Supplementary document for *Topo-Boundary*

May 20, 2021

Zhenhua Xu, Yuxiang Sun, Ming Liu

1 Introduction

In this supplementary document, we provide more details of the data structure and evaluation metrics.

2 Evaluation metrics

In our benchmark dataset, the evaluation metrics are three relaxed pixel-level metrics (i.e., Precision, Recall and F1-score), the naive connectivity metric used in [1] and [2], APLS (Average Path Length Similarity) [3] which is widely used in road-network evaluation, and our proposed ECM (Entropy-based Connectivity Metric). In this section, we mainly discuss the topology metrics and compare them with more examples.

2.1 APLS

APLS is based on finding the shortest paths of randomly sampled vertex pairs, which shares a very similar idea with TLTS [4]. First, sample N vertex pairs (a, b) in the ground-truth road boundaries, and make sure that vertex a and b belong to the same road-boundary instance (a can reach b). Find the length of the shortest path between a and b as l. Then, find the corresponding vertex pairs (a', b') in the predicted road-boundaries (by minimum Euclidean distance). Find the length of the shortest path between a' and b' as l'. Finally, calculated APLS by

$$APLS = 1 - \frac{1}{N}\sum^{N} min(1, \frac{|l - l'|}{l}) \tag{1}$$

If there is no path between a' and b' or either a' or b' is too far from the ground-truth road boundary, APLS of this pair is 0 (largest punishment). Larger APLS indicates better topology correctness. It is widely used to evaluate the topology correctness in many past works, especially in road-network detection tasks.

However, since the vertex pairs are randomly selected, the results may not be stable, and sometimes it gives very different scores. For better stability, more pairs should be sampled but it severely degrades the computation efficiency since for each vertex pair, two rounds of Dijkstra algorithm should be conducted. APLS is more appropriate to evaluate a complicated connected graph, such as the road network. For road boundaries that are simple polylines without branches, there are better options.

2.2 Naive connectivity metric

This is a very simple metric to evaluate the connectivity of the obtained graph. In [2], this metric is defined in this way: for each ground-truth boundary, let M be the number of its assigned predicted polylines, and

$$connectivity = \frac{1(M > 0)}{M} \tag{2}$$

It penalizes the assignment of multiple small predicted segments to a ground-truth road boundary. This metric is very simple and efficient, but it only considers the number of assigned predicted segments, which makes it very sensitive to noises and unaware of many properties of the obtained graph, such as the position and the length of the disconnection.

2.3 ECM

Following the core idea of the naive connectivity metric, we add more parameters and entropy into it for enhancement.

$$ECM = \sum_{i=1}^{N} \alpha_i e^{\sum_{j=1}^{M_i} p_j log(p_j)} \tag{3}$$

Each ground-truth boundary instance is assigned with M_i predicted instances. α_i is the ratio of the length of predicted instances projected to the ground-truth instance to the length of the current ground-truth instance, and it measures the completion of the prediction. The exponential term measures the entropy of the predicted instances. Larger entropy means the uncertainty to find a dominant predicted instance is higher so that the connectivity is poor and ECM becomes lower.

ECM is based on the naive connectivity metric, but it can take more factors into consideration, which makes this metric more comprehensive and reasonable to measure the quality of the obtained road-boundary graph.

Compare with APLS, ECM is more robust and efficient.

