Supplementary document for IEEE RA-L submission: *iCurb*

March 31, 2021

Zhenhua Xu, Yuxiang Sun, Ming Liu

1 Introduction

In this supplementary document more examples and demonstrations are provided to help readers to understand our paper. Our paper is also accompanies with a demo video at `https://tonyxuqaq.github.io/iCurb/`.

2 Training strategy

In this section, details of our methods are discussed.

In our paper, to generate training samples for imitation learning, we design an DAgger-based training strategy. It consists of two kinds of exploration methods (i.e., the restricted exploration method and the free exploration method). The pipeline is visualized in Fig. 1.

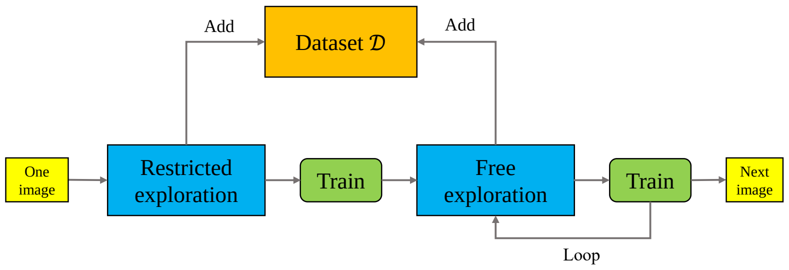


Figure 1: The diagram of our proposed training strategy. Given an aerial image as input, iCurb first runs the restricted exploration method. Then it adds the generated samples into the aggregated dataset $\mathcal{D}$ and we train iCurb on $\mathcal{D}$. After this, we run N rounds of free exploration. In each iteration, we aggregate the data and train iCurb on $\mathcal{D}$. To trade off effectiveness and efficiency, we set N as 5. Finally, iCurb finishes processing the current aerial image and turns to work on the next one.

2.1 Principles for label generation

We first clarify some principles that both exploration methods must satisfy. We want to make sure that the agent moves forward until the stop action is triggered. Thus the label v_t^* used to train the agent should meet two constraints: (1) v_{t+1}^* should be after v_t^*. Here "after" means v_{t+1}^* is more closed to the end vertex along the curb. (2) v_{t+1}^* should be away from v_t^* enough, in order to prevent the agent from being trapped in a local infinite loop. So in our implementation, we first obtain points of the ground-truth road curb that are after v_t^* and have large enough distance to v_t^*, and then find v_{t+1}^* among them. Such a process is visualized in Fig. 2.

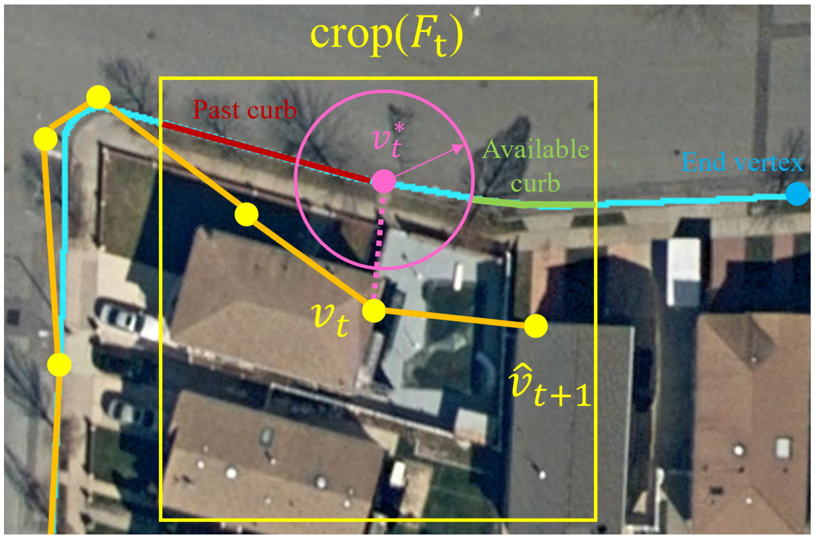


Figure 2: The visualization of the principles to generate labels for agent training. The ground-truth road curbs are shown by cyan lines. The label used to train the agent is shown by the pink node. The generated vertices are denoted by yellow nodes, and edges are solid orange line segments. The yellow rectangle is the attention region (i.e., $crop(F_t)$). Suppose now we have v_t and v_t^*. In this example, the agent moves from left to right towards the blue end vertex. Within $crop(F_t)$, the red lines represent explored road curb pixels (i.e., past road curb). Then, we draw a pink circle centering at v_t^*, whose radius is 15 pixels in our experiments. v_{t+1}^* should be outside of this circle. And in this example, v_{t+1}^* must locate on the green road curb pixels (i.e., available road curb). In this way, we can guarantee that v_{t+1}^* is after v_t^* and away from it far enough at the same time.

2.2 The restricted exploration

In the restricted exploration method, we have some restrictions on the agent $\boldsymbol{A}$: (1) $\boldsymbol{A}$ is restricted to be within 15 pixels to the ground-truth road curb graphs. (2) $\boldsymbol{A}$ stops when and only when it is closed to the end vertex. The above restrictions can make sure that $\boldsymbol{A}$ travels along road curbs and generates relatively high-quality training samples. The process is visualized in Fig. 3 for better demonstration.

