TraversRL: Traversable Pedestrian Pathway Generation With Reinforcement Learning

Bin Han[1], Robert Wolfe[2], and Bill Howe[1]

[1] University of Washington, Seattle, WA, USA
[2] Rutgers University, New Brunswick, NJ, USA
bh193@uw.edu, robert.wolfe@rutgers.edu, billhowe@uw.edu

Abstract. Automatically generating pedestrian pathways from aerial images requires producing a connected network suitable for routing, not just detecting where sidewalks appear. Sidewalks and crossings, in contrast to roads, may be partially occluded, implicitly defined, and exhibit complex connectivity patterns. Existing segmentation-based approaches focus on labeling pixels to infer segments, but often produce disconnected or fragmentary graphs that are unreliable for navigation. We introduce **TraversRL**, a vision-conditioned model that iteratively grows a pathway network from an aerial image, simulating a traveler navigating the built environment. TraversRL uses an action space of short and long direction–distance segments designed to adapt to complex patterns and span occlusions, and uses a combination of graph-level and step-wise rewards to balance overall connectivity with precise edge placement. Across three visual backbones and three intersection datasets, TraversRL substantially improves buffered IoU with the ground-truth graph relative to a state-of-the-art segmentation baseline, and more than doubles metrics of connectivity. Moreover, combining global and local rewards produces cleaner graphs with fewer spurious branches while further improving overall performance. These results demonstrate that modeling pathway extraction as a sequential decision process from the perspective of a traveler, while optimizing for final graph quality with reinforcement learning, produces significantly more reliable pedestrian networks.

Keywords: Pedestrian Pathway Extraction · Reinforcement Learning · Aerial Image Analysis · Iterative Graph Generation

1 Introduction

Accurate maps of pedestrian pathways are foundational for accessible navigation, mobility analysis, and urban planning. Yet these data are difficult to collect, and often incomplete or inaccurate, creating failure cases that disproportionately affect pedestrians and accessibility-critical users [31]. Inferring connected, traversible networks of sidewalks and crossings is a challenging vision problem: paths vary in width, shape, and material; they frequently occluded by trees in overhead imagery; visual cues can be subtle or absent; and small geometric errors can induce large topological failures that degrade end-to-end routability (Fig. 1).

Existing mapping pipelines rely heavily on segmentation: predicting pixel-level raster labels followed by post-processing. However, segmentation-based methods tend to produce fragmented graphs that are unreliable for navigation, as paths may not be visible from overhead, and travelers may cross the street at unlabeled crosswalks or cut through parking lots. An alternative approach is iterative graph construction, where a model repeatedly extends a partial network conditioned on local image information and the generated context, simulating a traveler making step-by-step decisions to navigate the built environment.

RL fine-tuning has played an increasingly important role for multi-step problem solving, particularly in settings where intermediate supervision is ambiguous but final outcomes are more easily scored [6,7,13,24,26,27]. This paradigm closely matches pathway generation: connectivity can be guaranteed step by step, and overall quality is measured on the final graph since small errors can accumulate.

We introduce **TraversRL**, a vision-conditioned generator that iteratively constructs pedestrian graphs from aerial imagery. It takes an intersection-level image crop and a partial graph to predict the next network extension, using a discretized direction–distance action space that includes both short and longer distance segments. We train TraversRL in two stages: **1** *Supervised pretraining.* The model learns to imitate ground-truth graph expansions step by step via cross-entropy loss, augmented with localization noise to improve robustness against drift during iterative rollouts. We denote this pretrained model as TraversRL-Pre. **2** *RL fine-tuning.* Initializing from TraversRL-Pre as a reference policy, we optimize outcome-level graph rewards under a KL-regularized objective, implemented via group-relative policy optimization (GRPO) [7, 21].

We investigate two RL reward designs. **1** *Global reward* uses a terminal, graph-level objective based on buffered geometry intersection-over-union (IoU), which aligns with final network quality. We refer to this model as TraversRL-Global. **2** *Local stepwise reward* augments the terminal objective with per-edge buffered overlap efficiency to encourage locally precise geometry. We refer to this model as TraversRL-Local. Overall, our main contributions are:

- We introduce TraversRL, a vision-conditioned generator that iteratively extracts complex pathway graphs from aerial imagery, using a discretized direction–distance action space tailored to more complicated pedestrian segments.
- We demonstrate that iterative generation with both supervised pre-training and RL fine-tuning is a natural fit for pathway tracing and consistently improves graph quality. Averaged across three backbones and three datasets, TraversRL-Global outperforms a state-of-the-art segmentation-based method in IoU by 43.6% and a sidewalk connectivity metric (TravSim) by 205%.
- We propose a local stepwise reward that measures geometric quality for advantage shaping during RL fine-tuning. On average, TraversRL-Local further improves IoU by 2.6% and TravSim by 1.6% relative to TraversRL-Global.
- We show that RL fine-tuning generalizes zero-shot to unseen cities without additional training. RL fine-tuning also yields cleaner and simpler graphs by reducing redundant edges while maintaining connectivity for navigation.

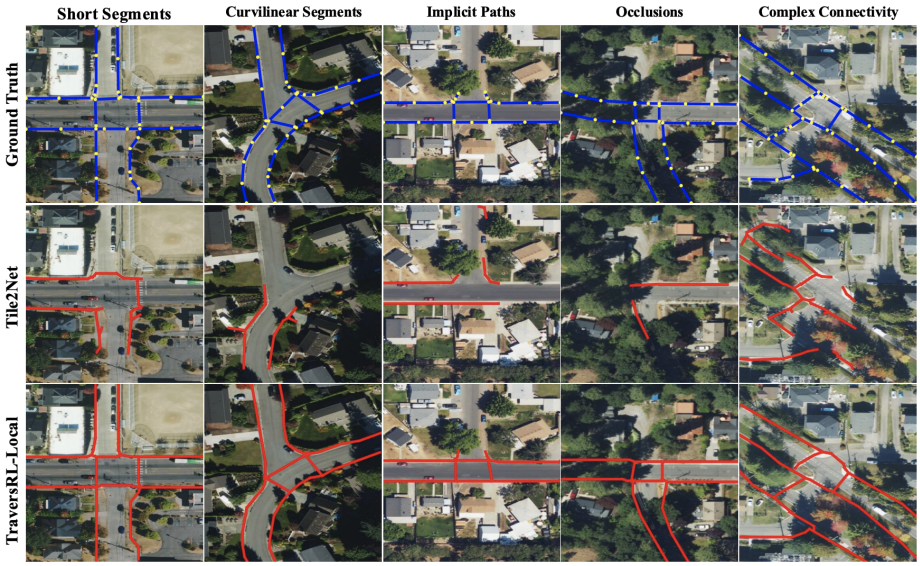

Fig. 1: Qualitative examples of pedestrian pathway generation. Representative aerial intersection scenes illustrating common challenges for pedestrian mapping: {short segments, curvilinear segments, implicit paths (no crossing marks), occlusions from trees, and complex connectivity}. **Top**: ground-truth graph (blue; yellow dots mark vertices). **Middle**: predictions from the state-of-the-art segmentation-based method, Tile2Net (red). **Bottom**: preductions from our TraversRL-Local model (red). These examples highlight the topology-sensitive nature of pedestrian networks, and demonstrate TraversRL's robust generation across diverse real-world challenges.

