LoD-Loc v3: Generalized Aerial Localization in Dense Cities using Instance Silhouette Alignment

Supplementary Material

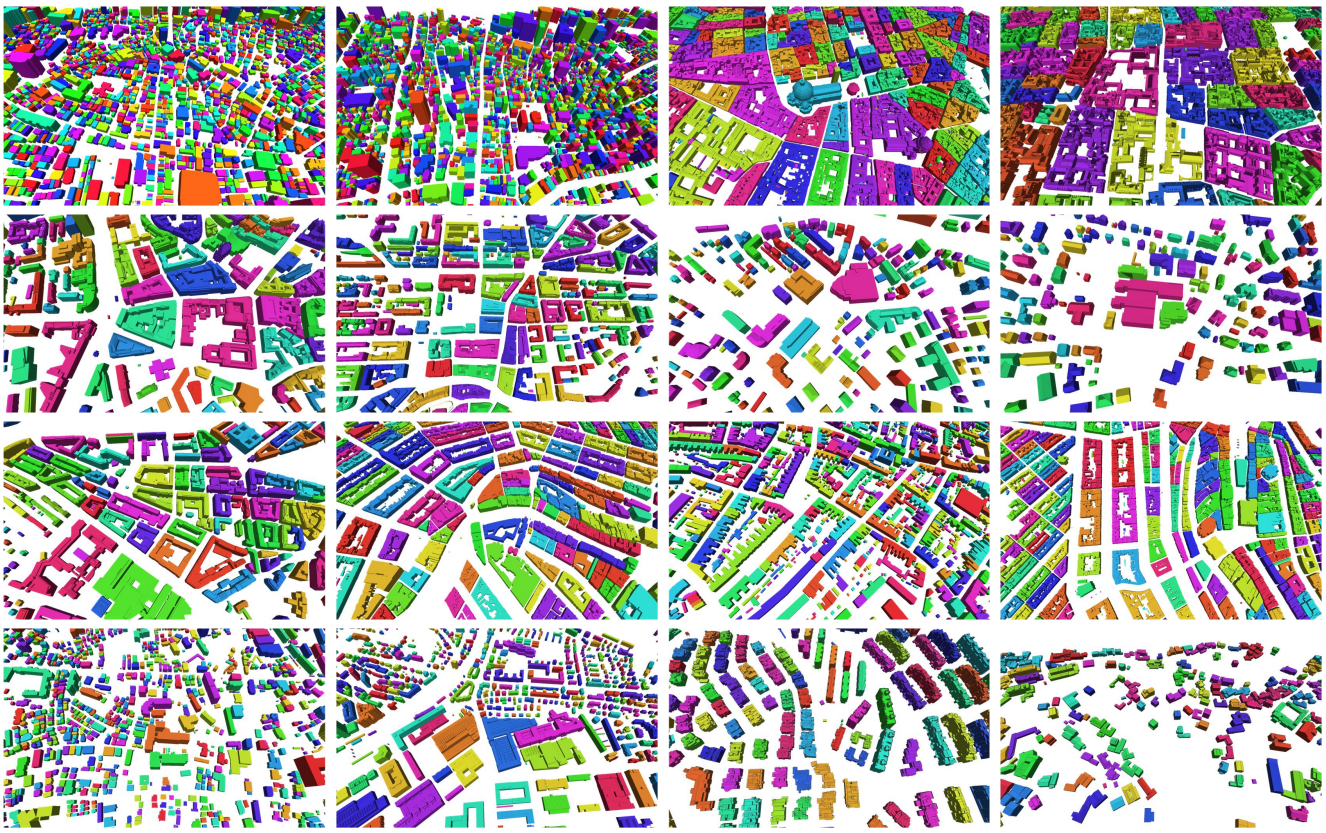

Figure 1. **Visualization of the instance LoD models in the InsLoD-Loc dataset.** These models, generated via our automated instancing pipeline, represent a subset of the 40 geographical regions across six countries. Each distinct color within an LoD model corresponds to a unique building instance identified through topological analysis.

1. Demo Video

The supplementary material includes a demonstration video showcasing our method's localization performance in challenging cross-scene and dense scenarios. This video features three distinct aerial sequences varying in region, altitude, and viewing angle. We visualize the results by overlaying the projected instance LoD model onto the query images based on the estimated poses. The results demonstrate that our approach effectively addresses poor cross-scene generalization and ambiguity challenges for UAVs, significantly outperforming SOTA baselines.

2. Instance LoD Model

This section provides a detailed exposition of the topological structure analysis method used for LoD model instancing. The primary objective is to programmatically assign a unique identifier to each building within a untextured 3D city model $\mathcal{M}$, thereby enabling instance-level rendering. Our method transforms this challenge into a graph partitioning problem by first interpreting the input $\mathcal{M}$ as a geometric graph $G = (V, E, F)$, where each building instance corresponds to a topologically connected component.

The core of the instancing process is to partition the model's faces into a set of disjoint subsets $\{B_i\}_{i=1}^{M}$, where each subset represents a single building. This partitioning is formally defined as:

$$\mathcal{M} = \bigcup_{i=1}^{M} B_i, \quad \text{s.t.} \quad B_i \cap B_j = \emptyset \quad \forall i \neq j \tag{1}$$

To achieve this, we first construct a face adjacency graph where each face $f \in F$ is a node, and an edge connects

two nodes if their corresponding faces share an edge in the 3D model. With this graph established, we apply a Breadth-First Search (BFS) algorithm to identify its connected components. Each discovered component, being a single contiguous mesh, is treated as an individual building instance B_i. This approach is highly effective for standard city models where buildings are represented as closed, non-intersecting meshes.

