Monocular-Camera-Based Autonomous Sidewalk Navigation Using an Efficient Machine-Learning-Driven Image Segmentation Approach

Lkhanaajav Mijiddorj[a], Tyler Beringer[a], Bilguunzaya Mijiddorj[a], Yan Yang[a], Alex N. Ho[a], Bin Xu[b] and Binbin Weng[a,*]

[a]Department of Electrical and Computer Engineering, University of Oklahoma, Norman, OK, USA
[b]Department of Aerospace and Mechanical Engineering, University of Oklahoma, Norman, OK, USA

ARTICLE INFO

Keywords:
autonomous micro-mobility
sidewalk navigation
monocular camera-based detection
semantic segmentation
bird's-eye view
skeletonization
embedded AI

ABSTRACT

Sidewalk-scale autonomy demands perception that runs reliably on compact, low-power hardware in cluttered, map-sparse environments. We propose a monocular segmentation–BEV–skeleton–path pipeline tailored for sidewalk-scale micro-mobility. A lightweight SegFormer-B0 model segments traversable sidewalk pixels from RGB video, and a fixed planar homography projects this mask into a metric Bird's-Eye View (BEV) frame. In BEV, mask refinement and main-component filtering suppress thin spurs and disconnected blobs before skeletonization. The cleaned mask is thinned using the Guo–Hall skeletonization algorithm, exposing the sidewalk centerline and junctions as a compact pixel graph. Pixel- and graph-level pruning remove short side branches, and a simple geometric cost function scores the remaining branches; the lowest-cost route is then smoothed into a continuous waypoint path. In this work, we evaluate the entire perception–planning stack offline on recorded video, with no onboard control executed. The teacher–student segmentation model is trained with 300 hand-labeled frames plus thousands of pseudo-labeled frames, using boundary-aware loss terms to sharpen mask edges. SegFormer-B0 runs in 46 ms per frame at 640×360 on CPU, and the full segmentation–BEV–skeleton–path pipeline runs in 109.5 ms per frame (≈ 9.1 FPS). We evaluate on campus video with curved sidewalks, T-junctions, shadows, and surface changes, reporting lateral path-centering error, Junction branch detection, and a mask–path alignment score (percentage of extracted path inside the sidewalk mask). In our experiments, the pipeline achieves a mean lateral path-centering error of 0.38 ± 0.66 m, chooses the correct branch at 95% of junctions, and maintains **98**% of the path inside the segmented sidewalk, indicating that a transparent segmentation–BEV–skeleton–path stack is feasible on embedded platforms.

1. Introduction

Autonomous micro-mobility is beginning to move from controlled demonstrations to everyday sidewalks. Delivery carts, assistive scooters and other pedestrian-speed robots must operate among pedestrians, signs, bicycles, vegetation, and irregular curb geometry, all under tight payload, power, and cost constraints. Sidewalks further complicate perception: widths vary, surfaces mix bricks and concrete, markings are inconsistent, and lighting changes rapidly under trees and buildings. High-definition pedestrian maps are scarce and GNSS is unreliable near canopies and urban canyons. In this setting, a vision-first navigation pipeline that is robust, interpretable, and efficient enough for single-board computers is essential.

We adopt a deliberately simple geometry-aware approach for local sidewalk navigation. A lightweight monocular segmentation model extracts the traversable sidewalk region, which is then projected into a bird's-eye view (BEV) via a fixed homography so that distances and projected horizons are expressed in meters rather than pixels. From this top-down mask, we build a skeleton-based representation that exposes the medial axis of the sidewalk and the branch structure at junctions, resulting in a compact graph of the forward paths of the candidates. Simple geometric metrics on this skeleton graph then support branch selection and centering, producing smooth candidate trajectories suitable for an embedded controller.

This direction complements several active research threads. End-to-end steering policies trained on rich sensors can perform well, but they are difficult to interpret and often assume GPU-class hardware [1, 13]. Dense monocular BEV reconstruction and diffusion-based free-space methods push accuracy but remain challenging to deploy on low-power platforms due to their memory and latency profiles [5, 17]. The following of classic semantic-mask corridors is highly efficient in structured environments, but does not explicitly expose the intersection topology and can struggle with irregular urban edges [10, 11]. In contrast, our stack is monocular, transparent, and explicitly topology-aware, while remaining light enough for CPU-only inference.

In this work, we focus on validating the perception and path-generation pipeline offline. Using video recorded from a forward-facing camera mounted on an electric scooter (Fig. 1), we apply the full segmentation–BEV–skeleton stack frame by frame and evaluate the resulting centerlines and branch choices without closing the loop on the vehicle. This isolates perception and topology reasoning, enabling detailed analysis of lateral alignment, junction detection, robustness to scene variation, and embedded efficiency.

Contributions.

*Corresponding author
*Email: binbin.weng@ou.edu
ORCID(s): 0000-0002-5706-5345 (B. Weng)

(a) Scooter platform with mounted forward-facing camera.

(b) Example first-person view from the scooter-mounted camera.

Figure 1: Hardware context for this study. (a) Electric scooter platform used for data collection, with a forward-facing monocular camera rigidly mounted to the handlebar stem. (b) Sample sidewalk scene as seen by the camera; all experiments are conducted offline on video recorded from this rig.

- A monocular segmentation → BEV → skeleton-graph pipeline for sidewalk navigation that is transparent, lightweight, and designed for embedded hardware.
- An interpretable branch-selection and centering strategy that produces smooth, well-aligned path candidates from noisy real-world segmentation masks.
- A training and deployment recipe for a compact SegFormer-B0 model (hybrid labels with boundary refinements, resolution/stride sweeps, and quantization-aware export) that supports CPU-only inference at practical frame rates.
- A comprehensive offline evaluation protocol for scooter-mounted sidewalk video, covering path-tracking quality (lateral path-centering error), Junction branch detection, robustness across lighting/geometry/occlusion conditions, and runtime/ablation analysis.