Correspondingly, the above restrictions are realized by the following principles: (1) When the distance between $\hat{v}_{t+1}$ and v_{t+1}^* is smaller than 15 pixels, we directly add $\hat{v}_{t+1}$ into the graph and continue. But if the distance is larger than 15 pixels, we will make a correction and add v_{t+1}^* into the graph instead to prevent $\boldsymbol{A}$ from being far from the ground-truth road curb. (2) If v_t^* is closed to the end vertex within a threshold distance, $\boldsymbol{A}$ should stop.

The restricted exploration method can generate more accurate training samples around the ground-truth road curbs so it can accelerate the convergence and improve the overall quality of the final results. But the restricted exploration method causes a gap between training and testing. During the testing period, without corrections, the agent $\boldsymbol{A}$ may take unpredictable actions if it runs into unexplored states (e.g., when $\boldsymbol{A}$ is too far from the right track). Thus methods only with restricted exploration may not be satisfactory for our task.

As the training goes on, when the free exploration methods can make more and more high-quality predictions, the benefit of the restricted exploration method drops (this is like when the student becomes stronger and stronger, the teacher becomes less and less important). For better efficiency, the restricted exploration methods could be gradually reduced or even removed. This is a trade-off between speed and quality.

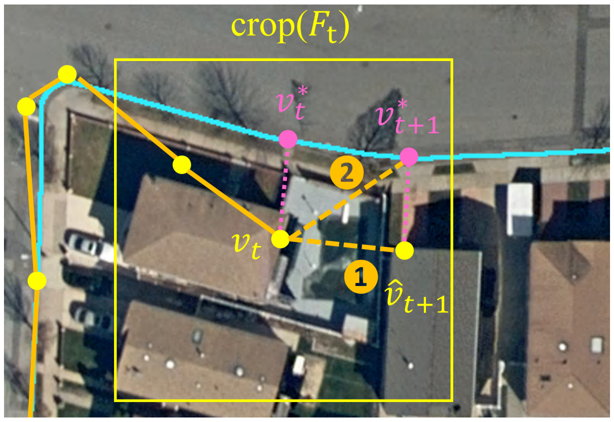


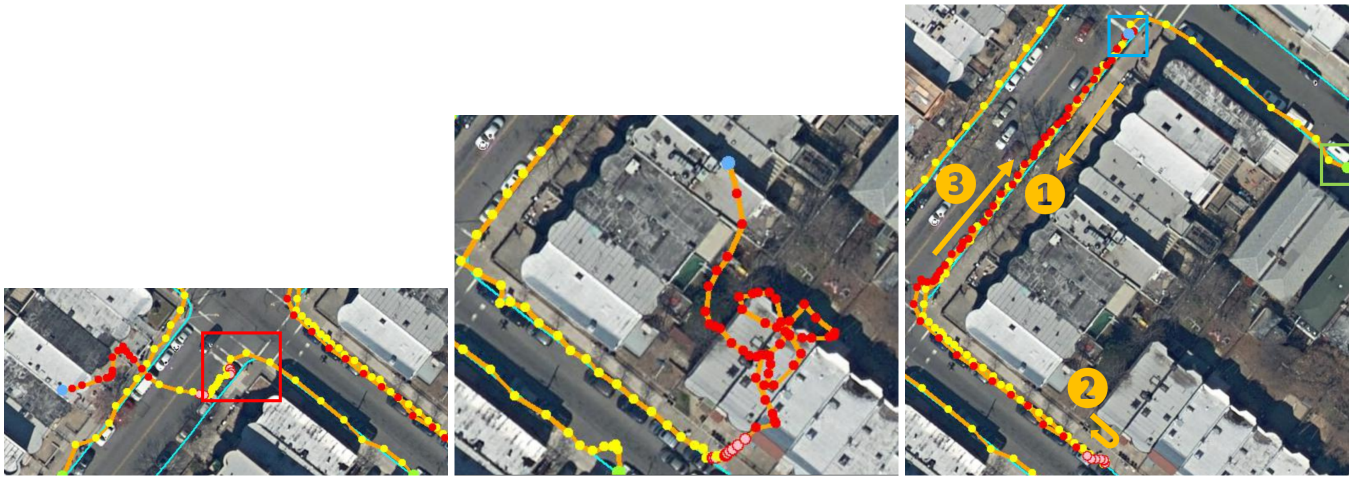
Figure 3: The demonstration of the restricted exploration method. The ground-truth road curbs are shown by cyan lines. The generated vertices are denoted by yellow nodes, and edges are solid orange line segments. The dashed orange lines represent possible edges that could be added in step $t+1$. Assume $\boldsymbol{A}$ is now at the vertex v_t whose ground-truth label is v_t^*. Based on $crop(F_t)$ represented by the yellow rectangle, the agent $\boldsymbol{A}$ makes the prediction of the next vertex $\hat{v}_{t+1}$. Then we find the nearest point of the ground-truth road curbs to $\hat{v}_{t+1}$ as the label v_{t+1}^*. There are some restrictions on v_{t+1}^*, e.g., it must be after v_t^* and the distance between them should be larger than a threshold. If the distance between v_{t+1}^* and $\hat{v}_{t+1}$ is lower than 15 pixels, $\hat{v}_{t+1}$ is directly added into the graph ($\hat{v}_{t+1}$ and dashed edge ① are picked); but if the distance is larger than 15 pixels, v_{t+1}^* is used instead to update the graph (v_{t+1}^* and dashed edge ② is picked).

