PointNT: Point Navigation Transformer

Jun-Gill Kang[1*], Taehong Kim[1*], Seonsoo Kim[1], Jihong Min[1], Seongil Hong[1], and Kiho Kwak[1]

[1]Agency for Defense Development(ADD)
* These authors contributed equally to this work.

Fig. 1: Point Navigation Transformer (PointNT) is a sequential transformer-based navigation policy equipped with a powerful 3D point cloud encoder, designed for navigation and exploration in diverse and challenging environments. We evaluate PointNT in both real-world and simulation settings, deploying it on two different robotic platforms (legged and wheeled) in real-world experiments, and three platforms (legged-wheel, wheeled, and tracked) in simulation, all using a same transformer policy, which showcase our ability of proposed approach.

***Abstract*—Humans can navigate and explore unfamiliar environments by leveraging prior experience to decide where to go and how to get there. This remarkable capability requires a geometric understanding of the surrounding scene. To this end, we introduce Point Navigation Transformer (PointNT), a foundational navigation and local exploration policy that utilizes raw 3D point cloud streams to propose plausible exploration targets along with waypoint trajectories to reach those targets. We replace PointNet's computationally expensive invariance layer with a lightweight encoder designed to capture egocentric, large-scale environmental context. This change reduces the total parameter count by 5.7× (0.97M to 0.17M) and lowers the average prediction error by 55% (3.71m to 1.68m). We further introduce an SE(2) matching loss to enhance spatial consistency. Compared to image-based approaches, our method demonstrates a richer understanding of geometric semantics, effectively distinguishing between similar and dissimilar scenes even without rich color information. We validate our approach through extensive indoor and outdoor experiments across previously unseen environments including mountainous terrain, dense forests, sandy beaches, and an underground tunnel using five different platforms in both simulation and the real world, all using a single policy in a zero-shot manner.**

I. Introduction

Navigation is a fundamental technology of robotic autonomy, powering diverse applications from autonomous exploration to real-world deployment in extreme environments. In particular, traversing highly unstructured terrains—such as mountains, forests, and tunnels—is critical for time-critical missions like search-and-rescue, disaster response, and reconnaissance. However, map-based methods [50] often demand heavy computation and are prone to failure from odometry drift caused by slippage, sharp accelerations, or sensor malfunctions. Even perfect maps require costly optimization to respect multiple robots' unique mobility limits. These challenges demand a paradigm shift in how robots perceive and plan their movements in unstructured environments.

Interestingly, intelligent agents such as humans navigate new environments efficiently by combining short-horizon local obstacle avoidance (e.g., dodging walls and rocks) with long-horizon goal-directed planning, leveraging high-level geometric clues. This capability allows exploration without prior terrain knowledge, highlighting the need for similar flexible, experience-driven approaches to robotic navigation.

For example, humans naturally center themselves in narrow hallways to quickly react to obstacles and, on uneven terrain, seek stable footholds while avoiding slippery or obstructed areas. In urban settings, they anticipate intersections by reading sidewalk patterns and building layouts even before an open

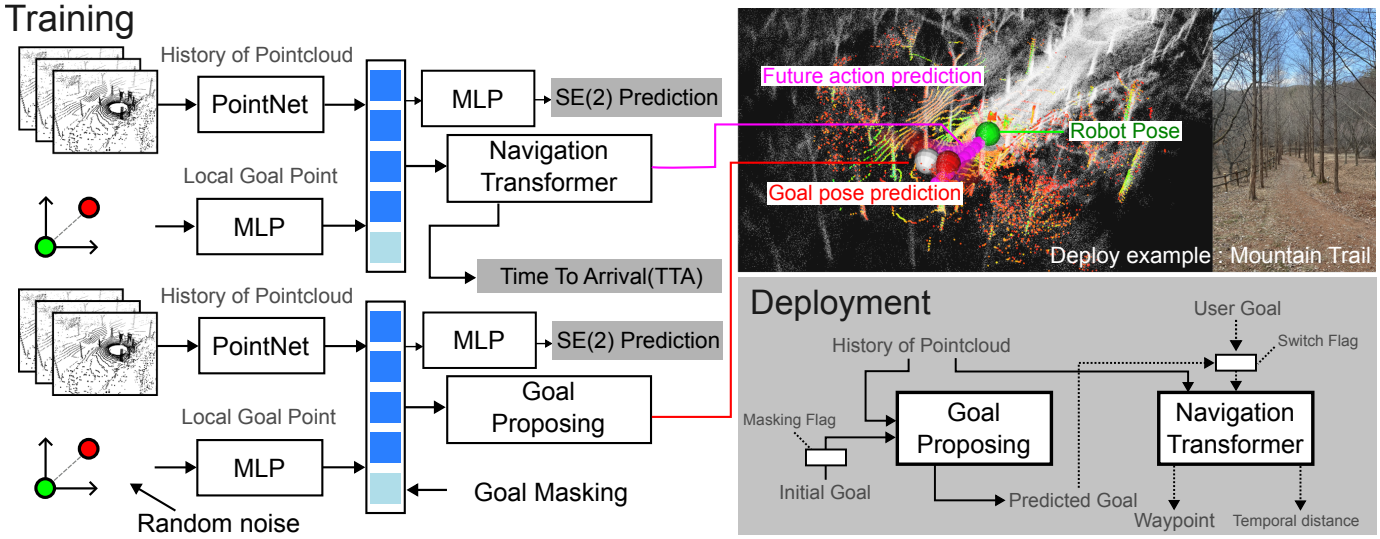

Fig. 2: PointNT comprises two modules: the **Navigation Transformer (NT)** and the **Goal Proposing (GP)**. The NT module is trained on LiDAR history and local goal position, producing waypoints to reach the goal, an estimated Time-to-Arrival (TTA), and the SE(2) transformation between LiDAR frames. The GP module shares these inputs but uses a goal-masking token to learn plausible goal poses: *unmasked* GP refines an existing goal to make it more feasible, while *masked* GP proposes a new exploration goal purely from LiDAR history. Both modules can function independently. Notably, using GP's undirected (masked) goal as input to NT enables local exploration without any prior map or external knowledge.

path is visible. Amid crowds, they dynamically adjust their trajectory by predicting others' movements, blending local obstacle avoidance with long-term path planning. Because these behaviors span countless scenarios, heuristic or model-based replication is impractical. Instead, data-driven approaches capture this "common sense," enabling end-to-end learning without explicit assumptions about the environment.

Existing work such as ViNT [39] is closely related to our method but differs in two ways: (1) We replace the image input with LiDAR, offering a broader field of view and more long-range information; (2) We substitute the goal image input with goal point for compatibility with other odometry-based algorithms and to greatly reduce inference time and model size.

In this paper, we introduce Point Navigation Transformer (PointNT), a general-purpose transformer-based foundation model for navigation that processes historical LiDAR scans encoded by PointNet [32]. It comprises two main components: the Navigation Transformer (NT) module for local avoidance and the Goal Proposing (GP) module for selecting effective exploration points.

We train PointNT on the publicly available SCAND [14] and GND [21] datasets, which feature diverse real-world trajectories across social and navigation scenarios. These datasets include LiDAR scans, odometry, and other sensory data, making them ideal for learning robust navigation. To downsample the LiDAR point clouds, we use random sampling, which achieves the fastest runtime and yields better accuracy compared to Farthest Point Sampling (FPS) and voxel grid methods.

Because our target navigation task operates within a planar SE(2) space, we introduce an SE(2) matching loss that aligns predicted poses more precisely in these dimensions. In addition, we modify PointNet by removing certain invariance layers—originally designed for full 3D invariance—thereby reducing unnecessary computations while still preserving the critical spatial relationships in SE(2). Crucially, this change decreases the model's size and computational overhead without compromising navigation performance.

We also conducted comparative evaluations against three state-of-the-art planning algorithms that utilize image and LiDAR-based navigation: Falco [50]: A LiDAR sensor-based local planner. ViNT [39]: An end-to-end vision-based navigation model. Nomad [42]: A diffusion-action modeled vision-based navigation system. To evaluate our method against these baselines, we deployed it across diverse real-world and simulated environments in terms of navigation and exploration with following four scenarios.

Navigation Performance in 3D Obstacle Courses To assess the navigation capabilities, we conducted a 3D indoor obstacle course experiment featuring varied terrain properties such as grass, wood, gravel, and marble plates, as well as diverse geometric structures including inclined slopes. Our method consistently outperformed all baselines and was the only approach capable of successfully traversing the entire obstacle course.

