

Deep State-Space Monocular Visual Odometry with Learnable Kalman Filtering

Mingyue Wang^(✉) and Yixuan Wu

School of Software, Tiangong University, Tianjin 300387, China
2311310310@tiangong.edu.cn

Abstract. Monocular visual odometry is important in autonomous driving, robotics, and related fields, and has attracted increasing attention in computer vision. Traditional geometric methods and end-to-end deep learning methods have achieved promising results in monocular visual odometry, but they still face limitations in temporal consistency, uncertainty representation, and abnormal observation handling. To overcome these limitations, this paper proposes a monocular visual odometry method that combines deep temporal features with a Kalman filtering module based on a state-space model. By explicitly modeling the temporal evolution of motion states in the state space, the method introduces continuity constraints into the pose estimation process. Meanwhile, the state transition matrix and the covariance matrices for process and measurement noise are estimated by neural networks, allowing the system to adaptively balance prediction and observation during state update. Experimental results on the KITTI VO dataset show that, compared with the baseline method, the proposed model improves pose estimation robustness and adaptability to incomplete observations, which verifies the effectiveness of combining classical filtering theory with deep learning for monocular visual odometry.

Keywords: Monocular Visual Odometry, Computer Vision, Kalman Filter, Deep Learning.

1 Introduction

Visual odometry aims to continuously estimate the motion state of a camera or platform from sensor observations and is a key technology in autonomous driving, mobile robotics, and related fields [1]. Compared with multi-sensor solutions, monocular visual odometry has advantages such as low cost and flexible deployment. However, without explicit depth and scale information, monocular visual odometry must estimate camera motion solely from image sequences, making the task particularly challenging. Therefore, how to estimate camera motion from monocular images in a stable and accurate manner remains an important research problem in computer vision and robotics.

Traditional visual odometry methods are mainly based on multiview geometry and motion constraints, and estimate camera motion through a sequence of modules such as feature extraction and matching, pose estimation, and backend optimization [1, 2].

These methods can achieve good performance in scenes with rich textures and limited viewpoint changes, but they are sensitive to scene conditions. In addition, they rely on manually designed multi-module pipelines and careful parameter tuning, which makes the system complex and limits its generalization ability across different datasets and application scenarios.

To improve system adaptability and reduce manual design, researchers have proposed end-to-end visual odometry methods based on deep learning [3–5] in recent years. These methods avoid explicit feature matching and geometric solving, and show a certain degree of robustness in complex scenes. However, the temporal dependencies in pose estimation are mainly captured implicitly by the network architecture. As a result, the estimated results are difficult to remain temporally continuous and consistent. At the same time, existing methods usually do not model predictive uncertainty, making it difficult for the model to adaptively balance current observations and historical information under degraded or missing observations. Therefore, it is necessary to introduce an estimation mechanism with explicit state evolution and uncertainty modeling on top of deep learning methods [6–8], so as to compensate for the lack of structural constraints and stability in purely data-driven modeling.

Based on the above analysis, this paper proposes a monocular visual odometry method that combines deep temporal features with a state-space model. The method constructs noisy observations from image features and performs recursive state estimation with a differentiable Kalman filter, while neural networks adaptively estimate the state transition and noise covariance parameters. By explicitly modeling state evolution and uncertainty, the method introduces structured constraints into an end-to-end framework. Comparative experiments, robustness tests under random Gaussian noise, and ablation studies on the KITTI VO dataset verify the effectiveness of the proposed method.

In summary, the main contributions of this paper are as follows:

- By introducing a differentiable Kalman filtering mechanism, the proposed method integrates a state-space physical model with deep temporal feature learning in an end-to-end trainable framework, enabling explicit modeling of motion-state dynamics for monocular visual odometry.
- Through learnable uncertainty modeling, the method adaptively balances prediction and observation, suppresses the interference of noise in state estimation, improves system robustness, and enhances estimation stability under degraded observations.
- The explicit recursive filtering process improves model interpretability, makes the fusion between prediction and observation analytically traceable, and provides a clearer basis for understanding model behavior and stability.

2 Related Work

This section reviews related research in monocular visual odometry and discusses the core ideas of different technical routes as well as their methodological differences. Existing studies can generally be divided into three categories: traditional methods based on geometric constraints, end-to-end methods based on deep learning, and hybrid

methods that combine deep learning with physical models, which have emerged in recent years.

