PiLoT: Neural Pixel-to-3D Registration for UAV-based Ego and Target Geo-localization

Supplementary Material

1. Method Details

1.1. Network Architecture

Our feature extraction backbone is a lightweight U-Net architecture, with its detailed data flow outlined in Table 1. The encoder path employs the first three stages of a pre-trained MobileOne-S0 [1] to efficiently extract a hierarchy of features at strides of 2, 4, and 8. At each stage of the decoder, two parallel heads are applied to generate the multi-level outputs. A projection head, consisting of lightweight 3x3 convolutions, processes the decoder features to produce a feature map. Concurrently, an uncertainty head predicts a per-level single-channel uncertainty map. This entire process yields a three-level pyramid of feature-uncertainty pairs, indexed from coarse to fine ($\ell = 0, 1, 2$) corresponding to resolutions of $1/4$, $1/2$, and 1.

Table 1. Detailed data flow of our MobileOne-UNet feature extractor. The table illustrates the transformation of tensor shapes through the encoder, decoder, and output heads for an input image of resolution $H \times W = 512 \times 512$.

Stage	Operation	Resolution	Channels	Description
Encoder (MobileOne-S0)				
Input	-	$H \times W$	3	Input RGB image
Stage 1 (E_1)	Stem + Stage 1	$(H/2) \times (W/2)$	C_1	First level of features (for skip)
Stage 2 (E_2)	Stage 2 Blocks	$(H/4) \times (W/4)$	C_2	Second level of features (for skip)
Stage 3 (E_3)	Stage 3 Blocks	$(H/8) \times (W/8)$	C_3	Deepest features fed to decoder
Decoder				
Block 1 (D_1)	Upsample(E_3) + Concat(E_2)	$(H/4) \times (W/4)$	128	Upsample and fuse with skip connection
Block 2 (D_2)	Upsample(D_1) + Concat(E_1)	$(H/2) \times (W/2)$	64	Upsample and fuse with skip connection
Block 3 (D_3)	Upsample(D_2)	$H \times W$	32	Final upsampling (no skip connection)
Output Heads (Applied to D_1, D_2, D_3***)***				
Projection	Conv 3x3	$(H/4) \times (W/4)$ $(H/2) \times (W/2)$ $H \times W$	32 32 32	3-level feature pyramid
Uncertainty	Conv 1x1 + Sigmoid	$(H/4) \times (W/4)$ $(H/2) \times (W/2)$ $H \times W$	1 1 1	3-level confidence maps

1.2. JNGO Optimizer Details

As illustrated in Fig. 1, the feature-metric cost landscape is often highly non-convex, posing a challenge for standard optimization methods. On one hand, exhaustive strategies like random sampling (Fig. 1 (a)) are too computationally expensive to be practical. On the other hand, efficient local search methods like gradient descent (Fig. 1 (b)) are highly sensitive to initialization and frequently get trapped in suboptimal local minima. To address this, JNGO synergizes stochastic and gradient-based optimization for effective global exploration and local refinement.

Iterative Linearization and Pose Update. The JNGO detailed here serves to minimize the feature-metric cost $\mathcal{C}_{\text{photo}}^{(m,\ell)}$ for each hypothesis, as defined in Eq.6 of the main paper.

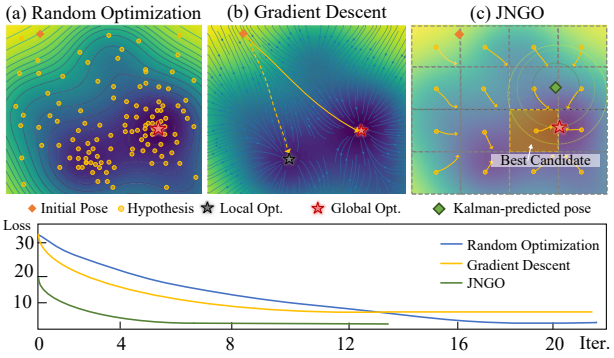

Figure 1. **Comparison of optimization strategies in a non-convex landscape.** While (a) Random Sampling is inefficient and (b) Gradient Descent is prone to local minima, our method (c) efficiently searches for the best solution. The convergence plot below illustrates how our method (green) achieves faster convergence and a lower final loss compared to alternatives.

We solve this non-linear optimization problem iteratively with a small, fixed number of updates per level to maintain real-time performance. Specifically, we perform 2, 3, and 4 LM iterations for the coarse, mid, and fine pyramid levels, respectively. While the full cost includes a robust Huber loss, the LM formulation solves the underlying non-linear least-squares problem based on the residual $\mathbf{r}_{j,\ell}^{(m)}$ from Eq.7.

Each LM iteration solves for a pose increment $\Delta\boldsymbol{\xi} \in \mathfrak{se}(3)$ by linearizing the residual function. For a single residual term $\mathbf{r}_j$ (where $j \in \{1, \ldots, N\}$), the linearization at $\tilde{\mathbf{T}}_m^{(k)}$ is:

$$\mathbf{r}_j(\exp(\Delta\boldsymbol{\xi}) \cdot \tilde{\mathbf{T}}_m^{(k)}) \approx \mathbf{r}_j(\tilde{\mathbf{T}}_m^{(k)}) + \mathbf{J}_j \Delta\boldsymbol{\xi}, \tag{1}$$

where $\mathbf{J}_j$ is the corresponding Jacobian. To solve for the update, we stack the residuals from all N 3D geo-anchors into a single vector $\mathbf{r} = [\mathbf{r}_1^\top, \ldots, \mathbf{r}_\mathbf{N}^\top]^\top$ and their Jacobians into a block matrix $\mathbf{J} = [\mathbf{J}_1^\top, \ldots, \mathbf{J}_\mathbf{N}^\top]^\top$. This yields the normal equations:

$$\left(\mathbf{J}^\top \mathbf{W} \mathbf{J} + \lambda \mathbf{I}\right) \Delta\boldsymbol{\xi} = -\mathbf{J}^\top \mathbf{W} \mathbf{r}, \tag{2}$$

where $\mathbf{W}$ is a diagonal matrix of learned uncertainty weights. The resulting increment updates the pose $\tilde{\mathbf{T}}_m^{(k+1)} = \exp(\Delta\boldsymbol{\xi}) \cdot \tilde{\mathbf{T}}_m^{(k)}$.