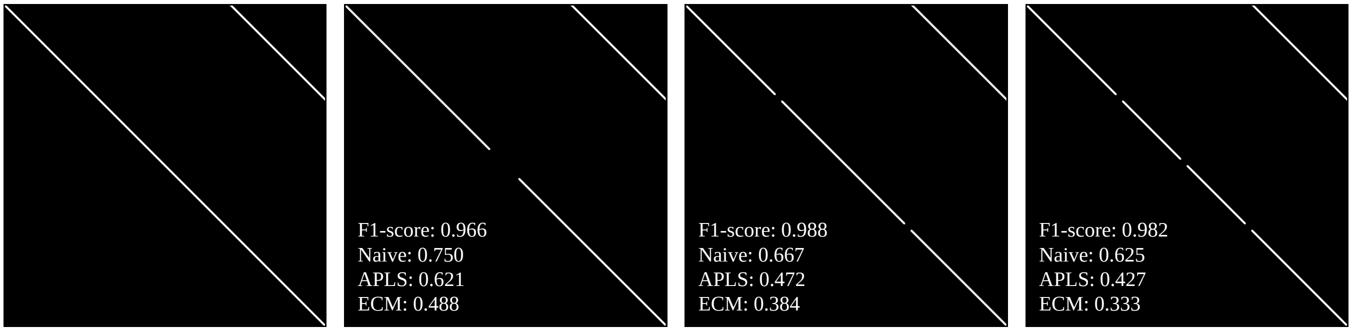
2.4 Comparison

In past works, like [5] and [6] , the topology correctness is measured by path-based metrics such as APLS and TLTS, or junction-based metrics. But since road boundaries usually do not have junctions (intersection vertices) as the road network, junction-based metrics are not applicable to our task. Path-based metrics are widely used in road-network detection works, and they share the same core idea. In this work, we only consider APLS.

We provide some visualizations in Fig. 1 - Fig. 4 to compare the following metrics: F1-score, naive connectivity, APLS and our proposed ECM. We hope the evaluation metric could (1) punish incorrect disconnections (Fig. 1) (2) give shorter ground-truth instances lower weights (Fig. 2), (3) give longer disconnections higher punishment (Fig. 3) and (4) give longer predicted boundaries higher score (Fig. 4). From the visualization, we find that ECM is obviously better than the naive metric while it has a competitive performance compared with APLS.

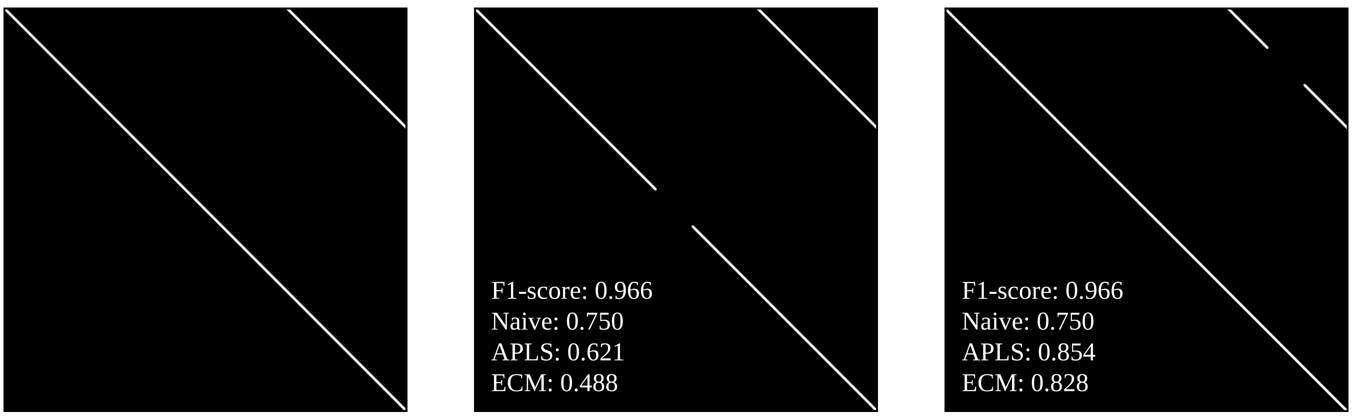
Even though ECM and APLS both have good ability for connectivity measurement, ECM has much better efficiency and stability than APLS. The inefficiency and randomness are the main reason that the authors of [2] propose the naive connectivity. Since APLS needs to randomly select vertex pairs, if the sampling number is not huge enough, the obtain evaluation score is not stable due to randomness; but if the number is too huge, the efficiency is severely degraded since the time complexity of the shortest path algorithm greatly expands. Therefore, trade-off between efficiency and stability should be made.

