Supplementary: Regularized Primitive Graph Learning for Unified Vector Mapping

Lei Wang[1], Min Dai[1], Jianan He[1], Jingwei Huang[1]
[1]Riemann Lab, Huawei Technologies.
{leiwang4, daimin11, hejianan2, huangjingwei6}@huawei.com

1. Implementation Details

1.1. Additional losses

Segmentation loss $\mathcal{L}_{seg}, \mathcal{L}_{kp}$. L_{seg} is a linear combination of cross-entropy loss and Lovász-softmax loss [1] applied to semantic segmentation map Y_{seg}. Lovász-softmax loss is similar to Dice loss [6] that penalizes the structure of segmentation maps but achieves more stable training. L_{kp} is a cross-entropy loss applied to key point segmentation map Y_{kp}.

Primitive deformation loss $\mathcal{L}_{off}$. L_{off} is a bi-projection loss [3] that penalizes the deviation of the deformed primitives V' from the ground truth primitives $\overline{V}$. It first matches the vertices in ground truth to their nearest predictions, then matches the rest of the predicted vertices to their nearest projections in ground truth shape. The mean *L2* distance between matches is reported.

1.2. Network structure

To be comparable with existing studies, we use Res-U-Net101 [5] as the backbone for all our experiments. All segmentation heads share the same structure of 2×(Conv1x1-BN-ReLU)-Conv1x1. Following the setting in [2], we use 6 MHA layers with 256-dimensional internal representations in primitive learning structure (PLS).

1.3. Training and testing

We decrease learning rate by 10 times when training loss plateaus. Training stops when validation score plateaus after learning rate decayed twice. Weight decay is disabled for both PLS networks. The data augmentation techniques used include Mixup [7], random rotation between [-30°, 30°], flipping, color jittering, and image re-scaling to [0.85, 1.3] of the original image size. During training, we sample a maximum number of 150 primitives, and randomly drop 10% of the points to increase the diversity of point densities. During inference, we sample up to 300 primitives per image. No test-time augmentation is used.

1.4. Building mapping

Hyper-parameters. We use the following loss configuration for building mapping training:

$$\begin{aligned}\mathcal{L} =&\mathcal{L}_{seg} + \mathcal{L}_{kp} + \mathcal{L}_{dir} + 10^{-1}\mathcal{L}_{off} + \\ &\mathcal{L}_{ret} + 10^{-3}\mathcal{L}_{reg} + 10^{-2}\mathcal{L}_{aux}\end{aligned} \tag{1}$$

, in which,

$$\mathcal{L}_{seg} = 0.8\mathcal{L}_{ce} + 0.2\mathcal{L}_{lovasz}. \tag{2}$$

Since the line segment topology is recovered for building mapping, predictor heads in both PLS networks use 3×3 convolution kernel to better aggregate local information between neighboring building line segments.

Post-processing. At inference time, we rotate predicted line segments V'' to their network predicted directions D' around their centers. With topology recovered by projecting line segments to traced contours from segmentation masks, we re-connect the end points of rotated line segments to form a closed polygon. Then, we simplify each polygon by iteratively removing short line segments and merging parallel neighboring line segments. More specifically, line segments with length less than threshold t_{len} are removed. For each removed line segment, its two neighboring line segments are merged if they are parallel (angle $< t_a$), otherwise they are extended to intersection. Due to the accurate direction estimation in D', mapping performance is not sensitive to the selection of t_a. We use $t_{len} = 2.0$ meters and $t_a = 10°$ for all our experiments.

1.5. Road network mapping.

Hyper-parameters. The different types of key points are sampled with the following priority order: junctions > overlays > ends points > interpolated

points > points sampled from segmentation mask. We use the following training loss configuration for road mapping:

$$\mathcal{L} = \mathcal{L}_{seg} + \mathcal{L}_{kp} + \mathcal{L}_{dir} + 10^{-1}\mathcal{L}_{off} + \mathcal{L}_{ret} + 10^{-2}\mathcal{L}_{aux} \quad (3)$$

