RLSAC: Reinforcement Learning enhanced Sample Consensus for End-to-End Robust Estimation

Chang Nie[1], Guangming Wang[1,3], Zhe Liu[2*], Luca Cavalli[3], Marc Pollefeys[3,4], Hesheng Wang[1*]
[1]Department of Automation, Key Laboratory of System Control
and Information Processing of Ministry of Education, Shanghai Jiao Tong University
[2] MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
[3] Department of Computer Science, ETH Zürich [4] Microsoft Mixed Reality and AI Zürich Lab
{changnie,wangguangming,liuzhesjtu,wanghesheng}@sjtu.edu.cn
lcavalli@ethz.ch marc.pollefeys@inf.ethz.ch

Abstract

Robust estimation is a crucial and still challenging task, which involves estimating model parameters in noisy environments. Although conventional sampling consensus-based algorithms sample several times to achieve robustness, these algorithms cannot use data features and historical information effectively. In this paper, we propose RLSAC, a novel Reinforcement Learning enhanced SAmple Consensus framework for end-to-end robust estimation. RLSAC employs a graph neural network to utilize both data and memory features to guide exploring directions for sampling the next minimum set. The feedback of downstream tasks serves as the reward for unsupervised training. Therefore, RLSAC can avoid differentiating to learn the features and the feedback of downstream tasks for end-to-end robust estimation. In addition, RLSAC integrates a state transition module that encodes both data and memory features. Our experimental results demonstrate that RLSAC can learn from features to gradually explore a better hypothesis. Through analysis, it is apparent that RLSAC can be easily transferred to other sampling consensus-based robust estimation tasks. To the best of our knowledge, RLSAC is also the first method that uses reinforcement learning to sample consensus for end-to-end robust estimation. We release our codes at `https://github.com/IRMVLab/RLSAC`.

1. Introduction

As a fundamental module in computer vision, robust estimation is crucial for many tasks, such as camera pose estimation [10, 33, 31, 32], motion segmentation [26, 19, 29, 30], short and wide baseline matching [22, 28], plane fitting [18], and line fitting [12, 20]. However, it is still difficult to exclude disturbances while estimating accurate models. To address this issue, sampling consensus-based algorithms are widely used, which are represented by the RANdom SAmple Consensus (RANSAC) [16] algorithm. It first samples the minimum set required for the task, *e.g.*, a minimum set size of 2 for 2D line fitting. Then, the hypothesis is solved by the minimum set. Next, all data is divided into inliers and outliers, according to their residuals to the hypothesis. Finally, the above process is repeated and the best hypothesis is selected based on the highest inlier ratio. RANSAC

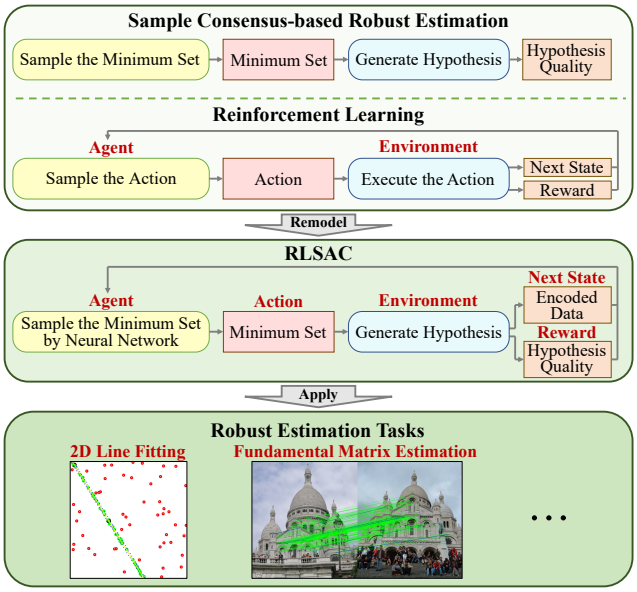

Figure 1. **RLSAC remodel the sampling consensus process.** By modeling the sampling consensus as a reinforcement learning process, RLSAC can achieve end-to-end learning on various robust estimation tasks.

*Corresponding Authors. The first two authors contributed equally.

can provide strong robustness and generalization, but as the outlier rate increases, the probability of sampling inliers decreases. As a result, the performance of RANSAC degrades rapidly. This is because RANSAC samples each data evenly, regardless of data features that can be used for classifying inliers and outliers. Additionally, the non-differentiable deterministic random sampling of RANSAC also limits it to be integrated into the learning-based pipeline.

Since sampling the minimum set from the data is quite similar to the process of sampling an action from the action space in reinforcement learning [17, 13], the sampling consensus can be integrated into the reinforcement learning framework. Sampling in reinforcement learning can be achieved through a neural network, which extracts the data features. In addition, the reward from the environment can be used to train the reinforcement learning framework without differentiation. Therefore, to learn from data features and avoid differentiating, we propose the RLSAC: Reinforcement Learning enhanced SAmple Consensus for end-to-end robust estimation. As shown in Figure 1, RLSAC regards sampling consensus as the process of interaction between the agent and environment in reinforcement learning. Specifically, the agent uses a neural network to sample the minimal set from the data as an action. The environment then performs model generation and evaluation based on the action and outputs the next state, which is used in the next iteration.

However, designing appropriate reward and state are very important and challenging in reinforcement learning. To achieve reinforcement learning enhanced sampling consensus, RLSAC proposes new state transition and reward modules. Specifically, the state is encoded by augmenting the original data features with memory features, including the current action, data residuals, and historical information. When the state is input into the agent, these features can provide more information about the quality of the previous action. This allows RLSAC to gradually explore a better hypothesis by utilizing this memory information.

Additionally, the evaluation result of the generated hypothesis can be used as the reward signal to train the neural network without differentiation. The reward signal enables the learning-based sampling consensus for end-to-end robust estimation. Furthermore, the evaluation result is the feedback from the downstream task. Thus, the neural network can learn to effectively use the data features and optimize the output to meet the requirements of the downstream task.