Jacobian Formulation. The Jacobian matrix $\mathbf{J}$ encapsulates the sensitivity of the feature residual to infinitesimal pose perturbations and is derived via the chain rule for each

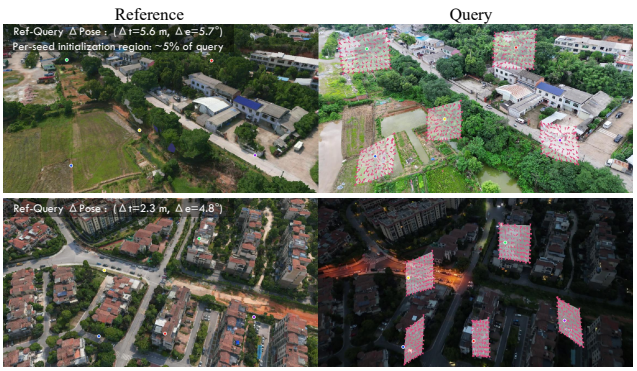

Figure 2. **Qualitative results of our multi-hypothesis refinement process.** The top row shows a challenging rural scene with significant viewpoint change, while the bottom row demonstrates robustness in a day-night urban setting. The 'convergence basin' visualization on the query image (right) demonstrates the subsequent refinement.

anchor point j:

$$\mathbf{J}_j = \frac{\partial \mathbf{r}_j}{\partial \boldsymbol{\xi}} = \underbrace{\frac{\partial \mathbf{f}_\ell^q(\tilde{\mathbf{p}}_j^q)}{\partial \tilde{\mathbf{p}}_j^q}}_{\text{Feature Gradient}} \cdot \underbrace{\frac{\partial \pi(\mathbf{P}_j^c)}{\partial \mathbf{P}_j^c}}_{\text{Projection Deriv.}} \cdot \underbrace{\frac{\partial (\tilde{\mathbf{T}}_m \mathbf{P}_j^W)}{\partial \boldsymbol{\xi}}}_{\text{Pose Deriv.}}. \quad (3)$$

The terms in this chain are:

- **Feature Gradient**: The $(C \times 2)$ matrix $\frac{\partial \mathbf{f}_\ell^q}{\partial \tilde{\mathbf{p}}_j^q}$ represents the spatial gradient of the C-dimensional query feature map, typically computed using finite differences.
- **Projection Derivative**: The (2×3) matrix $\frac{\partial \pi}{\partial \mathbf{P}_j^c}$ is the standard derivative of the pinhole camera projection with respect to the 3D point coordinates $\mathbf{P}_j^c$ in the camera frame.
- **Pose Derivative**: The (3×6) matrix $\frac{\partial (\tilde{\mathbf{T}}_m \mathbf{P}_j^W)}{\partial \boldsymbol{\xi}}$ describes how the 3D point moves in the camera frame as a result of a pose perturbation.

The product of these terms yields the final $(C \times 6)$ Jacobian block for a single anchor point, which links the 6-DoF pose update to changes in the feature residual. Figure 2 illustrates that the procedure begins by back-projecting 2D anchor points from the reference image to 3D, and then re-projecting them onto the query image based on initial pose hypotheses. For each seed point (colored dots in the reference image, left), our method initializes a local search region in the query image (right). The red arrows show the model predicting corrective pixel displacements, effectively converging towards the true location. For more convergence examples, please see Fig. 15.

Rotation-Aware Sampling Strategy. To justify our sampling design, we analyze the 6-DoF convergence basin of the proposed optimizer against the physical constraints of high-speed UAV motion. As reported in Table 2, we contrast our convergence limits with the maximum inter-frame motion (at 30 fps) derived from DJI Matrice 4 specifications. Our analysis reveals that while translation (T_x, T_y, T_z) and gimbal-stabilized Roll (ϕ) axes stay well within the convergence bounds (marked in green), the out-of-plane Yaw (ψ) and Pitch (θ) motions frequently exceed the initial convergence limits ($3.5° < 6.7°$ and $2.7° < 3.0°$, respectively). To bridge this gap and ensure robustness during aggressive maneuvers, we specifically expand the search space for Yaw and Pitch via the proposed *Rotation-Aware Sampling* strategy, effectively broadening the basin for these critical axes.

Algorithm 1 Fused CUDA Kernel for a Single LM Iteration

Input: Pose hypotheses $\{\tilde{\mathbf{T}}_m^{(k)}\}$, query features $\mathbf{f}^q$, anchor points $\{\mathbf{P}_j^W\}$, weights $\{w_j\}$
Output: System gradient $\mathbf{g}$ and Hessian $\mathbf{H}$ for each hypothesis
for all hypothesis $\tilde{\mathbf{T}}_m^{(k)}$ in parallel **do**
  Initialize global gradient $\mathbf{g} \in \mathbb{R}^6$ and Hessian $\mathbf{H} \in \mathbb{R}^{6\times 6}$ to zero.
  Launch one CUDA thread per anchor point $\mathbf{P}_j^W$
  `// --- In-thread computation (on-chip registers) ---`
  Project point: $\mathbf{P}_j^c = \tilde{\mathbf{T}}_m^{(k)} \mathbf{P}_j^W$, then $\tilde{\mathbf{p}}_j^q = \pi(\mathbf{K}, \mathbf{P}_j^c)$
  Compute residual $\mathbf{r}_j = \mathbf{f}^q(\tilde{\mathbf{p}}_j^q) - \mathbf{f}^r(\mathbf{p}_j^r)$
  Compute Jacobian $\mathbf{J}_j$ via chain rule (Eq. (3))
  Compute per-point contribution: $\mathbf{g}_j = \mathbf{J}_j^\top w_j \mathbf{r}_j$ and $\mathbf{H}_j = \mathbf{J}_j^\top w_j \mathbf{J}_j$
  `// --- Global memory reduction ---`
  Update global system using `atomicAdd`:
  $\mathbf{g} \leftarrow \mathbf{g} + \mathbf{g}_j$
  $\mathbf{H} \leftarrow \mathbf{H} + \mathbf{H}_j$
  end parallel for (threads sync implicitly)
end for
Solve $(\mathbf{H} + \lambda \mathbf{I})\Delta\boldsymbol{\xi} = -\mathbf{g}$ for all hypotheses on GPU.