Directed Exploration To evaluate the performance of directed exploration, we conducted a simulation-based study using the TARE planner [2, 3], where the frontier points generated by TARE were passed as input to our GP module. We observed that when TARE proposed exploration frontier points near walls or obstacles, our GP module corrected the goal to a safer,

more navigable area.

Undirected Exploration We tested undirected exploration (no explicit goal) against Nomad [42], a diffusion-based vision navigation approach with exploration capabilities. Our method achieved higher coverage, especially in narrow corridors, by leveraging the GP module to select effective goal locations. In contrast, Nomad exhibited less efficient coverage due to occasional suboptimal goal choices in confined spaces.

Zero-Shot Deployment Across Diverse Environments A key strength of our approach is its ability to generalize across previously unseen terrains without additional fine-tuning. We conducted zero-shot deployment of our method in challenging real-world environments, including: Mountainous terrain (uneven elevation, loose ground), Dense forests (high obstacle density, low visibility), Sandy beaches (low traction, varying surface consistency), Underground tunnels (narrow passages, poor lighting), Staircases (multi-level navigation).

Our method successfully navigated each of these settings by leveraging only historical LiDAR point cloud data, demonstrating robust adaptation to diverse and previously unseen environments without any environment-specific tuning. To our knowledge, this represents the first purely LiDAR-based navigation transformer, trained exclusively on point cloud and odometry data without external labels.

Detailed modeling of our architecture and experiments are on the supplements.

References

[1] Mahmoud Ali, Hassan Jardali, Nicholas Roy, and Lantao Liu. Autonomous navigation, mapping and exploration with gaussian processes. In *Robotics: Science and Systems*, 2023.

[2] Chao Cao, Hongbiao Zhu, Howie Choset, and Ji Zhang. Tare: A hierarchical framework for efficiently exploring complex 3d environments. In *Robotics: Science and Systems*, volume 5, page 2, 2021.

[3] Chao Cao, Hongbiao Zhu, Zhongqiang Ren, Howie Choset, and Ji Zhang. Representation granularity enables time-efficient autonomous exploration in large, complex worlds. *Science Robotics*, 8(80):eadf0970, 2023.

[4] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. *arXiv preprint arXiv:2004.05155*, 2020.

[5] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural topological slam for visual navigation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12875–12884, 2020.

[6] Joshua Julian Damanik, Jae-Won Jung, Chala Adane Deresa, and Han-Lim Choi. Lics: Navigation using learned-imitation on cluttered space. *IEEE Robotics and Automation Letters*, 2024.

[7] Jonas Frey, Matias Mattamala, Nived Chebrolu, Cesar Cadena, Maurice Fallon, and Marco Hutter. Fast traversability estimation for wild visual navigation. *arXiv preprint arXiv:2305.08510*, 2023.

[8] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous robots*, 34:189–206, 2013.

[9] Jiangpeng Hu, Fan Yang, Fang Nan, and Marco Hutter. Motion primitives planning for center-articulated vehicles. *arXiv preprint arXiv:2405.17127*, 2024.

[10] Gwanghyeon Ji, Juhyeok Mun, Hyeongjun Kim, and Jemin Hwangbo. Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion. *IEEE Robotics and Automation Letters*, 7(2): 4630–4637, 2022.

[11] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 international conference on robotics and automation (ICRA)*, pages 6023–6029. IEEE, 2019.

[12] Gregory Kahn, Pieter Abbeel, and Sergey Levine. Badgr: An autonomous self-supervised learning-based navigation system. *IEEE Robotics and Automation Letters*, 6(2): 1312–1319, 2021.

[13] Jun-Gill Kang, Dohyeon Lee, and Soohee Han. A highly maneuverable flying squirrel drone with controllable foldable wings. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6652–6659. IEEE, 2023.

[14] Haresh Karnan, Anirudh Nair, Xuesu Xiao, Garrett Warnell, Sören Pirk, Alexander Toshev, Justin Hart, Joydeep Biswas, and Peter Stone. Socially compliant navigation dataset (scand): A large-scale dataset of demonstrations for social navigation. *IEEE Robotics and Automation Letters*, 7(4):11807–11814, 2022.

[15] Junyoung Kim, Junwon Seo, and Jihong Min. Evidential semantic mapping in off-road environments with uncertainty-aware bayesian kernel inference. *arXiv preprint arXiv:2403.14138*, 2024.

[16] Taekyung Kim, Gyuhyun Park, Kiho Kwak, Jihwan Bae, and Wonsuk Lee. Smooth model predictive path integral control without smoothing. *IEEE Robotics and Automation Letters*, 7(4):10406–10413, 2022.

[17] Yunho Kim, Chanyoung Kim, and Jemin Hwangbo. Learning forward dynamics model and informed trajectory sampler for safe quadruped navigation. *arXiv preprint arXiv:2204.08647*, 2022.

[18] Jinche La, Jun-Gill Kang, and Dasol Lee. A robust, task-agnostic and fully-scalable voxel mapping system for large scale environments. *arXiv preprint arXiv:2409.15779*, 2024.

[19] Jeong Hyun Lee, Jinhyeok Choi, Simo Ryu, Hyunsik Oh, Suyoung Choi, and Jemin Hwangbo. Learning vehicle dynamics from cropped image patches for robot navigation in unpaved outdoor terrains. *IEEE Robotics and Automation Letters*, 2024.

[20] Joonho Lee, Marko Bjelonic, Alexander Reske, Lorenz Wellhausen, Takahiro Miki, and Marco Hutter. Learning robust autonomous navigation and locomotion for wheeled-legged robots. *Science Robotics*, 9(89):eadi9641, 2024.

[21] Jing Liang, Dibyendu Das, Daeun Song, Md Nahid Hasan Shuvo, Mohammad Durrani, Karthik Taranath, Ivan Penskiy, Dinesh Manocha, and Xuesu Xiao. Gnd: Global navigation dataset with multi-modal perception and multi-category traversability in outdoor campus environments. *arXiv preprint arXiv:2409.14262*, 2024.

[22] Jing Liang, Peng Gao, Xuesu Xiao, Adarsh Jagan Sathyamoorthy, Mohamed Elnoor, Ming C Lin, and Dinesh Manocha. Mtg: Mapless trajectory generator with traversability coverage for outdoor navigation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2396–2402. IEEE, 2024.

[23] Jing Liang, Amirreza Payandeh, Daeun Song, Xuesu Xiao, and Dinesh Manocha. Dtg: Diffusion-based trajectory generation for mapless global navigation. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5340–5347. IEEE, 2024.

[24] Bo Liu, Xuesu Xiao, and Peter Stone. A lifelong learning approach to mobile robot navigation. *IEEE Robotics and Automation Letters*, 6(2):1090–1096, 2021.

[25] Jianwei Liu, Maria Stamatopoulou, and Dimitrios Kanoulas. Dipper: Diffusion-based 2d path planner applied on legged robots. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9264–9270. IEEE, 2024.

[26] Ilya Loshchilov, Frank Hutter, et al. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 5, 2017.

[27] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[28] Gabriel B Margolis and Pulkit Agrawal. Walk these ways: Tuning robot control for generalization with multiplicity of behavior. In *Conference on Robot Learning*, pages 22–31. PMLR, 2023.

[29] Xiangyun Meng, Nathan Hatch, Alexander Lambert, Anqi Li, Nolan Wagener, Matthew Schmittle, JoonHo Lee, Wentao Yuan, Zoey Chen, Samuel Deng, et al. Terrainnet: Visual modeling of complex terrain for high-speed, off-road navigation. *arXiv preprint arXiv:2303.15771*, 2023.

[30] Ihab S Mohamed, Mahmoud Ali, and Lantao Liu. Gp-guided mppi for efficient navigation in complex unknown cluttered environments. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7463–7470. IEEE, 2023.

[31] Gianluca Monaci, Michel Aractingi, and Tomi Silander. Dipcan: Distilling privileged information for crowd-aware navigation. In *Robotics: Science and Systems*, 2022.

[32] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[33] Pascal Roth, Julian Nubert, Fan Yang, Mayank Mittal, and Marco Hutter. Viplanner: Visual semantic imperative learning for local navigation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5243–5249. IEEE, 2024.

[34] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022.

[35] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[36] Junwon Seo, Sungdae Sim, and Inwook Shim. Learning off-road terrain traversability with self-supervisions only. *IEEE Robotics and Automation Letters*, 8(8):4617–4624, 2023.

[37] Dhruv Shah and Sergey Levine. Viking: Vision-based kilometer-scale navigation with geographic hints. *arXiv preprint arXiv:2202.11271*, 2022.