Moreover, instead of directly predicting the final result from the data in one shot [27], RLSAC employs multiple episodes, each containing several sampling processes. In addition, RLSAC performs one random sampling at the beginning of each episode to form the initial state. This approach preserves the robustness of multiple random sampling and provides basic performance for RLSAC. Besides, RLSAC can be extended to other robust estimation tasks since it is not restricted to any specific task.

The proposed RLSAC is tested on two classic robust estimation tasks, which are the 2D line fitting task and the fundamental matrix estimation task. The experimental results show that RLSAC achieves great performance.

Our main contributions are as follows:

- We propose RLSAC: a novel Reinforcement Learning enhanced SAmple Consensus framework for end-to-end robust estimation. It learns data features to sample the minimum set. RLSAC retains the robustness of the multiple sampling process, while the initial random sampling can provide the basic performance.
- RLSAC proposes an approach for state encoding, which includes both current and historical information. This enables the agent to assess the quality of the previous actions and gradually explore better hypotheses. RLSAC is trained unsupervised using the reward function, which avoids differentiating the sampling process and achieves end-to-end robust estimation.
- RLSAC is evaluated on two robust estimation tasks. The 2D line fitting task demonstrates its robustness to disturbances and effective progressive exploration capability. In the fundamental matrix estimation task, RLSAC achieves state-of-the-art performance. Furthermore, RLSAC can be easily applied to other sampling consensus-based robust estimation tasks.

2. Related Work

Robust estimation is a basic module for many tasks. Although the simple repetitive random sampling strategy of RANSAC [16] is robust and generalized, it has some limitations, such as no further optimization, inefficient sampling, and non-differentiability.

There are several methods observing that local features help to optimize the sampling result. LO-RANSAC [15] continues to sample in the inliers with a smaller threshold after sampling the best model so far, aiming to find a better hypothesis near the current one. NAPSAC [25] uses a fixed hypersphere to acquire local data and samples from it, but this approach loses global features and may get stuck in local data. To address this, progressive NAPSAC [4] gradually expands the hypersphere to extend the sampling from local to global. GC-RANSAC [5] considers spatial continuity between inliers and surrounding points, modeling the connections of the data through graph-cut to divide inliers and outliers. MAGSAC++ [6] assesses the model quality by weighting various thresholds to reduce the sensitivity to the choice of a specific noise scale.

To improve sampling efficiency, some methods use guided sampling instead of random sampling. PROSAC [14] sorts the data based on a quality function and then samples the sorted data sequentially to improve efficiency. USAC

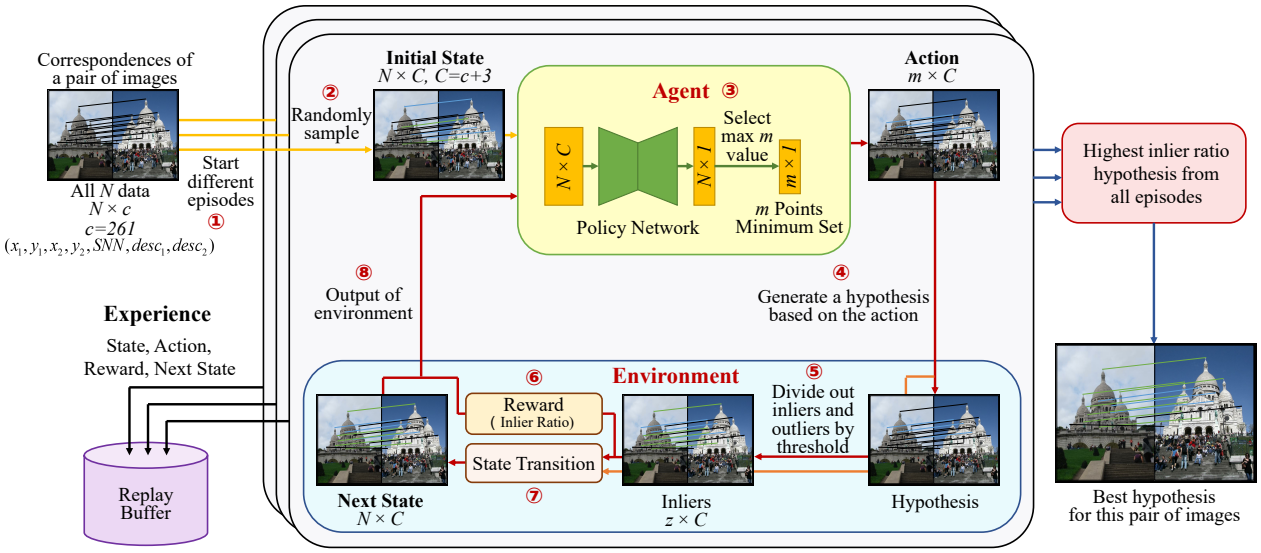

Figure 2. **The pipeline of the proposed RLSAC.** The fundamental matrix estimation problem is used as an example. The black, green, and blue lines represent outliers, inliers, and the minimum set, respectively. The yellow arrows are only used once during initialization. The red and orange arrows indicate the loop in an episode. The initial states are randomly sampled. The best hypothesis for this scene is selected by scoring all hypotheses. The collected experience is recorded in the replay buffer for training.

[24] integrates the advantages of various methods to achieve robustness and efficiency. To learn from the data, Yi et al. [36] are the first to use a neural network based on PointNet to directly classify inliers and outliers. Then, Zhang et al. [37] improve this work by proposing pooling and unpooling blocks to learn the local context of correspondences. NG-RANSAC [9] uses a neural network to calculate the sample probability for each point, achieving probability-based sampling. The neural network is trained using reinforcement learning, but it did not use reinforcement learning to achieve sampling consensus and progressive exploration. Barath et al. [2] propose the MQ-Net to learn from residual histograms and evaluate the quality of the model. Additionally, the authors design the MF-Net to learn to reject bad minimum sets early, further improving efficiency. NeFSAC [11] uses a neural network to reject the motion-inconsistent and poorly-conditioned minimal samples.