2.1 Geometry-based Visual Odometry

Visual odometry methods based on geometric constraints rely on multiview geometry and recover camera motion through feature matching or photometric consistency constraints, typically combined with geometric estimation and optimization procedures [1, 2, 9, 10]. Overall, such methods have clear geometric interpretations and well-defined optimization frameworks, and can achieve high accuracy in controlled environments. However, they are sensitive to complex environments and lack data-driven adaptability, which limits their robustness across diverse scenarios.

2.2 Learning-based Visual Odometry

Recent studies have explored the use of convolutional neural networks for camera motion estimation from image sequences. Such methods usually formulate visual odometry as an end-to-end regression problem, using deep networks to extract motion features, model temporal dependencies, and predict relative poses from image sequences [3–5]. Compared with traditional geometric methods, these learning-based methods can automatically extract expressive visual features from data and show stronger robustness in complex scenes. However, they lack explicit state-space modeling and error propagation mechanisms, making it difficult to impose structured constraints on historical information and correct accumulated errors, which may lead to trajectory drift over time.

2.3 Deep Learning with Physical Models

To address the above issues, some recent studies begin to combine deep learning with physical models or state-space frameworks [6–8], and introduce structured dynamic modeling and recursive estimation mechanisms into data-driven methods. Such methods perform structured modeling of the system in the state space, thereby establishing a connection between data-driven learning and physical constraints. Compared with purely deep learning methods, these approaches retain an explicit state evolution process during inference through state-space modeling, and achieve adaptive fusion of prediction and observation through uncertainty modeling, thereby alleviating error accumulation to some extent and improving estimation stability. On this basis, this paper further tightly integrates deep temporal features with a differentiable Kalman filtering framework. The network adaptively learns the state transition and noise parameters, and realizes structured fusion of prediction and observation in end-to-end training, thereby more effectively suppressing error accumulation and improving estimation robustness under degraded observations.

3 My Method

This section describes the proposed visual odometry framework in detail. The proposed framework contains three main modules: a visual feature extraction module based on a convolutional neural network, a temporal modeling module based on a recurrent neural network, and an embedded Kalman filtering recursion module. Its goal is to estimate the relative camera pose between adjacent time steps and thereby obtain the motion trajectory representation of the whole sequence. Unlike traditional end-to-end visual odometry methods that directly regress the pose at each time step, the proposed method takes the output of the deep network as the source of noisy observations and introduces a recursive estimation process in the state space to perform structured modeling of the hidden motion state. The overall framework is shown in Fig. 1.

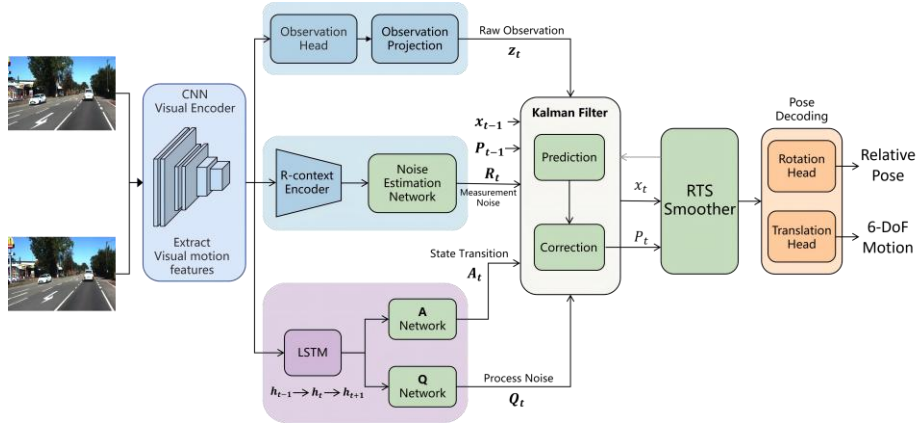


Fig. 1. Overall framework of the proposed method. Consecutive image pairs are processed to estimate camera motion via a recurrent network and a Kalman filter.

First, a convolutional neural network processes adjacent image pairs and generates a high-dimensional observation representation at each time step. Through a learnable observation encoder, this representation is mapped into the state space to produce the observation input required by Kalman filtering, $\mathbf{z}_t \in \mathbb{R}^D$, where $D = 128$. A recurrent neural network is then used to capture temporal dependencies across time steps and support subsequent state estimation.

