LEAP-VO: Long-term Effective Any Point Tracking for Visual Odometry

Supplementary Material

A. Implementation Details

A.1. LEAP

Training. We implement our approach in Python 3.10 with Pytorch 1.12. We train our model following the setting of CoTracker [18] on the TAP-Vid-Kubric [9] training set, which contains 11,000 synthetic sequences with 24 frames from MOVi-F dataset. Additionally, we extract the dynamic track label from the instance dynamic label provided by the Kubric simulator to supervise the dynamic track estimation. Notably, the MOVi-F dataset contains a limited number of scenes involving falling objects, accompanied by simple, linear camera motion at a constant velocity. In contrast, the TartanAir training set, used by TartanVO [40], DytanVO [28], and DPVO [33], contains a wider range of environments and features more complex camera motions, along with a greater number of images. The image resolution of the synthetic sequences is 512×512. During training, the input images are cropped to a resolution of 384×512, and the feature map stride is 4. We adopt the AdamW optimizer with the learning rate of $3e-4$ and linear warmup cosine annealing scheduler.

Model Architectures. For the image feature extractor, we employ the same Convolutional Neural Network (CNN) as used in PIPs [16]. It begins with a 7x7 convolution layer with a stride of 2, followed by an instance normalization layer and ReLU activation. The network then comprises several 3x3 kernel-sized residual blocks, each designed to process image features at different scales. The outputs of these layers are resized to a consistent scale using bilinear interpolation and then stacked. These stacked features are subsequently processed through 3x3 and 1x1 convolution layers. Regarding the refiner, we have explored integrating anchor-based inter-track attention with the MLP-based refiner from [16] and the transformer-based refiner from [18]. We find that both models achieve comparable point tracking accuracy given sufficient training time. However, the transformer-based network, which possesses five times more parameters, demonstrates faster convergence with fewer epochs of training. Consequently, we have chosen the transformer-based refiner for its training efficiency.

A.2. LEAP-VO

Keypoint Extration and Tracking. During the tracking process, we have introduced an additional hyperparameter, S_{KF}, to control the frequency of keypoint extraction, *i.e.*, keypoints are extracted every S_{KF} frame. The keypoints extracted within the most recent S_{LP} are tracked bi-directionally within the LEAP window. A smaller S_{KF} results in more frequent keypoint extraction, leading to a denser video representation and better capturing of motion patterns. However, extracting more keypoints also increases computational cost during the tracking front-end, especially due to the inter-track attention mechanism that exchanges information between every pair of tracks. We use $S_{KF} = 2$ and $S_{LP} = 12$ to balance performance and efficiency during the experiment. For the number of keypoints, we set $N = 64$ for both Replica and TartanAir-Shibuya, and $N = 100$ for MPI-Sintel.

Local Bundle Adjustment. In developing LEAP-VO, we build our system based on the local bundle adjustment (BA) implementation of DPVO [33]. The bundle adjustment window size, denoted as S_{BA}, is set to 15. This means that at each step, local BA optimizes the camera poses for the most recent 15 frames. Following local BA, we also incorporate the map outlier filtering process. This process filters out points with large projection errors, specifically those where the distance between the projected correspondences and LEAP correspondences is substantial. Implementing this step helps to enhance the accuracy and robustness of the entire system during incremental tracking.

Initialization. We initiate the visual odometry system after collecting initial frames, followed by 12 bundle adjustment (BA) steps to determine the initial camera pose and 3D scene points. As discussed in Section 3.4, keypoints are parameterized using their 2D-pixel coordinates from the extracted frame and a scalar depth value. Initially, we assign depth values by sampling from a uniform distribution within the range of $[0, 1)$. After VO initialization, depth values are initialized using the median depth from the past three frames [33]. An additional advantage of our approach, which utilizes continuous feature tracking within a sliding window, is the ability to reuse estimated point locations from the previous window for better initialization in the current one. Let $\mathbf{X}_{i\to j}$ denote the estimated point position in frame $\mathbf{I}_j$, associated with its source frame $\mathbf{I}_i$. For all point tracking targets in the current local window $\mathbf{I}_{t-S_{LP}+1:t}$, it is observed that for all $i, j \in [t - S_{LP} + 1, t - 1]$, estimations have already been made by the previous local window $\mathbf{I}_{t-S_{LP}:t-1}$. Therefore, instead of initializing the point trajectory $\mathbf{X}^0$ by duplicating the query position $\mathbf{x}_q$, we can leverage the previous estimation for a better initialization.