[38] Dhruv Shah, Ajay Sridhar, Arjun Bhorkar, Noriaki Hirose, and Sergey Levine. Gnm: A general navigation model to drive any robot. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7226–7233. IEEE, 2023.

[39] Dhruv Shah, Ajay Sridhar, Nitish Dashora, Kyle Stachowicz, Kevin Black, Noriaki Hirose, and Sergey Levine. Vint: A foundation model for visual navigation. *arXiv preprint arXiv:2306.14846*, 2023.

[40] AN Sivakumar, MV Gasparino, M McGuire, VAH Higuti, MU Akcal, and G Chowdhary. Demonstrating cropfollow++: Robust under-canopy navigation with keypoints. *Proceedings of Robotics: Science and Systems, Delft, Netherlands*, 2024.

[41] Juil Sock, Jun Kim, Jihong Min, and Kiho Kwak. Probabilistic traversability map generation using 3d-lidar and camera. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 5631–5637. IEEE, 2016.

[42] Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. Nomad: Goal masked diffusion policies for navigation and exploration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 63–70. IEEE, 2024.

[43] Maria Stamatopoulou, Jianwei Liu, and Dimitrios Kanoulas. Dippest: Diffusion-based path planner for synthesizing trajectories applied on quadruped robots. *arXiv preprint arXiv:2405.19232*, 2024.

[44] Haitong Wang, Aaron Hao Tan, and Goldie Nejat. Navformer: A transformer architecture for robot target-driven navigation in unknown and dynamic environments. *IEEE Robotics and Automation Letters*, 2024.

[45] Grady Williams, Paul Drews, Brian Goldfain, James M

Rehg, and Evangelos A Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440. IEEE, 2016.

[46] Wei Xu, Yixi Cai, Dongjiao He, Jiarong Lin, and Fu Zhang. Fast-lio2: Fast direct lidar-inertial odometry. *IEEE Transactions on Robotics*, 38(4):2053–2073, 2022.

[47] Fan Yang, Chao Cao, Hongbiao Zhu, Jean Oh, and Ji Zhang. Far planner: Fast, attemptable route planner using dynamic visibility update. In *2022 ieee/rsj international conference on intelligent robots and systems (iros)*, pages 9–16. IEEE, 2022.

[48] Fan Yang, Chen Wang, Cesar Cadena, and Marco Hutter. iplanner: Imperative path planning. *arXiv preprint arXiv:2302.11434*, 2023.

[49] Jonathan Yang, Catherine Glossop, Arjun Bhorkar, Dhruv Shah, Quan Vuong, Chelsea Finn, Dorsa Sadigh, and Sergey Levine. Pushing the limits of cross-embodiment learning for manipulation and navigation. *arXiv preprint arXiv:2402.19432*, 2024.

[50] Ji Zhang, Chen Hu, Rushat Gupta Chadha, and Sanjiv Singh. Falco: Fast likelihood-based collision avoidance with extension to human-guided navigation. *Journal of Field Robotics*, 37(8):1300–1313, 2020.

Supplementary for PointNT: Point Navigation Transformer

S1. Related Work

Navigation in unknown environments has been tackled using both map-based and learning-based methods. Map-based approaches [50, 47, 9] typically rely on accurate occupancy or semantic maps [8, 18, 15]. In contrast, learning-based methods capture high-dimensional "intent" from demonstrations, potentially generating traversability maps [41, 36, 7, 29] that planners (e.g., MPPI [45, 16]) can optimize—though cost tuning remains challenging. Some methods [12, 17, 19] learn dynamics from data while [48, 33] blend both approaches but involve costly optimization steps. Recently, direct policy-learning approaches have emerged that bypass two-stage pipelines altogether.

Certain systems learn from visual data for indoor or simulated settings [4, 5, 25, 43, 49], while others [37, 39, 38, 42, 40] handle outdoor scenarios. Beyond vision, some works rely on 2D or Bird's-Eye View (BEV) LiDAR [24, 14, 6]—thus ignoring 3D data—or require manual traversability labels when using 3D LiDAR [22, 21, 23]. Methods based on Gaussian processes [1, 30] can lack the "common sense" that human operators exhibit.

Potential approaches can be integrated with reinforcement learning (RL)-based navigation pipelines [44, 20, 31]. But this need extensive reward tuning, and hard to capture the "common sense" from demonstration and also there exists large sim to real gap of perception sensor data from current physical simulator.

Upon this prior work we introduce the PointNT that learns from history of point cloud as input to navigate and explore unknown 3D environment without any traversability label. The following section describes the detailed architecture of our model.

S2. Point Navigation Transformer

PointNT comprises two modules that address distinct navigation scales where detailed description is compromised in Fig. 2:

Navigation Transformer (NT) handles local navigation by processing historical point clouds and a goal in the robot's local frame. It predicts a waypoint to reach that goal, along with the Time-to-Arrival (TTA) and the SE(2) transformation between the last and current LiDAR frames.

Goal Proposing (GP) supervises global exploration. Given historical point clouds, it either proposes a new exploration goal when the goal point is masked—using dataset priors—or refines an existing goal into a more feasible position when unmasked.

During deployment, one can specify which of the three modes PointNT uses, as shown in Fig. S1:

Standalone NT (goal-conditioned navigation): The NT module alone navigates to a given goal by predicting waypoints. A separate low-level path tracker then follows these waypoints.

Unmasked GP (directed exploration): With a known goal, the GP module refines it for feasibility before passing it to NT for navigation.

Masked GP (undirected exploration): Without an explicit goal, the GP module proposes a new goal purely from the LiDAR history, and the Navigation Transformer (NT) then generates waypoints and navigates toward it.

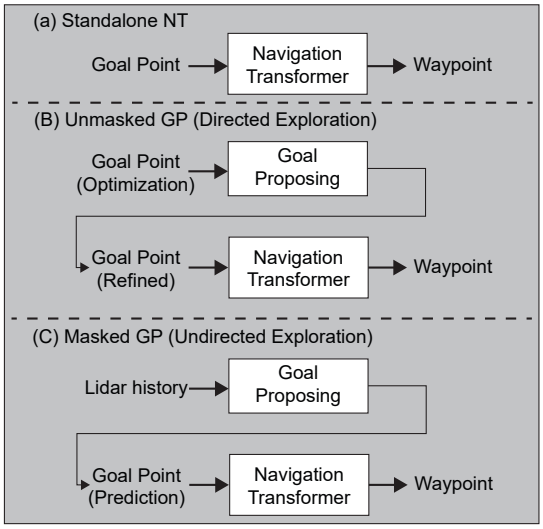

Fig. S1: Three operational modes of PointNT.

However, during real-world deployments, sensor noise or out-of-distribution inputs may cause the GP module to produce suboptimal or invalid goals. To mitigate this, we generate multiple *candidate* goal points near the GP's proposed output (Fig. S2). Concretely, let R be the distance from the robot to the GP output. We then uniformly sample seven goals over a circular ring centered on the GP output, with radial bounds $[R,\, R+dr]$ and angular bounds $[-d\theta,\, d\theta]$. Typical values are $dr = 1\text{m}$ and $d\theta = 45°$. For each candidate goal, we query the model's Time-to-Arrival (TTA). If any candidate yields a TTA lower than that of the original GP output by a certain threshold, we adopt the candidate as the current goal. In the figures, we color the final selected goal in red and show the original GP output in white if it is discarded. This additional sampling ensures more robust goal selection under uncertain or noisy conditions.

S3. Method

A. Dataset for training PointNT

We train on SCAND and GND [14, 21], totaling about 20 hours of demonstrations from wheeled (Jackal) and legged (Spot) robots. We synchronize odometry and LiDAR at 5 Hz

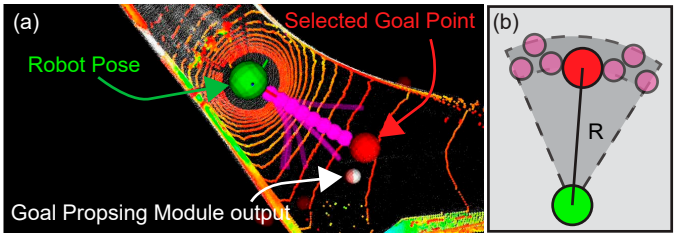

Fig. S2: Example of random sampling based multi goal proposal.

and retain only points within a 10-meter radius. We split the collected data into training and testing sets in an 80:20 ratio. Without any fine-tuning or extra robot-specific data, our model is deployed on both simulated and real robots (see Table S1). Although this dataset is smaller than the 100 hours used by ViNT [39] or NoMaD [42], our approach extends to 3D outdoor environments which is further discussed in results section.