The non-differentiability of RANSAC makes it challenging to integrate into an end-to-end learning pipeline. Some methods propose the differentiable RANSAC. For example, DSAC [8] replaces the deterministic selection process with probabilistic selection, allowing for differentiation with respect to the data. Wei et al. [35] achieve gradient propagation by predicting the probabilities of being inliers to guide sampling. To avoid differentiation, some methods use the loss as reward signals through reinforcement learning to train the network. Bhowmik et al. [7] use reinforcement learning to train the feature point detection network but not for the sampling consensus process. Truong et al. [27] iteratively delete points with reinforcement learning, seeking the maximum consensus model that meets the threshold. However, the authors encode the attributes of the points into the state, but does not include long-range historical information or the position of the hypothesis in the state space. In addition, this method does not include the sampling consensus process and may overlook better solutions.

3. Method

3.1. Problem Formulation

Robust estimation can be considered as generating a good hypothesis h given a set of data $\chi = \{\mathrm{x}_i\}_{i=1}^{N}$, which may be disturbed by noise. For instance, the hypothesis h could represent the parameters of a 2D line, and χ could be all the points in the 2D plane. Or h could be the fundamental matrix that represents the epipolar geometry of a pair of images, and χ could be all the correspondences. Similarly, the data contained in x_i varies with the task. As a commonly used robust estimation method, the sampling consensus method samples n minimum sets $\mathcal{M}$ from χ. The size m of a minimum set varies depending on the task. For example, when fitting a 2D line, the size $m = 2$. Then, these minimum sets $\mathcal{M}$ are solved by the minimum solver S to output hypotheses:

$$H = \{S(\mathcal{M}_j) \,|\mathcal{M}_j \in M, j = 1, 2, ..., n\}. \tag{1}$$

Next, the hypotheses H are evaluated using a scoring function f. The residuals of all data points χ can be computed and used to calculate the inlier ratio, which is commonly used as a metric for the hypothesis quality [16]. Finally, the hypothesis with the highest score is chosen as the best hypothesis h_{Best}:

$$h_{Best} = \underset{h \in H}{\arg\max}\, f(h, \chi). \tag{2}$$

By solving the problem using several minimum sets $\mathcal{M}$, the model can achieve robust estimation, which is less sensitive to outliers and can lead to more accurate results.

The RANSAC [16] algorithm performs random sampling to sample the minimum sets $\mathcal{M}$, which cannot fully exploit the data features. As shown in Figure 1, One alternative is to consider the sampling of a minimum set as an action taken by an agent based on the current state, within the reinforcement learning framework:

$$a_{t+1} \sim \pi_\phi \left(a_t \mid s_t \right). \tag{3}$$

Here, the policy network π_ϕ depends on the weights ϕ. The action a_{t+1} is sampled by the policy π_ϕ from the state s_t and a_t at time $t+1$, which can also be viewed as the process of sampling a minimum set $\mathcal{M}_j$ from all data χ. So the Eq. 2 can be rewritten that the best hypothesis is selected by a reinforcement learning enhanced sample consensus method:

$$h_{Best} = \underset{t=1,2,...,j}{\arg\max}\, f\left(S\left(a_t \right), \chi \right). \tag{4}$$

In addition, the policy π_ϕ is a trainable neural network, and the state s_t contains the features of the data χ. Therefore, the minimum set can be selected based on sufficient learned knowledge of the data features, rather than being chosen randomly. Furthermore, the evaluation result $f\left(S\left(a_t \right), \chi \right)$ of the minimum solver S can serve as the reward in reinforcement learning to achieve unsupervised learning.

3.2. System Framework

With the problem formulation of modeling the process of sampling consensus as a reinforcement learning problem, we introduce RLSAC, which is illustrated in Figure 2. Although the fundamental matrix estimation task is used as an example, the pipeline is also applicable to other robust estimation tasks. Firstly, RLSAC initiates multiple episodes for the correspondences of a pair of images χ at ①. An episode contains many steps. At the start of each episode, a random sampling of the minimum set $\mathcal{M}_0$ is performed to generate the initial state s_0 through state transition at ②. Specifically, the initial state s_0 is obtained by concatenating each data point with the memory features, which are consisting of action, residual, and historical features (see Section 3.5).

Next, the initial state s_0 is input into the sampling consensus loop. The agent receives the current state s_t and feeds it into the policy network π_ϕ at ③, as shown in Section 3.3. The network generates a probability for each data point. The m points with the highest probabilities are selected as the minimum set $\mathcal{M}_t$, instead of random sampling.

Then, the minimum set $\mathcal{M}_t$ serves as the action a_t for the agent, which generates a hypothesis h_t in the environment at ④, as shown in Section 3.4. The residuals of points to the hypothesis are compared with a threshold to classify points into inliers and outliers at ⑤. The ratio of inliers is utilized as the reward r_t for the current action a_t to train the policy network π_ϕ at ⑥. Additionally, the action, inliers, and residuals are used for the state transition to generate the next state s_{t+1} at ⑦, as shown in Section 3.5. Finally, the environment outputs the next state s_{t+1} and the reward r_t, which are used by the agent to start the next step at ⑧.

Furthermore, the state s_t, action a_t, reward r_t, and the next state s_{t+1} in every step are collected as experiences in the replay buffer [13] for training. After all episodes are complete, the hypothesis with the highest inlier ratio across all episodes is output as the best hypothesis h_{Best} for this pair of images.