2 Related Work

Street and Road Network Generation. Road and street network extraction from aerial imagery has been studied extensively. Existing methods broadly fall into two paradigms. *Segmentation-first* pipelines predict dense road masks and then recover graph structure through vectorization and post-processing [4, 10, 11, 17, 18, 22, 28, 33]. In contrast, *iterative graph construction* grows a graph step-by-step by conditioning on local image evidence and the already-generated context. RoadTracer [2] introduced this crop-and-canvas tracing paradigm. VecRoad [23] improved local exploration at complex junctions, while RNGDet and RNGDet++ [29, 30] further developed transformer-based iterative graph detection with topology-oriented supervision. NETracer [15] proposed a topology-aware tracing framework by explicitly modeling node–edge connectivity.

While these approaches establish strong foundational techniques for extracting road graphs, they are not directly transferable to the pedestrian environment. Pedestrian pathways exhibit different challenges: they are typically narrower, contain many short and curvilinear segments, are more susceptible to occlusions (e.g., tree canopies), and require the explicit modeling of crossings and inter-

section connectivity to for accessibility and navigation use cases [12, 32]. These inherent differences make pedestrian mapping more sensitive to rollout drift and localized geometric mispredictions during iterative generation.

Pedestrian Pathways From Aerial Imagery. Pedestrian pathway extraction from overhead imagery remains relatively understudied. Karimi et al. [14] explored pedestrian map generation at small scale and highlighted the importance of input data quality. Tile2Net [12] detects sidewalks, crosswalks, and footpaths from aerial imagery and converts them into pedestrian networks. Prophet [31] combines road-network priors with aerial-image segmentation and rasterized street maps to infer pathway graphs (we do not compare with Prophet as neither the code nor trained model was available at the time of this writing). PathwayBench [32] argues that pixel-level metrics can understate mobility-critical errors and instead emphasizes routability and intersection-scale connectivity. Other work focuses on specific pedestrian assets rather than full connected networks, specifically zebra-crossing detection [1], crosswalk detection from satellite imagery [3, 25], and curb-ramp detection from street-view imagery [8, 19]. In contrast, our goal is direct generation of connected pedestrian pathway graphs.

Reinforcement Learning (RL) and GRPO. Reinforcement learning is increasingly used to improve long-horizon decision-making. Reasoning LLMs trained with RL show strong gains with verifiable rewards (RLVR), where policies are optimized using outcome-level signals because intermediate supervision is ambiguous [6, 7, 13, 24, 26, 27]. More broadly, RL is a natural choice for optimizing non-differentiable trajectory-level objectives when errors can compound over many decisions. Among policy-gradient methods, PPO [20] is widely used for its stability. GRPO [21] is a PPO-style variant that replaces an explicit critic with group-relative baselines computed from multiple outputs, and typically uses KL regularization to a reference policy for stable updates. Our work uses GRPO.

These developments suggest RL is a natural fit for improving sequential generators when the desired objective is best expressed at the trajectory level. In pedestrian pathway generation, local geometric choices (e.g., sidewalk continuations and crossings) interact across many steps and must ultimately produce a globally coherent, routable network, while evaluation can be defined using geometry and connectivity aware graph metrics. We therefore adopt RL for fine-tuning and use GRPO as a practical, stable instantiation in our experiments.

3 Method

We propose **TraversRL**, a vision-conditioned generator that constructs pedestrian pathway graphs step-by-step from aerial imagery. TraversRL is trained with two stages — supervised pretraining (Sec. 3.1), followed by RL fine-tuning using either a global graph-level reward (Sec. 3.3) or with an additional local stepwise reward (Sec. 3.4). The supervised pretraining phase yields a reference policy, denoted as TraversRL-Pre. During the subsequent RL fine-tuning, we

initialize the trainable policy π_θ from TraversRL-Pre while maintaining a frozen copy π_{θ_0} to serve as the reference policy for KL divergence.

3.1 Supervised Pretraining of TraversRL (TraversRL-Pre)

Graph formulation. Given a satellite image and vector pathway annotations, we convert the annotations into an undirected graph $\mathcal{G} = (V, E)$. Each node $v \in V$ is a 2D point in normalized coordinates $v = (x, y) \in [0, 1]^2$, and each edge $e = \{u, v\} \in E$ is a LineString segment. Our goal is to learn a policy that incrementally reconstructs E by predicting local edge continuations.

State: aerial image crop + generation canvas. At each step, we expand from a current node u using a local observation centered at u: (1) an RGB crop I_u and (2) a single-channel generation canvas C_u that rasterizes the partial predicted graph (all previously generated edges). Both are extracted with a fixed normalized window (0.5×0.5 of the full extent) and resized to a fixed resolution (256×256; 224×224 for ViT). We concatenate them into a 4-channel input $X_u = \text{concat}(I_u, C_u) \in \mathbb{R}^{4\times H\times W}$, so that each decision is explicitly conditioned on local appearance and the previously generated edges.

Action space: discretized direction and distance. TraversRL predicts a discrete movement action from u or STOP. For a neighbor v of u, the displacement $\Delta = v - u$ is mapped to a class by discretizing angle and distance: $N_\theta = 36$ angle bins (10°) and $N_r = 10$ distance bins over $[0, 0.2]$ (normalized to the overall image), yielding $|\mathcal{A}| = N_\theta N_r + 1 = 361$ actions, where STOP indicates that no ungenerated edges remain at the current node. We denote $a \in \mathcal{A}$ as an action.