2.3 The free exploration

There are several kinds of uncommon states that have a great impact on the results: (1) Normal stop action. Normal stop action means that $\boldsymbol{A}$ stops when it is very close to the end vertex. (2) Stop action when $\boldsymbol{A}$ is too far from the current road curb. $\boldsymbol{A}$ may reach someplace where it loses sight of the current curb instance. In some cases, it finds another curbs instance and follows the wrong instance, which may happen when two curb instances are parallel and close to each other. (3) Stop action when $\boldsymbol{A}$ is trapped locally. $\boldsymbol{A}$ may be trapped in a local infinite loop, and it may or may not be able to get out, so it should be stopped after such an infinite loop is detected. Once such a state is not handled properly, the processing time would greatly increase. (4) Stop action when $\boldsymbol{A}$ goes backward (i.e., $\boldsymbol{A}$ turns back and moves towards the initial vertex instead of the end vertex along a road curb).

These states rarely happen during testing, but if the stop head fails to trigger the stop action when they occur, the obtained graph will have serious errors. For example, if $\boldsymbol{A}$ is not stopped from leaving the ground-truth road curbs, a long and incorrect curb instance will be generated. Thus, there should be enough samples for the agent to learn how to handle these cases during the training period. Some examples are visualized to show the above states occurred during the free exploration in Fig. 4. For more visualization, please see section 3.1.

Graph-based baselines in our comparison experiments cannot effectively handle these states due to their naive training strategy. Thus, they tend to generate more noisy outputs. The free exploration can generate enough samples to cover these uncommon states so that iCurb could achieve much better performance. Then, due to the unbalanced distribution of the samples, we use weighted loss function during training.



(a) State (2) (b) State (2) and state (1) (c) State (4)

Figure 4: Visualization of the generated graph once uncommon states is not properly handled. For all images, cyan lines are ground-truth curbs. Blue points denote the end vertices and green points denote initial vertices. Orange line segments are edges of the trajectories. Yellow points are normally added vertices, and red points represent vertices that stop action should have been triggered. Red points with pink fill mean that a local infinite loop is detected ($\boldsymbol{A}$ may or may not escape from the loop).
(a) Example of state (2). $\boldsymbol{A}$ fails to trigger a stop action or make corrections when it generates incorrect vertices (in the red rectangle). Then an obvious incorrect curb instance is obtained.
(b) Example of state (2) and state (1). The agent $\boldsymbol{A}$ fails to stop when it is closed to the end vertex, and turns in a wrong direction, and generate many meaningless vertices.
(c) Example of state (4). The agent correctly generates the graph at first (①). However, when it gets closed to the end vertex, it suddenly turns and goes back (②). Then it generates incorrectly vertices along the curb in a wrong direction towards the initial vertex (③).

2.4 Dynamic labeling

As Fig. 3 shows, we actually have many options for the label v^*_{t+1} in training. So it does not make sense to generate a fixed label for every training sample $crop(F_t)$ during explorations. Therefore, we generate the label v^*_{t+1} after $\hat{v}_{t+1}$ is obtained. For samples generated by the restricted exploration method, the label v^*_{t+1} is obtained by the way described in Fig. 3. For samples generated by the free exploration method, the label is calculated in a similar way but with no restrictions (i.e., we directly find the point of ground-truth road curbs that is nearest to $\hat{v}_{t+1}$ as v^*_{t+1}, and v^*_{t+1} has no restrictions). L1-loss is used for the coordinate head and weighted binary cross-entropy loss is used for the stop head.

In short, we generate all the labels to train the agent on-the-fly. Although this may require more training time and computation resources, the training is more reasonable and the final performance could be enhanced.

3 Additional qualitative results

3.1 Trajectories in training

The trajectories of iCurb during the training period are visualized in Fig. 5 (at beginning training stage), Fig. 6 (at early training stage) and Fig. 7 (at late training stage). Both the restricted exploration method and the free exploration method are visualized. Through these three figures, we show how two exploration methods work and how iCurb evolves as time goes by.

3.2 The results of iCurb

More examples of the results of iCurb are visualized in Fig. 8 and Fig. 9.

References

[1] Favyen Bastani, Songtao He, Sofiane Abbar, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, Sam Madden, and David DeWitt. Roadtracer: Automatic extraction of road networks from aerial images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4720–4728, 2018.