Moreover, APLS for road-boundary evaluation requires additional post-processing steps, i.e., simplify the skeleton of road boundaries into a graph with fewer vertices. The original skeleton can be treated as a graph whose vertices are all the foreground pixels, but this graph has too many vertices which take huge time to calculate APLS. So we need to simplify it for better efficiency. In our implementation, we uniquely sample vertices to maintain the effectiveness of APLS. The process is shown in Fig. 5.



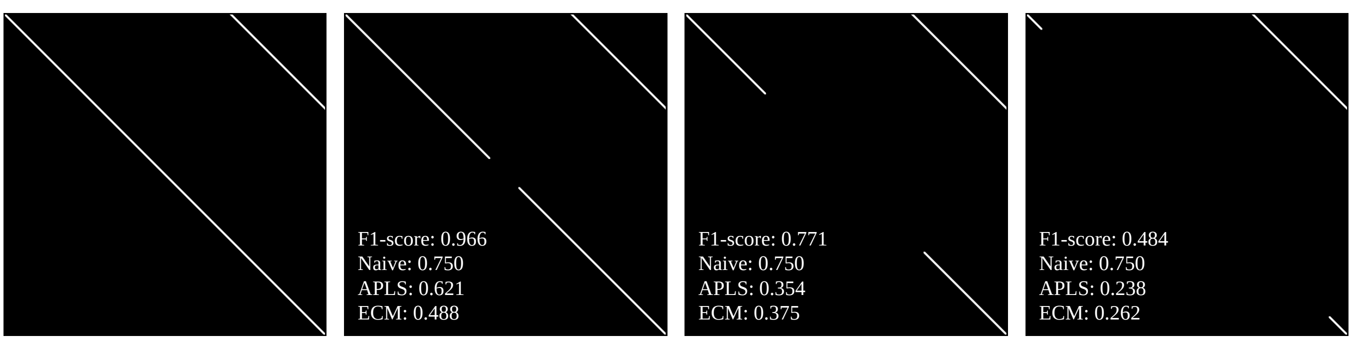
(a) Ground truth (b) Prediction 1 (c) Prediction 2 (d) Prediction 3

Figure 1: Metrics should punish disconnections. The more disconnections in a prediction, the worse the metric score should be. Compared with the naive connectivity metric and APLS, ECM is more sensitive to disconnections. Both APLS and ECM have good discriminative power.



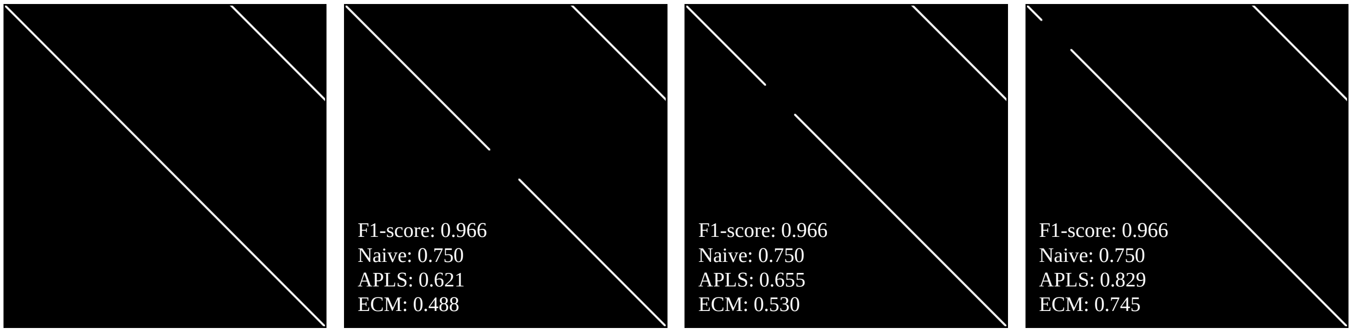
(a) Ground truth (b) Prediction 1 (c) Prediction 2

Figure 2: Metrics should give longer instances larger weights. Prediction 1 should have a lower score than prediction 2 since the disconnection happens in the longer instance. Both APLS and ECM perform well. But the naive connectivity metric and pixel-level F1-score fail.