Table 2. Comparison of 6-DoF convergence limits vs. maximum inter-frame motion (based on DJI specs at 30 fps).

Metric	**Yaw** (ψ)	**Pitch** (θ)	**Roll** (ϕ)	**Tx**	**Ty**	**Tz**
Conv. Limit	$3.5°$	$2.7°$	$3.2°$	5.6m	8.1m	7.4m
Max Motion (30 fps)	$6.7°$	$3.0°$	¡$0.1°$	0.7m	0.7m	0.3m

Coarse-to-Fine Analysis. We further investigate the evolution of the convergence basin across different optimization scales. As illustrated in Fig. 3, the convergence basin exhibits a characteristic narrowing as the precision increases (Coarse → Mid → Fine). While the *Fine* level provides high-precision localization, its limited basin is often insufficient to capture large inter-frame displacements. Conversely, the *Coarse* level offers a significantly broader basin, providing the necessary robustness to large initial pose er-

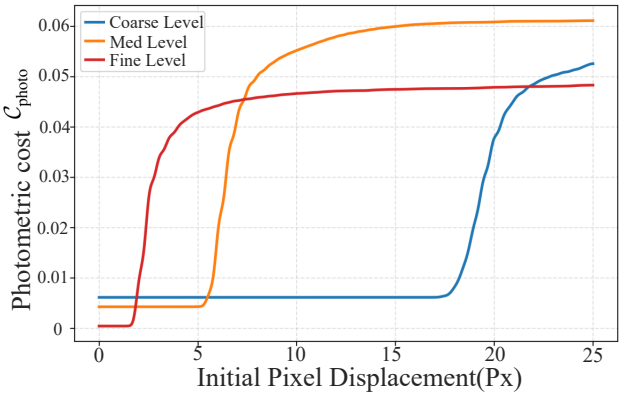

Figure 3. Convergence basin evolution across levels.

rors. This hierarchical behavior validates our coarse-to-fine design: the coarse stage provides a robust initialization that brings the pose within the tight convergence range of the fine stage, ultimately achieving a balance between global robustness and local precision.

High-Performance CUDA Implementation with Kernel Fusion. To achieve real-time performance, the computationally intensive JNGO is accelerated with a custom CUDA implementation. A naive GPU port would involve multiple kernel launches for each step of the Jacobian calculation (e.g., projection, feature gradient sampling, matrix products), leading to significant memory bandwidth bottlenecks and launch overhead. Our key optimization is the use of a single, highly-fused CUDA kernel. As outlined in Algorithm 1, we parallelize the computation by launching one GPU thread for each of the J anchor points. Each thread is responsible for the entire calculation chain for its assigned point:

1. Projecting the 3D world point $\mathbf{P}_j^W$ into the query camera frame.
2. Calculating the full Jacobian chain $\mathbf{J}_j = \frac{\partial \mathbf{f}^q}{\partial \mathbf{p}^q} \frac{\partial \pi}{\partial \mathbf{P}^c} \frac{\partial (\mathbf{T}\mathbf{P}^W)}{\partial \boldsymbol{\xi}}$ and the feature residual $\mathbf{r}_j$. All intermediate values are kept within fast on-chip registers.
3. Computing the per-point contribution to the final system: the gradient vector $\mathbf{g}_j = \mathbf{J}_j^\top \mathbf{W}_j \mathbf{r}_j$ and the Hessian matrix $\mathbf{H}_j = \mathbf{J}_j^\top \mathbf{W}_j \mathbf{J}_j$.
4. Atomically adding these local contributions to the global gradient vector $\mathbf{g}$ and Hessian matrix $\mathbf{H}$ in global memory.

This strategy minimizes costly global memory access, maximizing computational throughput. Once the kernel completes, the final small (6×6) system of normal equations is solved in a batch for all M hypotheses using a parallel Cholesky decomposition on the GPU. As shown in Tab. 3, our fused CUDA kernel dramatically reduces latency, achieving over 30x speedup, making our wide-area parallel search feasible in real-time.

Table 3. **Latency Comparison for a Single LM Iteration.** We compare different implementations for computing the cost, gradient, and Hessian. Our fused CUDA kernel dramatically reduces latency, achieving over 30x speedup compared to the initial ONNX-based implementation.

Image Size	ONNX Runtime	C++ Reference	CUDA Port	CUDA Fused Kernel
512 × 512	14.9 ms	8.44 ms	5.72 ms	**0.49 ms**
256 × 256	10.4 ms	8.35 ms	5.24 ms	**0.47 ms**
128 × 128	8.8 ms	8.27 ms	5.21 ms	**0.44 ms**

1.3. Dual-Thread Synchronization

Beyond low-level kernel optimization, PiLoT employs a synchronized dual-thread architecture to manage the interplay between map rendering and pose estimation. As detailed in Fig. 4, our pipeline ensures strict temporal alignment: a new localization cycle only commences after the preceding rendering task is completed. This design maintains a consistent 2-frame lag between the reference view and the current query frame, which is essential for stable GPU memory management during high-speed rendering.

To mitigate the resulting latency (approx. 25 ms), we incorporate a constant-velocity motion model that extrapolates the UAV's pose to the exact rendering timestamp. Given our system operates at video-rate (< 40 ms intervals), the pose drift during this brief lag remains minimal.

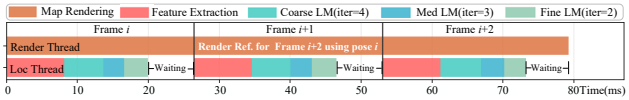

Figure 4. **Timing diagram of the dual-thread synchronization.** The pipeline maintains a fixed 2-frame lag, which is compensated by motion extrapolation to ensure real-time accuracy.

2. Dataset Details

2.1. Large-Scale Synthetic Dataset

In this section, we provide a detailed overview of our synthetic dataset, including the generation pipeline, quality validation procedures, and comprehensive statistics.