Hyperparameters Setting. In the LEAP-VO system, the

Method	MPI Sintel		
	ATE	RPE trans (m)	RPE rot (deg)
LEAP-VO (Gaussian)	0.053	0.064	1.277
LEAP-VO (Cauchy)	**0.037**	**0.055**	**1.263**

Table B.1. **Comparison of temporal distribution for camera tracking performance.** Our Cauchy distribution formulation achieves better results compared to the Gaussian distribution.

Method	TAP-Vid Davis (first)		
	AJ	$< \delta^x_{avg}$	OA
LEAP (Gaussian)	50.855	66.032	83.177
LEAP (Cauchy)	**56.889**	**73.301**	**85.005**

Table B.2. **Comparison of temporal distribution for point tracking performance.** Our Cauchy distribution formulation achieves better results compared to the Gaussian distribution.

track filtering parameters $(\gamma_v, \gamma_d, \gamma_u)$ play a pivotal role in determining the quality of point trajectories for bundle adjustment. Specifically, γ_v filters occluded points, γ_d identifies dynamic points, and γ_u manages points with high uncertainty. For those key hyperparameters, we perform *individual* coarse tuning. We tune $(\gamma_v, \gamma_d, \gamma_u)$ using three thresholds $\{0.3, 0.6, 0.9\}$, and then adjust them by ± 0.1. Once the LEAP network is trained, the majority of hyperparameters can be set and remain fixed for practical usage.

B. Additional Ablation Study

Temporal Probabilistic Distribution. In our Long-term Effective Any Point Tracking (LEAP) module, we employ a multivariate Cauchy distribution to model the temporal relationships among points from the same track. This approach is compared with the widely used multivariate Gaussian distribution. For this comparison, we train the LEAP network using the negative log-likelihood (NLL) loss with respect to the Gaussian distribution. The results of camera tracking using both LEAP (Cauchy) and LEAP (Gaussian) are presented in Table B.1. It is evident that LEAP-VO (Cauchy), with the LEAP (Cauchy) front-end, outperforms LEAP-VO (Gaussian) in all metrics. To further assess the differences, we evaluate the point tracking accuracy of both models on the TAP-Vid Davis First benchmark [9]. We use the standard metrics from [9], including the average Jaccard score (AJ), the average percentage of correct keypoints ($< \delta^x_{avg}$), and occlusion accuracy (OA), all calculated across various thresholds. The results, as shown in Table B.2, indicate that LEAP (Cauchy) surpasses LEAP (Gaussian) in point tracking accuracy, potentially leading to improved camera tracking performance in VO. We hypothesize that this improvement could be due to the L2 norm error term in the Gaussian NLL loss being less robust to outliers during training.

# Keypoints	N=16	N=64	N=144	N=256
ATE (m)	0.2449	0.0290	0.0312	0.0261

Table B.3. **Comparison of query numbers on TartanAir-Shibuya.** We use $N = 64$ to balance between accuracy and efficiency.

Influence of query point numbers. We compare the influence of different numbers of query point on the final VO performance, as shown in Table B.3. The query number can be tuned according to practical requirements for efficiency and accuracy.

# Frames	$S_{LP} = 8$	$S_{LP} = 12$	$S_{LP} = 16$
Time (ms)	224.46	380.56	526.37

Table B.4. **Efficiency of LEAP.** For a larger window, LEAP takes a longer time to track the query points.

Efficiency of LEAP. We evaluate the efficiency of our LEAP algorithm by tracking 256 points from the first frame of the window, using a single NVIDIA RTX A4000 GPU (16GB). The results are displayed in Table B.4. We set $S_{LP} = 12$ to balance between track length and track reliability. Increasing the feature map stride from 4 to 8 halves the inference time, and switching from global inter-track attention to local attention could further reduce it.