Policy network. TraversRL uses an ImageNet-pretrained backbone $f_\phi(\cdot)$ adapted to accept 4-channel inputs. We replace the first convolution with a 4-channel variant by copying the pretrained RGB weights and initializing the additional (canvas) channel to zero. Given X_u, the backbone produces a feature vector $h \in \mathbb{R}^d$, which is mapped to action probabilities by an MLP head:

$$\pi_\theta(a \mid X_u) = \text{softmax}\big(W_2\, \sigma\big(W_1\, \text{Dropout}(h)\big)\big)\,, \tag{1}$$

where $\sigma(\cdot)$ is ReLU and $\pi_\theta(\cdot \mid X_u) \in \mathbb{R}^{|\mathcal{A}|}$.

Supervised training. For each training graph, we supervise a sequence of actions at each node: the policy emits one action per ground-truth edge, then emits STOP and moves to the next node (Fig. 2). To improve robustness to rollout drift and mitigate error accumulation, we inject localization noise by perturbing the current node location while following the ground-truth edge:

$$\tilde{u} = u + \epsilon, \qquad \epsilon \sim \mathcal{U}\big([-\eta, \eta]^2\big)\,. \tag{2}$$

We extract crops centered at $\tilde{u}$ and compute the target class using the perturbed displacement $\Delta = v - \tilde{u}$ We minimize cross-entropy over the supervised action sequence. The supervised training approach is adapted from a similar technique for road extraction [2], but includes variable distances and a different architecture to adapt to topologically complex, occluded, high-density sidewalk paths.

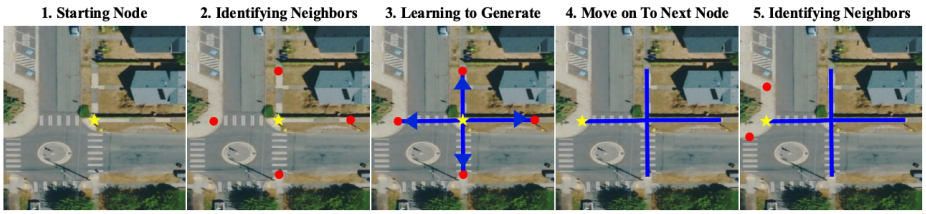

Fig. 2: Supervised node expansion in TraversRL pretraining. Given the current node (yellow star), the set of ground-truth neighbors (red) are determined. During supervised pretraining, the policy is trained to sequentially emit one direction–distance action per ground-truth edge (blue arrows), adding the corresponding edges to the canvas. After all edges at the current node have been generated, the policy emits STOP and the algorithm moves to the next node for expansion (steps 1–5).

3.2 Inference and Rollout Hyperparameters

At inference time, we roll out π_θ to construct a graph in continuous space. We initialize a start node (for training and evaluation, a random node in the ground-truth graph; in practice, a point selected by a user or based on segmentation pre-processing) and maintain a stack of active nodes. For each popped node u, we repeatedly call the policy until it emits STOP or reaches a per-node expansion budget. Each non-STOP action is decoded into a displacement using the corresponding angle–distance bin midpoints, creating a new node at $u + \Delta$ (clipped to $[0, 1]^2$). We add the edge, render it onto the global canvas, and push the new node onto the stack. To avoid trivial cycling, we disallow executing the same discrete action twice from the same node within one expansion.

Because backbones can differ in termination behaviors, we tune a small set of rollout hyperparameters on the validation set. We configure the following:

1. *Top-k backoff* (k): enumerate the k highest-logit actions and execute the first geometrically valid action. $k = 1$ equates to standard greedy decoding.
2. *Restarts per graph* (R): run the full rollout R times from independently sampled start nodes and retain the best result under an internal confidence score (defined as the sum of top-1 minus top-2 logit margins across all steps).
3. *STOP threshold* (τ_{stop}): accept STOP only when $\pi_\theta(\text{STOP} \mid s_t) \geq \tau_{\text{stop}}$.
4. *Maximum per-node expansions* (M): cap on the total number of actions permitted when expanding any single node.

We select the configuration that maximizes buffered IoU on the validation set and reuse it for the subsequent RL training and inference.

3.3 RL Fine-Tuning with Global Graph Reward (TraversRL-Global)

Motivation. Supervised pretraining optimizes per-step classification loss, whereas the quality evaluation is defined on the final graphs. Small errors can compound

over steps, causing geometric and topological failures. In addition, IoU is non-differentiable because it depends on decoding and rendering an entire trajectory. We therefore use RL fine-tuning to optimize an outcome-level graph objective under the same rollout dynamics used at test time.

RL with terminal graph reward. We initialize π_θ from TraversRL-Pre, and keep a copy as frozen reference policy π_{θ_0}. For each training sample, we draw a group of K rollouts $\{\tau_i\}_{i=1}^{K}$ using the same decoding configuration as inference. Each trajectory $\tau_i = (a_{i,1}, \ldots, a_{i,T_i})$ induces a predicted edge set $\hat{E}_i$ and receives a terminal, graph-level reward $R_i = R(E, \hat{E}_i)$, which is buffered IoU in our experiments. We compute standardized group-relative advantages:

$$\bar{R} = \frac{1}{K}\sum_{i=1}^{K} R_i, \qquad A_i = \frac{R_i - \bar{R}}{\mathrm{std}(\{R_j\}_{j=1}^{K}) + \epsilon}. \tag{3}$$

We update π_θ to increase the likelihood of actions from higher-advantage rollouts while constraining rapid drift from the frozen reference π_{θ_0} via a KL penalty. Our RL fine-tuning objective is defined as:

$$\begin{aligned}\mathcal{L}_{\mathrm{global}}(\theta) = &-\frac{1}{K}\sum_{i=1}^{K}\frac{A_i}{T_i}\sum_{t=1}^{T_i}\log \pi_\theta(a_{i,t} \mid s_{i,t}) \\ &+ \beta \cdot \frac{1}{K}\sum_{i=1}^{K}\frac{1}{T_i}\sum_{t=1}^{T_i} D_{\mathrm{KL}}(\pi_\theta(\cdot \mid s_{i,t}) \,\|\, \pi_{\theta_0}(\cdot \mid s_{i,t}))\end{aligned} \tag{4}$$

,where $s_{i,t}$ is the rollout state at step t (local image and canvas crops), β is the KL coefficient, and $1/T_i$ normalizes trajectories of different lengths. We compute the full-distribution KL at each step from the policy and reference logits.

3.4 RL Fine-Tuning with Local Stepwise Reward (TraversRL-Local)

Motivation. TraversRL-Global aligns fine-tuning with the final objective, but it assigns a single rollout-level advantage A_i to all actions within a trajectory, leading to coarse credit assignment. In practice, however, a rollout often contains a mixture of locally correct and incorrect steps, yet all steps are reinforced (or penalized) uniformly. As a result, terminal-only reward can improve global overlap, but under-optimizes fine-grained geometric choices (e.g., selecting a nearby but incorrect direction bin that yields similar buffered overlap).