Data Generation Workflow. We developed a fully automated data acquisition pipeline, as shown in Fig. 5. This pipeline is engineered specifically to overcome the common geospatial and temporal incoherence in simulation-based data generation, ensuring high spatio-temporal coherence. The core components of our system include:

1. Geospatial Foundation: Leveraging **Cesium for Unreal**, which streams high-resolution photogrammetric 3D Tiles from Google, to provide a high-fidelity digital twin of the real world.
2. Flight and Sensor Dynamics: Utilizing **AirSim** for simulating a multi-rotor UAV, enabling precise and repro-

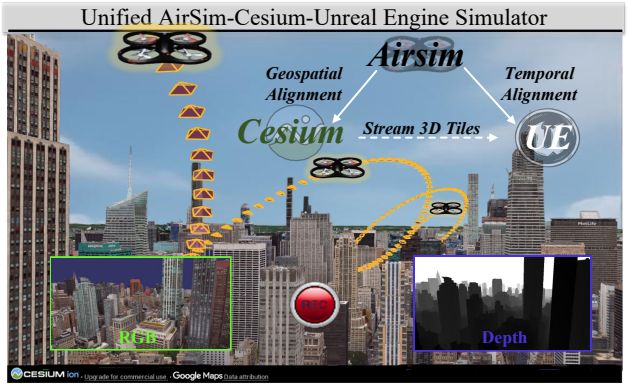

Figure 5. **Overview of the Synthetic Data Generation Pipeline.** Our system integrates Unreal Engine, Cesium, and AirSim.

ducible trajectory execution based on WGS84 waypoints.

3. Photorealistic Rendering: Harnessing **Unreal Engine**'s physically-based rendering engine to simulate a wide array of lighting and weather conditions.

In this section, we provide a detailed overview of our synthetic dataset, including comprehensive statistics and quality validation procedures.

Dataset Statistics and Diversity. Our dataset comprises over 1.1 million rendered images from 82 diverse regions, captured across over 650 km of UAV flight trajectories and featuring a rich mix of urban and natural landscapes. Environmental conditions are systematically varied across multiple weather conditions (Sunny, Cloudy, Rainy, Foggy, Snowy) and times of day (Day, Sunset, Night), and each region includes four trajectories rendered under distinct, randomly sampled weather settings to support cross-condition training and evaluation. The detailed distribution of these conditions and flight altitudes is summarized in Fig. 6. Our acquisition flights cover the sub-800 m UAV operating envelope, and the camera pitch is swept from 20° (oblique) to 90° (nadir) to approximate arbitrary viewpoints and enable rigorous evaluation of robustness to viewpoint changes.

All images are rendered at a resolution of 1600×1200. We use a pinhole camera model with per-frame intrinsics

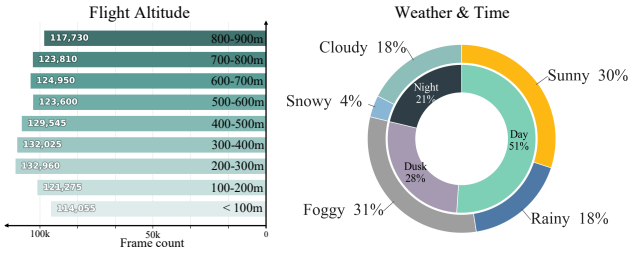

Figure 6. Dataset statistics for flight altitude (left) and environmental conditions with time of day (right).

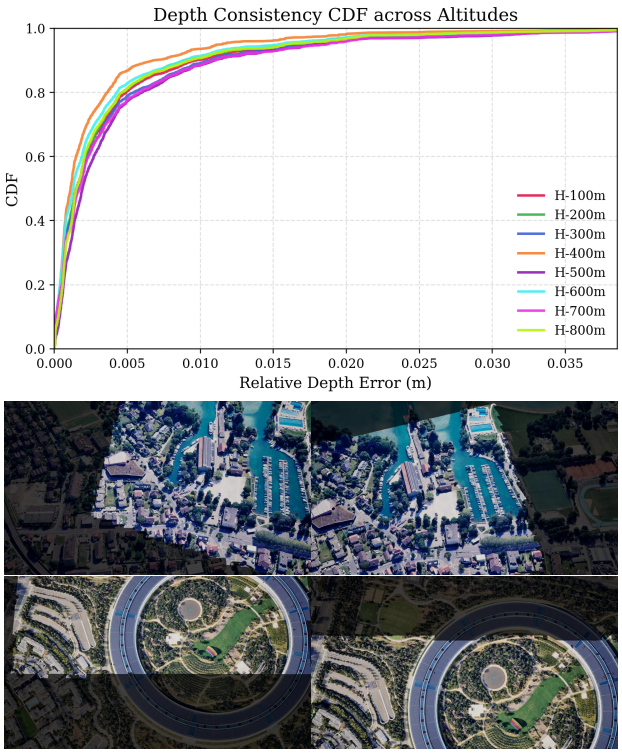

Figure 7. **Geometric Consistency Validation of Synthetic Data. Top:** The Cumulative Distribution Function (CDF) of bidirectional relative depth errors across different altitudes (100m–800m). **Bottom:** Qualitative alignment results. The middle and bottom row illustrates the accurate mapping of co-visible regions across disparate perspectives.

(f_x, f_y, c_x, c_y). For each frame, we provide the full 6-DoF pose, where the position is given in both WGS84 (longitude, latitude, height) and ECEF formats (X, Y, Z), and the rotation is provided as Euler angles (pitch, roll, yaw). This representation allows for seamless conversion to local project frames (e.g., for COLMAP or OSG) through well-defined transforms.

Data Quality and Validation. To ensure the geometric fidelity of our synthetic benchmark, we conduct both quantitative and qualitative validations. As illustrated in Fig. 7, the Cumulative Distribution Function (CDF) of bidirectional relative depth errors across diverse altitudes (100m–800m) demonstrates high numerical precision, with over 90% of pixels exhibiting a relative depth error of less than 0.01 m. Qualitatively, depth-based warping results show seamless pixel-level alignment and consistent projection of co-visible regions across disparate viewpoints.

