Government (MSIT) under Grant 2022R1A5A8026986.

ABSTRACT Estimating frame-to-frame (F2F) visual odometry with monocular images has significant problems of propagated accumulated drift. We propose a learning-based approach for F2F monocular visual odometry estimation with novel and simple methods that consider the coherence of camera trajectories without any post-processing. The proposed network consists of two stages: initial estimation and error relaxation. In the first stage, the network learns disparity images to extract features and predicts relative camera pose between adjacent two frames through the attention, rotation, and translation networks. Then, loss functions are proposed in the error relaxation stage to reduce the local drift, increasing consistency under dynamic driving scenes. Moreover, our skip-ordering scheme shows the effectiveness of dealing with sequential data. Experiments with the KITTI benchmark dataset show that our proposed network outperforms other approaches with higher and more stable performance.

INDEX TERMS Deep neural network, visual odometry, camera pose, odometry drift, camera trajectory.

I. INTRODUCTION

Estimating camera orientation and location (pose or ego-motion) in every frame, called visual odometry (VO), is an essential module for navigation, robotics, and augmented reality. When accurate camera poses are predicted from all frames, it is valuable and helpful to analyze the surrounding world based on the camera coordinate and understand both indoor and outdoor environments, including autonomous vehicles \cite{1, 2, 3}, unmanned robotics \cite{4, 5}, and quadrocopters \cite{6}.

Especially in robotics vision, such as Simultaneous Localization and Mapping (SLAM) system or Structure from Motion (SfM), figuring out camera pose through sequential images has been widely developed in an offline or online manner. One of the trials is geometric-based approaches, such as feature-based \cite{7, 8, 9}, semi-direct \cite{10, 11}, and direct-based methods \cite{12}. They predict the camera pose by matching corresponding features between two frames and optimize it with 3D geometry manners \cite{13} or optimizers like the weighted Gauss-Newton method. However, these methods heavily rely on continuous laborious camera pose optimization, such as bundle adjustment \cite{14} and loop closure \cite{15}, rather than estimating the precise F2F camera pose.

In our system, we aim to estimate the F2F camera pose with only two adjacent frames and replace those laborious optimizations with error relaxation methods. Furthermore, given that camera movements vary in a driving environment, we propose a novel and simple scheme called a skip-ordering method. By skipping one interval frame from the image sequences when training the network, we find that it is effective to learn the enriched camera pose containing various speeds of camera motion.
A typical pipeline in ORB-SLAM, suggested by Mur-Artal et al. [9], is to compute correspondences with ORB descriptor [21] and estimate camera pose by EPNP algorithm [22]. They also introduced a loop closing optimizer, which eliminates odometry drift when the camera revisits the same place they had visited using DBoW2 [23] with ORB. However, since this descriptor has the representative of FAST keypoint [24], the map has a low density.

To overcome this problem and improve the quality of the map, various studies have been attempted to view the entire image rather than the point. Newcombe et al. [12] used the alignment of previous and current images rather than matching interesting points for predicting camera pose and generating a highly-dense 3D map. Similar trials [25] were introduced using the depth map of each pixel which is the output of the Kinect sensor. However, using whole images to predict camera pose and generate a 3D map has difficulty updating them compared to using interest points.

Engel et al. [10] proposed semi-dense approaches based on a probabilistic depth map representation to solve these problems. They suggested a probabilistic depth map to align the high-gradient region of images such as corners and edges, called semi-dense, for effectively estimating camera pose and update. DSO [26] also proposed joint optimization of parameters, including geometry and camera motion, and it was designed to minimize the photometric error.

B. LEARNING-BASED APPROACHES

With the potential improvement of CNNs in many robotics vision areas [27], various approaches have attempted to solve the camera odometry estimation by modeling it as a regression problem. One of the first trials with CNN’s method was proposed by Konda [28], [29], [30] who showed the possibility of predicting the speed and directional changes of the camera using depth information. Since camera motion is differently predicted according to the distance of an object from the camera, they insisted that depth images can be used to estimate visual odometry.

Flowdometry [31] proposed that an optical flow image contains more motion information than other images to train the end-to-end convolutional neural network. It used pre-trained architecture [32], which is FlowNetS, to perform odometry regression. P-CNN [33] was also trained in both...
dense optical flow and RGB to extract more visual information. It explained that training both images is learning new feature representations.

