SE(2) Navigation Mesh: A Yaw-Aware Representation for Ground Robot Global Navigation

Shuyang Shi, Kaixian Qu, Changan Chen, Ines Kast, Yuntao Ma, Marco Hutter
Robotic Systems Lab, ETH Zurich, Zurich, Switzerland
Email: kaixqu@ethz.ch

***Abstract*—Global navigation for ground robots in complex multi-level environments requires representations that accurately capture traversable regions while enabling efficient path planning. Current approaches present key limitations: Point clouds and volumetric occupancy maps lack explicit surface structure for traversability estimation, whereas direct pathfinding on dense triangle meshes is computationally prohibitive. Navigation meshes mitigate these challenges through polygonal abstraction of the underlying mesh, but assume yaw-invariant traversability, rendering them unsuitable for non-circular robots in constrained spaces. We propose SE(2) Navigation Mesh (SE(2) NavMesh), a polygonal representation of traversable regions that encodes yaw-dependent traversability. Our method evaluates traversability using footprint masks and constructs a graph over yaw-specific layers with explicit translational and rotational connectivity. Grounded in this representation, we develop an A*-String Pulling-A* (ASA) pathfinding strategy that hierarchically optimizes robot position and heading. We further present an online method that incrementally updates the SE(2) NavMesh from streaming point clouds during concurrent geometry reconstruction. In simulation, the SE(2) NavMesh captures over $50\,\%$ more traversable area than classical NavMeshes, and the combined SE(2) NavMesh + ASA pipeline outperforms sampling-based baselines in constrained environments. Extensive real-world experiments on a physical robot validate real-time online generation and successful navigation across multiple environments.**

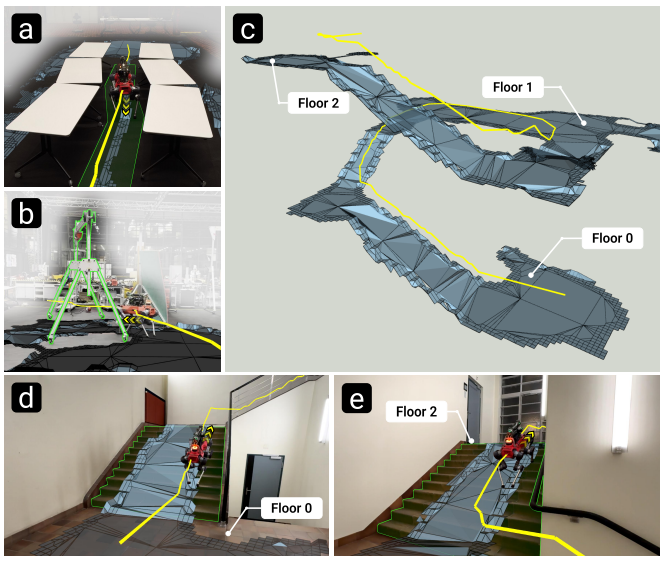


Fig. 1: Real-world robot navigation on the SE(2) NavMesh. Yellow polylines indicate planned paths, and blue polygons represent traversable regions. SE(2) NavMesh captures yaw-dependent traversability in constrained scenes such as (a) narrow passages ($0.8\,\mathrm{m}$ wide, only $0.27\,\mathrm{m}$ wider than the robot) and (b) overhanging obstacles. It also supports multi-level structures (c), enabling successful stair traversal across floors (d, e).

I. Introduction

Ground robots are increasingly deployed for infrastructure and industrial inspection [1, 2]. Many facilities, such as chemical plants and power stations, contain stairs, elevated platforms, and uneven terrain that are inaccessible to wheeled robots. Recent advances in legged locomotion enable robots to traverse rough terrain, stairs, and narrow structures [3, 4], making them promising inspection platforms [5].

Despite these advancing locomotion capabilities, accurately representing traversable regions that capture complex geometry while enabling efficient global navigation remains an open challenge. Point clouds are a common environment representation [6, 7], but their lack of structural organization makes traversability estimation inefficient, as surfaces and connectivity must be inferred through additional processing [8, 9]. Volumetric representations such as occupancy maps and signed distance fields [10, 11] enable efficient 3D obstacle reasoning but do not explicitly preserve terrain surface structure, making fine-grained traversability estimation difficult [12, 13]. At lower dimensionality, 2.5D elevation maps [14, 15] are widely used for legged locomotion but are inherently single-layer and cannot represent multi-level structures. Triangle meshes [16, 17] preserve surface geometry and adjacency, supporting traversability analysis and graph-based pathfinding. However, accurately representing complex environments requires dense meshes that incur substantial pathfinding overhead. Navigation Meshes (NavMeshes) address this by abstracting traversable regions into connected convex polygons [18], and are widely used in game engines [19] and, recently, in robot navigation [20]. However, the classical NavMesh [19] assumes yaw-invariant traversability, which fails to capture yaw-dependent traversability for ground robots with non-circular footprints in narrow environments [20].