Local stepwise reward . To inject a localized training signal without discarding the stable terminal objective, we define a stepwise reward that quantifies the local geometric quality of each newly added edge and use it for advantage shaping. Let P_t be the buffered union of predicted edges up to step t and let G be the buffered union of ground-truth edges. Define predicted area $A_t = \mathrm{area}(P_t)$ and

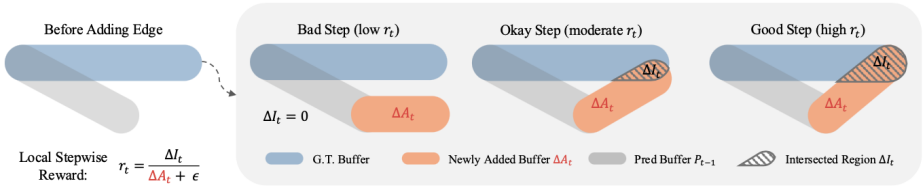

Fig. 3: Local stepwise reward. We compare the buffered ground-truth region G (blue) with the current predicted buffer P_{t-1} (gray). Adding a new edge at step t contributes newly added buffered area ΔA_t (orange), of which the overlapped portion with ground-truth is ΔI_t. We define the stepwise reward $r_t = \Delta I_t/(\Delta A_t + \epsilon)$: it is high when the added edge aligns better with ground-truth (good step), moderate under partial alignment, and near zero when the edge adds off-target geometry ($\Delta I_t = 0$).

overlap area $I_t = \text{area}(P_t \cap G)$. With a new edge, we compute $\Delta A_t = A_t - A_{t-1}$ and $\Delta I_t = I_t - I_{t-1}$, formulating the stepwise reward as:

$$r_t \;=\; \frac{\Delta I_t}{\Delta A_t + \epsilon}, \tag{5}$$

with $r_t = 0$ for non-edge actions (e.g., STOP). Intuitively, r_t measures what fraction of newly generated buffered area contributes to ground-truth overlap; edges that add mostly off-target geometry receive near-zero reward (Fig 3).

Additive advantage shaping. As in Sec. 3.3, we sample K rollouts and compute the terminal advantages A_i from the final rewards R_i. We then calculate and normalize the stepwise rewards within each individual rollout:

$$\tilde{r}_{i,t} \;=\; \frac{r_{i,t} - \mu_i}{\sigma_i + \epsilon}, \quad \mu_i = \frac{1}{T_i}\sum_{t=1}^{T_i} r_{i,t}, \quad \sigma_i = \text{std}\Big(\{r_{i,t}\}_{t=1}^{T_i}\Big), \tag{6}$$

We define an additively shaped stepwise advantage:

$$\hat{A}_{i,t} \;=\; A_i \;+\; \lambda\, \tilde{r}_{i,t}, \tag{7}$$

where λ controls the strength of shaping. The additive formulation preserves the outcome-level signal, while simultaneously encouraging locally high-quality steps even within imperfect rollouts. Since both A_i and $\tilde{r}_{i,t}$ are standardized, their magnitudes are comparable, making the shaping term well-scaled in practice. We then substitute the uniform rollout-level weighting in Eq. (4) with $\hat{A}_{i,t}$ while keeping the same KL regularizer to the frozen reference:

$$\begin{aligned}\mathcal{L}_{\text{local}}(\theta) = &-\frac{1}{K}\sum_{i=1}^{K}\frac{1}{T_i}\sum_{t=1}^{T_i}\hat{A}_{i,t}\log \pi_\theta(a_{i,t} \mid s_{i,t}) \\ &+ \beta \cdot \frac{1}{K}\sum_{i=1}^{K}\frac{1}{T_i}\sum_{t=1}^{T_i} D_{\text{KL}}(\pi_\theta(\cdot \mid s_{i,t}) \,\|\, \pi_{\theta_0}(\cdot \mid s_{i,t}))\end{aligned} \tag{8}$$

This procedure is visually demonstrated in Fig. 4.

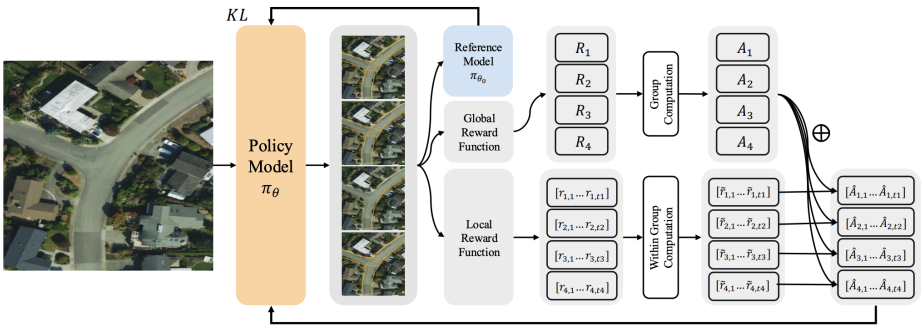

Fig. 4: RL fine-tuning with local stepwise reward shaping (TraversRL-Local). For each intersection, we sample K rollouts (here K=4) from π_θ, producing action sequences and generated graphs. A global reward assigns each rollout a terminal reward R_i, which is converted into group-relative advantages $\{A_i\}_{i=1}^{K}$. In parallel, a local reward computes per-step rewards $\{r_{i,t}\}$, which are normalized within each rollout to obtain $\{\tilde{r}_{i,t}\}$ and combined with A_i to form shaped advantages $\hat{A}_{i,t} = A_i + \lambda\tilde{r}_{i,t}$. The policy is updated using $\hat{A}_{i,t}$ together with $\mathrm{KL}(\pi_\theta \| \pi_{\theta_0})$; only π_θ is updated, while π_{θ_0} is frozen. Without local shaping (TraversRL-Global), the update reduces to $\hat{A}_{i,t} = A_i$.

Note: A natural alternative is to replace the terminal advantage with stepwise advantages normalized across rollouts at each step. However, we found this to be empirically unstable in our task. Step rewards are highly sensitive to small geometric perturbations and vary substantially across rollouts and step indices due to different rollout lengths and exploration paths. This high variance makes cross-rollout, per-step normalization noisy. In practice, using purely stepwise advantages causes rapid drift from the reference policy (KL increases sharply early in training) and leads to collapse, whereas advantage shaping preserves the stability of terminal-reward RL while still providing informative local credit.

4 Experiment

We evaluate TraversRL in both in-domain and zero-shot transfer settings. Sec. 4.1 details the evaluation datasets, Sec. 4.2 outlines the visual backbones and baseline models, and Sec. 4.3 specifies the training and evaluation protocols.