, in which,

$$\mathcal{L}_{seg} = 0.9\mathcal{L}_{ce} + 0.1\mathcal{L}_{lovasz}. \quad (4)$$

Ground truth generation. We generate key point segmentation ground truth mask by buffering each point to a circle of 5 pixel wide and rasterize these buffered regions. To reduce the ambiguity of connectivity at overlaying road, we expand each overlaying road point to multiple points by adding the intersections of a small circle centered at this point and the road network to ground truth key points. Then, we remove the overlaying road point. We dynamically compute the ground truth relationship between point primitives by matching predicted points to the ground truth road network. We adapt the matching method used in [3] for this purpose. More specifically, we first match each point in ground truth to their nearest point in predicted points. Then, the rest of the predicted points are matched to their nearest projection on road lines. With this matching strategy, the closest point to a road junction will be pulled to that junction, and points near overlaying road will be pushed away from the overlaying point. Therefore, relationship classification can be less ambiguous in these scenarios. To generate ground truth connectivity relationship between point primitives, two predicted points have positive connectivity ground truth if there is no other matched point between their matched points in road line segments.

1.6. Environment

Our implementation is based on Pytorch [4]. All experiments are conducted on a work station equipped with 1 Intel(R) Xeon(R) Gold 6278C CPU @ 2.60GHz and 4 NVIDIA GeForce RTX 3090 GPUs.

2. Parameter tuning and sensitivity

GraphMapper requires little parameter tuning for shape post-processing. In our building shape post-processing, a shorted line length term is used to control the output simplicity. This term is not data-dependent but task-dependent, e.g., it should be set according to the accuracy requirement of a mapping task. For road extraction, no post-processing is needed. GraphMapper is less sensitive to point sampling density compared to Sat2Graph during our test. A possible reason is that GraphMapper is trained to work with varying point densities.

We test the sensitivity of GraphMapper to the connectivity classification threshold for road mapping. As shown in Fig. 1, the sensitivity to connectivity classification threshold is reduced when using embedding space sorting instead of directly classifying connectivity through thresholding.

Sensitivity to connectivity threshold. Fig. 1.

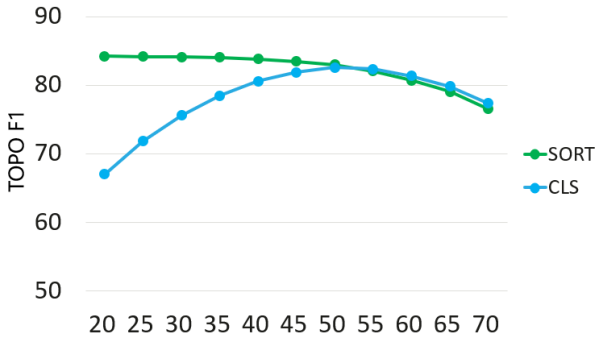

Figure 1: Threshold sensitivity for embedding space sorting (SORT) and connectivity classification (CLS).

3. Additional results

3.1. Road

Additional results. More results from City-Scale dataset is shown in Fig. 2. GraphMapper generates reasonable road networks at different scenarios, such as junctions, narrow overlaying rods, and parallel roads.

Road failures. We show some typical failure cases for road mapping in Fig. 3. In column 1-2, some long parallel lanes on highways are missed. In column 3, connections at wide overlaying roads are missed. Both are caused by miss-detected key points, which are removed when applying NMS to sampled key points for dense road scenarios. Looking for improved methods to sample key points is an interesting topic for our future studies. Some roads are not reconstructed due to incorrect segmentation of roads under trees or shadows (Fig. 3 column 4). When roads are not detected in segmentation maps, no points are sampled on these roads for reconstruction. Additionally, we can see that road segmentation is often broken at road junctions, but can be corrected by reconstruction.

3.2. Building

Results of different stages. We show initial polygons, refined polygons and final reconstructed polygons for building in Fig. 4. GraphMapper shows to reconstruct irregular buildings well. Small blocked roof areas