Upon completing the partitioning, we proceed to the instance identification and coloring stage. Each B_i is assigned a unique 24-bit integer identifier, id_i, a high-bit-depth space that can accommodate up to 16.7 million unique instances. This identifier is then deterministically mapped to a unique RGB color vector $\mathbf{c}_i$ via the bijective mapping function Φ:

$$\mathbf{c}_i = \Phi(id_i) \triangleq \begin{pmatrix} \lfloor id_i \cdot 256^{-2} \rfloor \\ \lfloor id_i \cdot 256^{-1} \rfloor \pmod{256} \\ id_i \pmod{256} \end{pmatrix} \quad (2)$$

Subsequently, this color is assigned as the diffuse material attribute for all faces belonging to instance B_i, generating the final instance-level LoD model, $\mathcal{M}_{ins}$.

The **InsLoD-Loc** dataset covers 40 distinct UAV flight areas across six countries in Europe and Asia, including Japan, Switzerland, China, France, Italy, and the Netherlands. Fig. 1 provides a visualization of the corresponding instanced LoD models for partial regions within this dataset.

3. Architecture of the Segmentation Module

This section provides a detailed architecture of our building silhouette instance segmentation module. Inspired by the prompt learning paradigm introduced by RSPrompter [2], we fine-tune the Segment Anything Model (SAM) [6] for the specific task of identifying building instances in aerial imagery. As illustrated in Fig. 2, the module consists of three main components: a frozen SAM image encoder, a trainable prompter module, and the original lightweight SAM mask decoder. The primary objective is to train the prompter module to automatically generate task-specific prompts that guide the SAM decoder to produce precise instance masks for all buildings within an input image I_q.

The complete data flow is illustrated in Fig. 2. We utilize the pre-trained and frozen `ViT-Huge` from SAM as the image encoder. This encoder extracts a final, high-dimensional image embedding and a set of intermediate feature maps. The core of the adaptation is the trainable prompter module, which is designed to be lightweight. It processes the intermediate features to automatically predict a set of task-specific prompts (e.g., points or boxes) and their corresponding classification scores. Finally, the original lightweight SAM mask decoder takes the main image embedding and these learned prompts as input to generate the final set of building instance masks.

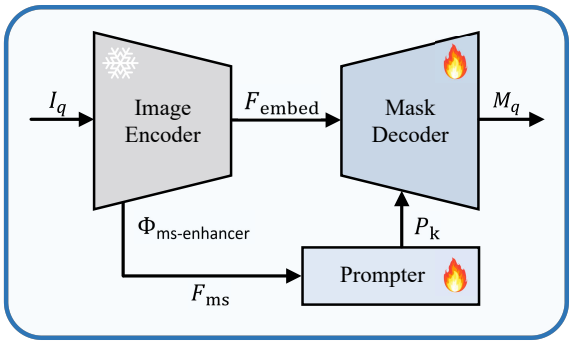

Figure 2. **The overall architecture of our segmentation module [2].** The process begins with feature extraction by the frozen SAM encoder, followed by the generation of task-specific prompts via the learnable prompter module, and culminates in mask prediction by the SAM decoder.

4. Details of the InsLoD-Loc Dataset

The **InsLoD-Loc** dataset contains 108,109 RGB-mask image pairs with 2,674,794 corresponding dense, pixel-perfect instance-level annotations. Fig. 3 and Fig. 4 showcase representative RGB images and their corresponding masks, while Tab. 1 provides a detailed breakdown of each area within the dataset. This section elaborates on the construction pipeline and composition of the **InsLoD-Loc** dataset.

4.1. Data Generation Pipeline

The generation of our dataset is a two-stage process involving a high-fidelity simulation environment for creating photorealistic RGB images and a parallel rendering pipeline for producing precisely aligned instance masks.

Photorealistic RGB Image Generation. To generate high-quality query images, we employed a comprehensive simulation framework built upon Unreal Engine 5 (UE5) version 5.1.1 [5], integrated with Microsoft Research's AirSim plugin version 2.1.0 [12]. AirSim provides a physics-accurate simulation environment that enables the development and validation of complex navigation algorithms in controlled virtual settings. The platform leverages UE5's advanced rendering technologies, such as the Nanite virtualized geometry system for high-polygon assets and the Lumen global illumination system for physically-based dynamic lighting, to achieve photorealism. Fig. 5 provides a snapshot of the drone rendering interface within our UE5 simulation environment.

For geospatially accurate environment construction, the Cesium for Unreal plugin [1] was used to stream high-resolution Google Photorealistic 3D Tileset data, enabling the creation of large-scale environments spanning up to 50km $\times$ 50km. The communication architecture utilizes TCP/IP protocols with Protocol Buffer (Protobuf) serialization for command transmission, while UDP protocols han-

