Online Adaptive Keypoint Extraction for Visual Odometry Across Different Scenes

Ruitao Zhang, Yafei Wang, *Member, IEEE*, Zexing Li, *Graduate Student Member, IEEE*, Fei Ding, Chongfeng Wei, *Senior Member, IEEE*, and Mingyu Wu, *Member, IEEE*

***Abstract*—Visualodometry needs to be robust against various environmental changes. Although Deep learning (DL) based methods can bring more robust features to visual odometry (VO) than traditional methods, the gap between training and test dataset restricts the performance of DL-based methods when encountering novel scenes. Additionally, due to non-differentiability of visual odometry optimization process, the scene independent geometric constraints of images is hardly used in DL-based methods, which further decreases the accuracy and generalization of odometry. In this letter, we propose a reinforcement learning based framework incorporating the geometric constraints to address the challenge of non-differentiability. Unlike conventional DL-based methods, our VO estimates the pose through traditional direct odometry. To facilitate online training, a weighted direct graph structure is utilized to efficiently organize and simplify local images. This online training scheme enables the keypoint extraction policy to adapt dynamically to the current scene of odometry with less training data. Evaluations against the state-of-the-art visual odometry were performed on the KITTI Odometry, EuRoC MAV and Tum RGBD dataset. The results demonstrate that our method outperform popular approaches in terms of tracking accuracy and generalization ability.**

***Index Terms*—Field robotics, reinforcement learning, SLAM.**

I. Introduction

VISUAL odometry provides an estimation of camera poses through input image sequences, which is important for numerous vision-based applications such as autonomous driving and robot localization. Traditional visual odometry over the past few decades have been focus on the optimization problem with geometry constraints to estimate the camera poses, which known as geometry based VO. [1], [2], [3], [4], [5], [6] These methods usually extract the geometry constraints based on the correspondences between sparse feature points or photometric consistency over pixels. They have gained satisfying results in scene with rich texture. However the lack of robustness in textureless scenes have restricted their further applications.

Since deep neural network can extract high level image features and perform end-to-end inference from learning data, many DL-based VO methods have been proposed to promote the robustness for pose estimation tasks. However, these methods ignore the classic optimization process over well defined geometric constraints and have limited success over traditional VO [15]. The main reason is that pose estimation from geometric constraints in many visual odometry algorithms are not differentiable. Therefore the scene-independent error signals from geometric constraints can hardly backpropagate to deep learning network, which causes DL-based methods decline in accuracy. In addition, training these methods requires large datasets with annotated ground true poses. DL-based VO often fails when there is a large gap between test and training data. The inability to generalize to unseen environments limits its wide applications.

To overcome the above limitations in DL-based visual odometry, we propose a reinforcement learning(RL) framework with online adjustment, which can utilize both geometric constraints of classic visual odometry and robust features of deep neural networks. Our method consist of two key components, namely policy-based keypoint extraction module, and graph-based odometry environment. The keypoint extraction module serves as an agent interacting with the visual odometry. This module processes the input image sequence and outputs probability distribution for each image as keypoint selection policy. This policy is online upgraded with reinforcement learning techniques which requires no differentiability for the visual odometry. Therefore this keypoint extraction module is compatible with any kind of traditional keypoint-based visual odometry. The graph-based odometry environment is modified from traditional photometric-based VO methods, which utilizes graph data structures to organize training data and provides feedback for the key point extraction policy. In addition to reduce the redundancy in sequential frames during odometry tasks, a graph pruning method is employed to remove unnecessary frame nodes based on their centrality.

In general, our method adopts a reinforcement learning framework for keypoint extraction and adjusts its policy online over the feedback from the graph-based odometry environment with the input image sequence. With these improvements, our method performs optimization over geometric constraints built with points extracted from deep network, which is online adjusted by reinforcement learning approaches. This enhanced our tracking

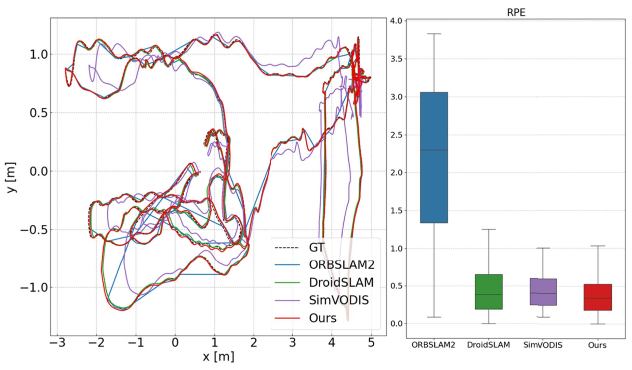
Received 20 December 2024; accepted 17 May 2025. Date of publication 2 June 2025; date of current version 16 June 2025. This article was recommended for publication by Associate Editor R. Sabzevari and Editor J. Civera upon evaluation of the reviewers' comments. This work was supported by the National Natural Science Foundation of China under Grant 52372417 and Grant 52072243. *(Corresponding author: Yafei Wang.)*

Ruitao Zhang, Yafei Wang, Zexing Li, and Mingyu Wu are with the School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: ida_zrt@sjtu.edu.cn; wyfjlu@sjtu.edu.cn; lizexing@sjtu.edu.cn; wmy_marvin@sjtu.edu.cn).