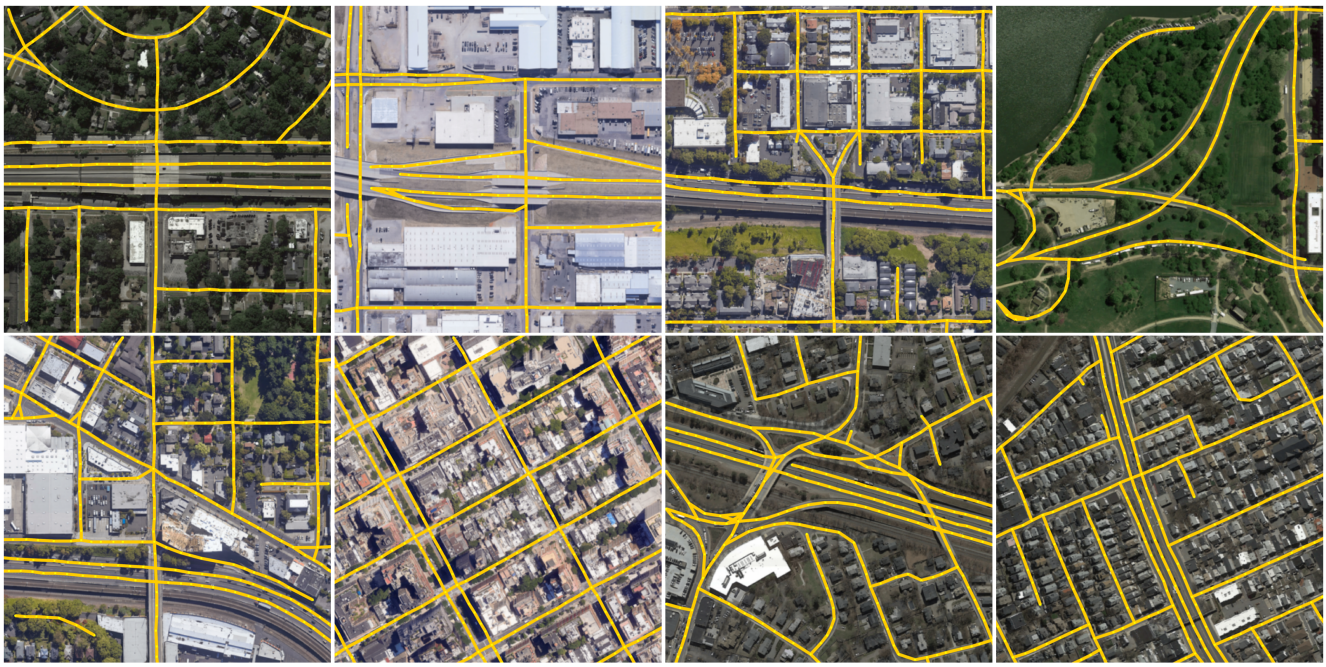

Figure 2: Additional road results from City-Scale dataset.