(a) Ground truth (b) Prediction 1 (c) Prediction 2 (d) Prediction 3

Figure 3: Metrics should give longer disconnections larger punishment. Both ECM and APLS work well.



(a) Ground truth (b) Prediction 1 (c) Prediction 2 (d) Prediction 3

Figure 4: Metrics should encourage longer predicted instances since longer instances represent better completion. And as prediction 3 shows, there might be a lot of very short predicted segments near the endpoints due to noise, thus they should not receive huge punishment. Both APLS and ECM work well.

All in all, compared with APLS, our proposed ECM is robust and effective. Besides, ECM shows much better efficiency than APLS and does not require additional post-processing steps. Compare with the naive connectivity metric, ECM can give a more comprehensive and reasonable measurement of the connectivity.

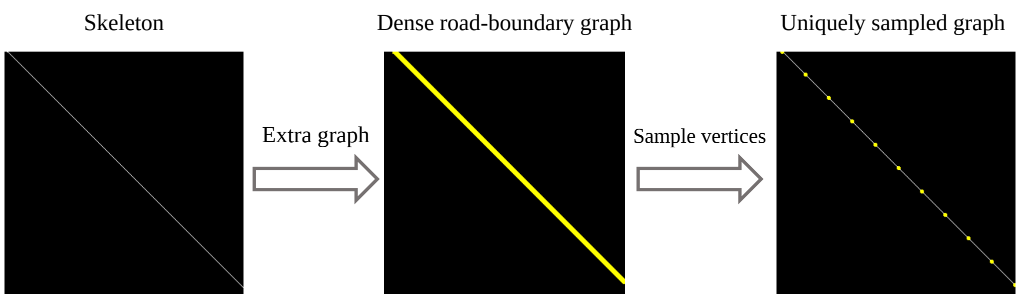


Figure 5: The post-processing step to convert the road-boundary skeleton to a graph. We first treat every foreground pixel as a vertex and obtained the dense graph (dense means every pixel is a vertex). The dense graph cannot be used for APLS since it requires too much time to calculate the shortest path. So we only sample some vertices for better efficiency. Unique sampling guarantees APLS punishes disconnections along the boundary equally.

In our implementation, after getting the predicted skeleton, we need to first convert it to a simplified graph following the above processing steps, and then run APLS. It takes relatively a long time to complete everything (more than 1 hour for all 3,289 testing images).

3 Data structure

3.1 Aerial image patch

The way we create the aerial image patch is shown in Fig. 6. Each image is 1000×1000-sized, and has 4 channels. The first 3 channels are R,G,B. All aerial images are in *tiff* format.

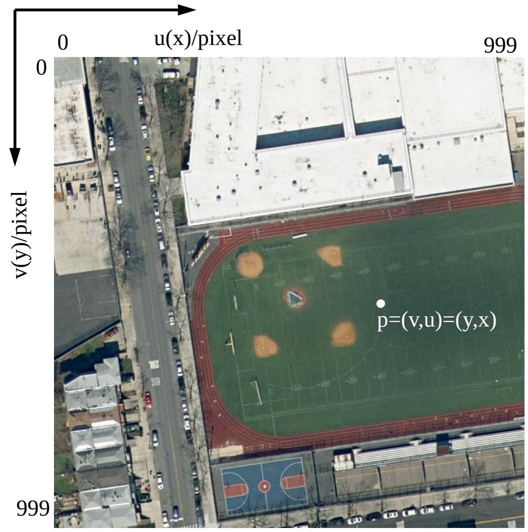


Figure 6: Aerial image patch. The original point is the most top left pixel. The axis of rows is v or y, and the axis of columns is u or x. All the points in our dataset is saved in the form of p=(v,u) or p=(y,x).

