Topo-boundary: A Benchmark Dataset on Topological Road-boundary Detection Using Aerial Images for Autonomous Driving

Zhenhua Xu, Yuxiang Sun, *Member, IEEE*, and Ming Liu, *Senior Member, IEEE*

***Abstract*—Road-boundary detection is important for autonomous driving. For example, it can be used to constrain vehicles running on road areas, which ensures driving safety. Compared with on-line road-boundary detection using on-vehicle cameras/Lidars, off-line detection using aerial images could alleviate the severe occlusion issue. Moreover, the off-line detection results can be directly used to annotate high-definition (HD) maps. In recent years, deep-learning technologies have been used in off-line detection. But there is still lacking a publicly available dataset for this task, which hinders the research progress in this area. So in this paper, we propose a new benchmark dataset, named *Topo-boundary*, for off-line topological road-boundary detection. The dataset contains 21,556 1000×1000-sized 4-channel aerial images. Each image is provided with 8 training labels for different sub-tasks. We also design a new entropy-based metric for connectivity evaluation, which could better handle noises or outliers. We implement and evaluate 3 segmentation-based baselines and 5 graph-based baselines using the dataset. We also propose a new imitation-learning-based baseline which is enhanced from our previous work. The superiority of our enhancement is demonstrated from the comparison. The dataset and our-implemented codes for the baselines are available at `https://sites.google.com/view/topo-boundary`.**

***Index Terms*—Road-boundary Detection, Imitation Learning, Large-scale Dataset, Autonomous Driving.**

I. Introduction

ROAD boundary refers to the dividing line between the road area and off-road area. It can be used to constrain self-driving cars running on road areas, which is important to the safety of autonomous driving. Currently, most existing works rely on vehicle-mounted sensors (e.g., cameras or Lidars) [1]–[4] for on-line road-boundary detection. However, on-line detection could be severely degraded by occlusions, which is very common in real road environments. Moreover, on-line detection could be restricted by limited computing resources on vehicles, especially for deep-learning models that require GPU. To relieve the aforementioned issues, some works resort to detecting road boundary (or its analogies, such as lane or curb) off-line using bird-eye-view (BEV) point-cloud maps or aerial images. As the results are presented in BEV images, they can also be directly utilized for annotating high-definition (HD) maps. So we detect road boundaries off-line in this paper. Moreover, our detection results are presented in the form of topological graphs (i.e., vertices and edges). This is important because besides pixel-level annotations, the topological graphs can also identify road-boundary instances and provide the spatial connection information of the boundaries.

The authors are with The Hong Kong University of Science and Technology. Email: `zxubg@cse.ust.hk`, {`eeyxsun,eelium`}`@ust.hk`. Ming Liu is the corresponding author.

The published literature working on topological road-boundary detection is very limited. Most of existing works focus on the analogy problem: line-shaped object detection, such as road lane, curb and network detection [5]–[11]. They can be generally divided into two categories: segmentation-based solutions and graph-based solutions. A typical pipeline of the former is first using semantic segmentation to get rough detection results, and then performing heuristic post-processing algorithms on the segmentation maps to refine the results. While the latter directly finds the graph structure for the line-shaped objects through iterative graph growing. With the great advancement of artificial intelligence, deep-learning technologies are adopted by the existing methods, and great superiority over traditional algorithms is demonstrated. However, deep-learning methods require large-scale datasets to train a model. To the best of our knowledge, there is no publicly available dataset for road boundary detection in BEV images, which hinders the research process in this area. So in this paper, we build a large-scale benchmark dataset, named *Topo-boundary*, for road boundary detection. Our dataset is derived from the *NYC Planimetric Database* [12].

The raw geographic data (a large whole map, see Fig. 1) in the *NYC Planimetric Database* covers the whole New York City, including 2,147 5000×5000-sized 4-channel (i.e., red, green, blue and an infrared channel) aerial image tiles (i.e., small squared aerial image constituting the large whole map) and a set of polylines as the road-boundary ground-truth label. In *Topo-boundary*, We convert the polyline label to graph (i.e., vertices and edges) and split every tile into 25 smaller 1000×1000-sized patches. After removing patches with inappropriate annotations, such as patches without road boundaries, 21,556 patches remain. Each patch consists of a 4-channel image and 8 labels for different sub-tasks, such as binary semantic segmentation, orientation learning [6], etc. The dataset can be used for both segmentation-based solutions and graph-based solutions.

To facilitate future research on road-boundary detection, we implement and evaluate multiple baselines, including 3 segmentation-based baselines, 5 graph-based baselines. We also design and compare a new imitation-learning-based baseline which is enhanced from our previous work *iCurb* [11]. For all the baselines, the input is a 4-channel aerial image, the output is the graph representing road boundaries. Our implemented codes for these baselines are available with our dataset. As line-shaped objects are usually long, thin and irregular, only using pixel-level metrics are not sufficient to evaluate the performance of a method. Moreover, topological correctness (i.e., the obtained graph should not have disconnections) is also important. Since in most cases road boundaries are simple polylines without

