A. More Experiments

We first assess the performance of our proposed method in an ideal scenario where the key points, correspond to the ones from the ground truth. To hinder training and inference, we insert additional key points as (1) random intermediate points between known edges and (2) randomly sampled locations in the images. Here, our assumption in Section 3 that the set $\mathcal{V}'$ suffices to generate the ground truth graph, holds by construction. We compare our method against several baselines that learn to connect edges between key points, using the same feature extraction pipeline, described in Section 3.1, as our model. *Cls* is a classification network that predicts for all pairs of key points a value $\{0, 1\}$ corresponding to the existence of an edge. *GCN* implements a graph neural network that predicts directly the adjacency matrix. We also present an autoregressive version of our model *ARM*, that is trained with cross-entropy loss to predict the pre-defined ordered sequence of key points. We use this model to initialize ours. Results are presented in Table 3 in the main text.

As expected, the ARM model achieves a low perplexity score when evaluated against the corresponding sequence, ordered according to the autoregressive order, but suffers in predicting the edges when in random order. The ARM underperforms because of frequent early terminations and the implicit inability to revisit key points, what the desired final metric is concerned, here APLS. Reward and value estimates enable a different training scheme that deeply correlates with the desired objective, fixing possible alignment issues.

B. Interpretability

We visualize attention (of the Transformer II module), using the attention flow proposed in [1], in Fig. 8. To create attention scores per edge, we aggregate scores for the pair of tokens that define each edge. New predictions lay increased attention to already generated junctions, parallel road segments, and other edges belonging to the same road segment.

We also compare APLS results achieved by varying the difficulty of the ground truth images in terms of the total number of junctions (vertices with a degree greater than 2) and in terms of the average length of road segments that are present, in Fig. 9. Our method explicitly captures information regarding the degree of the key points during the search, while it can encode better global information, even across larger distances. It is not a surprise perhaps then, that it outperforms the baselines more convincingly as the difficulty of the ground truth road network increases.

Finally, we visualize an example of an imagined rollout trajectory at a single step of our algorithm in Fig. 10. During a single inference step, our method uses tree search to look ahead into sequences of actions in the future. For our example, we have chosen a relatively smaller number of simulations (10) for better visual inspection. We also show the corresponding environment states reached, which are, however, not explicitly available to the model, as it is searching and planning using a learned model of the environment.

C. Dataset creation

We use CityEngine, a 3D modelling software for creating immersive urban environments. We generate a simple road network and apply a rural city texture on the created city blocks, provided by [23]. We then uniformly generate trees of varying height and size along the side walks of the generated streets. We then iteratively scan the generated city by passing a camera of specific orientation and height. We repeat the same process after suitable modifications to the texture, for the generation of the street masks, as well as the vegetation masks, that correspond to only the plants along the side walks. Some examples of the generated images are provided in Fig. 11. We note that additional occlusion can be caused by the relation of the camera with the 3D meshes corresponding to buildings. These occlusions are, however, not captured by our generated masks, and we can expect them to contribute partially to the fragmented segmentation results.

We train a segmentation-based model, LinkNet, as our baseline. We rasterize the ground truth graph to create pixel-level labels and train by maximizing the intersection over union, which is commonly done in practice. We note that there is a tradeoff between the nature of the predictions and the choice of the line-width with which the ground truth graph is rasterized. A large width achieves better results in terms of connectivity of the predicted graph but results in poorer accuracy in the final key points' locations. Furthermore, when providing a large width, areas in the image with more uncertainty, e.g. vegetation that is not above a road segment, are also predicted as road networks with high certainty, leading to spurious, disconnected road segments. To highlight the advantages of our method compared to this baseline and in order to promote more meaningful predictions, we select a relatively smaller width.

D. Architecture details

As an image backbone model, we use a ResNet-18 for the synthetic dataset and a ResNet-50 for the real dataset experiments. We extract features at four different scales, after each of the 4 ResNet layers. To extract features for each key point, we interpolate the backbone feature maps based on the key points' locations. We use different learned embeddings based on the actual key points' locations. For the key points embedding model, we use a transformer encoder with 12 self-attention layers and a dropout rate of 0.10. We

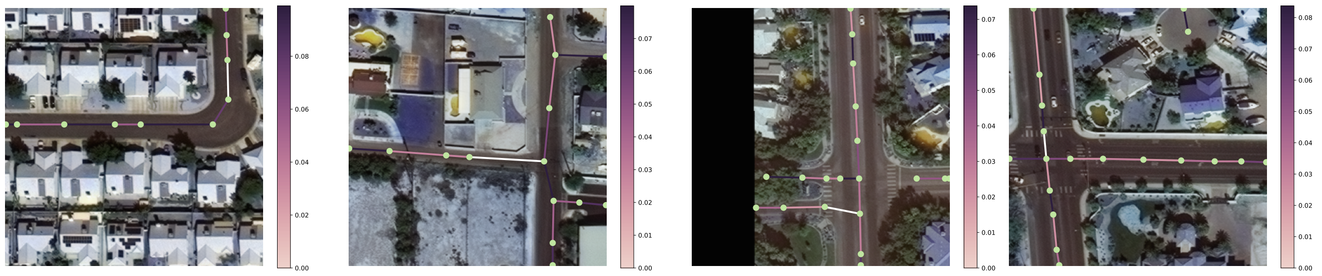

Figure 8. We visualize attention by aggregating attention scores for key points that form the same edge. White denotes the latest edge added, for which the attention scores are calculated. Colours indicate the amount of attention on any current edge in the graph.