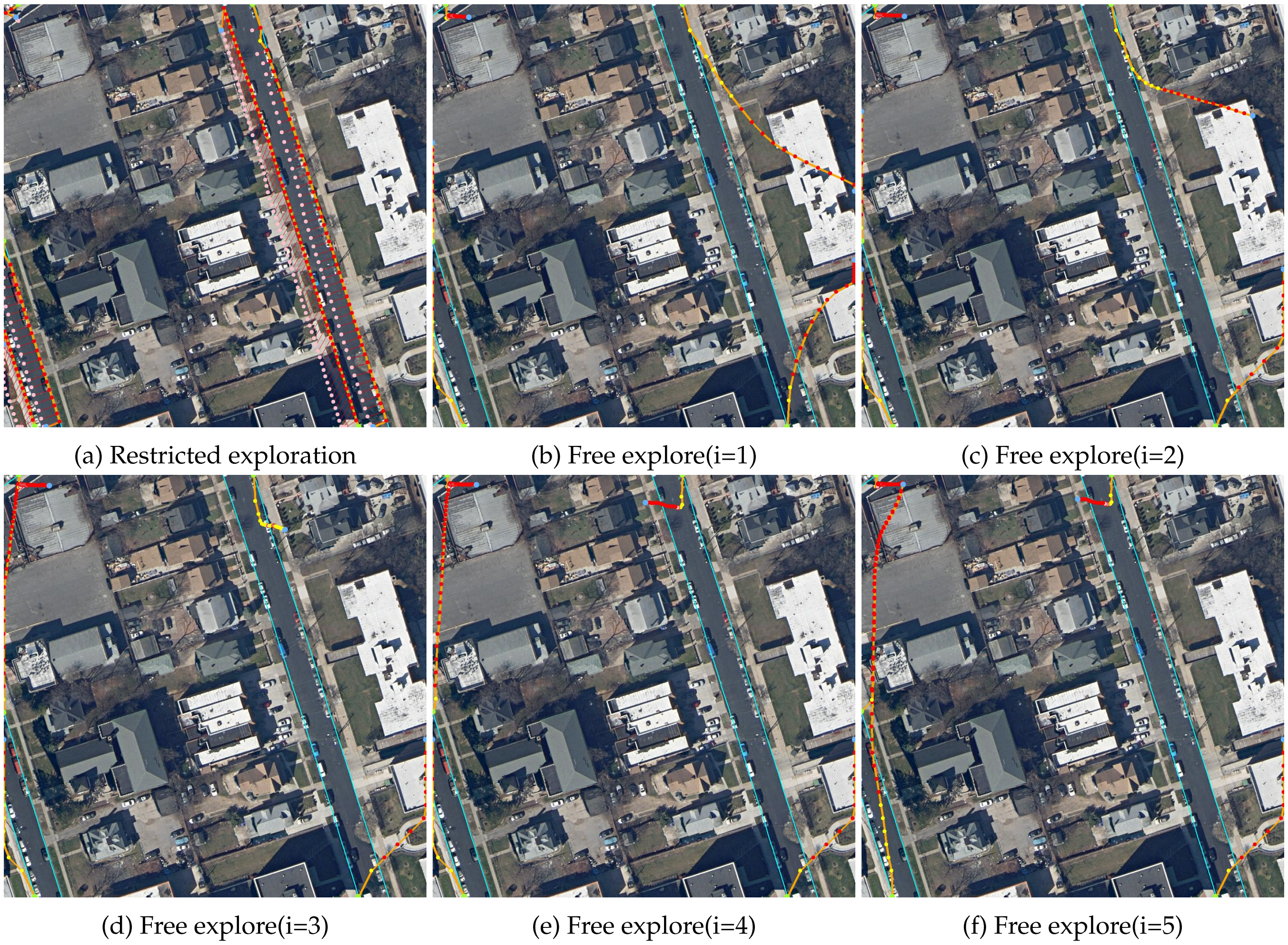
[2] Anil Batra, Suriya Singh, Guan Pang, Saikat Basu, CV Jawahar, and Manohar Paluri. Improved road connectivity by joint learning of orientation and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10385–10393, 2019.

[3] Namdar Homayounfar, Wei-Chiu Ma, Justin Liang, Xinyu Wu, Jack Fan, and Raquel Urtasun. Dagmapper: Learning to map by discovering lane topology. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2911–2920, 2019.

[4] Gellért Máttyus, Wenjie Luo, and Raquel Urtasun. Deeproadmapper: Extracting road topology from aerial images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3438–3446, 2017.