A recurrent neural network (RNN) is functional architecture for processing continuous data. Wang et al. [34] suggested an RNN, called DeepVO, to estimate accurate visual odometry using prior knowledge. They argued that it is essential to know much information from past images for computing camera motion. Wang et al [35] also presented RNN models called denseSLAMNet for 3D reconstruction from consecutive images. However, unlike CNN, these networks have many parameters, so it needs an enormous capacity to train successive images.

Recently, attracted to the fact that self-supervised learning does not need labels, several trials [36], [37], [38], [39] have been studied to train the relationship between depth information and camera pose. Given the exact depth value in one frame and camera intrinsic parameters, the 3D relative pose of the adjacent frame is computed geometrically. Even though these approaches are pretty fashionable, they focus on optimizing both depth and camera pose between adjacent frames, not predicting camera trajectory in all frames in a scene that might contain camera drift.

### III. PROPOSED METHOD

This section describes the proposed network architecture as shown in Fig. 2 and the framework for learning F2F camera odometry with the error relaxation methods in detail.

**A. NETWORK ARCHITECTURE**

1) **ENCODER**

Various CNN architecture, such as VGG [40], ResNet [41], and DenseNet [42], has been studied and shown tremendous performance in classification, recognition, and segmentation tasks. However, those networks are unsuitable for estimating exact camera poses in VO estimation. To extract geometry features for VO estimation, it is important not to lose the spatial information in the feature domain. For preserving this information, we use a convolution layer (stride 2) for embedding features rather than a pooling layer as mentioned in [43].

For acquiring rotation and translation from two images, each feature should be extracted in the same way and needs to be analyzed about the displacement of those features. Thus, the encoder shares their kernel weights, which means that each input image is calculated with the same convolution layer. Each output produced by the shared encoder is operated with the output of the attention network, and then channel-wise stacked features are passed through the rotation and translation networks.

2) **ATTENTION NETWORK**

To be robust to outliers when estimating VO in a geometric-based method, hand-crafted features are investigated whether these fit their model or not. For instance, the *survival of the fittest* policy is applied to reduce outliers in the ORB-SLAM system [9], and VISO-2 [7] also exploits Random sample consensus (RANSAC) for removing them.

Inspired by these concepts, the attention network is designed to assign weights to the output of the encoder. The disparity image has an inverse proportion to the depth map. Nearby objects, which have large pixel intensity in the disparity image, have more influence in predicting the poses of the camera. For example, predicting VO by comparing the displacement of features from distant objects or dynamic objects is quite challenging. This network helps to highlight and de-emphasize the features by multiplying the output of the encoder pixel-wisely. All layers in the attention network
use the ReLU function except for the last layer. The Sigmoid function is used in the last layer for setting the output between 0 and 1.

3) ROTATION AND TRANSLATION NETWORKS
The rotation and translation networks learn to predict camera orientation and location using geometric features. Producing orientation and location of camera motion through one network extracts the feature that should contain both rotation and translation information. Since this information has different feature spaces, we design our rotation and translation networks independently. The rotation network has two additional layers than the translation network because the rotation has higher non-linearity properties [44]. Our network outputs a camera pose with six values with fully connected layers.

B. CAMERA POSE
In this section, we describe the framework for learning F2F camera odometry from two monocular images, \( I_i \) and \( I_{i+1} \) as shown in Fig. 3. Our network outputs the transformation matrix \( T \), given by a 3D camera rotation matrix \( R \) and translation vector \( p \):

\[
T = [R|p]
\]  

where \( R \) and \( p \) belong to the \( SO(3) \) and \( \mathbb{R}^3 \) sets, respectively. For convenience, we notate the forward, rightward, and downward axis as \( z, x, \) and \( y \).

Every rotation matrix should satisfy the following condition: both orthogonality \((RR^T = I)\) and determinant of matrix \((\det(R) = 1)\), which is called the special orthogonal group. In robotics vision, various mathematical representations of 3D rotation exist, such as quaternion, Euler angle, and the Lie Group. Note that most of the driving scene is straight ahead; the fourth value of quaternion is abnormally high compared to the other values. This hinders learning in the network and interrupts the balance of rotation values. The Lie group has the disadvantage of not being able to know the reference axis in a stopped-driving environment. Therefore, our network produces the 6-DOF transformation between frames with three Euler angles and translation vectors. For convenient notation, Euler angle and translation vector are denoted as \( \theta \) and \( p \), and \( T \) is composed of 6 values, \([\theta_x, \theta_y, \theta_z, p_x, p_y, p_z]\).