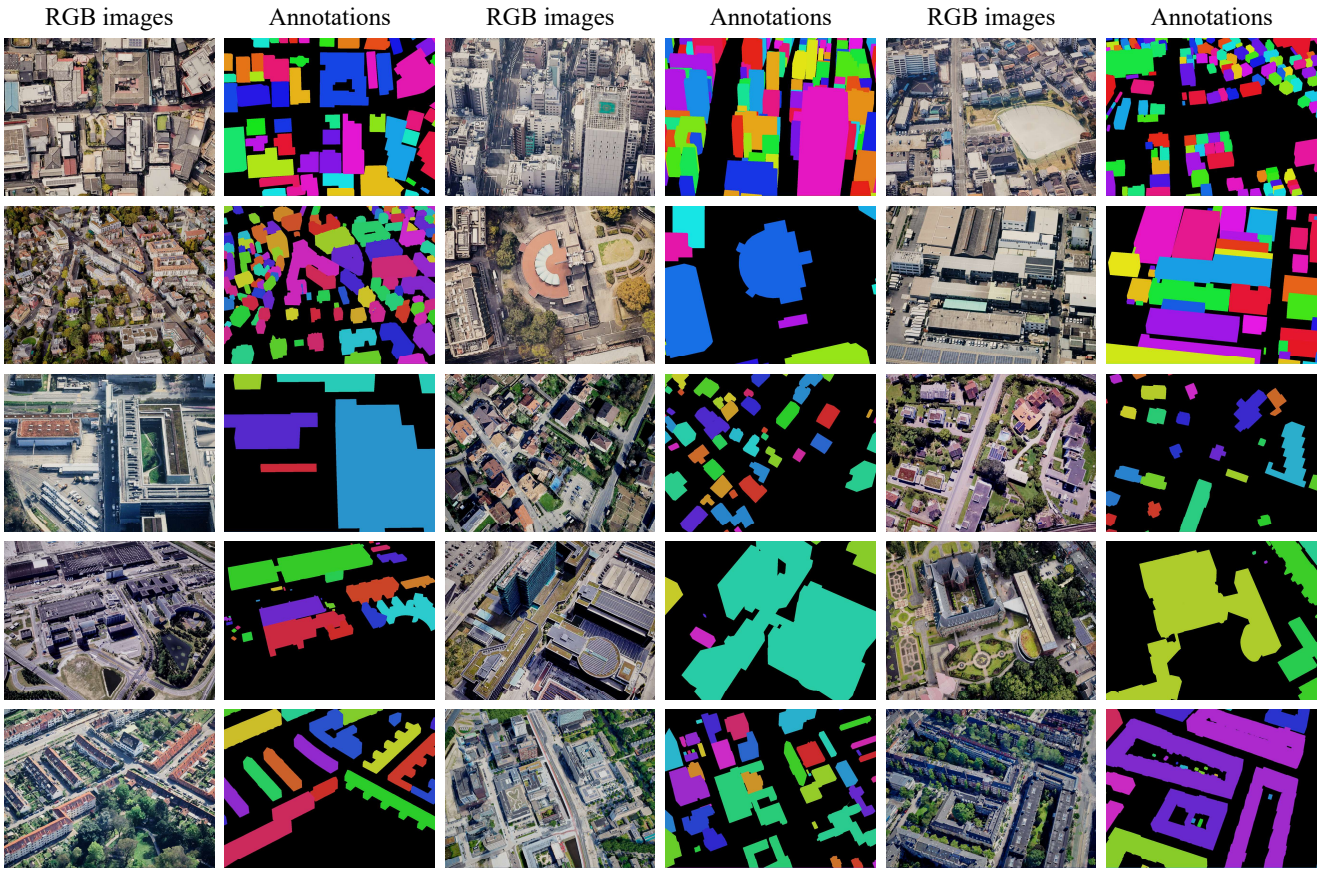

Figure 3. **Visualization of training samples from the InsLoD-Loc dataset.** The figure displays pairs of synthetic RGB images and their corresponding annotations. The first two rows are from the Japan region training set, the middle two rows are from the Switzerland region training set, and the last row is from the Netherlands training set.

dle real-time state information to minimize latency. Custom blueprint classes were developed to synchronize point-by-point trajectory execution with video streaming, ensuring precise data capture along predefined flight paths.

Automated Instance Mask Generation. To obtain instance masks that are pixel-perfectly aligned with the rendered RGB images, we developed a parallel rendering pipeline using a custom library based on OpenSceneGraph (OSG) [9], which is optimized for real-time rendering. The process begins by sourcing the corresponding LoD models [3, 7, 8, 16, 18] for the Google 3D Tileset data. These models are first converted into the OBJ mesh format. Crucially, their coordinate systems are rigorously unified into the EPSG:4978 framework to ensure strict spatial consistency with the simulation environment in UE5.

Subsequently, we utilize the instanced LoD model, $\mathcal{M}_{ins}$ (as described in Supp. Sec. 2), where each building has a unique color ID. For each rendered RGB frame from the UE5 environment, OSG renders a corresponding frame using the exact same camera intrinsic and extrinsic parameters. This guarantees perfect spatial alignment between the photorealistic image and the output, which is an instance map where pixel colors encode object IDs.

Finally, these rendered maps undergo a Connected Component Analysis (CCA) to generate pixel-perfectly aligned instance mask annotations. In image processing, a connected component S is defined as a set of foreground pixels where for any two pixels $p, q \in S$, there exists a path between them consisting entirely of foreground pixels. This algorithm is designed to partition the complete set of foreground pixels P_{fg} into individual connected components:

$$P_{fg} = \bigcup_{i=1}^{N} S_i, \quad \text{s.t.} \quad S_i \cap S_j = \emptyset \quad \forall i \neq j \tag{3}$$

where N is the total number of independent connected components in the image. In our context, all pixels of the same unique color within the rendered instance map are treated as a single component S_i. This approach allows for the efficient and precise generation of noise-free, instance-level ground-truth annotations for our large-scale dataset.