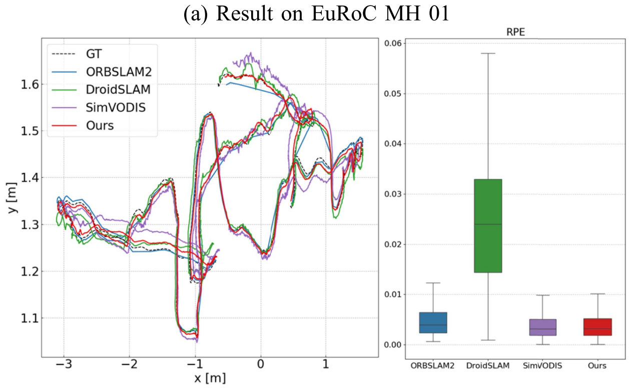
Fei Ding is with College of Mechanical and Vehicle Engineering, Hunan University, Changsha 410082, China (e-mail: dingfei@hnu.edu.cn).

Chongfeng Wei is with the James Watt School of Engineering at University of Glasgow, Glasgow G12 8QQ Belfast, Scotland (e-mail: chongfeng.wei@glasgow.ac.uk).

Digital Object Identifier 10.1109/LRA.2025.3575644



(a) Result on EuRoC MH 01



(b) Result on TUM fr3 lon

Fig. 1. Results of traditional, supervised and unsupervised learning methods on EuRoC MH 01 and Tum fr3 long household validation sequence. Our method updates the parameters solely based on the current sequence, yet still has the highest tracking accuracy under different scenes.

performance and generalization ability, as is illustrated in Fig. 1. Our contributions are summarized as follows:

- A reinforcement learning framework incorporating photometric error feedback is designed to train the keypoint extraction policy regardless the differentiability of visual odometry. Compared to existing DL-based methods, our method learns scene-independent features from geometric constraints and improves the generalization ability of visual odometry.
- An online training scheme is proposed to efficiently trains that dynamically adapts to the current scene in odometry tasks. In addition, the graph-based odometry environment for organizing training data is proposed to further improve the training efficiency. This enables our method to enhance odometry robustness with less training data and an unsupervised online approach.
- To verify the generalization of proposed method, we evaluated it on datasets with different environment, including KITTI [21], EuRoC [22] and TUM [23] datasets. The results show that our method outperforms state-of-the-art methods in accuracy and generalization.

II. Related Work

This section discusses the relevant works with three categories: traditional SLAM methods, deep learning-based VO/SLAM and keypoint extraction.

A. Traditional SLAM

Traditional visual SLAM methods can be mainly divided into feature-based methods [1], [2], [3], [4], [8] and direct methods [5], [6], [7]. Feature-based methods select pixels with highly repeatable features (e.g. corner points) and match them between frames to triangulate points and estimate frame poses. ORB-SLAM [2], [3], [4] is designed with camera tracking, local mapping and loop closing three thread structure. OV2slam [8] proposes a four-threads architecture with online Bag-of-Word loop closure. Direct Sparse Odometry [6] selects interesting points based on the intensity gradient. After initializing the point with epipolar search, it introduces a sliding window approach to optimize the pose of keyframes and inverse depth of pixels. The geometric constraints are formulated after optimizing photometric error related to all map points in the sliding window. LDSO [7] improves DSO with a loop closure detection module which detects loop closure from corner points selected in DSO frontend. The main drawback of traditional methods is the standard of selecting map points from pixels can not provide robust perception to various conditions. [9] Our proposed framework extend the keypoint selection process with feedback from visual odometry to improve the quality of visual map points.

B. DL-Based VO/SLAM

The convolution neural networks (CNNs) have demonstrated extraordinary abilities in regression or classification problems. Therefore, a natural idea is to formulate pose estimation as regression problem and train CNNs for such tasks. PoseNet [10] trains a CNN to regress the pose of current image frame for re-localization purpose. ESP-VO [11] adopts an RCNN structure to estimate the camera motion in an end-to-end scheme. SimVODIS [12] is developed based on the Mask-RCNN to complete multiple tasks for intelligent agents like object detection, instance segmentation and visual odometry.

There are also methods that utilize synthetic datasets to train their models, which cover a diverse range of environments and effectively improve the generalization of learning methods. TartanVO [14] designs a scale-aware loss function and incorporates intrinsic parameters into their model. Although the odometry is trained solely on synthetic datasets, it also demonstrates adaptability to real-world datasets. DroidSLAM [13] develops a Dense Bundle Adjustment layer to estimate the camera pose and pixel-wise depth.

Most DL-based VO/SLAM methods are difficult to form geometric constraints without basic frameworks of traditional SLAM. Our proposed method retains the basic frameworks of traditional SLAM and focuses more on training suitable feature points for camera tracking frontend.