Organization. We next review related work, then describe the proposed pipeline (segmentation, BEV mapping, skeleton/graph construction, and path selection), define the evaluation metrics, present the real-world data collection and offline evaluation setup, report results and ablations, and conclude with limitations and directions for on-scooter closed-loop deployment.

2. Related Work

We organize prior work into six threads and emphasize what each direction offers in the real world—what runs fast, what transfers, and what stays explainable to the teams who deploy it.

Target-driven and end-to-end navigation. Target-driven systems in structured indoor settings combine monocular depth and segmentation to follow objects without explicit mapping, and they often work impressively well in corridors and rooms [9, 18]. For sidewalks, end-to-end policies trained on RGB-D (sometimes with GNSS) can steer directly from raw streams and reach real-time control [1, 13]. The upside is simplicity at runtime: one model, one policy. The downside is familiar to practitioners—limited interpretability when lighting shifts, pedestrians occlude edges, or layouts differ from training. Diagnosis becomes guesswork, and many solutions quietly assume GPU-class compute. Our pipeline trades some expressivity for a modular stack that can be inspected stage by stage on embedded hardware.

Dense monocular bird's-eye reconstruction. Monocular BEV models that lift features or regress dense warps produce striking top-down reconstructions and are a natural fit for planning [17]. On modern GPUs, they shine. On single-board computers, memory and latency are still a stretch. A single, calibrated homography is less glamorous but predictable and cheap; for near-planar sidewalk patches and short look-ahead, it delivers a stable metric frame without the overhead of full 3D.

Segmentation for drivable/free space. Perception has seen two dominant pushes: accuracy via ensembles and efficiency via compact backbones. Ensemble approaches can drive mIoU higher with multiple heads/backbones, but they concentrate on perception alone and increase compute [12].

Lightweight networks (e.g., MobileNet- and transformer-based designs like SegFormer-B0) bring latency and memory down to embedded levels [2, 6, 15]. In both cases, the mask is often the end of the story. Our approach carries that mask into geometry (BEV) and topology (skeleton graph) and then closes the loop with a small, readable controller. The goal is not only accuracy, but a path from pixels to motion that field teams can tune.

Generative free-space priors and diffusion. Diffusion-based free-space predictors are exciting because they can hallucinate plausible corridors when the signal is weak [5]. For bench science, this is promising; for an embedded scooter, the footprint and tuning complexity are still a mismatch. In the near term, a small number of moving parts—each easy to monitor and fail gracefully—tends to win outdoors. We therefore keep generation out of the critical path and prefer deterministic geometry that behaves consistently as conditions change.

Corridor following and agricultural rows. The agricultural community has a long history of segmentation-driven navigation in row crops, where the "gap between rows" is a crisp cue [10, 11]. Histogram-of-columns minima are fast and surprisingly reliable when the world really is a corridor. Sidewalks inherit the corridor idea but add T- and 4-way junctions, driveways, curb ramps, and uneven edges. In our experience, a pure histogram lacks a notion of topology and struggles at junctions. We borrow the efficiency of the idea but make branch structure explicit via skeletons.

Skeletons and mid-level geometry. Learned skeletonization and graph extraction are gaining traction and can substantially reduce planning cost in simulation [3]. Classic thinning methods (e.g., Zhang–Suen [16] and Guo–Hall [4]) remain attractive in embedded contexts because they are stable, fast, and easy to reason about. In this work, we adopt classical thinning to expose sidewalk centerlines and junctions, construct a small graph, and choose among a handful of forward branches using an interpretable cost. The intent is practical: operators can see which branch was chosen and why, and they can adjust the weights without retraining a model.

Positioning our work. Across these threads, we see a gap for a *transparent, geometry-aware, monocular* stack that fits sidewalk variability and embedded budgets. Our contribution is to stitch together pieces that are individually simple—compact segmentation, planar BEV, skeleton-graph reasoning, and a centering controller—into a single deployable pipeline. It is not the most expressive model in isolation; it is a system that behaves predictably, degrades gracefully, and can be profiled and fixed on the curb when something changes.

3. Methodology

This work proposes a vision-based navigation pipeline for sidewalk-following micro-mobility platforms using only monocular RGB input. The system avoids reliance on GPS or external localization by leveraging a modular pipeline composed of interpretable geometric and learning-based modules. As illustrated in Fig. 2, the proposed architecture maps camera frames to controller-ready waypoints through six primary stages: (i) semantic segmentation, (ii) BEV projection, (iii) skeletonization, (iv) path scoring, (v) waypoint smoothing, and (vi) optional re-projection for visualization.

3.1. Segmentation Module and Supervision Strategy

The first stage of our navigation pipeline (Fig. 2) performs pixel-wise segmentation of sidewalk regions from monocular RGB input. This module is critical for all downstream steps—homography projection, BEV graph construction, and trajectory planning—and must balance high segmentation quality with real-time efficiency on embedded systems.

We adopt **SegFormer** [15] as our backbone segmentation architecture due to its strong performance on urban navigation benchmarks and its adaptability across model scales. Unlike conventional encoder–decoder networks, SegFormer combines a lightweight Transformer-based encoder with a multi-scale MLP decoder, achieving competitive accuracy without relying on positional encodings or complex upsampling schemes.

For deployment, we use **SegFormer-B0** (3.7M parameters) on the scooter platform due to its low latency and memory footprint. However, small models trained on limited data often produce noisy or incomplete masks. To mitigate this, we employ a **semi-supervised teacher–student framework** that leverages both high-quality human annotations and dense pseudo-labels.