In this work, we propose SE(2) Navigation Mesh (SE(2) NavMesh), a representation for global navigation of ground robots in complex 3D environments (Figure 1). SE(2) NavMesh extends the classical NavMesh by incorporating the robot's yaw into mesh construction, enabling more accurate

traversability estimation and navigation in constrained scenes. As the robot's motion is constrained to the polygon surfaces, the navigation space has two translational (x, y) and one rotational (yaw) degree of freedom, motivating the SE(2) naming. The main contributions are:

- **SE(2) NavMesh**, a novel representation that encodes yaw-dependent feasibility of traversable regions, representing narrow structures more accurately than the classical NavMesh.
- A three-stage **ASA (A*-String Pulling-A*)** pathfinding strategy that produces geometrically straightened paths with smooth rotation, supported by a yaw-specific layered representation with inter-layer and intra-layer connectivity.
- An **online generation** method that incrementally constructs and updates the mesh from point clouds, running efficiently on onboard CPUs for real-world deployment.

An overview of the pipeline is shown in Figure 2. A project page containing an interactive path planning explorer and additional experimental results is available at https://se2-navmesh.github.io/.

II. Related Work

Surface and volumetric representations. To enable navigation on point clouds, prior work infers local geometry and plans via graph search [8] or sampling-based planners [9], but statistics-based traversability evaluation struggles in complex terrain. 3D occupancy maps support learned voxel-wise traversability [13] or ground-region extraction with grid-based planning [12], yet learned models are coupled to a specific platform, and slope-based metrics may misclassify stairs. 2.5D elevation maps are widely adopted for legged locomotion [21, 22] but cannot represent multi-floor environments. Tomographic slicing [23] addresses this but assumes a circumscribed circular footprint that is overly conservative in narrow passages. Triangle-mesh methods [16, 17] provide continuous surface representations but rely on surface-based traversability estimation and incur high planning times for dense meshes.

Navigation meshes. NavMeshes abstract traversable surfaces into connected polygons. Recast [19] combines surface geometry with voxel-based clearance and connectivity analysis to produce polygons forming a navigation graph for efficient A* [24] search. GaitMesh [20] applies NavMeshes to ground-robot navigation in multi-floor environments via curvature-based terrain analysis and gait selection. However, these representations assume yaw-independent traversability and approximate the robot as a cylinder, preventing accurate representation of narrow passages with yaw-dependent traversability, and rely on offline construction. In contrast, our SE(2) NavMesh incorporates robot yaw during mesh construction and uniquely supports yaw-dependent traversability, configurable footprints, and online generation from onboard sensing.

III. Problem Formulation and Preliminaries

We consider a holonomic ground robot, typical of legged systems, capable of omnidirectional translation and yawing across traversable surfaces, with a fixed planar footprint and given traversal-capability parameters. The robot traverses a 3D environment represented as a polygonal surface mesh $\mathcal{M} \subset \mathbb{R}^3$. Its state is $\mathbf{x} = (\mathbf{p}, \psi)$, where $\mathbf{p} \in \mathcal{M}$ is the projection of the robot base reference point onto the nearest polygonal surface (along gravity) and $\psi \in S^1$ is the yaw. The configuration space is $\mathcal{X} = \mathcal{M} \times S^1$, and the traversable state space is

$$\mathcal{X}_{\text{free}} = \{(\mathbf{p}, \psi) \in \mathcal{X} \mid \mathbf{p} \in \mathcal{P}_{\text{free}}(\psi)\} , \tag{1}$$

where $\mathcal{P}_{\text{free}}(\psi) \subset \mathcal{M}$ is the set of feasible positions for a fixed yaw. Our objective is a representation that derives $\mathcal{X}_{\text{free}}$ from $\mathcal{M}$, supports pathfinding between $\mathbf{x}_{\text{start}}$ and $\mathbf{x}_{\text{goal}}$, and can be generated online from sensor input.

The classical NavMesh [19] models the robot as a cylinder and ignores yaw, requiring a position to be feasible for all yaws, i.e., it represents $\bigcap_{\psi \in S^1} \mathcal{P}_{\text{free}}(\psi)$. It partitions traversable positions into a set of convex polygonal regions $\mathcal{R}_{\text{trav}} = \{\mathcal{R}_1, \mathcal{R}_2, \ldots\}$ and builds a graph in which each region is a vertex and edges connect regions with overlapping boundaries. This conservative estimation yields $\mathcal{X}_{\text{trav}} = \mathcal{P}_{\text{trav}} \times S^1 \subseteq \mathcal{X}_{\text{free}}$, so feasible configurations valid only for specific yaws are not captured.

IV. SE(2) Navigation Mesh

A. Definitions

Yaw channel. We discretize $[0, 2\pi)$ uniformly into N_Ψ headings $\Psi = \{\psi_1, \ldots, \psi_{N_\Psi}\}$, $\psi_i = i \cdot 2\pi/N_\Psi$, with yaw-channel index set $\mathcal{I}_\Psi = \{1, \ldots, N_\Psi\}$. For a position $\mathbf{p} \in \mathcal{M}$, channel feasibility is $C_i(\mathbf{p}) = 1$ if the robot can occupy $\mathbf{p}$ with yaw ψ_i, and 0 otherwise.

Footprint mask. The footprint mask is a binary matrix representing the planar occupancy of the robot footprint, with one designated reference cell at the robot base (Figure 3b). An *isolated-yaw* mask captures occupancy at a single yaw (used for collision checking during sampling-based planning), while a *continuous-yaw* mask captures the union of areas swept over $[\psi_i - \frac{\psi_{\text{int}}}{2}, \psi_i + \frac{\psi_{\text{int}}}{2})$ with $\psi_{\text{int}} = 2\pi/N_\Psi$. Feasibility under the continuous-yaw masks of two adjacent headings guarantees a safe rotation between them.