3.2 Binary map

The binary map labels the ground-truth road boundaries as foreground pixels. The coordinate of this map is the same as that of the aerial image patch. The value of foreground pixels is 255. This map has 3 equal channels. The visualization is shown in Fig. 7.

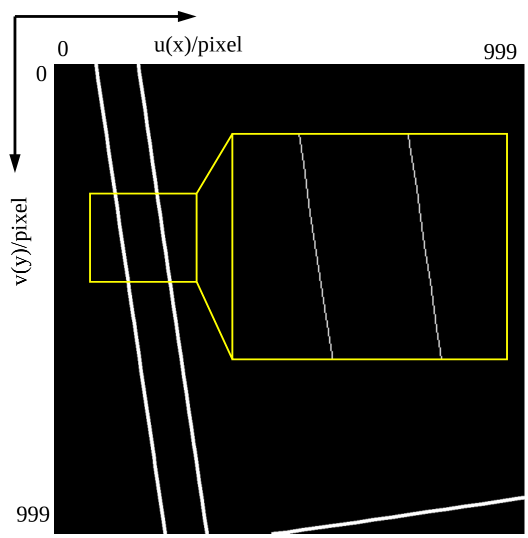


Figure 7: Binary map. For better visualization, the boundaries are usually widened, while they are actually of one-pixel width.

3.3 Instance map

Similar to the binary map. But different instance has different grey value, starting from 1 to N (number of instances). The value of foreground pixels are their corresponding instance index. This map has 3 equal channels. Please see Fig. 8.

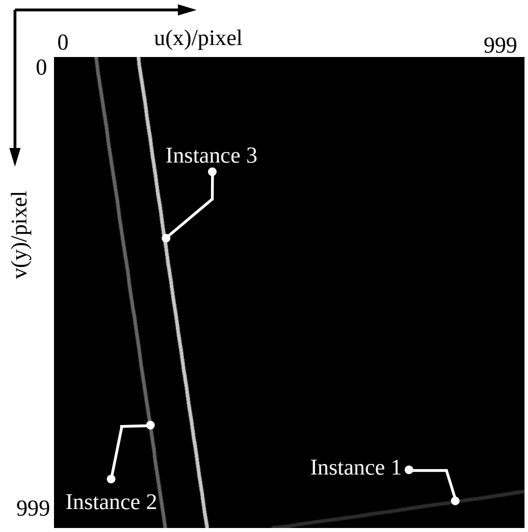


Figure 8: Instance map. For better visualization, the boundaries are usually widened, while they are actually of one-pixel width. The grey value of foreground pixels is also enlarged for better visualization.

3.4 Endpoint map

In this map, the foreground is the endpoints of each ground-truth road-boundary instance. For better supervision, each point is multiplied with a Gaussian kernel. The maximum grey value of this map is 255. This map has 3 equal channels. Please see Fig. 9.

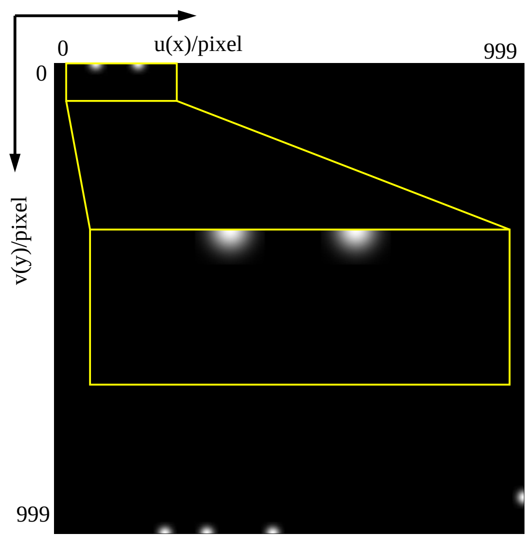


Figure 9: Endpoint map. We multiple each endpoint by the Gaussian function for better supervision ability.

3.5 Inverse distance map

In this map, the value of each pixel is the reciprocal of its shortest distance to the road boundary. The maximum grey value of this map is 255. This map has 3 equal channels. Please see Fig. 12.

