SuReNav: Superpixel Graph-based Constraint Relaxation for Navigation in Over-constrained Environments

Keonyoung Koh, Moonkyeong Jung, Samuel Seungsup Lee and Daehyung Park[†]

Abstract— We address the over-constrained planning problem in semi-static environments. The planning objective is to find a best-effort solution that avoids all hard constraint regions while minimally traversing the least risky areas. Conventional methods often rely on pre-defined area costs, limiting generalizations. Further, the spatial continuity of navigation spaces makes it difficult to identify regions that are passable without overestimation. To overcome these challenges, we propose SuReNav, a superpixel graph-based constraint relaxation and navigation method that imitates human-like safe and efficient navigation. Our framework consists of three components: 1) superpixel graph map generation with regional constraints, 2) regional-constraint relaxation using graph neural network trained on human demonstrations for safe and efficient navigation, and 3) interleaving relaxation, planning, and execution for complete navigation. We evaluate our method against state-of-the-art baselines on 2D semantic maps and 3D maps from OpenStreetMap, achieving the highest human-likeness score of complete navigation while maintaining a balanced trade-off between efficiency and safety. We finally demonstrate its scalability and generalization performance in real-world urban navigation with a quadruped robot, Spot.

I. INTRODUCTION

Consider the problem of over-constrained planning in semi-static environments. The planning is to find an optimal plan that satisfies all hard constraints while possibly violating less critical soft constraints. For example, a mobile robot navigating a park may encounter a blocked path on a pedestrian road (see Fig. 1). Although typical navigation rules dictate that the robot should not traverse the lawn, it can be more efficient to slightly deviate onto the grass rather than a long detour or backtracking. Such environmental changes frequently occur, leading to over-constrained planning in real-world settings—scenarios we define as semi-static environments.

Traditional approaches, often framed as constraint satisfaction problems, find the best solution by minimizing constraint violations. Common approaches include the minimum violation problem (MVP) [1], minimum constraint removal [2], and violation cost estimation [3]. For instance, Kris Hauser's minimum constraint displacement method improves the current best planning solution by selecting a subset of obstacles
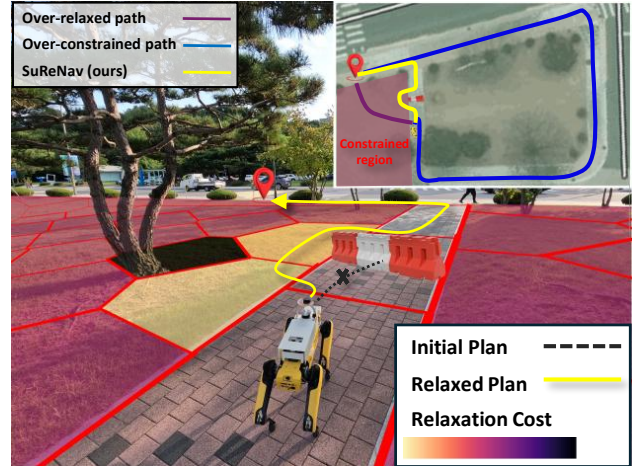
Fig. 1: A capture of regional-constraint relaxation experiment in an over-constrained navigation environment. A quadruped robot pre-plans a path on a known map, and when the path is blocked, our method finds a new path by relaxing a compact area with the least risky constraints to enable safe and efficient human-like navigation.

to displace [4]. Kim et al. relax a regional constraint that makes a current plan locally infeasible in temporal logic-based planning [5]. Despite these efforts, a major limitation is their reliance on uniform or pre-defined priorities for constraint selections, which are hard to generalize to complex environments with diverse properties of intricate navigation contexts (e.g., distinguishing between a necessary road crossing to follow a sidewalk and unsafe jaywalking). Therefore, desired approaches should relax constraints in a manner that simultaneously considers safety and efficiency.

Another challenge is that regional constraints should accurately represent to relax necessary and not overestimated space. Typical navigation approaches distinguish passable or impassable regions only, based on visual segments [6]. The representation is hard to indicate partial regional constraints, where the relaxation of entire segments, such as road or grass, potentially generates the overly relaxed shortest paths. Although pixel-based representation is another option, spatial continuity complicates the definition of *unit* constraints.

In this context, we propose SuReNav, a constrained-region-aware, superpixel graph-based relaxation and navigation method. SuReNav aims to generate human-like complete navigation paths without requiring predefined or overestimated regional constraints. To address these, our method comprises three key components: 1) superpixel graph-map generation, 2) graph neural network (GNN)-based regional-

constraint relaxation, 3) interleaving relaxation, planning, and execution. The superpixel graph models the navigation space with compact segments (i.e., superpixels) that preserve label consistency and boundaries within a graph structure. This graph representation enables the relaxation of less critical constraints by coupling a GNN-based relaxation-cost estimator with a differentiable A* planner, which propagates risks and task objectives in a manner inspired by human demonstrations. Finally, the interleaved relaxation, planning, and execution process ensures safe, efficient, and complete navigation.

We validate our method through quantitative and qualitative analyses in semi-static simulated and real-world navigation studies. For the statistical evaluation, we introduce a new navigation benchmark generating OpenStreetMap-based navigation scenarios with dynamic environmental changes reflecting semi-static conditions. Experimental results show that our proposed method outperforms four representative baselines in terms of human-likeness while achieving comparable completeness, safety, and efficiency. Finally, we deploy the system on a quadruped robot in a university campus demonstrating its scalability and generalizability under real-world operating conditions.

Our main contributions are as follows:
- We formalize a novel constrained-region relaxation and planning problem for reactive path planning in semi-static environments.
- We introduce a graph-based constrained-region relaxation cost estimator that enables safe and efficient human-like complete navigation.
- We propose a joint training framework that learns both the cost estimator and a neural path planner by leveraging the differential A* algorithm.
- We validate SuReNav through extensive evaluations in simulation and real-world navigation scenarios.

II. RELATED WORK

Traditional navigation planning methods typically require either predefining traversable regions [7] or estimating regional traversability [8] before or during navigation. While these approaches explicitly model spatial and semantic constraints, their reliance on human intervention limits generalization to novel environments, often reducing completeness or optimality [9]. Alternatively, recent end-to-end methods infer actions directly from observations without hand-crafted regional constraints by leveraging large-scale datasets [10], [11], [12]. However, the data-driven mappings lack interpretable mechanisms for preserving explicit constraints during planning [13].

On the other hand, a number of navigation planners address the MVP, even in over-constrained environments, by modeling semantically recognized navigation constraints as traversability costs [1], [14]. These cost-based planners assign violation costs to individual constraints and optimize navigation by minimizing the total cost trade-offs across constraints. However, cost or heuristic metrics often fail to capture complete context awareness [15] and struggle with

precise spatial constraint modeling [16]. Thus, developing robust constraint-to-cost mappings remains an open challenge.