Safe and restricted regions. For each region $\mathcal{R}_j \in \mathcal{R}_{\text{trav}}$, yaw feasibility is uniform within the region. We define its feasible yaw channel set $\mathcal{I}(\mathcal{R}_j) \subseteq \mathcal{I}_\Psi$. A region is *safe* if traversable under all headings ($\mathcal{I}(\mathcal{R}_j) = \mathcal{I}_\Psi$) and *restricted* if feasible only under a subset ($\emptyset \subsetneq \mathcal{I}(\mathcal{R}_j) \subsetneq \mathcal{I}_\Psi$), giving $\mathcal{R}_{\text{trav}} = \mathcal{R}_{\text{safe}} \cup \mathcal{R}_{\text{restricted}}$.

Yaw-specific layers. For each channel i, the yaw-specific traversability layer collects the regions traversable under heading ψ_i: $\mathcal{L}_i = \{\mathcal{R}_j \in \mathcal{R}_{\text{trav}} \mid i \in \mathcal{I}(\mathcal{R}_j)\}$, with $\mathcal{R}_{\text{safe}} = \bigcap_i \mathcal{L}_i$ and $\mathcal{R}_{\text{trav}} = \bigcup_i \mathcal{L}_i$. We denote by $\mathcal{R}_j^{(i)}$ the layer-specific region of $\mathcal{R}_j$ in layer $\mathcal{L}_i$, which exists iff $\mathcal{R}_j$ is feasible at ψ_i.

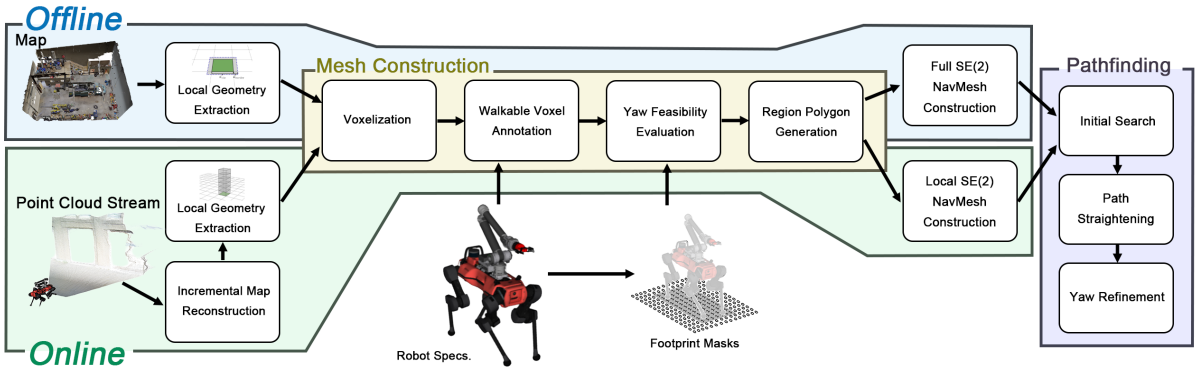
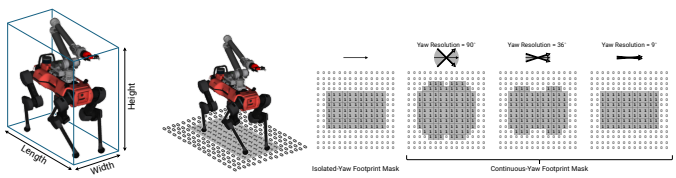


Fig. 2: Overview of the SE(2) NavMesh system. The offline pipeline constructs the mesh from a pre-built map, whereas the online pipeline updates it from a point cloud stream. During construction, map geometry is processed using robot-specific traversal constraints (traversal capabilities, height, footprint masks). Path planning is then performed on the generated SE(2) NavMesh using a three-stage pathfinding strategy.



(a) (b)

Fig. 3: (a) Cuboid robot approximation. (b) Binary footprint masks, where 1 denotes cells occupied by the footprint and 0 unoccupied cells.

Connectivity. Connectivity extends the classical definition by incorporating yaw into two types. Two layer-specific regions in the same layer are *translationally* connected if their parent regions share a boundary segment of positive measure:

$$\mathrm{Conn}_{\mathrm{trans}}(\mathcal{R}_a^{(i)}, \mathcal{R}_b^{(i)}) = 1 \iff \mathcal{H}^1(\partial\mathcal{R}_a \cap \partial\mathcal{R}_b) > 0\,. \tag{2}$$

Two layer-specific regions of the same region in adjacent layers are *rotationally* connected, representing in-place rotation:

$$\mathrm{Conn}_{\mathrm{rot}}(\mathcal{R}_k^{(i)}, \mathcal{R}_k^{(j)}) = 1 \iff j \equiv i \pm 1 \pmod{N_\Psi}\,, \tag{3}$$

where adjacency is cyclic. The navigation graph $G = (\mathcal{V}, \mathcal{E})$ instantiates a vertex at a representative point (e.g., midpoint) of each polygon edge in every layer-specific region. Vertices are joined by translational edges $\mathcal{E}_{\mathrm{trans}}$ within a layer and rotational edges $\mathcal{E}_{\mathrm{rot}}$ across adjacent layers. Each edge cost is the time required to execute the corresponding motion.

