Beyond Tracking:
Selecting Memory and Refining Poses for Deep Visual Odometry

Fei Xue[1,3], Xin Wang[1,3], Shunkai Li[1,3], Qiuyuan Wang[1,3], Junqiu Wang[2], and Hongbin Zha[1,3]

[1]Key Laboratory of Machine Perception (MOE), School of EECS, Peking University
[2]Beijing Changcheng Aviation Measurement and Control Institute, AVIC
[3]PKU-SenseTime Machine Vision Joint Lab
{feixue, xinwang_cis, lishunkai, wangqiuyuan}@pku.edu.cn
jerywangjq@foxmail.com, zha@cis.pku.edu.cn

Abstract

Most previous learning-based visual odometry (VO) methods take VO as a pure tracking problem. In contrast, we present a VO framework by incorporating two additional components called Memory and Refining. The Memory component preserves global information by employing an adaptive and efficient selection strategy. The Refining component ameliorates previous results with the contexts stored in the Memory by adopting a spatial-temporal attention mechanism for feature distilling. Experiments on the KITTI and TUM-RGBD benchmark datasets demonstrate that our method outperforms state-of-the-art learning-based methods by a large margin and produces competitive results against classic monocular VO approaches. Especially, our model achieves outstanding performance in challenging scenarios such as texture-less regions and abrupt motions, where classic VO algorithms tend to fail.

1. Introduction

Visual Odometry (VO) and Visual Simultaneous Localization And Mapping (V-SLAM) estimate camera poses from image sequences by exploiting the consistency between consecutive frames. As an essential task in autonomous driving and robotics, VO has been studied for decades and many outstanding algorithms have been developed [7, 8, 10, 20, 30]. Recently, as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) achieve impressive performance in many computer vision tasks [4,6,12,34], a number of end-to-end models have been proposed for VO estimation. These methods either learn depth and ego-motion jointly with CNNs [16,19,36,37,39], or leverage RNNs to introduce temporal information [14,
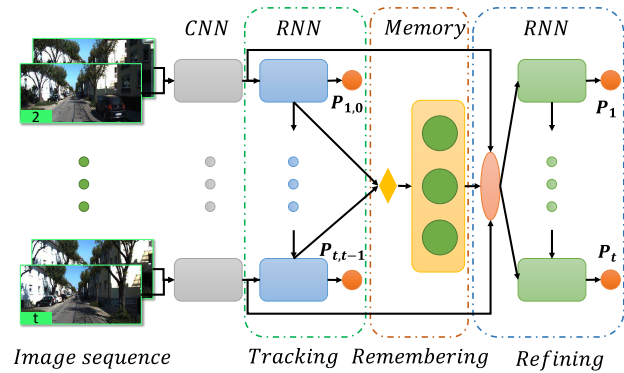

Figure 1. Overview of our framework. Compared with existing learning-based methods which formulate VO task as a pure tracking problem, we introduce two useful components called *Memory* and *Refining*. The *Memory* module preserves longer time information by adopting an adaptive context selection strategy. The *Refining* module ameliorates previous outputs by employing a spatial-temporal feature reorganization mechanism.

22, 31–33]. Due to the high dimensionality of depth maps, the number of frames is commonly limited to no more than 5. Although temporal information is aggregated through recurrent units, RNNs are incapable of remembering previous observations for long time [27], leading to the limited usage of historical information. Besides, the above methods pay little attention to the significance of incoming observations for refining previous results, which is crucial for VO tasks.

Direct estimation of camera motions from image snippets is prone to large errors due to the geometric uncertainty caused by *small baselines* (especially for handheld devices). Consequently, error accumulation is getting increasingly severe over time, as global poses are integrated from framewise poses. In classic VO/SLAM systems [20], a local map

is established according to the co-visibility graph over up to hundreds of frames, on which bundle adjustment is executed to jointly optimize all corresponding poses. Therefore, both previous and new observations are incorporated for optimization, and accumulated errors are thus alleviated.

Inspired by classic VO/SLAM systems [7, 20], we introduce an effective component, called *Memory*, which explicitly preserves the accumulated information adaptively. Owing to the high sample frequency, contents between consecutive frames are much overlapped. Rather than keeping accumulated information per time step with brute force, an intuitive and efficient strategy is utilized to reduce the redundancy. As errors of previous poses will propagate over time to current estimation, refining previous results becomes necessary. The *Memory* contains more *global* information, which can be leveraged naturally to refine previous results. Therefore, a *Refining* component is introduced. The *Refining* module takes the global pose estimation as a registration problem by aligning each view with the *Memory*. A spatial-temporal attention mechanism is applied to the contexts stored in the *Memory* for feature selection.

The overview of our framework is illustrated in Fig. 1. The encoder encodes paired images into high-level features. The *Tracking* module accepts sequential features as input, fuses current observation into accumulated information using convolution LSTMs [25] for preserving spatial connections, and produces relative poses. Hidden states of the *Tracking* RNN are adaptively preserved in the *Memory* slots. The *Refining* module ameliorates previous results using another convolutional LSTM, enabling refined results passing through recurrent units to improve the following outputs. Our contributions can be summarized as follows:

- We propose a novel end-to-end VO framework consisting of the *Tracking*, *Memory* and *Refining* components;

- An adaptive and efficient strategy is adopted for the *Memory* component to preserve accumulated information;

- A spatial-temporal attention mechanism is employed for the *Refining* component to distill valuable features.

Our method outperforms state-of-the-art learning-based methods and produces competitive results against classic algorithms. Additionally, it works well in challenging conditions where classic algorithms tend to fail due to insufficient textures or abrupt motions. The rest of this paper is organized as follows. In Sec. 2, related works on monocular odometry are discussed. In Sec. 3, our architecture is described in detail. The performance of the proposed approach is compared with current state-of-the-art methods in Sec. 4. We conclude the paper in Sec. 5.