C. Image Keypoint Extraction

Keypoint extraction servers as an important input for SLAM and VO applications. Deep learning based keypoint extraction methods refine the repeatability of detected points with high level features extracted from network. SuperPoinNet [16] utilizes an encoder-decoder network and is trained with synthetic dataset and homographic adaption. LIFT [17] adopts a

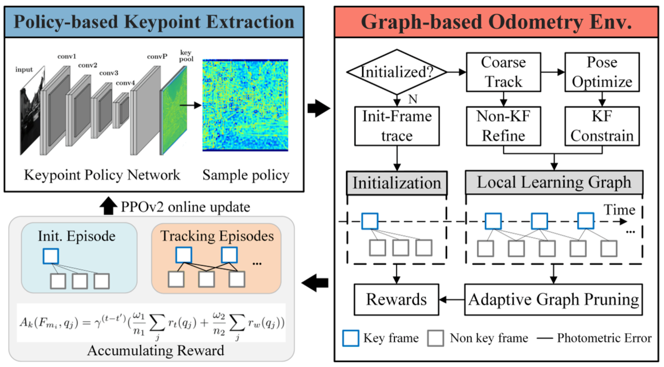


Fig. 2. Main pipeline for the proposed framework. Keypoint policy network provide sample policy to extract keypoints, while taking feedback from graph-based odometry to upgrade its parameters.

traditional patch-based detect and describe framework and learns the orientation of feature points. ASLFeat [18] incorporates a deformable convolutional network to focus on local shape estimation and improves the quality of keypoints. However, the supervision for keypoint learning remains a challenge issue due to the lack of annotated keypoint dataset. In our proposed method, we adopt the synthetic trained SuperPointNet as a basis of our keypoint detector and refine the keypoint quality for tracking image with reinforcement learning technique.

III. System Overview

A. Notations

We formulate our framework with the following notations through out the whole paper. For clarity, we also list several key training-related notations.

We denote i as the index of input frame and j as the point index within each frame. For each map point: $q_j^i = (u_j, v_j, 1)$ denotes pixel coordinates. $p_j^i = (x_j, y_j, z_j)$ denotes 3D coordinates in the global frame. l_j^i denotes the log probabilities. $r_s(q_j^i)$ denotes point-wise reward function in odometry stage s.

For an image sequence: F_i is a $3 \times h \times w$ matrix denoting the i-th frame. T_i denotes its world to camera pose matrix. $A(F_i, q_j^i)$ denotes the advantage function. L denotes the PPO loss function. n_{bs} is the target batch size and n_G is the maximum graph size for local learning graph.

B. Overview of VO Framework

Fig. 2 describes the overview of our proposed framework. The entire framework comprises two primary components: policy-based keypoint extraction and graph-based odometry environment. As illustrated in the figure, the first two components constitute an unsupervised reinforcement learning (RL) system. Policy-based keypoint extraction module servers as the agent of the RL system while the graph-based odometry environment interacts with it.

In order to run a visual odometry task, firstly the keypoint policy network takes the input image and generates a pixel-wise probability distribution. Then interesting pixels are sampled from the distribution and further refined by keypoint pooling. Lastly the visual odometry tracks the selected pixels and estimate camera motion by optimizing the photometric consistency in a sliding window. Section IV-B.

In order to train the keypoint policy network in an online manner, a graph structure is designed to organize frames and formulates the rewards according to tracking errors in the visual odometry. To further improve the efficiency for online training, we prune the graph structure within this local sliding window to reduce the information redundancy. This graph structure is denoted as local learning graph. PPOv2 (Proximal Policy Optimization) [20] algorithm is applied to train the keypoint policy network. PPOv2 is commonly used in reinforcement learning, which optimizes policies by limiting policy updates within a trust region using a clipped advantage function.

IV. Methodology

In this section, we will illustrate the key components of the proposed method and the pipeline for both unsupervised training and SLAM tasks with monocular image sequences.

A. Policy-Based Keypoint Extraction

The detailed network architecture of the policy-based keypoint extraction module is described in Fig. 2. We design the keypoint policy network on top of the detector part in SuperPointNet [16].

For keypoint extraction on a given Frame F_i, firstly the SuperPointNet generates a probability distribution P_i from the input image, representing the possibility of keypoint for each pixel. Next, the keypoint pool layer uses max-pool and max-unpool layers with stride $h_m \times h_m$ to process P_i. This pool layer ensures there is at most one keypoint in each grid after slicing the image into $h_m \times h_m$ grids, which helps the visual odometry to track camera pose with a wider and more evenly distributed keypoints. Lastly we sample m pixels as keypoints q_j^i and record the log probabilities $l_j^i = \log(P_i(u_j, v_j))$ of selected pixels. With these output, the training of the whole network is designed using reinforcement learning approach.

B. Graph-Based Odometry Environment

Our Graph-based Odometry Environment is adapted from traditional direct SLAM framework DSO [6]. This section gives a brief introduction to DSO algorithm and illustrates the construction of local learning graph designed for the interaction between DSO and policy-based keypoint extraction system.

The DSO system is initialized by optimizing the photometric error between the first frame and the following initializer frame. After initialization, it launches a coarse tracking for subsequent frames and map points.

In the coarse tracking phase, each point has its trace status in GOOD, OUTLIER, SKIPPED, BAD CONDITION, UNINITIALIZED [6], as explained in Table I. Then the DSO system maintains a sliding window of frames to establish a small factor graph to optimize photometric constraints between keyframes. The sliding window optimization is performed iteratively to

TABLE I
TRACE STATUS AND LOOKUP TABLE

Trace Status	Immature Point Condition	g
GOOD	traced well and good	5
OUTLIER	trace outlier	-5
SKIPPED	inliner but not actually traced	3
BAD CONDITION	not traced because of bad condition	-5
UNINITIALIZED	not even traced once	-3

keep tracking the camera poses and mapping the keypoints. The detailed process can be found in [6].