Under an explicit state transition model and noise assumptions, the Kalman filtering module recursively estimates the hidden motion state $\mathbf{x}_t \in \mathbb{R}^D$, and characterizes motion evolution and observation reliability through prediction and update steps. Finally, the filtered state is mapped back to the six-degree-of-freedom pose space through a learnable state decoder, yielding the final pose output

$$\hat{\mathbf{p}}_t^{KF} = g(\mathbf{x}_t) \in \mathbb{R}^6. \quad (1)$$

3.1 Image Preprocessing and Convolutional Feature Extraction

Given an image sequence of length T , denoted by $\{\mathbf{I}_t\}_{t=1}^T$, each image frame is first subjected to a unified preprocessing procedure, including resizing or cropping, tensor conversion, and normalization, so as to ensure consistency in spatial resolution and value distribution across the input data.

To explicitly model the motion information between adjacent frames, this paper adopts a frame-pair-based input form. Specifically, two consecutive frames are concatenated along the channel dimension to form a 6-channel input:

$$\mathbf{X}_t = \text{Concat}(\mathbf{I}_t, \mathbf{I}_{t+1}), \quad \mathbf{X}_t \in \mathbb{R}^{6 \times H \times W}. \quad (2)$$

The concatenated image pairs are then fed into a convolutional neural network for feature extraction. The convolutional encoder is designed or pretrained based on the idea of optical flow modeling [11] to enhance the representation of motion information between adjacent frames. The convolutional network consists of multiple convolutional blocks with downsampling, which progressively reduce the spatial resolution and increase the feature dimension. Let the convolutional feature at time step t be

$$\mathbf{f}_t = \Phi_{\text{CNN}}(\mathbf{X}_t), \quad (3)$$

where $\Phi_{\text{CNN}}(\cdot)$ denotes the convolutional encoder, and $\mathbf{f}_t$ is a high-dimensional visual feature vector.

The resulting feature sequence $\{\mathbf{f}_t\}_{t=1}^T$ preserves the temporal order and serves as the input to the subsequent recurrent neural network for temporal modeling. In addition, the convolutional feature is used to generate the noisy observation $\mathbf{z}_t$, which is fed into the Kalman filter as the observation input. The feature is also used to construct the measurement noise covariance $\mathbf{R}_t$, which provides observation uncertainty information for the subsequent state-space model.

3.2 Temporal Modeling with Recurrent Neural Network

The visual feature sequence $\{\mathbf{f}_t\}_{t=1}^T$ extracted by the convolutional network preserves temporal order information, but it does not directly model the dynamic dependencies across time steps. To address this issue, this paper introduces a recurrent neural network to perform temporal modeling on the feature sequence, so as to capture long-term dependencies in the motion process.

During the temporal modeling stage, a multi-layer long short-term memory network (LSTM) [12] is used to recursively process the convolutional features, and its hidden state update can be expressed as

$$\mathbf{h}_t = \text{LSTM}(\mathbf{f}_t, \mathbf{h}_{t-1}), \quad (4)$$

where $\mathbf{h}_t$ denotes the hidden state vector at time step t . This hidden state serves as a compact representation of the temporal context and encodes the current observation together with its historical information.

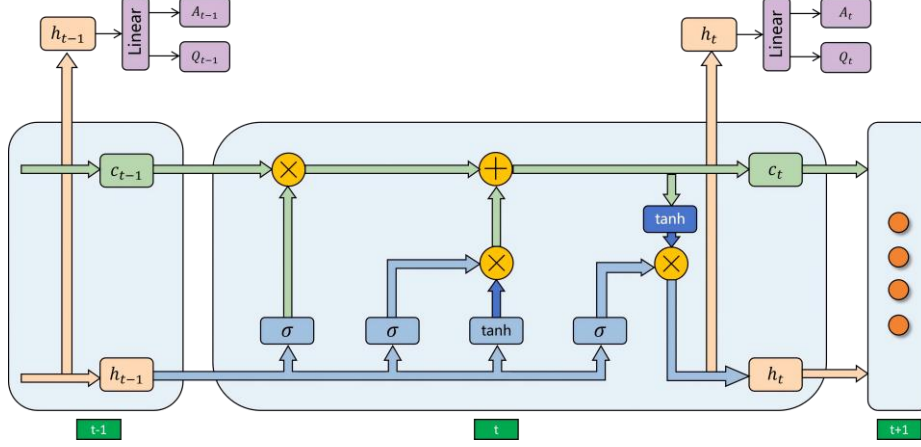


Fig. 2. Temporal modeling and dynamic parameter generation using LSTM. The recurrent network produces time-dependent state transition and process noise parameters for Kalman filtering.