4.1 Datasets

We evaluate across three urban-intersection datasets: **1 WashingtonInter (in-domain)**. A curated dataset of three- and four-way urban intersections from 13 cities in Washington State. We first extract intersection bounding boxes from OpenStreetMap and prune overlapping regions to avoid spatial train–test leakage. We then retrieve the corresponding pedestrian pathway annotations from our internal database and discard scenes whose ground-truth graph is disconnected. After a final manual visual inspection to remove severe aerial-to-map

misalignments, the dataset contains 2,287 training scenes, 144 validation scenes, and 426 test scenes. **2 PathwayBench, Seattle (zero-shot)** [32]. We evaluate 342 intersection scenes from Seattle. Since PathwayBench does not provide training and validation splits, we evaluate in a *zero-shot* setting: models are trained on WashingtonInter and applied directly to PathwayBench without fine-tuning. We ensure that there is no overlap intersection with the WashingtonInter data. **3 PathwayBench, Washington, D.C. (zero-shot)** [32]. 589 intersection scenes from Washington, D.C., evaluated under the same zero-shot setting.

PathwayBench uses different scene extents and spatial resolutions (an effective zoom level of roughly 19, compared to approximately 20 in WashingtonInter). Therefore, we refer to all PathwayBench evaluations as zero-shot transfer.

4.2 Backbones and Baselines

We initiate TraversRL with three ImageNet-pretrained visual backbones: ResNet-18 [9], ViT-S [5], and Swin-T [16]. All are adapted to 4-channel inputs by modifying the first projection layer to accept an additional canvas channel while retaining pretrained RGB weights (the extra channel is initialized to zero). ViT uses 224×224 inputs, while ResNet and Swin use 256×256. We compare against Tile2Net [12], a state-of-the-art pipeline that predicts sidewalks, crosswalks, and footpaths from aerial imagery and converts them into pedestrian networks.

We also evaluate two road-oriented graph extraction methods, VecRoad [23] and NETracer [15], to test whether existing road-tracing pipelines transfer to pedestrian pathways. For NETracer, we convert pathway annotations into the required raster/SWC and seed/centerline inputs. For VecRoad, we convert pathway graphs into OSM-style supervision, derive junction labels from graph topology, and tune key tracing parameters. These baselines provide graph-based comparisons while highlighting the domain gap: pedestrian pathways are typically narrower, shorter, more occluded, and have subtler junction cues than roads.

4.3 Training & Evaluation

Supervised pretraining. For each backbone, we train a supervised reference model following Sec. 3.1. Unless otherwise noted, we apply localization error with $\eta = 0.02$ (in normalized coordinates), and optimize with AdamW (learning rate 1×10^{-4}, weight decay 1×10^{-4}, gradient clipping at 1.0). Models are trained for 50 epochs with gradient accumulation over 16 node-expansions, with max_steps=200. For ResNet-18 backbones, BatchNorm statistics are frozen. We select the best checkpoint by validation loss and use it to initialize RL fine-tuning.

RL fine-tuning. We fine-tune each pretrained checkpoint using GRPO (Secs. 3.3, 3.4). Unless otherwise noted, we use AdamW (learning rate 3×10^{-5}, weight decay 1×10^{-4}, gradient clipping at 1.0), train for 15 epochs with gradient accumulation over 8 graphs, and set the KL coefficient to $\beta = 0.05$. For each training graph, we sample a group of $K = 4$ rollouts, capping the maximum rollout length at 200 steps. Rewards use a buffer radius $r = 0.01$ (≈ 1 meter).

For the local shaping variant, we set $\lambda = 0.1$ and clip $\tilde{r}_{i,t}$ to be within $[-3, 3]$ for stability. We select the best RL checkpoint by validation buffered IoU.

We evaluate each method over five randomly sampled start nodes and report averages. We use two metrics: **1** **Buffered geometry IoU**, computed between the predicted and ground-truth edge sets by buffering each polyline with a 1-meter radius and calculating the IoU of the resulting regions. and **2** **Traversability similarity (TravSim)** [32], a routability-oriented metric that compares boundary-to-boundary connectivity within local tiles. A boundary pair is considered traversable if a valid path exists within the clipped tile connecting a terminating node on one boundary to a terminating node on another. Each tile thus induces a set of traversable boundary pairs for both prediction and ground-truth, and TravSim is the Jaccard similarity between these two sets, averaged over tiles. This metric does not require node/edge correspondence and directly measures whether the generated graph affords the same traversal possibilities as the ground-truth within intersection-scale regions.

5 Results

In this section, we present detailed generation performances. We structure these findings into *takeaways* to facilitate understanding, denoted as **T**#.

5.1 (T1) Supervised Policies Strongly Outperform the Baseline

Table 1 compares pretrained checkpoints (TraversRL-Pre) against Tile2Net across three datasets. Even before RL fine-tuning, the pretrained iterative graph generator yields substantially higher geometric overlap and connectivity.

1 **WashingtonInter (in-domain).** Tile2Net achieves 0.264 IoU and 0.114 TravSim on average. In contrast, TraversRL-Pre improves both metrics for every backbone, reaching 0.531 IoU and 0.605 TravSim with Swin-T. Averaged across backbones, TraversRL-Pre exceeds Tile2Net by 0.258 IoU and 0.481 TravSim. The large TravSim gain suggests the iterative formulation better preserves intersection topology than raster-to-network post-processing, where small geometric artifacts can disconnect crossings and sidewalk continuations.

2 **PathwayBench (zero-shot transfer).** The same trend holds under distribution shift. On Seattle data, TraversRL-Pre outperforms Tile2Net by an average of 0.026 IoU and 0.215 TravSim. On D.C. data, TraversRL-Pre improves TravSim by 0.196 on average, while achieving competitive IoU: Swin-T exceeds Tile2Net in IoU (0.231 vs. 0.226), whereas ResNet-18 and ViT-S fall marginally behind. Overall, cross-city transfer is more robust for connectivity.

