CRESTE: Scalable Mapless Navigation with Internet Scale Priors and Counterfactual Guidance

Arthur Zhang, Harshit Sikchi, Amy Zhang, Joydeep Biswas
The University of Texas at Austin

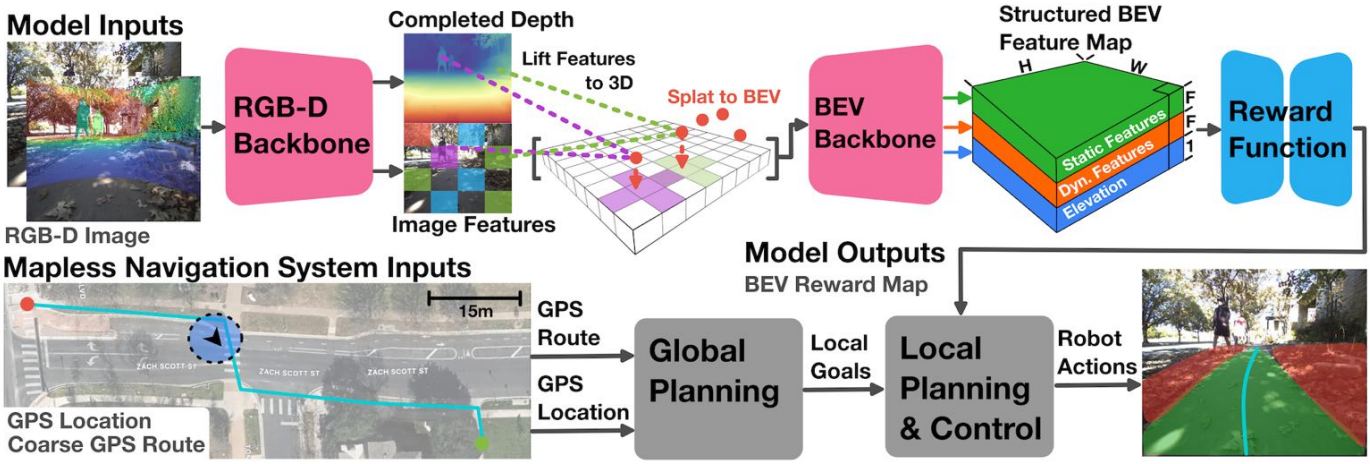

Fig. 1: CRESTE learns to predict bird's eye view (BEV) feature and reward maps for mapless navigation in urban environments. Our RGB-D and BEV backbone work together to predict a structured BEV feature map that consists of separate feature maps for static/dynamic entities and elevation, which our reward function uses to predict expert-aligned rewards. To learn this feature map, CRESTE proposes a novel method of distilling internet scale priors from visual foundation models. We also introduce an active learning framework and training objective that improves reward alignment using counterfactual demonstrations. We integrate CRESTE in a modular navigation system that uses coarse GPS guidance and predicted reward maps to produce robot actions that reach navigation goals safely.

Abstract*—We address the long-horizon mapless navigation problem: enabling robots to traverse novel environments without relying on high-definition maps or precise waypoints that specify exactly where to navigate. There are two major challenges arise: (1) learning robust, generalizable perceptual representations of the environment—where it is impossible to pre-enumerate all relevant factors and ensure robustness to perceptual aliasing—and (2) planning human-aligned navigation paths using these learned features. Existing solutions often struggle to generalize due to their reliance on: (a) hand-curated object lists, which overlook new, unforeseen factors, (b) end-to-end learning of navigation-relevant features, which is constrained by the limited availability of real robot data, (c) large sets of expert demonstrations, which provide insufficient guidance on the most critical perceptual cues, or (d) handcrafted reward functions for learning, which are difficult to design and adapt for new scenarios. To overcome these limitations, we propose CRESTE, the first approach for learning representations that address the full mapless navigation problem. We introduce a framework for representation and policy learning that does not require large-scale robot datasets or manually specified feature sets. First, CRESTE leverages visual foundation models trained on internet-scale data to learn continuous bird's-eye-view representations capturing elevation, semantics, and instance-level features. Second, it incorporates a counterfactual-based loss and active learning procedure to focus on the perceptual cues that matter most by querying humans for counterfactual trajectory annotations in challenging scenes. We evaluate CRESTE in kilometer-scale navigation tasks across six distinct urban environments. CRESTE significantly outperforms all existing state-of-the-art approaches with 70% fewer human interventions per mission, including a 2-kilometer mission in an unseen environment with just 1 intervention; showcasing its robustness and effectiveness for long-horizon mapless navigation.

I. Introduction

Mapless navigation is the task of reaching user-specified goals without high-definition (HD) maps and precise navigation waypoints. Instead, mapless navigation approaches rely on egocentric sensor observations (e.g. RGB images, point clouds, GPS), coarse waypoints from public routing services, and satellite imagery to plan safe paths aligned with human preferences. While this task is relevant to many settings, our work focuses on mapless navigation for challenging urban scenes like downtowns, parks, and residential neighborhoods.

Mapless navigation is desirable as it enables rapid deployment of robots to novel environments without laborious map construction and maintenance. Furthermore, solutions to mapless navigation broadly improve robustness to real-world factors that affect both map-centric and mapless methods, such as unexpected environment changes and traffic patterns for dynamic entities.