Dataset	Robot	LiDAR	Distance (km)	Duration (min.)
SCAND	Jackal, Spot	Velodyne VLP-16	40	522
GND	Jackal, Spot	Velodyne VLP-16, Ouster OS1-32	53	668

TABLE S1: Specification of dataset.

B. Navigation Transformer (NT)

We address the challenge of mapless navigation by enabling a mobile robot to reach a specified goal without relying on a predefined map.

1) Architecture

The Navigation Transformer employs a transformer-based architecture with distinct encoders for observations and goals, and a decoder for trajectory generation:

1) **Observation Encoder**: We use a modified PointNet [32] to process each downsampled 5,000-point LiDAR scan. Unlike the original PointNet, which includes Input and Feature Transformer Networks (STN-I, STN-F), we omit these modules to reduce computational overhead. As shown in our ablation study (Sec. S5-A), removing the STN layers does not degrade performance but substantially lowers the model's complexity. After aggregating pointwise and global features, the encoder outputs an *observation token*, which is then passed both to a small MLP (for predicting the local planar transformation parameters $(\Delta x,\ \Delta y,\ \Delta\text{yaw})$) and to the Transformer decoder (along with the goal token) for multi-step trajectory and Time-to-Arrival (TTA) estimation.
2) **Goal Encoder**: A linear layer maps the goal point (x, y) to an goal token representing the target in the local frame.
3) **Transformer Decoder**: The transformer decoder takes the observation and goal tokens as inputs and outputs the predicted future trajectory and TTA.

2) SE(2) Matching Loss

One key objective of our framework is ensuring that the PointNet encoder learns robust, geometry-aware representations of the environment. To this end, we introduce an SE(2) Matching Loss that explicitly supervises the encoder's ability to capture short-horizon egomotion in the planar space.

Concretely, we extract embeddings from two LiDAR frames in the history buffer: the earliest frame $\mathbf{o}_{t-P}$ and the most recent frame $\mathbf{o}_t$. Each frame is processed by the PointNet encoder, yielding two embeddings $\mathbf{e}_{t-P}$ and $\mathbf{e}_t$. These embeddings are then concatenated and fed into a small MLP, which predicts the planar transform $(\Delta x,\ \Delta y,\ \Delta\text{yaw})$ between the two frames in the robot's local reference frame.:

$$(\Delta x_{\text{pred}}, \Delta y_{\text{pred}}, \Delta\text{yaw}_{\text{pred}}) \;=\; \text{MLP}\big(\mathbf{e}_{t-P}, \mathbf{e}_t\big). \tag{1}$$

We obtain the ground-truth relative transformation $(\Delta x_{\text{true}},\ \Delta y_{\text{true}},\ \Delta\text{yaw}_{\text{true}})$ from the robot's odometry. The SE(2) Matching Loss is defined as:

$$\begin{aligned}\mathcal{L}_{\text{SE(2)}} \;&= (\Delta x_{\text{pred}} - \Delta x_{\text{true}})^2 + (\Delta y_{\text{pred}} - \Delta y_{\text{true}})^2 \\ &\quad + (\Delta\text{yaw}_{\text{pred}} - \Delta\text{yaw}_{\text{true}})^2.\end{aligned} \tag{2}$$

3) Point Cloud Sampling

Directly processing the full raw point cloud is impractical for real-time applications, so we evaluated several downsampling methods. In our ablation study (see supplementary material), we compared random sampling, voxel grid sampling, and Farthest Point Sampling (FPS) on the SCAND dataset. The results showed that random sampling not only achieved the lowest action prediction error with shortest training time. We adopt random sampling as the default strategy in PointNT.

4) Training

The Navigation Transformer is trained using imitation learning, leveraging expert demonstrations to replicate optimal trajectories.

a) Temporal Context and Goal Point Sampling

For each trajectory τ in the dataset:

- **Temporal Context**: Select P consecutive observations $\mathcal{O}_{\text{context}} = \{\mathbf{o}_{t-P}, \mathbf{o}_{t-P+1}, \ldots, \mathbf{o}_t\}$ to capture the recent environmental and movement history.
- **Goal Point**: Sample the goal point $\mathbf{g} = (x, y)$ from one of the future time steps within the range of 0 to 40 steps ahead.

b) Label Generation

- **Future Trajectory**: Extract $H = 10$ future actions $\hat{\mathbf{a}} = \{\mathbf{a}_t, \mathbf{a}_{t+1}, \ldots, \mathbf{a}_{t+H-1}\}$ from expert demonstrations.
- **Time-to-Arrival (TTA)**: For each sample, we define TTA as the number of discrete time steps (in the expert demonstration) needed to go from the current position to the goal.
- **SE(2) Transformation**: The planar transformation between the first and last LiDAR frames in the history.

c) Loss Function

The training objective now includes both the trajectory prediction, Time-to-Arrival (TTA), and the SE(2) transformation:

$$\mathcal{L}_{\text{NavTrans}}(\phi, \psi, f) = \mathbb{E}_{\tau}\Bigg[\mathbb{E}_{t,d}\Big[\log p\Big(\hat{\mathbf{a}} \mid f\big(\psi(\mathcal{O}_{\text{context}}), \phi(\mathbf{g})\big)\Big) + \lambda_1 \log p\Big(\hat{d} \mid f\big(\psi(\mathcal{O}_{\text{context}}), \phi(\mathbf{g})\big)\Big) + \lambda_2 \mathcal{L}_{\text{SE(2)}}\Big]\Bigg] \tag{3}$$

where:

- λ_1 and λ_2 are weighting factors balancing the trajectory, TTA, and SE(2) matching losses.

C. Goal Proposing (GP) Model

The Goal Proposing Model is responsible for generating suitable navigation goals, enabling both directed and undirected exploration. It operates in two modes—directed (goal-conditioned) and undirected (autonomous exploration)—facilitating versatile deployment scenarios.

1) Architecture

The GP model employs the same architectural designs for the Observation Encoder and Goal Encoder as the Navigation Transformer, ensuring consistent feature extraction between the modules. The key architectural components are:

- **Goal Token Masking**: Introduces a binary mask variable to control goal input:
  - **Directed Mode** (mask $= 0$): Incorporates the provided goal point.
  - **Undirected Mode** (mask $= 1$): Masks the goal token, allowing autonomous goal generation.
- **Transformer Decoder**: Outputs the proposed goal position in the robot's local frame.

2) Training

The GP model is trained using imitation learning, similar to the NT, with additional mechanisms for goal generation.

a) Temporal Context and Masking

For each trajectory τ:

- Select P consecutive observations to form $\mathcal{O}_{\text{context}}$.
- Randomly set the mask variable to 0 or 1 with equal probability to work as directed and undirected modes.

b) Label Generation

- **Expert Goal**: Extract the expert goal $\hat{\mathbf{g}}$ from demonstrations.
- **Time-to-Arrival (TTA)**: For each sample, we define TTA as the number of discrete time steps (in the expert demonstration) needed to go from the current position to the goal. Specifically, if the robot reaches the goal at time step $t + d$, then TTA $= d$.
- **SE(2) Transformation**: The planar transformation between the first and last LiDAR frames in the history.

c) Loss Function

The GP model's loss function maximizes the likelihood of predicting the correct goal, Time-to-Arrival (TTA), and accurately estimating the SE(2) transformation:

$$\mathcal{L}_{\text{GoalPropose}} = \mathbb{E}_{\tau}\Bigg[\mathbb{E}_{t,d}\Big[\log p\Big(\hat{\mathbf{g}} \mid f\big(\psi(\mathcal{O}_{\text{context}}), \phi(\mathbf{g})\big)\Big) + \lambda_1 \log p\Big(\hat{d} \mid f\big(\psi(\mathcal{O}_{\text{context}}), \phi(\mathbf{g})\big)\Big) + \lambda_2 \mathcal{L}_{\text{SE(2)}}\Big]\Bigg] \tag{4}$$

where:

- λ_1 and λ_2 are weighting factors balancing the goal prediction, TTA, and SE(2) matching losses.

S4. Experimental Setup

A. Environment and Robot Details

Hardware Details: We conduct real-world tests on a Unitree Go1 quadruped and a Clearpath Jackal UGV. The Go1 uses an Ouster OS0-32 LiDAR and operates with either its built-in controller or a custom RL-based velocity-tracking controller. Due to ROS2 incompatibility with the Go1, we run our algorithm on an external laptop. Meanwhile, the Jackal, equipped with an Ouster OS1-64 LiDAR, can run the algorithm directly on its onboard computer, leveraging its default controller.