B. Offline Generation

Given an input triangle mesh, robot parameters $(l_{\mathrm{robot}}, w_{\mathrm{robot}}, h_{\mathrm{robot}}, h_{\mathrm{step}}, \theta_{\mathrm{climb}})$, and map settings (voxel resolution, N_Ψ), generation proceeds per tile to enable parallelism, following the TileMesh approach of Recast [19]. (i) *Footprint masks* are precomputed (continuous-yaw) for the discretized headings. (ii) The mesh is *voxelized*. Vertically connected voxels form solid spans, and the empty intervals between them (free-space spans) are classified as walkable or not based on surface geometry, robot height, and step constraints, yielding a tile terrain voxel map. (iii) A *distance map* to the nearest locally-invalid voxel drives yaw-feasibility evaluation: voxels with distance below the footprint inradius are inaccessible; those above the circumradius are safe; intermediate voxels are tested against each yaw's continuous-yaw footprint mask via breadth-first search over occupied cells, marking them restricted if any heading is feasible. (iv) A watershed [25] partitions voxels with identical feasible yaw sets into regions, which are triangulated and merged into convex polygons [19]. Per-tile regions are merged globally, with translational and rotational connectivity established as defined above.

C. ASA Pathfinding

We propose a three-stage strategy for the SE(2) NavMesh, ASA (A*-String Pulling-A*), illustrated in Figure 4.

Initial search. We locate the layer-specific start and goal regions, instantiate temporary query vertices, and run A* over the navigation graph. The cost is the estimated traversal time. For a translational edge in layer i with displacement $\Delta\mathbf{p} = \mathbf{p}_b - \mathbf{p}_a$ and local longitudinal and lateral axes $\mathbf{n}_{\mathrm{long}} = (\cos\psi_i, \sin\psi_i)^\top$, $\mathbf{n}_{\mathrm{lat}} = (-\sin\psi_i, \cos\psi_i)^\top$,

$$\mathrm{Cost}(\mathbf{x}_a, \mathbf{x}_b) = \frac{|\Delta\mathbf{p}\cdot\mathbf{n}_{\mathrm{long}}|}{v_{\mathrm{long}}} + \frac{|\Delta\mathbf{p}\cdot\mathbf{n}_{\mathrm{lat}}|}{v_{\mathrm{lat}}}\,, \tag{4}$$

while a rotational edge (in-place rotation by ψ_{int}) costs $2\pi/(N_\Psi\,\omega)$, where $v_{\mathrm{long}}, v_{\mathrm{lat}}, \omega$ are the robot's longitudinal/lateral speeds and yaw rate. The admissible heuristic combines lower bounds on translation and rotation time:

$$\mathrm{Heur}(\mathbf{x}_j) = \frac{\|\mathbf{p}_{\mathrm{goal}} - \mathbf{p}_j\|_2}{\max\{v_{\mathrm{long}}, v_{\mathrm{lat}}\}} + \frac{\min\{|\psi_{\mathrm{goal}} - \psi_j|, 2\pi - |\psi_{\mathrm{goal}} - \psi_j|\}}{\omega}\,. \tag{5}$$

The search returns a state sequence spanning multiple layers and the associated ordered layer-specific region sequence Π_{init}.

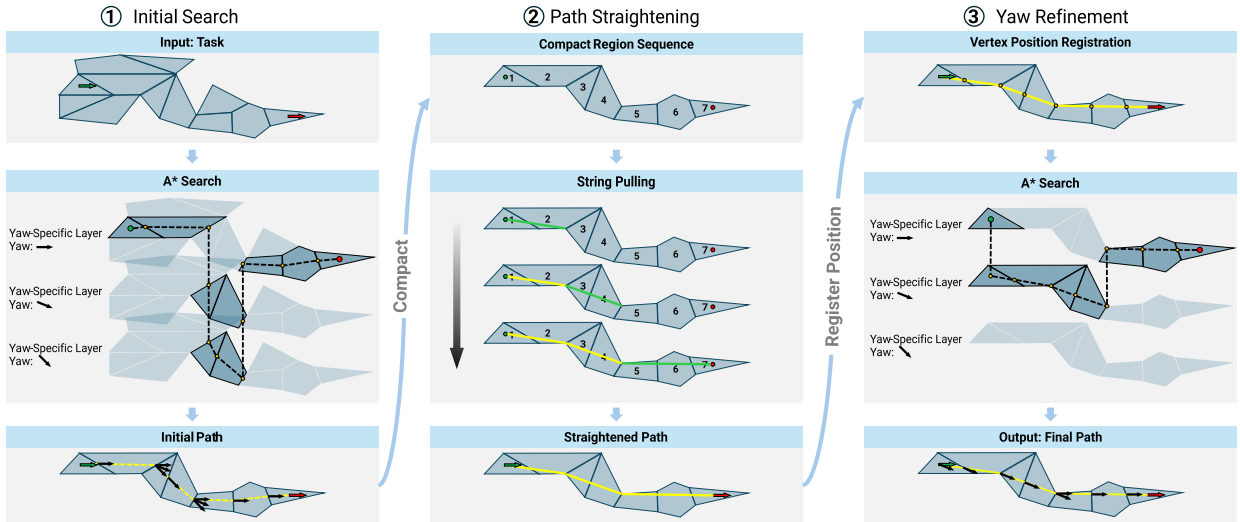


Fig. 4: Overview of the ASA pathfinding strategy: (1) an initial A* search obtains a feasible path and layer-specific region sequence, (2) string pulling straightens the path, and (3) a second A* search refines headings while respecting the straightened-path constraints.