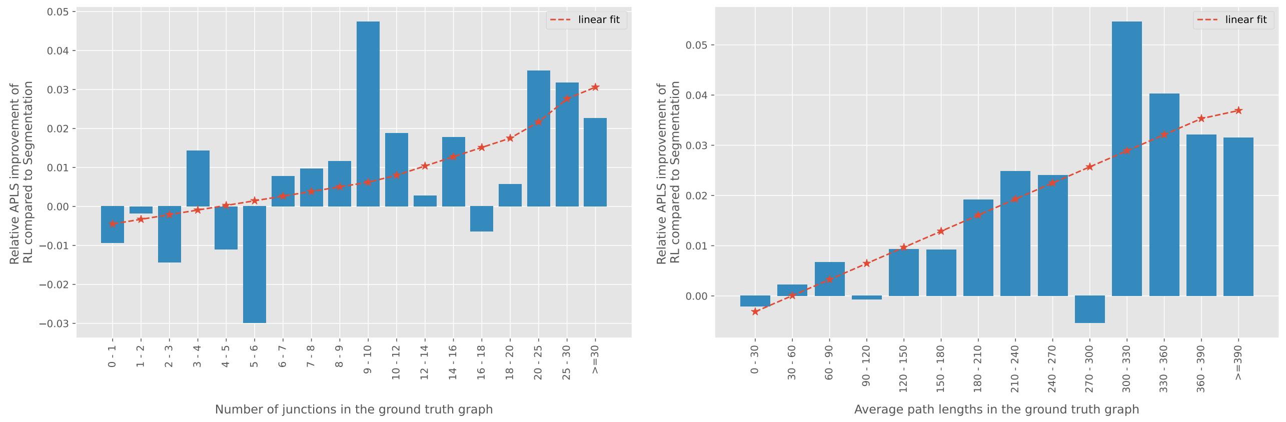

Figure 9. APLS improvement of our method, compared to the Orientation method [8], based on images capturing a ground distance area of 400×400 meters of the SpaceNet dataset. (Left) We vary the number of junctions present, and (right) the average road segment length (in meters).

use layer normalization and GELU activation functions.

For the edge-embeddings model, we use the respective key points embedding, along with learned position and type embeddings, which we all sum together. As aforementioned, we can initialize the current edge sequence based on previous predictions, allowing our model to refine any initial prediction provided. Again, we use the same transformer architecture with 12 self-attention layers, and a dropout rate of 0.10.

Finally, the architecture of the dynamics network and the value prediction network are shown in Fig. 12. For the value estimation, we also provide the current environment step, as we execute steps in an environment with a bounded time horizon. Finally, the type embeddings, inspired by [36], help to distinguish between starting and ending nodes for each edge.

E. Implementation details

E.1. Evaluation metrics

APLS [60] constitutes a graph theoretic metric that faithfully describes routing properties. APLS is defined as

$$\text{APLS} = 1 - \frac{1}{N_p} \sum_{p_{v_1 v_2} < \infty} \min \left\{ 1, \frac{|p_{v_1 v_2} - p_{v_1{}' v_2'}|}{p_{v_1 v_2}} \right\}, \tag{4}$$

where v and v' denote a source node and its closest point on the predicted graph if such exists within a buffer. N_p denotes the number of paths sampled and $p_{v_1 v_2}$ the length of the shortest path between two nodes. Similarly, the *Too Long Too Short (TLTS)* metric [65] compares lengths of the shortest paths between randomly chosen points of the two graphs, classifying them as *infeasible*, *correct*, or too-long or too-short (*2l+2s*) if the length of the path on the predicted graph does not differ by more than a threshold (5%) compared to the ground truth path. Since small perturbations to the predicted graph can have larger implications to pixel-level predictions, the definitions of precision, recall