Simulation Setup: We create a digital twin of an underground tunnel in Isaac Sim and test three robot types—wheel-based(OS1-64), track-based(OS1-128), and wheel-legged(OS1-128).

Supplementary materials detail the robot, sensor configuration, and simulation settings.

B. Evaluation Metrics

To assess the performance of our navigation models, we employ several evaluation metrics, including Action Prediction Error (APE).

Action Prediction Error (APE) quantifies the discrepancy between the actions predicted by the model and the ground truth actions derived from expert demonstrations. Specifically, APE is calculated as the Mean Squared Error (MSE) between the predicted waypoints and the actual waypoints taken by the robot in the dataset. Mathematically, it is defined as:

$$\text{APE} = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{a}_{\text{pred},i} - \mathbf{a}_{\text{true},i}\|^2 \tag{5}$$

where $\mathbf{a}_{\text{pred},i}$ is the action predicted by the model for the i-th timestep, $\mathbf{a}_{\text{true},i}$ is the corresponding ground truth action, and N is the total number of predictions.

S5. Results and Analysis

A. Ablation Study

1) Comparison of Input Modalities and computation efficiency

To understand the impact of different input modalities on the performance of navigation models, we conducted an ablation study using the SCAND dataset. Specifically, we compared the performance of two variants of PointNT: PointNT(goal_pcd) was adapted to accept goal point clouds, aligning with ViNT's image-based goal inputs, whereas PointNT(goal_point) maintained its original design, using goal points as input.

Model	Input Modality	Parameters	Inference Time (s)	Action Prediction Error
ViNT	RGB	15.75M	0.0180	0.1079
PointNT(goal_pcd)	LiDAR	5.06M	0.0046	0.0785
PointNT(goal_point)	LiDAR	4.56M	0.0038	0.0631

TABLE S2: Performance comparison between PointNT variants (PointNT(goal_pcd), PointNT(goal_point)) and ViNT on the SCAND dataset. We set the temporal context size P as 5, token size as 256, and future action horizon H as 5, following the original setting in ViNT [39, 42].

In Table S2, PointNT(goal_pcd) reduces the number of model parameters from 15.75M to 5.06M (a reduction of approximately 68%), and it achieves an inference time of 0.0046 seconds per decision step—about 3.9× faster than ViNT's 0.018 seconds. Additionally, the action prediction error is reduced from 0.1079 to 0.0785, which is an improvement of roughly 27%.

PointNT(goal_point) reduces the model size further to 4.56M parameters (a 71% reduction compared to ViNT). It achieves fastest inference time of 0.0038 seconds and further lowers the action prediction error to 0.0631, demonstrating the effectiveness of PointNT's design choices for optimized robotic navigation.

All computations were conducted on an Nvidia RTX A6000 GPU and an Intel Xeon Gold 6346 CPU, which has lower computational speed than typical desktop processors. This ensures that our comparison of computational efficiency is conservative, as most desktop setups would yield even faster results.

2) Impact of SE(2) Matching Loss and Goal Orientation

In this section we analyze SE(2) matching loss and goal orientation on action prediction accuracy. The original goal information was provided only as a 2D point (x, y). However, we extend this by including a goal orientation(yaw) component to represents the heading at the goal. This aims to inform the model not only about the target location but also about the optimal approach direction.

For this ablation study, we trained on SCAND and GND datasets, evaluating Action Prediction Error (APE) on the test set with a future action horizon H=10. Table S3 summarizes the ablation results for the test APE, while Table S11 presents the corresponding training APE. Notably, the configuration that applies the SE(2) matching loss while excluding goal orientation (yaw) achieved the lowest test action prediction error (1.681), underlining the significance of core spatial relationships for reliable navigation. Although including goal orientation reduced training APE (as shown in Table S11), it increased test errors—indicating potential overfitting to the training data, as discussed in the supplementary material. Therefore, we adopt the SE(2) matching loss without incorporating the goal orientation (yaw) information in our final model, a setup that consistently outperforms other configurations in both efficiency and accuracy.

SE(2) Loss Included	Yaw Included	Input Transform	Parameters	Action Prediction Error
✓	✓	✗	0.17M	1.788
✓	✗	✗	0.17M	**1.681**
✗	✓	✗	0.17M	1.843
✗	✗	✗	0.17M	1.956
✓	✓	✓	0.97M	2.51
✓	✗	✓	0.97M	2.352
✗	✓	✓	0.97M	2.782
✗	✗	✓	0.97M	3.705

TABLE S3: Ablation study results demonstrating the impact of including SE(2) matching loss and goal orientation (yaw) information on Action Prediction Error. ✓indicates inclusion, while ✗indicates exclusion. We set the temporal context size P as 3, token size as 32, and future action horizon H as 10. We choose the model in the second row (colored in light gray) as our final configuration.

3) t-SNE Analysis of Observation Encoders

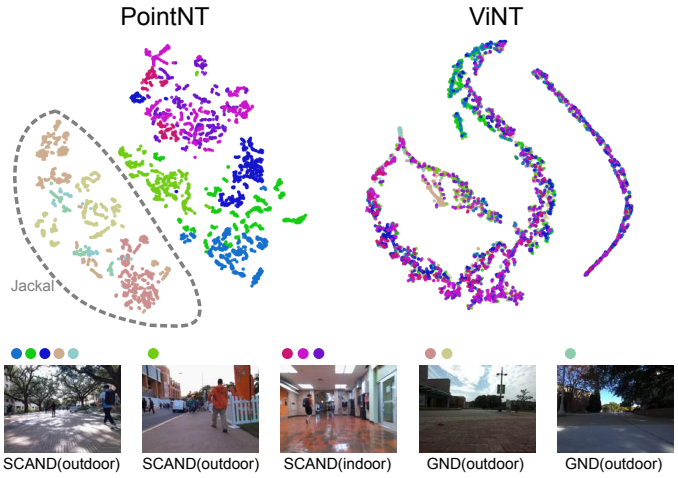

Fig. S3: t-SNE projections of observation-encoder embeddings for PointNT (left) and ViNT (right). Each color denotes a trajectory sequence (low-saturation hues = Jackal, high-saturation hues = Spot). PointNT clusters separate clearly by platform and scene type, whereas ViNT clusters overlap, indicating less discriminative spatial encoding.

To further evaluate the scene understanding capability of our model, we conducted a t-SNE analysis on the encoded representations of observation inputs generated by PointNT's observation encoder (PointNet) and ViNT's observation encoder (EfficientNet). We selected trajectories from the test dataset spanning diverse scenarios (e.g., indoor, outdoor, with varying campus). For PointNT, point clouds from the selected trajectories were fed into PointNet to produce embeddings. For

ViNT, corresponding RGB images were input into EfficientNet to generate embeddings.

As shown in Fig. S3, PointNT's embeddings form clearly separated clusters, indicating stronger discrimination between different scenes and trajectories. In contrast, ViNT's embeddings exhibit greater overlap, suggesting that RGB inputs capture fewer distinct spatial or geometric features.

B. Indoor Navigation Experiments

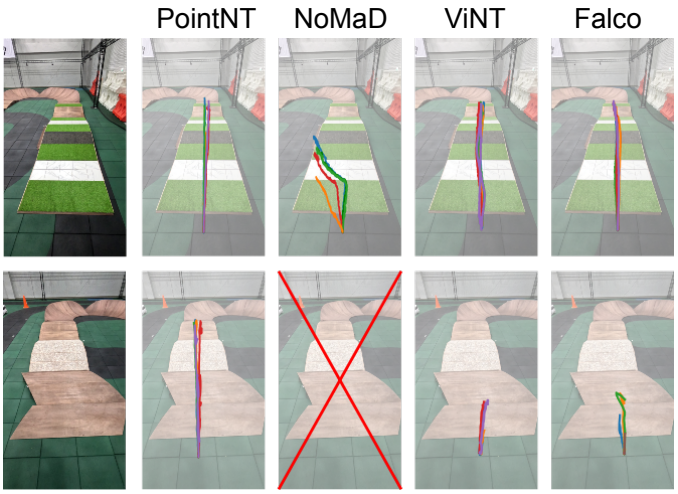

Fig. S4: Performance comparison of navigation models in indoor experiments. The upper row illustrates results in easy terrain, where PointNT, Falco, and ViNT achieved 100% success rates, while Nomad failed all trials. The lower row shows results in hard terrain, with PointNT maintaining 100% success rates, whereas Falco and ViNT failed all trials. Nomad was excluded from hard terrain experiments due to consistent failures in easy terrain.