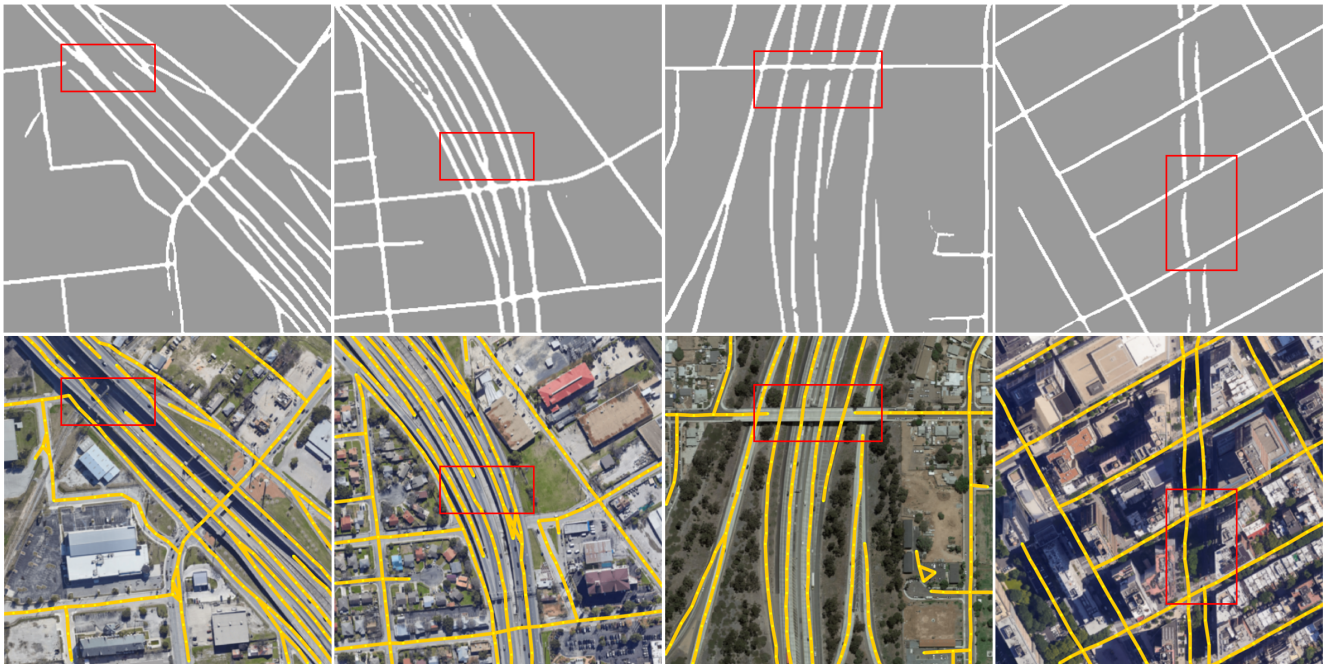

Figure 3: Road failure cases on City-Scale dataset. Top row: segmentation results, bottom row: reconstructed roads. Missing road caused by: missing segmentation and key points (column 1), missing key points (column 2-3), and missing road segmentation under shadows and trees (column 4).

can be properly inferred as shown in the first column. Some details that are not well captured in segmentation maps are recovered at later stages (regions pointed by the yellow arrows).

Failure cases. We show some typical failure cases in Fig. 5. Large segmentation errors are the most important cause of modeling failure. We also see a few failure cases of incorrect direction estimations (Fig. 5

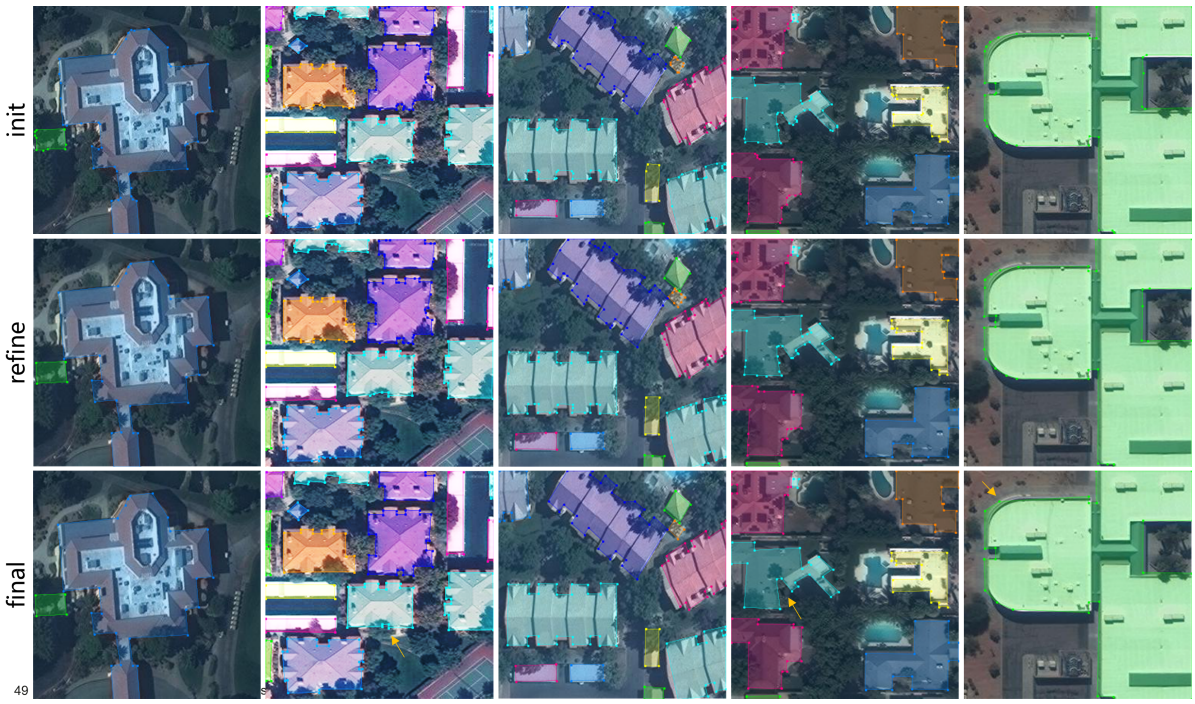

Figure 4: Building reconstruction results of different stages. Irregular buildings can be correctly modeled. Yellow arrows point to properly recovered building details.

last column), which might be caused by the lack of very long edges in our training dataset.