The advent of search-guided learnable planners offers a novel direction for addressing over-constrained planning problems. Recent approaches learn costs and heuristics with neural networks [17], [18], or perform efficient searches over traversal-cost graphs generated by GNNs [19]. By learning cost functions or heuristics from data, these methods encode context-specific risks while preserving the interpretability and completeness of search-based planning. Building on this paradigm, our model leverages data-driven learning to estimate violation costs for regional constraints defined on a superpixel-based region graph, thereby enabling constraint-aware relaxation and safe, efficient navigation in over-constrained environments.

III. PRELIMINARIES

We review the *differentiable graph-based A* planner* proposed by GraphMP [19], to enable end-to-end training of a relaxation cost estimator using the search result from the region–adjacency graph $G = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ and $\mathcal{E}$ denote the node and edge sets, respectively.

GraphMP extends Neural A* [17]'s grid-based differentiable A* search module to general graphs. The method maintains three containers: the open-set mask $\mathbf{o} \in \mathbb{B}^{|\mathcal{V}|}$, the closed-set mask $\mathbf{c} \in \mathbb{B}^{|\mathcal{V}|}$, and the accumulated path costs $\mathbf{g} \in \mathbb{R}^{|\mathcal{V}|}$, where the entries with 1 denote the included nodes.

At each expansion step with the open-set mask $\mathbf{o}$, the search process first forms a temperature-controlled soft priority over the open set with heuristic values $\mathbf{h} \in \mathbb{R}^{|\mathcal{V}|}$ and then returns a one-hot selection:

$$\mathbf{v}_{\text{sel}} = I_{\max} \left(\frac{\exp(-(\mathbf{g} + \mathbf{h})/\lambda) \odot \mathbf{o}}{\exp(-(\mathbf{g} + \mathbf{h})/\lambda)\mathbf{o}} \right), \qquad (1)$$

where $\lambda > 0$ is a temperature, $\odot$ denotes the Hadamard product, and $I_{\max}$ produces a minimal-distance node mask $\mathbf{v}_{\text{sel}} \in \mathbb{B}^{|\mathcal{V}|}$.

After selecting $\mathbf{v}_{\text{sel}}$, differentiable graph-based A* updates containers and identifies undiscovered neighbors $\mathbf{v}_{\text{nbr}} \in \mathbb{B}^{|\mathcal{V}|}$ using the adjacency matrix $\mathbf{A} \in \mathbb{B}^{|\mathcal{V}| \times |\mathcal{V}|}$:

$$\mathbf{o} \leftarrow \mathbf{o} - \mathbf{v}_{\text{sel}}, \ \ \mathbf{c} \leftarrow \mathbf{c} + \mathbf{v}_{\text{sel}}, \mathbf{v}_{\text{nbr}} = \mathbf{A}\mathbf{v}_{\text{sel}} \odot (\mathbf{1} - \mathbf{c}). \quad (2)$$

where $\mathbf{1}$ for all-one vector. Upon the identification, the search module then computes the accumulated costs $\mathbf{g}'$ associated with the path containing the selected node $\mathbf{v}_{\text{sel}}$,

$$\mathbf{g}' = \mathbf{g} \odot \mathbf{v}_{\text{sel}} + \mathbf{W}\mathbf{v}_{\text{sel}}, \qquad (3)$$

where a weighted adjacency matrix $\mathbf{W} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$. Finally, the module updates the accumulated costs $\mathbf{g}$ comparing the node-wise cost minimums and expands the open-set mask $\mathbf{o}$ including all eligible neighbors following Eq. (7) and (8) in [19] (see [19] for the full formulation).

GraphMP jointly trains the neural heuristic estimator and the search procedure end-to-end. We exploit its graph-based gradient flow to learn the relaxation-cost estimator comparing the output plan and human demonstration paths.
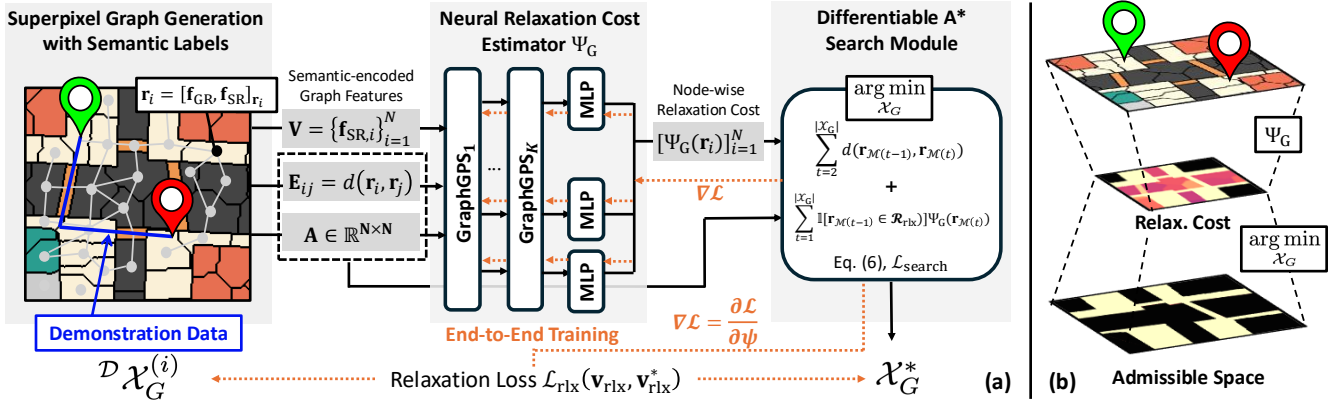
Fig. 2: Overview architecture of SuReNav that automatically relaxes regional constraints while planning a graph path $\mathcal{X}_G$ in semi-static navigation environment. (a) In the training phase, SuReNav generates a superpixel graph from a 2-dimensional map with safety-relevant semantic features. A neural relaxation cost estimator Ψ_G then computes a node-wise relaxation cost, which is incorporated as an optimization term in a differentiable search process. This enables end-to-end training of the model from human demonstrations. (b) During the planning phase, the estimator generates node-wise relaxation costs, and a discrete graph search process selects regions to relax, inducing an admissible space to facilitate path planning.

IV. METHOD

We introduce our interleaving planning and execution with regional constraint relaxation in semi-static environments.

A. Problem formulation

Consider a navigation space $\mathcal{R}$ partitioned into regions with no, soft, and hard constraints: $\mathcal{R} = \mathcal{R}_{\text{free}} \cup \mathcal{R}_{\text{soft}} \cup \mathcal{R}_{\text{hard}}$. We aim to find a safe and efficient route $\mathcal{X} = \{\mathbf{x}(s)|s \in [0,1]\}$, where $\mathbf{x}(s)$ is a configuration at the phase s, that traverses the least-critical (i.e., relaxed) region $\mathcal{R}_{\text{rlx}} \subseteq \mathcal{R}_{\text{soft}}$, yielding the admissible set $\mathcal{R}_{\text{safe}} = \mathcal{R}_{\text{free}} \cup \mathcal{R}_{\text{rlx}}$. We formulate the route search and relaxation problem as

$$\underset{\mathcal{X}, \mathcal{R}_{\text{rlx}} \subseteq \mathcal{R}_{\text{soft}}}{\arg\min} \int_0^1 \|\dot{\mathbf{x}}(s)\| ds + \Psi_{\text{cost}}(\mathcal{R}_{\text{rlx}}) \quad (4)$$
$$\text{such that} \quad \mathbf{x}(s) \in \mathcal{R}_{\text{free}} \cup \mathcal{R}_{\text{rlx}} \ \ \forall s \in [0,1],$$
$$\mathbf{x}(0) = \text{start}, \mathbf{x}(1) = \text{goal},$$

where Ψ_{cost} assigns a non-negative relaxation penalty cost ($\in \mathbb{R}^+$) to traversing the relaxed region $\mathcal{R}_{\text{rlx}}$. This problem is generally NP-hard due to the non-additive coupling between the continuous length and the discrete choice of relaxations.