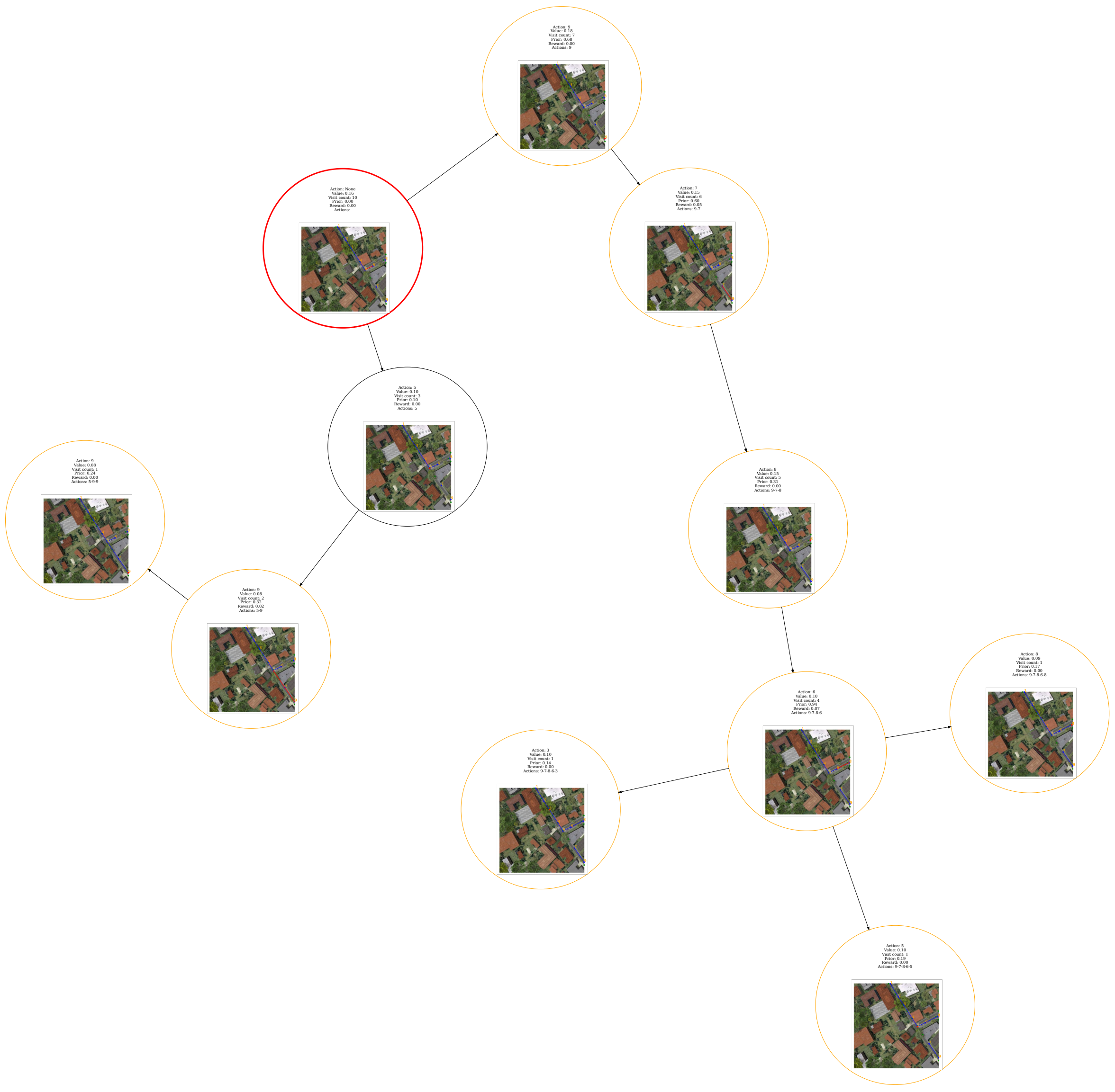

Figure 10. Example of a "dreamt" tree search of our model (we recommend zooming in for more details). Here, images correspond to the environment observations by following the respective sequence of actions, which are, however, not given as input to the model but only portrayed for visual purposes. The model has only access to the original observation and "dreamt" trajectories on the learned latent space. The root of the search tree is indicated by the colour red. Orange nodes correspond to the children that attain the highest estimated value. We also provide reward and value estimates of our model based on the current latent representation. We perform a smaller number of simulations into the future, for visual purposes.

and intersection over union were relaxed in [66, 64] leading to the metrics *Correctness/Completeness/Quality (CCQ)*.

Still, some types of errors, such as double roads or over-connections, are not penalized from the above metrics [14]. We therefore additionally include new metrics introduced in [14] that compare *Paths*, *Junctions* and *Sub-graphs* of the graphs in question, producing respectively *precision*, *recall* and f_1 scores. For the final similarity score used in Eq. 3, we use a linear combination of the aforementioned metrics, more details are available in the supplementary material.

E.2. Dataset Information

We use the following datasets to train our models, i.e. baselines and our newly proposed RL agent.