In the initial estimation stage, we train our network with the following forward loss function:

\[
L_f = \lambda_0 \sum ||\theta - \hat{\theta}||^2 + \sum ||p - \hat{p}||^2
\]  

where \((\cdot)\) and \((\hat{\cdot})\) denote the label and estimated camera pose, and \(\lambda\) is a scale factor to balance them. This loss function is composed of the Mean Square Error (MSE) of Euler \(\theta\) and translation \(p\).

C. POSE RELAXATION METHODS
The set of transformations with respect to the first frame is computed as follows:

\[
T_{0,n} = \prod_{i=1}^{n} T_{i-1,i}
\]  

The fundamental issue of F2F pose estimation is reducing propagated errors. As shown in Eq. 3, if drift occurs at the beginning, the trajectory error might be accumulated, and the gap continues to grow. Therefore, the process of relaxing this error is vital for predicting the F2F camera pose.

On the other hand, geometric-based approaches clean up this error through loop closure and bundle adjustment continuously [46]. Moreover, they are optimizing predicted camera poses locally and globally at post-processing. However, since our purpose is to design a network for learning both prediction and correction at once, we have divided the training steps into two stages; initial estimation and error relaxation, as shown in Fig. 1. Furthermore, the novel and simple loss function, called a correction loss which is described in detail in Section III-C2, is defined to decrease the odometry drift in the error relaxation stage, which causes higher and more stable performances than other networks.
Note that ground-truths provided by the KITTI benchmark [45] are unbalanced, as shown in Fig. 4. Even though the distribution of the y-axis is widely spread, the Euler angle of all axes is usually Gaussian. In the F2F translation z-axis, however, has only positive values. This means that the network only learns the forward direction, which leads to a one-way learning process. To obtain accurate camera poses, we defined our additional loss function with an adjacent estimated camera hypothesis that the proposed network learned with only disparity images, I_i, I_{i-1}, and I_{i-2} as a skip-ordering method that allows learning large movements of pixels (e.g., high speed). We also conducted the experiment by skipping two frames, the pair of I_i and I_{i-3}. However, the results were poor because few overlapped regions between images exist. Our network becomes robust against various speeds due to the skip ordering scheme, which is explicitly explained in Sec. IV.

IV. EXPERIMENT

In this section, we evaluate our proposed method with the KITTI dataset [45]. Our network is compared to other geometric-based and learning-based approaches using only monocular images. Since monocular visual odometry has a fundamental problem; scale ambiguity, some algorithms are re-scaled by referring to the ground-truth.

A. DATASETS

KITTI datasets are well-known benchmarks that offer data related to the driving scene and evaluate submitted algorithms. Notably, they provide stereo images and GPS/INS information in the odometry dataset. We used the left original image (370 × 1220) to train the network and made F2F labels using provided INS data. KITTI Datasets consist of 11 sequences (00-10) with labels and 11 (11-21) without labels. We set the experimental environment identical to that of other algorithms. The training and test sets consisted of from 00 to 07 (16338 images) and 08 to 10 (6863 images) sequences, respectively.

The training image is an output of the method proposed by Godard et al. [16]. Their network was trained on Cityscape [47] and the KITTI dataset and evaluated on a different dataset [48]. We leverage the pre-trained network to train our network because the disparity has spatial clues on each pixel. Although color images have more texture than disparity, they have redundancy features, thus it prevents training camera poses. The experiment results support our hypothesis that the proposed network learned with only disparity images outperforms the network learned with color or color + disparity, as described in Sec. IV-D.

B. IMPLEMENTATION DETAILS

To preserve geometric features, we did not employ batch normalization in any layer. Instead, we heuristically set \( \lambda_{\theta} \) to 350 to balance rotation and translation losses. We used Adam [49] optimization and Xavier initialization. After training our network until 30 epochs for the initial estimation stage, bi-directional and correction losses are added to re-train our network for 10 epochs in the error relaxation stage.