In order to adapt DSO for interaction with the policy-based keypoint extraction system, we designed a local learning graph to efficiently organize frames during the DSO optimization process. The local learning graph is a weighted directed graph data structure $G_m = < V_F, E_p >$ representing frames as nodes V_F and inter-frame projections as edges E_p. This graph retains all the frames that can contribute to the selection policy by dynamically incorporating those within the sliding window. During the tracking process, new frames from the DSO sliding window is constantly added into the graph, where the main frame nodes $\{F_{m_i}\}$ host 3D world points that is optimized and non-main frame nodes accept projections from main frame nodes. Therefore, the directed edge points from main frame to target frame. The weight of each edge $e_{ij} \in E_p$ can be denoted as

$$w_{ij} = \frac{|\{q_k^i | e_{ij}\}|}{|\{q_k^i \in F_i\}|} \tag{1}$$

which is the ratio of pixels projected from the main node F_i to the target frame F_j, normalized by the total number of points hosted by F_i. When the size of local learning graph exceeds the maximum graph size n_G, it is pruned to a sub-graph of size n_{bs} for computing the reward function and updating the policy network.

C. Adaptive Graph Pruning

Due to the complex camera motion, co-visibility within the local learning graph may vary differently, resulting in an unstable data size and potential data redundancy. To address this issue, the local learning graph need be pruned into a fixed batch size n_{bs} to promote data efficiency.

Before the pruning process, we evaluate the importance of main nodes is evaluated by the eigenvector centrality [19] of the weight graph $G_m = < V_F, E_p >$. The eigenvector centrality for node i is calculated by the following unique principal eigenvector C:

$$BC = \lambda C \tag{2}$$

where B is the adjacency matrix of the graph G_m, and $C = \{c_1, c_2, \ldots c_{|G_m|}\}$ where c_i is the centrality of node i. These centrality scores help the pruning process to remove less important nodes to maintain a fixed batch size of training frames.

The pruning process is triggered once the graph size (number of frame nodes) is larger than n_G. It begins by arranging all main nodes in chronological order and applying non-maximum suppression based on their c_i values. Those suppressed main nodes are pruned from the graph, together with isolated non-main nodes. This eliminates less important main nodes surrounding those with a local maximum c_i, reducing the potential information redundancy caused by neighboring main nodes. After this, additional non-main nodes with low c_i value are pruned. The number of remaining frame nodes are constantly checked to ensure it decreases to the desired size.

Algorithm 1: Algorithm: Graph Pruning.

```
 Input: Graph $G_m$ and target batch size $n_{bs}$;
 Output: Pruned graph $G'_m$;
 1:  Compute eigenvector centrality $C = \{c_1, c_2, \ldots c_{|G_m|}\}$
 2:  Arrange scores for main nodes
     $C_m = (c_{m_1}, c_{m_2}, \ldots, c_{m_i})$
 3:  Non max suppression for main nodes:
 4:    $I_m \leftarrow find\_local\_maxima\_indices(C_m)$
 5:    $n_w = 2$
 6:    for all $i \in I_m$
 7:      $C_{local} = C_m[i - n_w : i + n_w]$
 8:      suppress nodes whose $arg(C_{local} > 0.5 * c_i)$
 9:    end for
10:    Prune isolated non-main nodes
11:  while $|G_m| > n_{bs}$ do
12:    Select $c_i$ for non-main nodes
       $C_n = \{c_{n_1}, c_{n_2}, \ldots c_{n_i}\}$
13:    $i = argmin(C_n)$;
14:    Prune non-main node $i$
15:  end while
16:  return $G'_m$.
```

The procedure of graph pruning is better illustrated in Algorithm 1. These steps ensure a fixed batch size of image data is fed into the network for propagation.

D. Reward Function Design

The reward functions are designed for each keypoint in the local learning graph, as they constitute the smallest unit of selection policy. These functions are defined based on the current stage s of visual odometry which is divided into initialization, coarse tracking and fine optimization. The latter two stage constitute the main tracking frontend. A wrapper function f_r (6) is also needed to transform the error expression e into reward r based on the statistics of error values. The design of reward functions are illustrated as follows:

In the case of initialization, the depth of each pixels are randomly initialized and iterated updated with photometric error in DSO. Accordingly, we introduces both photometric error e_b and variance of depth updates σ_{d_j} during optimization into the reward function.

$$r_{init}(q_j^1) = r(q_j^1) = f_r(e_b(q_j^1) + \lambda_0 \sigma_{d_j}) \tag{3}$$

Here, the photometric error e_b is the L2 norm term and the depth variance σ_{d_j} serves as a regulation term. This reward reduces the

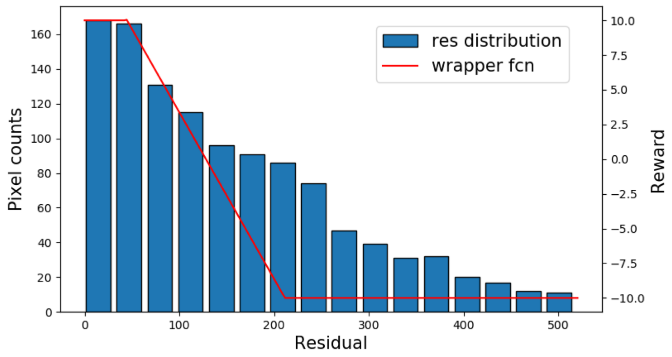


Fig. 3. Wrapper function that maps photometric residuals to reward values. This function provides a statistically even distribution of reward values to help training in reinforcement learning.

photometric error of initialization and encourages faster depth convergences for extracted keypoint.