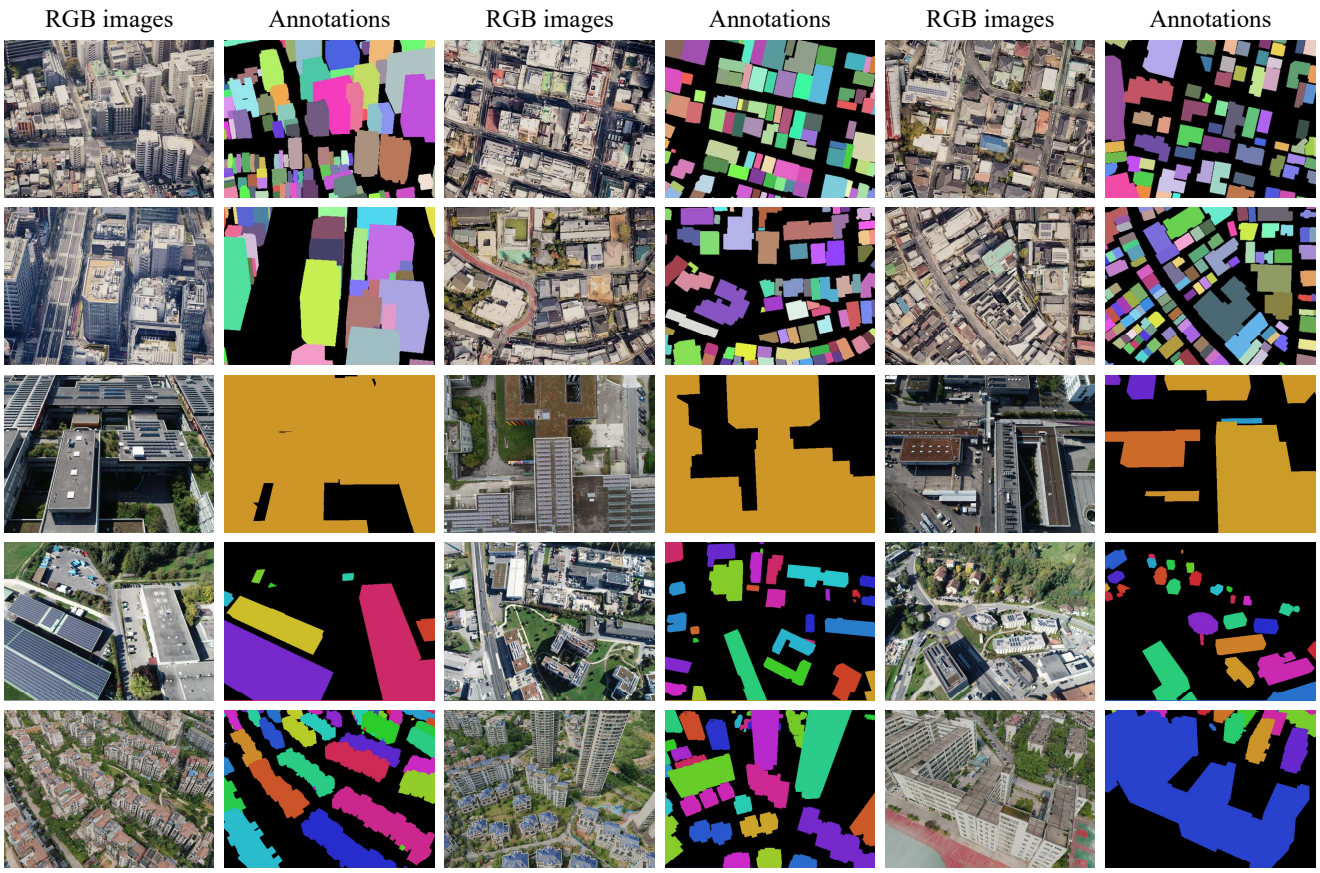

Figure 4. **Visualization of samples from the test datasets.** The first two rows show samples from Tokyo-LoDv3, the middle two rows from Swiss-EPFLv2, and the last row from UAVD4L-LoDv2.

4.2. Dataset Acquisition and Composition

The **InsLoD-Loc** dataset is designed to provide a comprehensive representation of diverse real-world environments. It incorporates a wide range of land-use categories, including commercial, industrial, and residential, as well as educational, facilities, and suburban zones. Fig. 6 illustrates the distribution of these land-use types across the dataset and provides representative examples. To ensure statistical independence between the training and testing subsets while maintaining a faithful class distribution, flight trajectories were designed as spatially adjacent but non-overlapping sequences. Furthermore, multi-altitude and multi-perspective data collection protocols were implemented to enhance dataset robustness and promote algorithmic generalization, providing a solid foundation for advancing drone-based computer vision research.

5. Details of Baseline

CadLoc. The implementation of CadLoc [11] adheres to the two-stage localization pipeline, involving retrieval and matching to determine the camera pose. Our experimental protocol is based on [17], utilizing a subset of the UAVD4L dataset [14] as the reference image database. The initial retrieval stage employs a sensor-guided approach to constrain the search space based on the following criterion:

$$^{q}\mathcal{I} = \{\mathbf{I}_i^r \mid \|\mathbf{t}_i^r - \mathbf{t}^q\| \leq \delta_t \wedge \arccos(\mathbf{R}_i^r, \mathbf{R}^q) \leq \delta_o\} \quad (4)$$

Here, $\|\cdot\|$ denotes the Frobenius Norm for translation matrices, and $\arccos(\cdot)$ computes the angular difference between rotation matrices. In accordance with [17], the thresholds for translation δ_t and orientation δ_o are set to 150 and 30, respectively, to filter candidates based on spatial proximity and rotational similarity effectively.

MC-Loc. We utilize the default configuration of MC-Loc [13], benchmarking with both DINOv2 [10] and RoMa [4] as feature extractors. The resolution for both input images and rendered candidates is maintained at $(256, 340)$. The method's three-stage optimization is configured with 40, 30, and 60 iterations, generating 52, 40, and 32 pose candidates in each respective stage. A dual-beam search is employed for the first two stages, transitioning to a single-beam search for the final stage. Further implementation details can be found in the original paper [13].

