ORB-SLAM Implementation Using Deep Learning Methods for Visual Odometry

Yibo Cao, Yijun Luo, Tianyu Wang

***Abstract*— Although state-of-the-art visual odometry algorithms have shown excellent performance in modern SLAM systems, they still suffer from heavy parameter tuning processes while being implemented in completely new environments. Deep learning based visual odometry has been seen as a candidate for solving this problem and adding accuracy and robustness to visual odometry modules in SLAM systems. In this project we make a comparison of an end-to-end deep learning based visual odometry with a traditional geometry based visual odometry in ORB-SLAM pipeline. DeepVO leverages the advantages of CNNs and RNNs on image processing and neighbouring information co-relating. It has the advantage of only using raw RGB images as input without any other carefully tuned parameters, such as camera calibration. While the visual odometry of ORB-SLAM2 uses corner detector to extract features for bag of words method, then it applies camera calibration and careful parameter tuning for pose estimation. By testing both visual odometry module in ORB-SLAM system and the deep learning based visual odometry algorithm on KITTI dataset, we found that the deep learning based visual odometry still cannot outperform the conventional visual odometry methods. But we believe that the deep learning based visual odometry can be used as a good complementary of the conventional methods to improve its robustness in different environments.**

I. Introduction

Visual odometry, an essential module in SLAM pipelines, mainly aims to incrementally estimate the path of the robot pose after pose and perform local optimization for local consistency. Convectional visual odometry system typically includes camera calibration, feature extraction, feature matching, outlier rejection motion estimation, scale estimation and local optimisation. Among these modules in typical visual odometry frameworks, feature extraction is an important part for visual odometry to get accurate frame correspondence, which is later used to calculate the robot's pose. Although state-of-the-art visual odometry algorithms have shown excellent performances in different SLAM pipelines, they are usually specifically hard-coded for applications in an environment and need a significant of engineering effort in tuning parameters in various modules.

Recently learning based approach has arrived SLAM problem and has surpassed traditional method in both accuracy and robustness on object detection. We propose to replace the visual odometry part of odometry pose estimation in ORB-SLAM [1], [2] system with a deep learning based algorithm, the idea of which comes from the paper [3]. In detail, we will use the end-to-end learning based visual odometry, which is based on learnt optical flow features, to replace the visual odometry module in ORB-SLAM2, and test the odometry performance in the whole ORB-SLAM2 system based on KITTI dataset.

Optical flow has a relatively much denser information compared to feature extraction method such as the corner detection method used in ORB-SLAM2 paper [2]. Using denser feature information should be able to improve the robustness of odometry estimation. The learning based visual odometry method proposed in [3] shows a great accuracy improvement compared to traditional methods, while its computing time is still fast. Implementing this method into an odometry system could provide us with a more robust but still fast SLAM system.

The rest of this report is organized as follows: Section II provides a brief review of the literature that inspires our work and we build our work upon; Section III describes the theories and methods we implemented in our ORB-SLAM pipeline and learning-based visual odometry module; Section IV discusses the data and code we implemented; Section V demonstrates the results of our method and comparisons with the baseline; Section VI includes our discussions on the results; Section VII discusses of challenges you faced and Section VIII concludes our project.

II. Literature Review

A. ORB-SLAM

1) ORB-SLAM: We choose ORB-SLAM [1] as the basic architecture of our implementation. This paper introduces its components and major contributions. ORB-SLAM uses ORB features, oriented multi-scale FAST corners with a 256 bits descriptor associated, for all tasks because it is real time without GPU. The algorithm operates in three threads: tracking, localizing the camera with every frame and deciding when to insert a new keyframe; mapping, processing new keyframes and performing local bundle adjustment for optimal reconstruction; and loop closing, searching for loops with every new keyframe. The system maintains a covisibility graph which stores covisibility information between keyframes as an undirected weighted graph where each node is a keyframe and an edge exists if the two connected keyframes share observations of same map points (over 15). The essential graph is a novelty of ORB-SLAM in that during pose graph optimization, in order not to include all edges in the dense covisibility graph, the system builds the essential graph with less edges and still maintains high accuracy. The system embeds a bag of words place recognition module for loop detection and relocalization and when building the

The authors are with Carnegie Mellon University. {yiboc, yijunl, tianyuw2}@andrew.cmu.edu

database, groups keyframes connected in the covisibility graph.