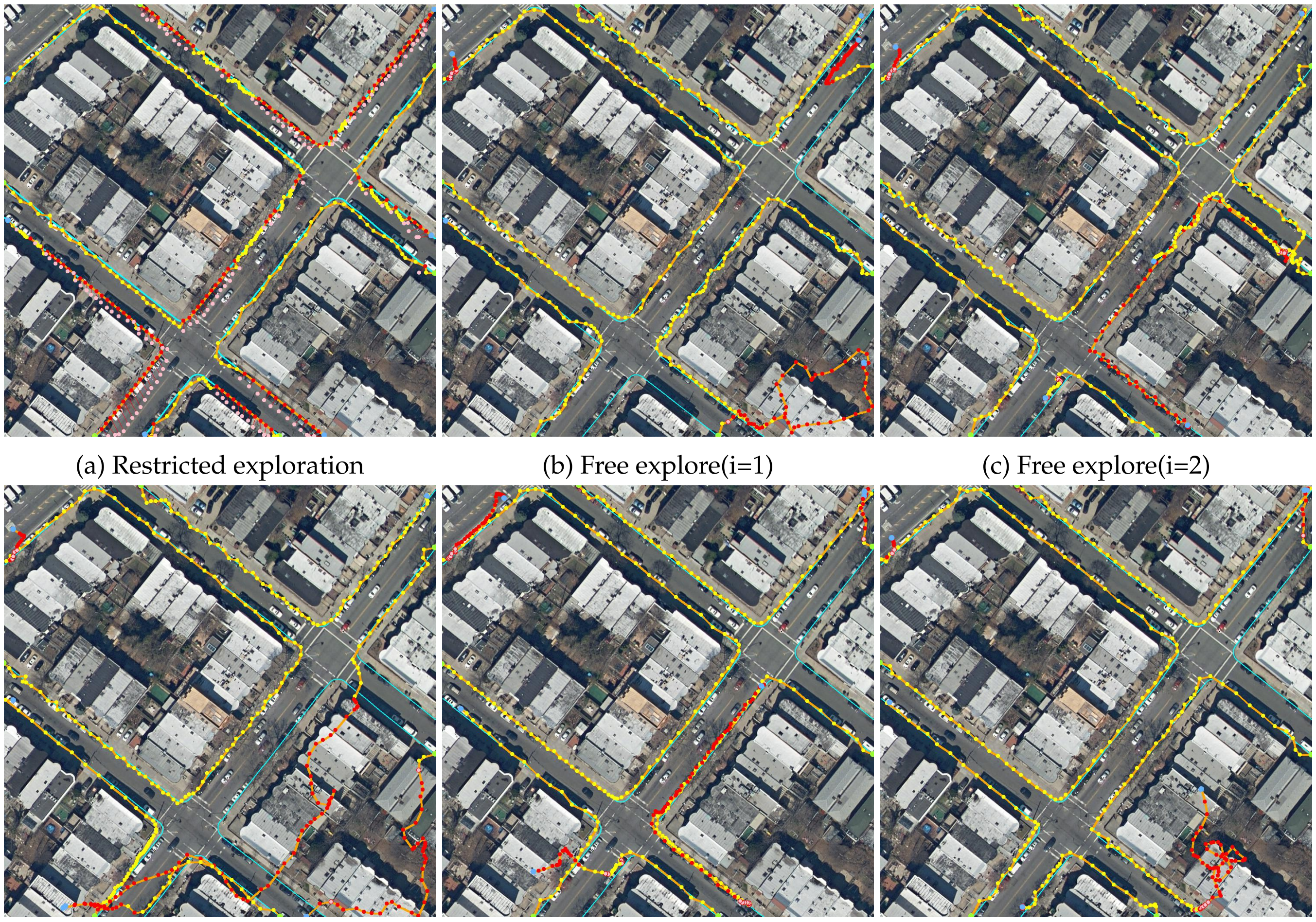
[5] Volodymyr Mnih and Geoffrey E Hinton. Learning to detect roads in high-resolution aerial images. In *European Conference on Computer Vision*, pages 210–223. Springer, 2010.

[6] Yong-Qiang Tan, Shang-Hua Gao, Xuan-Yi Li, Ming-Ming Cheng, and Bo Ren. Vecroad: Point-based iterative graph exploration for road graphs extraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8910–8918, 2020.


(a) Restricted exploration (b) Free explore(i=1) (c) Free explore(i=2)

(d) Free explore(i=3) (e) Free explore(i=4) (f) Free explore(i=5)