Based on the LSTM output state $\mathbf{h}_t$, this paper generates the dynamic parameters in the state-space model through several learnable linear mappings. Specifically, these mappings are used to predict the state transition matrix $\mathbf{A}_t$ and the process noise covariance $\mathbf{Q}_t$, thereby providing a prior dynamic model for the subsequent filtering process. The relationship between the LSTM and the state-space parameters is shown in Fig. 2.

Through the above design, the deep network extracts high-dimensional feature representations from visual inputs and generates the key parameters in the state-space model accordingly, while the filtering process itself is carried out within an explicit probabilistic framework.

3.3 Kalman Filter Integration

This paper embeds Kalman filtering [13] into the overall framework as a differentiable recursive module to estimate the latent motion state at each time step. In this process, the observation input to the filter is the encoded high-dimensional observation representation, while the final pose output is obtained by decoding the filtered state. The overall workflow of the Kalman filtering module is illustrated in Fig. 3.

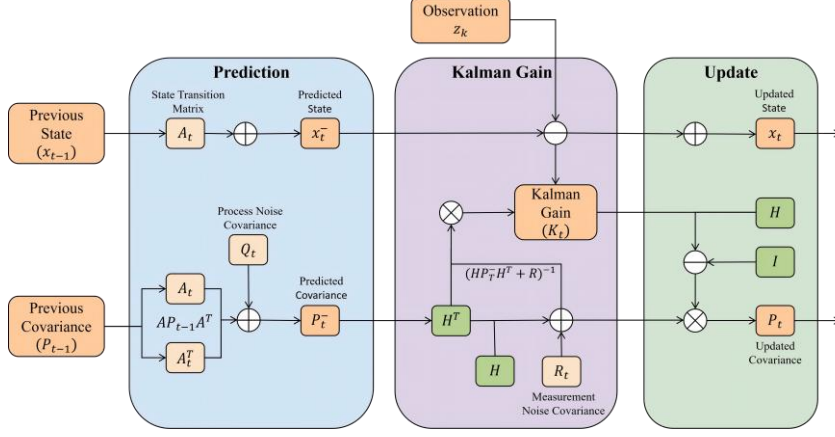


Fig. 3. Overall workflow of the Kalman filtering module.

Prediction: State Propagation. For each time step $t \geq 1$, the model is given a time-dependent state transition matrix $\mathbf{A}_t$ and process noise covariance $\mathbf{Q}_t$. On this basis, the linear Gaussian dynamical model performs prediction of the state and covariance, as shown in Eqs. (5)–(6).

$$\mathbf{x}_{t|t-1} = \mathbf{A}_t \mathbf{x}_{t-1}, \quad (5)$$

$$\mathbf{P}_{t|t-1} = \mathbf{A}_t \mathbf{P}_{t-1} \mathbf{A}_t^\top + \mathbf{Q}_t. \quad (6)$$

Update: Observation Correction. At time step t , the model is given the high-dimensional observation $\mathbf{z}_t$ and its corresponding measurement noise covariance $\mathbf{R}_t$. Based on these, the innovation covariance and the Kalman gain can be defined, and the predicted state is then updated, as shown in Eqs. (7)–(9). Meanwhile, to ensure numerical stability, this paper adopts the Joseph form to update the covariance, as shown in Eq.(10):

$$\mathbf{S}_t = \mathbf{H} \mathbf{P}_{t|t-1} \mathbf{H}^\top + \mathbf{R}_t \quad (7)$$

$$\mathbf{K}_t = \mathbf{P}_{t|t-1} \mathbf{H}^\top \mathbf{S}_t^{-1} \quad (8)$$

$$\mathbf{x}_t = \mathbf{x}_{t|t-1} + \mathbf{K}_t (\mathbf{z}_t - \mathbf{H} \mathbf{x}_{t|t-1}) \quad (9)$$

$$\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t \mathbf{H}) \mathbf{P}_{t|t-1} (\mathbf{I} - \mathbf{K}_t \mathbf{H})^\top + \mathbf{K}_t \mathbf{R}_t \mathbf{K}_t^\top \quad (10)$$