C. ODOMETRY RESULT

To evaluate our performance, we utilized KITTI benchmark evaluation, \( T_{\text{error}}(\%) \), and \( R_{\text{error}}(\%/100\text{m}) \) based on the KITTI
TABLE 1. Visual Odometry results evaluated on Sequence 08, 09, and 10. Both $T_{error}$ and $R_{error}$ represent translation and rotation drift. Since each sequence has different frames, we also computed the standard deviation (std) to prove the stability of the odometry result. Algorithms marked with stars are indicated to be re-scaled by force.

| Algorithms      | Seq. 08   | Seq. 09   | Seq. 10   | Mean (±std) |
|-----------------|-----------|-----------|-----------|-------------|
|                 |   T_{error} [%] |   R_{error} [°/100m] |   T_{error} [%] |   R_{error} [°/100m] |   T_{error} [%] |   R_{error} [°/100m] |
| VISO2-M [7]      | 26.21    | 2.47      | 4.09      | 1.24        | 60.02      | 6.69        | 24.91 (±0.2352) | 2.66 (±0.0023) |
| ORB-SLAM (LC) [9]| -        | -         | 16.23     | 1.36        | -          | -          | 16.23 (-)        | 1.36 (-)          |
| *SFMlearner [38]| -        | -         | 37.07     | 6.82        | 41.38      | 17.69       | 38.47 (±0.3212) | 10.37 (±0.0062) |
| Zhai et al. [37]| -        | -         | 11.92     | 3.60        | 12.62      | 3.43        | 12.15 (-)        | 3.54 (-)          |
| SVR-VO [18]      | 15.42    | 3.63      | 10.50     | 4.45        | 21.97      | 5.45        | 15.02 (±0.0486) | 4.01 (±0.0009) |
| Flowdometry [31] | 9.98     | 5.44      | 12.64     | 8.04        | 11.65      | 7.28        | 10.77 (-)        | 6.23 (-)          |
| P-CNN VO [33]    | 7.60     | 1.87      | 6.75      | 2.52        | 21.23      | 4.05        | 8.96 (±0.0721)  | 2.35 (±0.0010)  |
| Ours             | **5.88** | **2.25**  | **7.77**  | **3.51**    | **8.82**   | **2.62**    | **6.63 (±0.0359)** | **2.67 (±0.0028)** |

FIGURE 6. Comparison of an average of rotation and translation error of sequences 08, 09, and 10 in terms of length [m] and speed [km/s].

Odometry dataset. They measure the difference between estimation and ground-truth about translation and rotation depending on the length [m] and speed [km/s]. These are the major evaluation metrics of the KITTI Odometry benchmark for ranking various odometry algorithms.

Table 1 compares the proposed network with other algorithms, including handcrafted [7], [9] and learning-based methods [18], [31], [33], [38]. Table 1. reports the average error and standard deviation of camera odometry in each test sequence. VISO2-M [7] and SFMlearner [38] algorithms, which are labeled as (*) mark, cannot solve scale problems themselves. Therefore, we recovered these scale issues using the ratio of estimated and ground-truth. Our translation accuracy is the most accurate among all algorithms, and rotation error shows competitive results. Moreover, we find that our results are stable by computing the standard deviation. ORB-SLAM (LC) with loop closure using ORB descriptors to match features in monocular sequential images shows low performance in terms of translation. In spite of using optimization, such as loop closure and bundle adjustment, we can find that our error relaxation network effectively works by showing lower translation errors. Interestingly, geometric-based methods (e.g. VISO2-M, ORB-SLAM) are profoundly low rotation errors compared to the learning-based method.

SVR-VO [18] exploit support vector machine to handle non-linear relations between features. Since SVM has
TABLE 2. Ablation Experiment. RGB and RGB+Disparity stand for the type of training image. A, S, and C represent whether attention network, skip-ordering, and correction loss are included or not.