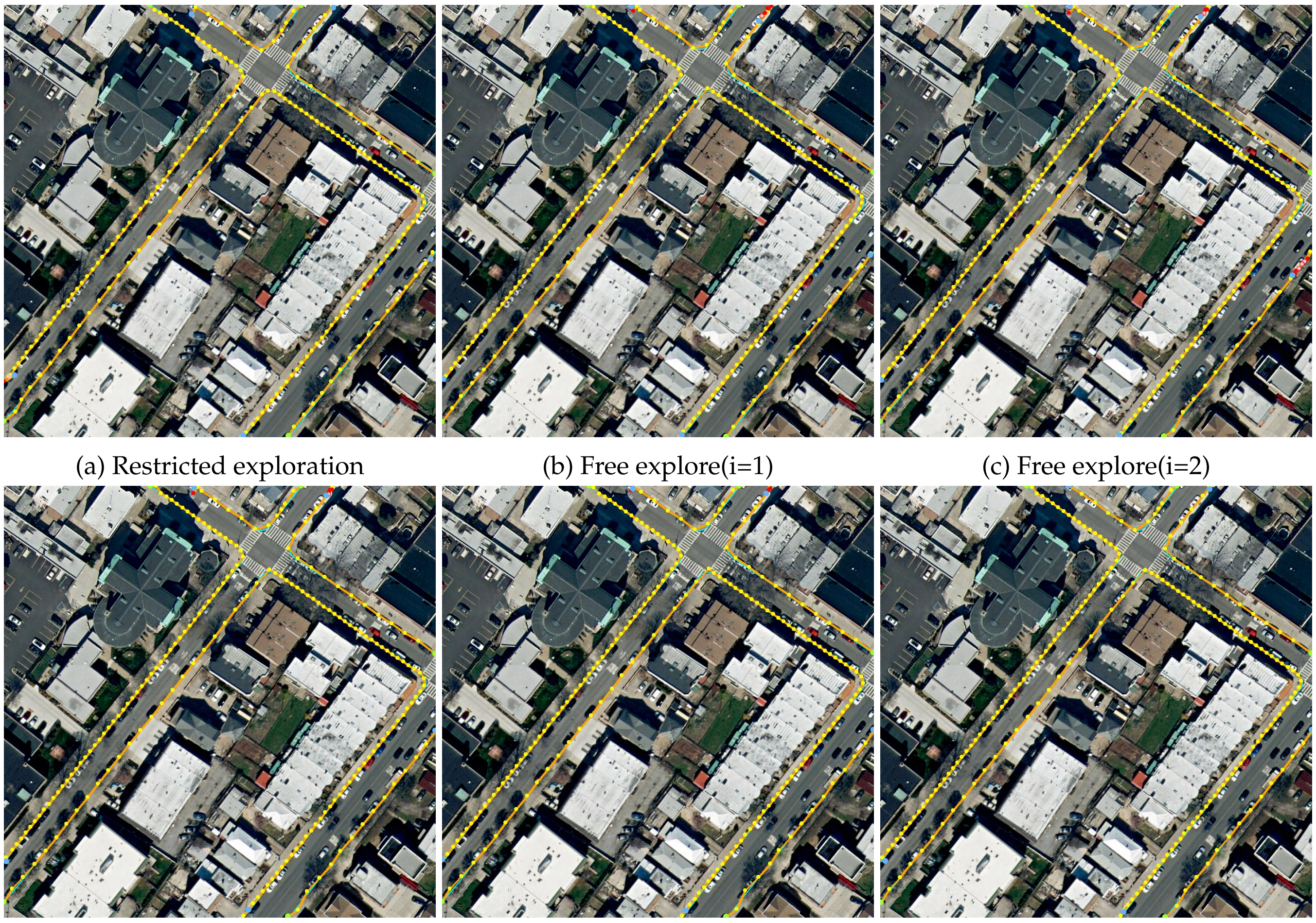
Figure 5: The trajectories of iCurb during training at beginning. For all images, cyan lines are ground-truth curbs. Blue points denote the end vertices and green points denote initial vertices. Orange line segments are edges of the trajectories. **(a)** Trajectories of the restricted exploration method. Yellow points are normally added vertices ($\hat{v}_{t+1}$), and red points represent added v^*_{t+1} (i.e., $\hat{v}_{t+1}$ is 15 pixels away from the ground-truth and v^*_{t+1} is used to update the graph). To visualize such a correction process, we use pink points to represent $\hat{v}_{t+1}$ that are far away and we connect them with the corresponding v^*_{t+1} by red lines. **(b)-(f)** 5 rounds of free exploration. Yellow points are correctly added vertices. Points with red outline mean that error occurs and $\boldsymbol{A}$ fails to stop. Pure red points represent uncommon states (2) or (4) in 2.3. Red points with pink fill represent state (3) in 2.3. **Note:** At very early stage, the agent $\boldsymbol{A}$ cannot effectively generate the correct trajectories for sampling by itself, so it generates meaningless trajectories for the free exploration methods. The restricted exploration method can obtain useful training samples which accelerate the convergence, which helps a lot at the beginning.



(a) Restricted exploration (b) Free explore(i=1) (c) Free explore(i=2)

(d) Free explore(i=3) (e) Free explore(i=4) (f) Free explore(i=5)

Figure 6: The trajectories of iCurb during training at early stage. For all images, cyan lines are ground-truth curbs. Blue points denote the end vertices and green points denote initial vertices. Orange line segments are edges of the trajectories. **(a)** Trajectories of the restricted exploration method. Yellow points are normally added vertices ($\hat{v}_{t+1}$), and red points represent added v^*_{t+1} (i.e., $\hat{v}_{t+1}$ is 15 pixels away from the ground-truth and v^*_{t+1} is used to update the graph). To visualize such a correction process, we use pink points to represent $\hat{v}_{t+1}$ that are far away and we connect them with the corresponding v^*_{t+1} by red lines. **(b)-(f)** 5 rounds of free exploration. Yellow points are correctly added vertices. Points with red outline mean that error occurs and $\boldsymbol{A}$ fails to stop. Pure red points represent uncommon states (2) or (4) in 2.3. Red points with pink fill represent state (3) in 2.3. **Note:** After training for some time, $\boldsymbol{A}$ could make reasonable predictions for the majority of cases. Compared with Fig. 5, the corrections made by the restricted exploration (red points) obviously decrease. Without restrictions, $\boldsymbol{A}$ could generate generally acceptable trajectories, while it still makes severe errors in some curb instances.