Teacher–Student Supervision We fine-tune a high-capacity **SegFormer-B2** teacher model (24M parameters) on 300 hand-labeled sidewalk frames. Once trained, this model generates dense soft masks on 2,300 additional unlabeled frames. For each unlabeled image $x \in D_U$, the teacher predicts pixel-wise logits $z_T = f_T(x)$, which are converted to probabilities via softmax: $p_T = \sigma(z_T)$. A confidence threshold τ is applied to produce a binary mask:

$$\tilde{y}(u) = \begin{cases} 1, & \text{if } p_T(u) \geq \tau \\ 0, & \text{otherwise} \end{cases} \quad \text{with } u \in \Omega$$

where Ω denotes the set of pixel locations. The pseudo-labels are further cleaned with connected-component filtering and morphological smoothing

Hybrid Training Dataset To construct a training set $\mathcal{D} = D_L \cup D_P$, we merge the pseudo-labeled frames D_P with our hand-labeled set D_L. In frames where both are available,

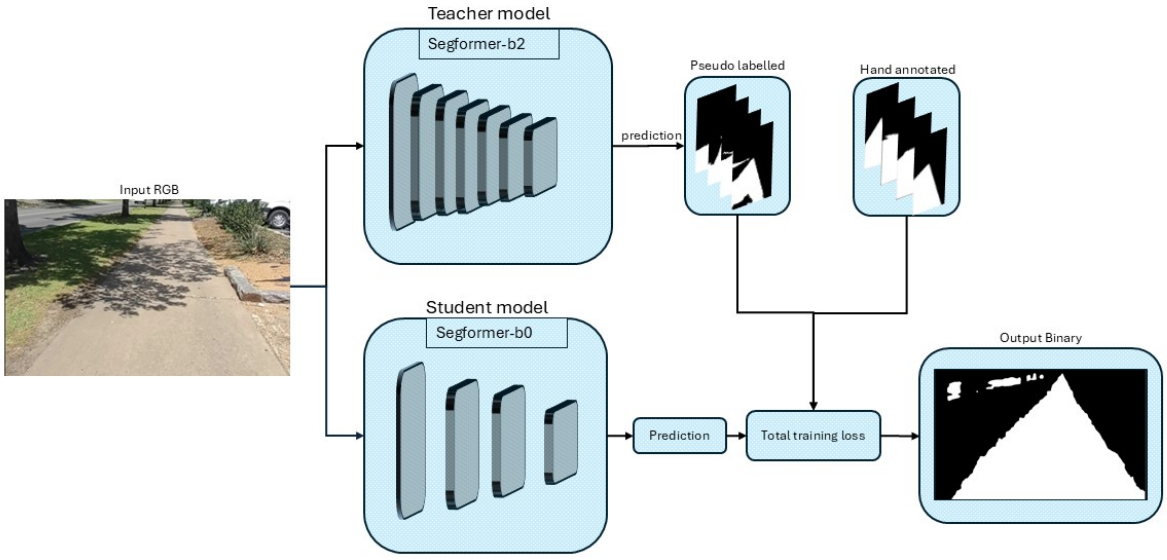


Figure 2: Overview of the sidewalk navigation pipeline. Hand-annotated and pseudo-labeled data are merged to train a student segmentation model, followed by geometric processing to extract traversable paths.

hand labels take precedence:

$$y_i = \begin{cases} y_i^{\text{hand}}, & \text{if labeled} \\ \tilde{y}_i^{\text{pseudo}}, & \text{otherwise} \end{cases}$$

Loss Function The student model f_S is trained to minimize a composite loss:

$$\mathcal{L} = \lambda_{\text{CE}}\mathcal{L}_{\text{CE}} + \lambda_{\text{Dice}}\mathcal{L}_{\text{Dice}} + \lambda_{\text{B}}\mathcal{L}_{\text{boundary}}$$

where $\mathcal{L}_{\text{CE}}$ is binary cross-entropy, $\mathcal{L}_{\text{Dice}}$ improves mask completeness in sparse regions, and $\mathcal{L}_{\text{boundary}}$ sharpens the sidewalk edges. The result is a crisp, real-time binary mask $\widehat{M}_{\text{img}} \in \{0,1\}^{H\times W}$ identifying traversable sidewalk regions.

3.2. Resolution Trade-Off Analysis

To balance segmentation accuracy and real-time performance, we conducted a resolution sweep experiment using the SegFormer-B0 model on CPU. We evaluated multiple input sizes, ranging from 160×90 to 1280×720, and measured both segmentation latency and visual quality.

Figure 3 shows representative mask outputs at each resolution. Lower resolutions (e.g., 320×180 or below) achieved higher frame rates but produced masks that were overly coarse—frequently omitting curb boundaries or fragmenting continuous sidewalk segments. Conversely, higher resolutions offered slightly sharper outputs but incurred prohibitively high latency unsuitable for embedded deployment.

Table 1
SegFormer-B0 inference latency and throughput at various input resolutions (CPU-only).

Resolution	Latency (ms)	FPS (1000/ms)
native 540×540	72.80	13.74
960×540	152.45	6.56
1280×720	352.40	2.84
768×432	93.55	10.69
640×360	46.00	21.74
480×270	42.05	23.78
320×180	25.35	39.45
160×90	19.84	50.40

Table 1 reports the average latency and throughput (FPS) for each resolution. Based on both qualitative and quantitative results, we selected 640×360 as a practical operating point. At this resolution, the model produces accurate and consistent segmentation while maintaining sub-50 ms inference time on CPU. This balance allows the downstream BEV projection and skeleton graph modules to operate on clean, connected masks without introducing runtime bottlenecks.