3.3. Sampling Minimum Set by Agent

The agent receives the state $s_t \in \mathbb{R}^{N \times C}$, where N is the number of the correspondences of a pair of images. The channel $C = c+3$, with c denoting the dimension of data features and 3 representing the dimension of memory features (see Section 3.5). In addition, the first hypothesis h_0 of each episode is generated using a randomly sampled minimum set, which provides a basic performance for RLSAC.

Then the state s_t is fed into the policy network π_ϕ. To achieve permutation invariance for the input data, RLSAC uses edge convolution (EdgeConv) from the DGCNN [34] as the basic module to establish the policy network π_ϕ. The EdgeConv can model the interrelationship of correspondences by graph neural networks. It can extract features from neighboring nodes in a graph and aggregate them into a central node. The policy network π_ϕ extracts data features and calculates the probabilities by softmax for each point of being in the minimum set. The probability can be interpreted as the probability of obtaining a greater return on the action for the long term, rather than the probability of achieving the best result in the current state. Consequently, the m points with the highest probability are selected as the minimum set $\mathcal{M}_t$ in the current state s_t. Importantly, all previously used minimum sets are recorded to avoid selecting a duplicate minimum set. Thus, when a new minimum set is selected, it is checked whether it has been used. If the new minimum set has already been used, then another m data set is selected following the probability, which will be checked as well. Until the unused data set is found, it is output as the minimum set. Ultimately, the minimum set is as the action $a_t \in \mathbb{R}^{m \times C}$ output by the agent.

3.4. Evaluating Hypothesis in Environment

The environment generates a hypothesis h_t based on the received action a_t from the agent. Next, the residuals R are calculated for all data χ with respect to the hypothesis:

$$R_t = \left\{ D\left(x_i, h_t \right) | x_i \in \chi, i = 1, 2, ..., N \right\}, \tag{5}$$

where the function D is for residual calculation. The scalar valued residuals R are calculated at all data χ. With the

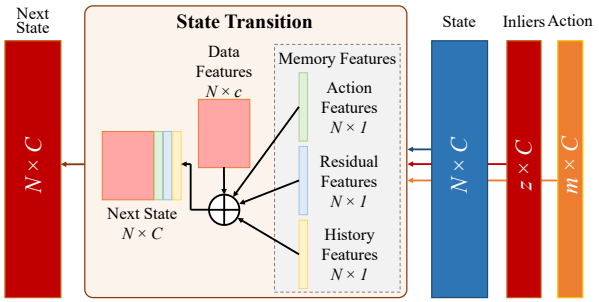

Figure 3. **The state transition in RLSAC.** To form the next state, the data features are added with the memory features, which contain action features, residual features, and historical features.

pre-defined threshold, the data χ is divided into inliers $I_t \in \mathbb{R}^{z \times C}$ and outliers $O_t \in \mathbb{R}^{(N-z) \times C}$ based to the residuals.

In the sampling consensus methods, the inlier ratio commonly represents the quality of the hypothesis. Since the hypothesis h_t is generated by the action a_t in RLSAC, the inlier ratio can be considered as the reward r_t of the current action a_t to train the policy network π_ϕ without ground truth.

With the hypotheses related-rewards, RLSAC can update weights ϕ of the policy network π without differentiating the sampling process. By modeling sampling consensus as an interactive process in reinforcement learning, RLSAC achieves end-to-end robust estimation while avoiding differentiating.

3.5. State Transition

The state transition module allows RLSAC to encode the previous state and action, enabling RLSAC to explore the state space effectively. As illustrated in Figure 3, the next state s_{t+1} is obtained by concatenating the original data features $N \times c$ with memory features $\{A, R, \mathcal{H}\}$, which comprise action features $N \times 1$, residual features $N \times 1$ and historical features $N \times 1$.

The action features are derived from the current action a_t. If a data point x_i is used in the action a_t, the corresponding entry in the action features α_i is set to 1, otherwise it is -1:

$$A_t = \{\alpha_i\}, \alpha_i = \begin{cases} 1 & \text{if } x_i \in a_t \\ -1 & \text{otherwise} \end{cases}, i = 1, 2, ..., N. \tag{6}$$

The action features could provide the agent with information on the current action, like the current location in the data for the next state.

The residual features are obtained from the residuals R_t computed in Eq. 5. They encode the relative relationship between the hypothesis and the data, which can be thought of as the direction for the agent.

The historical features collect how often is a data point used up to the current step. They are initialized to 0 and updated as follow: for each data point x_i used in the current action a_t, the corresponding entry in the historical features τ_i is incremented by 1:

$$\mathcal{H}_t = \{\tau_i\}, \tau_i = \tau_i + 1 \text{ if } x_i \in a_t, i = 1, 2, ..., N. \tag{7}$$

The historical features can provide the agent with information on the actions taken so far, enabling it to keep track of its path through the data.

By encoding the state in this way, the agent has access to both current and historical information, allowing it to reach the goal state. Furthermore, the memory features serve as a director of local navigation, guiding the agent to explore the data features and reach its destination gradually.

4. Experiments

We evaluate the performance of RLSAC on classic tasks, including 2D line fitting and fundamental matrix estimation. The 2D line fitting task serves as a basic benchmark to assess the performance of the robust estimation algorithm and provides visualization of the sampling process. The fundamental matrix estimation task can show the performance of algorithms on complex camera pose estimation task in the real world, which is a critical task in computer vision.

4.1. Settings

RLSAC is built on the widely used reinforcement learning framework SAC-Discrete [13]. As shown in Figure 2, the collected replay buffer is retrieved for off-policy to update the actor and critic network in reinforcement learning.

Since RLSAC uses the inlier ratio as the reward for training, it is an unsupervised method. To explore more images, only one episode with multiple steps is collected per image pair during training. The framework is trained for 100 epochs. During training, the episode termination conditions are set as follows: (i) if the number of inliers is unchanged for $\kappa = 2$ steps; (ii) if the inlier ratio does not exceed the maximum inlier ratio in the episode for $\varsigma = 3$ steps; (iii) if the maximum number of steps $\psi = 15$ is reached. During testing, only the third condition is valid, and each pair of images can be tested in ν episodes.