2. Related Works

Visual odometry has been studied for decades, and many excellent approaches have been proposed. Traditionally, VO is tackled by minimizing geometric reprojection errors [10, 18, 20] or photometric errors [7, 8, 30]. These methods mostly work in regular environments, but will fail in challenging scenarios such as textureless scenes or abrupt motions. After the advent of CNNs and RNNs, the VO task has been explored with deep learning techniques. A number of approaches have been proposed to deal with the challenges in classic monocular VO/SLAM systems such as feature detection [1], depth initialization [28, 34], scale correction [35], depth representation [2] and data association [3, 17]. Despite their promising performance, they utilize the classic framework as backend, and thus cannot been deployed in an end-to-end fashion. In this paper, we mainly focus on learning-based end-to-end monocular VO works.

Unsupervised methods Mimicking the conventional structure from motion, SfmLearner [39] learns the single view depth and ego-motion from monocular image snippets using photometric errors as supervisory signals. Following the same scenario, Vid2Depth [19] adopts a differential ICP (Iterative Closest Point) loss executed on estimated 3D point clouds to enforce the consistency of predicted depth maps of two consecutive frames. GeoNet [36] estimates the depth, optical flow and ego-motion jointly from monocular views. To cope with the scale ambiguity of motions recovered from monocular image sequences, Depth-VO-Feat [37] and UnDeepVO [16] extend the work of SfmLearner to accept stereo image pairs as input and recover the absolute scale with the known baseline.

Although these unsupervised methods break the limitation of requiring massive labeled data for training, only a limited number of consecutive frames can be processed in a sequence due to the fragility of photometric losses, leading to high geometric uncertainty and severe error accumulation.

Supervised methods DeMoN [29] jointly estimates the depth and camera poses in an end-to-end fashion by formulating structure from motion as a supervised learning problem. DeepTAM [38] extends DTAM [21] via two individual subnetworks indicating *tracking* and *mapping* for the pose and depth estimation respectively. Both DeMoN and DeepTAM achieve promising results, yet require highly labeled data (depth, optical flow and camera pose) for training. MapNet [12] presents an allocentric spatial memory for localization, but only discrete directions and positions can be obtained in synthetic environments.

VO can be formulated as a sequential learning problem via RNNs. DeepVO [31] harnesses the LSTM [13] to

introduce historical knowledge for current relative motion prediction. Based on DeepVO, ESP-VO [32] infers poses and uncertainties in a unified framework. GFS-VO [33] considers the discriminability of features to different motion patterns and estimates the rotation and translation separately with a dual-branch LSTM. In addition, the ConvL-STM unit [25] is adopted to retain the spatial connections of features. There are some other works focusing on reducing localization errors by imposing constraints of relative poses [4, 14, 22].

Geometric uncertainty can be partially reduced by aggregating more temporal information using RNNs or LSTMs. Unfortunately, RNNs or LSTMs are limited for remembering long historical knowledge [27]. Here, we extend the field of view by adaptively preserving hidden states of recurrent units as memories. Therefore, previous valuable information can be inherited longer than being kept in only the single current hidden state. Besides, all these methods ignore the importance of new observations for refining previous poses, which is essential for VO tasks. By incorporating the *Refining* module, previous poses can be updated by aligning filtered features with the *Memory*. Therefore, error accumulation is further mitigated.

3. Approach

The encoder extracts high-level features from consecutive RGB images in Sec. 3.1. The *Tracking* module accepts sequential features as input, aggregates temporal information, and produces relative poses in Sec. 3.2. Hidden states of the *Tracking* RNN are adaptively selected to construct the *Memory* (Sec. 3.3) for further *Refining* previous results in Sec. 3.4. We design the loss function considering both relative and absolute pose errors in Sec. 3.5.

3.1. Encoder

We harness CNNs to encode images into high-level features. Optical flow has been proved useful for estimating frame-to-frame ego-motion by lots of current works [22,31–33,38]. We design the encoder based on the Flownet [6] which predicts optical flow between two images. The encoder retains the first 9 convolutional layers of Flownet encoding a pair of images, concatenated along RGB channels, into a 1024-channel 2D feature-map. The process can be described as:

$$X_t = \mathcal{F}(I_{t-1}, I_t) . \tag{1}$$

$X_t \in \mathbb{R}^{C \times H \times W}$ denotes the encoded feature-map at time t by function $\mathcal{F}$ from two consecutive images I_{t-1} and I_t. H, W and C represent the height, width and channel of obtained feature maps.
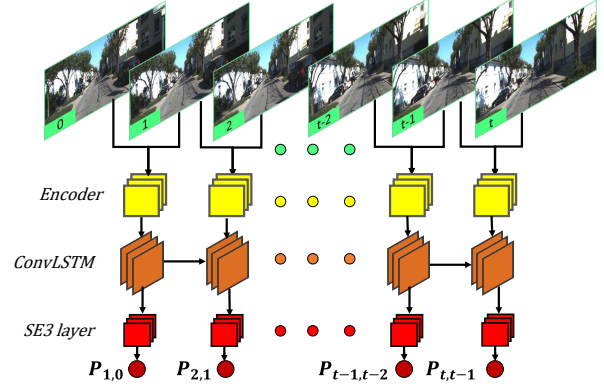

Figure 2. The *Tracking* module of our framework is implemented on a convolutional LSTM [25]. Relative camera poses are produced by the $\mathbb{SE}$ (3) layer [5] from the outputs of recurrent units. Temporal information is preserved in the hidden states.

3.2. Tracking

The *Tracking* module fuses current observations into accumulated information and calculates relative camera motions between two consecutive views as shown in Fig. 2.