In the case of coarse tracking, the quality of immature keypoint updates is critical to the optimization and the map quality, thus both the photometric error and the number of good point traces are considered in the reward function. the traces status $s_{j_{trace}}$ of immature points is used to construct the reward with a lookup table g (Table I):

$$r_t(q_j^i) = r(q_j^i) + g(s_{j_{trace}}) \tag{4}$$

This reward makes the selection network favour successful traces to improve the quality of map points. In the case of fine optimization, the reward function has a similar design to r_{init} with an additional penalize term for tracking outliers:

$$r_w(q_j^i) = r(q_j^i) + \lambda_p(1 - \delta(q_j^i)) \tag{5}$$

$\delta(q_j^i)$ is an indicator function that represents whether the optimization of q_j^i is successful and here $\lambda_p = 10$ is used to penalize tracking outliers in our training.

The wrapper function f_r is a slope function with min and max values. It maps the reward into a proper range with desired mean value and positive reward rate.

$$f_r(e) = \begin{cases} R_{\max} & e > e_1 \\ (e - e_0)\frac{(R_{\max} - R_{\min})}{e_1 - e_0} + R_{\min} & e \in [e_0, e_1] \\ R_{\min} & e < e_0 \end{cases} \tag{6}$$

The function is better illustrated in the plot together with the histogram of error values in Fig. 3. It prevents extreme reward values by setting the min-max threshold and maps most error value into slope area to adjust their policy.

The above reward function is accumulated for each keypoint in the key frame. Once the system is initialized, or the size of local learning graph reaches over the desired batch size, a back-propagation process is triggered to update the network parameter. This is illustrated in the next section.

E. Training Scheme

The training process of the keypoint policy network is divided into two episodes: initialization and tracking. Since the initialization episode is relatively short. The rewards are accumulated over each initializer frame and backpropagation is instantly performed once the initialization is completed. Its advantage function A_0 is formulated as:

$$A_0(F_i, q_j) = \sum_j r_{init}(q_j) \tag{7}$$

The tracking episode involves a larger amount of data. Therefore, we divide it into sub episodes based on the changes in the local learning graph illustrated in Section IV-B and update the network parameters in each sub episode.

After acquiring a pruned local learning graph for the k-th sub episode, we calculate the advantage function through its structure. Firstly, all the related reward for the map points of each main node is summed up and normalized as a advantage term $A_k(F, q_j)$. The above steps is formulated as follows:

$$A_k(F_{m_i}, q_j) = \gamma^{(t-t')}\left(\frac{\omega_1}{n_1}\sum_j r_t(q_j) + \frac{\omega_2}{n_2}\sum_j r_w(q_j)\right) \tag{8}$$

For calculating the advantage term r_{adv}, ω is the weight for reward term and $n_{1,2}$ is the number of each accumulated reward. $\gamma^{(t-t')}$ is the reward decay term where t is the current game step and t' is the game step when r is accumulated. Propagating advantage between the local learning graph improves the data efficiency of PPO algorithm. To compute the advantage function for more pixels, after computing A_k for main frame F_{m_i}, the value is projected onto the non-main frame connected with F_{m_i}, which can be formulated as:

$$A_k(F_{n_i}, \pi_K(T_{n_i} \cdot T_{m_i}^{-1}, q_j \cdot d_k)) = A_k(F_{m_i}, q_j) \tag{9}$$

Then, for each projected pixels in non-main nodes, the advantage term is propagated from its map points projections. Lastly a PPOv2 loss function is calculated for each pixel in the local learning graph and ppo iteration starts. The final loss function of PPOv2 is expressed as follows.

$$L_{init} = \sum_{F,j} \min(\eta A_0, clip(\eta, 1-\varepsilon, 1+\varepsilon)A_0) \tag{10}$$

$$L_{track} = \sum_{F,j} \min(\eta A_k, clip(\eta, 1-\varepsilon, 1+\varepsilon)A_k) \tag{11}$$

where $\eta = e^{l_j^{i'} - l_j^i}$ is the importance resampling factor and ε is the maximum policy change. l_j^i is the log probability for point q_j^i. L_{init} and L_{track} represent different episode loss. After calculating advantage functions for each nodes in the local learning graph structure, we start the PPOv2 iteration and update the network parameter.

V. Evaluation

This section presents the evaluation settings for verifying the performance of our proposed method. We compared the accuracy of odometry localization with several famous algorithms. To further evaluate the generalization ability of our proposed method, we performed evaluation among different datasets including indoor and outdoor environments.