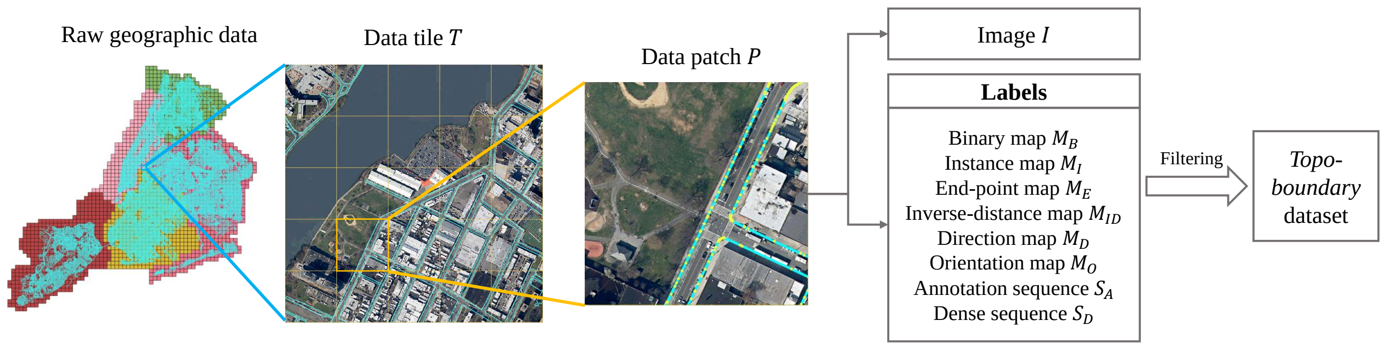


Fig. 1: The pipeline to create our dataset from the *NYC Planimetric Database*. The raw geographic data contains 2,147 data tiles. The different colors represent the 5 boroughs and water areas in New York City. Each tile (T) is with 5000×5000 size. We split each tile into 25 smaller patches ($T = \{P_i\}_{i=1}^{25}$). Each patch (P) contains a 4-channel 1000×1000-sized aerial image I and ground-truth road boundaries G_{gt}. The G_{gt} in the figure is annotated by polylines, of which edges and vertices are denoted by cyan lines and yellow points, respectively. Based on G_{gt}, we generate 8 types of ground-truth labels for different deep-learning tasks. The filtering step is to remove patches based on pre-defined rules. After this step, 21,556 patches finally remain in our *Topo-boundary*. For better visualization, only RGB channels of the patch image are visualized. Moreover, the width of the ground-truth polylines is increased (the actual width is one pixel). This figure is best viewed in color.

branches, inspired by [13] we propose a new entropy-based connectivity metric (ECM) to evaluate the connectivity of the obtained graph for road boundaries.

We summarize our contributions here:

1) We release the *Topo-boundary* benchmark dataset for topological road-boundary detection using aerial images. To the best of our knowledge, this is the first publicly available dataset for this task.
2) We propose new evaluation metrics for the task, including relaxed pixel-level metrics and a new entropy-based connectivity metric.
3) We evaluate 9 baseline models on the dataset, which could facilitate the comparison for future methods.
4) We open-source the codes we implemented for the baselines. The codes can be modified with a little effort for other line-shaped object detection.

II. Related works

A. Segmentation-based line-shaped object detection

There are very few segmentation-based works directly detecting road boundaries topologically. So we review several line-shaped object detection works [5], [6], [14]. Volodymyr *et al.* [14] proposed the first deep-learning model with iterative refinement for road-network detection using aerial images. Their solution was further improved by Anil *et al.* [6], in which the authors not only proposed better networks and a training scheme, but also utilized the orientation map to enhance the semantic segmentation results. Different from the above papers using iterative refinement, the solution proposed by Mattyus *et al.* [5] first put forward connection candidates to correct disconnections, and then trained another network to filter candidates. Although segmentation-based solutions are efficient to carry out, they can only produce results at the pixel-level. Moreover, they were often implemented with multi-stage pipelines, so they could not be optimized as a whole, making the graph sensitive to incorrect topology.

B. Graph-based line-shaped object detection

Taken images as input, graph-based methods can directly output the graph representing target objects. Iterative graph growing is the most commonly used technique, which generates vertices along with the line-shaped object starting from an initial vertex. Past works focusing on other line-shaped objects are similar to our task, such as road lane detection [9], [15], road network detection [7], [8], [16] and road curb detection [11]. *RoadTracer* [7] is believed to be the first work in this category. In [7], the authors trained a multi-layer CNN and successfully extracted the graph of very large road networks. Liang *et al.* [13] directly worked on road boundary detection in the BEV point cloud map. They facilitated semantic segmentation by direction map prediction and proposed cSnake for iterative graph growing. To better solve the graph growing task, line-shaped object detection was first analyzed from the perspective of imitation learning in our previous work *iCurb* [11]. *iCurb* presented superiority over other works on road curb detection owing to the training strategy and graph growing policy.

C. Evaluation metrics for topological correctness

In the past works on road network detection, topological correctness was usually measured by shortest-path-based metrics, such as APLS [17] and the sample-based metric proposed in [18]. However, due to the huge difference between the topology of road networks and road boundaries (a road network is a connected graph while road boundaries are scattered polylines without branches), these metrics are not applicable in our evaluation. In [13], a naive metric measuring the connectivity of road boundaries was proposed, but it is too sensitive to noise and may cause incorrect evaluation results. *iCurb* modified the naive metric and addressed the problem to some extent. However, incorrect evaluation still happens. So in this work, we make further revisions to this metric and propose an entropy-based evaluation metric (ECM) for connectivity measurement.

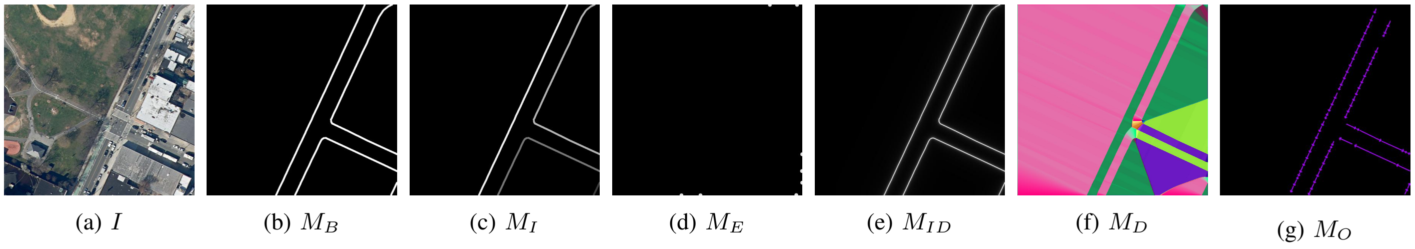


Fig. 2: Visualization of patch label maps. (a) 4-channel patch image I (only RGB is visualized here). (b) Binary map M_B. (c) Instance map M_I. Different road boundary instances are labeled with different gray values. (d) End-point map M_E. (e) Inverse-distance map M_{ID}. (f) Direction map M_D. The red channel is the Sobel derivative of columns, the green channel is the Sobel derivative of rows and the blue channel is the sum of the other two channels. (g) Orientation map M_O. Purple arrows are the schematic demonstration of directional information of the road boundary. For better visualization, the width of the lines is increased (the actual width is one pixel). This figure is best viewed in color. Please zoom in for details.