Sequence modeling We adopt the prevent LSTM [13] to model the image sequence. In this case, the feature flow passing through recurrent units carries rich accumulated information of previous inputs to infer the current output. Note that the standard LSTM unit used by DeepVO [31] and ESP-VO [32] requires 1D vector as input in which the spatial structure of features is ignored. The ConvLSTM unit [25], an extension of LSTM with convolution underneath, is adopted in the *Tracking* RNN for preserving the spatial formulation of visual cues and expanding the capacity of recurrent units for remembering more knowledge. The recurrent process can be controlled by

$$O_t, H_t = \mathcal{U}(X_t, H_{t-1}) . \tag{2}$$

O_t denotes the output at time t. H_t and H_{t-1} are the hidden states at current and the last time step.

Relative pose estimation Relative motions can be directly recovered from paired images. Unfortunately, direct estimation is prone to error accumulation due to the geometric uncertainty brought by short baselines. The problem can be mitigated by introducing more historical information. Inheriting accumulated knowledge, the output of recurrent unit at each time step is naturally used for pose estimation. The $\mathbb{SE}$ (3) [5] layer generates the 6-DoF motion $P_{t,t-1}$ from the output at time t.

Theoretically, the global pose of each view can be recovered by integrating predicted relative poses as $P_t = \prod_{i=1}^{t} P_{i,i-1} P_0$ (P_0 denotes the origin pose of the world coordinate) just as DeepVO [31] and ESP-VO [32]. The accumulated error, however, will get increasingly severe, and
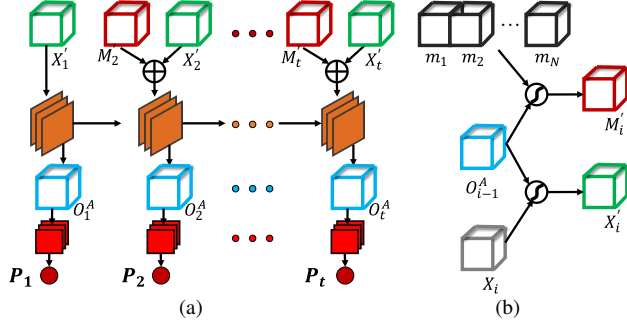
Figure 3. (a) The *Refining* module aligns current observation with the contexts stored in the *Memory* module for absolute pose estimation. (b) Both contexts and the current observation are re-organized utilizing the last output as guidance.
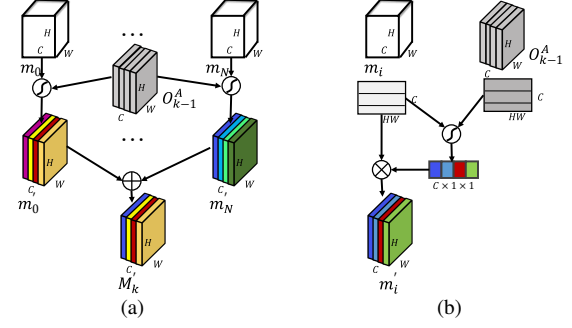

Figure 4. Extracting features from *Memory* using the last output as guidance. We consider the correlation of both each context stored in the *Memory* in (a) and every channel of the context in (b).

thus degrades the performance of the entire system. Due to the lack of explicit geometric representation of the 3D environments, neural networks, however, are incapable of building a global map to assist tracking. Fortunately, the temporal information is recorded in the hidden states of recurrent units. Although the information is short-time, these hidden states at different time points can be gathered and re-organized as parts of an *implicit* map (discussed in Sec. 3.3).

3.3. Remembering

The *Memory* module is a neural analogue of the *local map* commonly used in classic VO/SLAM systems [20]. Considering the LSTM cannot remember information for long time [27], we explicitly store hidden states of recurrent units at different time points to extend the time span.

A vanilla choice is to take each time step into account via storing all hidden states over the whole sequence as $M = \{m_1, m_2, ..., m_{N-1}, m_N\}$, where m_i denotes the ith hidden state in the sequence, and N is the size of the memory buffer. Since contents of two consecutive images are much overlapped, it's redundant to remember each hidden state. Instead, only key states are selected. As the difference between two frames coincides with the poses, we utilize the motion distance as a metric to decide if current hidden state should be stored.

Specifically, the current hidden state would not be put into the *Memory*, unless the parallax between the current and the latest view in the slot is large enough. Here, the rotational and translational distances are utilized:

$$||Rot_{m_i} - Rot_{m_{i-1}}||_2 \geq \theta_{Rot} , \qquad (3)$$

$$||Trans_{m_i} - Trans_{m_{i-1}}||_2 \geq \theta_{Trans} . \qquad (4)$$

This strategy guarantees both the co-visibility of different views and the existence of global information. As both previous and new observations are gathered, the *Memory* can be used to optimize previous poses.

3.4. Refining

Once the *Memory* is constructed, the *Refining* module estimates the absolute pose of each view by aligning corresponding observation with the *Memory*, as shown in Fig. 3. We adopt another recurrent branch using ConvLSTM, enabling previously refined outputs passing through recurrent units to improve the next estimation, as:

$$O_t^A, H_t^A = \mathcal{U}^A(X_t^A, H_{t-1}^A) . \qquad (5)$$

X_t^A, O_t^A and H_t^A are the input, output and hidden state at time t. H_{t-1}^A denotes the hidden state at time $t-1$. The $\mathcal{U}^A$ indicates the recurrent branch for the *Absolute* pose estimation. All these variables are 3D tensors to be discussed in the following sections.

Spatial-temporal attention Although all observations are fused and distributed in N hidden states, each hidden state stored in the *Memory* contributes discriminatively to different views. In order to distinguish related information, an attention mechanism is adopted. We utilize the last output O_{t-1}^A as guidance, since motions between two consecutive views in a sequence are very small.

In specific, we generate selected memories $M_t^{'}$ for current view t with the function $\mathcal{G}$ as:

$$M_t^{'} = \mathcal{G}(O_{t-1}^A, M) . \qquad (6)$$

The temporal attention aims to re-weight elements in the *Memory* considering the contribution of each m_i to the pose estimation of specific views. Therefore, as shown in Fig. 4(a), $M_t^{'}$ can be defined as the linear averaging of all elements in M as $M_t^{'} = \sum_{i=1}^{N} \alpha_i m_i$. The $\alpha_i = \frac{\exp(w_i)}{\sum_{k=1}^{N} \exp(w_i)}$ denotes the normalized weight. The $w_i = S(O_{t-1}^A, m_i)$ is the weight computed according to the cosine similarity function denoted as S.