C. More Visualization Results

Qualitative Results for Visual Odometry. We visualize the qualitative results for camera tracking, including dynamic track estimation, temporal probabilistic modeling, and comparisons of VO performance. Figure D.1 displays results from sample sequences of MPI-Sintel [4], TartanAir-Shibuya [26], and Replica [29]. In dynamic environments, methods that explicitly handle dynamic objects, such as DytanVO and ours, demonstrate superior performance compared to TartanVO, which does not address dynamic elements. DPVO, through implicitly learning the uncertainty from point correspondences, manages to handle dynamic scenes to some extent. However, it is not as effective as DytanVO and our method. This underscores the effectiveness of explicitly modeling dynamic elements in the formulation. Our method further surpasses DytanVO, showcasing the advantages of long-term point tracking over the two-view method.

Dynamic Track Estimation. We present additional visualizations of the dynamic track estimation results on the DAVIS dataset [19], as illustrated in Figure D.2. These visualizations demonstrate that our LEAP module is adept at handling dynamic scenes across a variety of scenarios.

Temporal Uncertainty Estimation. We present additional visualizations of the uncertainty estimation results on the

MPI-Sintel [4] and Replica [29] datasets, as shown in Figure D.3 and Figure D.4, respectively. Guided by the NLL loss, our model tends to assign high uncertainty to areas with low texture and dynamic objects.

D. Discussion and Future Work

Learning-based Anchor Selections. We have incorporated anchor-based methods into the LEAP pipeline to select points that capture motion patterns effectively. Our current method, which chooses anchors based on distributed image gradient sampling, outperforms uniform or random strategies. This technique selects easier-to-track anchors from high-gradient areas, ensuring comprehensive sequence coverage. However, it remains unclear whether this anchor selection strategy is *optimal* for the specific LEAP front-end. Exploring a learning-based anchor selection, integrated end-to-end with the point-tracking framework, could be beneficial. The main challenge lies in developing effective loss functions or self-supervised objectives for learning the anchor selection process.

Joint Refinement with Camera Tracking. In our visual odometry (VO) system, we distinctively handle the refinement of point trajectories and the refinement of camera pose and 3D map points as separate components. The refinement of point trajectories is managed using the refiner module in LEAP, which predicts iterative updates for point state variables including position, features, and other distribution parameters. For refining camera pose and 3D map points, we use a sliding-window local bundle adjustment module that leverages the Gauss-Newton method for updates. A promising future direction would be to merge point trajectory refinement with camera pose optimization into a unified refinement system.