3 **Road-oriented methods.** Road-oriented graph methods do not transfer well to pedestrian pathway generation: NETracer produced fragmented graphs likely because its road-tuned tracing rules prematurely reject valid but subtle pathway continuations. VecRoad produced near-empty outputs because it relies on high-confidence junction maps, while our generated pathway junctions are

Model		Stage	WashingtonInter		Seattle (PB)		D.C. (PB)	
			IoU	TravSim	IoU	TravSim	IoU	TravSim
VecRoad		–	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001
NETracer		–	0.098	0.005	0.080	0.003	0.007	0.002
Tile2Net		–	$0.264_{\pm 0.046}$	$0.114_{\pm 0.017}$	$0.169_{\pm 0.021}$	$0.175_{\pm 0.037}$	$0.226_{\pm 0.018}$	$0.317_{\pm 0.023}$
TraversRL	ResNet18	pre	$0.510_{\pm 0.003}$	$0.586_{\pm 0.005}$	$0.186_{\pm 0.003}$	$0.386_{\pm 0.003}$	$0.212_{\pm 0.003}$	$0.506_{\pm 0.004}$
		global	$0.539^{\dagger}_{\pm 0.001}$	$0.591^{\dagger}_{\pm 0.008}$	$0.189^{\dagger}_{\pm 0.002}$	$0.387^{\dagger}_{\pm 0.004}$	$0.223_{\pm 0.002}$	$0.507^{\dagger}_{\pm 0.003}$
		local	$0.540^{*}_{\pm 0.002}$	$0.605^{*}_{\pm 0.004}$	$0.199^{*}_{\pm 0.003}$	$0.403^{*}_{\pm 0.002}$	$0.227^{*}_{\pm 0.002}$	$0.513^{*}_{\pm 0.002}$
	ViT-S	pre	$0.525_{\pm 0.003}$	$0.593_{\pm 0.005}$	$0.194_{\pm 0.003}$	$0.382_{\pm 0.006}$	$0.208_{\pm 0.002}$	$0.499_{\pm 0.002}$
		global	$0.548^{\dagger}_{\pm 0.003}$	$0.597^{\dagger}_{\pm 0.003}$	$0.211^{\dagger}_{\pm 0.002}$	$0.401^{\dagger}_{\pm 0.005}$	$0.226^{\dagger}_{\pm 0.001}$	$0.506^{\dagger}_{\pm 0.004}$
		local	$0.548^{\dagger}_{\pm 0.001}$	$0.605^{*}_{\pm 0.005}$	$0.214^{*}_{\pm 0.003}$	$0.411^{*}_{\pm 0.003}$	$0.221_{\pm 0.002}$	$0.515^{*}_{\pm 0.004}$
	Swin-T	pre	$0.531_{\pm 0.004}$	$0.605_{\pm 0.006}$	$0.205_{\pm 0.004}$	$0.401_{\pm 0.004}$	$0.231_{\pm 0.002}$	$\mathbf{0.535}_{\pm 0.004}$
		global	$0.559^{\dagger}_{\pm 0.002}$	$0.607^{\dagger}_{\pm 0.004}$	$0.218^{\dagger}_{\pm 0.001}$	$0.416^{\dagger}_{\pm 0.004}$	$0.236^{\dagger}_{\pm 0.001}$	$\mathbf{0.535}_{\pm 0.004}$
		local	$\mathbf{0.584}^{*}_{\pm 0.003}$	$\mathbf{0.611}^{*}_{\pm 0.008}$	$\mathbf{0.230}^{*}_{\pm 0.005}$	$\mathbf{0.418}^{*}_{\pm 0.007}$	$\mathbf{0.252}^{*}_{\pm 0.003}$	$\mathbf{0.535}_{\pm 0.003}$

Table 1: Quantitative results on WashingtonInter (in-domain) and PathwayBench test cities (zero-shot transfer), reported as mean ± std over five seeds. **pre**, **global**, and **local** denote supervised pretraining, RL fine-tuning with a terminal graph-level reward, and RL fine-tuning with additional local stepwise reward shaping, respectively. **Bold** indicates the best value within each dataset/metric column. † denotes an improvement over both baselines and the corresponding **pre** checkpoint (same backbone). * denotes an improvement over the corresponding **global** checkpoint (same backbone).

topologically valid but sparse and visually weak. These results suggest that released road-tracing pipelines cannot be directly applied to pedestrian pathways without substantial method-specific redesign.

5.2 (T2) RL Fine-Tuning Consistently Improves Performance Across Backbones and Datasets

Table 1 shows that RL fine-tuning with an outcome-level reward improves performance across backbones and datasets, despite the already strong TraversRL-Pre baseline. With the strict 1 m buffer, IoU becomes highly sensitive to geometric misalignment, making improvements more indicative of true precision.

1 Global reward improves graph quality. With a terminal graph-level reward, TraversRL-Global improves IoU for every backbone on every dataset over TraversRL-Pre. Averaged across backbones, the relative IoU gains are 5.1% on WashingtonInter, 5.6% on Seattle, and 5.2% on D.C. TravSim also increases by 0.6%, 3.0%, and 0.5%, respectively. These consistent trends indicate that optimizing a terminal objective aligns the policy toward decisions that improve both geometric overlap and connectivity patterns.

2 Local reward shaping adds further gains. Relative to TraversRL-Global, TraversRL-Local yields additional IoU gains of 1.6% (WashingtonInter), 4.1% (Seattle), and 2.1% (D.C.), with one minor exception (ViT-S on D.C.). Relative to TraversRL-Pre, these gains surge to 6.7%, 9.8%, and 7.5%. Local

Fig. 5: RL fine-tuning produces cleaner, sparser graphs. Qualitative examples comparing the ground-truth graph (top, blue) to predictions from TraversRL-Pre (middle, red, "messy") and TraversRL-Local (bottom, red, "cleaner"). Yellow dashed boxes highlight artifacts of the supervised policy — redundant branches, spurious short connectors, and noisy junction geometry —that are pruned after RL fine-tuning.

shaping also consistently improves TravSim, by 1.5% on WashingtonInter, 2.4% on Seattle, and 1.0% on D.C. These results support the intended mechanism of the local reward: finer credit assignment encourages locally aligned edge placement while preserving stable terminal-reward optimization.

Additionally, the standard deviations in Table 1 are generally small across five random seeds. In most settings, the RL gains in IoU are comparable to, or larger than, the seed-to-seed variation, indicating that the gains are not an artifact of a favorable seed but reflect a systematic improvement in graph quality.