As with the standard reinforcement learning [13], RLSAC uses probabilistic sampling during training and max sampling during testing. For the EdgeConv module in the policy network, the value of k-nearest neighbors is set to $k = 15$. The network architecture and additional information can be found in the supplementary material. Moreover, we have included ablation experiments to analyze the impact of specific network details, also provided in the supplementary material. Similar to the RANSAC implementation, RLSAC also employs refinement using the inliers of the final best hypothesis as the final model polishing.

The experiments are conducted on a Linux computer with an Intel i7 3.6GHz CPU and an NVIDIA RTX 2080Ti GPU. Our implementation is based on PyTorch.

Table 1. **2D line fitting.** The mAA@0.5° and median error($^\circ$) on various outlier rates are reported.

Method	0.1		0.2		0.3		0.4		0.5		0.6		0.7	
	mAA ↑	Mid. ↓	mAA ↑	Mid. ↓	mAA ↑	Mid. ↓	mAA ↑	Mid. ↓	mAA ↑	Mid. ↓	mAA ↑	Mid. ↓	mAA ↑	Mid. ↓
RANSAC [16]	0.870	0.049	0.863	0.052	0.850	0.056	0.829	0.061	0.796	0.071	0.746	0.087	0.608	0.135
Ours	**0.875**	**0.047**	**0.874**	**0.049**	**0.872**	**0.048**	**0.865**	**0.050**	**0.858**	**0.052**	**0.845**	**0.056**	**0.824**	**0.062**
Ours-0.5	0.849	0.053	0.850	0.055	0.854	0.053	0.858	0.052	**0.858**	**0.052**	**0.864**	**0.050**	**0.849**	**0.054**

4.2. Case Study 1: 2D Line Fitting

The 2D line fitting task is a basic problem in robust estimation, which can visualize the process of robust estimation and assess the robustness quantitatively to different outlier rates. In this task, each data point contains only the coordinates, thus all data points can be expressed as: $\chi = \{[x_i, y_i] \mid i = 1, 2, ..., N\}$, where χ represents the data features $N \times c$, with $c = 2$. Since two points are sufficient to determine a 2D line, the agent can generate a hypothesis by selecting two points from the set of all data points as the minimum set $N \times m$, where $m = 2$. For comparison, the RANSAC [16] algorithm is evaluated with the same task.

To evaluate the performance of RLSAC in the 2D line fitting task, we synthesize $N = 100$ data points for both training and testing. Specifically, a ground truth 2D line is randomly generated in a 10×10 picture. True inliers are then randomly generated on the line based on the set outlier rate. Next, the true inliers are uniformly disturbed within a range of 0.1 around the line. Thus, the inlier threshold could be set to $\varepsilon = 0.1$. Additionally, true outliers are randomly scattered throughout the picture. Moreover, it is possible for true outliers to be located within the inlier region, as is often the case in real-world scenes.

For accurately evaluating and comparing the performance of different methods, we adopt the mean Average Accuracy (mAA) metric in [2]. The angular difference between the estimated line and the ground truth line is used as the error metric, which is used to calculate the mAA metric with a tolerance threshold of 0.5°.

The performance of RLSAC and RANSAC at different outlier rates with 150 iterations is evaluated and the results are presented in Table 1. When the outlier rate is lower than 0.5, RANSAC can achieve a similar performance to RLSAC. This is because the repeated random selection of the minimum set can achieve good performance in simple low outlier rate scenes. However, the performance of RANSAC degrades more quickly than that of RLSAC as the outlier rate increases. This suggests that RLSAC can explore hypotheses closer to the ground truth in noisy scenes.

As the outlier rate increases beyond 0.5, RLSAC can still maintain a low error and high mAA score. This indicates that RLSAC is more robust to disturbances, providing a more stable performance compared to RANSAC.

The qualitative results of RLSAC are shown in Figure 4, which demonstrates its ability to quickly find the better hypothesis even from a bad initial state. This means that the memory features can serve as a director of local navigation, allowing RLSAC to move towards a better state through outputting actions. Moreover, once RLSAC finds a high inlier ratio hypothesis that is close to the ground truth, it continues to explore the local state to further improve the result, rather than randomly transitioning to other states.

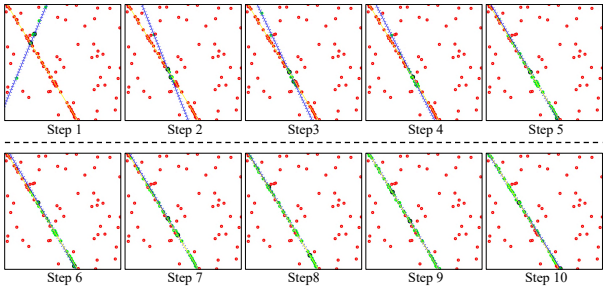

Figure 4. **The qualitative results of RLSAC on 2D line fitting.** The hypothesis converges to the ground truth as the number of steps increases. The green points represent inliers, while the red points represent outliers. The sampled minimum set points are denoted by black edges. The ground truth is represented by a yellow line, while the hypothesis and inlier threshold are represented by blue and dashed lines respectively.

The results of the 2D line fitting task show that RLSAC can effectively utilize both data features and memory features, maintaining robust and stable performance in noisy environments. In addition, it can gradually explore a better hypothesis with the help of memory features.