SpaceNet [60] includes a road network of over 8000 Km over four different cities: Vegas, Paris, Shanghai, and Khartoum, where the complexity and quality, and regularity of

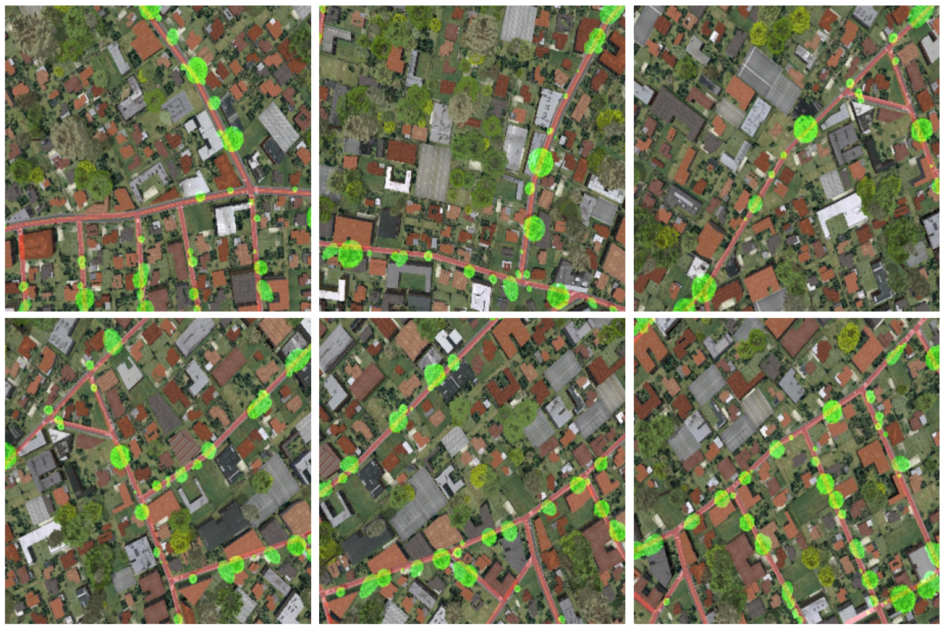

Figure 11. Samples from the synthetic dataset. We generate images and the corresponding street mask, overlaid with the colour red, along with the masks of plants that are occluding the ground truth road network, overlaid with the colour green.

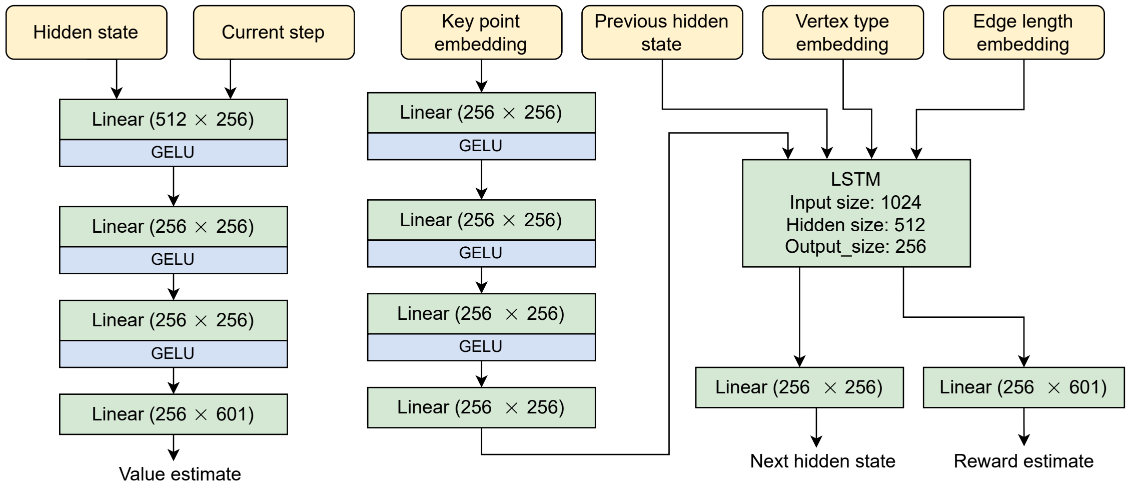

Figure 12. Architecture details of the dynamics network and the value prediction network. Reward and value are determined by predicting support of size 601. Final scalar values are calculated by an invertible transform of this support, similar to [44].

the road network depend on the city of origin. Satellite images are provided at a pixel resolution of 1300×1300, corresponding to a ground resolution of $30cm$ per pixel. We split the 2780 total images into crops of size 400×400 with an overlap of 100 pixels for training. To better highlight the diversity of the satellite images from these four different locations, we have included some randomly sampled examples in Fig. 13.

DeepGlobe [15] contains satellite images from 3 different locations with pixel-level annotations. Images have a resolution of 1024×1024, with a ground resolution of $50cm$ per pixel. We crop the 6226 images into tiles, leading to a similar ground truth resolution per pixel compared to SpaceNet.