D. Datasets for line-shaped object detection

To the best of our knowledge, [13] is the only published work on road-boundary detection using BEV images. In [13], a dataset was created, which contained BEV point-cloud images, RGB images and elevation gradient images. But unfortunately, the dataset was not publicly available. In the field of road network detection, the data was obtained from commercial geospatial datasets [17] or by processing raw geographic images collected from public platforms [19]. Although these datasets are publicly available, due to the topological differences between road networks and road boundaries, they could not be used in our task.

III. The Proposed Dataset

To create our *Topo-boundary* dataset, we generate images and ground-truth labels from the raw data collected from the *NYC Planimetric Database* [12]. The WGS84 coordinate system is used in [12], while in *Topo-boundary*, all the coordinates are transformed to the image coordinate system. Fig. 1 shows the processing pipeline to create our *Topo-boundary*. We first split large data tiles into patches, then generate ground-truth labels for each patch, and finally remove inappropriate patches according to predefined rules.

A. Splitting data tiles into patches

The raw data covers the whole area of New York City, including 2,147 data tiles. Each tile (T) is a 5000×5000-sized image with the road-boundary ground-truth label. Each pixel represents 1 foot (around 15.2cm). We split each tile into 25 smaller patches ($T = \{P_i\}_{i=1}^{25}$) to reduce the memory cost during training. Since our task is mainly performed at the patch level, the subscript i is removed from P_i for expression conciseness. Each patch (P) contains a 4-channel (red, green, blue, infrared, the infrared channel is more sensitive to vegetation and soil) 1000×1000-sized aerial image I and ground-truth road boundaries G_{gt}. We use G_{gt} to generate 8 labels for different sub-tasks.

The raw ground-truth label for road boundaries is the `PAVEMENT_EDGE` feature from the *NYC Planimetric Database*. `PAVEMENT_EDGE` is manually annotated in the form of polylines (a kind of graph without branches), of which the vertices and edges are recorded using the WGS84 coordinate system. The polylines cover the edges of the road surfaces, airport runways and alleys. From the whole map `PAVEMENT_EDGE`, we obtain the ground-truth road boundary G_{gt} for each patch P.

Unlike tile images, `PAVEMENT_EDGE` cannot be directly cut and split because: (1) some edges $e = (v_a, v_b)$ may have one vertex v_a in a patch while the other vertex v_b in another patch; (2) some road boundaries may be cut into multiple pieces by different patches. For the former, we replace the vertex outside the patch (i.e., v_b) with the intersection point of e and the patch edge. For the latter, we split the corresponding road boundaries into multiple instances and update `PAVEMENT_EDGE`.

After processing `PAVEMENT_EDGE`, for each patch, the ground-truth road boundary is still a set of polylines $G_{gt} = \{G_{gt}^i\}_{i=1}^N$. Each polyline represents an road-boundary graph instance $G_{gt}^i = (V_i, E_i)$, where V_i and E_i are the set of vertices and edges in G_{gt}^i, respectively. Since edges in a graph can be easily obtained by connecting two adjacent vertices, the edge set E_i is omitted. Note that G_{gt}^i is sparsely distributed (i.e., adjacent vertices are not eight-neighbor to each other in the image coordinate system).

B. Generating ground-truth labels for each patch

For each patch, we generate 8 different labels for different sub-tasks. The labels are: annotation sequence S_A, dense sequence S_D, binary map M_B, instance map M_I, end-point map M_E, inverse-distance map M_{ID}, direction map M_D and orientation map M_O. All the labels are generated from ground-truth road boundaries (G_{gt}) within the current patch by our processing algorithms, and are recorded in the image coordinate system of the current patch.

1) Annotation sequence and dense sequence: As the ground-truth road boundaries are annotated vertex by vertex as chain sequences, the vertices of a road boundary instance G_{gt}^i are ordered starting from an initial vertex to the end vertex. We call the ordered vertices of G_{gt}^i as the annotation sequence S_A. S_A is critical to generate other labels and can be applied to train graph-based solutions. However, graph-based solutions trained by S_A suffer from the *teacher forcing* problem [20], which is caused by the data distribution mismatch between the training and inference period. It is also analyzed in [7]. To address this problem, the training label for iterative graph generation should be generated on-the-fly. Therefore, S_A needs to be densified. For

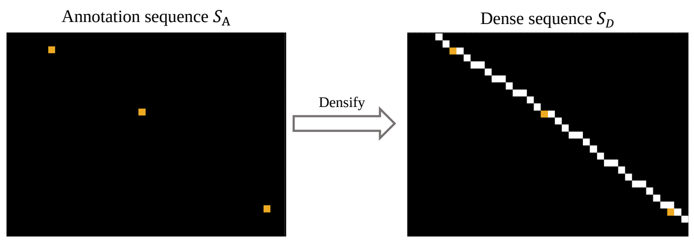


Fig. 3: The sequence densification process for a part of a sample patch. The orange pixels represent vertices in the annotation sequence S_A. They are interpolated to realize every two adjacent vertices eight-neighboring to each other. The interpolation result is the dense sequence S_D. S_A can be regarded as a subset of S_D, which contains only the key vertices. The figure is best viewed in color.

every two adjacent vertices in S_A, interpolation is conducted and then the dense sequence S_D is obtained. In S_D, any two adjacent vertices are eight-neighbor to each other. The densification process is visualized in Fig. 3.