4.3. Case Study 2: Fundamental Matrix Estimation

The fundamental matrix estimation is a crucial robust estimation task in computer vision, which solves the fundamental matrix by correspondences in a pair of images. In this study, we use the data and settings in CVPR tutorial *RANSAC in 2020* [3], where the correspondences are detected by RootSIFT [1] and matched by the nearest neighbor. The training data comprises 12 scenes, each with 100k image pairs, while the test data includes 2 scenes, with 4950 image pairs. The dataset uses Second Nearest Neighbor (SNN) ratio as the matching score. The correspondences are sorted in descending order by SNN. Then the top $N = 150$ SNN correspondences are selected. Next, each point extracts a 128-dimensional descriptor $desc$ through the SIFT algorithm. Thus, the data points are: $\chi = \left\{ \left[x_1^i, y_1^i, x_2^i, y_2^i, SNN^i, desc_1^i, desc_2^i \right] | i = 1, 2, ..., N \right\} \in \mathbb{R}^{N \times c}$, where $c = 261$. The 8-point solver is used to solve

Table 2. **Fundamental matrix estimation.** The mAA@10° and median error($\epsilon_{\mathbf{R}}$ and $\epsilon_{\mathbf{t}}$) of rotation and the direction of translation at 1k iterations are reported in degrees.

Method	mAA@10° ↑		Median (°) ↓	
	R	**t**	$\epsilon_{\mathbf{R}}$	$\epsilon_{\mathbf{t}}$
RANSAC[16]	0.644	0.488	2.307	5.100
USAC[24]	0.741	0.604	1.036	2.157
MAGSAC++[6]	0.753	0.614	**0.924**	1.895
Ours	**0.760**	**0.622**	0.926	**1.751**

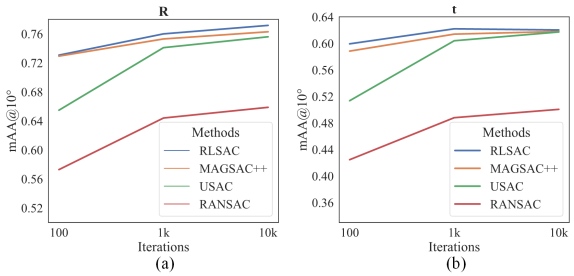

Figure 5. **The mAA@10° at different iterations.** The results on the fundamental matrix estimation task at different iterations.

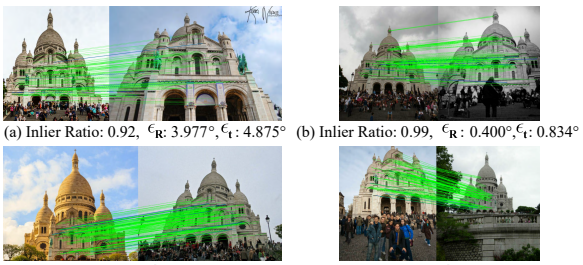

Figure 6. **The qualitative results of RLSAC on the fundamental matrix estimation task.** Inlier rate, rotation and translation errors are reported. The blue lines represent the sampled minimum set points, and the green lines represent the inliers.

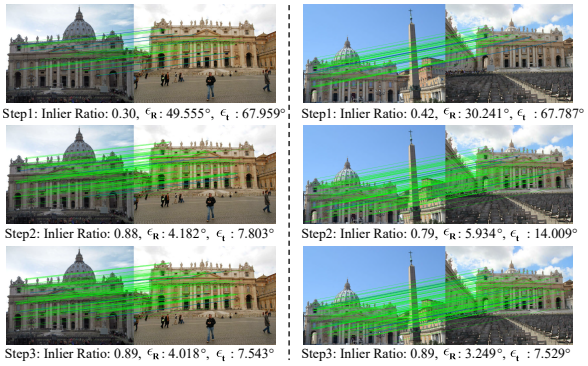

Figure 7. **The step results of RLSAC on the fundamental matrix estimation task.** The meaning of the lines and the evaluation metrics are consistent with Figure 6.

the hypotheses [21]. The inlier threshold of RLSAC is set to $\varepsilon = 4$. The evaluation settings follow [3]. To compare with other methods, RANSAC [16], USAC [24], MAGSAC++ [6] are tested, which use the recommended settings in [3].

As shown in Figure 5, RLSAC outperforms other methods in estimating both the rotation matrix and translation vector. Remarkably, RLSAC can achieve high accuracy with only 100 iterations. Comparisons reveals that RLSAC possesses a higher upper bound in performance than MAGSAC.

The quantitative results of the methods at 1k iterations are presented in Table 2. RLSAC achieves great performance with small errors, a little higher than MAGSAC++[6]. Specifically, RLSAC shows high performance in estimating the direction of the translation vector compared to other methods. Since RLSAC could effectively learn features from the data to exclude noise disturbances, it has significant performance on translation vector estimation.

Figure 6 shows the qualitative results of the fundamental matrix estimation of RLSAC in each scene. The figure demonstrates that RLSAC selects rigid and fixed feature points on buildings as the minimum sets, which can help to find better poses. Furthermore, the step-by-step fundamental matrix estimation results of RLSAC are visualized in Figure 7. It demonstrates a gradual increase in the inlier ratio of the hypothesis and a decrease in the rotation and translation errors. This illustrates that RLSAC can progressively explore the state space to find a better hypothesis.

5. Ablation Study

We perform ablation studies by modifying or removing modules to analyze their effectiveness. The experimental data and settings remain the same as in Section 4.

Robustness and Generalization of RLSAC in 2D Line Fitting: The *Ours-0.5* in Table 1 shows the result of evaluating the robustness and generalization of RLSAC in 2D Line Fitting. We train RLSAC on data with the outlier rate of 0.5 and then test it on various outlier rates. In most cases, *Ours-0.5* can outperform RANSAC. Notably, when the outlier rate exceeds 0.5, *Ours-0.5* performs even better than the model trained at that outlier rate. This is because, in scenes with high outlier rates, the point distribution is closer to a uniform distribution, which lacks the features of a dense distribution. As a result, the model $RLSAC_{High}$ trained on high outlier rates may not have learned the dense distribution in low outlier rates. This results in a mediocre performance of $RLSAC_{High}$ in low outlier rates scenes. Conversely, scenes of low outlier rates exhibit both sparse and dense point distributions in different regions, providing more diverse features for the model $RLSAC_{Low}$ to learn. Consequently, the $RLSAC_{Low}$ can predict better results when the dense distribution of points is occasionally present in the high outlier scene. These results demonstrate that RLSAC can effectively learn the distribution in point groups to classify inliers and outliers.