| Model                        | Seq. 08 \( \text{T}_{\text{error}} \) [%] | Seq. 09 \( \text{T}_{\text{error}} \) [%] | Seq. 10 \( \text{T}_{\text{error}} \) [%] | Mean (±std)                                                                 
|------------------------------|------------------------------------------|------------------------------------------|------------------------------------------|---------------------------------------------------------------------------|
| RGB / +SA                    | 16.87                                    | 5.27                                     | 13.32                                    | 22.80                                     | 5.93                                     | 16.71 (±0.1649)                | 5.31 (±0.0421)                |
| RGB+Disparity / +SA          | 19.48                                    | 4.73                                     | 18.50                                    | 7.18                                      | 21.13                                    | 4.30                                     | 19.43 (±0.1570)                | 5.24 (±0.0534)                |
| Disparity / +SA              | 10.10                                    | 3.69                                     | 10.94                                    | 3.24                                      | 13.86                                    | 8.43                                     | 10.70 (±0.0941)                | 4.11 (±0.0050)                |
| Disparity / +A               | 9.85                                     | 3.17                                     | 12.26                                    | 3.11                                      | 11.38                                    | 6.77                                     | 10.56 (±0.1001)                | 3.55 (±0.0054)                |
| Disparity / +S               | 7.28                                     | 3.31                                     | 7.44                                     | 2.71                                      | 9.65                                     | 3.90                                     | 7.57 (±0.0632)                | 3.24 (±0.0049)                |
| Disparity / +SA              | 6.96                                     | 2.45                                     | 7.95                                     | 2.82                                      | 9.73                                     | 3.42                                     | 7.26 (±0.0580)                | 2.64 (±0.0026)                |
| Disparity / +SAC             | 5.88                                     | 2.25                                     | 7.77                                     | 3.51                                      | 8.82                                     | 2.62                                     | 6.63 (±0.0359)                | 2.67 (±0.0028)                |

FIGURE 7. Qualitative results on visual odometry. Sequences 08, 09, and 10 are plotted with other algorithms. The unit of the x and y-axis is a meter.

Regarding the deep learning-based approaches, P-CNN VO [33] shows better performance than other networks. P-CNN [33] trains both color and dense optical flow images to extract more visual information. Since the optical flow image implies the movements over the temporal axis, both P-CNN and Flowdometry [31] show low rotation error compared to other methods. However, our network shows a lower translation error than optical flow-based networks.

Detailed errors in terms of length [m] and speed [km/s] are shown in Fig. 6. Our methods appear comparative results compared with other algorithms. Due to the skip-ordering and error correction loss, we achieve stable results in the high-speed scene. We plot the test sequences to evaluate qualitative trajectories as shown in Fig. 7. Sequence 08, 09, and 10 have 4071, 1591, and 1201 frames, respectively. Since the 08 sequence has a large number of frames (4071), lots of errors are accumulated continuously compared to our network.

D. ABLATION EXPERIMENT

To prove our proposed method, we validate several ablation experiments as shown in Table 2. The model, such as RGB and RGB+Disparity (stacked channel-wisely) stands for the type of training image. We find that the network that is trained by color image shows a higher error than the network that is trained by disparity due to the redundancy features of the color image. The results of the RGB+Disparity model also appear to have higher translation errors than the RGB model.

A, S, and C represent whether attention network, skip-ordering, and correction loss are included or not. Although Disparity+/+S shows the best performance in Seq 09, error correction loss reduces the overall loss. Interestingly, just skipping the interval image shows that the average translation and rotation error reduced from 10.70 to 7.57 and 4.11 to 3.24. Furthermore, the attention network helps to reduce both translation and rotation errors. As shown in Fig. 8, we find that it filters out the non-confidential region.

FIGURE 8. (a) Left images of 00 sequence provided by KITTI benchmark. (b) Disparity image that we used to train our model. (c) Feature map of the attention network output. The area of the moving bike marked red means that it is close to zero.
Fig. 9 shows how our correction loss functions are useful to estimate camera odometry qualitatively. At the beginning of the frame, the network which trained only forward loss function cannot remove accumulated errors. However, the network with error relaxation loss eliminates them effectively even in the complicated sequence of 09.

V. CONCLUSION

We have investigated several experiments for achieving F2F camera location and orientation simultaneously with the learning-based method. The essential issue in predicting F2F VO is propagating errors continuously. The geometric-based method optimizes their initial post-process estimations to solve these problems using well-known loop closing and bundle adjustment. To replace that optimizer, we define bi-directional and correction loss functions in the error relaxation stage. These loss functions increase the coherence of camera trajectories and remove errors that occurred in the initial estimation stage. In future work, we plan to extend our model to train the multimodal dataset (e.g., LiDAR and RGB) of another dataset to increase the accuracy of camera odometry. Color images have plentiful features, and LiDAR has a high-accuracy depth value, thus incorporating their strengths could improve the performance.