2) Binary map, instance map and end-point map: Binary map M_B is commonly used for semantic segmentation. The dense sequence S_D can be directly used to generate M_B by setting all pixels covered by S_D as 1 (i.e., foreground pixels) while leaving other pixels as 0 (i.e., background pixels). In this way, we get the binary map of a patch and the topological correctness of it is guaranteed. The instance map M_I can be obtained similarly, but pixels covered by different instances are multiplied with instance IDs. M_I could be useful for connection candidates generation [5], instance segmentation and evaluation.

To facilitate segmentation and generate candidate initial vertices for iterative graph generation, endpoint prediction is required in some solutions [11], [13]. For each road boundary instance G^i_{gt} in a patch, there are 2 endpoints. We set all the pixels of M_E whose distance is less than 5 pixels to any endpoint as foreground while leaving others as background.

3) Inverse-distance map and direction map: Inverse-distance map (M_{ID}) prediction is first proposed in [21] to facilitate semantic segmentation. The value of each pixel in this map is the reciprocal of its shortest distance to the ground-truth road boundary. Compared with the binary map M_B, M_{ID} is more informative considering that all the pixels of M_{ID} are encoded with information. The generation of M_{ID} is accelerated by CUDA in our implementation.

To further make use of the spatial information in M_{ID}, the authors in [13] generate the direction map M_D based on M_{ID}. The direction map is a vector field ($M_D \in \mathbb{R}^{2\times H\times W}$) of normal directions to the road boundary. It is calculated by taking the Sobel derivative of M_{ID} followed by a normalization step. Intuitively, M_D records the direction to the closest road boundary of each pixel by unit vectors. The application of M_D and M_{ID} encourages the neural network to focus on road boundaries.

4) Orientation map: Orientation learning [6] makes use of the direction and connection information of road networks, which greatly enhances the performance of semantic segmentation. Thus we provide the orientation map for this task to assist road boundary detection. Due to the bi-direction nature

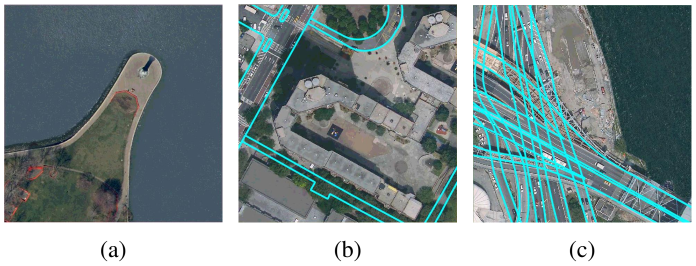


Fig. 4: Sample inappropriate patches. Cyan lines represent the ground-truth polylines for road boundaries. (a) A patch without road boundaries inside. Such a case happens frequently near ocean and suburb areas. (b) A patch of which ground-truth road boundaries have intersection points. This is usually caused by alleys that snapped to the road boundary. Intersection points are not reasonable considering the topological characteristics of the road boundary. (c) A patch with very complicated scenarios. For these patches, few methods can obtain reasonable results, especially for graph-based methods. For better visualization, only RGB channels of the patch image are visualized. Moreover, the width of the ground-truth polylines is increased (the actual width is one pixel).

of road boundaries, we regard them as undirected graphs. For each road boundary instance G^i_{gt}, we randomly select one end vertex as the starting point while the other as the ending point. Then from the starting point, for every two adjacent vertices, we calculate a directional vector and convert it to a radian value r. For all pixels covered by the edge connecting these two adjacent vertices, their values are set to r. Intuitively, the value of each foreground pixel is the radian value of the edge it belongs to. Fig. 2 displays all the label maps for a sample patch.

C. Removing patches according to pre-defined rules

We remove the inappropriate patches according to the following predefined rules:

1) The patch with no road boundary inside. Such a case is quite common, especially in suburb area images;
2) The patch with intersection points. In the raw ground-truth label of the road boundary, some alleys are snapped to it and cause intersections, which is not appropriate. Thus we remove these patches;
3) The patch that has very complex scenarios, such as overlapping interchanges.

Sample inappropriate patches are visualized in Fig. 4. After this step, there are finally 21,556 patches remaining in our *Topo-boundary*. We randomly split these patches into a training set (10,057 patches), a validation set (1,092 patches), a testing set (2,085 patches) and a pretraining set (8,322 patches). The pretraining set is used for multi-stage methods like [5], or pretraining the feature extraction module of graph-based solutions to accelerate the training convergence such as [9]. If the pretraining set is not needed for a solution, samples in this set could be used for training.

IV. Evaluation Metrics

To ensure comprehensive and fair comparison, in this section we introduce our evaluation metrics, including 3 relaxed

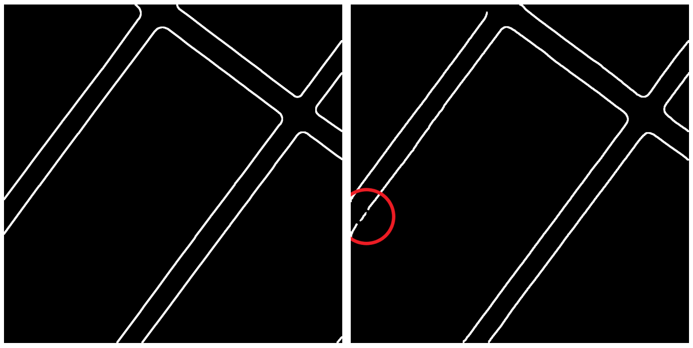

(a) Ground truth (b) Prediction

Fig. 5: Comparison for the connectivity metrics. (a) The ground truth of road boundaries. (b) Predicted road boundaries. The prediction results are generally fine, because there are only two very short disconnections (marked by the red circle) which could not significantly affect the overall prediction. The results of the naive metric, *iCurb* metric and our proposed ECM are 0.722, 0.700 and 0.874, respectively. In many cases like this example, the short disconnections (even one-pixel-length ones caused by noises) could cause serious degradation for the naive and *iCurb* metrics. Compared to these two metrics, we can see that ECM has a larger value. This indicates that our ECM could lower the weights of short segments to make the evaluation results more fair and convincing. Note that for better visualization, the width of the road boundary is increased (the actual width is one pixel).

pixel-level metrics (i.e., Precision, Recall and F1-score) and an entropy-based connectivity evaluation metric (ECM) for topological correctness measurement. For all the metrics, larger values indicate better performance.