Effect of Descriptors: To investigate the effect of descriptors with image semantic features, we conduct experiments as shown in Table 3 (a). This suggests that descriptors with

Table 3. **The ablation study results** of RLSAC on fundamental matrix estimation.

	Method	mAA@10° ↑		Median (°) ↓	
		R	**t**	$\epsilon_{\mathbf{R}}$	$\epsilon_{\mathbf{t}}$
(a)	Ours (w/o descriptors)	0.702	0.568	1.400	2.963
	Ours (full, with 128 DIM descriptors)	**0.760**	**0.622**	**0.926**	**1.751**
(b)	Ours (with max sampling in training and max sampling in testing)	0.730	0.591	1.132	2.331
	Ours (with max sampling in training and probabilistic sampling in testing)	0.706	0.531	1.415	2.893
	Ours (with probabilistic sampling in training and probabilistic sampling in testing)	0.720	0.581	1.150	2.508
	Ours (full, with probabilistic sampling in training and max sampling in testing)	**0.760**	**0.622**	**0.926**	**1.751**
(c)	Ours (with N=100 points)	0.727	0.588	1.216	2.464
	Ours (with N=200 points)	0.747	0.604	1.050	2.108
	Ours (with N=300 points)	0.733	0.594	1.180	2.364
	Ours (full, with N=150 points)	**0.760**	**0.622**	**0.926**	**1.751**

semantic features are helpful for RLSAC to effectively learn and sample minimum sets.

Different Sampling Approaches: In Table 3 (b), four different sampling strategies are compared. The results illustrate that the best performance can be achieved by probabilistic sampling during training and max sampling during testing. This is consistent with the sampling strategy for actions in reinforcement learning. Specifically, probabilistic sampling can provide randomness and exploratory during training, while max sampling can output the optimal strategy during testing to improve performance and efficiency.

Number of State Points: In RLSAC, the state points are selected from the top N correspondences sorted by SNN. Different values of N are evaluated in Table 3 (c). The best performance is obtained when $N = 150$. We count the number of correspondences in images at $SNN < 0.8$ recommended in [3], which is also around 150 points. This is reasonable that a smaller number of points may exclude points, which would solve the better hypothesis. While more points will introduce more noise, making it more difficult for RLSAC to learn effective sampling strategies.

6. Discussion

The methods of estimating results directly with neural networks in one shot [23] are not robust and generalized in the scenes that have not been learned. Moreover, their poor interpretability leads to poor practicality in engineering applications. In contrast, traditional methods based on mathematical theory, such as multiple sampling consensus for model estimation and noise covariance matrix estimation in simultaneous localization and mapping (SLAM), offer clear interpretability and well-defined scopes of application. However, many traditional methods cannot be integrated into a learning-based framework due to their non-differentiability.

Therefore, combining these traditional methods with learning-based methods can provide both interpretability and high performance. RLSAC combines sampling consensus with reinforcement learning to avoid differentiation of sampling while better extracting data features and memory features. In addition, RLSAC can be easily transferred to other sampling consensus tasks due to:

- The input data of RLSAC is not limited to coordinates and descriptors. Additional features such as depth estimation information and semantic segmentation information can also be used as input for the policy network.
- RLSAC retains random sampling at the beginning of each episode and each sample result is deduplicated. This provides basic performance for sampling consensus-based tasks.
- The rewards in RLSAC are linked with the evaluation of the hypothesis. Therefore, RLSAC can be extended to other robust estimation tasks, which require different evaluation methods. Moreover, RLSAC does not require differentiation of the sampling process, the hypothesis solving process, and the evaluation process.

In this way, RLSAC enables end-to-end learning that incorporates traditional methods. This allows for consistent optimization objectives across modules.

7. Conclusion

In this paper, we propose RLSAC, a reinforcement learning enhanced sample consensus framework for end-to-end robust estimation. RLSAC models the sampling consensus process as a reinforcement learning task to achieve end-to-end robust estimation. The basic performance of RLSAC is provided by the initial random sampling in each episode. In RLSAC, a new state transition strategy is designed to effectively extract current and historical information, which can guide RLSAC to explore state space. Furthermore, the inlier ratio of the hypothesis is used as a reward to realize unsupervised policy network learning. In experiments, the 2D line fitting task illustrates that RLSAC is robust to various disturbances and exhibits strong generalization. The visualization of the sampling process shows that RLSAC can progressively explore to better models and continues to explore locally. Additionally, the fundamental matrix estimation task demonstrates that RLSAC outperforms other methods in the complex camera pose estimation problem. Notably, RLSAC is not limited to specific tasks and can easily perform end-to-end learning on various sampling consensus-based tasks. In future work, it is worthwhile to explore the integration of traditional and learning-based methods within this framework, while trying to maintain the strengths of each method.

8. Acknowledgement

This work was supported in part by the Natural Science Foundation of China under Grant 62225309, 62073222, U21A20480 and U1913204. The authors gratefully appreciate the contribution of Yiqing Xu from CUMT.

References

[1] Relja Arandjelović and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2911–2918, 2012.

[2] Daniel Barath, Luca Cavalli, and Marc Pollefeys. Learning to Find Good Models in RANSAC. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15723–15732, 2022.

[3] D Barath, TJ Chin, O Chum, D Mishkin, R Ranftl, and J Matas. RANSAC in 2020 tutorial. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[4] Daniel Barath, Maksym Ivashechkin, and Jiri Matas. Progressive NAPSAC: sampling from gradually growing neighborhoods. *arXiv preprint arXiv:1906.02295*, 2019.