REFERENCES

[1] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, “A solution to the simultaneous localization and map building (SLAM) problem,” IEEE Trans. Robot. Autom., vol. 17, no. 3, pp. 229–241, Jun. 2001.

[2] A. Howard, “Real-time stereo visual odometry for autonomous ground vehicles,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., Sep. 2008, pp. 3946–3952.

[3] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, “Visual odometry and mapping for autonomous flight using an RGB-D camera,” in Proc. Int. Symp. Robot. Res., 2011, pp. 235–252.

[4] F. A. Aviat Cheein and R. Carelli, “Agricultural robotics: Unmanned robotic service units in agricultural tasks,” IEEE Ind. Electron. Mag., vol. 7, no. 3, pp. 48–58, Sep. 2013.

[5] J. Kelly, S. Saripalli, and G. S. Sukhatme, “Combined visual and inertial navigation for an unmanned aerial vehicle,” in Field and Service Robotics. Berlin, Germany: Springer, 2008, pp. 255–264.

[6] J. Engel, J. Sturm, and D. Cremers, “Camera-based navigation of a low-cost quadrocopter,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., Oct. 2012, pp. 2815–2821.

[7] A. Geiger, J. Ziegler, and C. Stiller, “StereoScan: Dense 3D reconstruction in real-time,” in Proc. IEEE Intell. Vehicles Symp. (IV), Jun. 2011, pp. 963–968.

[8] G. Klein and D. Murray, “Parallel tracking and mapping for small AR workspaces,” in Proc. 6th IEEE/ACM Int. Symp. Mixed Augmented Reality, Nov. 2007, pp. 225–234.

[9] R. Mur-Artal, J. M. Montiel, and J. D. Tardós, “ORB-SLAM: A versatile and accurate monocular SLAM system,” IEEE Trans. Robot., vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[10] J. Engel, T. Schöps, and D. Cremers, “LSD-SLAM: Large-scale direct monocular SLAM,” in Proc. Eur. Conf. Comput. Vis., 2014, pp. 834–849.

[11] C. Forster, M. Pizzoli, and D. Scaramuzza, “SVO: Fast semi-direct monocular visual odometry,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2014, pp. 15–22.

[12] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, “DTAM: Dense tracking and mapping in real-time,” in Proc. Int. Conf. Comput. Vis., Nov. 2011, pp. 2320–2327.

[13] D. Nistér, O. Naroditsky, and J. Bergen, “Visual odometry,” in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 1, Jun. 2004, pp. 1–8.

[14] K. Konolige and M. Agrawal, “FrameSLAM: From bundle adjustment to real-time visual mapping,” IEEE Trans. Robot., vol. 24, no. 5, pp. 1066–1077, Oct. 2008.

[15] M. Labbe and F. Michaud, “Online global loop closure detection for large-scale multi-session graph-based SLAM,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., Sep. 2014, pp. 2661–2666.

[16] G. Godard, O. M. Aodha, and G. J. Brostow, “Unsupervised monocular depth estimation with left-right consistency,” in Proc. IEEE/RSJ Conf. Vis. Pattern Recognit. (CVPR), Jul. 2017, vol. 2, no. 6, p. 6.

[17] S. Meister, J. Hur, and S. Roth, “UnFlow: Unsupervised learning of optical flow with a bidirectional census loss,” 2017, arXiv:1711.07837.

[18] T. A. C_surveyg, G. Costante, P. Valigi, and E. Ricci, “Evaluation of non-geometric methods for visual odometry,” Robot. Auto. Syst., vol. 62, no. 12, pp. 1717–1730, Dec. 2014.

[19] J. A. K. Suykens and J. Vandewalle, “Least squares support vector machine classifiers,” Neural Process. Lett., vol. 9, no. 3, pp. 293–300, Jun. 1999.

[20] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, “Bundle adjustment—A modern synthesis,” in Proc. Int. Workshop Vis. Algoritms (ICCV), 2000, pp. 298–372.

[21] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” in Proc. Int. Conf. Comput. Vis., Nov. 2011, pp. 2564–2571.

[22] V. Lepetit, F. Moreno-Noguer, and P. Fua, “EPnP: An accurate O(n) solution to the PNP problem,” Int. J. Comput. Vis., vol. 80, no. 2, p. 155, 2009.