TABLE S4: Results of indoor navigation experiments.

Algorithm	PointNT	NoMaD	ViNT	Falco
Easy	5/5	0/5	5/5	5/5

To assess Point Navigation Transformer (PointNT) under controlled indoor conditions, we conducted experiments on artificially constructed terrain and compared its performance to three state-of-the-art navigation methods—Nomad [42], ViNT [39], and Falco [50]. As shown in Fig. S4, each model ran five trials on two terrain types—one relatively flat (easy) and one featuring elevated platforms, narrow passages, and obstacles (hard)—with a constant speed of 0.4 m/s. We used FAST-LIO2 [46] for position tracking.

In easy-terrain trials, PointNT, Falco, and ViNT each successfully completed all five runs, while Nomad failed them all and was excluded from subsequent hard-terrain tests. On the harder terrain, PointNT again achieved a perfect 5/5 success rate, whereas Falco and ViNT failed every run, revealing their difficulty handling steep pitch changes. Such large pitch motions often lead to odometry drift in map-based algorithms and hinder ViNT's ability to recognize previously visited locations.

C. Directed Exploration with optimization based exploration planner

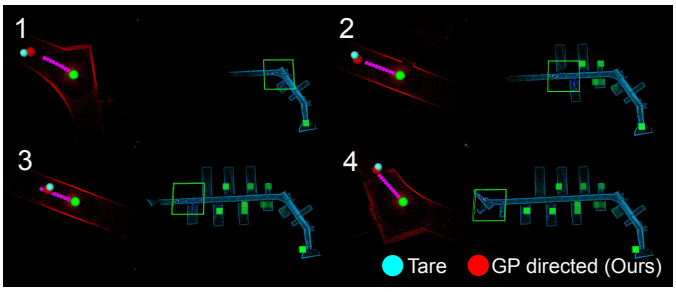

Fig. S5: We evaluate PointNT's directed GP module in simulation, using TARE to optimize and propose exploration frontiers and refined by directed GP module. The directed GP module, trained to refine random perturbations from ground-truth data, adjusts TARE's frontier points—typically centering them in hallway.

For directed exploration, we feed optimization based TARE planner-generated frontier points into our GP-directed model. As shown in Fig. S5, the GP-directed module refines TARE's frontiers to center them within hallways, producing safer trajectories and successfully navigating an underground tunnel simulation with a small, wheeled-leg robot. In the figure, red denotes the current LiDAR input, blue shows TARE's output, red points mark our refined frontiers, and green indicates the robot's current position. On the right, we illustrate the resulting exploration coverage. Notably, because our module operates purely on local-frame odometry, it easily integrates with existing map-based algorithms.

D. Undirected Exploration - indoor room exploration

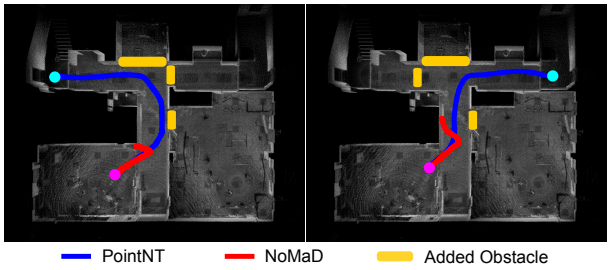

Fig. S6: We evaluate PointNT with Nomad [42] to explore a previously unseen indoor environment with varying obstacles.

As shown in Fig. S6, we compare our method with Nomad in a previously unseen real-world indoor environment using a Jackal robot with a maximum command velocity of 0.5 m/s. Neither approach receives demonstrations specific to this environment, nor do they retain local graph structures or any other memory during the experiment. We introduce obstacles to force a single route, so there is no need to store visited

TABLE S5: Traveled distance for indoor exploration: Comparison between PointNT (PointNT) and Nomad. The table shows the total distance covered when turning left and right in separate trials.

Algorithm	PointNT	Nomad
Distance (left)	22.30 m	7.01 m
Distance (right)	20.83 m	7.18 m

locations. During experiments, robot positions are logged only for visualization using FAST-LIO2 [46].

In each trial, PointNT successfully explores the environment, turning left or right around corners based on obstacle placement and covering all reachable areas via its GP module. In contrast, Nomad struggles to proceed once it encounters a narrow hallway and fails to find a viable path in this purely out-of-distribution setting. As further evidence of its robust generalization, Table S5 shows that PointNT covers substantially more distance (over 20 m) compared to Nomad (approximately 7 m) in both left and right turning scenarios.

E. Undirected Exploration - outdoor zero shot deployment

We demonstrate the performance of PointNT with simple PD controller on several challenging outdoor environments as shown in Fig. 1. We successfully conducted exploration and navigation tasks in sand beaches, mountainous terrains, and complex 3D environments both indoors and outdoors (e.g., stairs, underground tunnels) using multiple platforms including legged and wheeled robots.

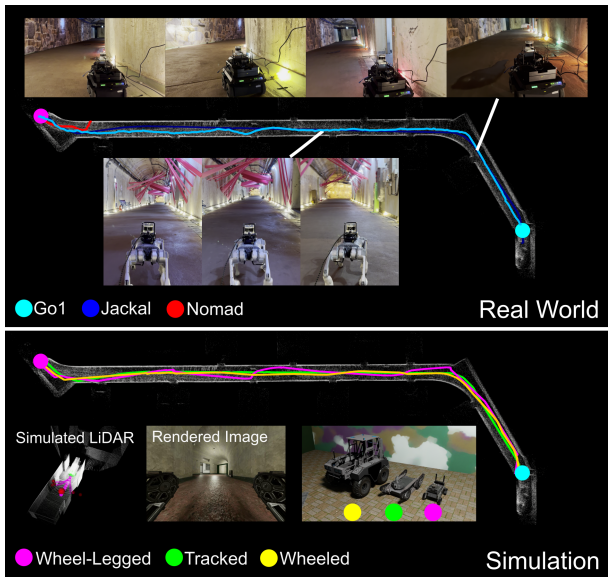

Fig. S7: We evaluate PointNT in real-world and digital twin tunnel exploration scenarios using an undirected exploration strategy. For Nomad we used Jackal platform in real-world.

In Fig. S7, we compare our method with Nomad under similar conditions to those used in our indoor exploration experiments, in a previously unseen real-world underground tunnel environment with Go1 and Jackal robots operating at speeds between 0.8 m/s and 1.0 m/s. Robot positions were logged solely for visualization purposes using FAST-LIO2 [46]. To further validate our controller, we first scanned the real-world tunnel to create a digital twin in Isaac Simulation.

Even with different embodiments, sensor types (with varying LiDAR channels: 32 or 64 in real-world deployments, and 64 or 128 in simulation), and heterogeneous robot platforms (Go1 and Jackal in the real world; wheel-legged, wheeled, and tracked in simulation), PointNT successfully explored the tunnel environment from entrance to exit without any prior input. In contrast, Nomad exhibited similar difficulties as in the indoor experiments: low lighting and sparse geometric features in the tunnel environment impaired its ability to distinguish open space from walls. This led Nomad to predict invalid exploration points and cover a much smaller area, as illustrated by the red trajectory in Fig. S7. Table S6 shows that PointNT covers substantially more distance (over 140 m) compared to Nomad (approximately 11 m) in underground tunnel experiments.

Fig. S8 visualizes the undirected exploration points generated by the GP module across various outdoor environments. By combining the NT and GP modules, we successfully traversed and navigated diverse outdoor terrains without additional fine-tuning. The distance measurements presented in Table S6 clearly demonstrate that PointNT consistently covers a large distance (over 150 m in most cases) in the tunnel environment, whereas Nomad manages only around 11 m, highlighting the

TABLE S6: Traveled distance in the underground tunnel.