RTS Smoothing. The state estimate $\mathbf{x}_t$ and its covariance $\mathbf{P}_t$ obtained through the above prediction and update steps are the results of the forward recursion of Kalman filtering. However, when the full sequence is available, the state estimates at later time steps can also provide additional information for the state estimate at the current time step. To further improve the accuracy of state estimation, this paper introduces the Rauch-Tung-Striebel (RTS) smoothing method [14] after Kalman filtering. RTS smoothing performs backward recursion based on the filtering results and corrects the

state estimate at the current time step by incorporating the state estimates at future time steps, thereby obtaining more accurate state estimates. Let $\mathbf{x}_t^s$ and $\mathbf{P}_t^s$ denote the smoothed state estimate and its covariance matrix, respectively. Then the smoothing process can be written as:

$$\mathbf{G}_t = \mathbf{P}_t \mathbf{A}_{t+1}^\top \mathbf{P}_{t+1|t}^{-1}, \quad (11)$$

$$\mathbf{x}_t^s = \mathbf{x}_t + \mathbf{G}_t (\mathbf{x}_{t+1}^s - \mathbf{x}_{t+1|t}), \quad (12)$$

$$\mathbf{P}_t^s = \mathbf{P}_t + \mathbf{G}_t (\mathbf{P}_{t+1}^s - \mathbf{P}_{t+1|t}) \mathbf{G}_t^\top. \quad (13)$$

Through the above backward recursion process, the state estimate can make use of not only the current and past information, but also the state estimates at later time steps, thereby further reducing the estimation error over time. It should be noted that RTS smoothing uses future estimated states rather than future ground-truth poses, and therefore does not introduce label information during inference. Since it relies on information from later time steps, it is used here as a full-sequence refinement module.

3.4 Pose Decoding and Loss

The outputs of Kalman filtering and RTS smoothing are still latent states and cannot be directly used as six-degree-of-freedom relative poses. To address this issue, a decoding function $g(\cdot)$ is introduced to map the state representation to the pose space. For the filtered state $\mathbf{x}_t$ and the smoothed state $\mathbf{x}_t^s$ at time step t , the pose estimates are defined as

$$\hat{\mathbf{p}}_t = g(\mathbf{x}_t), \quad \hat{\mathbf{p}}_t^s = g(\mathbf{x}_t^s), \quad g: \mathbb{R}^D \rightarrow \mathbb{R}^6. \quad (14)$$

where $\hat{\mathbf{p}}_t$ and $\hat{\mathbf{p}}_t^s$ denote the relative pose estimates obtained from the filtering result and the RTS smoothing result, respectively. Let the ground-truth relative pose be $\mathbf{p}_t = [\mathbf{r}_t, \mathbf{u}_t]$, where $\mathbf{r}_t$ and $\mathbf{u}_t$ denote the relative rotation and relative translation, respectively. Considering that the training process needs to constrain both the pose regression error and the consistency of observation update and trajectory evolution, the loss function consists of three terms.

To constrain the relative pose errors of the filtering result and the RTS smoothing result, unified relative pose supervision is adopted. Let $b \in \{f, s\}$ denote the filtering result and the smoothing result, respectively, where $\hat{\mathbf{p}}_t^f = \hat{\mathbf{p}}_t$. Then the relative pose loss is defined as

$$\mathcal{L}_{\text{relpose}} = \frac{1}{T} \sum_{t=1}^T \sum_{b \in \{f, s\}} w_b \left[\lambda_r^b \rho \left(d_{\text{geo}}(\hat{\mathbf{R}}_t^b, \mathbf{R}_t^{\text{gt}}) \right) + \lambda_u^b \rho(\hat{\mathbf{u}}_t^b - \mathbf{u}_t) \right], \quad (15)$$

where $\rho(\cdot)$ denotes the Smooth L1 loss, and $d_{\text{geo}}(\cdot, \cdot)$ denotes the geodesic distance between rotation matrices. $\hat{\mathbf{R}}_t^b$ and $\mathbf{R}_t^{\text{gt}}$ are the rotation matrices corresponding to the b -th output and the ground-truth pose, respectively, and $w_f = 1$, $w_s = \alpha$. The relative pose supervision jointly constrains the consistency of both the forward filtering result and the RTS smoothing result with the ground-truth pose. The rotation error and translation

error are modeled separately, which helps reduce the effect of scale differences on optimization.

In addition to pose supervision, the observation representation corresponding to the observation update and its measurement noise covariance estimation are also constrained during training. To constrain the matching relationship between the observation residual and the uncertainty estimation, a negative log-likelihood constraint based on the innovation is further introduced in the observation update stage. Let the innovation at time step t be $\mathbf{y}_t = \mathbf{z}_t - \mathbf{H}\mathbf{x}_t|t-1$, and let $\mathbf{S}_t$ be the corresponding innovation covariance. Then, after omitting the constant term, the observation loss can be written as

$$\mathcal{L}_{\text{obs}} = \frac{1}{2T} \sum_{t=1}^T (\mathbf{y}_t^\top \mathbf{S}_t^{-1} \mathbf{y}_t + \log |\mathbf{S}_t|). \quad (16)$$

This loss term acts directly on the filtering update stage and is used to constrain the statistical consistency between the innovation magnitude and the covariance estimation.