A. Pixel-level metrics

Pixel-level metrics (i.e., Precision, Recall and F1-score) measure the prediction accuracy of every pixel. Unlike most past works that directly compare the prediction results with the ground truth, in this work we use the relaxed version of these metrics following [9], [11], [13].

Let G_{pre} denote the predicted graph and G_{gt} denote the ground-truth graph. Note that both graphs have been densified. Precision is the ratio of pixels in G_{pre} that fall within τ distance to G_{gt}. Recall is the ratio of pixels in G_{gt} that fall within τ distance to G_{pre}. Distance τ reflects the tolerance of inaccuracy. If $\tau = 1$, the relaxed pixel-level metrics degenerate into the common-used hard pixel-level metrics. In this paper, we report the evaluation results with τ as 1, 2, 5 and 10 pixels, respectively.

B. The Proposed Entropy-based connectivity metric (ECM)

To calculate the metric measuring connectivity, two problems need to be solved. First, how to match the predicted road boundary instances ($G_{pre} = \{G^j_{pre}\}^M_{j=1}$) with the ground-truth road boundary instance ($G_{gt} = \{G^i_{gt}\}^N_{i=1}$). Second, how to measure the connectivity. In [13], a naive metric is proposed. The authors first match instances in G_{pre} with instances in G_{gt} by minimizing the Hausdorff distance. If an instance G^j_{pre} is matches with G^i_{gt}, then G^j_{pre} is said to be assigned to G^i_{gt}. For each ground-truth instance G^i_{gt}, let M_i be the number of assigned predicted instances. Then the connectivity of instance G^i_{gt} is $C_i = \frac{1(M_i>0)}{M_i}$, and the final connectivity of the whole patch is the average sum of C_i of all ground-truth instances.

This naive metric can reflect the connectivity of the prediction to some extent, but it is very sensitive to noises and outliers. This is caused by several reasons. Firstly, Hausdorff distance itself is sensitive to noises, so the matching results may not be reasonable sometimes. Secondly, assigning every predicted instance G^j_{pre} with the same weight to calculate C_i makes this metric very sensitive to short segments (e.g., a one-pixel-length predicted instance could greatly change M_i and affect C_i). Moreover, every C_i has an equal contribution to the final connectivity, which magnifies the errors of short ground-truth instances.

To relieve the aforementioned problems, *iCurb* [11] makes some adjustments to the naive metric. The matching method is changed to a voting scheme. Each pixel of G^j_{pre} finds its Euclidean-closest G^i_{gt} and makes a vote, then G^j_{pre} is assigned to the ground-truth instance G^i_{gt} that wins the most votes. This voting scheme is more stable than Hausdorff distance. Then, a weighting hyper-parameter α_i is assigned to each G^i_{gt}, which represents the ratio of the number of pixels in G^i_{gt} to the number of pixels of G_{gt}. Intuitively, α_i gives longer ground-truth instances larger weights to calculate the final connectivity.

However, the connectivity metric of *iCurb* still suffers from the affection of short predicted instances. Therefore, in this paper, we propose an entropy-based connectivity evaluation metric (ECM), which is shown in the following equation.

$$ECM = \sum_{i=1}^{N} \alpha_i e^{-C_i}, \quad \text{where } C_i = \sum_{j=1}^{M_i} -p_j log(p_j) \tag{1}$$

C_i is the connectivity of the $i-th$ ground-truth instance G^i_{gt}; N is the total number of G^i_{gt} in the current patch; M_i is the number of predicted road boundary instances G^j_{pre} that are assigned to G^i_{gt} and p_j is the ratio of the number of pixels in G^j_{pre} to the number of pixels in all the predicted instances assigned to G^i_{gt}. Entropy is a definition created by Shannon in information theory to measure information content. An event with greater uncertainty tends to have larger entropy. In our problem, entropy can measure the certainty to find a "dominant" predicted instance G^{j0}_{pre} for G^i_{gt}. "Dominant" means G^{j0}_{pre} can approximate G^i_{gt} very well. If G^{j0}_{pre} is easily found, then G^i_{gt} should have good connectivity measure C_i. In short, ECM assigns longer predicted graph instances with larger weights to calculate C_i, so that short instance could not greatly affect the final evaluation result. Therefore, ECM shows better ability to handle noise and outliers. ECM ranges from 0 to 1, and larger ECM indicates better connectivity. An example comparing different connectivity metrics is shown in Fig. 5.

V. Baseline Models

In this work, we implement several segmentation-based baselines and graph-based baselines. We also design an imitation-learning-based method which is enhanced from our previous work *iCurb* [11].

A. Segmentation-based baselines

1) Naive baseline: This baseline is proposed by ourselves. Firstly, we employ U-net [22] to obtain the segmentation map of road boundaries. Then, the segmentation results are refined by filtering noisy segments. Finally, the refined segmentation map is skeletonized to get the graph for the road boundaries.

2) DeepRoadMapper: This baseline (ICCV2017) [5] proposes to fix disconnections in road network predictions. Due to the similarity between our task and road network detection, we adapt their method for road boundary detection.