2) ORB-SLAM2: ORB-SLAM2 [2] as an extension of monocular-version ORB-SLAM, is a complete simultaneous localization and mapping system for monocular, stereo and RGB-D cameras, including relocalization, loop closing, and map reusing in real time on standard CPUs in a wide variety of environments. The whole system contains three parallel threads: the tracking to localize the camera, local mapping to manage and optimize the local map, and loop closing to detect large loops and correct the accumulated drift. The system can do place recognition based on DBoW2 for relocalization. The system links any two keyframes observing common points and constructs a minimum spanning tree connecting all keyframes using a covisibiliy graph. For tracking, mapping, and relocalization tasks, the system uses the same ORB features, which are robust to rotation and scale and fast to extract and match.

B. DeepVO

Visual odometry (VO) is one of the most essential techniques for pose estimation and robot localisation. Conventional visual odometry methods usually consist of feature extraction, feature matching, motion estimation, local optimisation modules. However, conventional visual odometry pipelines always need to be carefully tuned for good performances in a specific environment. Wang et al. [3] proposed a novel end-to-end framework for monocular visual odometry by using deep Recurrent Convolutional Neural Networks (RCNNs). This framework can learn effective feature representation automatically and implicitly model sequential dynamics and relations. Moreover, its end-to-end manner enables it to infer poses directly from a raw RGB image sequence without adopting any module in the conventional visual odometry pipeline. Based on the KITTI visual odometry benchmark, it is verified that it can produce accurate visual odometry results with precise scales and work well in completely new scenarios.

C. Optical Flow

1) DCVP: There are some papers talking about using deep learning methods to enhance the efficiency and accuracy in the feature matching phase. Among them, this paper [4] talks about how deep learning could be used in optical flow estimation and thus speeding up the four-dimensional cost volume processing. Using a four-layer convolutional network, a nonlinear feature embedding is learnt and the image patches are transformed into a compact and discriminative feature space which is robust to geometric and radiometric distortions that are commonly encountered in optical flow estimation. After learning the feature embeddings for each pixel in the image, the optical flow cost volume is constructed by calculating the displacement between the embeddings using vector products. The semi-global matching (SGM) method is used to remove outliers and regularize the estimated flow. During the postprocessing stage of the estimated optical flow, the remaining matches are lifted to the original resolution, resulting in a semi-dense correspondence field. And inpainting and variational refinement are used to obtain a dense subpixel-resolution flow field.

2) FlowNet: Optical flow estimation needs precise per-pixel localization and requires finding correspondences between two input images. Dosovitskiy et al. [5] proposed convolutional neural networks which are capable of solving the optical flow estimation problem as a supervised learning task. Other than a standard CNN architecture, they additionally developed an architecture with a correlation layer that explicitly provides matching capabilities in an end-to-end manner. This networks predict optical flow at up to 10 image pairs per second on the full resolution, achieving state-of-the-art accuracy among real-time methods.

III. Theory and Method

ORB-SLAM uses the same features for the mapping and tracking, and place recognition, which makes the system efficient and avoids the need to interpolate the depth of the recognition features from near SLAM features. Due to the speed requirement and rotation invariance quality, ORB, which are oriented multi-scale FAST corners with a 256 bits descriptor associated are chosen to generate the features used in the system. ORB features are extremely fast to compute and match, while they have good invariance to viewpoint. ORB-SLAM creates a bag of words for place recognition, which contains offline created vocabulary extracted from a large set of images. If the images are general enough, the same vocabulary can be used for different environments getting a good performance. After matching the ORB features from two different frames, if the matches are enough, meaning that tracking is successful, the system uses a constant velocity motion model to predict the camera pose and perform a guided search of the map points observed in the last frame. If not, the system converts the frame into bag of words and queries the recognition database for keyframe candidates for global relocalization. Correspondences with ORB associated to map points in each keyframe are computed. The system then performs RANSAC for each keyframe to find a camera pose using the PnP algorithm.