[5] Daniel Barath and Jiří Matas. Graph-cut RANSAC. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6733–6741, 2018.

[6] Daniel Barath, Jana Noskova, and Jiri Matas. Marginalizing Sample Consensus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):8420–8432, 2022.

[7] Aritra Bhowmik, Stefan Gumhold, Carsten Rother, and Eric Brachmann. Reinforced feature points: Optimizing feature detection and description for a high-level task. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4948–4957, 2020.

[8] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. DSAC — Differentiable RANSAC for Camera Localization. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2492–2500, 2017.

[9] Eric Brachmann and Carsten Rother. Neural-guided RANSAC: Learning where to sample model hypotheses. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, pages 4322–4331, 2019.

[10] Federico Camposeco, Andrea Cohen, Marc Pollefeys, and Torsten Sattler. Hybrid Camera Pose Estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 136–144, 2018.

[11] Luca Cavalli, Marc Pollefeys, and Daniel Barath. NeFSAC: Neurally Filtered Minimal Samples. In *Computer Vision – ECCV 2022*, pages 351–366, 2022.

[12] Tat-Jun Chin, Hanzi Wang, and David Suter. Robust fitting of multiple structures: The statistical learning approach. In *2009 IEEE 12th International Conference on Computer Vision (ICCV)*, pages 413–420, 2009.

[13] Petros Christodoulou. Soft actor-critic for discrete action settings. *arXiv preprint arXiv:1910.07207*, 2019.

[14] Ondrej Chum and Jiri Matas. Matching with PROSAC-progressive sample consensus. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 220–226, 2005.

[15] Ondřej Chum, Jiří Matas, and Josef Kittler. Locally optimized RANSAC. In *Joint Pattern Recognition Symposium*, pages 236–243, 2003.

[16] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[17] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1861–1870, 2018.

[18] M. Hofer, B. Odehnal, H. Pottmann, T. Steiner, and J. Wallner. 3D shape recognition and reconstruction based on line element geometry. In *Tenth IEEE International Conference on Computer Vision (ICCV'05)*, pages 1532–1538, 2005.

[19] Chaokang Jiang, Guangming Wang, Yanzi Miao, and Hesheng Wang. 3D scene flow estimation on pseudo-lidar: Bridging the gap on estimating point motion. *IEEE Transactions on Industrial Informatics*, pages 7346–7354, 2022.

[20] Hongdong Li. Consensus set maximization with guaranteed global optimality for robust geometry estimation. In *2009 IEEE 12th International Conference on Computer Vision (ICCV)*, pages 1074–1080, 2009.

[21] H Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135, 1981.

[22] Dmytro Mishkin, Jiri Matas, and Michal Perdoch. MODS: Fast and robust method for two-view matching. *Computer Vision and Image Understanding*, 141:81–93, 2015.

[23] Omid Poursaeed, Guandao Yang, Aditya Prakash, Qiuren Fang, Hanqing Jiang, Bharath Hariharan, and Serge Belongie. Deep Fundamental Matrix Estimation without Correspondences. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018.

[24] Rahul Raguram, Ondrej Chum, Marc Pollefeys, Jiri Matas, and Jan-Michael Frahm. USAC: A Universal Framework for Random Sample Consensus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):2022–2038, 2013.

[25] Philip Hilaire Torr, Slawomir J Nasuto, and John Mark Bishop. Napsac: High noise, high dimensional robust estimation-it's in the bag. In *British Machine Vision Conference (BMVC)*, page 3, 2002.

[26] P. H. S. Torr and A. Zisserman. MLESAC: A New Robust Estimator with Application to Estimating Image Geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000.

[27] Giang Truong, Huu Le, Erchuan Zhang, David Suter, and Syed Zulqarnain Gilani. Unsupervised Learning for Maximum Consensus Robust Fitting: A Reinforcement Learning Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 3890–3903, 2022.

[28] Guangming Wang, Jiahao Qiu, Yanfeng Guo, and Hesheng Wang. Fusionnet: Coarse-to-fine extrinsic calibration network of lidar and camera with hierarchical point-pixel fusion. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8964–8970, 2022.

[29] Guangming Wang, Shuaiqi Ren, and Hesheng Wang. Unsupervised learning of optical flow with non-occlusion from geometry. *IEEE Transactions on Intelligent Transportation Systems*, 23(11):20850–20859, 2022.

[30] Guangming Wang, Xiaoyu Tian, Ruiqi Ding, and Hesheng Wang. Unsupervised learning of 3d scene flow from monocular camera. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4325–4331, 2021.

[31] Guangming Wang, Xinrui Wu, Shuyang Jiang, Zhe Liu, and Hesheng Wang. Efficient 3d deep lidar odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5749–5765, 2022.

[32] Guangming Wang, Xinrui Wu, Zhe Liu, and Hesheng Wang. Pwclo-net: Deep lidar odometry in 3d point clouds using hierarchical embedding mask optimization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 15910–15919, 2021.

[33] Guangming Wang, Jiquan Zhong, Shijie Zhao, Wenhua Wu, Zhe Liu, and Hesheng Wang. 3D hierarchical refinement and augmentation for unsupervised learning of depth and pose from monocular video. *IEEE Transactions on Circuits and Systems for Video Technology*, 33(4):1776–1786, 2022.

[34] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

[35] Tong Wei, Yash Patel, Jiri Matas, and Daniel Barath. Fully Differentiable RANSAC. *arXiv preprint arXiv:2212.13185*, 2022.

[36] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2666–2674, 2018.

[37] Jiahui Zhang, Dawei Sun, Zixin Luo, Anbang Yao, Lei Zhou, Tianwei Shen, Yurong Chen, Long Quan, and Hongen Liao. Learning two-view correspondences and geometry using order-aware network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5845–5854, 2019.