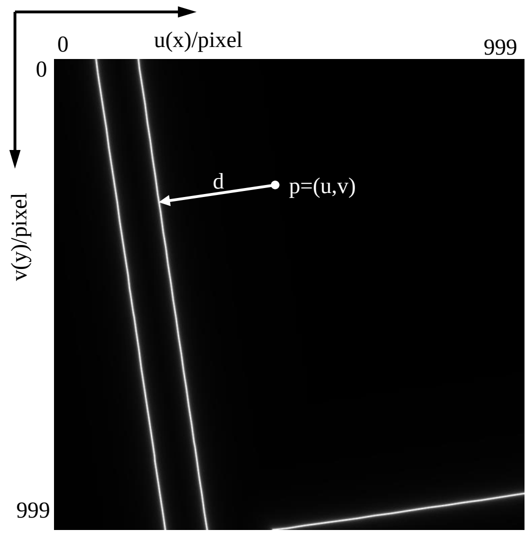


Figure 10: Inverse distance map. Each pixel p first find its shortest distance to the road boundary, and the distance is recorded as d. Then we have $d = max(d, 1)$ to remove to small distance values. Finally take the reciprocal and multiply 255.

3.6 Direction map

This map is similar to the inverse distance map. At each pixel $p = (y, x)$, this map records a unit vector pointing to the shortest road boundaries. This map has 2 channels. Please see Fig. 12.

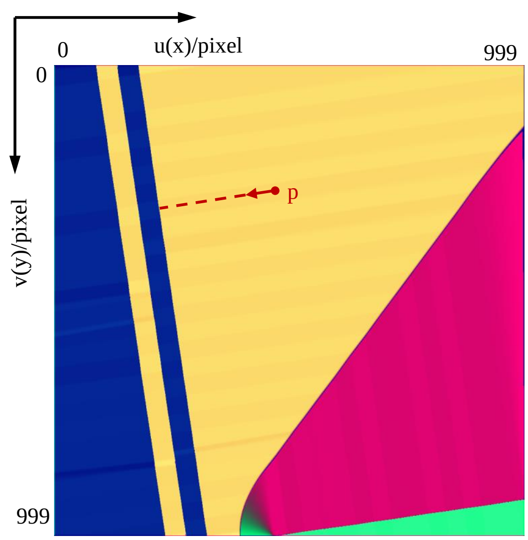


Figure 11: Direction map visualization. At each pixel $p = (y, x)$, this map records an unit vector (v_y, v_x). For visualization, we have v_y as R, v_x as G and $\frac{v_x+v_y}{2}$ as B. Then multiply 255 to obtain the visualization. But this map is actually of 2 channels.

3.7 Orientation map

The foreground pixels of this map are the same as that of the binary map. While the pixel value $|p|$ is the angle θ between the road boundary and the v axis. To simplify the learning problem, we convert the angle to a 64-class classification label ($|p| = \lfloor \frac{32\theta}{\pi} \rfloor$). The value of this map ranges from 1 to 64. This map has 3 equal channels. Please see Fig. 12.

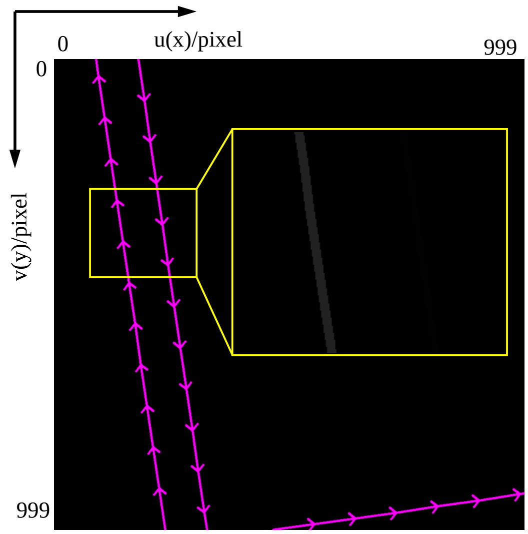


Figure 12: Orientation map visualization. The pink arrows show the direction of the road boundary, while the actual value of this map is shown in the zoom-in region. Within the zoom-in region The pixel value of the left instance is 33 and the right is 2.