DeepVO [3] is an end-to-end trainable network aiming to infer poses directly from a sequence of raw RGB images (videos). Unlike the conventional VO pipeline, DeepVO is a RCNN network. The video frames are input as raw RGB images with any preprocessing. Each two consecutive images are stacked together and fed into a convolutional network to generate high dimensional features, then the features are fed into a recurrent neural network with LSTM modules. The RNN model is used to discover some temporal dependencies between the sequence inputs. With LSTM models, long trajectory dependency could be learnt. The final output from the RNN is a sequence of pose estimation regarding to the timestamps. The convolutional network uses similar architecture as FlowNet, which is shown to have great effects on prediction the optical flow from videos. DeepVO takes advantage of this structure in hope that the CNN part will

be capable of extracting some motion information from the stacked image input. For the loss function, MSE is used in order to minimize a weighted difference from the estimation prediction and the ground truth. The DeepVO network is trained on KITTI VO benchmark with 11 fully annotated sequences.

Since the visual odometry of ORB-SLAM needs to be carefully tuned and the performance will be effected based on the quality of the feature extraction and the bag of words for place recognition, we think replacing it with a deep learning system will help to generalize to some extends. Therefore we implement the two visual odometries to compare their performance.

IV. Data and Code

A. Collected Data

For both ORB-SLAM model and DeepVO, we use the KITTI visual odometry dataset (KITTI VO benchmark) [6]. This KITTI VO benchmark has 22 sequences of images. For this benchmark, 11 sequences (Sequence 00-10) are associated with ground truth. The other 10 sequences (Sequence 11-21) are only raw sensor data. To be able to do both training and testing, we use separate the Sequence 00-10 into training and testing set, which is Sequence 00-08 are used for training and Sequence 09, 10 are used for testing. Also, the same as illustrated in DeepVO [3], Sequence 00, 02, 08 are relatively long, so the trajectories are segmented to different subsets in order to generate more data for training.

B. Implemented Code

We have re-implemented ORB-SLAM2 (follow this repo: `https://github.com/raulmur/ORB_SLAM2.git`) and DeepVO (follow this repo: `https://github.com/ChiWeiHsiao/DeepVO-pytorch.git`) basing on the theories illustrated in [2] and [3], which are also described in the above section. Then we retrain the DeepVO neural network using a pretrained model as well as fine-tuning its hyperparameters. Also, to visualize the estimated route compared with ground truth route, another python script is written.

V. Results

First, we implemented ORB-SLAM2 system. We also tested the odometry module and went through the basic algorithm of it. Figure 1 shows a screenshot of running ORB-SLAM2 algorithm.

We processed the KITTI dataset and use it for testing original ORB-SLAM2 algorithm. Figure 2 shows the results of the ORB-SLAM2 on KITTI dataset. As shown in Figure 3, for the city environment dataset, there are multiple small loop closures on the map because of the rapidly turning of the car, so that the odometry drift can be continuous corrected and the online drift of odometry could be small. However, when a car is running on a highway or other environment where does not request great turning, very limited loop closures would happen. In this case, the SLAM system will be greatly based on odometry accuracy. Figure 3 shows the