The above two constraints mainly act on individual time steps and are still insufficient to fully suppress the accumulated error over long sequences. To constrain the temporal continuity of the whole trajectory, the predicted relative pose sequence is accumulated over time to obtain the corresponding absolute trajectory, and a consistency constraint is introduced at the trajectory level. Let $\Phi(\cdot)$ denote the accumulation operator that recovers absolute poses from relative poses. Then the consistency loss can be expressed as

$$\mathcal{L}_{\text{cons}} = \frac{1}{T} \sum_{t=1}^T [\lambda_r^c \rho(d_{\text{geo}}(\Phi_R(\hat{\mathbf{p}}_{1:t}), \Phi_R(\mathbf{p}_{1:t}))) + \lambda_u^c \rho(\Phi_u(\hat{\mathbf{p}}_{1:t}) - \Phi_u(\mathbf{p}_{1:t}))]. \quad (17)$$

This constraint is used to suppress long-term drift and maintain the geometric consistency and motion continuity of the recovered trajectory.

By combining the above three constraints, the final training objective can be written as

$$\mathcal{L} = \mathcal{L}_{\text{relpose}} + \gamma \mathcal{L}_{\text{obs}} + \beta \mathcal{L}_{\text{cons}}, \quad (18)$$

where α controls the contribution of the smoothing result in the relative pose supervision through w_s , and γ and β control the weights of the observation constraint and the trajectory consistency constraint, respectively. The three loss terms correspond to relative pose supervision, observation update constraint, and trajectory consistency constraint, respectively, and together form the final training objective.

4 Experiment

The proposed method is implemented based on the PyTorch framework [15], and all experiments are conducted on an NVIDIA RTX A6000 GPU. The experiments are performed on the KITTI Visual Odometry dataset [16], where sequences 00, 01, 02, 03,

04, and 08 are used for training, sequences 05, 06, and 07 are used for validation, and sequences 09 and 10 are used for testing.

The input data are constructed using a sliding-window scheme, where each sample contains 9 consecutive image frames. All input images are uniformly resized to a resolution of 640×192 . During training, data augmentation is performed using brightness, contrast, and gamma perturbations. The AdamW optimizer [17] is used together with a cosine annealing learning rate schedule. The learning rate for the backbone is set to 1×10^{-4} , and that for the state-space branch is set to 3×10^{-5} . The weight decay is set to 1×10^{-4} , the batch size is 64, and validation is performed after each epoch.

During testing, the model performance is quantitatively evaluated using the commonly adopted KITTI metrics of translational error and rotational error.

4.1 Comparison

To evaluate the pose estimation performance of the proposed method, this paper compares the proposed method with several representative visual odometry methods on KITTI sequences 09 and 10, with DeepVO used as the primary learning-based baseline. For qualitative trajectory comparison, DeepVO is selected as the main learning-based baseline. Table 1 presents the quantitative results, while Fig. 4 and Fig. 5 show the trajectory comparison and the position changes along the three axes, respectively.

Table 1. Quantitative comparison of different methods on KITTI sequences 09 and 10.

Methods	Seq. 09		Seq. 10	
	t_{rel} (%)	r_{rel} (deg/m)	t_{rel} (%)	r_{rel} (deg/m)
ORB-SLAM [2]	15.30	0.26	3.68	0.48
Zhou et al. [4]	17.84	0.0678	37.91	0.1778
Zhan et al. [5]	11.93	0.0391	12.45	0.0346
DeepVO [3]	11.0523	0.0390	11.1915	0.0260
Ours	9.6302	0.0261	8.7219	0.0342

As shown in Table 1, the proposed method achieves substantially lower translational errors than DeepVO on both test sequences, indicating better overall accuracy. In terms of rotational error, the proposed method performs better on sequence 09, while it is slightly worse than DeepVO on sequence 10. This suggests that the improvement of the proposed method is more evident in translational estimation than in rotational accuracy. These results indicate that the proposed method has good robustness and generalization ability across different scenes.

As shown in Fig. 4, the predicted trajectory of the proposed method is overall closer to the ground-truth trajectory, with smaller drift in turning regions and over long accumulated distances, whereas DeepVO exhibits more evident error accumulation.