Path straightening. Π_{init} is projected onto the original regions (discarding layer information) and consecutive duplicates merged into a compact sequence Π_{compact}, which defines a continuous polygonal corridor. String pulling [26] computes the Euclidean shortest path γ within this corridor. Although yaw is ignored here, γ remains feasible because the corridor was obtained from an A* search over yaw-specific layers, which already guarantees feasible yaw transitions between consecutive regions.

Yaw refinement. Because γ changes motion direction along its segments, we replan yaw constrained to γ: At each position we retrieve the feasible yaw channel set of the containing regions, register vertices for all feasible headings, rebuild G, and run a second A* with the same cost and heuristic to obtain an optimal heading-consistent sequence.

D. Online Generation

The online method incrementally builds and updates the SE(2) NavMesh from onboard point clouds, reusing most offline steps. We employ Voxblox [10] to fuse point clouds into a TSDF and extract a triangle mesh via marching cubes [27]. Because map updates are spatially localized, we avoid global reconstruction by dividing each tile vertically into equal-height *slabs* aligned with the Voxblox block grid. When the geometry within a slab changes, all slabs in a surrounding 3D update window are reconstructed locally (reusing the offline voxelization, yaw feasibility evaluation, and polygon generation steps) and merged into the global mesh, re-establishing connectivity with vertically and horizontally adjacent slabs.

V. Experiments

We assess the SE(2) NavMesh in terms of representation quality, planning performance, and real-world deployability. Simulations use an ANYmal [28] model, a quadruped capable of locomotion over stairs and sloped terrain, whose parameters and the mesh-generation settings are listed in Table I. We evaluate six diverse indoor scenes from HM3D [29] that cover multi-level structures, narrow passages, and doorways. For the classical NavMesh, the robot circumradius is derived from l_{robot} and w_{robot}. All simulation experiments run on an Intel i7-11800H CPU.

TABLE I: Robot (ANYmal) and Mesh Generation Parameters

Category	Parameters	Values
Dimensions	$l_{\text{robot}}, w_{\text{robot}}, h_{\text{robot}}$	$0.93\,\text{m}, 0.53\,\text{m}, 0.89\,\text{m}$
Traversal Cap.	$h_{\text{step}}, \theta_{\text{climb}}$	$0.25\,\text{m}, 30^\circ$
Speed	$v_{\text{long}}, v_{\text{lat}}, \omega$	$0.5, 0.1, 0.5$
Voxel	$s_{\text{voxel}} \times s_{\text{voxel}} \times h_{\text{voxel}}$	$0.1 \times 0.1 \times 0.1\,\text{m}$
Tile / Yaw	$d_{\text{tile}} \times d_{\text{tile}},\ N_\Psi$	$16 \times 16,\ 40$

A. Coverage: NavMesh vs. SE(2) NavMesh

By accounting for yaw-dependent traversability, SE(2) NavMesh preserves narrow passages and near-boundary regions discarded by the classical NavMesh (Figure 5). In open areas safe voxels merge into large convex polygons, whereas in corridors and staircases the feasible yaw set varies across neighboring voxels, yielding more, smaller polygons that encode local yaw-dependent traversability—an adaptive discretization that keeps fine resolution where it matters.

Quantitatively, we measure the total traversable area and the area of the largest connected traversable component. Owing to the inclusion of restricted regions and a more accurate yaw-dependent evaluation, SE(2) NavMesh captures over $50\,\%$ more traversable area than the classical NavMesh in every scene. Moreover, restricted regions bridge previously disconnected safe regions, substantially enlarging the connected component and improving reachability. This enrichment comes at modest computational cost (Table II): despite producing roughly 8–10 times more region polygons, the total build time