To approximate the problem, we reformulate Eq. (5) leveraging a graph-based region abstraction. The graph $G = (\mathcal{V}, \mathcal{E})$ consists of region nodes $\mathcal{V} = \{\mathbf{r}_i\}_{i=1}^N$ and region-adjacency edges $\mathcal{E} = \{(\mathbf{r}_i, \mathbf{r}_j)|i \neq j, \mathcal{H}(\mathbf{r}_i, \mathbf{r}_j) \geq \tau\}$ where each node $\mathbf{r}_i \subset \mathcal{R}_{\text{free}} \cup \mathcal{R}_{\text{soft}}$ denotes a spatial segment with centroid $c(\mathbf{r}_i) \in \mathbb{R}^2$, $\mathcal{H}$ measures boundary affinity, and τ is the affinity threshold for local motion. We then project the continuous path $\mathcal{X}$ onto G as a sequence of region memberships:

$$\mathcal{X}_G = [\mathbf{r}_{\mathcal{M}(1)}, ..., \mathbf{r}_{\mathcal{M}(|\mathcal{X}|)}], \ \mathcal{X}[t] \in \mathbf{r}_{\mathcal{M}(t)}, \quad (5)$$

where $t \in [1, |\mathcal{X}|]$ and $\mathcal{M} : \{1, ..., |\mathcal{X}|\} \to \{1, ..., N\}$, which maps each configuration $\mathcal{X}[t]$ to its containing region node. Equivalently, the projected path connects centroids

$\{c(\mathbf{r}_{\mathcal{M}(t)})\}_{t=1}^{|\mathcal{X}|}$ on the graph map. We then represent the graph-based optimization problem as

$$\underset{\mathcal{X}_G}{\arg\min} \quad \sum_{t=2}^{|\mathcal{X}_G|} d(\mathbf{r}_{\mathcal{M}(t-1)}, \mathbf{r}_{\mathcal{M}(t)})$$
$$+ \sum_{t=1}^{|\mathcal{X}_G|} \mathbb{1}\left[\mathbf{r}_{\mathcal{M}(t)} \in \mathcal{R}_{\text{rlx}}\right] \Psi_G(\mathbf{r}_{\mathcal{M}(t)})) \quad (6)$$
$$\text{such that} \quad \mathbf{r}_{\mathcal{M}(t)} \in \mathcal{R}_{\text{free}} \cup \mathcal{R}_{\text{rlx}} \ \ \forall t \in [1, |\mathcal{X}_G|],$$
$$(\mathbf{r}_{\mathcal{M}(t-1)}, \mathbf{r}_{\mathcal{M}(t)}) \in \mathcal{E} \ \ \forall t \in [1, |\mathcal{X}_G|],$$
$$\mathbf{r}_{\mathcal{M}(1)} = \mathbf{r}_{\text{start}}, \mathbf{r}_{\mathcal{M}(|\mathcal{X}_G|)} = \mathbf{r}_{\text{goal}},$$
$$\mathcal{R}_{\text{rlx}} = \mathcal{R}_{\text{soft}} \cap \mathcal{X}_G,$$

where d is a geodesic distance measure between the center of two nodes and Ψ_G is a relaxation cost function on the graph G. For notational convenience, we denote the objective function in Eq. (6) by $\mathcal{L}_{\text{search}}$.

Fig. 2 shows the overall architecture of our method. It optimizes navigation by constructing a superpixel-based graph G, training Ψ_G, and applying a differentiable A* algorithm (see Sec. IV-C). The process consists of 1) superpixel graph construction (Sec. IV-B), 2) graph-based path planning with constraint relaxation (Sec. IV-C), and 3) reactive planning that interleaves path search and constraint relaxation (Sec. V).

B. Superpixel Graph Generation with Semantic Labels

We construct a superpixel graph map $G = (\mathcal{V}, \mathcal{E})$ for human-like relaxation region selection and path planning, where $\mathcal{V}$ and $\mathcal{E}$ denote superpixel-based node regions and region-adjacency edges, respectively. Superpixels partition an image-based map into over-segmented regions that do not cross object boundaries [20]. By constraining pixels within the same region, superpixels enable homogeneous semantic-labels, improving prediction consistency and sharpening relaxation-region boundaries. Further, their compact

graph representation offers controllable granularity with near-uniform regions that support stable planning [21].

To construct G, we partition the traversable space $\mathcal{R}_{\text{free}} \cup \mathcal{R}_{\text{soft}}$ into N node regions $\mathcal{V} = \{\mathbf{r}_i\}_{i=1}^N$ using SLIC [22]—a K-Means based pixel clustering algorithm. Note that all locations within each node region $\mathbf{r}_i$ share the same semantic label. Each node $\mathbf{r}_i \in \mathcal{V}$ contains a pair of vectors $[\mathbf{f}_{\text{GR}}, \mathbf{f}_{\text{SR}}]_{\mathbf{r}_i}$:

- $\mathbf{f}_{\text{GR}}$ contains geometric information, the center coordinate $c(\mathbf{r}_i) \in \mathbb{R}^2$ for distance computation in planning.
- $\mathbf{f}_{\text{SR}}$ contains semantic information, including the semantic label, a start-region indicator, and a goal-region indicator of the current node $\mathbf{r}_i$:

$$\mathbf{f}_{\text{SR}} = [\Phi_{\text{label}}(\mathbf{r}_i), \mathbb{1}[\mathbf{x}_{\text{start}} \in \mathbf{r}_i], \mathbb{1}[\mathbf{x}_{\text{goal}} \in \mathbf{r}_i]], \quad (7)$$

where $\Phi_{\text{label}} : \mathcal{V} \to \mathbb{B}^M$ is a semantic classifier returning a one-hot vector of dimension M.

Each edge $e_{ij} \in \mathcal{E}$ contains a distance measure between two geometrically adjacent regions; $e_{ij} = d(\mathbf{r}_i, \mathbf{r}_j)$.

After the initial construction of G, we incrementally update the graph by re-segmenting each region with new observations and re-constructing edges. This update mechanism maintains the label consistency and compactness of the graph allowing interleaving planning and execution.