3 TraversRL-Local recovers subtle missing paths. Recall that TravSim compares the set of traversable boundary-to-boundary pairs induced by the predicted vs. ground-truth graphs within each tile (Sec. 4.3). Let S_{pred} and S_{gt} denote these per-tile sets of traversable boundary-pair relations for the prediction and ground-truth, respectively. We decompose their disagreement into: $\text{TP} = |S_{\text{pred}} \cap S_{\text{gt}}|$, $\text{FP} = |S_{\text{pred}}| - \text{TP}$, and $\text{FN} = |S_{\text{gt}}| - \text{TP}$. On WashingtonInter, averaging per-tile set sizes over tiles and then over seeds/backbones, TraversRL-Pre predicts approximately 1.86 boundary-pair relations per tile, with $\text{TP} \approx 1.55$, $\text{FP} \approx 0.31$, and $\text{FN} \approx 1.09$ (i.e., missed connectivity relations dominate spurious ones by $\sim 3.5\times$). RL fine-tuning primarily reduces this missed-connectivity term: TraversRL-Local increases TP from 1.55 $\rightarrow$ 1.60 ($+3.5\%$) and reduces FN from 1.09 $\rightarrow$ 1.04 (-5.0%). Consistent with this, at the scene level TravSim improves in 59% of intersections; among those improved scenes, 92% exhibit increased TP and 91% exhibit reduced FN. Overall, RL improves TravSim by recovering missing ground-truth paths.

Dataset	AvgDeg↓			TravSim↑		
	Pre	**Global (Δ)**	**Local (Δ)**	**Pre**	**Global (Δ)**	**Local (Δ)**
WashingtonInter	0.429	0.416 (-3.0%)	0.407 (-5.1%)	0.595	0.599 (+0.6%)	0.607 (+2.0%)
Seattle (PB)	0.574	0.541 (-5.7%)	0.519 (-9.6%)	0.389	0.401 (+3.1%)	0.411 (+5.7%)
D.C. (PB)	0.553	0.528 (-4.5%)	0.520 (-6.0%)	0.513	0.516 (+0.6%)	0.521 (+1.6%)

Table 2: Graph cleanliness and connectivity agreement averaged over backbones and five seeds. AvgDeg denotes average node degree (lower indicates sparser graphs), and TravSim denotes traversability similarity (higher indicates closer agreement with ground-truth boundary connectivity). For Global and Local, we report the relative change compared to Pre on the same dataset.

5.3 (T3) TraversRL Produces Cleaner, Sparser Graphs While Preserving Connectivity.

Beyond buffered IoU, we analyze how RL changes graph structure. Following PathwayBench, we compute all graph statistics within each tile on the clipped predicted graph and then average over tiles. We report: (1) average node degree (AvgDeg) as a proxy for graph density/cleanliness (lower typically indicates fewer branches) and (2) TravSim as a proxy for whether the predicted graph preserves ground-truth connectivity. AvgDeg alone cannot distinguish clean pruning from missing edges, so we interpret it jointly with TravSim.

RL reduces graph density while improving connectivity agreement. A consistent pattern emerges across all datasets (Table 2): RL fine-tuning yields sparser graphs while maintaining or improving connectivity. Averaged across backbones, TraversRL-Global reduces AvgDeg relative to TraversRL-Pre by 3.0% on WashingtonInter, 5.7% on Seattle, and 4.5% on D.C., while increasing TravSim by 0.6%, 3.1%, and 0.6%, respectively. TraversRL-Local strengthens this trend, further reducing AvgDeg (5.1%, 9.6%, 6.0%) while improving TravSim (2.0%, 5.7%, 1.6%). Fig. 5 shows the same effect qualitatively: RL suppresses redundant edges and noisy branches while preserving intersection structure.

5.4 (T4) Buffer-Size Ablation: Stricter Buffers Reveal Larger Gains.

To study metric sensitivity, we re-evaluated predictions with wider buffers. On WashingtonInter dataset, averaged across backbones and seeds, TraversRL-Global improves IoU over TraversRL-Pre by 2.1% at 4m, which grows to 3.1% at 2m and 5.2% at 1m. The same pattern holds for TraversRL-Local, whose improvements increase by 2.5%, 4.0% and 6.8% respectively. These observations show that outcome-level RL fine-tuning improves geometric precision in a way that becomes increasingly visible as the overlap criterion tightens, with local shaping providing the strongest improvements under strict alignment. The effect is strongest for local shaping: relative to global-reward RL, TraversRL-Local gains widen as the buffer shrinks, consistent with the reward encouraging locally aligned edge extensions and discouraging off-target geometry.

5.5 (T5) Action-Space Ablation: Finer Actions Improve In-Domain Performance but Show Mixed Transfer.

We study sensitivity to action-space granularity by fixing the number of distance bins to 10 and varying the number of angle bins to 18 and 72. We run this ablation with ResNet-18 and TraversRL-Local. We observe that: (1) Reducing the action space to $|\mathcal{A}| = 181$ consistently hurts performance, lowering IoU/TravSim from 0.540/0.605 to 0.448/0.581 on WashingtonInter, from 0.199/0.403 to 0.174/0.385 on Seattle, and from 0.227/0.513 to 0.195/0.503 on D.C. (2) Increasing the action space to $|\mathcal{A}| = 721$ improves in-domain performance to 0.569/0.618, but shows mixed transfer: Seattle slightly drops to 0.198/0.383, while D.C. improves to 0.237/0.520. These results suggest that finer actions can reduce angular discretization error, but may also overfit on the source data and increase sensitivity to dataset-specific scale or noise. Overall, the default $|\mathcal{A}| = 361$ provides a reasonable tradeoff between precision and cross-domain robustness.

6 Conclusion

We introduced TraversRL, a vision-conditioned iterative generator for constructing pedestrian pathway graphs from aerial imagery. TraversRL represents state with a local image crop and a generation canvas encoding the partial graph, and predicts discretized direction–distance actions tailored to short pedestrian segments. Across backbones and datasets, RL fine-tuning consistently improves buffered IoU and connectivity agreement, while also simplifying graph structure, demonstrating effectiveness for long-horizon pedestrian network generation.

Our work also suggests several directions for improvement. (1) Scale variation. TraversRL uses a fixed crop size in normalized coordinates, so the effective physical scale varies across datasets and scene extents. This can obscure thin pathways and exacerbate domain shift. Scale-aware inputs could improve cross-city robustness. (2) Start-node initialization. For evaluation, rollouts are initialized from a start node on the ground-truth graph. In deployment, this information is unavailable, so a practical system will require a seeding mechanism, e.g., predicting likely start node from imagery from a segmentation prior. (3) Coverage of disconnected components. Some scenes contain multiple disconnected pathway components. A single-seed rollout can only recover the component reachable from that seed. Full coverage requires multi-seed inference and a strategy for proposing and filtering seeds in unexplored regions.

Ultimately, we envision an extended TraversRL model to "walk the city" and produce high-quality de novo pedestrian networks without human intervention.

Acknowledgements

We thank the Taskar Center for Accessible Technology for allowing us to derive the WashingtonInter dataset from the OS-CONNECT project.

