Supplementary Material for QuaDreamer: Controllable Panoramic Video Generation for Quadruped Robots

Sheng Wu[1,*], **Fei Teng**[1,*], **Hao Shi**[2,3,*], **Qi Jiang**[2], **Kai Luo**[1], **Kaiwei Wang**[2], **Kailun Yang**[1,†]
[1]Hunan University [2]Zhejiang University [3]Nanyang Technological University

1 Experimental Details

Details. We conducted the full training on a single NVIDIA A6000 48G GPU for 320 epochs, totaling $200,000$ training steps, with the entire process taking 78 hours. The experiment utilized the accelerate library to streamline the training workflow, employing a batch size of 2 and gradient accumulation steps of 1 to optimize memory utilization. We set an initial learning rate of $3e-5$ with a linear warm-up over the first 750 steps to stabilize early-stage convergence. Additionally, mixed-precision training (FP16) was enabled, significantly improving training speed and reducing memory consumption while maintaining model accuracy. During generation, frames are sampled using the DPM-Solver [1] scheduler for 30 steps. Other comparative models were trained and inferred using the same settings to ensure a fair evaluation. We use the pre-trained Stable Video Diffusion [2] as the base generative model, combined with the frozen-parameter CameraCtrl camera encoder [3] to encode jitter, a setup that significantly reduces the number of parameters that need to be trained.

Datasets and Evaluation Methods. Given that QuadTrack [4] is currently the only existing quadruped robots perspective panoramic dataset, we conduct training and controllability evaluation on this dataset, as well as verify the effectiveness of our framework for downstream tasks. QuadTrack contains 32 video clips with 600 frames each, a total of $19,200$ panoramic images, and $189,876$ high-quality bounding boxes with manual annotations. The dataset contains 17 sets of video clips as the training set, with the remaining 15 sets as the test set. We designate 9 sets from the test set as the new test set for evaluating generation quality and controllability, whereas the remaining 6 sets are used as the validation set for downstream task evaluation. Specifically, we train our generative model using the training set and generate images on the test set using the first frame's panoramic image and object bounding box prompts, then compare and evaluate the generation quality against the real images. For controllability, we first trained OmniTrack, a multi-object tracking model for quadruped robots, using the training set. Then, we used it to track the images generated from the test set in order to evaluate the controllability of our generated outputs. For downstream tasks, following the previous tasks TrackDiffusion [5] and MagicDrive [6], we add the best two video clips generated by different models into OmniTrack's training set to train the OmniTrack model, and then evaluate it on the validation set.

2 PTrack Evaluation Method

To evaluate whether the generated videos exhibit camera motion jitter consistent with real-world counterparts, we propose PTrack — a novel evaluation metric leveraging CoTracker3 [7] for feature point tracking to extract jitter patterns. CoTracker3 is a streamlined yet powerful deep learning tracking model capable of maintaining high-fidelity arbitrary point correspondence even in videos with complex motion dynamics and severe occlusion conditions. Let the video frame resolution be $H \times W$, and define G^2 uniformly sampled grid points $P = \{P_1, P_2, \dots, P_{G^2}\}$, where the column spacing between adjacent points is $\Delta_x = \frac{W-1}{G}$, and the row spacing is $\Delta_y = \frac{H-1}{G}$. For each sampled point, the Jectory tracked across T frames in the video is denoted as $\mathcal{J}_{P_i} = \{P_i^1, P_i^2, \dots, P_i^T\}$, where $P_i^t = \{x_i^t, y_i^t\}$ represents the coordinates of the $i-th$ sampled point P_i at the $t-th$ frame. For each tracking Jectory $\mathcal{J}_{P_i}$, calculate its temporal variance:

$$\begin{cases} \sigma^x_{\mathcal{J}_{P_i}} = \dfrac{1}{T}\displaystyle\sum_{t=1}^{T}(x_i^t - \bar{x_i}^t)^2 \\ \sigma^y_{\mathcal{J}_{P_i}} = \dfrac{1}{T}\displaystyle\sum_{t=1}^{T}(y_i^t - \bar{y_i}^t)^2 \end{cases}, \tag{1}$$

where $\sigma_{\mathcal{J}_{P_i}}$ is the temporal variance of the $i-th$ sampled point P_i, and $\bar{x_i}^t$ and $\bar{y_i}^t$ are the average coordinates of P_i in the $x-$direction and $y-$direction at the $t-th$ frame, respectively. The tracked trajectory $\mathcal{J}_{P_i}$ includes both the object motion and the jitter. To evaluate the intensity of the jitter trajectory, we remove the object motion trajectory, resulting in the jitter trajectory $\mathcal{J}$:

$$\mathcal{J} = \{\mathcal{J}_{P_i} \mid \sigma^x_{\mathcal{J}_{P_i}} \leq Q_{0.8}(\{\sigma^x_{\mathcal{J}_{P_i}}\}_{i=1}^{G^2}) \wedge \sigma^y_{\mathcal{J}_{P_i}} \leq Q_{0.8}(\{\sigma^y_{\mathcal{J}_{P_i}}\}_{i=1}^{G^2})\}. \tag{2}$$

Here, the expression $Q_{0.8}(\{\sigma^x_{\mathcal{J}_{P_i}}\}_{i=1}^{G^2})$ and $Q_{0.8}(\{\sigma^y_{\mathcal{J}_{P_i}}\}_{i=1}^{G^2})$ represent the $80th$ percentile of the temporal variance of the x-direction and y-direction coordinates of all sampled points, respectively. Finally, calculate the PTrack metric:

$$PTrack = \frac{1}{|\mathcal{J}|}\sum_{P_i \in \mathcal{J}}\left[\frac{1}{T}\sum_{t=1}^{T}\left(\frac{y^t_{gen,p} - y^t_{real,p}}{\Delta y}\right)^2 \cdot \omega_p\right], \tag{3}$$

where $y^t_{gen,p} - y^t_{real,p}$ represents the difference between the generated vertical jitter and the real vertical jitter. To prevent the mean trap—where the generated video only replicates the mean trajectory of the real jitter while ignoring the fluctuation characteristics—we introduce a variance difference term $\omega_p = (\sigma^y_{real,p} - \sigma^y_{gen,p})^2$ to compensate for this flaw. In our evaluation, we set $G = 10$. For the 378 14-frame video clips in the test set, we calculate their PTrack metric and compute the average.

3 Controllability Analysis

Comparison with or without filtering. We use a Butterworth high-pass filter to extract the high-frequency components from the object bounding box trajectories as the jitter data for the quadruped robot. We also explore the case without filtering and compare the results, as shown in Tab. 1. It can

Method	**PTrack ↓**	**HOTA ↑**	**MOTA ↑**	**DetA ↑**	**AssA ↑**
No Filtering	17.4361	9.3875	**-19.592**	11.374	7.986
Filtering	**14.1744**	**9.6542**	-21.285	**11.628**	**8.2799**

Table 1: **Comparison with or without filtering.** "No Filtering" means that we directly use the object bounding box coordinates as jitter input into the framework for training and generation.

be observed that, except for the MOTA metric, the controllability of the model has improved in all other metrics when using the filtered data. This is because MOTA primarily focuses on issues such as false positives, false negatives, and identity switches during object detection and tracking. When no filtered data is used, the control signals in SOC are a fusion of object bounding box coordinates with two different encodings, which is why the MOTA metric performs better. Additionally, we found that this method of not using filtering completely causes the model to lose the ability to control jitter.

The VJE component consists of both the filtering mechanism and the camera encoder, which serves as a crucial input to SOC. As the camera encoder is treated as a fixed-weight (frozen) module and integrated as a whole input, it is not feasible to ablate its sub-components separately in the current design. Nonetheless, as shown in Tab. 2, we additionally conduct an ablation study using only SOC, with no filtering applied in the VJE module. The ablation further demonstrates the effectiveness of the filtering module in VJE.

Setting	VJE*	SOC	PE	FVD ↓	LPIPS ↓	PSNR↑	SSIM↑	HOTA ↑	MOTA ↑	PTrack↓
(a)		✓		904.37	0.2703	14.33	0.3867	9.3886	-20.72	21.5676
(b)	✓	✓		896.07	0.2620	14.51	0.3950	9.3924	-20.006	15.9676

Table 2: VJE* refers to the configuration in which the VJE module skips filtering and performs only camera encoding on the data.

The impact of the module on control performance. Through experimental verification, we found that the temporal attention blocks introduced in the SOC module are the key factor in improving the model's control effectiveness. Additionally, if the feature F_{sum} input to the Gated attention in the SOC module is not fused with the encoded jitter feature A_{bg}, the model will lose the ability to control jitter. This is because using only the encoding of the object bounding box positions in the gated attention causes the model to confuse the jitter of the object bounding box coordinates with the extracted jitter features.