C. Graph-based Path-Planning and Relaxation

We present a GNN-based regional constraint-relaxation method that couples a neural relaxation-cost estimator with a differentiable A* planner to generate preferable paths. To imitate human-like relaxed path generation, we learn the relaxation penalty Ψ_G in Eq. (6) from human navigation demonstrations collected in over-constrained environments so that the planner emulates human-like relaxation behavior. Below, we describe the demonstration data and the two learning stages in detail.

1) Expert demonstrations: We collect a N_D number of expert demonstrations $\mathcal{D} = \{^{\mathcal{D}}\mathcal{X}^{(i)}\}_{i=1}^{N_D}$, where each sequence $^{\mathcal{D}}\mathcal{X}^{(i)} = [\mathbf{x}(0), ..., \mathbf{x}(1)]$ is a sequence of 2-D coordinates. Each path illustrates how a human expert simultaneously traces a continuous path and, when necessary, passes through one or more regions labeled with soft constraints $\mathcal{R}_{\text{soft}}$. We then project the demonstrations $\mathcal{D}$ onto the graph map G by converting each path $^{\mathcal{D}}\mathcal{X}^{(i)}$ into $^{\mathcal{D}}\mathcal{X}_G^{(i)} = [\mathbf{r}_{\mathcal{M}(0)}, ..., \mathbf{r}_{\mathcal{M}(|^{\mathcal{D}}\mathcal{X}^{(i)}|)}]$ where $^{\mathcal{D}}\mathcal{X}^{(i)}[t] \in \mathbf{r}_{\mathcal{M}(t)}$. The data collection detail is available in Sec. VI.

2) Relaxation cost estimation: We model a neural relaxation-cost estimator $\Psi_G : \mathcal{V} \to \mathbb{R}^+$ to find a path balancing the path length and relaxation penalty via a differentiable graph-based A* planner. The estimator is a GNN model using a superpixel graph $G_s = (\mathcal{V}_s, \mathcal{E})$, which is the priorly constructed G with a part of node features, $\mathbf{f}_{\text{SR}}$. As shown in Fig. 2, the model consists of a stack of GPS layers [23], where each layer is a hybrid layer of a message-passing neural network (MPNN) and a global attention network that captures long-range dependency in a graph. Following GraphGPS [23], we use GatedGCN [24] as the MPNN layer and Transformer [25] as the global attention layer.

The GPS layers update encoded node and edge features for the node-wise cost estimation. Let $\mathbf{V}^k \in \mathbb{R}^{N \times (M+2)}$ and $\mathbf{E}^k \in \mathbb{R}^{|\mathcal{E}|}$ be the encoded node features and edge features, respectively. The updated process is

$$\mathbf{V}^{k+1}, \mathbf{E}^{k+1} = \text{GPS}^k(\mathbf{V}^k, \mathbf{E}^k, \mathbf{A}), \quad (8)$$

where $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix of the graph G_s and $k \in \{1, ..., K\}$ is the number of GPS layers. Finally, applying a node-wise shared multi-layer perceptrons (MLPs) with a softplus activation, our cost estimator outputs node-wise scores $\{\Psi_G(\mathbf{r}_i)\}_{i=1}^N$. Note that we zero-mask the scores for nodes in free space $\mathbf{r}_i \in \mathcal{R}_{\text{free}}$ not to assign relaxation cost for non-constrained regions.

3) GNN-based Planning with Relaxation Cost: Our method searches for an optimal path that may traverse constrained regions while explicitly accounting for relaxation costs via differentiable graph-basd A* planner. We incorporate these costs by augmenting the accumulated path costs to

$$\mathbf{g}^{\text{ext}} = \mathbf{g} + [\Psi_G(\mathbf{r}_i)]_{i=1}^N. \quad (9)$$

The extended costs modify the planner's selection rule, replacing Eq. (1) as

$$\mathbf{v}_{\text{sel}} = I_{\max}\left(\frac{\exp(-(\mathbf{g}^{\text{ext}} + \mathbf{h})/\lambda) \odot \mathbf{o}}{\exp(-(\mathbf{g}^{\text{ext}} + \mathbf{h})/\lambda)\mathbf{o}}\right). \quad (10)$$

We then add the region entry penalty of the selected node parent by redefining Eq. (3) as

$$\mathbf{g}' = \mathbf{g}^{\text{ext}} \odot \mathbf{v}_{\text{sel}} + \mathbf{W}\mathbf{v}_{\text{sel}}. \quad (11)$$

As described in Sec. III, these modifications allow GraphMP to find the minimum-cost path $\mathcal{X}^*$ under relaxation penalties across $\mathcal{R}_{\text{free}} \cup \mathcal{R}_{\text{soft}}$.

D. End-to-End Training

We learn the relaxation cost estimator Ψ_G from human demonstration $\mathcal{D}$ by leveraging the differential A* planner. As shown in Fig. 2, the differentiable search module propagates the relaxation error that we will describe in Eq. (12) backward enabling end-to-end training of the cost estimator Ψ_G. We further update Ψ_G introducing a mini-batch loss $\mathcal{L}_{\text{batch}}$ to minimize false positive and negative relaxations.

Consider a batch $\mathcal{B} \subset \mathcal{D}$. The false region relaxation loss is a measure of how much the A* planner traverses the regions, with soft constraints, that were not explored in each demonstration $^{\mathcal{B}}\mathcal{X}^{(i)}$. The A* planner yields an explored constraint region mask $\mathbf{v}_{\text{rlx}}^{(i)} \in \mathbb{B}^N$ that is the intersection of the closed set $\mathbf{c}$ and $\mathcal{R}_{\text{soft}}$. From each demonstration, we also extract a mask $^{\mathcal{B}}\mathbf{v}_{\text{rlx}}^{(i)} \in \mathbb{B}^N$ for relaxed regions with constraints in $\mathcal{R}_{\text{soft}}$. Then, we measure the false relaxation as the weighted sum of false positives and negatives,

$$\mathcal{L}_{\text{rlx}}(\mathbf{v}_{\text{rlx}}, \mathbf{v}_{\text{rlx}}^*) = \frac{w_{\text{fp}}\mathbf{v}_{\text{rlx}}^\top(\mathbf{1} - \mathbf{v}_{\text{rlx}}^*) + w_{\text{fn}}(\mathbf{1} - \mathbf{v}_{\text{rlx}})^\top \mathbf{v}_{\text{rlx}}^*}{1 + \|\mathbf{v}_{\text{rlx}}^*\|_1} \quad (12)$$

where w_{fp} and w_{fn} are hyper parameters ($\in \mathbb{R}$), and the denominator is for normalization. Note that the A* search tends to increase false positives so that we set $w_{\text{fn}} > w_{\text{fp}}$ to reduce sample bias.

Algorithm 1 Interleaving planning and execution with constrained region relaxation. O_t is the sensor observation at time step t.