	Go1	Jackal	Nomad (Jackal)	Wheel-Legged	Tracked	Wheeled
Distance	152.44 m	147.96 m	11.44 m	174.83 m	166.86 m	166.03 m

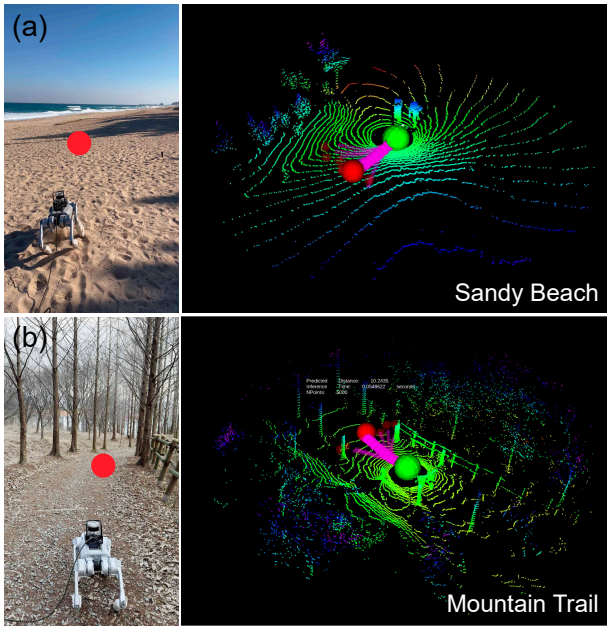

Fig. S8: Example of generated undirected exploration point using PointNT. (a) Sandy beach (b) Mountain trail course.

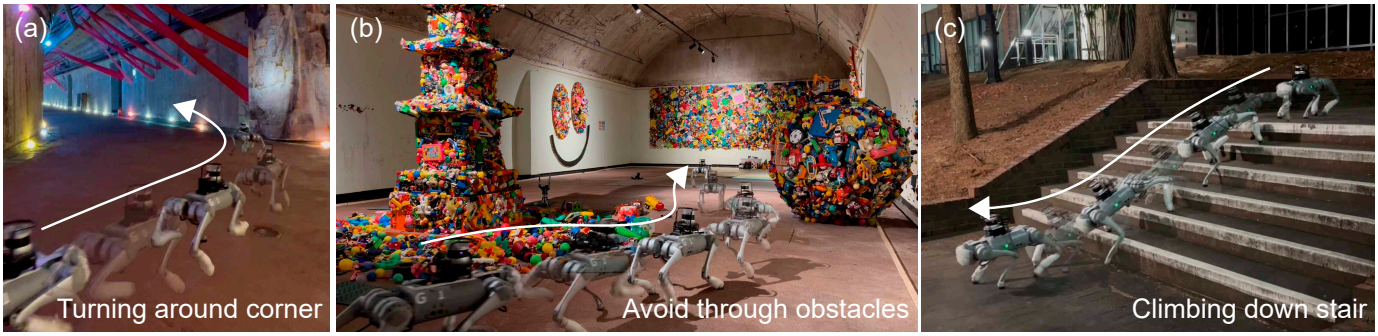

Fig. S9: We evaluate PointNT in several challenging real-world exploration scenarios using an undirected exploration strategy. (a) The robot encounters a corner with only one available path. PointNT successfully predicts a goal point on the left side, enabling the robot to turn the corner and continue exploring a broader geometric space. (b) The robot navigates through a cluttered environment with complex, unseen artifacts. Despite these objects not being present in the training dataset, PointNT generalizes effectively, demonstrating strong scene understanding. (c) The robot encounters a staircase, introducing abrupt pitch motions and oscillations in the perception sensor. Despite these challenges, PointNT successfully identifies an exploration point at the base of the stairs, guiding the robot to a safe and stable descent.

robust generalization and exploration capability of PointNT.

During outdoor experiments, we tested multiple policies, including the default controller and a Reinforcement Learning(RL) based custom controller, each exhibiting different traversability characteristics and gait patterns. Despite these differences, PointNT consistently generated exploration points, allowing seamless navigation across all environments using a single navigation module.

Additionally, during experiments, the legged robot often encountered difficulties such as getting its feet stuck in dense vegetation on mountainous and forest terrains or slipping on sandy beaches, leading to odometry failures. However, since our GP module proposes global exploration points in the robot's local frame and the NT module also operates in the local frame, these failures do not affect the robot's ability to continue exploring. Further discussion on odometry drift can be found in the supplement.

Fig. S9 highlights various navigation behaviors of PointNT in diverse scenarios. These include a one-way corner where the robot must infer future geometry to turn correctly, previously unseen artificial structures requiring obstacle avoidance, and a 3D outdoor staircase where large pitch oscillations challenge the model's ability to predict future trajectories.

S6. Limitations and Future Work

Despite strong performance, PointNT has several limitations that open avenues for future research.

- Lack of Persistent Mapping: Since PointNT operates without a global map, it may inefficiently revisit explored areas in large-scale environments. Integrating hierarchical memory or SLAM-based augmentation could improve long-horizon planning.
- Handling Dynamic Obstacles: The model implicitly learns obstacle avoidance but lacks explicit motion prediction for crowded or dynamic settings. Future work should explore social navigation frameworks for real-time interaction.
- Limited Semantic Awareness: Unlike vision-based methods, PointNT lacks semantic understanding of the environment. Incorporating semantic segmentation or vision-language models (VLMs) could improve high-level decision-making.
- Simulation-Real Gap: While PointNT performs well in real-world tests, improving policy transferability through NeRF-based simulations and pre-training fine-tuning remains a promising direction [11, 13].

Future work will focus on expanding training diversity, integrating memory-based navigation, enhancing real-time obstacle handling, and improving multi-modal perception for robust deployment in unstructured environments.

S7. Conclusion

We presented Point Navigation Transformer (PointNT), a transformer-based navigation framework that learns from raw 3D LiDAR point clouds and robot motion histories to plan reliable, goal-directed paths—enabling both directed and undirected exploration. Trained on heterogeneous datasets spanning different robotic platforms and sensor configurations, PointNT demonstrates a remarkable capacity for zero-shot generalization to real-world scenarios, including dense forests, mountainous terrain, sandy beaches, and underground tunnels. Through extensive empirical evaluations, we show that PointNT not only outperforms classical LiDAR-based planners and state-of-the-art vision-based learning methods in challenging environments but also remains computationally efficient and straightforward to deploy across platforms.

By leveraging a purely data-driven pipeline, PointNT effectively captures traversability in complex 3D scenes without explicitly maintaining or optimizing over a global map. Beyond static, short-horizon navigation, our methodology provides a foundation for more sophisticated capabilities, such as higher-level exploration strategies, integration with advanced mapping systems, and multi-modal perception fusion. We believe that

PointNT represents a significant step toward robust, general-purpose, and platform-agnostic navigation policies that extend reliably across diverse environments and robotic embodiments.

S8. Model architecture, Training Hyperparameter of PointNT

Detailed architecture and hyperparameters of PointNT and PointNet encoder is described in Table.S7.

Hyperparameter	Value
PointNT Model	
# Parameters	0.17M
Point Cloud Input	5000×3
Encoder	PointNet (modified)
Token Dimension	32
Attn. hidden dim.	128
# Attention Layers n_L	4
# Attention Heads n_H	4
Temporal Context P	3
Prediction Horizon H	10
Goal Encoding MLP layers	(2, 64)
PointNT Training	
# Epochs n_{ep}	50
Batch Size	512
Learning Rate	5×10^{-4}
Optimizer	AdamW [26]
Warmup Epochs	4
LR Schedule	Cosine
Scheduler Period	10
Compute Resources	A6000
Training Time	30 hours
PointNet Encoder	
Input Dimensions	3
Output Dimensions	128
Number of Conv Layers	3
Convolutional Layers	$3 \to 64 \to 128 \to 128$
Batch Normalization	True
Activation Function	ReLU
Global Feature	True
Feature Transform	False
Input Transform	False
SE2 Predictor	
Channels	(256, 128, 64, 32, 3)

TABLE S7: Hyperparameters for training PointNT, including detailed PointNet Encoder configurations.

S9. Robot specification

For real world robot, we used Unitree Go1 and Clearpath Jackal.

Go1 : Quadrupedal robot equipped with Ouster OS0-32.

Jackal : Wheeled robot equipped with Ouster OS1-64.

For real simulated robot, we used in house developed three types of UGVs.

Tracked UGV : Mid size tracked robot equipped with Ouster OS1-128.

Wheel-Legged UGV : Small size wheel-Legged robot equipped with Ouster OS1-128.

Wheeled UGV : Big size wheeled robot equipped with Ouster OS1-64.

S10. Impact of Point Cloud Sampling Methods

To evaluate the impact of different point cloud sampling methods on the performance of our navigation model, we conducted an ablation study using the SCAND dataset, focusing on three sampling techniques: random sampling, voxel grid sampling, and Farthest Point Sampling (FPS). Initially, each raw LiDAR point cloud was truncated at a distance of 10 meters to concentrate on the most relevant spatial information. Subsequently, random sampling selected a subset of points arbitrarily, voxel grid sampling divided the space into uniform voxels and selected representative points from each, and FPS iteratively chose points that maximized spatial diversity. The results, summarized in Table S8, indicate that random sampling achieved the lowest action prediction error (0.0357) and the shortest training time (3.56 hours), outperforming Farthest Point Sampling (FPS) with an error of 0.0375 and training time of 34.46 hours, and voxel grid sampling with an error of 0.0401 and training time of 20.06 hours.