TABLE II
COMPARISON OF THE RMSE OF THE RPE (M) ON KITTI DATASET

Sequences	Traditional			Supervised		Unsupervised		
	ORBSLAM3 (LC) [4]	OV^2 SLAM (LC) [8]	DSO [6]	DroidSLAM (LC) [13]	TartanVO [14]	SimVODIS [12]	Ours (w/o OA)	Ours
00	**1.2043**	6.4084	4.6554	14.0355	4.4045	7.3538	4.8469	4.7783
01	52.8684	8.4011	2.8954	21.9170	2.0931	9.5363	1.5005	**1.1226**
02	1.6093	12.9344	5.7408	19.0125	**1.6002**	3.7775	2.6686	2.2143
03	0.3712	0.8545	0.7153	1.4015	0.4724	0.3778	0.3936	**0.3434**
04	0.2923	0.2893	0.3286	1.6354	0.8057	0.8724	0.3326	**0.1946**
05	**1.0909**	5.1753	2.9780	12.1196	1.3505	1.4881	2.9073	2.7335
06	4.3262	15.7912	4.7840	14.4931	**0.8732**	2.2761	2.4850	1.9142
07	0.8189	7.8392	1.4132	10.7648	1.0132	2.4021	1.3601	**0.7132**
08	5.5758	6.5177	4.4052	9.8335	**3.5129**	5.1461	5.8003	4.4590
09	2.1611	3.6408	2.7879	14.7476	1.0370	2.6398	1.3587	**1.0094**
10	1.4902	2.0676	1.4265	9.3889	0.8292	1.6247	1.3228	**0.5552**

Best results are shown in bold.

A. Evaluation Settings

To evaluate the accuracy and generalization of odometry, we utilize three diverse datasets: KITTI [21], EuRoC MAV [22] and TUM [23]. For all datasets we only use monocular image sequences in gray scale. Since our method is a visual odometry approach, we selected sequences with fewer loop closures and no kidnapping scenarios.

Our model is implemented on the computing server with AMD EPYC 7763 processor and Nvidia A6000 48 G GPU. Detailed model parameters are given as follows. The keypoint policy is configured to generate $m_1 = 3000$ keypoints, distributed across an 8x8 grid. Training of the keypoint policy network starts with the pretrained weights of SuperPointNet. The wrapper function f_r is set to map the error value to a reward range in $[-10, 10]$, and provide 50% positive rewards. For reward calculation we set $\lambda_0 = 0.5$. For the PPOv2 parameters, we set $\omega_1 = 1$, $\omega_2 = 0.5$, $\gamma = 0.6$, $\epsilon = 0.1$, and PPO epoch $= 12$ for each sub episode. Adam solver with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a learning rate of 1×10^{-6} is employed to update the network parameters. We set batch size $n_{bs} = 16$ and maximum graph size $n_G = 32$ for online adaption. In comparison with DSO, we keep the same evaluation parameters such as number of keypoints, number of key frames, iteration count for optimizing pose, etc.

To evaluate the effectiveness of online adaptation, two versions of the model are tested: one is trained by the aforementioned online adaptation strategy, with training data comes from current view of the active sequence. The other one is trained on a fixed sequence (KITTI 00 or KITTI 05, EuRoC MH02, TUM fr3 cabinet) with network parameters frozen during testing. We choose several famous algorithms including traditional method, supervised and unsupervised deep learning method to compete with our proposed method. We trained our offline method on KITTI 05 sequence to evaluate it on KITTI 00. The overall performance of the pose estimation is evaluated by calculating the RMSE of Relative Pose Error (RPE) with $\Delta = 1$ s.

B. Results on KITTI Odometry Dataset

The KITTI Odometry dataset is a well-known dataset for evaluating SLAM systems. We adopted KITTI dataset to evaluate the generalization ability of our method in common outdoor driving environment.

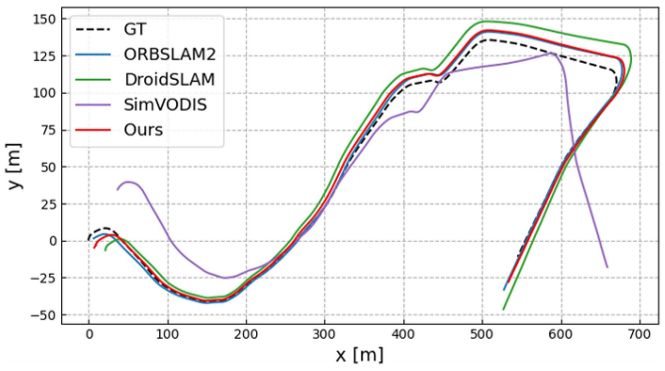


Fig. 4. Examples of trajectory results on KITTI 10.

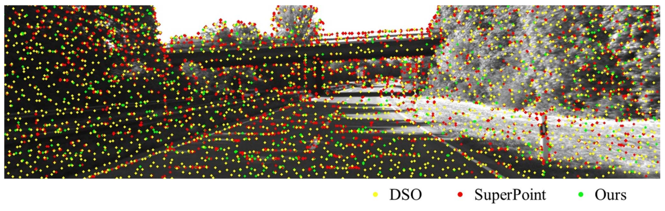


Fig. 5. Examples of point selection results on KITTI dataset.