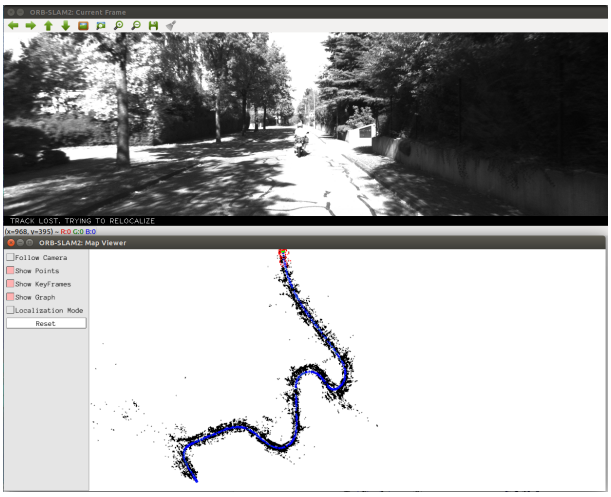

Fig. 1. Running ORB-SLAM2 algorithm on KITTI monocular dataset.

result of ORB-SLAM2 algorithm on highway environment, the drift is great here because the error of odometry is accumulated and cannot be corrected until a loop closure happens. The results above shows a shortcoming of original corner feature detector: it is not accurate for large scale environment especially where the environment texture is changing too smoothly.

Because ORB-SLAM2 [2] is an integral SLAM system, it has loop closure module which could optimize the route globally when loop closure happens. In order to make it easy for comparison, we use Sequence 09 and 10, which has no loop closure, for the visual odometry performance comparison. Figure 4 and 5 are the results of ORB-SLAM2 on Sequence 09 and 10 separately (refer to this link for the video of a running ORB-SLAM2 on KITTI Sequence 10: `https://drive.google.com/file/d/1AJ9ku5h5BaO-BgvBsVXqiiEiXPI90IAW/view?usp=sharing`). Figure 6 and 7 show the result of DeepVO on these two Sequences.

VI. Discussion

In Figure 7, it can be seen that the visual odometry performance of DeepVO [3] still can not meet the level of that in ORB-SLAM2. This is reasonable because DeepVO [3] is an end-to-end neural network, the input of this network is only raw rgb images without any other information. The network is trained to learn pose estimation itself supervisedly with no geometry constrain such as camera matrix or stereo information. So the testing results are highly depending on the environment's similarity between the training data and evaluation data. In contrast, the visual odometry in ORB-SLAM2 is based on a corner detector which is further incorporated with a bag of words model to extract features, and the extracted features are then used to do pose estimation based on camera calibration matraix which is very accurate, leading to a smaller error between neighbouring frames.

Also, the performance of DeepVO is not stable and highly depends on the training process. Figure 8 shows the result

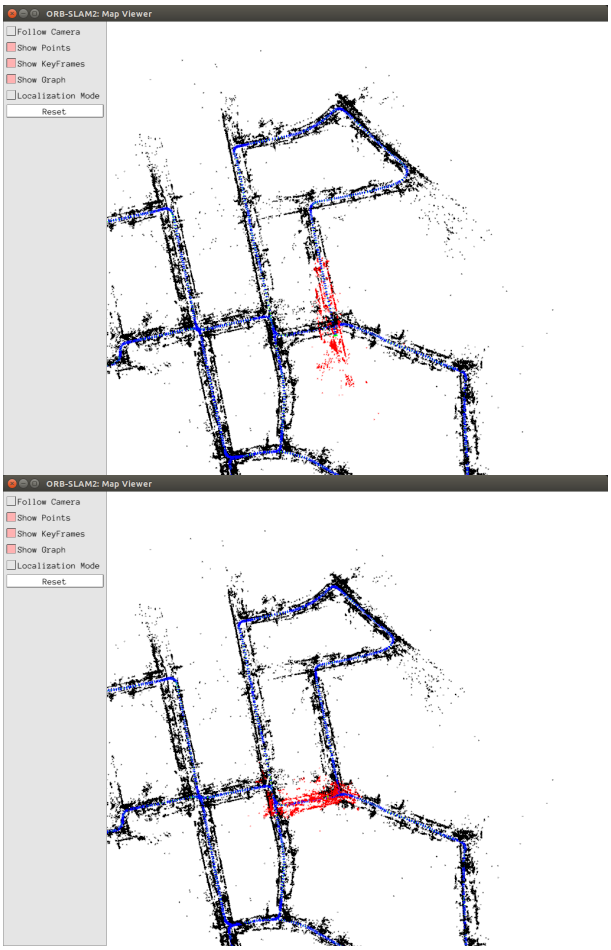

Fig. 2. ORB-SLAM2 results on city environment. The red points in the upper figure represents where loop closure is detected, and the lower figure shows the result of this local map after loop closure optimization.