3) OrientationRefine: This baseline (CVPR2019) [6] utilizes the orientation map to enhance the segmentation of road networks and achieves improvements. Besides, an extra network is trained to refine the segmentation results iteratively. This work can be directly used for our task without many modifications.

B. Graph-based baselines

1) RoadTracer: This baseline (CVPR2018) [7] is believed to be the first work that solves the line-shaped object detection problem by an iterative graph generation approach. However, the network for feature extraction is merely a multi-layer CNN without skip connections. Moreover, the training strategy is naive.

2) VecRoad: This baseline (CVPR2020) [8] greatly improves *RoadTracer*, but follows the same core idea. Res2Net [23] is utilized as the backbone for multi-scale feature extraction. However, there is a limited improvement to the training strategy. Both *VecRoad* and *RoadTracer* work on road network detection, which has different topological characteristics with the road boundary (i.e., road boundaries are scattered polylines without branches while the road network is a connected graph with many intersections), thus we modified their exploration algorithm and the overall pipeline to make them applicable for our task. Besides, road network detection only requires coarse detection of the road center line, while road boundary detection expects the fine structure of both sides of the road. Therefore, our task has higher demands compared with road network detection.

3) ConvBoundary: This baseline (CVPR2019) [13] is the only graph-based work that directly focuses on road boundary detection. It is a 2-stage solution. First, it predicts the inverse-distance map, endpoint map and direction map simultaneously. Then, based on these 3 obtained maps, a model named cSnake is trained to generate the final graph vertex by vertex.

4) DagMapper: This baseline (ICCV2019) [9] is designed for road-lane detection. The input of both *DagMapper* and *ConvBoundary* are BEV images of pre-built point-cloud map. Compared with aerial images, BEV images of point-cloud map has much higher resolution and simpler scenarios. For example, the occlusion issue in the point-cloud map is relieved and only the near-road area is covered, which could be regarded as a kind of "attention". However, point-cloud map is time-consuming and expensive to obtain and update, while aerial images are more and more publicly available all over the world. Therefore, our task is more potential for wide application but is also more challenging.

5) iCurb: This baseline (RAL 2021) [11] is the first work that solves the iterative graph generation from the perspective of imitation learning. *iCurb* has a DAgger-based training strategy, which greatly enhances the final results. This work focuses on road curb detection, which is a subclass of the road boundary, thus it can be directly applied to our task.

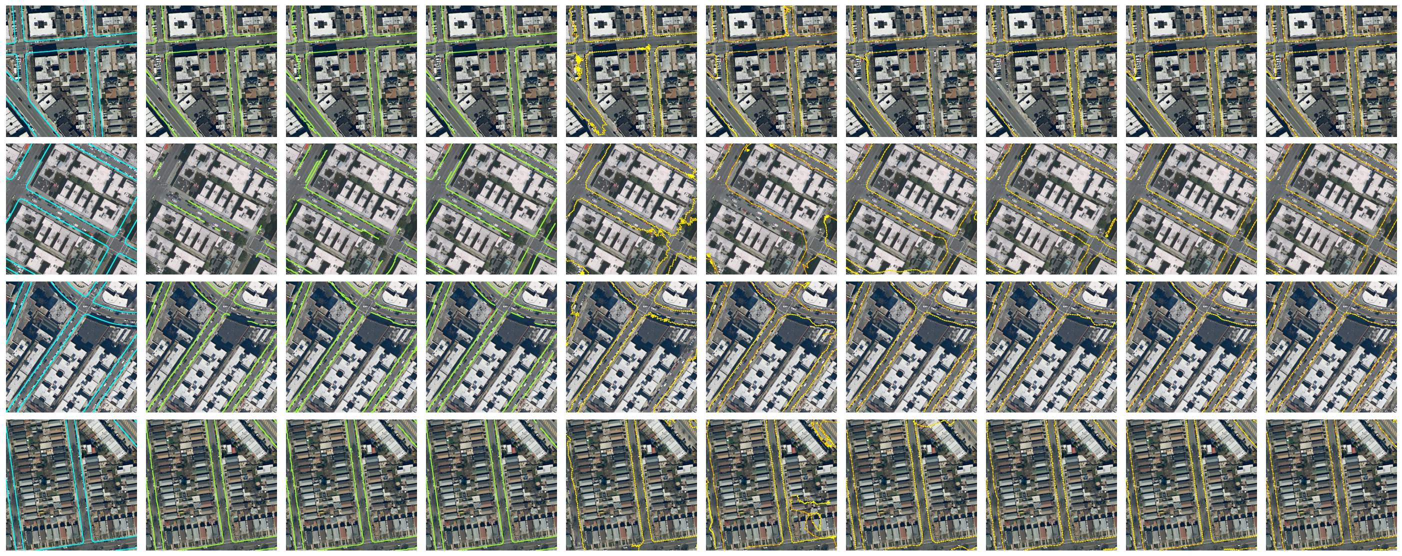
Except *RoadTracer*, the codes of all the graph-based baselines are not publicly available. So we implement them by ourselves.

C. The Proposed imitation-learning-based baseline

We propose a new imitation-learning-based solution based on our previous work *iCurb* [11], and the new solution is named as *enhanced-iCurb*. To the best of our knowledge, *iCurb* is the first work to solve the line-shaped object detection from the perspective of imitation learning. In [11], a DAgger-based solution is proposed. For each patch, the solution runs a round of restricted exploration as well as N rounds of free exploration, and aggregates the training dataset using the generated samples. Let π^* denote the expert policy and $\hat{\pi}$ denote the learner policy. The task is to learn the $\hat{\pi}$ to mimic π^*. Let $\hat{v}_t$ and v_t^* respectively denote the actions produced by $\hat{\pi}$ and π^* at time t. During the sampling period, v_t is used to denote the vertex to update the graph.