Inputs $\mathbf{x}_{\text{start}}$, $\mathbf{x}_{\text{goal}}$, G_0

$t \leftarrow 1, \mathcal{R}_{\text{free},0} \leftarrow \emptyset, \mathcal{R}_{\text{soft},0} \leftarrow \emptyset, \mathcal{R}_{\text{hard},0} \leftarrow \emptyset$
while true **do**

    $\mathcal{R}_{\text{free},t}, \mathcal{R}_{\text{soft},t}, \mathcal{R}_{\text{hard},t} \xleftarrow{\text{update}} (\mathcal{R}_{\text{free},t-1}, \mathcal{R}_{\text{soft},t-1}, \mathcal{R}_{\text{hard},t-1}, O_t)$

    $G_t \xleftarrow{\text{update}} (G_{t-1}, \mathcal{R}_{\text{free},t}, \mathcal{R}_{\text{soft},t})$ % $G_t = (\mathcal{V}_t, \mathcal{E}_t)$

    $\mathbf{g}^{\text{ext}} \leftarrow \mathbf{g} + \Psi_G(\mathcal{V}_t)$

    $\mathcal{X}_G \leftarrow \text{A*}(G_t, \mathbf{x}_{\text{start}}, \mathbf{x}_{\text{goal}}, \mathbf{g}^{\text{ext}})$ % Graph-based search

    $\mathcal{R}_{\text{rlx}} \leftarrow \mathcal{R}_{\text{soft},t} \cap \mathcal{X}_G$

    $\mathcal{X} \leftarrow \text{A*}(\mathcal{R}_{\text{free}} \cup \mathcal{R}_{\text{rlx}}, \mathbf{x}_{\text{start}}, \mathbf{x}_{\text{goal}})$ % Fine grid-based search

    $O_{t+1} \leftarrow \text{EXECUTE}(\mathcal{X})$

    if $\text{REACHED}(\mathbf{x}_{\text{start}}, \mathbf{x}_{\text{goal}})$ **or** $\text{TERMINATED}()$ **then break**

    $t \leftarrow t+1$

We finally formulate a mini-batch loss $\mathcal{L}_{\text{batch}}$ with per-sample weight $w^{(i)}$ as

$$\mathcal{L}_{\text{batch}} = \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} w^{(i)} \mathcal{L}_{\text{rlx}}(\mathbf{v}_{\text{rlx}}^{(i)}, {}^{\mathcal{B}}\mathbf{v}_{\text{rlx}}^{(i)}), \tag{13}$$

$$w^{(i)} = \min\left(1, \mathcal{L}_{\text{rlx}}(\mathbf{v}_{\text{rlx}}^{(i)}, {}^{\mathcal{B}}\mathbf{v}_{\text{rlx}}^{(i)})^\gamma\right) \tag{14}$$

where $\gamma > 0$. In this work, we use another set of hyper parameters, $w_{\text{fp}} = 0.5$ and $w_{\text{fn}} = 0.5$, for the per-sample weight $w^{(i)}$.

V. INTERLEAVING PLANNING AND EXECUTION

We introduce an interleaving planning and relaxation framework to achieve asymptotic completeness in semi-static environments. Algorithm 1 shows the pseudo code of the interleaving planning and execution framework while relaxing constrained region according to the relaxation cost-based graph-path search until it reaches a goal. Our algorithm updates the navigation space $\mathcal{R}_t$ and corresponding graph G_t upon new observation at each time step t. Leveraging an A* planner with the proposed relaxation cost estimator Ψ_G, our method finds the graph-based safe and efficient path from the start $\mathbf{x}_s$ to the goal $\mathbf{x}_g$. Note that we assume the continuous space start and goal to be projected to the region nodes on the graph internally.

In low-level planning, our framework provides multiple discretizations and coverages of path plans in the continuous navigation space $\mathcal{R}$.

- **Graph plan**: By default, we produce a best-effort graph plan $\mathcal{X}_G$ which is a sequence of regions. The robot tracks the centroid waypoints $\{c(\mathcal{X}_G[t])\}_{t=1}^{|\mathcal{X}_G|}$. In practice, the centroid-to-centroid motion between adjacent regions may not be locally feasible, we optionally extract a portal-sequence with shared boundary segments and plan local motions between consecutive portals.
- **Continuous plan within** $\mathcal{X}_G$: We refine the route by computing a continuous path $\mathcal{X}$ restricted to $\mathcal{X}_G$, using a finer-grid search (e.g., standard A*) or a local continuous planner. This enforces local feasibility and improves smoothness while respecting the region selection.

- **Continuous plan in** $\mathcal{X}_G \cup \mathcal{R}_{\text{free}}$: We further allow $\mathcal{X}$ to traverse both the planned regions and nearby free space, seeding the search with $\mathcal{X}_G \cup \mathcal{R}_{\text{free}}$. A finer-grid planner in this unified space yields shorter and safer trajectories while preserving the high-level guidance of the graph plan.

In this work, we use the continuous plan in $\mathcal{X}_G \cup \mathcal{R}_{\text{free}}$.

VI. EXPERIMENTAL SETUP

A. Quantitative Evaluation

We create a semi-static navigation benchmark to evaluate the human-likeness and the safety–efficiency trade-off. The benchmark consists of 34 $1\,\text{km} \times 1\,\text{km}$ OpenStreetMap [26] tiles sampled from global cities and aligned with satellite imagery. We split them into 26 for training, 5 for validation, and 3 for testing. We convert each tile into a 2D semantic bird-eye-view map using MIA [27], with up to 10 semantic labels: 1) *sidewalk* for $\mathcal{R}_{\text{free}}$, 2) *crosswalk*, *road*, *living street*, *parking lot*, *grass*, and *rough terrain* for $\mathcal{R}_{\text{soft}}$, and 3) *building* and *water* for $\mathcal{R}_{\text{hard}}$.

For training and validation, we construct $1,240$ navigation environments by randomly sampling 20 start-goal pairs per map and creating two variants for each: one with static conditions and one with semi-static conditions. We constrain start-goal pairs to *sidewalks* and space them $30\,\text{m}$-$200\,\text{m}$ apart. We collect one demonstration per environment, yielding $1,240$ demonstrations. For each static environment, a human expert draws a trajectory using a mouse. To synthesize the corresponding semi-static variant, we randomly sample a superpixel seed along the trajectory, alter neighboring superpixels to obstruct the pathway, and ask the expert to redraw the trajectory from that point, reusing the prefix up to the perturbed superpixel. This results in 620 static and 620 semi-static demonstrations.

To evaluate the safety–efficiency trade-off, we construct 300 environments from three diverse test maps—Milan, Baltimore, and Capitol Hill—by randomly sampling 100 start-goal pairs per map. We perturb them sampling up to three superpixel seeds along the Open-Source Routing Machine (OSRM) [28] -generated subgoal paths, with all changes aggregated per map to increase disruption likelihood. Each method plans on the original map but navigates in the modified one. For human-likeness evaluation, we generate 100 semi-static environments with demonstrations from five test maps, including two addional maps, sampling 20 start-goal pairs per map.