3.3. Bird's-Eye View Projection

To transform the segmented sidewalk mask from the image plane into a spatially consistent top-down representation, we apply a single planar homography. This projection produces a Bird's-Eye View (BEV) binary mask in which

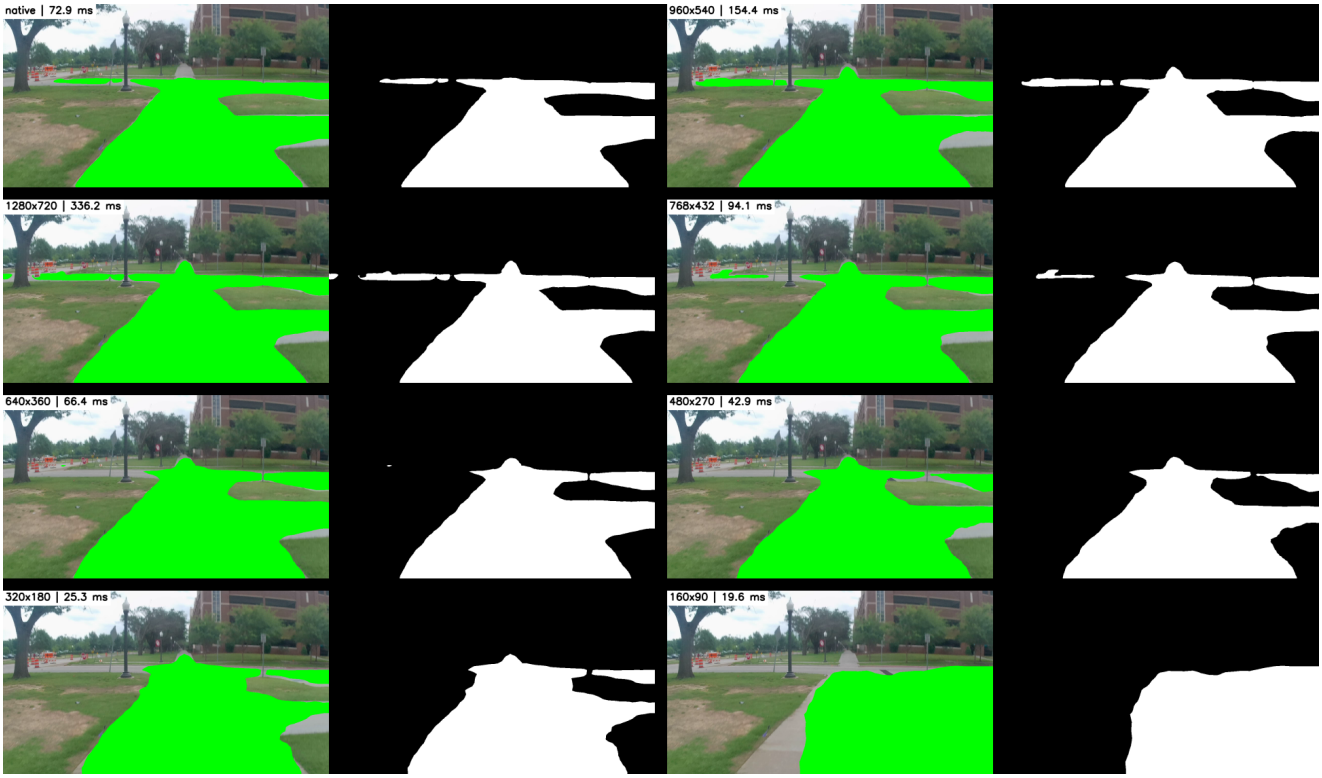


Figure 3: Resolution sweep for SegFormer-B0. Each column shows input overlay (left) and resulting segmentation mask (right) at different input resolutions. Higher resolutions improve edge detail but increase latency.

pixel distances approximate real-world ground distances, thereby simplifying geometric reasoning and path planning.

Let $p = [u, v, 1]^\top$ be a homogeneous image pixel coordinate and $p' = [x, y, 1]^\top$ be its corresponding BEV coordinate on the ground plane. The mapping is defined by a 3×3 homography matrix H:

$$p' \sim Hp \quad \text{or} \quad \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim H \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

In practice, H is estimated via manual selection of four point correspondences between the camera image and a flat ground reference, assuming a fixed camera pose and locally planar sidewalk. For inverse projection (e.g., BEV-to-image reprojection), we apply H^{-1} followed by dehomogenization.

The input binary mask $\widehat{M}_{\text{img}} \in \{0, 1\}^{H \times W}$ is warped to a BEV mask $\widehat{M}_{\text{bev}} \in \{0, 1\}^{H_b \times W_b}$ using backward warping:

$$\widehat{M}_{\text{bev}}(x, y) = \widehat{M}_{\text{img}} \left(\pi(H^{-1}[x, y, 1]^\top) \right)$$

where $\pi([x, y, z]^\top) = (x/z,\ y/z)$ denotes the dehomogenization operator.

The resulting $\widehat{M}_{\text{bev}}$ is a normalized binary occupancy map aligned with the robot's frame of reference. We typically center the robot at the bottom-middle of the BEV grid, with forward motion aligned to the vertical axis. This transformation ensures consistency across frames and removes perspective distortions from the original camera image.

One limitation of this method is the assumption of a flat plane and fixed camera pose. In practice, small deviations (e.g., bumps or ramps) cause minor distortions but do not critically affect downstream processing. Additional robustness can be achieved by periodically re-calibrating H or estimating local ground planes from stereo or depth if available.

3.4. Skeletonization and Graph Abstraction

To convert the binary BEV mask into a navigable geometric representation, we extract the medial axis of the sidewalk region using skeletonization. This process reduces extended pixel blobs into a minimal-width structure that preserves the shape and connectivity of traversable paths.

We use the Guo–Hall parallel thinning algorithm [4], a widely used two-subiteration approach that erodes foreground pixels while maintaining topological structure. Applied to the BEV mask $\widehat{M}_{\text{bev}} \in \{0, 1\}^{H_b \times W_b}$, it produces a skeleton mask S where each pixel marks the centerline of a connected sidewalk component.

From this skeleton, we construct an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$:

- Each nonzero skeleton pixel becomes a node $v_i \in \mathcal{V}$ with BEV coordinates (x_i, y_i).
- Edges $(v_i, v_j) \in \mathcal{E}$ connect 8-neighbor pixels, preserving local connectivity.