Full 6-DoF control. Our SOC framework supports full 6-DoF control. Internally, it uses camera encoders to represent and regulate camera pose, which has been validated in CameraCtrl to support both translation (including side-to-side sway) and rotation (yaw, pitch, and roll). In our current experiments, we focus on vertical jitter suppression, as it is the most prominent visual disturbance during quadruped locomotion. We also simulate yaw-axis variations to explore SOC's generalization beyond vertical control. As in Fig. 1, SOC remains robust under such conditions.

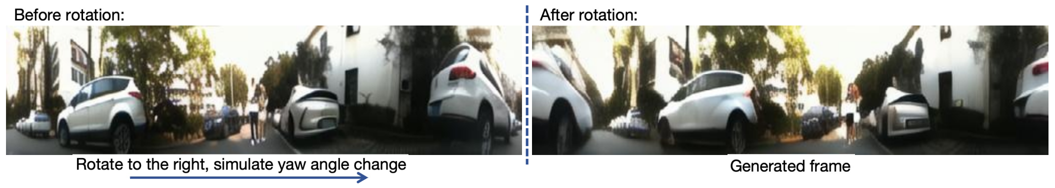

Figure 1: Reasoning for simulated yaw angle changes

Model generalization ability. The proposed method generalizes to different types of quadruped robots. Our dataset includes diverse motion data from both Unitree Robotics Go2 and DEEP Robotics Lite3, with Tab. 3 showing the generation performance on these two platforms. Lite3 exhibits more jerky motion, whereas Go2 has higher speeds but features a more limited set of movement patterns. We will clarify the data sources of the two platforms in the final version. We use one model to support controllable generation for different robots. The model is applicable to different robots exhibiting significant vertical jitter during movement.

Robots	**Percentage**	FVD ↓	LPIPS ↓	PSNR ↑	SSIM ↑	HOTA ↑	MOTA ↑	PTrack ↓
Go2	44.44%	958.87	0.2743	14.1166	0.3691	7.9721	-21.59	16.3713
Lite3	55.56%	845.15	0.2512	14.8332	0.4152	11.304	-21.81	11.4282

Table 3: Evaluation on test data from different platforms.

4 Visualization Results

Comparison with other models. We present more visual results in Fig. 2, 3, and 5, where the rainbow-colored trajectories represent the vertical jitter tracked using CoTracker3. The red boxes highlight the prominent regions. It can be observed that the video jitter generated by QuaDreamer (Ours) is closest to the ground truth video (GT).

Performance in blurry scenes. As shown in Fig 4, due to the high-frequency vertical jitter caused by the unique gait of the robot dog, some panoramic images become blurred during capture due to long exposure times. However, the data generated by our model effectively avoids this blur, providing more effective data support for downstream tasks.