We evaluate SuReNav against all baselines on 2D maps, except ViPlanner [6], which requires visual input. For ViPlanner, we reconstruct the 2D maps in the Isaac Lab 3D simulator [29], manually refining OpenStreetMap nodes to preserve pedestrian topology and routing consistency. Below, we detail the baselines:

- **ViPlanner$^{\text{OSRM}}$** [6], [28]: A vision-based local navigation approach extended from ViPlanner [6]. Due to the local nature, ViPlanner requires subgoals to reach distant targets. To address this, we integrate ViPlanner with OSRM to provide global pedestrian routes as subgoal sequences. If
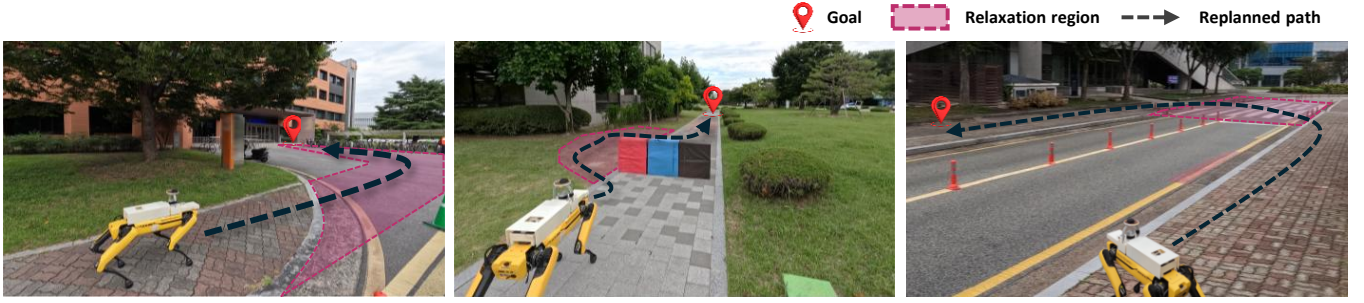
Fig. 3: Real-world evaluation in semi-static settings. The quadruped robot begins by following a path planned using an outdated prior map. Upon encountering newly constrained regions, SuReNav dynamically relaxes a red-colored soft-constraint region and adapts its behavior to perform human-like navigation that balances safety and efficiency. Video demonstrations are available in the supplementary material.

consecutive subgoals exceed $15\,\mathrm{m}$, we insert intermediate subgoals to satisfy ViPlanner's goal specification. We also align ViPlanner's semantic label table with the region priorities and costs used in other baselines.

- **D* Lite** [16]: D* Lite is an incremental heuristic search algorithm that replans by repairing the shortest path as edge costs change. We derive traversal costs from the reciprocal proportion of demonstration-traversed superpixels with 0-1 scaling, enabling pixel-level trajectory optimization under the evolving cost field.
- **COA*** [30]: Class-Ordered A* (COA*) is a generalized extension of A* that incorporates hierarchical class ordering into its search process. We determine the class priority using the traversal frequency observed in the expert demonstration data, allowing COA* to relax constraints that respect a fixed, data-driven priority order.
- **Rule-based Constraint Relaxation (RCR)** [5]: We denote this unnamed method as RCR, a reactive framework that relaxes constraints by manually scoring regions based on predefined label-to-risk mappings and distances to the goal. We also replace our learned cost estimator with RCR's scoring function to compare our learning-based approach and predefined heuristic-based strategy.

We assess human-likeness by comparing planned trajectories with expert demonstrations focusing on geometric similarity and constraint violations using the following metrics:

- **Fréchet distance**: Distance between two continuous curves [31], with each sample's Fréchet distance normalized by the Euclidean distance between start and goal.
- **Relaxation IoU**: Intersection-over-Union (IoU) between predicted and ground-truth relaxed region sets.

We then benchmark using the following metrics:

- **Success - (SR)**: Proportion of episodes in which the robot reaches its goal without entering any hard-constrained regions, $\mathcal{R}_{\mathrm{hard}}$.
- **Success weighted by path length (SPL)**: Efficiency metric comparing the executed path length to the shortest feasible path length, weighting zero to failed episodes. When robot reaches its goal without entering any hard-constrained regions $\mathcal{R}_{\mathrm{hard}}$, the episode is noted as success.
- **Total Risk**: Cumulative per pixel risk along a trajectory, where each label's unit risk is the mean 0-1 scaled score

TABLE I: Comparison of human-likeness to expert demonstrations. We compute Fréchet distance and relaxation IoU metrics against 100 expert demonstration trajectories from five unseen global city maps.

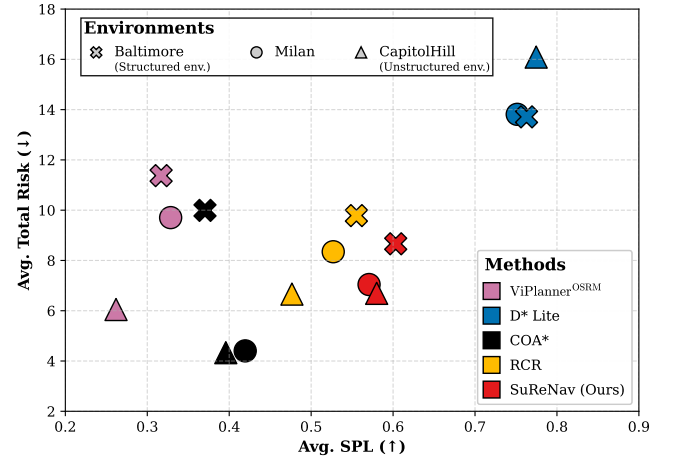| Method	Fréchet distance ($\downarrow$)	Relaxation IoU ($\uparrow$)
D* Lite	0.429	0.223
COA*	0.701	0.271
RCR	0.626	0.259
SuReNav (Ours)	**0.334**	**0.416**

Fig. 4: Comparison of the proposed SuReNav and four baseline method in 300 simulated navigation environments from three urban areas—Milan, Baltimore, and Capitol Hill. We evaluate average SPL and Total Risk metrics where performance improves toward the lower-right corner that is higher SPL and lower Total Risk.

from three commercial large language models—ChatGPT, Gemini, and Claude. This metric captures both risk magnitude and exposure length for assessing safety.