Fig. 3. ORB-SLAM2 results on highway environment. The red points in the upper figure represents where loop closure is detected, and the lower figure shows the result of this local map after loop closure optimization. The blue line in the left figure demonstrates how the corrected map should be like after loop closure for comparison.

of Sequence 10 with same training hyperparameters but the training process is stopped later after the network has already been overfitting. The performance of this retrained model has a relatively better performance now, showing that DeepVO requires strictly parameter fine-tuning and traing process control.

VII. Challenge

We have successfully implemented and tested the ORB-SLAM system by November 4th. We planned to implement and train the optical flow network (DCVP) later on. However, the DCVP pipeline uses Caffe framework which is hard to config and deploy. Therefore instead we changed to the DeepVO pipeline which is based on pytorch and able to be trained. We managed to deliver the comparison results from both ORB-SLAM and DeepVO by Dec 2nd.

VIII. Conclusion

In this project we make a comparison of an end-to-end deep learning based visual odometry(DeepVO [3]) with a traditional geometry based visual odometry(visual odometry module extracted form ORB-SLAM2 [2]). DeepVO leverages the advantages of CNNs and RNNs on image processing and neighbouring information co-relating. It has the advantage of only using raw rgb images as input without any other carefully tuned parameters(such as camera calibration). While the visual odometry of ORB-SLAM2 uses corner detector to extract features for bag of words method, then it applies camera calibration and careful parameter tuning for pose estimation.

In conclusion, end-to-end deep learning methods are still too unstable to be used as odometry, so this DeepVO [3] method may only be used as a completion for traditional visual odometry. However, deep learning methods still prove its promising performance on feature extraction and object detection area, so one promising use of deep learning in SLAM system is to replace the traditional feature extracting module with neural network module, for example, using U-shape network to extract dense semantic segmentation information as features to do neighbouring feature correlating and pose estimation. Another feasible work is to fuse the