Table II demonstrate the quantitative performance of our method compared to baseline methods. Fig. 4 describe a quantitative trajectory result of our method compared to baseline method on KITTI 01 and 10 sequence. Fig. 5 illustrates a comparison of point selection across the different methods. Compared with original DSO, our method demonstrates much better performance on tracking accuracy, indicating that the keypoint selection in our method has been enhanced by the policy network. In addition, the online adaption of our method further improves the tracking accuracy by adjust the keypoint selection policy with online feedback generated from current view of sequence. Although our method exhibits larger errors on the KITTI sequences 00, 02, 05, 06 and 08 due to the absence of

TABLE III
COMPARISON OF THE RMSE OF THE RPE (M) ON EUROC MAV DATASET

Sequences	Traditional			Supervised		Unsupervised		
	ORBSLAM3 (LC) [4]	OV^2 SLAM (LC) [8]	DSO [6]	DroidSLAM (LC) [13]	TartanVO [14]	SimVODIS [12]	Ours (w/o OA)	Ours
MH 01	1.4256	0.5248	0.5628	0.5847	0.6924	3.9851	0.5244	**0.4767**
MH 03	3.8613	1.4181	1.4168	1.6942	1.4086	1.8876	1.4116	**1.1809**
MH 04	2.2414	1.3714	8.4102	**1.5068**	1.6730	2.1305	8.3908	8.0874
MH 05	5.1651	1.2079	1.3826	1.2776	1.5780	1.9430	1.2296	**1.2247**
V1 01	2.0621	**0.4267**	0.4987	0.5989	0.4314	3.6694	0.5005	0.4960
V1 02	1.2443	0.9876	2.0345	**1.0932**	1.1474	1.8995	1.9900	1.9310
V2 01	1.2913	1.6711	0.3382	0.3774	0.4032	2.3006	0.3369	**0.3072**
V2 02	2.2113	2.5237	0.7900	0.8716	0.8572	1.7077	0.7976	**0.7280**
V2 03	1.3352	2.5862	1.0097	1.1578	**0.9584**	1.8978	1.0011	0.9969

Best results are shown in bold.

TABLE IV
COMPARISON OF THE RMSE OF THE RPE (M) ON TUM RGBD DATASET

Sequences	Traditional			Supervised		Unsupervised		
	ORBSLAM3 (LC) [4]	OV^2 SLAM (LC) [8]	DSO [6]	DroidSLAM (LC) [13]	TartanVO [14]	SimVODIS [12]	Ours (w/o OA)	Ours
fr2 large no loop	0.6481	1.5171	1.0429	6.1907	7.2219	7.2830	1.1334	**0.5408**
fr3 long office	0.2305	0.1403	0.1296	0.8074	0.1700	1.5438	0.1473	**0.1277**
fr2 pioneer slam	-	0.6552	0.6560	0.6989	0.5331	1.6124	0.7187	**0.3202**
fr2 desk	0.3216	**0.0967**	0.4614	2.7081	0.2544	1.0493	0.4241	0.2851

Best results are shown in bold. - indicates failed sequences.

loop closure detection, it still achieves comparable results with other pure visual odometry methods.

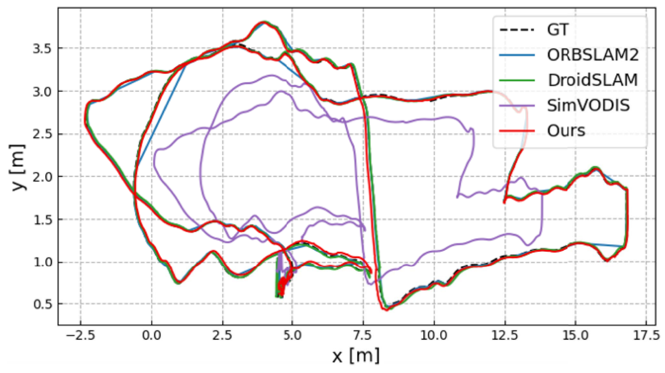
C. Generalizing Results to Other Datasets

To evaluate how well our method generalizes to novel scenes, we conducted experiments on EuRoC MAV dataset and TUM RGBD dataset. The EuRoC MAV dataset is collected by MAVs in an indoor factory environments, where as the TUM RGBD dataset is collected with handheld devices in indoor environments. Both datasets are collected in a different environment and platform setting compared with KITTI odometry dataset.

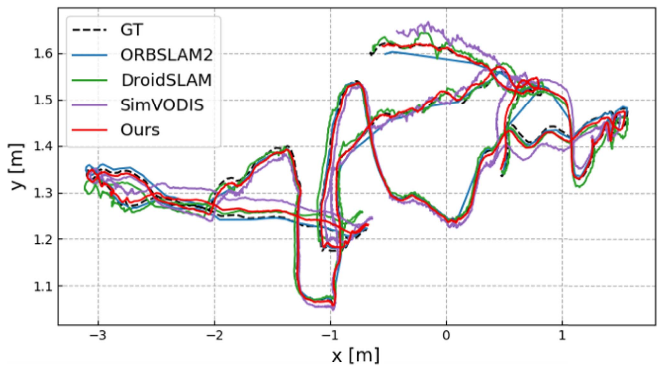
The results for EuRoC MAV dataset and TUM RGBD dataset are presented in Tables III and IV and Fig. 6. Since SimVODIS was only trained on the KITTI dataset, its performance declines on these two different datasets. DroidSLAM and TartanVO utilize a large number of synthetic datasets for network training thus better adapted to different environments. In contrast, our online training scheme only utilize images in current odometry optimization to update the keypoint selection policy, yet still outperforms other method in most evaluated sequences. Our method only performs less accurate in long sequences with loop closure. It is observed that our algorithm demonstrates generalization to different motion patterns and for both indoor and outdoor environments, and performs better in selecting robust visual features.