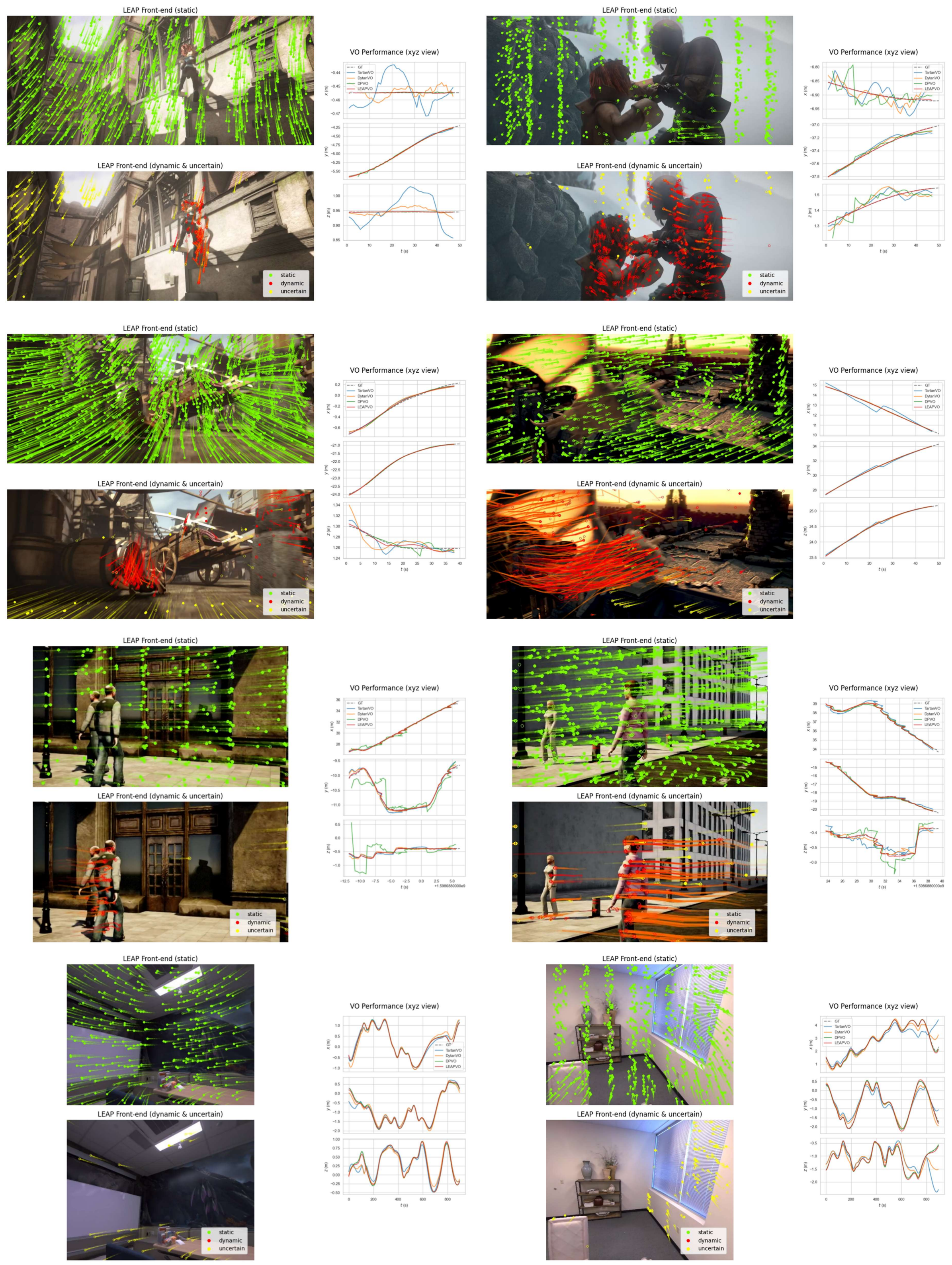

Figure D.1. **Qualitative results for Visual Odometry on MPI-Sintel [4], TartanAir-Shibuya [26], and Replica [29]. Upper left**: image sample with static (green) point tracking. **Lower left**: image sample with dynamic (red) and uncertain (yellow) point tracking. **Right**: comparison with the state-of-the-art VO methods.