As all elements in the *Memory* are formulated as 3D tensors, spatial connections are retained. In this framework, we focus on not only which element in the *Memory* plays a

more important role but also where each element influences the final results more significantly. We try to find corresponding co-visible contents at the feature level. Hence, we extend the attention mechanism from the temporal domain to the spatial-temporal domain incorporating an additional channel favored feature attention mechanism. Feature-map of each channel is taken as a unit and re-weighted for each view according to the last output. As shown in Fig. 4(b), the process is described as:

$$M_t^{'} = \sum_{i=1}^{N} \alpha_i \mathcal{C}(\beta_{i1}m_{i1}, \beta_{i2}m_{i2}, ..., \beta_{iC}m_{iC}). \quad (7)$$

The $m_{ij} \in \mathbb{R}^{H \times W}$ denotes the jth channel of the ith element in the *Memory*. The β_{ij} is the normalized weight defined on the correlation between the jth channel of O_{t-1} and m_i. $\mathcal{C}$ concatenates all reweighted feature maps along the channel dimension. We calculate the cosine similarity between two vectorized feature-maps to assign the correlation weights.

Absolute pose estimation The guidance is also executed on the observations encoded as high-level features to distill related visual cues, denoted as $X_t^{'}$. Both reorganized memories and observations are stacked along channels and passed through two convolutional layers with kernel size of 3 for fusion. The fused feature denoted as X_t^A is the final input to be fed into convolutional recurrent units. Then the $\mathbb{SE}$ (3) layer calculates the absolute pose from the output O_t^A. Note that, through recurrent units, the hidden state propagating refined results to next time point further improves the following prediction.

3.5. Loss Function

Our model learns relative and absolute poses in the *Tracking* and *Refining* modules separately. Therefore, consisting of both relative and absolute pose errors, the loss functions are defined as:

$$\mathcal{L}_{local} = \frac{1}{t} \sum_{i=1}^{t} ||\hat{\boldsymbol{p}}_{i-1,i} - \boldsymbol{p}_{i-1,i}||_2 + k||\hat{\boldsymbol{\phi}}_{i-1,i} - \boldsymbol{\phi}_{i-1,i}||_2,$$
$$(8)$$

$$\mathcal{L}_{global} = \sum_{i=1}^{t} \frac{1}{i}(||\hat{\boldsymbol{p}}_{0,i} - \boldsymbol{p}_{0,i}||_2 + k||\hat{\boldsymbol{\phi}}_{0,i} - \boldsymbol{\phi}_{0,i}||_2), \quad (9)$$

$$\mathcal{L}_{total} = \mathcal{L}_{local} + \mathcal{L}_{global}, \quad (10)$$

where $\hat{\boldsymbol{p}}_{i-1,i}, \boldsymbol{p}_{i-1,i}, \hat{\boldsymbol{\phi}}_{i-1,i}$, and $\boldsymbol{\phi}_{i-1,i}$ respectively represent the predicted and ground-truth relative translations and rotations in three directions; $\hat{\boldsymbol{p}}_{0,i}, \boldsymbol{p}_{0,i}, \hat{\boldsymbol{\phi}}_{0,i}$, and $\boldsymbol{\phi}_{0,i}$ represent the predicted and ground-truth absolute translations and rotations. $\mathcal{L}_{local}, \mathcal{L}_{global}$ and $\mathcal{L}_{total}$ denote the local, global, and total losses respectively. t is the current frame index in a sequence. k is a fixed parameter for balancing the rotational and translational errors.

4. Experiments

We first discuss the implementation details of our framework in Sec. 4.1. Next, we compare our method with state-of-the-art approaches on the KITTI [9] and TUM-RGBD [26] datasets in Sec. 4.2 and Sec. 4.3, respectively. Finally, an ablation study is performed in Sec. 4.4.

4.1. Implementation

Training Our network takes monocular RGB image sequences as input. The image size can be arbitrary because our model has no requirement of compressing features into vectors as DeepVO [31] and ESP-VO [32]. We use 11 consecutive images to construct a sequence, yet our model can accept dynamic lengths of inputs. The parameter k is set to 100 and 1 for the KITTI and TUM-RGBD dataset. The θ_{Rot} and θ_{Trans} are set to 0.005 (rad) and 0.6 (m) for the KITTI dataset. While for the TUM-RGBD dataset, the values are 0.01 (rad) and 0.01 (m). The buffer size N is initialized with the sequence length, yet the buffer can be used without being fully occupied.

Network The encoder is pretrained on the FlyingChairs dataset [6], while other parts of the network are initialized with MSRA [11]. Our model is implemented by PyTorch [23] on an NVIDIA 1080Ti GPU. Adam [15] with $\beta_1 = 0.9, \beta_2 = 0.99$ is used as the optimizer. The network is trained with a batch size of 4, a weight decay of 4×10^{-4} for 150,000 iterations in total. The initial learning rate is set to 10^{-4} and reduced by half every 60,000 iterations.

4.2. Results on the KITTI Dataset

The KITTI dataset [9], one of the most influential outdoor VO/SLAM benchmark datasets, is widely used in both classic [10, 20] and learning-based works [16, 19, 31, 32, 36, 37, 39]. It consists of 22 sequences captured in urban and highway environments at a relatively low sample frequency (10 fps) at the speed up to 90km/h. Seq 00-10 provide raw data with ground-truth represented as 6-DoF motion parameters considering the complicated urban environments, while Seq 11-21 provide only raw data. In our experiments, the left RGB images are resized to 1280 x 384 for training and testing. We adopt the same train/test split as DeepVO [31] and GFS-VO [33] by using Seq 00, 02, 08, 09 for training and Seq 03, 04, 05, 06, 07, 10 for evaluation.