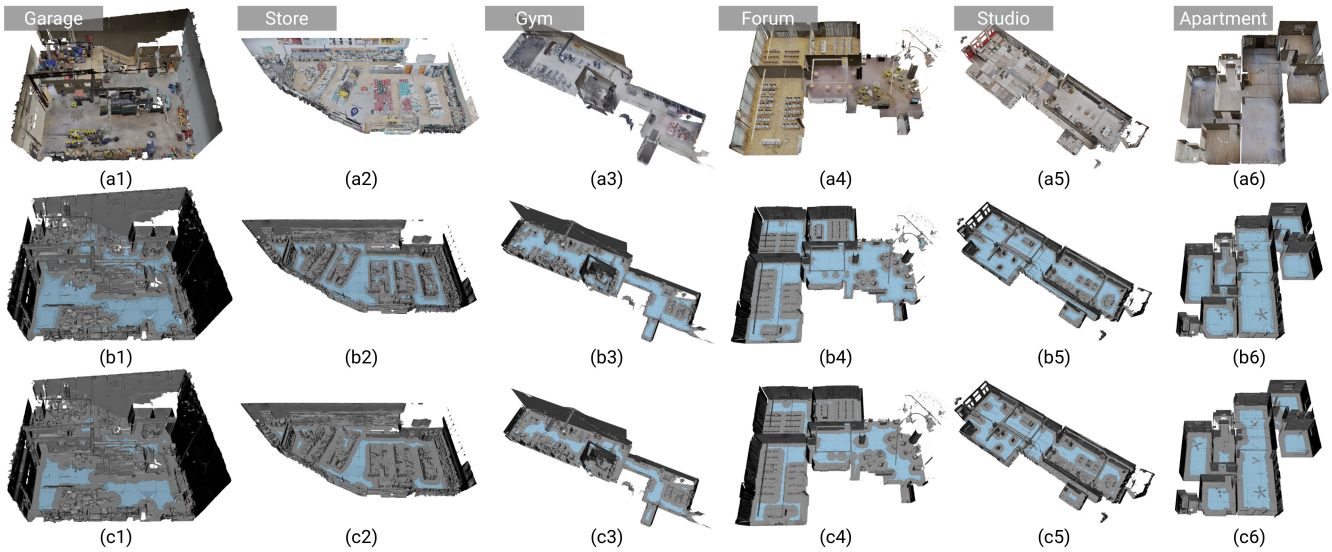


Fig. 5: Offline generation across six scenes. Row 1: 3D scene models. Row 2 (b1–b6): traversable regions from SE(2) NavMesh. Row 3 (c1–c6): traversable regions from the classical NavMesh. Traversable regions are blue.

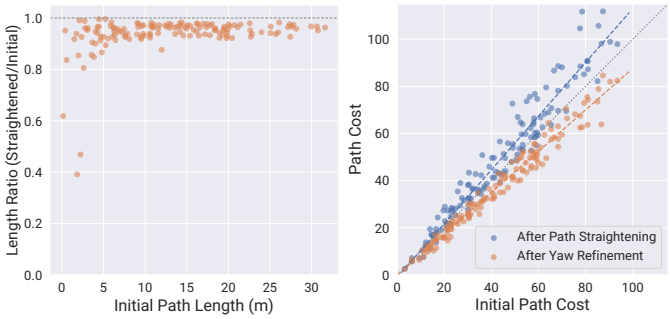
TABLE II: Build time and region-polygon count across simulation scenes

Method	Mask Gen. (ms)	Mesh Const. (ms)	Total (ms)	# Region Polygons
Garage				
NavMesh	-	37.80	37.80	278
SE(2) NavMesh	2.44	86.25	88.69	2859
Store				
NavMesh	-	48.15	48.15	374
SE(2) NavMesh	2.48	91.96	94.44	4007
Gym				
NavMesh	-	43.53	43.53	478
SE(2) NavMesh	2.37	89.15	91.52	4362
Forum				
NavMesh	-	42.02	42.02	628
SE(2) NavMesh	2.16	107.25	109.41	5055
Studio				
NavMesh	-	42.87	42.87	603
SE(2) NavMesh	2.37	102.17	104.54	4521
Apartment				
NavMesh	-	25.83	25.83	193
SE(2) NavMesh	2.28	60.47	62.75	1729

grows only 2–3 times, with continuous-yaw mask generation adding only $\approx 2.35\,\mathrm{ms}$ on average.

B. Pathfinding Benchmark

We benchmark ASA against RRT [30], RRT* [31], and PRM [32] (implemented in OMPL [33]), using two state-validity checkers: a voxel-based checker and an SE(2)-NavMesh-based checker (planners using the latter are suffixed -SE2NM). We report the success rate (SR) and success weighted by inverse path cost (SPC), $\mathrm{SPC} = \frac{1}{N}\sum_i S_i\, c_i^* / \max(c_i, c_i^*)$, where S_i is a binary success indicator, c_i the achieved path cost, and c_i^* the best cost found across all methods. SPC jointly captures success and solution quality.



(a) Path length reduction. (b) Path cost comparison.

Fig. 6: Path quality analysis of the ASA pipeline.

We evaluate 100 randomly sampled start-goal pairs in two representative scenes (Table III). ASA achieves the highest SR and SPC in both, with SPC close to 1, indicating its solutions are consistently among the lowest-cost found. Region connectivity strongly affects SR: the fragmented *Gym* yields a lower average SR than the largely connected *Studio*. Analyzing the ASA stages (Figure 6), string pulling shortens the initial path by $6.2\,\%$ on average but can raise cost by $\sim 10\,\%$ due to mismatched yaw. The final yaw-refinement stage restores consistency, yielding a final cost averaging $87\,\%$ of the initial.

C. Real-World Experiments

We deploy the online SE(2) NavMesh on an ANYmal with a 6-DoF manipulator and a wrist-mounted ZED X Mini camera, with image processing and mesh generation running onboard an NVIDIA Jetson Orin. In real-world experiments, the robot is first teleoperated while the SE(2) NavMesh is generated incrementally (Figure 7), after which the SE(2) NavMesh is used for navigation. Comparing local against global updates at a target rate of $4\,\mathrm{Hz}$ (Figure 8), global updates grow approxi-