E.3. Training details

At each MCTS search step, we perform several simulations from the root state s^0 for a number of steps $k = 1, \ldots$ and select an action that maximizes the upper confidence

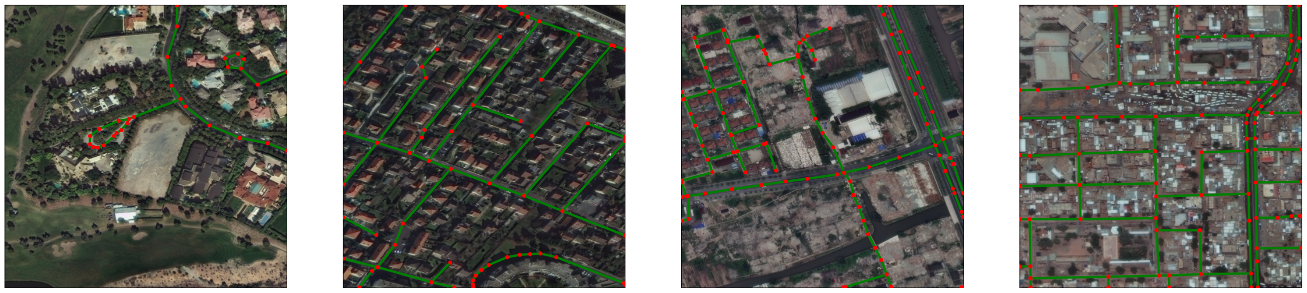

Figure 13. Example images sampled from the four cities of the SpaceNet dataset. Images from different cities exhibit different regularity in their road networks. The quality of the overhead satellite images may also vary.

bound [51],

$$a^k = \arg\max_a \left[Q(s,a) + P(s,a) \frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)} \left(c_1 + \log\left(\frac{\sum_b N(s,b) + c_2 + 1}{c_2} \right) \right) \right],$$

where $N(s,a), Q(s,a), P(s,a)$ corresponds to the visit counts, mean values and policies, as calculated by the current search statistics. Constants c_1, c_2 balance exploration and exploitation. Based on a state s^{k-1} and a selected action a^k, a new state s^k and reward $\hat{r}^k$ are estimated through the dynamics network. We update the mean values based on bootstrapped values of the estimated value functions and rewards. We experimented with training the reward and value support predictions with both mean squared error (MSE) and cross-entropy loss. We opted for MSE because of its stability. For a more in depth description of the training scheme of MuZero we recommend [48] and [71].

As hinted in the main text, we train using intermediate rewards, a linear combination of topological metrics. We experimented using a variety of different scores and metrics, but ended up using *APLS*, *Path-based f1*, *Junction-based f1* and *Sub-graph-based f1* at a relative scale of $(0.35, 0.25, 0.25, 0.15)$. We found the *Sub-graph-based f1* to be more sensitive to small perturbations and therefore weighted it less in the final combination. The metrics mentioned above are highly correlated, as examined in [8]. This correlation, though, holds when comparing the final predictions. Intermediate incremental rewards are more independent, so we still found it useful to use a mixture of them. Initially, to let our network learn basic stable rewards, we use the segmentation prediction mask as target. That means that we train our model to predict the graph that can be extracted after post-processing the segmentation model's prediction.

After pre-training the autoregressive model, we experimented with fine-tuning using RL with two different learning rates, where a slower by a factor $(0-1]$ rate was chosen for the pre-trained modules. Here, we noticed that the model still performed better than the ARM baseline. As it has trouble though to escape the autoregressive order, compared to the single learning rate model, results are less optimal.

We finally note that by avoiding type and position encoding in the Transformer II module, we can ensure the embedded graph is permutation invariant regarding the sequence of edges and the order of key points within an edge. Our search graph can then be formulated into a directed acyclic graph, circumventing unnecessary division of the search space [9, 12], enabling more efficient sampling [47]. These updated search statistics are cumbersome to compute, though, and we found no significant efficiency improvement. They do, however, confirm our model's potential ability to handle the input graph as an unordered set, as the problem suggests.

E.4. Producing key points

We initially train a segmentation model for predicting pixel-level accurate masks of the road network. For this step, we can use any model from the literature. We extract the predicted graph by skeletonizing the predicted mask and simplifying the graph by a smoothing threshold. We then sample intermediate vertices along the largest in terms of ground length edges, to enlarge the action space. We illustrate a toy example of such a process in Fig. 14. To accelerate inference, we can also initialize our prediction graph based on the provided segmentation mask. In such a case, our method closer resembles previous refinement approaches. We additionally remove edges of connected components with small overall size and edges belonging to roads segments leading to dead ends (that means vertices of degree one), keeping though the corresponding key points in the environment state. Thus, if our model deems the existence of the respective edges necessary, it can add them once more. We plan to further investigate augmenting the action space with the ability to remove edges in future work, that would not require such a pre-processing strategy.