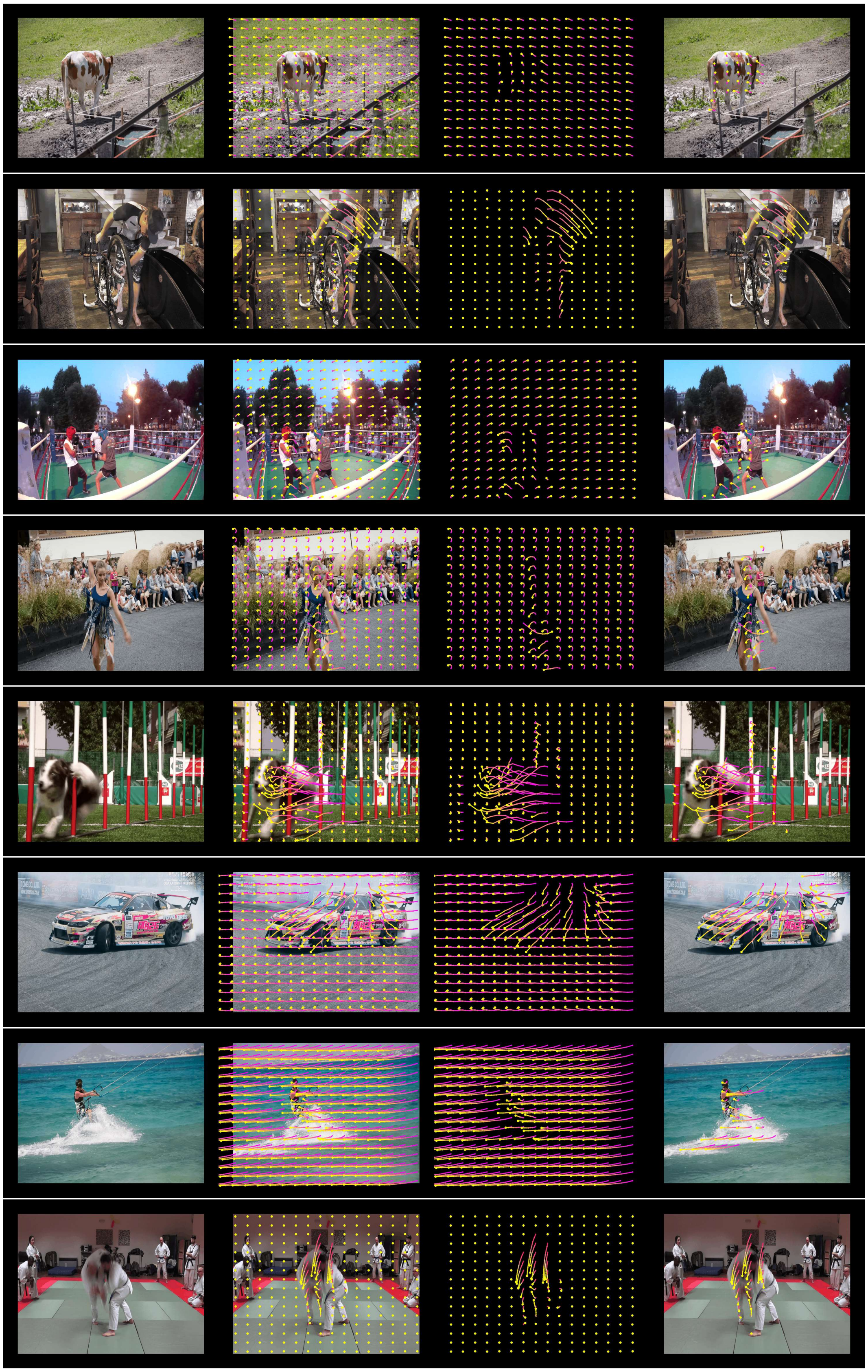

Figure D.2. **Visualization of dynamic track estimation on DAVIS [19]. From left to right:** sample from image sequence, image with all point trajectories, all point trajectories, and estimated dynamic point trajectories.