Name	Area	Images	Sequence	Height[m]	Proportion[km^2]	Size[$pixel$]	View Angle[°]
Netherlands_01	Amsterdam	3203	✓	500	18.3879	1600×1200	[0, 30]
Netherlands_02	Amsterdam	3182	✓	200	17.3980	1600×1200	[30, 50]
Netherlands_03	Hague	3209	✓	300	16.9807	1600×1200	[0, 30]
Netherlands_04	Delft	3075	✓	200	20.1190	1600×1200	[0, 30]
Netherlands_05	Amsterdam	1747	✓	500	16.9844	1600×1200	[30, 50]
Netherlands_06	Tilburg	3151	✓	200	19.7211	1600×1200	[0, 30]
Switzerland_01	Basel	4771	×	300, 400	1.7458	1920×1080	0, 45
Switzerland_02	Ecublens	13563	×	400, 500	8.6432	1920×1080	0, 45
Switzerland_03	Zurich	1783	✓	300	13.0968	1600×1200	[50, 70]
Switzerland_04	Zurich	3209	✓	200	17.9719	1600×1200	[0, 30]
Switzerland_05	Lausanne	3210	✓	200	8.8370	1600×1200	[30, 50]
Switzerland_06	Geneva	3210	✓	200	17.9286	1600×1200	[0, 30]
Switzerland_07	Bern	1916	✓	300	15.9526	1600×1200	[30, 50]
Switzerland_08	Bern	3209	✓	200	17.5524	1600×1200	[50, 70]
Switzerland_09	Lucerne	3210	✓	200	15.4045	1600×1200	[0, 30]
Switzerland_10	Zurich	3210	✓	200	17.2618	1600×1200	[30, 50]
Switzerland_11	Geneva	3179	✓	200	2.7146	1600×1200	[30, 50]
Switzerland_12	Geneva	3202	✓	300	0.7120	1600×1200	[0, 30]
Switzerland_13	Geneva	3209	✓	200	13.5406	1600×1200	[0, 30]
Switzerland_14	Bill	3210	✓	200	17.2042	1600×1200	[0, 30]
Switzerland_15	Lausanne	4300	✓	500	10.1067	1600×1200	[0, 30]
Japan_01	Osaka	2214	×	250, 350	2.6631	1920×1080	0, 45
Japan_02	Tokyo	1440	×	350, 450	0.4028	1920×1080	0, 45
Japan_03	Tokyo	3260	×	350, 450	0.9747	1920×1080	0, 45
Japan_04	Tokyo	682	×	350, 450	0.7171	1920×1080	0, 45
Japan_05	Tokyo	1000	×	350, 450	0.3302	1920×1080	0, 45
Japan_06	Tokyo	495	×	350, 450	0.6754	1920×1080	0, 45
Japan_07	Tokyo	875	×	350, 450	0.3283	1920×1080	0, 45
Japan_08	Tokyo	750	✓	450, 500	1.3225	1920×1080	0, 45
Japan_09	Tokyo	381	✓	450, 500	0.8281	1920×1080	0, 45
Japan_09*	Tokyo	381	✓	450, 500	0.8281	1920×1080	0, 45
Japan_10	Tokyo	450	✓	450, 500	0.9025	1920×1080	0, 45
Japan_11	Tokyo	450	✓	450, 500	0.8464	1920×1080	0, 45
China_01	Changsha	1604	✓	120	1.6384	4056×3040	0, 45
China_02	Changsha	2192	✓	[90, 150]	1.4884	5280×3956	[30, 60]
Italy_01	Florence	155	×	400, 500	0.8778	1920×1080	0, 45
Italy_02	Florence	2631	×	400, 500	0.1571	1920×1080	0, 45
France_01	Florence	421	×	400, 500	0.4865	1920×1080	0, 45
France_02	Paris	222	×	400, 500	0.2751	1920×1080	0, 45
France_03	Paris	2764	×	400, 500	0.7890	1920×1080	0, 45

Table 1. **Detailed characteristics of the regional datasets composing the InsLoD-Loc dataset.** This table provides a comprehensive breakdown of each geographical area, detailing its name, image count, sequence status, flight altitude range in meters, total geographical area in square kilometers, image resolution in pixels, and captured viewing angles in degrees.

LoD-Loc. The implementation of LoD-Loc [17] follows its default settings. This includes uniform sampling counts of 10, 10, 30, and 8 along the x, y, z, and yaw axes, respectively, and a lambda value of 0.8. The primary modification in our work is the use of instantiated LoD1/1.3 models for the experiments, with a 3D sampling interval of 1 meter.

LoD-Loc v2. The implementation of LoD-Loc v2 [18] follows its default settings. The input image resolution for the segmentation module is set to $(1024, 1024)$. In the coarse pose selection stage, mask resolutions for the query and rendered candidates are $(640, 360)$, $(602, 448)$, and $(720, 480)$, with a sampling interval of 10 meters. The subsequent fine