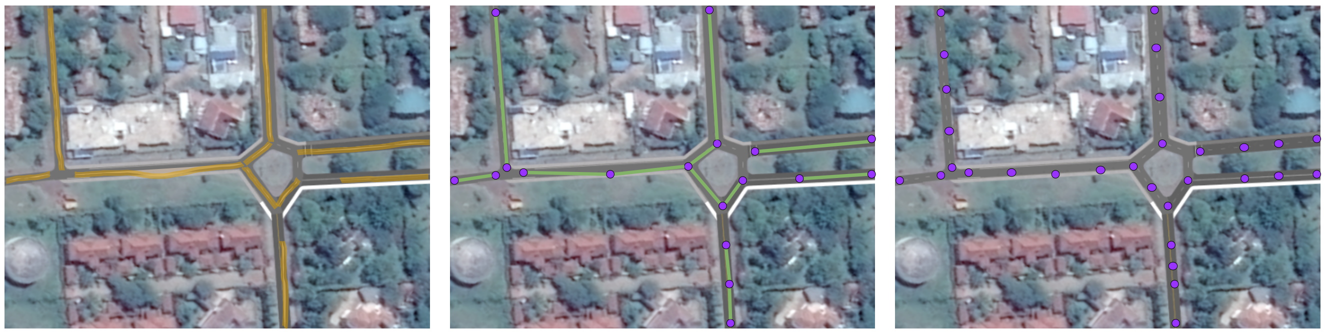

Figure 14. Example of how key points are generated. We start by evaluating a segmentation model (left) and extracting the predicted graph (middle). We then over-sample vertices along edges to enlarge the action space (right).

E.5. Combining predictions

When creating the final per image prediction, we initially simply generated predictions on non-overlapping patches and fused them together. To overcome small pixel location differences in the predicted graphs, we fuse by rasterizing the individual graphs in the pixel domain with a line width larger than 1. What we found more successful was to perform inference on overlapping patches and to initialize the currently predicted graph based on the predictions made so far. This is particularly useful, as road segments are often close to the boundaries of our cropped image. Individual inference and simple fuse can often lead to over-connected predictions. We visualize a toy example of such a process in Fig. 15.

For the segmentation baselines, unless specified in their respective documentation, we perform inference by cropping images to overlapping patches and normalizing the final predicted mask based on the number of overlapping predictions per pixel location. We also pad images around their boundary, as done in [2]. We note some small differences in the final scores for the Orientation model [8] and the SpaceNet dataset, compared to the ones in [14]. We assume these are an outcome of different chosen parameters for the calculation of metrics. We keep these parameters fixed when calculating scores for all methods.

E.6. More Comparisons with Baselines

We elaborate more on the evaluation method on Sat2Graph. The authors provided predictions corresponding only to a center crop of the original SpaceNet dataset images. For each 400×400 pixel image, predictions are made for the center 352×352 area of the image. One could expect slightly better results if trained in the same conditions but that the gap does still seem large enough to show the merits of our approach.

Other baselines like Neural turtle graphics [13] and Topological Map Extraction [26] do not have an implementation available. We do not compare against VecRoad [56] or RoadTracer [7], as different datasets were used for the current evaluations. These baselines have been already shown to underperform though in the literature, by methods that we are comparing against.

F. More examples

We showcase in Fig. 16 and Fig. 17 more examples of the environment state progression, for the synthetic dataset.