This abstraction exposes key geometric features, such as path continuity and branch points, while remaining invariant to sidewalk width.

To suppress spurious branches caused by mask noise or boundary fuzziness, we apply a two-stage pruning process:

- **Length-based pruning** removes paths shorter than a set threshold (e.g., 1.0 m in BEV space).
- **Component filtering** discards isolated subgraphs disconnected from the main sidewalk trunk.

Compared to contour tracing or blob extraction, skeletonization provides a compact, interpretable representation ideal for topological reasoning and path generation in unstructured sidewalk environments.

3.5. Path Enumeration and Scoring

Once the skeleton graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is constructed, we generate candidate navigation trajectories by traversing paths from the agent's origin node. We assume the robot's current position corresponds to the bottom-center of the BEV frame, which maps to a starting node $v_0 \in \mathcal{V}$.

We use Dijkstra's algorithm to enumerate all feasible paths from v_0 to reachable endpoints in the graph, subject to a maximum path length constraint (e.g., 5–7 m in BEV space). Each candidate path p_j is a sequence of connected nodes

$$p_j = \{v_0, v_1, \ldots, v_n\}.$$

Scoring. Each path is evaluated using a simple geometric cost function that prefers smooth, forward-looking branches and penalizes large sideways excursions:

$$C(p_j) = \alpha \cdot \text{Curvature}(p_j) + \beta \cdot \text{LateralShift}(p_j),$$

where

- Curvature(p_j) measures the accumulated change in heading along the path, and
- LateralShift(p_j) penalizes paths whose nodes deviate far from the forward (vertical) image axis in BEV.

The weights α and β control the trade-off between smoothness and staying roughly centered in the visible corridor. We do not explicitly classify or prune "road" branches; instead, branches that require strong turning or large lateral shifts naturally receive higher cost and are seldom selected.

Output. The resulting set of feasible paths is typically small (2–4 branches per frame) and forms the input to the next stage: smoothing and waypoint generation. Among remaining candidates, the one with the lowest score is selected as the primary forward trajectory. In ambiguous settings (e.g., intersections), this scoring encourages natural continuation of the current direction while still keeping alternative branches available if required.

This graph-based enumeration strategy ensures that trajectories remain grounded in the underlying scene geometry and adapts gracefully to T-junctions, curves, and open plazas.

3.6. Evaluation Metrics

To quantify the quality of the generated paths, we evaluate the pipeline offline on video recorded from the scooter platform and define three metrics: lateral path-centering error, Junction branch detection, and mask–path alignment. All metrics are computed frame-by-frame (or at detected junctions) using only the segmentation, BEV, and skeleton outputs; no closed-loop control is executed on the scooter in this study.

Lateral path-centering error. In BEV space, we first derive a geometric reference centerline from the binary sidewalk mask by taking, for each row y, the midpoint between the left and right sidewalk boundaries at that row. For each sample point (x, y) along the selected skeleton path, we measure the lateral deviation $|x - x_{\text{ref}}(y)|$ from this centerline. We report the mean and standard deviation of this deviation over all evaluated frames. The deviations are first computed in BEV pixels and then converted to meters using the known BEV scale when reporting numerical results. This metric captures how well the chosen path stays near the middle of the traversable sidewalk region implied by the segmentation.

Junction branch detection. At skeleton branch points corresponding to T- and 4-way intersections, we would like the graph to expose all physically available sidewalk branches (e.g., left, straight, right). For each junction j, we annotate the number of true branches $B_{\text{gt}}^{(j)}$ visible in the scene and count the number of corresponding branches $B_{\text{det}}^{(j)}$ present in the skeleton graph. We then define a junction branch detection rate as

$$\text{BranchRecall} = \frac{\sum_j B_{\text{det}}^{(j)}}{\sum_j B_{\text{gt}}^{(j)}} \times 100\%.$$

This metric measures how completely the skeleton-and-graph stage recovers the available sidewalk options at real-world intersections, independent of which branch is ultimately selected for navigation.

Mask–path alignment. To assess whether the selected skeleton path remains inside the traversable sidewalk region, we compute a mask–path alignment score. For each frame, we count the number of skeleton pixels along the selected path that lie inside the binary sidewalk mask, L_{in}, and the total number of skeleton pixels in that path, L_{total}. The per-frame alignment ratio is

$$a_i = \frac{L_{\text{in}}}{L_{\text{total}}}.$$

We then report the average alignment across all frames with a valid path:

$$\text{Alignment} = \frac{1}{N} \sum_{i=1}^{N} a_i \times 100\%.$$

(a) RGB input (b) Raw segmentation (c) BEV Segmentation

Figure 4: Segmentation pipeline. (a) RGB input frame, (b) raw sidewalk segmentation in the camera view, and (c) BEV projection of the cleaned sidewalk mask. Morphological filtering and connected-component analysis remove small spurs and disconnected blobs to stabilize the mask before skeletonization.

Higher values indicate that the path stays well within the segmented sidewalk area, even in the presence of small segmentation errors or partial occlusions.

Dynamic obstacles. The current pipeline does not explicitly detect or track dynamic obstacles such as pedestrians or cyclists. Instead, such objects appear as non-traversable regions in the segmentation mask, and the skeleton is extracted only from the remaining sidewalk pixels. In practice, this means that moving obstacles create temporary gaps or detours in the mask, and the resulting skeleton path implicitly routes around them when possible. A full treatment of dynamic agent behavior is left for future work.

4. Results

We present qualitative results and runtime analysis of the proposed sidewalk navigation pipeline, highlighting each core component—from initial perception to trajectory generation. Our evaluation is conducted on urban campus environments collected at the University of Oklahoma, which feature diverse sidewalk layouts including intersections, curves, tree occlusions, and varying surface textures. These real-world conditions test the system's robustness to lighting changes, shadows, clutter, and geometric irregularities without reliance on high-precision localization.