B. Qualitative Evaluation

We demonstrate our method in campus environments on a quadruped robot, Boston Dynamics Spot, equipped with an Intel Realsense D435i camera and an 128-channel Ouster LiDAR. The robot constructs a superpixel region-adjacency graph by performing PIDNet-based semantic segmentation [32] and aggregating pixel labels into regions. To
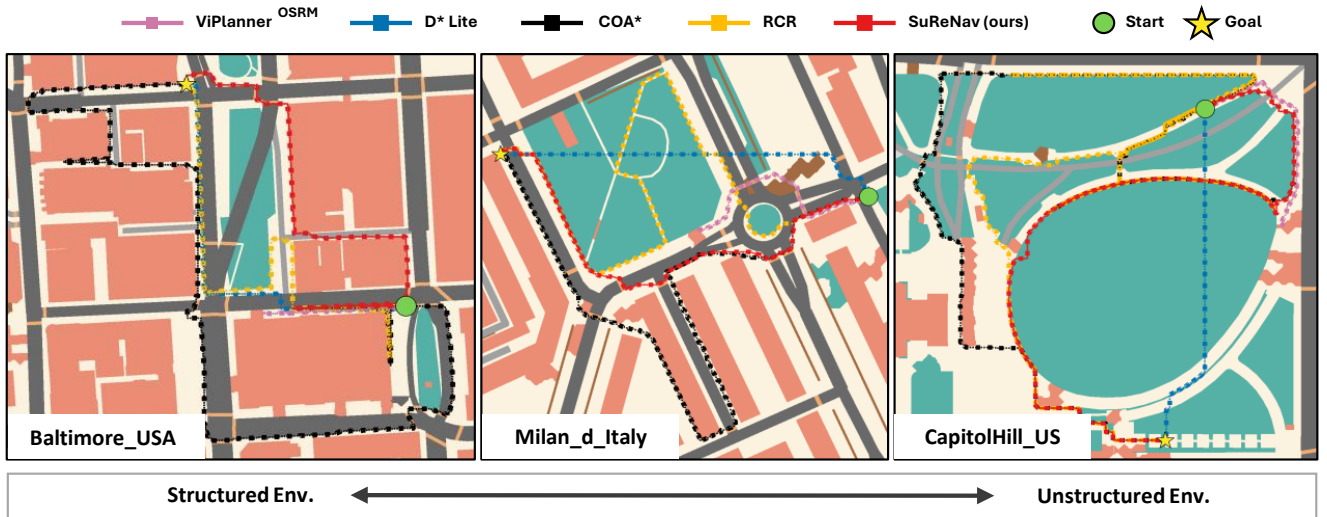
Fig. 5: Examples of navigation behaviors across three urban scenarios. Green circles and yellow stars denote start and goal locations, respectively. The ivory, green, light orange, gray, orange, and blue regions correspond to *sidewalks*, *lawns*, *crosswalks*, *roads*, *buildings*, and *water* areas, respectively. Colored line segments indicate the trajectories of individual navigation examples.
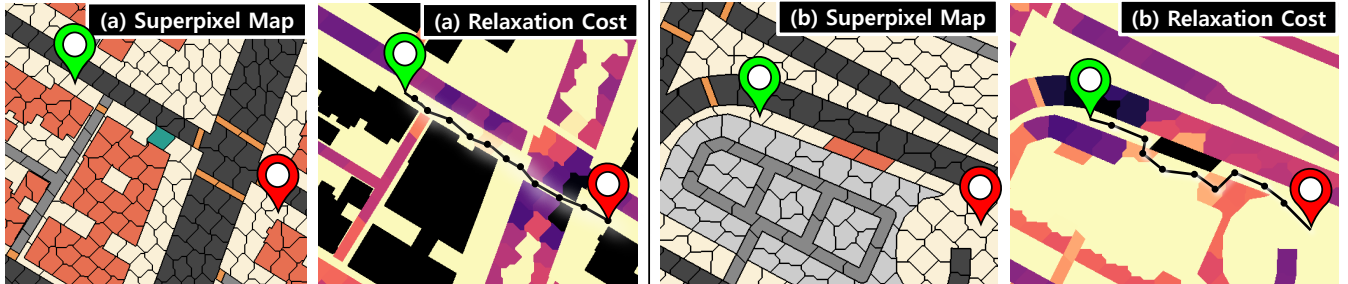

Fig. 6: Visualization of learned relaxation cost distributions in two distinct navigation scenarios. The ivory, green, light orange, light gray, gray, dark gray, and orange regions in the superpixel maps correspond to *sidewalks*, *lawns*, *crosswalks*, *parking lots*, *living streets*, *roads* and *buildings*, respectively. Brighter colors indicate lower relaxation costs, marking regions as suitable for traversal during planning. In (a), SuReNav relaxes a grass region to create an efficient shortcut, while in (b), it navigates around a newly introduced obstacle by identifying the adjacent parking lot as the optimal path for relaxation.

emulate semi-static conditions, we inject regional constraints such as adding obstacles during navigation or obfuscating portions of the prior map before execution as shown in Fig. 3.

VII. EVALUATION

We evaluate the human-likeness of relaxation-based navigation against 100 expert demonstrations in semi-static environments. As shown in Table I, SuReNav achieves the smallest Fréchet distance to the demonstrations while relaxing a set of regions that closely matches human selections. Notably, its relaxation IoU is 44% higher than the next-best approach COA* since all baselines rely on pre-defined, fixed regional costs. In contrast, SuReNav predicts context-conditioned regional costs from demonstrations, enabled by the graph-based relaxation cost estimator that jointly accounts for nearby risky regions and navigation context. This enables SuReNav to effectively learn human navigation behaviors in semi-static environments. Note that we exclude ViPlanner[OSRM], as it fails to generate complete subgoals

connecting the start or goal of human demonstrations due to the limited distribution of OSRM nodes.

Fig. 4 compares SuReNav with four baselines in simulated semi-static environments; higher SPL and lower RPM indicate better performance. SuReNav achieves comparable success and efficiency while producing lower-risk trajectories across all structure levels of environments. COA* occasionally yields lower risk in unstructured cities—Milan and Capitol Hill—by minimizing the number of constraint violations, but its trajectories are about 50% longer than SuReNav's. D* Lite attains the highest SPLs, yet its reliance on fixed regional costs leads to roughly twice the Total Risk in unseen settings. In contrast, SuReNav does not require manual cost specification and generalizes through learning from demonstrations. Note that all methods achieve 100% SR except ViPlanner, which records 51%.

Fig. 5 illustrates examplar navigation behaviors in three urban environments. In the Baltimore and Milan maps, SuReNav efficiently reaches the goals using only *sidewalks*

and *crosswalks*, avoiding jaywalking or traversing *lawns*. In detail, as shown in Fig. 6, SuReNav creates safe and efficient paths by assigning low costs only to specific superpixels on $\mathcal{R}_{\text{soft}}$, enabling the formation of complete paths, while assigning high costs to the surrounding areas to guide against unexpected violations. In contrast, D* Lite often traverses *roads* or *lawns* to produce shorter paths in unstructured city setups, as its incremental local search fails to account for accumulated risks along the route. On the other hand, COA* exhibits the longest trajectories in both environments since COA* prioritizes minimizing the number of constraint relaxations, thus producing the most conservative routes.

We finally demonstrated SuReNav in real-world campus navigation scenarios using the quadruped robot Spot. Fig. 3 presents three examples of human-like constraint-relaxation behaviors. In one case, the robot crosses a *road* at the end of a *sidewalk* by relaxing high-risk *road* regions (red), mirroring human navigation strategies. In another example (middle), SuReNav relaxes only a small portion of a constrained region instead of traversing a large area, demonstrating precise and selective relaxation. Finally, as shown on the right, SuReNav directs Spot to take a distant crosswalk, sacrificing efficiency to ensure safety. This emergent behavior, learned from demonstrations rather than manual programming or cost defining, highlights the framework's ability to balance efficiency and adherence to traffic regulations in novel environments. Experiment videos are available in the supplementary.