Baseline methods The learning-based baselines include supervised approaches such as DeepVO [31], ESP-VO [32], GFS-VO [33], and unsupervised approaches such as SfmLearner [39], Depth-VO-Feat [37], GeoNet [36], Vid2Depth [19] and UndeepVO [16]. Monocular VISO2 [10] (VISO2-M) and ORB-SLAM2 [20] are used as classic baselines. The error metrics, i.e., averaged Root Mean Square Errors (RMSE) of the translational and rotational errors, are adopted for all the test sequences of the lengths ranging from 100, 200 to 800 meters.

Method	Sequence													
	03		04		05		06		07		10		Avg	
	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}
UnDeepVO [16]	5.00	6.17	5.49	2.13	3.40	1.50	6.20	1.98	3.15	2.48	10.63	4.65	5.65	3.15
Depth-VO-Feat [37]	15.58	10.69	2.92	2.06	4.94	2.35	5.80	2.07	6.48	3.60	12.45	3.46	7.98	4.04
GeoNet [36]	19.21	9.78	9.09	7.54	20.12	7.67	9.28	4.34	8.27	5.93	20.73	9.04	13.12	7.38
Vid2Depth [19]	27.02	10.39	18.92	**1.19**	51.13	21.86	58.07	26.83	51.22	36.64	21.54	12.54	37.98	18.24
SfmLearner [39]	10.78	3.92	4.49	5.24	18.67	4.10	25.88	4.80	21.33	6.65	14.33	3.30	15.91	4.67
DeepVO [31]	8.49	6.89	7.19	6.97	2.62	3.61	5.42	5.82	3.91	4.60	8.11	8.83	5.96	6.12
ESP-VO [32]	6.72	6.46	6.33	6.08	3.35	4.93	7.24	7.29	3.52	5.02	9.77	10.2	6.15	6.66
GFS-VO-RNN [33]	6.36	3.62	5.95	2.36	5.85	2.55	14.58	4.98	5.88	2.64	7.44	3.19	7.68	3.22
GFS-VO [33]	5.44	3.32	**2.91**	1.30	3.27	1.62	8.50	2.74	3.37	2.25	6.32	2.33	4.97	2.26
Ours	**3.32**	**2.10**	2.96	1.76	**2.59**	**1.25**	**4.93**	**1.90**	**3.07**	**1.76**	**3.94**	**1.72**	**3.47**	**1.75**

t_{rel} : average translational RMSE drift (%) on length from 100, 200 to 800 m.

r_{rel} : average rotational RMSE drift (°/100m) on length from 100, 200 to 800 m.

Table 1. Results on the KITTI dataset. DeepVO [31], ESP-VO [32], GFS-VO [33] and our model are supervised methods trained on Seq 00, 02, 08 and 09. SfmLearner [39], GeoNet [36], Vid2Depth [19], Depth-VO-Feat [37], and UndeepVO [16] are trained on Seq 00-08 in an unsupervised manner. The results of SfmLearner and UnDeepVO are from [34], while for GeoNet, Vid2Depth and Depth-VO-Feat, the poses are recovered from officially released pretrained models. The best results are highlighted.
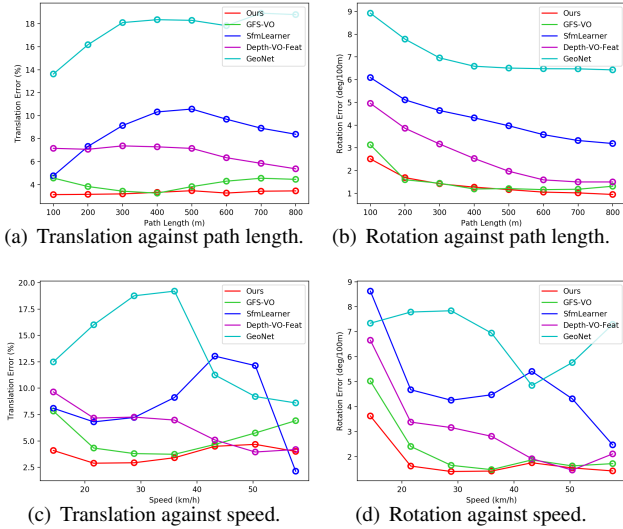

(a) Translation against path length.

(b) Rotation against path length.

(c) Translation against speed.

(d) Rotation against speed.

Figure 5. Average errors on translation and rotation against different path lengths and speeds.

Comparison with learning-based methods As shown in Table 1, our method outperforms DeepVO [31], ESP-VO [32] and GFS-VO-RNN [33] (without motion decoupling) on all of the test sequences by a large margin. Since DeepVO, ESP-VO and GFS-VO only consider historical knowledge stored in a single hidden state, error accumulates severely. The problem is partially mitigated by considering the discriminative ability of features to different motion patterns in GFS-VO, while our method is more effective.

Meanwhile, we provide the results of unsupervised approaches in Table 1. As monocular VO methods including SfmLearner [39], GeoNet [36], Vid2Depth [19] suffer from scale ambiguity, frame-to-frame motions of short sequence snippets are aligned individually with ground-truths to fix scales. Although they achieve promising performance on sequences consisting of 5 (SfmLearner, GeoNet) or 3 (Vid2Depth) frames, they suffer from heavy error accumulation when integrating poses over the entire sequence. Benefited from stereo images in scale recovery, UnDeepVO [16] and Depth-VO-Feat [37] obtain competitive results against DeepVO, ESP-VO, and GFS-VO, while our results are still much better. Note that only monocular images are used in our model.

We further evaluate the average rotation and translation errors for different path lengths and speeds in Fig. 5. The accumulated errors over long path lengths are effectively mitigated by our method owing to the new information for refining previous results. Moreover, this advantage of our algorithm can also be seen in handling high speed situations. GFS-VO [33] also achieves promising rotation estimation by decoupling the motions. Unfortunately, it does not provide robust translation results.