D. Ablation Study

To further evaluate the effectiveness of our proposed reinforcement learning framework, we first conducted an ablation



(a) EuRoC MH05



(b) TUM fr3 long office

Fig. 6. Examples of trajectory results on other dataset.

TABLE V
ABLATION STUDY ON TRAINING METHODS

Methods	KITTI 10	MH05	fr3 long off	Avg.FPS
DSO	1.4265	1.3826	0.1296	64.5
SuperP	1.3482	1.2466	0.1498	15.2
Ours (w/o OA)	1.3228	1.2296	0.1473	13.4
Ours (w/o LG)	0.9473	1.3651	0.1374	1.3
Ours ($n_{bs} = 4$)	0.7300	1.2961	0.1421	10.6
Ours ($n_{bs} = 8$)	0.8942	1.3114	0.1349	9.7
Ours ($n_{bs} = 16$)	0.5552	1.2247	**0.1277**	8.4
Ours ($n_{bs} = 32$)	**0.5365**	**1.1874**	0.1242	6.5

study on the training methods and the roles of different training modules. We choose KITTI 10, MH05 and TUM fr3 long office image sequence from the above three different datasets to cover different environments. We compared DSO, SuperPointNet + DSO (SuperP), offline training, online training with and without the local learning graph (w/o LG) to validate the effectiveness of our different modules. In addition, to further validate the impact of batch size on VO performance and real-time capabilities, we conducted experiments with different batch sizes and evaluated average fps. The results are illustrated in Table V.

It can be noticed that reinforcement learning improved the accuracy of our visual odometry. The local learning graph in RL enables the policy to adapt to a wider view of current image sequence online and improves the tracking performance. Moreover, the pruning process of local learning graph also reduces the amount of data in online adaption, which increases the real time performance. However the computational cost for online adaption still increases since it involves backpropagation through the active parameters in the neural network. Tuning the batch size can balance real-time performance and accuracy for online version of our proposed method.

VI. Conclusion

In this letter, our proposed method demonstrates improved tracking accuracy and generalization across various datasets compared with baseline methods. Our method utilizes reinforcement learning to adjust its parameters online with reward functions constructed from tracking error evaluations. In comparison to traditional methods, our approach offers a more robust feature extraction for image matching, and unlike supervised learning methods, our approach is capable of adapting to novel scene during the tracking process. Moreover, the reinforcement learning framework proposed in this paper can adapt to other traditional odometry method based on keypoint extraction, thus introducing geometric or photometric constraints from traditional visual odometry to improve the performance of DL-based methods.

References

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

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

[3] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[4] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1874–1890, Dec. 2021.

[5] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proc. Eur. Conf. Comput.*, Sep. 2014, pp. 834–849.

[6] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 611–625, Mar. 2018.

[7] X. Gao, R. Wang, N. Demmel, and D. Cremers, "LDSO: Direct sparse odometry with loop closure," in *Proc. 2018 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2018, pp. 2198–2204.

[8] M. Ferrera, A. Eudes, J. Moras, M. Sanfourche, and G. Le Besnerais, "OV ^{2} SLAM: A fully online and versatile visual SLAM for real-time applications," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 1399–1406, Apr. 2021.

[9] C. Cadena et al., "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016.

[10] 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.*, 2015, pp. 2938–2946.

[11] S. Wang, R. Clark, H. Wen, and N. Trigoni, "End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks," *Int. J. Robot. Res.*, vol. 37, no. 4–5, pp. 513–542, 2018.

[12] U. H. Kim, S. H. Kim, and J. H. Kim, "SimVODIS: Simultaneous visual odometry, object detection, and instance segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 1, pp. 428–441, Jan. 2022.

[13] Z. Teed and J. Deng, "DROID-SLAM: Deep visual slam for monocular, stereo, and RGB-D cameras," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 16558–16569.

[14] W. Wang, Y. Hu, and S. Scherer, "TartanVO: A generalizable learning-based VO," in *Proc. Conf. Robot Learn.*, Oct. 2021, pp. 1761–1772.

[15] K. M. Jatavallabhula, S. Saryazdi, G. Iyer, and L. Paull, "gradSLAM: Automagically differentiable SLAM," 2019, *arXiv:1910.10672*.

[16] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018, pp. 224–236.

[17] K. M. Yi, et al., "Lift: Learned invariant feature transform," in *Proc. 14th Eur. Conf. Comput. Vis.*. Springer, 2016, pp. 467–483.

[18] Z. Luo et al., "Aslfeat: Learning local features of accurate shape and localization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6588–6597.

[19] P. Bonacich, "Power and centrality: A family of measures," *Amer. J. Sociol.*, vol. 92, no. 5, pp. 1170–1182, 1986.

[20] J. Schulman et al., "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[21] 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.*, 2012, pp. 3354–3361.

[22] M. Burri et al., "The EuRoC micro aerial vehicle datasets," *Int. J. Robot. Res.*, vol. 35, no. 10, pp. 1157–1163, 2016.

[23] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 573–580.