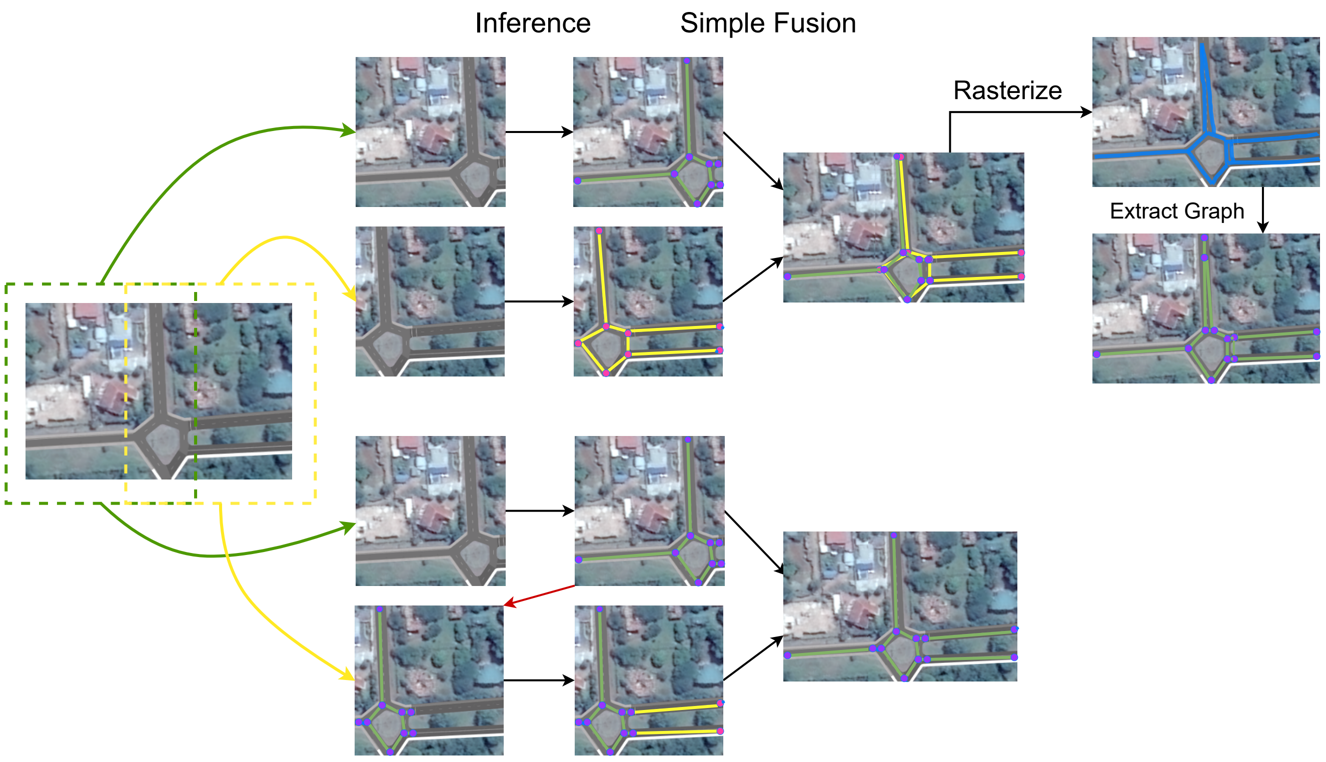

Figure 15. Toy example of how inference is performed in larger images. We start by cropping the image to overlapping patches. (Top) Naive fusion leads to over-connections because of perturbation in the key points' locations. (Bottom) We initialize subsequent graph predictions based on the key points and edge predictions of the so far generated output from previous patches if such exists.