[23] D. Galvez-López and J. D. Tardos, “Bags of binary words for fast place recognition in image sequences,” IEEE Trans. Robot., vol. 26, no. 5, pp. 1188–1197, Oct. 2012.

[24] M. Ozusyal, M. Calonder, V. Lepetit, and P. Fua, “Fast keypoint recognition using random ferns,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 32, no. 3, pp. 448–461, Mar. 2010.

[25] C. Kerl, J. Sturm, and D. Cremers, “Robust odometry estimation for RGB-D cameras,” in Proc. IEEE Int. Conf. Robot. Autom., May 2013, pp. 3748–3754.
[26] J. Engel, V. Kolotun, and D. Cremers, “Direct sparse odometry,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 40, no. 3, pp. 611–625, Mar. 2017.

[27] X. Yin, X. Wang, X. Du, and Q. Chen, “Scale recovery for monocular visual odometry using depth estimated with deep convolutional neural fields,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 5870–5878.

[28] K. Konda and R. Memisevic, “Learning visual odometry with a convolutional network,” in Proc. 10th Int. Conf. Comput. Vis. Theory Appl., 2015, pp. 486–490.

[29] X. Liu, H. Zhao, M. Tian, L. Sheng, J. Shao, S. Yi, J. Yan, and X. Wang, “HydraPlus-Net: Attentive deep features for pedestrian analysis,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 350–359.

[30] S. Nam and S. J. Kim, “Modelling the scene dependent imaging in cameras with a deep neural network,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 1717–1725.

[31] P. Müller and A. Savakis, “Flowdometry: An optical flow and deep learning based approach to visual odometry,” in Proc. IEEE Winter Conf. Comput. Appl. Comput. Vis. (WACV), Mar. 2017, pp. 624–631.

[32] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. V. D. Smagt, D. Cremers, and T. Brox, “FlowNet: Learning optical flow with convolutional networks,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 2758–2766.

[33] G. Costante, M. Mancini, P. Valigi, and T. A. Ciarfuglia, “Exploring representation learning with CNNs for frame-to-frame ego-motion estimation,” IEEE Robot. Autom. Lett., vol. 1, no. 1, pp. 18–25, Jan. 2016.

[34] S. Wang, R. Clark, H. Wen, and N. Trigoni, “DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2017, pp. 2043–2050.

[35] R. Wang, J.-M. Frahm, and S. M. Pizer, “Recurrent neural network for learning densedepth and ego-motion from video,” 2018, arXiv:1805.06558.

[36] R. Li, S. Wang, Z. Long, and D. Gu, “UnDeepVO: Monocular visual odometry through unsupervised deep learning,” 2017, arXiv:1709.06841.

[37] H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and J. M. Reid, “Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jan. 2018, pp. 340–349.

[38] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, “Unsupervised learning of depth and ego-motion from video,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 1851–1858.

[39] N. Yang, L. von Stumberg, R. Wang, and D. Cremers, “D3 VO: Deep depth, deep pose and deep uncertainty for monocular visual odometry,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 1281–1292.

[40] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2014, arXiv:1409.1556.

[41] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 770–778.

[42] P. Isola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, “DenseNet: Implementing efficient ConvNet descriptor pyramids,” 2014, arXiv:1404.1869.

[43] A. Handa, M. Bloesch, V. Pátráucesc, S. Stent, J. Mccormac, and A. Davison, “gvcnn: Neural network library for geometric computer vision,” in Proc. Eur. Conf. Comput. Vis., Oct. 2016, pp. 67–82.

[44] A. Kendall, M. Grimes, and R. Cipolla, “PoseNet: A convolutional network for real-time 6-DOF camera relocalization,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 2938–2946.

[45] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? The KITTI vision benchmark suite,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2012, pp. 3354–3361.

[46] R. Mur-Artal and J. D. Tardós, “Fast relocalisation and loop closing in keyframe-based SLAM,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2014, pp. 846–853.

[47] J. Nistér, O. Naroditsky, and J. Bergen, “Visual odometry with a城区, itsees dataset for semantic urban scene understanding,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 3213–3223.

[48] A. Saxena, M. Sun, and A. Y. Ng, “Make3D: Learning 3D scene structure from a single still image,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 31, no. 5, pp. 824–840, May 2009.

[49] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014, arXiv:1412.6980.