Comparison with classic methods The results of VISO2-M [10], ORB-SLAM2 [20] (with and without loop closure), and our method are shown in Table 2. VISO2-M is a pure monocular VO algorithm recovering framewise poses. ORB-SLAM2, however, is a strong baseline, because both versions utilize local bundle adjustment for jointly optimizing poses and a global map. Our model outperforms VISO2-M consistently. ORB-SLAM2 [20] achieves superior performance in terms of rotation estimation owing to the global explicit geometric constraints. However, it suffers more from error accumulation in translation on long sequences (Seq 05, 06, 07) than our approach,

Seq	Ours		VISO2-M [10]		ORB-SLAM2 [20]		ORB-SLAM2 (LC) [20]	
	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}
03	3.32	2.10	8.47	8.82	2.28	0.40	2.17	0.39
04	2.96	1.76	4.69	4.49	1.41	0.14	1.07	0.17
05	2.59	1.25	19.22	17.58	13.21	0.22	1.86	0.24
06	4.93	1.90	7.30	6.14	18.68	0.26	4.96	0.18
07	3.07	1.76	23.61	19.11	10.96	0.37	1.87	0.39
10	3.94	1.72	41.56	32.99	3.71	0.30	3.76	0.29
Avg	3.47	1.75	17.48	16.52	8.38	0.28	2.62	0.28

Table 2. Results of VISO2-M [10], ORB-SLAM2 (with and without loop closure) [20] and our method on the KITTI dataset.
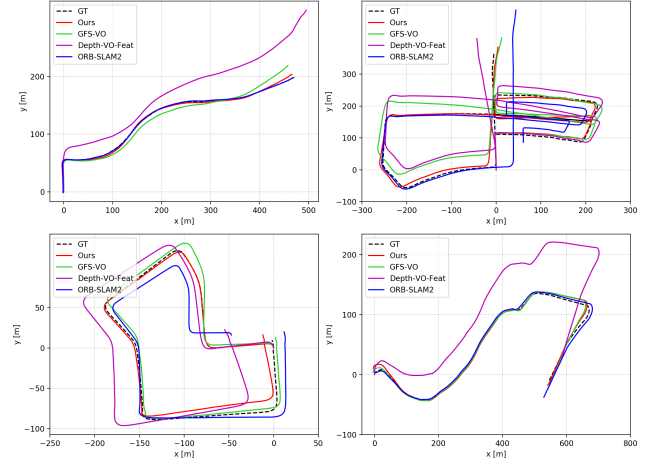

Figure 6. The trajectories of ground-truth, ORB-SLAM2 [20], Depth-VO-Feat [37], GFS-VO [33] and our model on Seq 03, 05, 07 and 10 (from left to right) of the KITTI benchmark.

which is reduced by global bundle adjustment. While for short sequences (Seq 03, 04, 10), performances of the two versions and our method are very close. The small differences between the results of ORB-SLAM2 with loop close and our method suggest that global information is retained and effectively used by our novel framework.

A visualization of the trajectories estimated by Depth-VO-Feat, GFS-VO, ORB-SLAM2 and our method is illustrated in Fig. 6. Depth-VO-Feat suffers from sever error accumulation though trained on stereo images. GFS-VO and ORB-SLAM2 produces close results with our model in simple environments (Seq 03, 10), while our method outperforms them in complicated scenes (Seq 05, 07).

4.3. Results on the TUM-RGBD Dataset

We test the generalization ability of our model on the TUM-RGBD dataset [26], a prevalent public benchmark used by a number of VO/SLAM algorithms [8, 20, 38]. The dataset was collected by handheld cameras in indoor environments with various conditions including dynamic objects, textureless regions and abrupt motions. The dataset provides both color and depth images, while only the monocular RGB images are used in our experiments. Different from datasets captured by moving cars, motions in this benchmark contain complicated patterns due to the handheld capture mode. We select some sequences for training and others for testing (The details can be found in the supplementary material), and evaluate the performance in both regular and challenging conditions using the averaged Absolute Trajectory Errors (ATE).

Comparison with classic methods Since few monocular learning-based VO algorithms have attempted to handle complicated motions recorded by handheld cameras, we alternatively compare our approach against current state-of-the-art classic methods including ORB-SLAM2 [20] and DSO [7]. As shown in Table 3, they yield promising results on scenes with rich textures (fr2/desk, fr2/360_kidnap, fr3/sitting_static, fr3/nstr_tex_near_loop, fr3/str_tex_far), yet

our results are comparable.

As ORB-SLAM2 [20] relies on ORB [24] features to establish correspondences, it fails in scenes without rich textures (fr3/nstr_ntex_near_loop, fr3/str_ntex_far, fr2/large_cabinet). Utilizing pixels with large gradients for tracking, DSO [7] works well in scenes with structures or edges (fr3/str_ntex_far, fr3/str_tex_far). It cannot achieve good performance when textures are insufficient. Both ORB-SLAM2 and DSO can hardly work in scenes without texture and structure (fr2/large_cabinet, fr3/nstr_ntex_near_loop) and tend to fail when facing abrupt motions (fr2_pioneer_360, fr2/_pioneer_slam3). In contrast, our method is capable of dealing with these challenges owing to the ability of deep learning in extracting high-level features, and the efficacy of our proposal for error reduction. A visualization of trajectories is shown in Fig. 7.

4.4. Ablation Study

Table 3 also shows an ablation study illustrating the importance of each component in our framework. The baseline is our model removing the *Memory* and *Refining* modules, similar to [31–33]. The *Tracking* model works poorly in both regular and challenging conditions, because historical knowledge in a single hidden state is inefficient to reduce accumulated errors. Fortunately, the *Memory* component mitigates the problem by explicitly introducing more global information and considerably improves results of the *Tracking* model both on regular and challenging sequences.