Flight Trajectories Generation. We generate a set of "barrel-roll–inspired" style flight trajectories for data collection, as shown in Fig. 8. These trajectories combine horizontal orbiting around a central axis at a fixed radius with

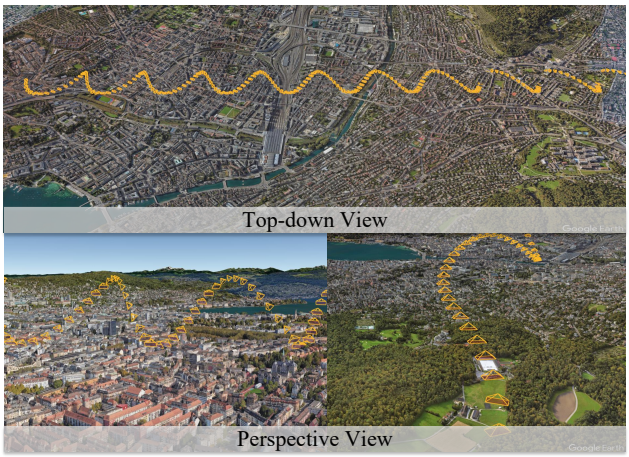

Figure 8. **Visualization of the designed "barrel-roll" trajectory.** The top-down view (top) shows yaw variation along the S-shaped path. The perspective views (bottom) illustrate changes in altitude (up-and-down arcs) and pitch (tilting of camera frustums).

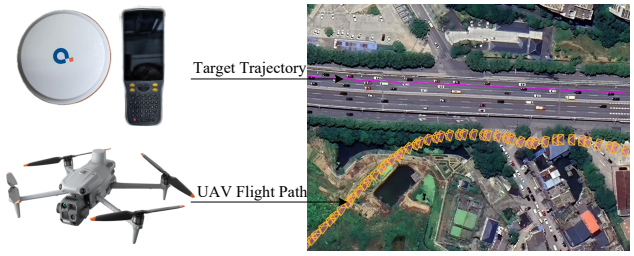

Figure 9. **Data collection setup and trajectory visualization.** (Left) The hardware used for data acquisition: a DJI M4T drone for capturing query images and Qianxun RTK instruments for ground truth positioning. (Right) A top-down view of the UAV flight path (query trajectory, shown in orange) and the corresponding ground truth (target trajectory, shown in purple).

step-wise adjustments in *pitch*, *yaw* angle and *altitude*. The flight paths also integrate straight-line cruising and curved sweeps to introduce sufficient translational and viewpoint changes, providing the necessary data for training models to be robust against cross-view and cross-scale challenges.

2.2. Evaluation Datasets

SynthCity-6 Dataset. We construct a synthetic test set, SynthCity-6, using the same data generation workflow as our training set. The test set uses six new locations from different regions of Switzerland and the USA, ensuring no geographic overlap with the training data. As detailed in Tab. 4, each sequence is rendered under 5 weather/illumination conditions (sunny, sunset, night, foggy, cloudy/rainy) and at two different altitudes (200m and 500m) to introduce significant scale variations. In total, SynthCity-6 contains 54,000 camera frames with synchronized 6-DoF poses, creating a comprehensive and challenging benchmark for assessing model robustness. For detailed trajectory visualizations and localization results on this dataset, please refer to Fig. 12 and Fig. 13.

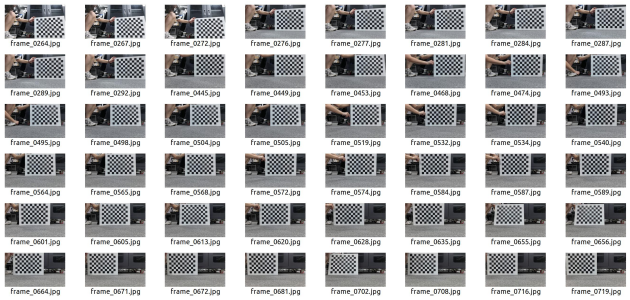

Figure 10. Sample checkerboard images used for intrinsic calibration.

UAVD4L-2yr Dataset. We build a new dataset for UAV-based ego and target geo-localization. For the reference map, we utilize the publicly available UAVD4L dataset [2], which covers a large-scale urban area ($100,000$ m^2) containing diverse buildings, streets, and vegetation. Our query sequences are captured using a DJI Matrice 4T (M4T) drone [1]. As detailed in Tab. 5, we collect data under varying illuminations (day and night), different environments (dense urban and sparse rural areas), and dynamic scenes with moving objects. Notably, a two-year time gap separates the query data from the reference map, posing a significant challenge due to long-term appearance variations. The flight paths and the target's trajectory are visualized in Fig. 9.

The drone is equipped with a centimeter-level RTK module and a high-precision IMU to record its 6-DoF ground-truth pose. Concurrently, the ground-truth poses of the moving target were independently captured using a handheld Qianxun RTK device [2]. Both the drone's and the target's RTK systems were synchronized to a common UTC time source. All ground-truth poses were subsequently transformed into a unified ECEF coordinate system to ensure precise spatio-temporal alignment between the query and target trajectories. The drone's video camera intrinsics were pre-calibrated using Zhang's method, as detailed in Fig. 10. For detailed trajectory visualizations and localization results on this dataset, please refer to Fig. 13.

References

[1] Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. Mobileone: An improved one millisecond mobile backbone. In *CVPR*, 2023. 1

[1] https://enterprise.dji.com/cn/matrice-4-series
[2] https://en.qxwz.com/en-product/Q300