References

1. Ahmetovic, D., Manduchi, R., Coughlan, J.M., Mascetti, S.: Zebra crossing spotter: Automatic population of spatial databases for increased safety of blind travelers. In: Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility. pp. 251–258 (2015)
2. Bastani, F., He, S., Abbar, S., Alizadeh, M., Balakrishnan, H., Chawla, S., Madden, S., DeWitt, D.: Roadtracer: Automatic extraction of road networks from aerial images. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4720–4728 (2018)
3. Bhuyan, Z., Xie, Y., Rith, A., Yan, X., Apostolov, N., Oke, J., Ai, C.: Crosswalknet: An optimized deep learning framework for pedestrian crosswalk detection in aerial images with high-performance computing. arXiv preprint arXiv:2506.07885 (2025)
4. Deng, L., Deng, Y., Meng, Y., Chen, J., Xi, Z., Liu, D., Chu, Q.: Gld-road: A global–local decoding road network extraction model for remote sensing images. ISPRS Journal of Photogrammetry and Remote Sensing **228**, 741–755 (2025)
5. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
6. Gao, J., Xu, S., Ye, W., Liu, W., He, C., Fu, W., Mei, Z., Wang, G., Wu, Y.: On designing effective rl reward at training time for llm reasoning. arXiv preprint arXiv:2410.15115 (2024)
7. Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., et al.: Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv preprint arXiv:2501.12948 (2025)
8. Hara, K., Sun, J., Moore, R., Jacobs, D., Froehlich, J.: Tohme: detecting curb ramps in google street view using crowdsourcing, computer vision, and machine learning. In: Proceedings of the 27th annual ACM symposium on User interface software and technology. pp. 189–204 (2014)
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
10. He, S., Bastani, F., Jagwani, S., Alizadeh, M., Balakrishnan, H., Chawla, S., Elshrif, M.M., Madden, S., Sadeghi, M.A.: Sat2graph: Road graph extraction through graph-tensor encoding. In: European Conference on Computer Vision. pp. 51–67. Springer (2020)
11. Hetang, C., Xue, H., Le, C., Yue, T., Wang, W., He, Y.: Segment anything model for road network graph extraction. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 2556–2566 (2024)
12. Hosseini, M., Sevtsuk, A., Miranda, F., Cesar Jr, R.M., Silva, C.T.: Mapping the walk: A scalable computer vision approach for generating sidewalk network datasets from aerial imagery. Computers, Environment and Urban Systems **101**, 101950 (2023)
13. Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Carney, A., et al.: Openai o1 system card. arXiv preprint arXiv:2412.16720 (2024)
14. Karimi, H.A., Kasemsuppakorn, P.: Pedestrian network map generation approaches and recommendation. International Journal of Geographical Information Science **27**(5), 947–962 (2013)

15. Liu, C., Jiang, Y., Zheng, N.: Netracer: A topology-aware iterative tracing approach for tubular structure extraction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 20593–20602 (2025)
16. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 10012–10022 (2021)
17. Lu, X., Zhong, Y., Zheng, Z., Zhang, L.: Gamsnet: Globally aware road detection network with multi-scale residual learning. ISPRS Journal of Photogrammetry and Remote Sensing **175**, 340–352 (2021)
18. Mi, L., Zhao, H., Nash, C., Jin, X., Gao, J., Sun, C., Schmid, C., Shavit, N., Chai, Y., Anguelov, D.: Hdmapgen: A hierarchical graph generative model of high definition maps. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4227–4236 (2021)
19. O'Meara, J.S., Hwang, J., Wang, Z., Saugstad, M., Froehlich, J.E.: Rampnet: A two-stage pipeline for bootstrapping curb ramp detection in streetscape images from open government metadata. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6656–6665 (2025)
20. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
21. Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al.: Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv. org/abs/2402.03300 **2**(3), 5 (2024)
22. Sun, T., Di, Z., Che, P., Liu, C., Wang, Y.: Leveraging crowdsourced gps data for road extraction from aerial imagery. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7509–7518 (2019)
23. Tan, Y.Q., Gao, S.H., Li, X.Y., Cheng, M.M., Ren, B.: Vecroad: Point-based iterative graph exploration for road graphs extraction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8910–8918 (2020)
24. Team, K., Du, A., Gao, B., Xing, B., Jiang, C., Chen, C., Li, C., Xiao, C., Du, C., Liao, C., et al.: Kimi k1. 5: Scaling reinforcement learning with llms. arXiv preprint arXiv:2501.12599 (2025)
25. Verma, R., Ukkusuri, S.V.: Crosswalk detection from satellite imagery for pedestrian network completion. Transportation research record **2678**(7), 845–856 (2024)
26. Wang, Y., Yang, Q., Zeng, Z., Ren, L., Liu, L., Peng, B., Cheng, H., He, X., Wang, K., Gao, J., et al.: Reinforcement learning for reasoning in large language models with one training example. arXiv preprint arXiv:2504.20571 (2025)
27. Wen, X., Liu, Z., Zheng, S., Ye, S., Wu, Z., Wang, Y., Xu, Z., Liang, X., Li, J., Miao, Z., et al.: Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. arXiv preprint arXiv:2506.14245 (2025)
28. Xu, Q., Long, C., Yu, L., Zhang, C.: Road extraction with satellite images and partial road maps. IEEE Transactions on Geoscience and Remote Sensing **61**, 1–14 (2023)
29. Xu, Z., Liu, Y., Gan, L., Sun, Y., Wu, X., Liu, M., Wang, L.: Rngdet: Road network graph detection by transformer in aerial images. IEEE Transactions on Geoscience and Remote Sensing **60**, 1–12 (2022)
30. Xu, Z., Liu, Y., Sun, Y., Liu, M., Wang, L.: Rngdet++: Road network graph detection by transformer with instance segmentation and multi-scale features enhancement. IEEE Robotics and Automation Letters **8**(5), 2991–2998 (2023)

31. Zhang, Y., Bolten, N., Mehta, S., Caspi, A.: Ape: An open and shared annotated dataset for learning urban pedestrian path networks. arXiv preprint arXiv:2303.02323 (2023)
32. Zhang, Y., Howe, B., Mehta, S., Bolten, N.J., Caspi, A.: Pathwaybench: Assessing routability of pedestrian pathway networks inferred from multi-city imagery. arXiv preprint arXiv:2407.16875 (2024)
33. Zhou, K., Xie, Y., Gao, Z., Miao, F., Zhang, L.: Funet: A novel road extraction network with fusion of location data and remote sensing imagery. ISPRS International Journal of Geo-Information **10**(1), 39 (2021)