To ensure clarity, we organize the results by pipeline stage: segmentation, bird's-eye view (BEV) projection, skeleton graph abstraction, and control path generation. For each stage, we provide representative visualizations that illustrate the role of each transformation in narrowing raw input to a control-ready trajectory. Finally, we include a detailed runtime breakdown to demonstrate the feasibility of deploying the full pipeline on constrained CPU hardware at interactive frame rates.

4.1. Segmentation Output

The first stage of the pipeline employs SegFormer-B0 to perform semantic segmentation of the RGB image, identifying traversable sidewalk regions at a resolution of 640×360. This compact transformer-based model was selected for its excellent speed–accuracy tradeoff on embedded hardware. Figure 4 illustrates a typical output: the left panel shows the raw RGB input, the middle panel shows the binary sidewalk mask directly predicted by the model in the image plane, and the right panel shows the same mask after projection into BEV coordinates.

The raw segmentation can contain small noise artifacts and contour irregularities—especially near texture edges, curbs, and occlusions. To address this, we apply morphological operations (closing, opening) and connected-component filtering to stabilize boundaries and remove spurious fragments. Although not explicitly visualized in Figure 4, this refinement stage is critical to prevent fragmentation in the later skeleton graph and to ensure spatial continuity in narrow sidewalk corridors.

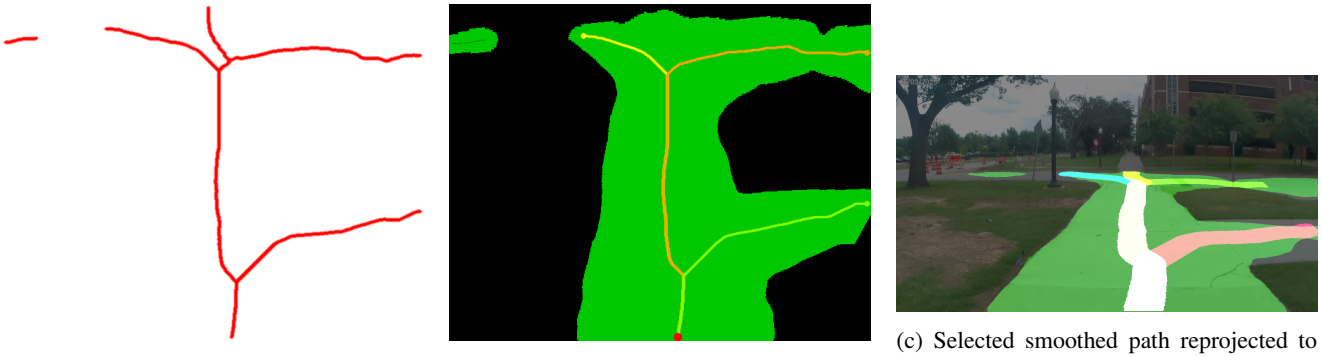
The resulting refined mask is compact and robust across diverse conditions including shadows, texture seams, and uneven surfaces, forming a stable basis for subsequent geometric projection and topological reasoning.

4.2. Bird's-Eye View Projection

Following segmentation, the binary mask is transformed into a top-down bird's-eye view (BEV) using a fixed planar homography. This transformation reprojects pixels from the image plane into a metric-aligned space where distances, angles, and paths can be interpreted geometrically. The homography matrix is calibrated offline using known correspondences between image points and sidewalk plane coordinates.

As shown in Figure 4, the raw sidewalk mask predicted in the image plane (middle) is warped into BEV space (right), producing a rectified top-down view of the sidewalk layout. This spatial normalization simplifies path planning and allows skeletonization to proceed in a consistent coordinate frame. In practice, we apply the same homography to the refined (morphologically filtered) mask used for downstream processing.

Overall, BEV projection provides a stable canvas for interpreting topology, handling curves and junctions without



(a) Skeletonized BEV mask. (b) Candidate paths on the skeleton graph. (c) Selected smoothed path reprojected to the camera view.

Figure 5: Skeleton and path graph. (a) Skeletonized BEV sidewalk mask after Guo–Hall thinning, (b) candidate paths on the skeleton graph computed via Dijkstra's search, and (c) final smoothed path reprojected into the original camera view. The overlay in (c) confirms that the selected trajectory remains centered within the sidewalk and aligned with the visible scene geometry; in each BEV panel, the bottom of the image corresponds to the scooter's current position.

introducing spatial distortion that would otherwise challenge path extraction in image space.

4.3. Skeletonization and Path Graph

Once the BEV mask is computed, we apply the Guo–Hall parallel thinning algorithm (Section 3.4) to extract the medial axis of the sidewalk region. This produces a 1-pixel-wide skeleton that preserves the overall topology while significantly reducing data complexity. The skeleton serves as a geometric scaffold from which connectivity and directionality can be inferred.

As shown in Figure 5, the resulting graph captures the underlying layout of the sidewalk, including curved segments and intersections. Each skeleton pixel is treated as a node, and edges are formed based on 8-neighbor connectivity. Short branches and isolated artifacts are pruned to stabilize the graph.

We use Dijkstra's algorithm to enumerate candidate paths from the scooter's position (defined as the bottom center of the BEV frame) to reachable endpoints. These paths are later scored based on geometric constraints, including heading continuity and lateral shift, to select the most appropriate trajectory for control. The skeleton thus bridges raw perception with structured planning by transforming dense pixel-wise outputs into a sparse, interpretable graph.

4.4. Camera Reprojection

To visually verify consistency between the planned trajectory and the underlying scene geometry, the final selected path in BEV space is reprojected back into the original camera frame using the inverse homography. The resulting polyline is overlaid on the RGB image, as illustrated in Figure 5(c).