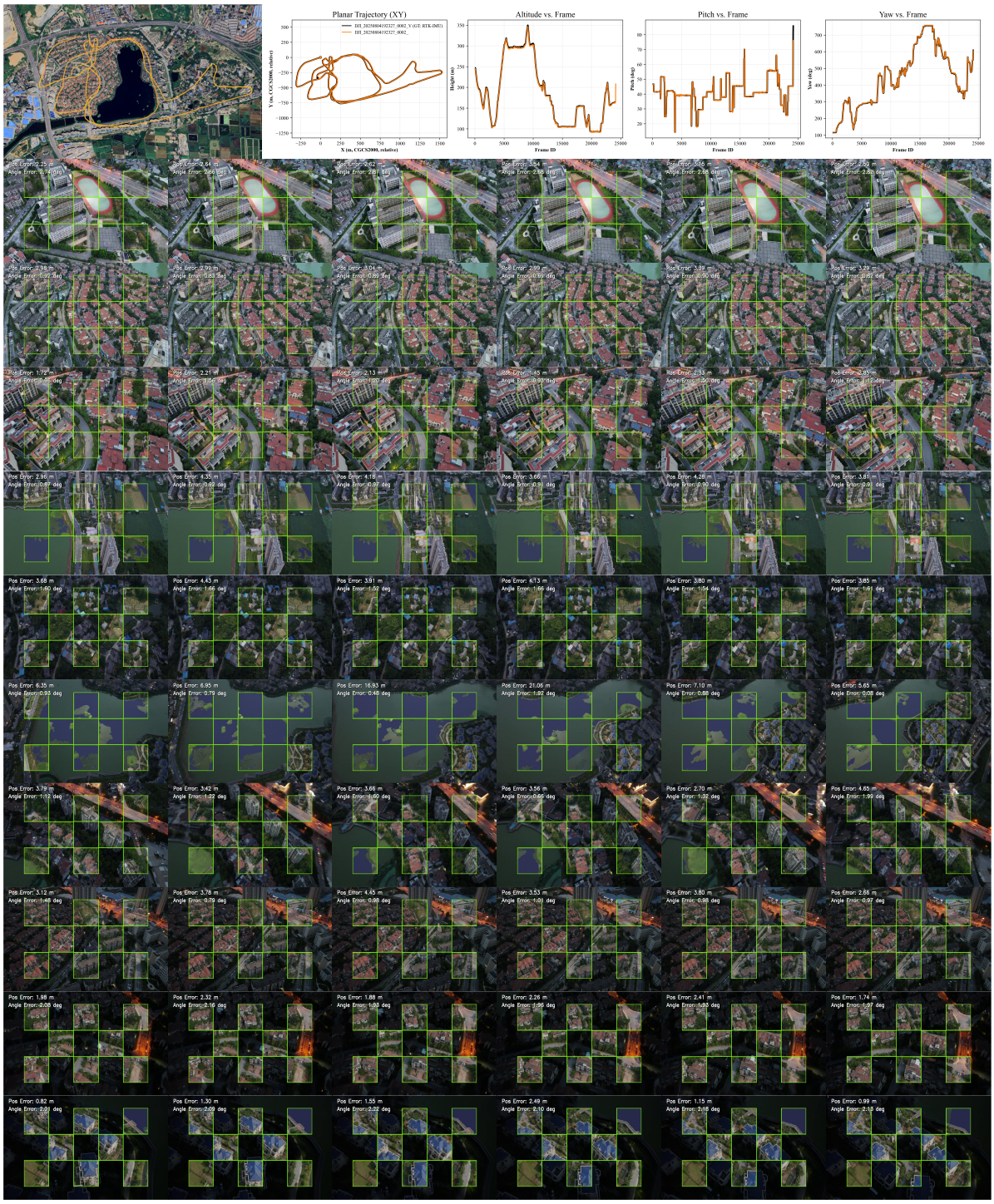

Figure 11. **Trajectory estimation results on *Long Trajectory Flights*.** We compare our method's estimated trajectory (orange) against the ground truth from an RTK-IMU system (black). The plots show high consistency for planar position (XY), altitude, pitch, and yaw. The checkerboard insets provide an AR visualization, which overlays the live camera views with the rendered views based on our estimated pose.

[2] Rouwan Wu, Xiaoya Cheng, Juelin Zhu, Yuxiang Liu, Maojun Zhang, and Shen Yan. Uavd4l: A large-scale dataset for uav 6-dof localization. In *3DV*, 2024. 5

Figure 12. **Trajectory estimation results on various synthetic scenes from the *SynthCity-6* dataset.** The figure showcases our method's performance across diverse synthetic conditions, including night (*Switzerland-seq4, -seq7*), cloudy (*Switzerland-seq12*), sunny (*USA-seq2*), and foggy (*USA-seq5*).