(a) Restricted exploration (b) Free explore(i=1) (c) Free explore(i=2)

(d) Free explore(i=3) (e) Free explore(i=4) (f) Free explore(i=5)

Figure 7: The trajectories of iCurb during training at late stage. For all images, cyan lines are ground-truth curbs. Blue points denote the end vertices and green points denote initial vertices. Orange line segments are edges of the trajectories. **(a)** Trajectories of the restricted exploration method. Yellow points are normally added vertices ($\hat{v}_{t+1}$), and red points represent added v^*_{t+1} (i.e., $\hat{v}_{t+1}$ is 15 pixels away from the ground-truth and v^*_{t+1} is used to update the graph). To visualize such a correction process, we use pink points to represent $\hat{v}_{t+1}$ that are far away and we connect them with the corresponding v^*_{t+1} by red lines. **(b)-(f)** 5 rounds of free exploration. Yellow points are correctly added vertices. Points with red outline mean that error occurs and $\boldsymbol{A}$ fails to stop. Pure red points represent uncommon states (2) or (4) in 2.3. Red points with pink fill represent state (3) in 2.3. **Note:** After a long time of training, $\boldsymbol{A}$ can generate relatively high-quality trajectories. The training process aims to further improve the accuracy of the obtain graph and teach $\boldsymbol{A}$ to better handle some minor issues.