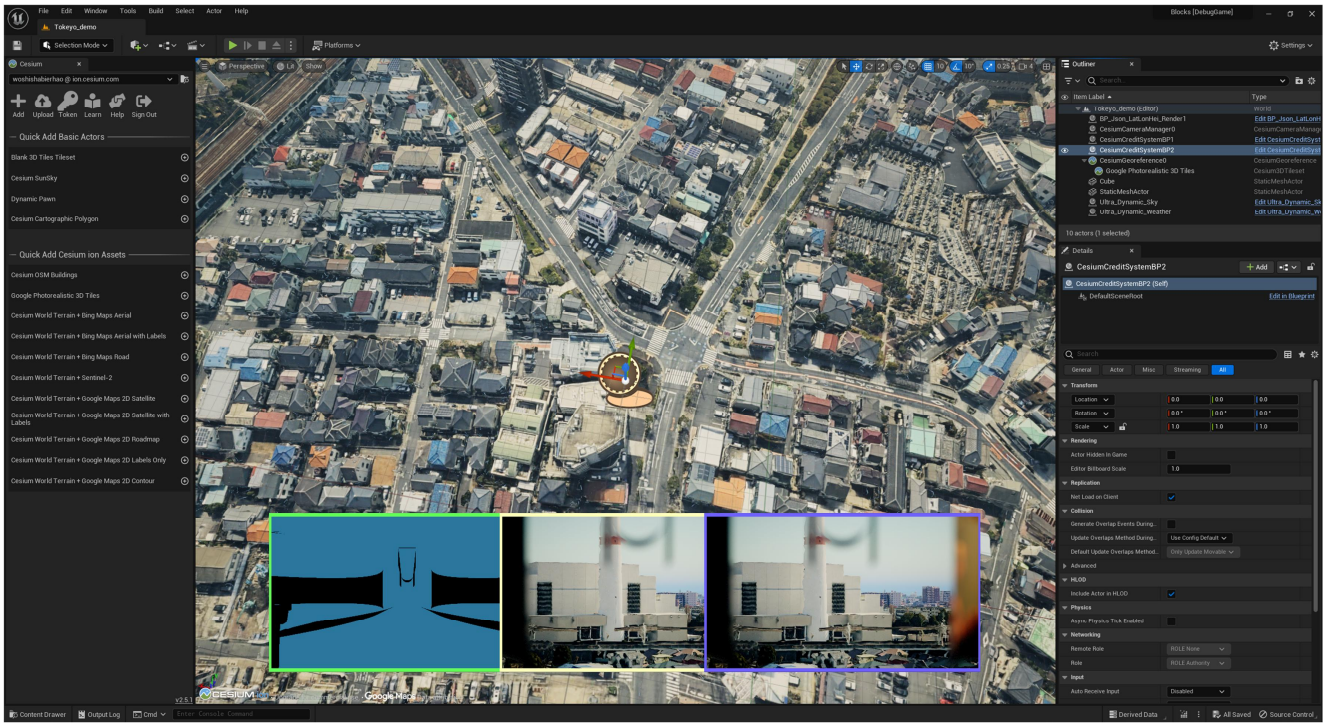

Figure 5. **Snapshot of the AirSim drone rendering interface within the Unreal Engine 5 environment.** This interface demonstrates the photorealistic rendering quality used for generating RGB query images.

Dataset	*in (Grid)-Traj. (Place.)*			*out-of (Sequence)-Traj. (Place.)*		
	Image	Density	Ratio (%)	Image	Density	Ratio (%)
UAVD4L-LoDv2	1604	14.50	43.34	2192	27.78	44.55
Swiss-EPFLv2	712	3.67	49.44	379	45.51	27.93
Tokyo-LoDv3	1370	105.80	74.70	1581	94.85	67.14

Table 2. **Statistics of the test datasets.** The table details the number of images, the building density calculated as the average number of instances per image, and the annotation ratio, which represents the average annotated building area relative to the total image area.

Method	Mem.(MB)		Time(s)			
	Sem.	Loc.	Sem.	Coarse	Fine	Total
LoD-Loc v3	5066	1860	0.26	0.34	3.03	3.63

Table 3. **Computational Efficiency.** Breakdown of runtime and memory footprint across different pipeline stages, evaluated on a single NVIDIA RTX 4090 GPU.

Epoch	IoU	AP	*in-Traj.*
5	0.808	0.332	87.30/95.30/98.00
10	0.828	0.408	93.50/97.00/98.40
20	0.889	0.556	97.60/99.10/99.70

Table 4. **Ablation on Segmentation Quality.** Localization performance evaluated across different training epochs.

pose estimation stage uses the same mask resolutions and runs for 40 iterations with 2 beams. Angular perturbations are set to 2°, while translational perturbations are drawn from a Gaussian distribution $X \sim \mathcal{N}(0, \sigma^2)$ where $\sigma = 1.5$. A decay factor of $\gamma = 0.3$ is applied, generating $n = 52$ candidates per iteration. Furthermore, to clarify the baseline variants presented in our evaluation tables, **no select** refers to the model skipping the coarse pose selection stage, and **no refine** corresponds to the model disabling the fine pose estimation stage.