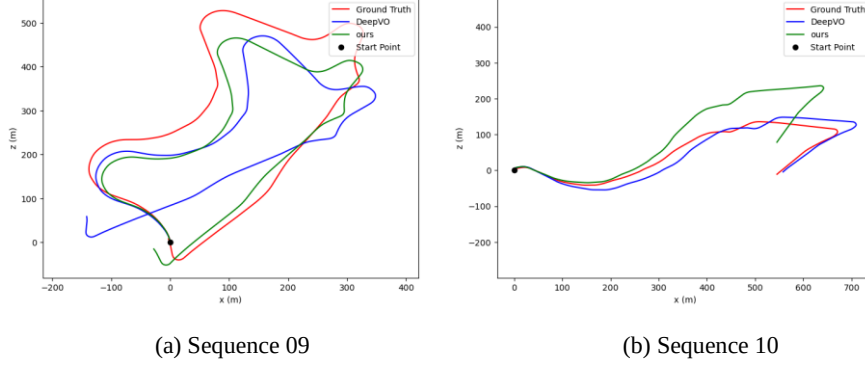


Fig. 4. Trajectory comparison between DeepVO and the proposed method on KITTI sequences 09 and 10.

Further observations from Fig. 5 show that the proposed method produces smoother position variations along the x , y , and z axes over multiple segments, and its overall trend is more consistent with the ground-truth trajectory. This indicates that the method has better stability in local temporal modeling, which is closely related to the introduced state-space modeling mechanism.

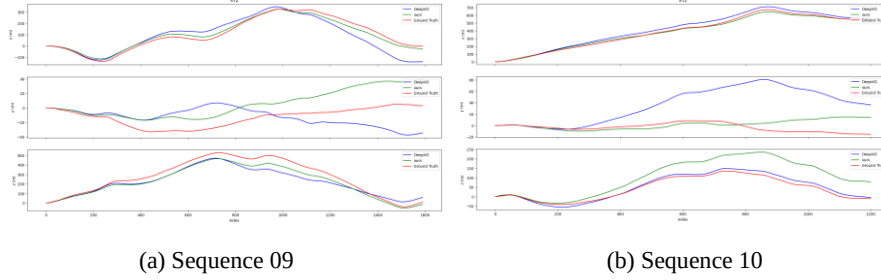


Fig. 5. Comparison of position components along the x , y , and z axes on KITTI sequences 09 and 10.

4.2 Robustness under Noisy Observations

In practical application scenarios, visual observations are inevitably affected by noise. To evaluate the robustness of the model under degraded observation conditions, Gaussian noise with different intensities is added to the input images during testing, and comparative experiments are conducted on KITTI sequences 09 and 10. Table 2 and Table 3 present the corresponding quantitative results, respectively.

The results show that, as the noise intensity increases, the errors of both methods rise, but the performance of the proposed method changes more gradually, indicating better stability. On sequence 09, the proposed method achieves lower translational and rotational errors than DeepVO across all noise levels. On sequence 10, the proposed method consistently maintains a clear advantage in translational error, while its

rotational error is slightly higher than that of DeepVO, although the overall trend remains stable.

Overall, the proposed method maintains stable pose estimation performance under noise interference, demonstrating good robustness. This result is related to the introduced uncertainty modeling mechanism, which can adaptively adjust the weight of state updates.

Table 2. Quantitative comparison under noisy observations on KITTI sequence 09.

Noise Range	DeepVO		Ours	
	t_{rel} (%)	r_{rel} (deg/m)	t_{rel} (%)	r_{rel} (deg/m)
0-0.02	11.073	0.03918	9.677	0.02665
0-0.05	11.506	0.04115	9.930	0.02915
0-0.10	12.645	0.04629	10.549	0.03540

Table 3. Quantitative comparison under noisy observations on KITTI sequence 10.

Noise Range	DeepVO		Ours	
	t_{rel} (%)	r_{rel} (deg/m)	t_{rel} (%)	r_{rel} (deg/m)
0-0.02	11.291	0.02613	8.727	0.03469
0-0.05	11.800	0.02733	8.574	0.03552
0-0.10	13.498	0.03197	8.554	0.03720

4.3 Ablation Study

To analyze the effect of each module on model performance, ablation studies are conducted on KITTI sequences 09 and 10, including two settings: removing the LSTM temporal modeling module and replacing the learnable noise covariances with fixed values. Table 4 presents the quantitative results. From the overall results, the complete model obtains the best results on both sequences, indicating that all modules contribute positively to the performance improvement.

Table 4. Ablation studies on KITTI sequences 09 and 10.