References

[1] Maxim Berman, Amal Rannen Triki, and Matthew B Blaschko. The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4413–4421, 2018. 1

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 213–229, Cham, 2020. Springer International Publishing. 1

[3] Qi Chen, Lei Wang, Steven L. Waslander, and Xiuguo Liu. An end-to-end shape modeling framework for vectorized building outline generation from aerial images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 170:114–126, 2020. 1, 2

[4] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 2

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 1

[6] Carole H. Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M. Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In M. Jorge Cardoso, Tal Arbel, Gustavo Carneiro, Tanveer Syeda-Mahmood, João Manuel R.S. Tavares, Mehdi Moradi, Andrew Bradley, Hayit Greenspan, João Paulo Papa, Anant Madabhushi, Jacinto C. Nascimento, Jaime S. Cardoso, Vasileios Belagiannis, and Zhi Lu, editors, *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 240–248, Cham, 2017. Springer International Publishing. 1

[7] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 1

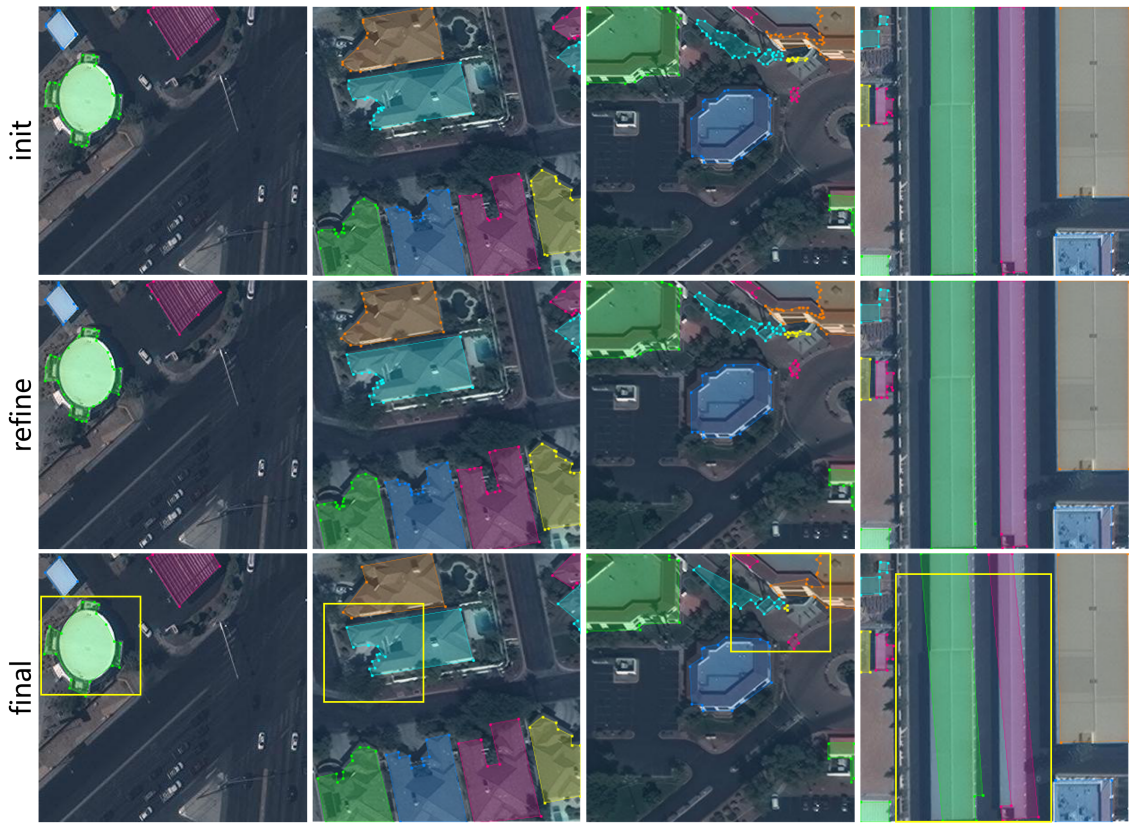

Figure 5: Building failures. Most building failures are caused by segmentation errors (column 1-3). The last column has incorrect direction prediction.