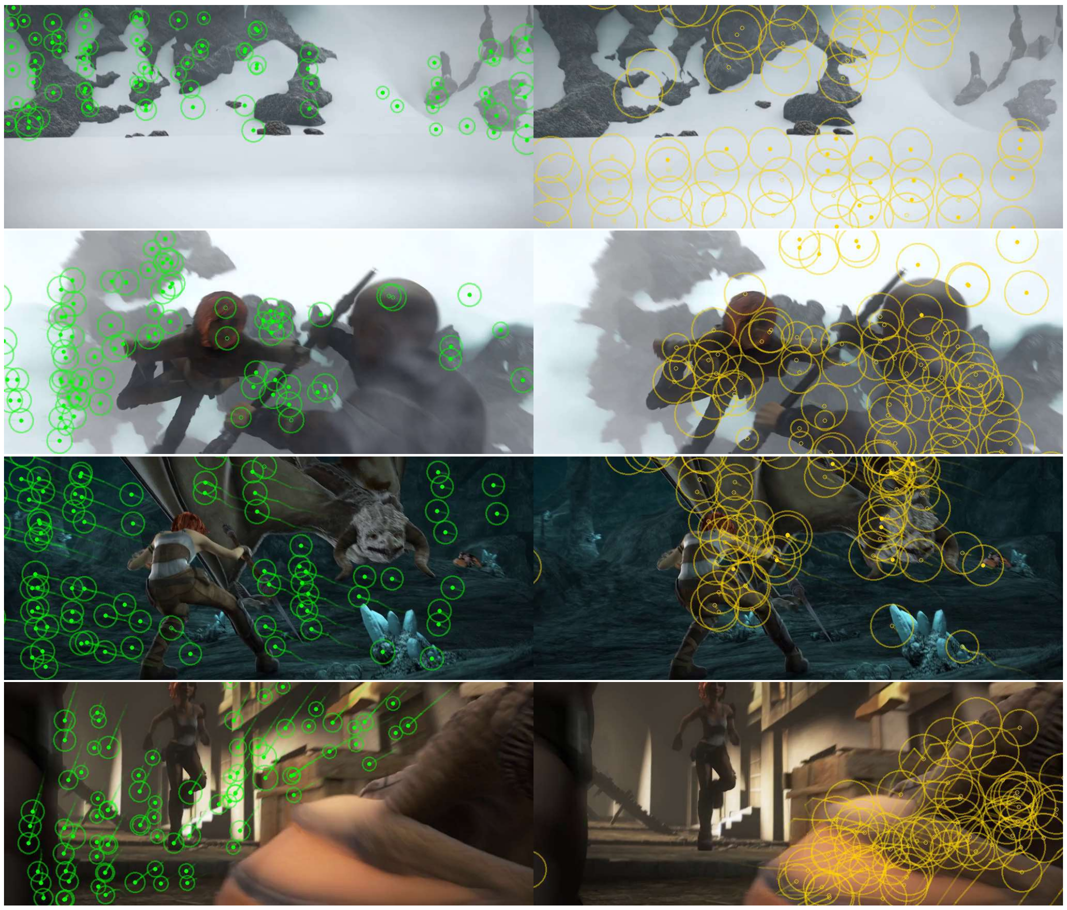

Figure D.3. **Visualization of point-wise uncertainty measurements on MPI-Sintel [4].** Keypoints with the lowest 20% (left) and highest 20% (right) uncertainty are shown in green and yellow, respectively.

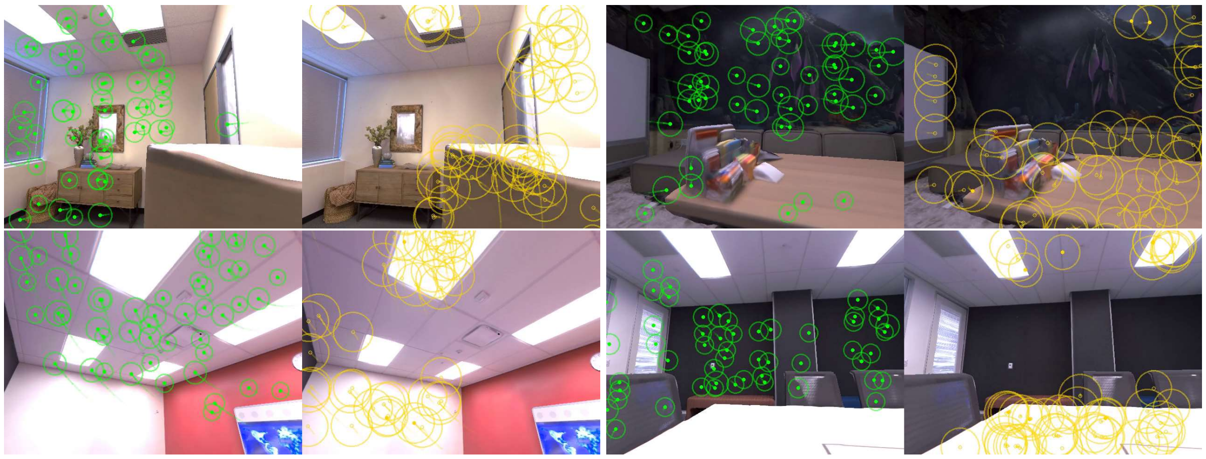

Figure D.4. **Visualization of point-wise uncertainty measurements on Replica [29].** Keypoints with the lowest 20% (left) and highest 20% (right) uncertainty are shown in green and yellow, respectively.