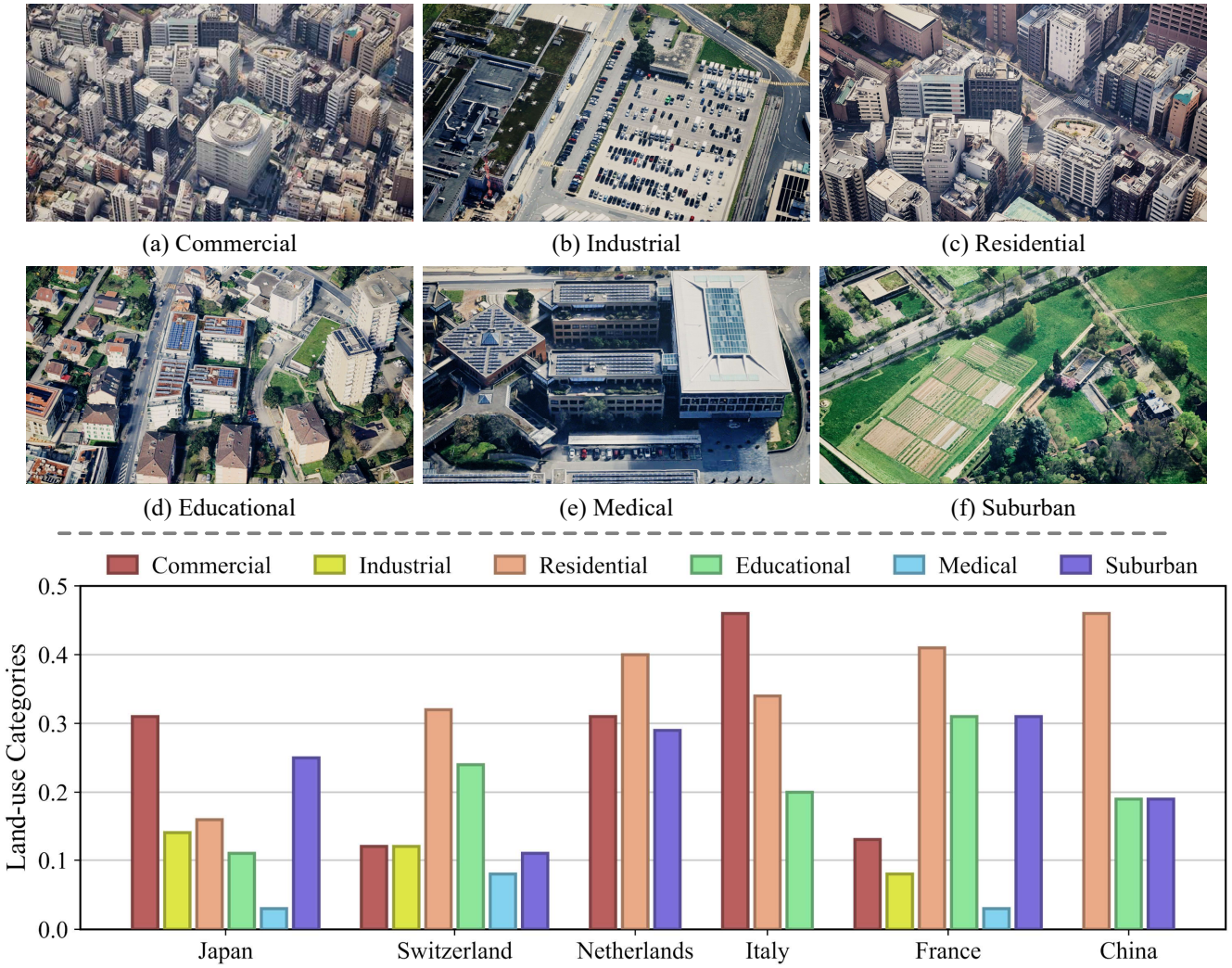

Figure 6. **Composition and examples of land-use types within the InsLoD-Loc dataset.** The top panel, subfigures (a)-(f), showcases representative RGB image examples for each category. The bottom panel presents a bar chart illustrating the proportion of each land-use type across the different countries in the dataset.

Method	*Japan_09**			
	2m-2°	3m-3°	5m-5°	T.e./R.e.
Prior	0.52	0.79	7.35	8.14/1.80
LoD-Loc v2	1.80	3.10	8.70	14.69/0.99
LoD-Loc v3	**24.70**	**55.10**	**93.70**	**2.86/0.23**

Table 5. **Quantitative comparison of different methods on the Japan_09*.** The evaluation was conducted under simulated adverse weather conditions, rain, and thunderstorms. The results demonstrate the robustness of our approach.

6. Details of Experiments

6.1. Visualization of Training Data

The training dataset comprises pairs of synthetic RGB images and their corresponding pixel-aligned instance masks.

For the synthetic images, the training query set is derived from the designated training split of our **InsLoD-Loc** dataset, comprising a total of 88,493 RGB images.

For the building masks, we apply CCA to the corresponding instance masks, resulting in 2,325,232 instance annotations in total. Fig. 3 provides a visualization of selected samples from the training dataset.

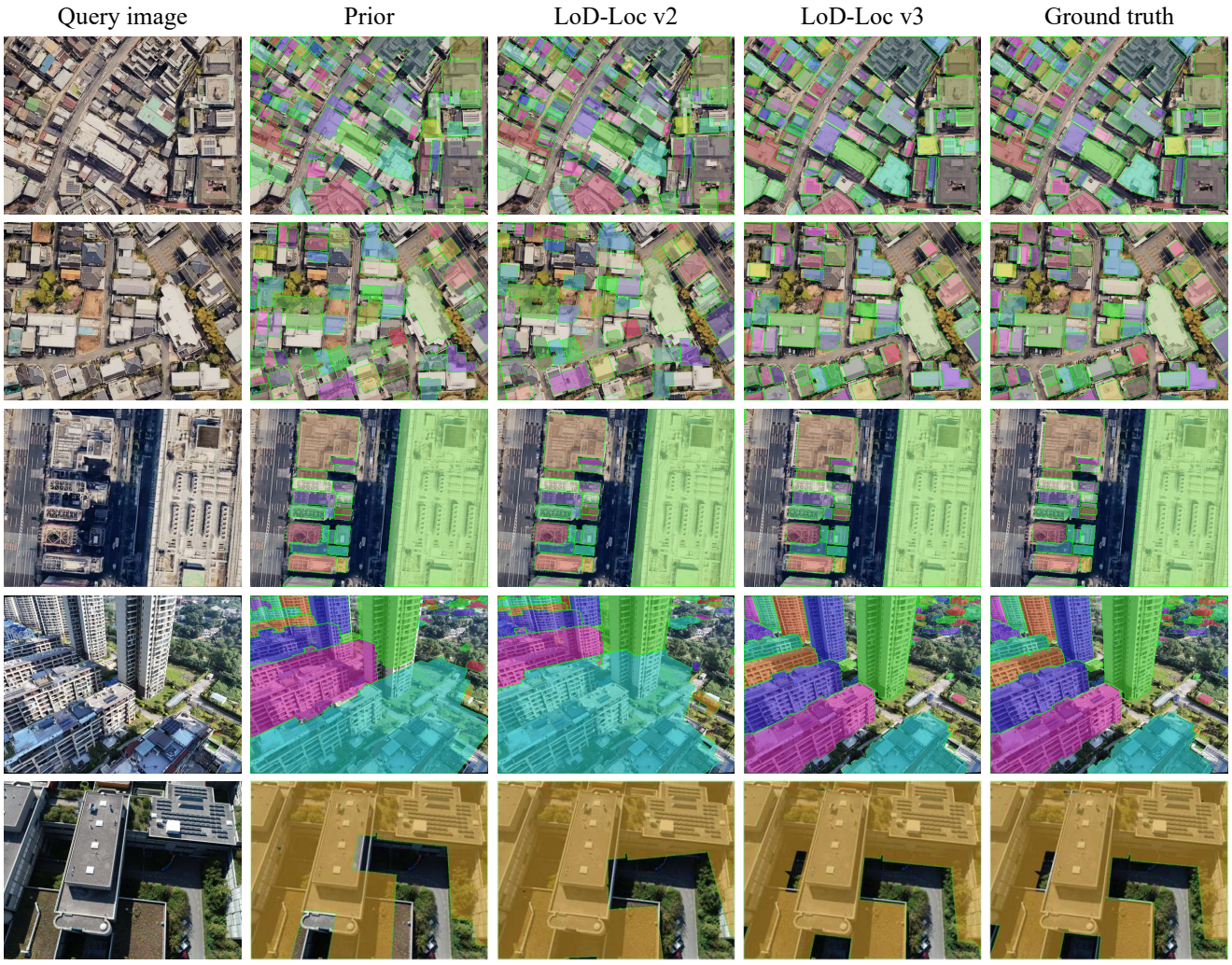

Figure 7. **Qualitative results of the ablation study on the three test datasets.** The visualization results demonstrate that our method not only possesses excellent generalization capabilities but also effectively addresses the challenge of localization failure in dense building scenes. The query images in the first two rows are from the Tokyo-LoDv3 dataset, while those in the bottom three rows are from UAVD4L-LoDv2 and Swiss-EPFLv2, respectively.