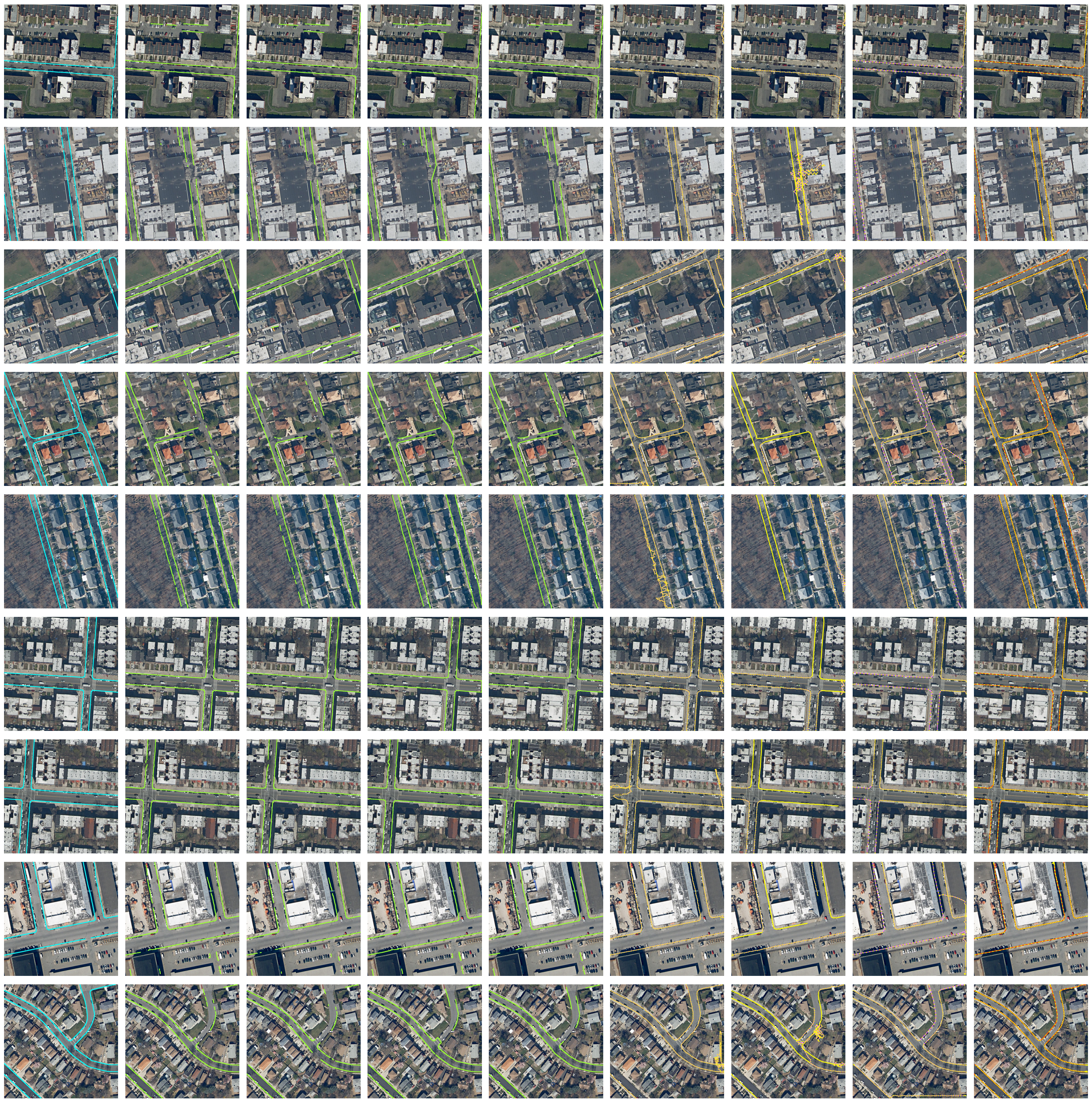
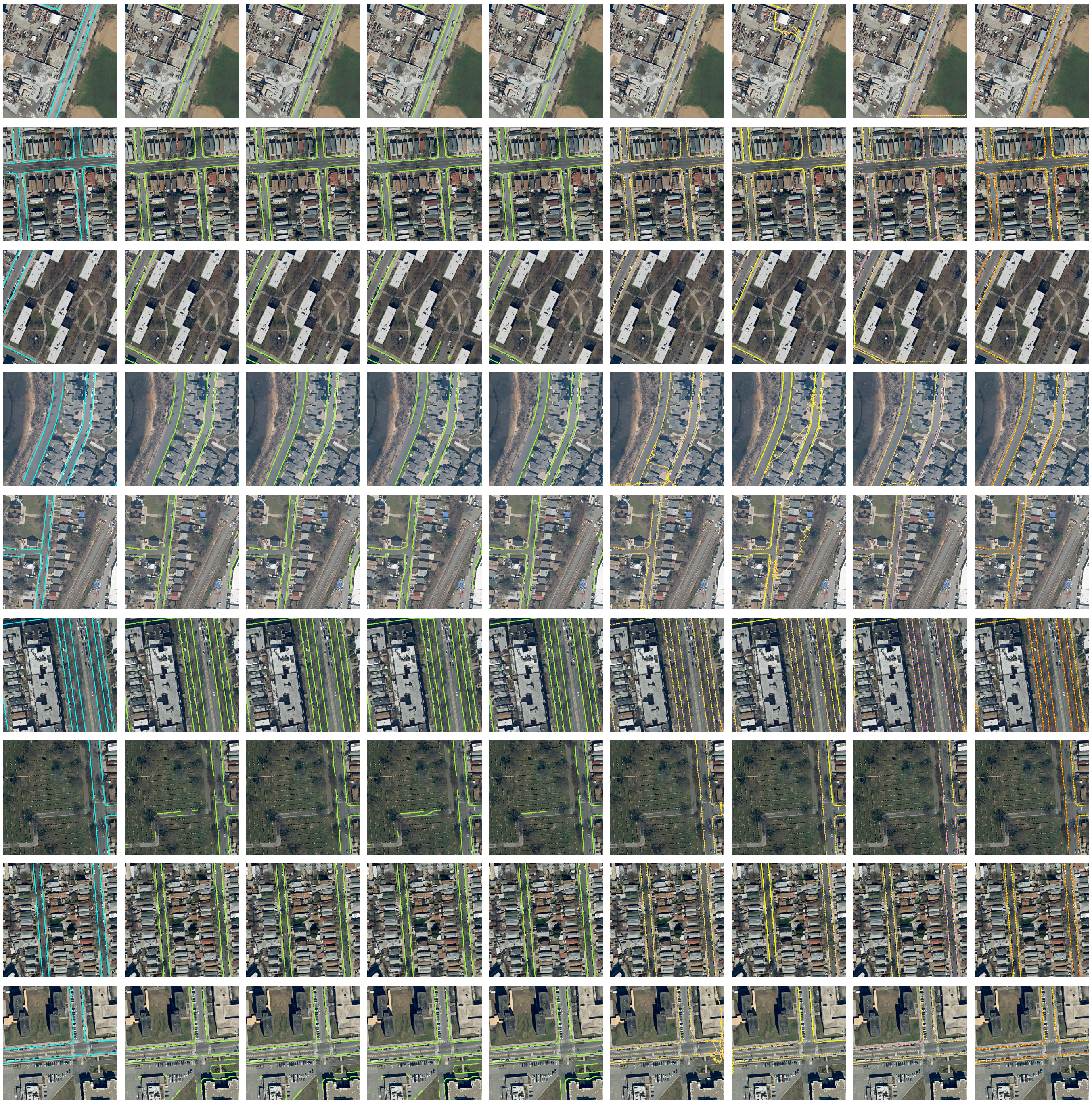


Figure 8: The demonstration of sample experimental results. The left column represents the ground-truth (cyan lines); columns (b) to (e) show the results of the segmentation- based baselines (green lines), and columns (f) to (h) are the results of the graph-based baselines (pink lines for edges and yellow nodes for vertices); the most right column shows the results of our model named iCurb (orange lines for edges and yellow nodes for vertices).



(a) GT (b) Naive (c) [5] (d) [4] (e) [2] (f) [1] (g) [6] (h) [3] (i) Ours

Figure 9: The left column represents the ground-truth (cyan lines); columns (b) to (e) show the results of the segmentation- based baselines (green lines), and columns (f) to (h) are the results of the graph-based baselines (pink lines for edges and yellow nodes for vertices); the most right column shows the results of our model named iCurb (orange lines for edges and yellow nodes for vertices).