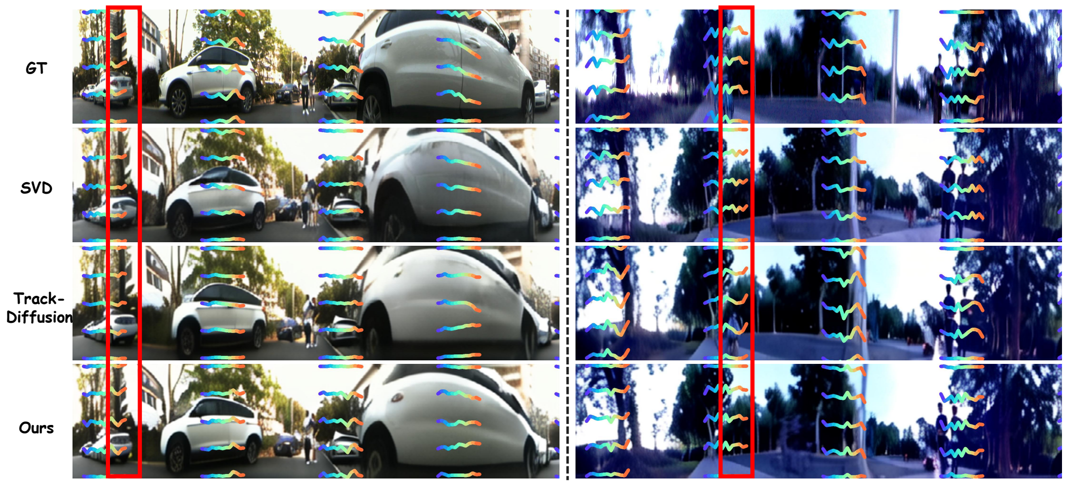

Figure 2: **Visual Comparison 1**

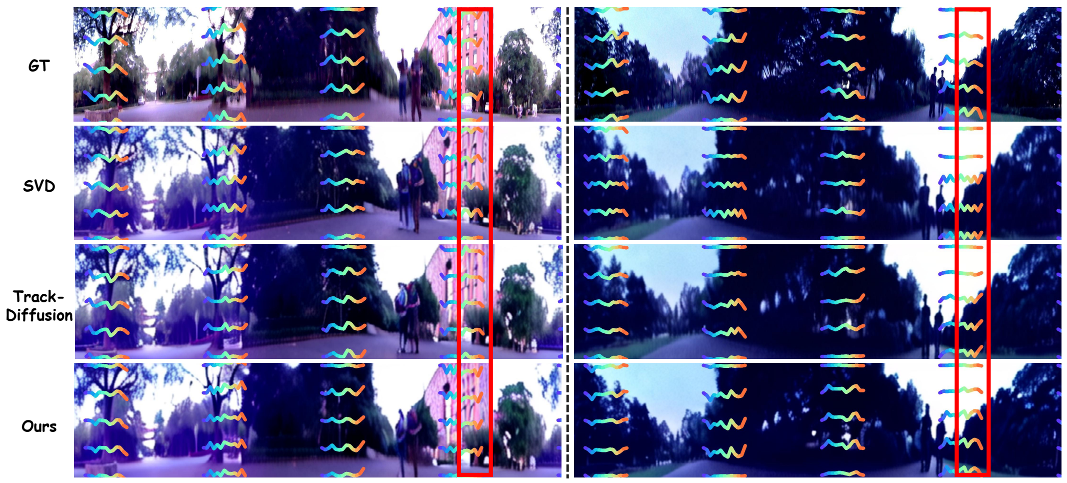

Figure 3: **Visual Comparison 2**

5 Research Significance

This research has significant theoretical significance and practical value in enhancing the panoramic perception capabilities of quadruped robots. Currently, quadruped robots in complex scenarios such as inspection, rescue, and security urgently require panoramic vision systems to provide comprehensive environmental understanding. However, due to issues such as robot body motion jitter and sensor calibration difficulties, acquiring high-quality panoramic data in real-world environments is extremely costly. This data bottleneck severely restricts the training and deployment of perception models for quadruped robots. The QuaDreamer proposed in this paper, as the first panoramic data generation engine customized for quadruped robots, generates high-quality panoramic videos with vertical vibration features due to quadruped robot movement and dynamic interactions with moving targets. These videos not only provide high-quality and controllable training data for perception models but also lay the foundation for building robust panoramic vision systems. This plays a key role in advancing the autonomous interaction capabilities of quadruped robots in open environments.

From the perspective of technological innovation, the Vertical Jitter Encoding (VJE), Scene-Object Controller (SOC), and Panoramic Enhancer (PE) proposed in this study provide crucial technical support for generating panoramic videos from the perspective of a quadruped robot. VJE employs a high-pass filter to decouple the low-frequency object-relative trajectories from the high-frequency vertical jitter, and the high-frequency vertical jitter characteristics are encoded through the camera encoder. SOC utilizes a multimodal fusion mechanism with attention to control both jitter and object

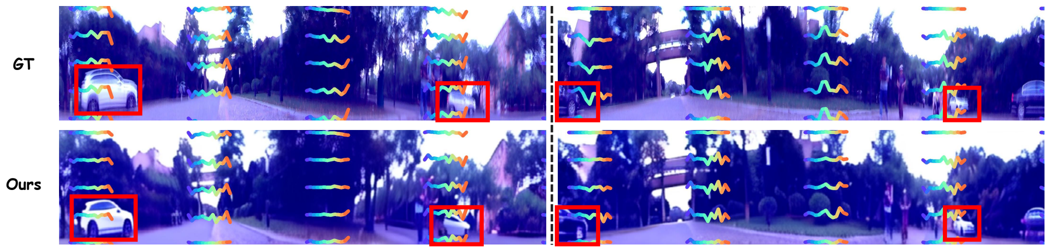

Figure 4: **Visualization in blurry scenes.** The object within the red box in the ground truth shows vertical blur, while the video generated by QuaDreamer effectively eliminates this issue.