6.2. Visualization of Test Data

The test data is composed of three parts: two real-world datasets, UAVD4L-LoDv2 and Swiss-EPFLv2, and one synthetic dataset, Tokyo-LoDv3.

The UAVD4L-LoDv2 dataset covers an area of 2.5 km^2 and includes a geo-referenced LoD1 model, which was automatically generated from the textured mesh model of UAVD4L using the DP modeler [15]. The Swiss-EPFLv2 dataset provides a semi-automatically constructed LoD1 model covering an 8.2 km^2 area. The query images for these two datasets originate from the UAVD4L-LoD and Swiss-EPFL datasets [17], respectively.

Tokyo-LoDv3 is the test split from our proposed **InsLoD-Loc** dataset, covering an area of 4.06 km^2. It contains five distinct types of dense urban scenes with corresponding instanced LoD1 models, designated as *Japan_06*, *Japan_07*, *Japan_08*, *Japan_09*, and *Japan_10*.

Fig. 4 visualizes selected samples from the test datasets. Tab. 2 provides detailed statistics for the three test sets, including the number of images, building density, and annotation ratio for each.

6.3. Efficiency and Runtime

To assess the computational efficiency of our proposed pipeline, we profile its runtime and memory footprint. Evaluated on a single NVIDIA RTX 4090 GPU, the rendering and instance alignment stages require only 685μs and 1.2ms, respectively. A comprehensive breakdown of the memory usage and execution time across all pipeline stages

is provided in Tab. 3.

6.4. Additional Ablation Studies

In addition to the ablation studies presented in the main text, we provide further qualitative and quantitative evaluations to validate our design choices.

Silhouette Representation. We present qualitative localization results on the UAVD4L-LoDv2, Swiss-EPFLv2, and Tokyo-LoDv3 datasets, as illustrated in Fig. 7. These visual comparisons intuitively confirm the superiority of instance-level silhouette representation in resolving semantic ambiguities when compared to the baseline method.

Segmentation Quality. To evaluate the impact of segmentation quality, we measure localization performance using model checkpoints from various training epochs. As reported in Tab. 4, higher mask accuracy directly correlates with improved final localization performance.

6.5. Evaluation under Extreme Weather Conditions

To further evaluate the robustness of our method in more complex environments, we conducted additional localization experiments under simulated Extreme weather conditions. Specifically, we simulated rain and thunderstorm effects within the UE5 environment while rendering the flight trajectory for the *Japan_09* region. Tab. 5 demonstrates that our proposed localization method maintains robust performance even under these challenging conditions. The specific visual results for this scenario are presented in the supplementary demo video.

References

[1] CesiumGS. Cesium GS, 2024. https://cesium.com/platform. 2

[2] Keyan Chen, Chenyang Liu, Hao Chen, Haotian Zhang, Wenyuan Li, Zhengxia Zou, and Zhenwei Shi. RSPrompter: Learning to prompt for remote sensing instance segmentation based on visual foundation model. *IEEE Transactions on Geoscience and Remote Sensing*, 62:1–17, 2024. 2

[3] Delft University of Technology. 3D BAG, 2025. https://3d.bk.tudelft.nl/projects/3dbag/. 3

[4] Johan Edstedt, Qiyu Sun, Georg Bökman, Mårten Wadenbäck, and Michael Felsberg. RoMa: Robust dense feature matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19790–19800, 2024. 4

[5] Epic Games. Unreal Engine, 2025. https://www.unrealengine.com/en-US. 2

[6] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023. 2

[7] Transport Ministry of Land, Infrastructure and Urban Policy Division Tourism. PLATEAU (3D City Model), 2020. https://www.geospatial.jp/ckan/dataset/plateau. 3

[8] Swiss Federal Office of Topography. swissBUILDINGS3D, 2025. https://www.swisstopo.admin.ch/en/landscape-model-swissbuildings3d-2-0. 3

[9] OpenSceneGraph Project. OpenSceneGraph, 1999. http://www.openscenegraph.com/. 3

[10] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 4

[11] Vojtech Panek, Zuzana Kukelova, and Torsten Sattler. Visual localization using imperfect 3D models from the internet. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13175–13186, 2023. 4

[12] Microsoft Research. Microsoft Research, 2025. https://www.microsoft.com/en-us/research/. 2

[13] Gabriele Trivigno, Carlo Masone, Barbara Caputo, and Torsten Sattler. The unreasonable effectiveness of pre-trained features for camera pose refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12786–12798, 2024. 4

[14] Rouwan Wu, Xiaoya Cheng, Juelin Zhu, Yuxiang Liu, Maojun Zhang, and Shen Yan. UAVD4L: A large-scale dataset for UAV 6-DoF localization. In *Proceedings of the International Conference on 3D Vision*, pages 1574–1583, 2024. 4

[15] Wuhan Tianjihang Information Technology Co.,ltd. DP Modeler, 2014. https://www.whulabs.com/DPModeler/index.aspx. 8

[16] Amir R Zamir, Tilman Wekel, Pulkit Agrawal, Colin Wei, Jitendra Malik, and Silvio Savarese. Generic 3D representation via pose estimation and matching. In *Proceedings of the European Conference on Computer Vision*, pages 535–553, 2016. 3

[17] Juelin Zhu, Shen Yan, Long Wang, Shengyue Zhang, Yu Liu, and Maojun Zhang. LoD-Loc: Aerial visual localization using LoD 3D map with neural wireframe alignment. In *Advances in Neural Information Processing Systems*, pages 119063–119098, 2024. 4, 5, 8

[18] Juelin Zhu, Shuaibang Peng, Long Wang, Hanlin Tan, Yu Liu, Maojun Zhang, and Shen Yan. LoD-Loc v2: Aerial visual localization over low level-of-detail city models using explicit silhouette alignment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 26610–26621, 2025. 3, 5