We further test the spatial-temporal attention strategy adopted for selecting features from memories and observations by removing the *temporal attention* and *spatial attention* progressively. We observe that both of the two attention techniques are crucial to improve the re-
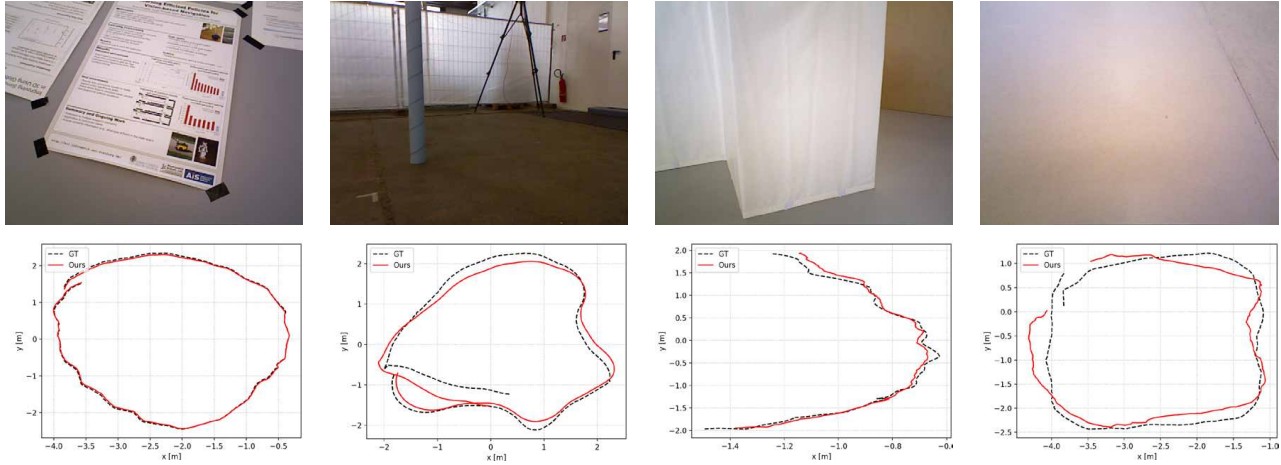
Figure 7. The raw images (top) and trajectories (bottom) recovered by our method on the TUM-RGBD dataset [26] (from left to right: fr3/str_tex_far, fr2/poineer_360, fr3/str_ntex_far, fr3/nstr_ntex_near_loop). Trajectories are aligned with ground-truths for scale recovery.

Sequence	Desc. str/tex/abrupt motion	Frames	ORB-SLAM2 [20]	DSO [7]	Ours (tracking)	Ours (w/o temp atten)	Ours (w/o spat atten)	Ours
fr2/desk	Y/Y/N	2965	**0.041**	X	0.183	0.164	0.159	0.153
fr2/360_kidnap	Y/Y/N	1431	**0.184**	0.197	0.313	0.225	0.224	0.208
fr2/pioneer_360	Y/Y/Y	1225	X	X	0.241	0.1338	0.076	**0.056**
fr2/pioneer_slam3	Y/Y/Y	2544	X	0.737	0.149	0.1065	0.085	**0.070**
fr2/large_cabinet	Y/N/N	1011	X	X	0.193	0.193	0.177	**0.172**
fr3/sitting_static	Y/Y/N	707	X	0.082	0.017	0.018	0.017	**0.015**
fr3/nstr_ntex_near_loop	N/N/N	1125	X	X	0.371	0.195	0.157	**0.123**
fr3/nstr_tex_near_loop	N/Y/N	1682	0.057	0.093	0.046	0.011	0.010	**0.007**
fr3/str_ntex_far	Y/N/N	814	X	0.543	0.069	0.047	0.039	**0.035**
fr3/str_tex_far	Y/Y/N	938	**0.018**	0.040	0.080	0.049	0.046	0.042

Table 3. Evaluation on the TUM-RGBD dataset [26]. The values describe the translational RMSE in [m/s]. Results of ORB-SLAM2 [20] and DSO [7] are generated from the officially released source code with recommended parameters. **Ours (tracking)** is a network which contains only the tracking component. **Ours (w/o temp atten)** indicates the model averaging the all memories as input without temporal attention. **Ours (w/o spat atten)** is the model removing the spatial attention yet retaining the temporal attention.

sults, especially in challenging conditions (fr2/pioneer_360, fr2/pioneer_slam3, fr3/nstr_ntex_near_loop).

5. Conclusion

In this paper, we present a novel framework for learning monocular visual odometry in an end-to-end fashion. In the framework, we incorporate two helpful components called *Memory* and *Refining*, which focus on introducing more global information and ameliorating previous results with these information respectively. We utilize an adaptive and efficient selection strategy to construct the *Memory*. Besides, a spatial-temporal attention mechanism is employed for feature selection when recovering the absolute poses in the *Refining* module. The refined results propagating information through recurrent units, further improve the following estimation. Experiments demonstrate that our

model outperforms previous learning-based monocular VO methods and gives competitive results against classic VO approaches on the KITTI and TUM-RGBD benchmarks respectively. Moreover, our model obtains outstanding results under challenging conditions including texture-less regions and abrupt motions, where classic methods tend to fail.

In the future, we consider to extend the work to a full SLAM system consisting tracking, mapping and global optimization. Moreover, auxiliary information, such as IMU and GPS data will also be introduced to enhance the system.