This overlay is not used for control, but it provides an intuitive check that the path remains centered within the sidewalk, bends appropriately around corners, and respects junction structure as seen by the camera. In practice, we found this view especially useful for diagnosing failures (e.g., when segmentation leaks onto grass or misses a curb) and for qualitatively validating that the skeleton-derived path aligns with what a human driver would consider a reasonable route.

4.5. Quantitative Path Quality Metrics

Using the metrics defined in Section 3.6, we quantitatively evaluate how well the pipeline centers its paths within the sidewalk, chooses branches at junctions, and keeps the selected path inside the segmented mask. All numbers reported here are computed offline on held-out campus video sequences that were not used for training the teacher or student segmentation models.

Junction branch detection. We next evaluate how completely the skeleton graph recovers available sidewalk options at intersections. For each T- and 4-way junction in the test videos, we annotate the number of physically present sidewalk branches (e.g., left, straight, right) and compare this with the number of corresponding branches exposed by the skeleton graph. Across all evaluated junctions, the branch detection rate is **95%**, indicating that in most cases the graph recovers nearly all of the navigable options at each intersection.

Mask–path alignment. We also assess how much of the selected path remains inside the segmented sidewalk area. For each frame, we compute the fraction of skeleton pixels along the chosen path that fall within the binary sidewalk mask and then average this ratio across all frames. The resulting average mask–path alignment is **98%**, indicating that the path is consistently supported by the underlying segmentation and rarely drifts into regions labeled as non-traversable.

Across the held-out campus sequences, the lateral path-centering error is 0.38 ± 0.66 m, with a 95th-percentile error of 2.01 m (Table 2). This indicates that, in most frames, the selected path remains within roughly one sidewalk-width of the geometric centerline, even under real-world noise and segmentation imperfections.

Table 2
Summary of path quality and runtime on held-out campus sequences.

Metric	Value
Mean lateral path-centering error [m]	0.38 ± 0.66
95th percentile lateral path-centering error	2.01
Junction branch detection [%]	95
Mask–path alignment [%]	98
Pipeline rate (no viz) [FPS]	9.1

Table 3
Per-module runtime at 640×360 resolution (CPU-only).

Module	Mean Time (ms)
SegFormer Inference	46.0
Mask Refinement	8.5
BEV Projection	0.9
Skeleton & Path Extraction	54.1
Total (no visualization)	109.5
Visualization (optional)	15.3

4.6. Failure Modes and Limitations

While the system performs robustly on most campus paths, we observed occasional failures in visually ambiguous scenes. The most common failure cases include:

- **Wide intersections or plazas:** At large open junctions with multiple paths and weak edge cues, the segmentation may fail to resolve sidewalk continuation, leading to incorrect branch selection.
- **Partial occlusions:** Pedestrians or parked objects occasionally block part of the sidewalk, causing the skeleton to veer toward an edge or temporarily break.
- **Junction ambiguity:** In a few scenes, the desired "natural" direction is difficult to define, and human labelers disagreed on the correct path.
- **Edge bleeding in segmentation:** The student model, while efficient, can sometimes overshoot the sidewalk boundary in bright or low-contrast regions.

These cases contribute to the small number of errors reported in junction accuracy (5%) and lateral centering outliers (e.g., 2.01 m at the 95th percentile). Improving robustness under occlusion and at complex intersections is a target for future work.

4.7. System Runtime

Table 3 summarizes the average per-module runtime measured on a CPU-only system using 640×360 input resolution. Summing SegFormer inference, mask refinement, BEV projection, and skeleton/path extraction yields a total of 109.5 ms per frame, corresponding to approximately 9.1 FPS. Including visualization increases the per-frame time to 124.8 ms (about 8.0 FPS), but visualization is not required in a deployed setting. During profiling, the complete implementation (SegFormer-B0 weights plus BEV and skeleton buffers) used approximately 900 MB of system RAM at this resolution, which fits comfortably within the 4–8 GB memory budgets of typical embedded CPU platforms.

At these rates, the perception–to–path pipeline runs significantly faster than typical human visual reaction times, which are on the order of 180–250 ms for simple visual stimuli.[8, 14] In other words, the system can update its local path roughly twice during the time a human would take to perceive a change and initiate a steering correction.

The achieved 8–9 FPS update frequency is also consistent with many camera-based mobile robot controllers, which often operate in the 2–10 FPS range when closing the loop on visual feedback.[7] For a pedestrian-speed scooter, this corresponds to path updates every few tens of centimeters of travel, which is sufficient for negotiating curves and sidewalk intersections while remaining compatible with modest embedded CPU and memory budgets.

5. Conclusion

We presented a modular vision-based sidewalk navigation pipeline designed for compact, embedded platforms. The system integrates monocular semantic segmentation, bird's-eye view projection via a calibrated homography, skeleton-based graph extraction, and lightweight path selection to generate control-ready trajectories for a scooter platform. By combining hand-labeled and pseudo-labeled data in a teacher–student setup, we trained a compact SegFormer-B0 model that achieves robust sidewalk segmentation under diverse campus lighting and geometry while remaining suitable for CPU-only inference.

Our experiments on real sidewalk video collected at the University of Oklahoma demonstrate that this decomposition from pixels to skeleton graph yields interpretable intermediate representations and reliable paths. The pipeline runs at approximately 9 FPS at 640×360 resolution on CPU-only hardware, selects the correct branch at T- and 4-way junctions in 95% of evaluated cases, and maintains an average mask–path alignment of 98%. These results indicate that the system can produce smooth, well-centered trajectories that remain consistent with the underlying segmentation, while exposing topology explicitly enough to diagnose failures and tune behavior in the field.

Several limitations remain. The current formulation assumes a largely planar ground and static obstacles, and all evaluations are conducted offline without closing the loop on the physical scooter. Dynamic agents such as pedestrians and cyclists are not modeled explicitly, and the pipeline processes each frame independently without exploiting temporal consistency across video. The homography is fixed for a single camera pose, which may limit transfer to different mounting configurations without recalibration.