Seq.	No LSTM		Fixed Q/R		Ours	
	t_{rel} (%)	r_{rel} (deg/m)	t_{rel} (%)	r_{rel} (deg/m)	t_{rel} (%)	r_{rel} (deg/m)
09	28.7618	0.1055	18.2846	0.0598	9.6302	0.0261
10	35.1767	0.1618	12.3910	0.0502	8.7219	0.0342
mean	31.9693	0.1336	15.3378	0.0550	9.1761	0.0302

No LSTM. As shown in the table, removing the LSTM leads to a clear degradation in model performance, with both translational and rotational errors increasing substantially. This indicates that temporal modeling is crucial for exploiting continuous motion information.

Fixed Q/R . When the process noise covariance matrix and the measurement noise covariance matrix are set as fixed parameters, the model performance also degrades. Although the overall performance remains better than that of No LSTM, it still falls short of the full model, which indicates that adaptive noise modeling helps improve estimation accuracy.

In summary, both LSTM-based temporal modeling and adaptive estimation of the noise covariance play key roles in improving performance, and the best results are achieved when the two are combined.

5 Conclusion

This paper proposes a monocular visual odometry method that combines deep temporal features with a state-space model. The method uses a convolutional neural network to extract visual features and generate the measurement noise covariance $\mathbf{R}_t$, uses a recurrent neural network to generate the state transition matrix $\mathbf{A}_t$ and the process noise covariance $\mathbf{Q}_t$, and combines differentiable Kalman filtering with RTS smoothing to achieve recursive estimation and refinement of the motion state. Compared with purely end-to-end regression methods, this framework explicitly introduces state evolution constraints and uncertainty modeling, thereby enhancing the interpretability, temporal consistency, and robustness of the model. Experimental results on the KITTI dataset show that the proposed method outperforms the comparison method in overall trajectory estimation performance. These results verify the effectiveness of combining classical filtering theory with deep learning methods for monocular visual odometry and These results also indicate that the method has good application potential in complex dynamic scenes, while its performance in broader and more diverse dynamic scenarios requires further investigation.

References

1. Scaramuzza, D., Fraundorfer, F.: Visual Odometry [Tutorial]. *IEEE Robot. Autom. Mag.* **18**(4), 80–92 (2011)
2. Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.: ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Trans. Robot.* **31**(5), 1147–1163 (2015)
3. Wang, S. et al.: DeepVO: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2043–2050 (2017)
4. Zhou, T. et al.: Unsupervised Learning of Depth and Ego-Motion from Video. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6612–6619 (2017)

5. Zhan, H. et al.: Unsupervised Learning of Monocular Depth Estimation and Visual Odometry with Deep Feature Reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 340–349 (2018)
6. Haarnoja, T. et al.: Backprop KF: Learning Discriminative Deterministic State Estimators. arXiv preprint arXiv:1605.07148 (2016)
7. Karl, M. et al.: Deep Variational Bayes Filters: Unsupervised Learning of State Space Models from Raw Data. arXiv preprint arXiv:1605.06432 (2016)
8. Becker, P. et al.: Recurrent Kalman Networks: Factorized Inference in High-Dimensional Deep Feature Spaces. In: Proceedings of the 36th International Conference on Machine Learning, vol. 97, pp. 544–552. PMLR (2019)
9. Engel, J., Sturm, J., Cremers, D.: Semi-dense Visual Odometry for a Monocular Camera. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 1449–1456 (2013)
10. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: Large-Scale Direct Monocular SLAM. In: Fleet, D. et al. (eds.) Computer Vision – ECCV 2014, vol. 8690, pp. 834–849. Springer, Cham (2014)
11. Fischer, P. et al.: FlowNet: Learning Optical Flow with Convolutional Networks. arXiv preprint arXiv:1504.06852 (2015)
12. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. *Neural Comput.* **9**(8), 1735–1780 (1997)
13. Kalman, R.E.: A new approach to linear filtering and prediction problems. *J. Basic Eng.* **82**(1), 35–45 (1960)
14. Rauch, H.E., Tung, F., Striebel, C.T.: Maximum likelihood estimates of linear dynamic systems. *AIAA J.* **3**(8), 1445–1450 (1965)
15. Paszke, A. et al.: PyTorch: An Imperative Style, High-Performance Deep Learning Library. arXiv preprint arXiv:1912.01703 (2019)
16. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? The KITTI vision benchmark suite. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3354–3361 (2012)
17. Loshchilov, I., Hutter, F.: Decoupled Weight Decay Regularization. arXiv preprint arXiv:1711.05101 (2019)