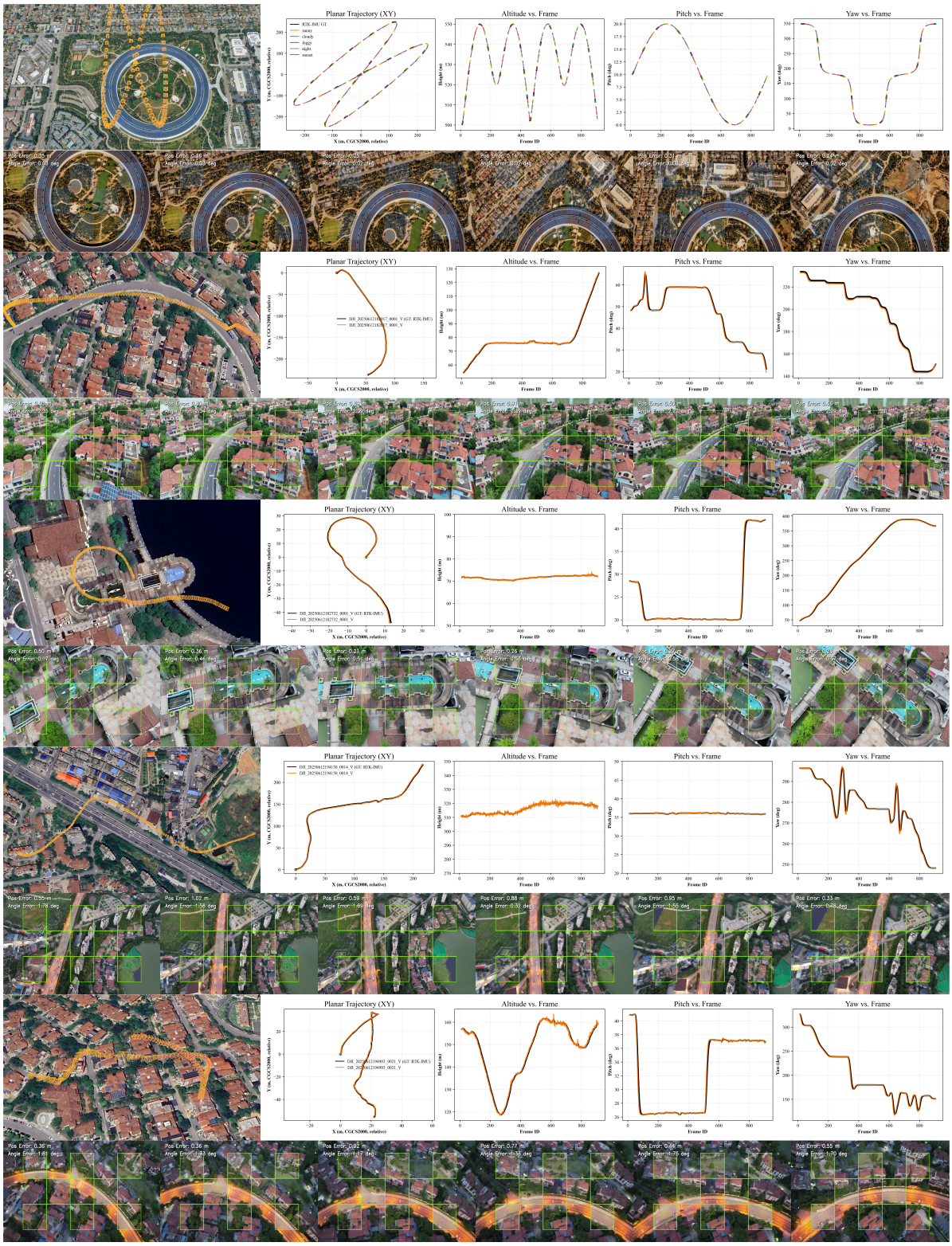

Figure 13. **Trajectory estimation results on challenging real-world and long-term scenarios.** This figure demonstrates the model's robustness by evaluating on: (1) a synthetic sunset scene from *SynthCity-6 (USA-seq8)*, and (2) 4 real-world sequences from the *UAVD4L-2yr* dataset, which include challenging day/night conditions (*seq2, seq3, seq6, seq8*).

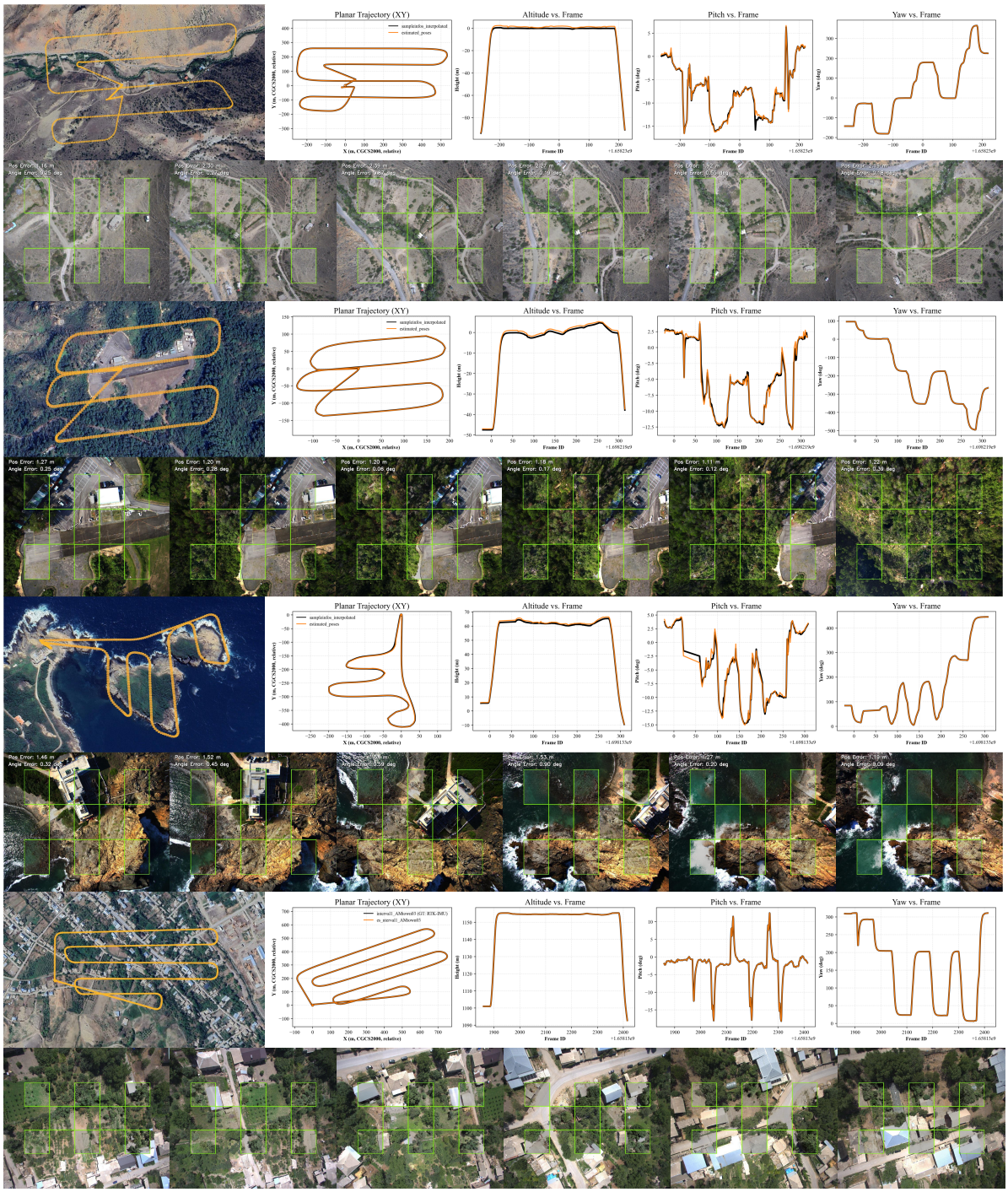

Figure 14. **Generalization performance on the standard *UAVScenes* benchmark dataset.** This figure validates our model's strong generalization capability on four diverse, unseen scenes from the public UAVScenes benchmark. The scenes cover a wide range of environments: a town (*AMtown*), a natural valley (*AMvalley*), an airport (*HKairport*), and an island (*HKisland*).