3.8 Annotation sequence

Annotation sequence records the graph information of each ground-truth road-boundary instance. This label is recorded in a JSON file. The JSON structure is shown in Fig. 13.

[
{
# instance 1
"init_vertex" :[y_1^1, x_1^1],
"end_vertex" :[$y_{N^1}^1, x_{N^1}^1$],
"seq" :[[y_1^1, x_1^1],
[y_2^1, x_2^1],
...
[$y_{N^1}^1, x_{N^1}^1$]]
},
{
# instance 2
"init_vertex" :[y_1^2, x_1^2],
"end_vertex" :[$y_{N^2}^2, x_{N^2}^2$],
"seq" :[[y_1^2, x_1^2],
[y_2^2, x_2^2],
...
[$y_{N^2}^2, x_{N^2}^2$]]
},
......
]

Figure 13: Annotation sequence. Each instance contains an initial vertex coordinate, an end vertex coordinate and a sequence recording all the vertices of the current boundary instance. Each vertex is adjacent to vertices before and after it.

3.9 Dense sequence

Same as the annotation sequence, this sequence is recorded in a JSON file and it has the same JSON structure as the annotation sequence. But the "seq" in this sequence is densified or saying rasterized, and the process is visualized in Fig. 14.

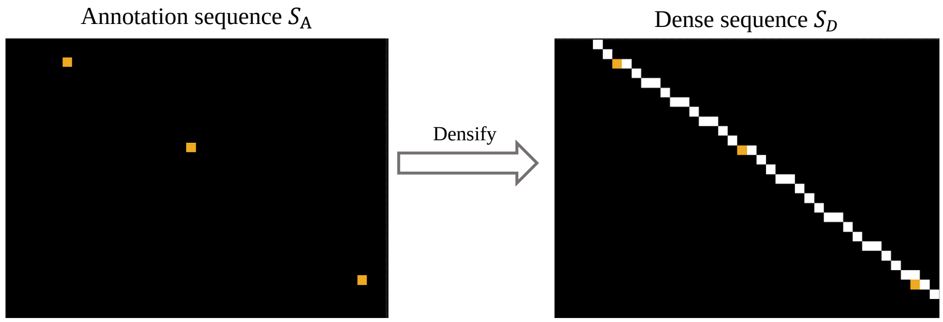


Figure 14: Densification/Rasterization of the annotation sequence. The orange pixels represent vertices in the annotation sequence S_A. They are interpolated to realize every two adjacent vertices eight-neighboring to each other. The interpolation result is the dense sequence S_D. S_A can be regarded as a subset of S_D, which contains only the key vertices.

References

[1] N. Homayounfar, W.-C. Ma, J. Liang, X. Wu, J. Fan, and R. Urtasun, "Dagmapper: Learning to map by discovering lane topology," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2911–2920.

[2] J. Liang, N. Homayounfar, W.-C. Ma, S. Wang, and R. Urtasun, "Convolutional recurrent network for road boundary extraction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9512–9521.

[3] A. Van Etten, D. Lindenbaum, and T. M. Bacastow, "Spacenet: A remote sensing dataset and challenge series," *arXiv preprint arXiv:1807.01232*, 2018.

[4] J. D. Wegner, J. A. Montoya-Zegarra, and K. Schindler, "A higher-order crf model for road network extraction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1698–1705.

[5] F. Bastani, S. He, S. Abbar, M. Alizadeh, H. Balakrishnan, S. Chawla, S. Madden, and D. DeWitt, "Roadtracer: Automatic extraction of road networks from aerial images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4720–4728.

[6] Y.-Q. Tan, S.-H. Gao, X.-Y. Li, M.-M. Cheng, and B. Ren, "Vecroad: Point-based iterative graph exploration for road graphs extraction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8910–8918.