Acknowledgement

References

[1] P. Agrawal, J. Carreira, and J. Malik. Learning to See by Moving. In *ICCV*, 2015. 2

[2] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison. CodeSLAM-Learning a Compact, Optimisable Representation for Dense Visual SLAM. In *CVPR*, 2018. 2

[3] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas. Probabilistic Data Association for Semantic SLAM. In *ICRA*, 2017. 2

[4] S. Brahmbhatt, J. Gu, K. Kim, J. Hays, and J. Kautz. MapNet: Geometry-aware Learning of Maps for Camera Localization. In *CVPR*, 2018. 1, 3

[5] R. Clark, S. Wang, A. Markham, N. Trigoni, and H. Wen. VidLoc: A Deep Spatio-temporal Model for 6-DoF Videoclip Relocalization. In *CVPR*, 2017. 3

[6] 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 *ICCV*, 2015. 1, 3, 5

[7] J. Engel, V. Koltun, and D. Cremers. Direct Sparse Odometry. *TPAMI*, 2018. 1, 2, 7, 8

[8] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale Direct Monocular SLAM. In *ECCV*, 2014. 1, 2, 7

[9] A. Geiger, P. Lenz, and R. Urtasun. Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *CVPR*, 2012. 5

[10] A. Geiger, J. Ziegler, and C. Stiller. Stereoscan: Dense 3D Reconstruction in Real-time. In *IV*, 2011. 1, 2, 5, 6, 7

[11] K. He, X. Zhang, S. Ren, and J. Sun. Delving Deep into Rectifiers: Surpassing Human-level Performance on Imagenet Classification. In *ICCV*, 2015. 5

[12] J. F. Henriques and A. Vedaldi. MapNet: An Allocentric Spatial Memory for Mapping Environments. In *CVPR*, 2018. 1, 2

[13] S. Hochreiter and J. Schmidhuber. Long Short-term Memory. *Neural Computation*, 1997. 2, 3

[14] G. Iyer, J. K. Murthy, K. Gunshi Gupta, and L. Paull. Geometric Consistency for Self-supervised End-to-end Visual Odometry. In *CVPR Workshops*, 2018. 1, 3

[15] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015. 5

[16] R. Li, S. Wang, Z. Long, and D. Gu. UnDeepVO: Monocular Visual Odometry through Unsupervised Deep Learning. In *ICRA*, 2018. 1, 2, 5, 6

[17] K.-N. Lianos, J. L. Schönberger, M. Pollefeys, and T. Sattler. VSO: Visual Semantic Odometry. In *ECCV*, 2018. 2

[18] H. Liu, M. Chen, G. Zhang, H. Bao, and Y. Bao. ICE-BA: Incremental, Consistent and Efficient Bundle Adjustment for Visual-Inertial SLAM. In *CVPR*, 2018. 2

[19] R. Mahjourian, M. Wicke, and A. Angelova. Unsupervised Learning of Depth and Ego-motion from Monocular Video Using 3D Geometric Constraints. In *CVPR*, 2018. 1, 2, 5, 6

[20] R. Mur-Artal and J. D. Tardós. ORB-SLAM2: An Open-source SLAM System for Monocular, Stereo, and RGB-D Cameras. *T-RO*, 2017. 1, 2, 4, 5, 6, 7, 8

[21] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense Tracking and Mapping in Real-time. In *ICCV*, 2011. 2

[22] E. Parisotto, D. Singh Chaplot, J. Zhang, and R. Salakhutdinov. Global Pose Estimation with an Attention-based Recurrent Network. In *CVPR Workshops*, 2018. 1, 3

[23] A. Paszke, S. Gross, S. Chintala, and G. Chanan. Pytorch. https://github.com/pytorch/pytorch, 2017. 5

[24] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An Efficient Alternative to SIFT or SURF. In *ICCV*, 2011. 7

[25] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *NIPS*, 2015. 2, 3

[26] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A Benchmark for the Evaluation of RGB-D SLAM Systems. In *IROS*, 2012. 5, 7, 8

[27] S. Sukhbaatar, a. szlam, J. Weston, and R. Fergus. End-to-end Memory Networks. In *NIPS*, 2015. 1, 3, 4

[28] K. Tateno, F. Tombari, I. Laina, and N. Navab. CNN-SLAM: Real-time Dense Monocular SLAM with Learned Depth Prediction. In *CVPR*, 2017. 2

[29] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. DeMoN: Depth and Motion Network for Learning Monocular Stereo. In *CVPR*, 2017. 2

[30] R. Wang, M. Schworer, and D. Cremers. Stereo DSO: Large-scale Direct Sparse Visual Odometry with Stereo Cameras. In *ICCV*, 2017. 1, 2

[31] S. Wang, R. Clark, H. Wen, and N. Trigoni. DeepVO: Towards End-to-end Visual Odometry with Deep Recurrent Convolutional Neural Networks. In *ICRA*, 2017. 1, 2, 3, 5, 6, 7

[32] S. Wang, R. Clark, H. Wen, and N. Trigoni. End-to-end, Sequence-to-sequence Probabilistic Visual Odometry through Deep Neural Networks. *IJRR*, 2018. 1, 3, 5, 6, 7

[33] F. Xue, Q. Wang, X. Wang, W. Dong, J. Wang, and H. Zha. Guided Feature Selection for Deep Visual Odometry. In *ACCV*, 2018. 1, 3, 5, 6, 7

[34] N. Yang, R. Wang, J. Stückler, and D. Cremers. Deep Virtual Stereo Odometry: Leveraging Deep Depth Prediction for Monocular Direct Sparse Odometry. In *ECCV*, 2018. 1, 2, 6

[35] X. Yin, X. Wang, X. Du, and Q. Chen. Scale Recovery for Monocular Visual Odometry Using Depth Estimated with Deep Convolutional Neural Fields. In *ICCV*, 2017. 2

[36] Z. Yin and J. Shi. GeoNet: Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose. In *CVPR*, 2018. 1, 2, 5, 6

[37] H. Zhan, R. Garg, C. Saroj Weerasekera, K. Li, H. Agarwal, and I. Reid. Unsupervised Learning of Monocular Depth Estimation and Visual Odometry with Deep Feature Reconstruction. In *CVPR*, 2018. 1, 2, 5, 6, 7

[38] H. Zhou, B. Ummenhofer, and T. Brox. DeepTAM: Deep Tracking and Mapping. In *ECCV*, 2018. 2, 3, 7

[39] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised Learning of Depth and Ego-motion from Video. In *CVPR*, 2017. 1, 2, 5, 6