Future work will extend the system along three directions. First, we plan to integrate dynamic obstacle detection and short-horizon prediction into the BEV and skeleton layers, enabling path selection that explicitly reasons about moving agents. Second, we aim to incorporate temporal cues

through video-based segmentation and trajectory smoothing to improve robustness in challenging conditions such as occlusions and transient segmentation errors. Third, we will deploy the full stack in closed-loop experiments on the scooter platform, study failure modes at scale, and explore coupling the local skeleton-based planner with higher-level global navigation and mapping modules.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Declaration of generative AI and AI-assisted technologies in the manuscript preparation process

During the preparation of this work, the authors used ChatGPT (OpenAI) to assist with language editing, organization, and consistency checks. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of this paper.

Data availability

Original code and configuration files for the segmentation–BEV–skeleton–path pipeline are available in the ScooterProject repository at https://github.com/Lkhanaajav/ScooterProject. The raw campus video recordings contain identifiable pedestrians and cannot be shared publicly due to privacy considerations. Aggregated evaluation logs and example anonymized frames are available from the corresponding author on reasonable request.

CRediT authorship contribution statement

L. Mijiddorj: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Visualization, Writing – original draft, Writing – review & editing. B. Mijiddorj: Conceptualization, Investigation, Writing – review & editing. Y. Yang: Conceptualization, Data curation, Writing – review & editing. A. Ho: Conceptualization, Writing – review & editing. T. Beringer: Conceptualization, Investigation, Writing – review & editing. B. Xu: Conceptualization, Supervision, Writing – review & editing. B. Weng: Conceptualization, Supervision, Project administration, Funding acquisition, Resources, Writing – review & editing.

References

[1] Bojarski, M., Del Testa, D., Dworakowski, D., et al., 2016. End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316 NVIDIA paper.

[2] Che, Q.H., Nguyen, D.P., Pham, M.Q., Lam, D.K., 2023. Twinlitenet: An efficient and lightweight model for driveable area and lane segmentation in self-driving cars, in: International Conference on Multimedia Analysis and Pattern Recognition (MAPR). ArXiv:2307.10705.

[3] Flores-Aquino, G.O., Gutierrez-Frias, O., Vasquez-Gomez, J.I., 2025. Path planning using a one-shot-sampling skeleton map. arXiv preprint arXiv:2507.02328 doi:10.48550/arXiv.2507.02328.

[4] Guo, Z., Hall, R.W., 1989. Parallel thinning with two-subiteration algorithms. Communications of the ACM 32, 359–373. doi:10.1145/62065.62074.

[5] Gupta, K., Stanley, T.S., Paul, P., Singh, A.K., Krishna, K.M., 2025. Diffusion-FS: Multimodal free-space prediction via diffusion for autonomous driving, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). ArXiv:2507.18763.

[6] Howard, A., Sandler, M., Chu, G., Chen, L., et al., 2019. Searching for mobilenetv3, in: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).

[7] Huang, W., et al., 2025. π_0: A Vision-Language-Action Flow Model for Robot Control. arXiv preprint. URL: https://www.physicalintelligence.company/download/pi0.pdf. reports typical camera-based control frequencies of 2–10 Hz.

[8] Kosinski, R.J., 2013. Reaction times to sound, light and touch. URL: https://bionumbers.hms.harvard.edu/bionumber.aspx?id=110800. bioNumbers ID 110800, visual reaction times 180–200 ms.

[9] Machkour, Z., Ortiz-Arroyo, D., Durdevic, P., 2023. Monocular based navigation system for autonomous ground robots using multiple deep learning models. International Journal of Computational Intelligence Systems 16, 79–97. doi:10.1007/s44196-023-00250-5.

[10] Navone, A., Martini, M., Ostuni, A., Angarano, S., Chiaberge, M., 2023. Autonomous navigation in rows of trees and high crops with deep semantic segmentation, in: European Conference on Mobile Robots (ECMR). doi:10.1109/ECMR59166.2023.10256334. arXiv:2304.08988.

[11] Shi, J., Bai, Y., Diao, Z., Zhou, J., Yao, X., Zhang, B., 2023. Row detection based navigation and guidance for agricultural robots and autonomous vehicles in row-crop fields: Methods and applications. Agronomy 13, 1780. doi:10.3390/agronomy13071780.

[12] Shihab, I.F., Alvee, B.I., Bhagat, S.R., Sharma, A., 2024. Precise and robust sidewalk detection: Leveraging ensemble learning to surpass llm limitations in urban environments. arXiv preprint arXiv:2405.14876 .

[13] Viteri, J., Li, C.G., 2024. Autonomous sidewalk navigation featuring end-to-end RGB-D dual-convnet steering, in: IEEE International Conference on Advanced Intelligent Mechatronics (AIM), Boston, MA, USA. pp. 703–708. doi:10.1109/AIM55361.2024.10637141.

[14] Wikipedia contributors, 2025. Mental chronometry. https://en.wikipedia.org/wiki/Mental_chronometry. Average simple visual reaction time ∼200 ms; accessed December 10, 2025.

[15] Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J.M., Luo, P., 2021. Segformer: Simple and efficient design for semantic segmentation with transformers, in: Advances in Neural Information Processing Systems, pp. 12077–12090.

[16] Zhang, T.Y., Suen, C.Y., 1984. A fast parallel algorithm for thinning digital patterns. Communications of the ACM 27, 236–239.

[17] Zhao, J., Jiang, Q., Li, X., Luo, J., 2024. Focus on BEV: Self-calibrated cycle view transformation for monocular birds-eye-view segmentation. arXiv preprint arXiv:2410.15932. URL: https://arxiv.org/abs/2410.15932.

[18] Zhu, Y., Mottaghi, R., Kolve, E., Lim, J.J., Gupta, A., Fei-Fei, L., Farhadi, A., 2017. Target-driven visual navigation in indoor scenes using deep reinforcement learning, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 3357–3364.