With the obtained prediction $\hat{v}_t$, we set the closest pixel of the ground-truth road boundary to $\hat{v}_t$ as the label v_t^* to train *iCurb*. This algorithm generates the label on-the-fly and can ensure *iCurb* not be affected by the *teacher-forcing* problem. However, v_t^* heavily relies on $\hat{v}_t$ and does not have a unique value, thus $\hat{\pi}$ may make unpredictable actions and may converge to a sub-optimal policy. In Fig. 6, for example, we find that the edge length of the road boundary predicted by *iCurb* is almost random. To address this issue, the orientation map M_O is leveraged to calculate v_t^* and v_t^* has a unique value. We first obtain the radian r_{t-1} of the previous vertex v_{t-1}, then we find the first vertex of the ground-truth road boundary in M_O whose radian value has large enough difference with r_{t-1}, and this vertex is used as v_t^* to train *iCurb*. The new algorithm guarantees unique v_t^*, and the trained agent has even-distributed vertices, which improves the quality of the final graph. So the label v_t^* generated by *enhanced-iCurb* is more reasonable and effective.

In *iCurb*, to better leverage the presence of expert, we proposed the restricted exploration method which updates the graph by $v_t = v_t^*$. Although the restricted exploration method assists the learner agent and accelerates the convergence, it is not necessary to run it on every patch, especially at the later training stage when $\hat{\pi}$ could generate acceptable graphs by itself. Thus, to enhance the training efficiency and provide the learner with sufficient freedom to explore the state space, we modify the restricted exploration method, in which the graph is updated by deterministic interpolation of v_t^* and $\hat{v}_t$. In SIMILE [24], deterministic interpolation is proved to be better than stochastic interpolation used in DAgger and SEARN [25] to update the trajectory. A parameter β that decays as the patch number i increases is utilized for the interpolation. v_t is calculated by $v_t = \beta v_t^* + (1-\beta)\hat{v}_t$. At the early training stage, π^* can guide $\hat{\pi}$ to accelerate the convergence; while at the later training stage, $\hat{\pi}$ itself can make accurate enough predictions to grow the graph, thus the guidance of π^* is gradually removed.



(a) GT (b) Naive (c) [5] (d) [6] (e) [7] (f) [8] (g) [13] (h) [9] (i) [11] (j) Ours

Fig. 6: Qualitative demonstrations. Each row shows an example. The first column is the ground truth (cyan lines); column (b) to (d) are segmentation-based baselines (green lines); column (e) to (i) are graph-based baselines and the last column shows the results of the new proposed imitation-learning-based baseline (orange lines for edges and yellow nodes for vertices). For better visualization, the lines are drawn in a thicker width while they are actually one-pixel width. The figure is best viewed in color. Please zoom in for details.

TABLE I: The quantitative results for the comparison. The best results are highlighted in bold font. For all the metrics, larger values indicate better performance.

Methods	Precision				Recall				F1-score				*ECM*
	1.0	2.0	5.0	10.0	1.0	2.0	5.0	10.0	1.0	2.0	5.0	10.0	
Naive baseline	0.166	0.547	0.861	0.916	0.136	0.442	0.689	0.731	0.144	0.470	0.734	0.779	0.636
Deeproadmapper [5]	0.152	0.491	0.774	0.833	0.161	0.521	0.817	0.870	0.154	0.499	0.784	0.840	0.645
OrientationRefine [6]	**0.194**	**0.608**	**0.882**	0.919	**0.180**	**0.561**	**0.811**	0.845	**0.184**	**0.576**	**0.833**	0.867	0.752
RoadTracer [7]	0.080	0.273	0.550	0.692	0.082	0.275	0.535	0.644	0.079	0.268	0.530	0.650	0.736
VecRoad [8]	0.106	0.353	0.652	0.777	0.108	0.349	0.620	0.724	0.106	0.347	0.628	0.741	0.777
ConvBoundary [13]	0.088	0.303	0.623	0.781	0.082	0.281	0.569	0.708	0.084	0.287	0.604	0.731	0.859
DAGMapper [9]	0.089	0.308	0.646	0.813	0.081	0.279	0.572	0.706	0.084	0.289	0.615	0.743	0.847
iCurb [11]	0.149	0.490	0.828	0.910	0.144	0.466	0.778	0.848	0.145	0.474	0.796	0.869	0.928
Enhanced-iCurb	0.152	0.503	0.841	**0.924**	0.146	0.476	0.784	**0.854**	0.147	0.485	0.804	**0.878**	**0.941**

VI. Experimental Results and Discussions

A. Experimental Setup

We conduct experiments on a PC with a single NVIDIA GTX1080Ti GPU card and an i7-8700K CPU. All the baselines are trained until the convergence of loss. The checkpoint with the best performance on the validation set is selected for inference. To measure the efficiency, we record the training time cost as well as the inference time cost of each baseline and report their average time cost on each patch. For graph-based baselines, the ground-truth initial vertices added with Gaussian noise are used to start the iterative graph generation process. Considering the trade-off between efficiency and effectiveness, the number of rounds of the exploration is set to 3 for both *iCurb* and *enhanced-iCurb*.

B. Evaluation Results

The comparative results are shown in Tab. I. Some qualitative demonstrations are shown in Fig. 6. The average processing time for efficiency evaluation is reported in Tab. II. As segmentation-based baselines directly optimize on pixel values, they tend to present good F1-scores, even the naive baseline. Segmentation-based baselines take relatively less time than graph-based baselines for training since they do not require an iterative process. However, these baselines do not have a satisfactory performance on connectivity, because the spatial information of the patch image cannot be fully leveraged. Compared with *Naive Baseline*, *DeepRoadMapper* can propose candidate connections to correct some disconnection and enhance the connectivity to some extent. But it is still restricted by the accuracy of the semantic segmentation. Once the segmentation network gives poor results, *DeepRoadMapper* cannot effectively improve the final performance. Moreover, generating correction candidates enlarges the time consumption. *OrientationRefine* iteratively refines the obtained segmentation map in pixel-level with an extra network, which can obtain more concise correction results. Thus *OrientationRefine* achieves good connectivity and outperforms all other methods on F1-Score.