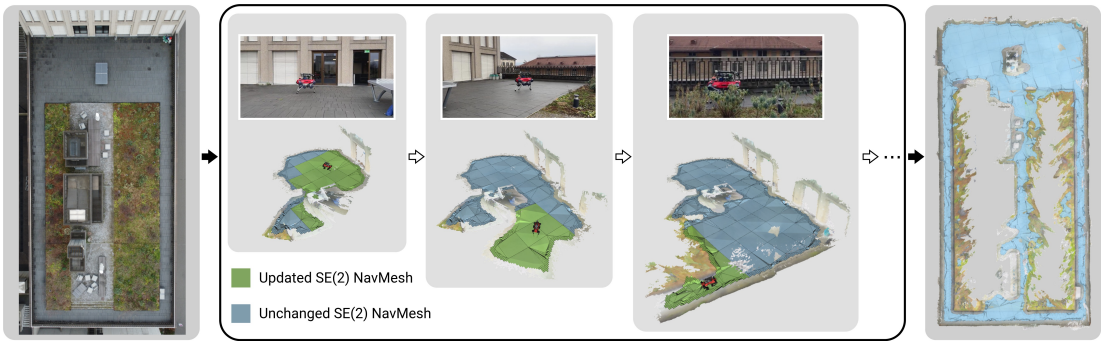


Fig. 7: Online SE(2) NavMesh generation in a garden. The left panel shows the real-world scene. The middle panel shows incremental geometry reconstruction and local SE(2) NavMesh updates. The right panel shows the final reconstructed geometry and the generated SE(2) NavMesh.

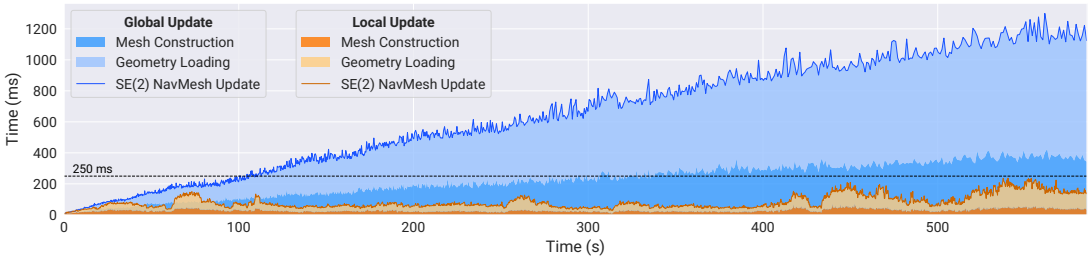


Fig. 8: Local vs. global SE(2) NavMesh update times during online generation in the garden scene. Local updates maintain the target real-time rate, whereas global updates incur growing latency as the reconstructed map expands.

TABLE III: SR and SPC on Randomly Sampled Tasks

Planner	Gym		Studio	
	SR ↑	SPC ↑	SR ↑	SPC ↑
ASA	**41 %**	**0.41**	**98 %**	**0.98**
RRT-SE2NM	40 %	0.13	97 %	0.29
RRT*-SE2NM	32 %	0.18	82 %	0.26
PRM-SE2NM	**41 %**	0.27	96 %	0.63
RRT	**41 %**	0.13	97 %	0.29
RRT*	36 %	0.16	80 %	0.25
PRM	36 %	0.15	84 %	0.42

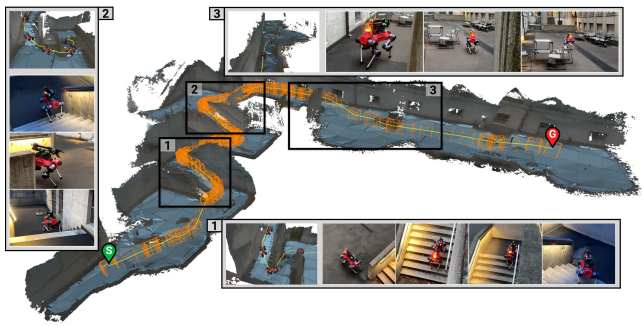


Fig. 9: Real-world navigation in an outdoor multi-level environment. The planned path is visualized as a yellow polyline, with orange wireframe boxes indicating the robot states

mately linearly and fail to sustain the target rate once the map becomes large ($\approx 100\,\mathrm{s}$), whereas local updates peak at $237\,\mathrm{ms}$ (average $83\,\mathrm{ms}$), consistently maintaining $4\,\mathrm{Hz}$. We validate the proposed framework across multiple real-world scenes. Using the generated SE(2) NavMesh, the robot successfully ascends multi-level stairs (Figure 9), traverses a $0.8\,\mathrm{m}$ passage (only $0.27\,\mathrm{m}$ wider than the robot), and navigates beneath an overhanging crane (Figure 1), validating the proposed online SE(2) NavMesh for navigation in constrained multi-level environments.

VI. Conclusion

We presented SE(2) Navigation Mesh, a representation for global navigation of ground robots in complex 3D environments. By incorporating robot yaw into navigation mesh construction, it captures yaw-dependent traversability that classical NavMeshes cannot, via a yaw-specific layered representation with inter- and intra-layer connectivity. The ASA pathfinding strategy produces geometrically efficient paths that respect yaw feasibility, and an online pipeline constructs the mesh efficiently from point clouds. Experiments show over $50\,\%$ greater traversable-area coverage than classical NavMeshes at modest computational overhead, superior pathfinding in constrained environments, and reliable real-world online generation and navigation. Current limitations include a fixed agent height and reliance on purely geometric cues. Integrating variable body height, semantic segmentation, and autonomous exploration are promising directions for future work.

References

[1] D. Lattanzi and G. Miller, "Review of robotic infrastructure inspection systems," *Journal of Infrastructure Systems*, vol. 23, no. 3, p. 04017004, 2017.