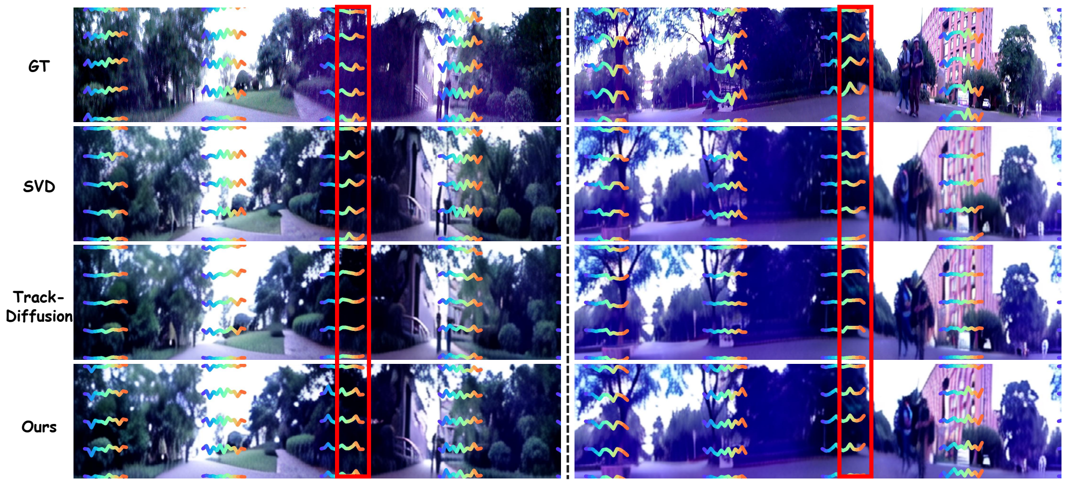

Figure 5: **Visual Comparison 3.**

interactions. PE innovatively integrates frequency-domain features and spatial-domain structures through a dual-stream architecture, enhancing the quality of the generated panoramic videos. These technological innovations significantly improve the quality and control accuracy of panoramic data generation. Experiments show that synthetic data can effectively enhance the performance metrics of panoramic multi-object tracking tasks, validating its practical value in training perception models and providing an expandable technical framework for the development of future robotic simulation systems. The open-source release of this engine will further promote collaborative innovation within the quadruped robot vision community and accelerate the deployment of quadruped robot technology in complex scenarios.

6 Future Work

Our work is based on Box and Jitter information; however, in real-world robotic applications, the working environment typically requires collaboration among multiple sensors (*e.g.*, depth, infrared, and event cameras). Extending our framework to an RGB-X generation model is a promising direction for future exploration.

The current work reduces the reliance on data by extracting jitter information from the coordinates of the frames. In the future, we consider directly obtaining the quadruped robot's posture information from the IMU and implementing more control effects, such as generating movements like forward/backward and turning.

Furthermore, leveraging large language models to enable generation under coarsely annotated conditions also presents a valuable and underexplored avenue.

References

[1] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu. DPM-Solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In *NeurIPS*, pages 5775–5787, 2022.

[2] A. Blattmann, T. Dockhorn, S. Kulal, D. Mendelevitch, M. Kilian, D. Lorenz, Y. Levi, Z. English, V. Voleti, A. Letts, V. Jampani, and R. Rombach. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[3] H. He, Y. Xu, Y. Guo, G. Wetzstein, B. Dai, H. Li, and C. Yang. CameraCtrl: Enabling camera control for text-to-video generation. *arXiv preprint arXiv:2404.02101*, 2024.

[4] K. Luo, H. Shi, S. Wu, F. Teng, M. Duan, C. Huang, Y. Wang, K. Wang, and K. Yang. Omnidirectional multi-object tracking. In *CVPR*, 2025.

[5] P. Li, K. Chen, Z. Liu, R. Gao, L. Hong, G. Zhou, H. Yao, D.-Y. Yeung, H. Lu, and X. Jia. TrackDiffusion: Tracklet-conditioned video generation via diffusion models. In *WACV*, pages 3539–3548, 2025.

[6] R. Gao, K. Chen, E. Xie, L. Hong, Z. Li, D.-Y. Yeung, and Q. Xu. MagicDrive: Street view generation with diverse 3D geometry control. In *ICLR*, 2024.

[7] N. Karaev, I. Makarov, J. Wang, N. Neverova, A. Vedaldi, and C. Rupprecht. CoTracker3: Simpler and better point tracking by pseudo-labelling real videos. *arXiv preprint arXiv:2410.11831*, 2024.