TABLE II: The time consumption of evaluated baselines. We report the average time taken to process a single image.

Methods	Training	Inference
Naive baseline	1.12 s/image	0.67 s/image
Deeproadmapper	3.49 s/image	4.41 s/image
OrientationRefine	2.56 s/image	2.33 s/image
RoadTracer	5.52 s/image	0.88 s/image
VecRoad	18.66 s/image	3.45 s/image
ConvBoundary	6.31 s/image	0.91 s/image
DAGMapper	7.43s/image	0.90 s/image
iCurb	9.67 s/image	1.28 s/image
Enhanced-iCurb	6.91 s/image	0.89 s/image

Instead of working on pixels, graph-based baselines generate the graph of road boundaries directly. Thus, they perform well on connectivity. However, due to the sequential process for graph generation, this kind of methods exhibit relatively lower pixel-level accuracy when the patch image has complicated scenarios. The training process of *RoadTracer* is fast since it is a small network. But the performance is inferior to others. *VecRoad* follows a similar idea of *RoadTracer* but the network backbone is replaced with a more powerful one, so its performance is improved but the training efficiency is degraded. Even though *ConvBoundary* and *DagMapper* present good results with point-cloud maps on their original tasks, they do not present satisfactory performance on this task (i.e., road-boundary detection using aerial images). *iCurb* provides a solution to this task using imitation learning. It gives better F1-Score and connectivity thanks to the DAgger-based training strategy. Since *iCurb* needs to run several rounds of exploration on every patch, more training time cost is required. But its inference speed is still competitive. *Enhanced-iCurb* utilized different algorithms to generate the training label and update the graph. It presents relatively more effective and efficient results, and the qualitative visualization has more even-distributed vertices.

VII. Conclusions and Future Work

In this paper, we proposed a publicly available benchmark dataset named *Topo-boundary* for topological road-boundary detection using BEV aerial images. *Topo-boundary* had 21,556 patches. Each patch consisted of a 4-channel aerial image and 8 labels for different deep-learning sub-tasks. We also designed a new metric for better connectivity evaluation. It was employed to compare the 9 baselines (3 segmentation-based, 5 graph-based and 1 imitation-learning-based) together with the relaxed pixel-level metrics. The dataset and our implemented codes for the baselines were publicly available on our project page. In the future work, we plan to assign the prediction difficulty scores (i.e., smaller values for easy cases, and larger values for hard cases) to further label the dataset, so that the networks can be trained to be well adapted to various scenarios.

References

[1] X. Lu, Y. Ai, and B. Tian, “Real-time mine road boundary detection and tracking for autonomous truck,” *Sensors*, vol. 20, no. 4, p. 1121, 2020.

[2] X. Zhu, M. Gao, and S. Li, “A real-time road boundary detection algorithm based on driverless cars,” in *2015 4th National Conference on Electrical, Electronics and Computer Engineering*. Atlantis Press, 2015, pp. 843–848.

[3] P. Sun, X. Zhao, Z. Xu, R. Wang, and H. Min, “A 3d lidar data-based dedicated road boundary detection algorithm for autonomous vehicles,” *IEEE Access*, vol. 7, pp. 29 623–29 638, 2019.

[4] H. Wang, Y. Sun, and M. Liu, “Self-supervised drivable area and road anomaly segmentation using rgb-d data for robotic wheelchairs,” *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4386–4393, 2019.

[5] G. Máttyus, W. Luo, and R. Urtasun, “Deeproadmapper: Extracting road topology from aerial images,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3438–3446.

[6] A. Batra, S. Singh, G. Pang, S. Basu, C. Jawahar, and M. Paluri, “Improved road connectivity by joint learning of orientation and segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 385–10 393.

[7] 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.

[8] 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.

[9] 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.

[10] D. Belli and T. Kipf, “Image-conditioned graph generation for road network extraction,” *NeurIPS 2019 workshop on Graph Representation Learning*, 2019.

[11] Z. Xu, Y. Sun, and M. Liu, “icurb: Imitation learning-based detection of road curbs using aerial images for autonomous driving,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1097–1104, 2021.

[12] N. O. Department of Information Technology & Telecommunications (DoITT), “NYC-Planimetrics Database,” https://github.com/CityOfNewYork/nyc-planimetrics, 2019.

[13] 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.

[14] V. Mnih and G. E. Hinton, “Learning to detect roads in high-resolution aerial images,” in *European Conference on Computer Vision*. Springer, 2010, pp. 210–223.

[15] N. Homayounfar, W.-C. Ma, S. Kowshika Lakshmikanth, and R. Urtasun, “Hierarchical recurrent attention networks for structured online maps,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3417–3426.

[16] Z. Li, J. D. Wegner, and A. Lucchi, “Topological map extraction from overhead images,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1715–1724.

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

[18] 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.

[19] M. Haklay and P. Weber, “Openstreetmap: User-generated street maps,” *IEEE Pervasive computing*, vol. 7, no. 4, pp. 12–18, 2008.

[20] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, “Sequence level training with recurrent neural networks,” *arXiv preprint arXiv:1511.06732*, 2015.

[21] J. Liang and R. Urtasun, “End-to-end deep structured models for drawing crosswalks,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 396–412.

[22] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[23] S. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. H. Torr, “Res2net: A new multi-scale backbone architecture,” *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[24] H. Le, A. Kang, Y. Yue, and P. Carr, “Smooth imitation learning for online sequence prediction,” in *International Conference on Machine Learning*. PMLR, 2016, pp. 680–688.

[25] H. Daumé, J. Langford, and D. Marcu, “Search-based structured prediction,” *Machine learning*, vol. 75, no. 3, pp. 297–325, 2009.