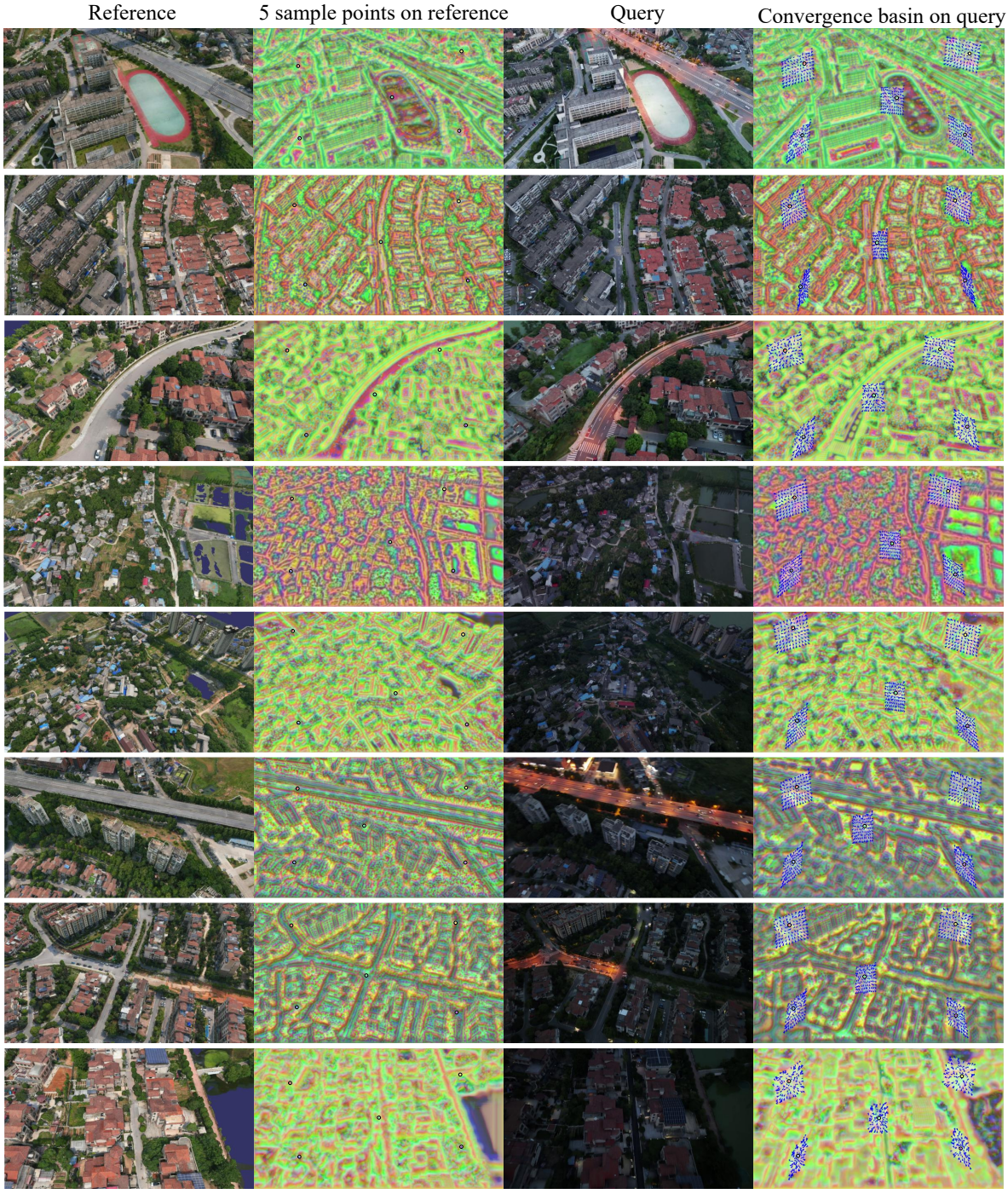

Figure 15. **Additional examples of the refinement process and its convergence basin.** For various anchor points sampled in the reference image, the initial estimates in the query image (covering a wide area, shown as a field of arrows) are effectively guided by our learned features to converge to a single, precise location.

Table 4. **Summary of SynthCity-6 dataset.** Each trajectory provides camera frames with synchronized 6-DoF poses under varying *weather, illumination, and altitude* conditions.

Sequence	Weather / Illumination	Altitude (m)	Per Trajectory	Total Camera Frames	Location
Switzerland-seq4	Foggy/Night/Rainy/Sunny/Sunset	200/500	900	9000	Thun, Switzerland
Switzerland-seq7	Foggy/Night/Rainy/Sunny/Sunset	200/500	900	9000	Kirchlindach, Switzerland
Switzerland-seq12	Foggy/Night/Rainy/Sunny/Sunset	200/500	900	9000	Lausanne, Switzerland
USA-seq2	Foggy/Night/Cloudy/Sunny/Sunset	200/500	900	9000	Chicago, USA
USA-seq5	Foggy/Night/Cloudy/Sunny/Sunset	200/500	900	9000	New York, USA
USA-seq8	Foggy/Night/Cloudy/Sunny/Sunset	200/500	900	9000	California, USA
Total	–	–	–	54k	–

Table 5. **Summary of UAVD4L-2yr dataset.** Each sequence contains camera frames captured under varying *illumination and altitude*, covering different *scene types* with annotated primary *target objects*.

Sequence	Illumination	Altitude (m)	Camera Frames	Scene Type & Environment	Target Object
UAVD4L-2yr-seq1	Daytime	70	900	suburban	dynamic person
UAVD4L-2yr-seq2	Daytime	70	900	urban	dynamic vehicles
UAVD4L-2yr-seq3	Daytime	70	900	urban	static landmark
UAVD4L-2yr-seq4	Daytime	110	900	suburban	dynamic vehicles
UAVD4L-2yr-seq5	Night	220	900	urban	dynamic vehicles
UAVD4L-2yr-seq6	Night	310	900	urban	dynamic vehicles
UAVD4L-2yr-seq7	Night	150	900	suburban	dynamic vehicles
UAVD4L-2yr-seq8	Night	150	900	urban	dynamic vehicles
Total	–	–	7.2k	–	–

Table 6. **Summary of UAVD4L-SynTarget dataset.** Each sequence contains camera frames captured under varying *weather/illumination* conditions and *flight altitudes*, with multiple *target objects* annotated along with their quantities.

Sequence	Weather / Illumination	Altitude (m)	Camera Frames	Target Object Number
UAVD4L-SynTarget-seq1	Sunny	130	304	$>$100
UAVD4L-SynTarget-seq2	Sunny	130	344	$>$100
UAVD4L-SynTarget-seq3	Sunny	180	1136	$>$100
UAVD4L-SynTarget-seq4	Foggy	180	1136	$>$100
UAVD4L-SynTarget-seq5	Night	180	1136	$>$100
UAVD4L-SynTarget-seq6	Foggy	150	1192	$>$100
UAVD4L-SynTarget-seq7	Sunny	240	808	$>$100
Total	–	–	6k	–