VIII. CONCLUSION

This paper introduced SuReNav, a constraint-aware framework that interleaves relaxation, planning, and execution to balance safety and efficiency in semi-static environments. SuReNav learns human-like relaxation and planning behaviors from expert demonstrations, formalizing the problem with superpixel-graph map construction, regional relaxation-cost estimation, and relaxed path search by jointly training a relaxation cost estimator and a differentiable graph-based A* planner. The estimator treats trivially safe areas (e.g., *sidewalks*) as unconstrained, while handling other traversable regions as soft constraints—initially forbidden but admissible upon explicit relaxation. Through extensive evaluations in simulated environments, we show that SuReNav produces human-like, safe, and efficient navigation in novel settings. Finally, its real-world deployment on a quadruped robot validates its generalization beyond simulation.

REFERENCES

[1] L. I. Reyes Castro, P. Chaudhari, J. Tůmová, S. Karaman, E. Frazzoli, and D. Rus, "Incremental sampling-based algorithm for minimum-violation motion planning," in *Conf. on Decision and Control*, pp. 3217–3224, IEEE, 2013.

[2] K. Hauser, "The minimum constraint removal problem with three robotics applications," *Int'l J. of Robotics Research*, 2014.

[3] N. Ganganath *et al.*, "Shortest path planning for energy-constrained mobile platforms navigating on uneven terrains," *Trans. on Industrial Informatics*, 2018.

[4] K. K. Hauser, "Minimum constraint displacement motion planning.," in *Proc. Robotics: Science and Systems*, vol. 6, p. 2, 2013.

[5] J. Kim, K. Koh, S. S. Lee, Y. Park, and D. Park, "Reactive constraint relaxation for urban environment navigation," in *Proc. Int'l Conf. on Robot Intelligence Technology and Applications*, Springer, 2024.

[6] P. Roth, J. Nubert, F. Yang, M. Mittal, and M. Hutter, "Viplanner: Visual semantic imperative learning for local navigation," in *Proc. Int'l Conf. on Robotics and Automation*, pp. 5243–5249, IEEE, 2024.

[7] S. Gupta, V. Tolani, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," *Int'l J. on Computer Vision*, vol. 128, p. 1311–1330, 2020.

[8] R. Drouilly, P. Rives, and B. Morisset, "Semantic representation for navigation in large-scale environments," in *Proc. Int'l Conf. on Robotics and Automation*, pp. 1106–1111, IEEE, 2015.

[9] Y. Zhao, P. Liu, W. Xue, R. Miao, Z. Gong, and R. Ying, "Semantic probabilistic traversable map generation for robot path planning," in *Int'l Conf. on Robotics and Biomimetics*, pp. 2576–2582, IEEE, 2019.

[10] A. Mousavian, A. Toshev, M. Fišer, J. Košecká, A. Wahid, and J. Davidson, "Visual representations for semantic target driven navigation," in *Proc. Int'l Conf. on Robotics and Automation*, IEEE, 2019.

[11] A. Sridhar, D. Shah, C. Glossop, and S. Levine, "NoMaD: Goal Masked Diffusion Policies for Navigation and Exploration," *Proc. Int'l Conf. on Robotics and Automation*, pp. 63–70, 2024.

[12] D. S. Chaplot, D. Gandhi, A. Gupta, and R. Salakhutdinov, "Object goal navigation using goal-oriented semantic exploration," in *Conf. on Neural Information Processing Systems*, pp. 4247–4258, 2020.

[13] J. Chen, G. Li, S. Kumar, B. Ghanem, and F. Yu, "How to not train your dragon: Training-free embodied object navigation with semantic frontiers," in *Proc. Robotics: Science and Systems*, 2023.

[14] A. Kreutzmann, D. Wolter, F. Dylla, and J. H. Lee, "Towards safe navigation by formalizing navigation rules," *Int'l J. on Marine Navigation and Safety of Sea Transportation*, vol. 7, no. 2, 2013.

[15] K. Zhou, K. Zheng, *et al.*, "ESC: Exploration with soft commonsense constraints for zero-shot object navigation," in *Proc. Int'l Conf. on Machine Learning*, PMLR, 2023.

[16] S. Koenig and M. Likhachev, "D* lite," in *Proc. Nat'l Conf. on Artificial Intelligence*, pp. 476–483, 2002.

[17] R. Yonetani, T. Taniai, M. Barekatain, M. Nishimura, and A. Kanezaki, "Path planning using neural a* search," in *Proc. Int'l Conf. on Machine Learning*, pp. 12029–12039, PMLR, 2021.

[18] G. Kahn, P. Abbeel, and S. Levine, "Badgr: An autonomous self-supervised learning-based navigation system," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1312–1319, 2021.

[19] X. Zang, M. Yin, J. Xiao, S. Zonouz, and B. Yuan, "Graphmp: Graph neural network-based motion planning with efficient graph search," *Conf. on Neural Information Processing Systems*, 2023.

[20] D. Stutz, A. Hermans, and B. Leibe, "Superpixels: An evaluation of the state-of-the-art," *Computer Vision and Image Understanding*, 2018.

[21] K. Chen *et al.*, "Constraint-aware zero-shot vision-language navigation in continuous environments," *Trans. on Pattern Analysis and Machine Intelligence*, 2025.

[22] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, S. Süsstrunk, *et al.*, "Slic superpixels," tech. rep., Technical report EPFL, 2010.

[23] L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini, "Recipe for a general, powerful, scalable graph transformer," *Conf. on Neural Information Processing Systems*, 2022.

[24] X. Bresson and T. Laurent, "Residual gated graph convnets," *arXiv preprint*, 2017.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Conf. on Neural Information Processing Systems*, vol. 30, 2017.

[26] OpenStreetMap contributors. `https://www.openstreetmap.org`, 2017.

[27] C. Ho, J. Zou, O. Alama, *et al.*, "Map it anywhere: Empowering bev map prediction using large-scale public datasets," *Conf. on Neural Information Processing Systems*, 2024.

[28] D. Luxen and C. Vetter, "Real-time routing with openstreetmap data," in *Proc. Int'l Conf. on Advances in Geographic Information Systems*, GIS '11, pp. 513–516, ACM, 2011.

[29] M. Mittal *et al.*, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, 2023.

[30] J. Lim and P. Tsiotras, "A generalized a* algorithm for finding globally optimal paths in weighted colored graphs," in *Proc. Int'l Conf. on Robotics and Automation*, pp. 7503–7509, IEEE, 2021.

[31] T. Eiter, H. Mannila, *et al.*, "Computing discrete fréchet distance," *Technical Report*, 1994.

[32] J. Xu, Z. Xiong, and S. P. Bhattacharyya, "Pidnet: A real-time semantic segmentation network inspired by pid controllers," in *Proc. Conf. on Computer Vision and Pattern Recognition*, pp. 19529–19539, 2023.