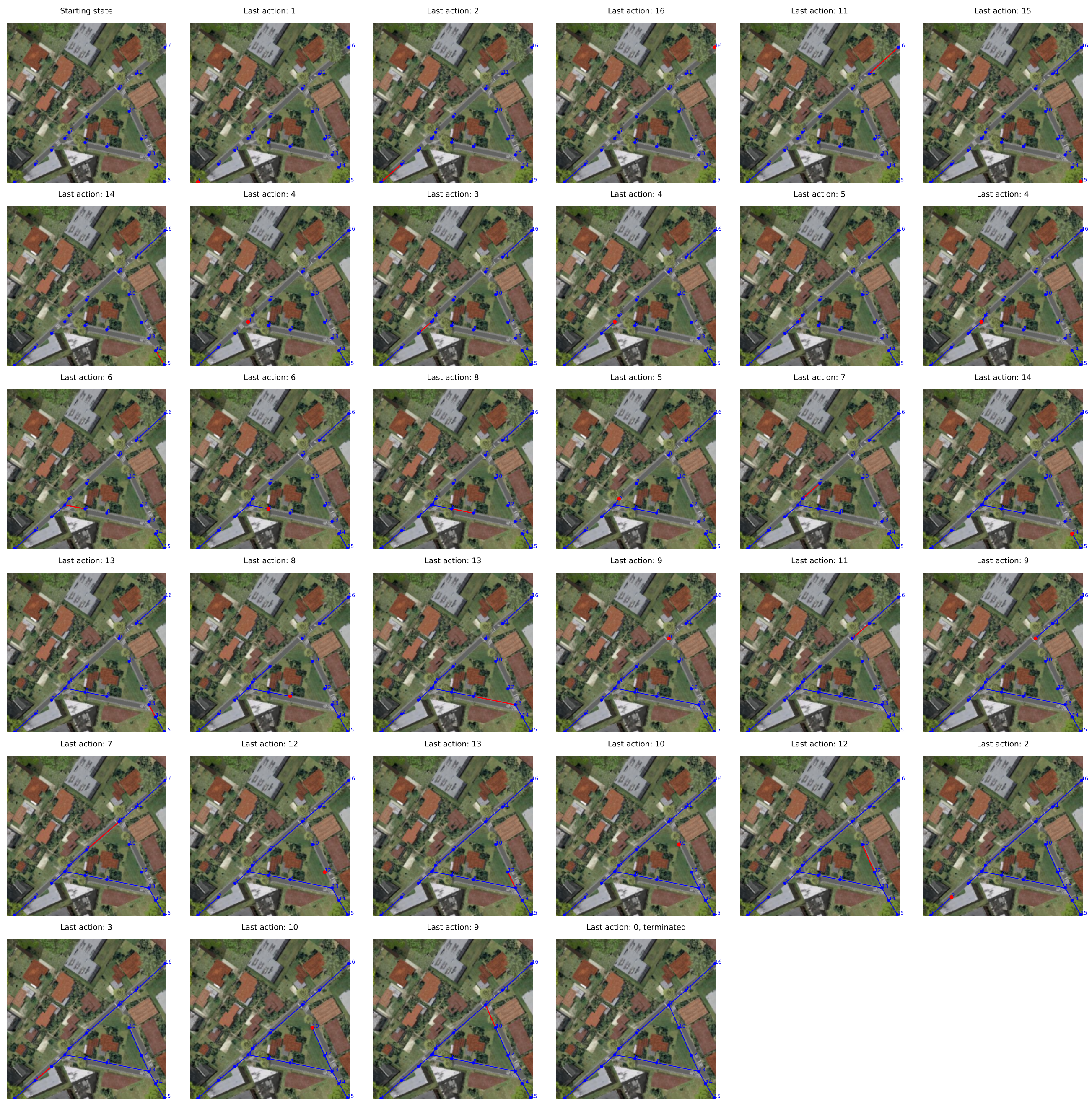

Figure 16. Example of an environment progression for the synthetic dataset. Key points' locations are shown in blue. By over-sampling initial segmentation predictions as shown in Fig. 14, we can generate key points in possibly occluded areas of the image.

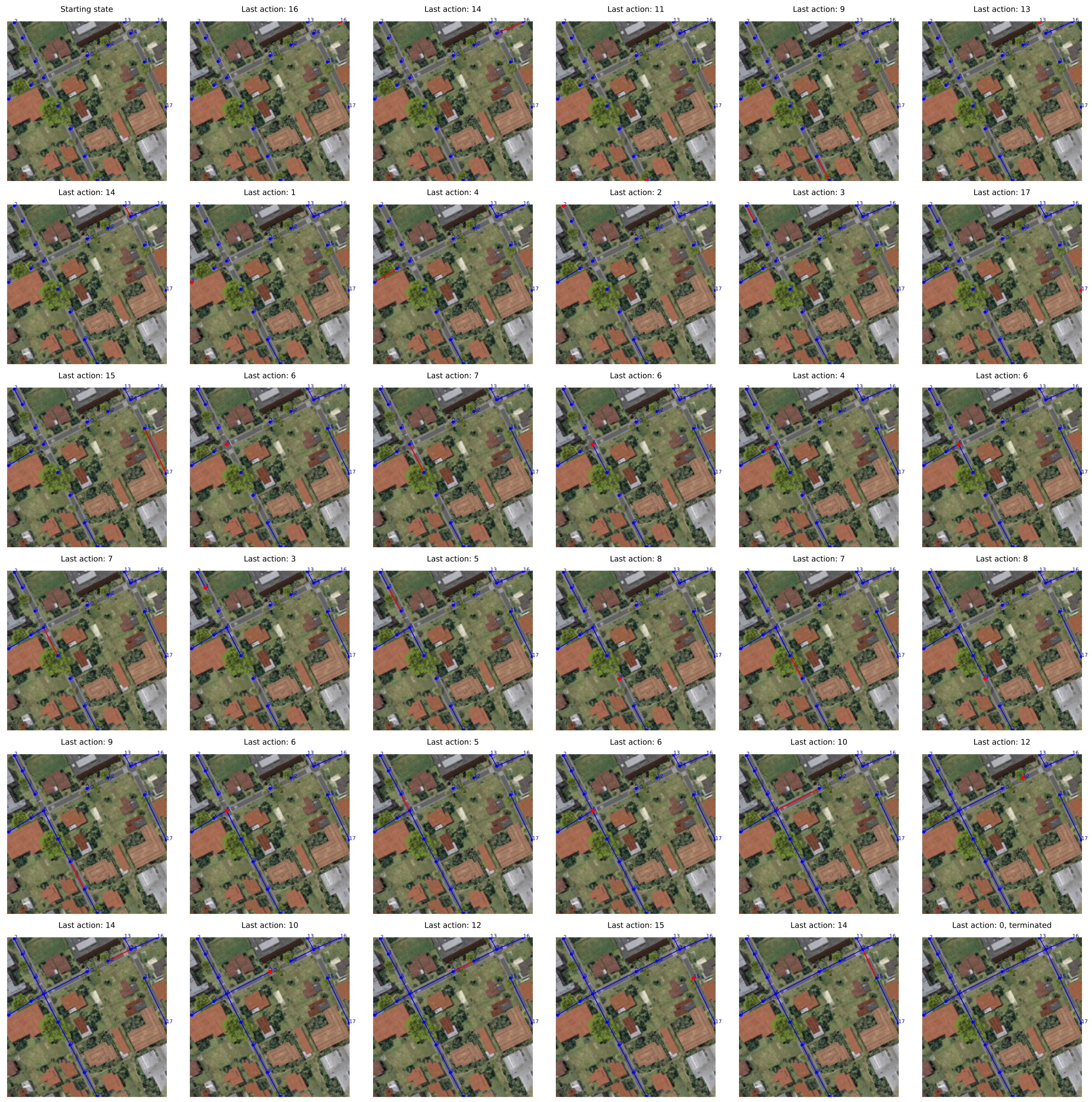

Figure 17. Example of an environment progression for the synthetic dataset. Generating the same edge twice (between key points 6 and 7), although unintuitive, does not lead to a different final predicted graph.