TABLE S8: Comparison of Action Prediction Error and Training Time (lower is better) depending on Point Cloud Sampling Methods.

Sampling Method	**Action Prediction Error**	**Training Time**
Random Sampling	0.0357	3.56 hr
Voxel Grid Sampling	0.0401	20.06 hr
Farthest Point Sampling (FPS)	0.0375	34.46 hr

This demonstrates that random sampling effectively preserves essential spatial and geometric features critical for accurate navigation by maintaining a diverse and representative set of points while also being computationally efficient with the shortest training time. In contrast, while FPS also aims to maximize spatial diversity, it results in a slightly higher prediction error and requires the longest training time. Voxel grid sampling, although more structured, shows the highest action prediction error among the tested methods and has a moderate training time. Consequently, random sampling is adopted as the default sampling strategy in PointNT, balancing computational efficiency with superior navigational accuracy.

S11. Training of Reinforcement Learning policy

To train the Reinforcement Learning(RL) policy of quadrupedal robot locomotion policy for rough terrain, we employ PPO [35] as our policy gradient algorithm within the Isaac Gym simulator [27].

We used a concurrent training architecture to estimate proprioceptive value while learning the policy [10].

For the training curriculum we used only pyramid stair and slope terrain from [34] with 4096 parallel environments. We largely follows motor gain, reward and randomization from [28] to enable robust sim-to-real transfer.

For the linear body velocity, body height, foot height and contact probability we use the concurrent estimation strategy using ground truth data from simulation. We also note that we give true value to the critic network during training the policy.

TABLE S9: Observation Types and Dimensions

Observation Type	Input	Dim.
Proprioception	Linear body velocity estimation	3
	Angular body velocity	3
	Body height estimation	1
	Foot height estimation	4
	Contact probability estimation	4
	Command	3
	Projected gravity vector	3
	Action	12
	Joint position	12
	Joint velocity	12
	Action (2 time steps ago)	12
	Joint position (2 time steps ago)	12
	Joint velocity history (2 time steps ago)	12
	Action (4 time steps ago)	12
	Joint position (4 time steps ago)	12
	Joint velocity history (4 time steps ago)	12

TABLE S10: Hyperparameters for PPO

Parameter	Value
horizon length (dt : 0.02)	50
learning rate	3.0 E-4
kl threshold	0.008
discount factor	0.99
entropy coef	0.001
clip ratio	0.2
batch size	204800
mini batch size	40960

S12. Additional Training and Deployment Details

For training, we used an Nvidia RTX A6000 GPU (48Gb VRAM) and an Intel Xeon Gold 6346 CPU (3.1GHz, 32 Core), which has a high core count but lower core speed.

For deployment, we tested our method in various environments with diverse properties, as shown in Fig. S10. Despite the limited interaction data with diverse wild terrains in the training dataset, our approach effectively captures important geometric scene understanding, enabling zero-shot deployment across a wide range of terrains.

S13. Additional Analysis

A. Impact of SE(2) Loss and Goal Yaw

To further investigate the effects of SE(2) loss and yaw information under varying model configurations, we conducted an additional ablation study with a temporal context size P of 4 and a token size of 64. The results are presented in Table S11. Consistent with the main findings, the configuration that includes SE(2) loss while excluding yaw information achieved the lowest test Action Prediction Error (APE) of 1.681, as highlighted in light gray. This configuration also demonstrated a balanced training APE of 0.7675, indicating effective generalization without overfitting. In contrast, configurations that incorporated yaw information tended to have lower training APE but higher test APE, which demonstrates overfitting when yaw is included. These supplementary results support our decision to adopt the SE(2) loss without yaw in the final model, ensuring optimal performance across different model settings.

Fig. S10: Tested terrain for undirected exploration using PointNT. (a) Underground Tunnel (b) Mountain Trail (c) Sandy beach (d) Forest type 1 (e) Forset type 2

SE(2) Loss	Yaw	Input Transform	APE (Train)	APE (Test)
✓	✓	✗	0.7083	1.788
✓	✗	✗	0.7675	**1.681**
✗	✓	✗	0.6963	1.843
✗	✗	✗	0.7182	1.956
✗	✓	✓	0.1419	2.782
✗	✗	✓	0.1009	3.705
✓	✓	✓	0.7166	2.51
✓	✗	✓	0.8283	2.352

TABLE S11: Ablation study results demonstrating the impact of including SE(2) loss and yaw information on Action Prediction Error. ✓indicates inclusion, while ✗indicates exclusion. We set the temporal context size P as 4 and token size as 64. We choose the model in second row, colored in light gray.

B. Learned traversability

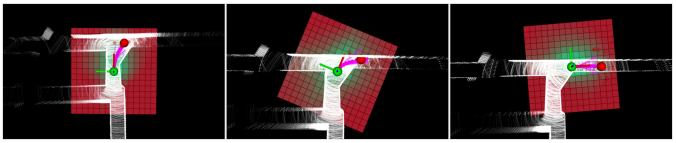

Fig. S11: Implicit traversability map generated by PointNT.

As shown in Fig. S11, our model internally constructs a traversability map via its learned Time-to-Arrival (TTA). Because PointNT predicts both navigation commands and the distance from the robot to a goal, we can systematically vary the goal point around the robot's position to infer implicit traversability from LiDAR data.

When turning a corner in the simulated tunnel, for instance, we query TTA values on a 15×15, 1m step grid in the robot's local frame. The model predicts a short TTA along the feasible path—where the robot can easily move—and large TTA for

regions beyond the walls, implicitly capturing which areas are difficult or impossible to traverse.

This can also lead to interesting combination with model based path planner which requires traversability map since PointNT can give local traversability map with point cloud data only.

C. Robustness to Odometry Drift

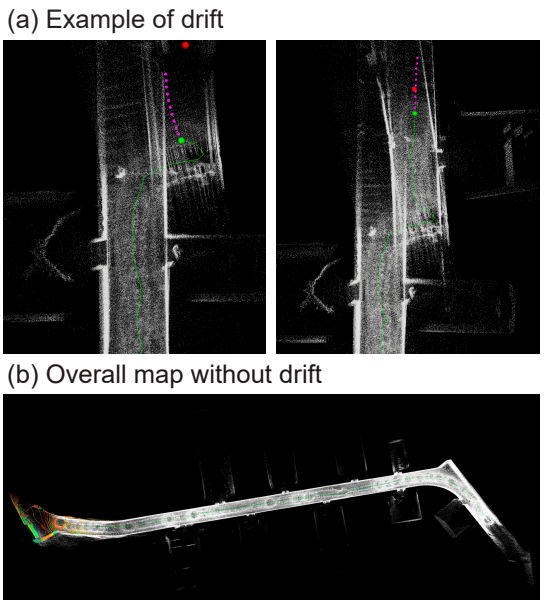

Fig. S12: (a) PointNT successfully reaches the goal despite drift in the external odometry algorithm. (b) Example map of the tested environment without drift.

As shown in Fig. S12, PointNT is robust to odometry drift since it operates in a fully mapless manner without relying on odometry. This significantly expands the applicability of our approach—while PointNT can seamlessly integrate with odometry-based algorithms, it remains unaffected by their failures, preventing the robot from getting stuck due to odometry drift.

D. Further use-case : scanning inside the room

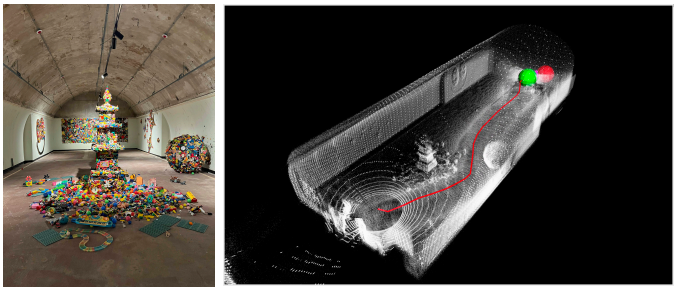

Fig. S13: Scanning of the indoor environment while avoiding obstacles.

As shown in Fig. S13, since PointNT uses LiDAR point clouds and odometry as inputs, it can be easily integrated with other LiDAR-based odometry algorithms. This enables PointNT to efficiently explore complex indoor environments with obstacles while accurately scanning the surroundings.