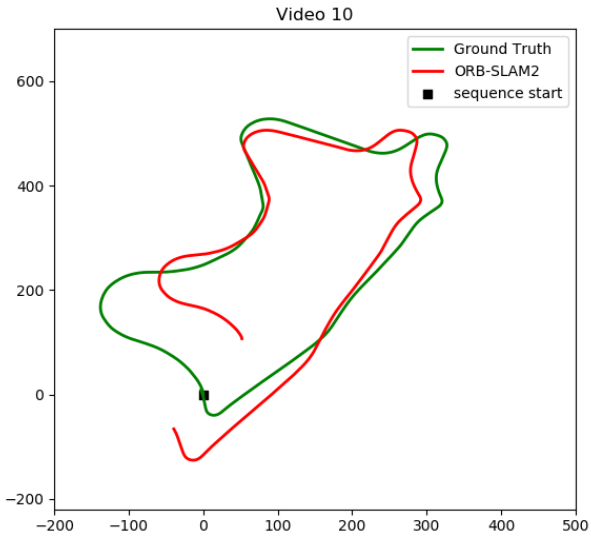

Fig. 4. The odometry result of ORB-SLAM2 on KITTI Sequence 09

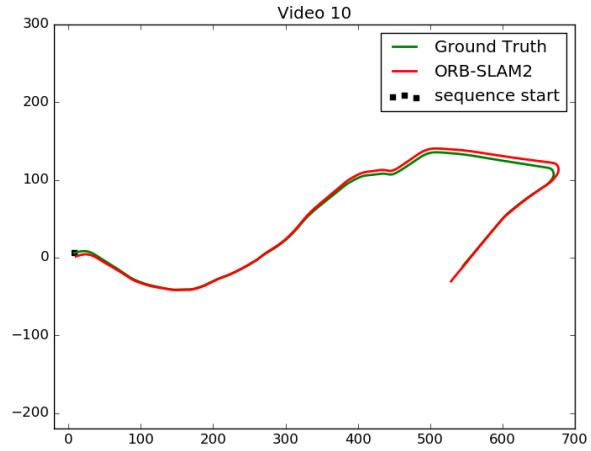

Fig. 5. The odometry result of ORB-SLAM2 on KITTI Sequence 10

accurate camera calibration information into the DeepVO [3] framework, offering a strict geometry constrain to the training process, so that in this way it could at least achieve a similar performance compared to traditional methods.

REFERENCES

[1] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[2] 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*, vol. 33, no. 5, pp. 1255–1262, 2017.

[3] S. Wang, R. Clark, H. Wen, and N. Trigoni, "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2043–2050.

[4] J. Xu, R. Ranftl, and V. Koltun, "Accurate optical flow via direct cost volume processing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1289–1297.

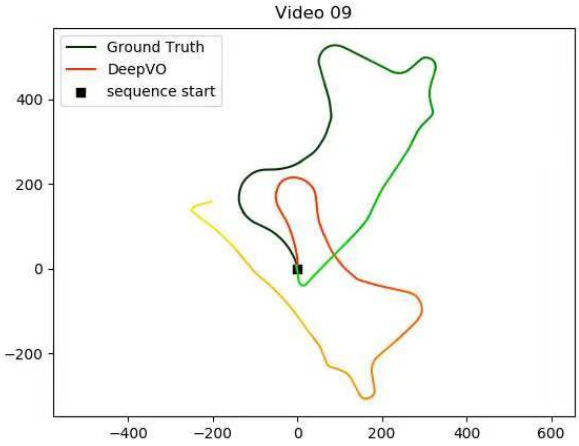

Fig. 6. The odometry result of DeepVO on KITTI Sequence 09: The result of this sequence is highly drifted.

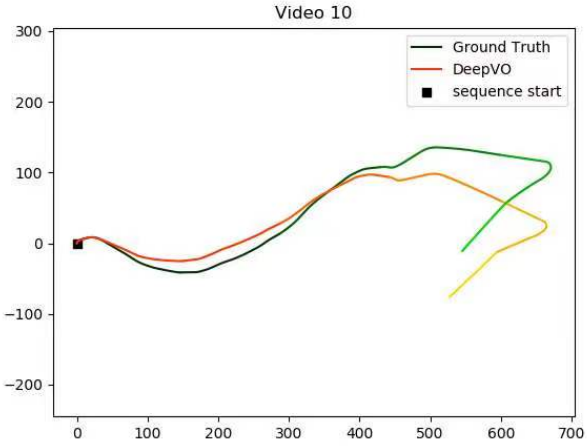

Fig. 7. The odometry result of DeepVO on KITTI Sequence 10

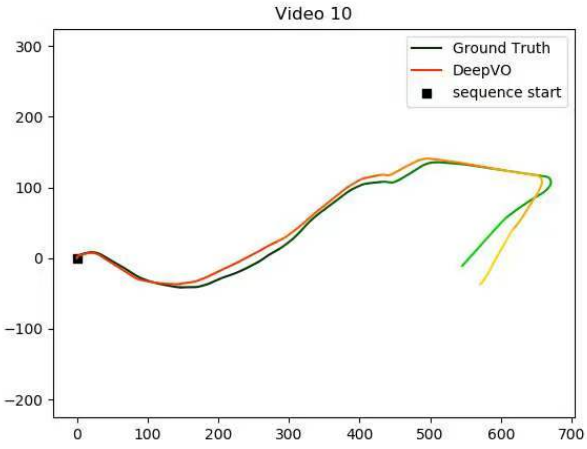

Fig. 8. The odometry result of DeepVO on KITTI Sequence 10 under later stop time.

[5] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2758–2766.

[6] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.