[2] S. Halder and K. Afsari, "Robots in inspection and monitoring of buildings and infrastructure: A systematic review," *Applied Sciences*, vol. 13, no. 4, p. 2304, 2023.

[3] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[4] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged locomotion in challenging terrains using egocentric vision," in *Conference on robot learning*. PMLR, 2023, pp. 403–415.

[5] H. Kolvenbach, D. Wisth, R. Buchanan, G. Valsecchi, R. Grandia, M. Fallon, and M. Hutter, "Towards autonomous inspection of concrete deterioration in sewers with legged robots," *Journal of field robotics*, vol. 37, no. 8, pp. 1314–1327, 2020.

[6] N. Pérez-Higueras, A. Jardón, Á. Rodríguez, and C. Balaguer, "3d exploration and navigation with optimal-rrt planners for ground robots in indoor incidents," *Sensors*, vol. 20, no. 1, p. 220, 2019.

[7] F. Yang, C. Cao, H. Zhu, J. Oh, and J. Zhang, "Far planner: Fast, attemptable route planner using dynamic visibility update," in *2022 ieee/rsj international conference on intelligent robots and systems (iros)*. IEEE, 2022, pp. 9–16.

[8] M. Liu, "Robotic online path planning on point cloud," *IEEE transactions on cybernetics*, vol. 46, no. 5, pp. 1217–1228, 2015.

[9] P. Krüsi, P. Furgale, M. Bosse, and R. Siegwart, "Driving on point clouds: Motion planning, trajectory optimization, and terrain assessment in generic nonplanar environments," *Journal of Field Robotics*, vol. 34, no. 5, pp. 940–984, 2017.

[10] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1366–1373.

[11] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, no. 3, pp. 189–206, 2013.

[12] J. Wang, L. Xu, H. Fu, Z. Meng, C. Xu, Y. Cao, X. Lyu, and F. Gao, "Towards efficient trajectory generation for ground robots beyond 2d environment," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 7858–7864.

[13] J. Frey, D. Hoeller, S. Khattak, and M. Hutter, "Locomotion policy guided traversability learning using volumetric representations of complex environments," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 5722–5729.

[14] P. Fankhauser, M. Bloesch, C. Gehring, M. Hutter, and R. Siegwart, "Robot-centric elevation mapping with uncertainty estimates," in *Mobile Service Robotics*. World Scientific, 2014, pp. 433–440.

[15] P. Fankhauser, M. Bloesch, and M. Hutter, "Probabilistic terrain mapping for mobile robots with uncertain localization," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3019–3026, 2018.

[16] F. Ruetz, E. Hernández, M. Pfeiffer, H. Oleynikova, M. Cox, T. Lowe, and P. Borges, "Ovpc mesh: 3d free-space representation for local ground vehicle navigation," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8648–8654.

[17] S. Pütz, T. Wiemann, M. K. Piening, and J. Hertzberg, "Continuous shortest path vector field navigation on 3d triangular meshes for mobile robots," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 2256–2263.

[18] W. Van Toll, R. Triesscheijn, M. Kallmann, R. Oliva, N. Pelechano, J. Pettré, and R. Geraerts, "A comparative study of navigation meshes," in *Proceedings of the 9th International Conference on Motion in Games*, 2016, pp. 91–100.

[19] M. Mononen, "Recast Navigation," https://github.com/recastnavigation/recastnavigation, 2014.

[20] M. Brandao, O. B. Aladag, and I. Havoutis, "GaitMesh: Controller-aware navigation meshes for long-range legged locomotion planning in multi-layered environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3596–3603, 2020.

[21] L. Wellhausen and M. Hutter, "Rough terrain navigation for legged robots using reachability planning and template learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 6914–6921.

[22] B. Yang, L. Wellhausen, T. Miki, M. Liu, and M. Hutter, "Real-time optimal navigation planning using learned motion costs," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 9283–9289.

[23] B. Yang, J. Cheng, B. Xue, J. Jiao, and M. Liu, "Efficient global navigational planning in 3-d structures based on point cloud tomography," *IEEE/ASME Transactions on Mechatronics*, vol. 30, no. 1, pp. 321–332, 2025.

[24] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[25] S. Beucher, "Use of watersheds in contour detection," in *Proc. Int. Workshop on Image Processing, Sept. 1979*, 1979, pp. 17–21.

[26] M. Mononen, "Simple Stupid Funnel Algorithm," https://digestingduck.blogspot.com/2010/03/simple-stupid-funnel-algorithm.html, 2010.

[27] W. E. Lorensen and H. E. Cline, "Marching cubes: A

high resolution 3d surface construction algorithm," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 163–169, 1987.

[28] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch *et al.*, "Anymal-a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2016, pp. 38–44.

[29] S. K. Ramakrishnan, A. Gokaslan, E. Wijmans, O. Maksymets, A. Clegg, J. Turner, E. Undersander, W. Galuba, A. Westbury, A. X. Chang *et al.*, "Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai," *arXiv preprint arXiv:2109.08238*, 2021.

[30] S. M. LaValle, "Rapidly-exploring random trees: Progress and prospects," *Algorithmic and Computational Robotics: New Directions 2000 WAFR*, p. 293, 2001.

[31] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[32] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 2002.

[33] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.