Geometric-only methods [3, 45] are one approach to mapless navigation that considers geometric factors like static

obstacles. While understanding geometric factors is important, they are insufficient alone for explaining navigation preferences in diverse urban scenes that require jointly reasoning about terrain preferences (grass vs. concrete), semantic cues (crosswalks and crossing signs), and uneven terrain (sharp curb dropoffs). Furthermore, methods need to reason about the factors above without pre-enumerating the list of semantic classes and entities a priori to scale gracefully in diverse environments. Lastly, planning modules that reason about these factors must identify the most salient features and understand how they influence which paths are most likely to be aligned with human preferences.

Mapless navigation approaches that consider factors beyond geometry broadly consist of single-factor perception, hand-curated multi-factor perception, end-to-end learning, and zero-shot pre-trained large language model (LLM)/visual language model (VLM) transfer. Single-factor [23, 46, 13] and multi-factor perception [24, 37] approaches encode either single or multiple factors (e.g. terrain, elevation, semantics) to a learned representation and use analytical methods like model predictive control [35] for path planning. Single-factor methods generalize poorly to complex settings, while multi-factor methods, though more flexible, rely on a hand-curated list of semantic classes and terrains, limiting generalization to unseen classes. End-to-end learning methods [38, 40, 39] jointly learn the representation and policy from expert demonstrations, but are prone to overfitting without access to large-scale robot datasets. While recent works [47, 25] show that pre-trained factors in LLMs and VLMs can be leveraged for navigation, we empirically demonstrate that they are not well-attuned to urban navigation, leading to poor zero-shot transfer performance in complex scenes.

We address the aforementioned limitations with CRESTE, **Counterfactuals for Reward Enhancement with Structured Embeddings**, the first method that learns generalizable perceptual representations for the full mapless navigation problem in urban environments. Our key insights lie in how we leverage visual foundation models and counterfactual demonstrations to learn structured representations and policies for mapless navigation. First, CRESTE learns a perceptual encoder by synergizing navigation priors from multiple visual foundation models (VFMs) trained on internet-scale data, yielding semantic, geometric, and entity aware representations that are robust to perceptual aliasing and geometrically grounded in the robot's navigation horizon. We find it is essential to unify priors from multiple VFMs as each model is best suited for different factors: SegmentAnything [33, 20] for distinguishing between entities and CLIP [32]/Dino [5, 28] for inferring visual-language and semantic features. Second, CRESTE learns a reward function for navigation using our principled active learning framework and IRL objective that queries humans for counterfactuals to align learned policies with human preferences. Counterfactuals (i.e. alternate trajectories that do not align with preferences) enable reward functions to reason about the most salient perceptual features and learn complex navigation behaviors under a unified objective, removing the need for large-scale robot datasets or careful reward design.

We summarize these contributions below:

- **Representation Learning Through Model Distillation.** A new model architecture and distillation objective for distilling navigation priors from visual foundation models to a lightweight image to BEV map backbone.
- **Counterfactually Aligned Rewards.** An active learning framework and principled counterfactual IRL formulation for reward alignment using counterfactual and expert demonstrations.

We demonstrate our approach's effectiveness through real-world kilometer-scale navigation experiments, such as the one shown in Fig. 5. In total, we evaluate our approach in **6** distinct seen and unseen urban environments, outperforming existing state-of-the-art imitation learning, inverse reinforcement learning, and heuristic-based methods on the task of mapless navigation.

II. Related Work

In this section, we position our work within the broader context of methods that perform mapless navigation by learning representations and policies for safe local path planning. Existing solutions broadly consist of hybrid approaches that combine single/multi-factor learned representations with analytical path planning, end-to-end methods that learn representations and policies jointly from data, and VLM/LLM-based methods that leverage large foundation models zero-shot for navigation. Underlying these approaches are two key challenges: learning robust, generalizable perceptual representations that encode a sufficient set of navigation factors, and identifying and reasoning about the important factors to plan safe paths.

Single [16, 23] or multi-factor [10, 24, 37] approaches explicitly learn high-level representations that consider factors such as terrain, semantic, or geometric understanding. Using manually curated cost functions to balance these factors, these methods map representations to scalar costmaps for analytical planning and control methods like model predictive control (MPC) [35]. While these methods can jointly reason about many navigation factors, they assume the full set of semantic classes and terrains are static and known apriori, and consequently generalize poorly to unexpected semantic classes, terrains, or other factors not seen during training. Furthermore, they require expert tuning of complex multi-objective cost functions to jointly consider multiple factors.

End-to-end methods learn a representation and policy together using either behavior cloning, RL, or IRL. Behavior cloning methods [17, 40] implicitly learn representations with navigation cues and how to reason about them by mimicking the expert policy. These approaches scale effectively given large-scale datasets with expert demonstrations but are prone to overfitting otherwise. RL [12] and IRL [49] methods also mimic the expert policy, but additionally leverage handcrafted reward functions (e.g. minimize vibrations [14] and likelihood of interventions [15]) to guide representation and policy learning. This improves learning efficiency at the cost of careful reward tuning to learn fine-grained behaviors that are difficult

to infer from expert demonstrations alone (e.g. avoid collisions with walls but not tall grass). Reward functions also offer more interpretability, particularly during failures where reward functions make it possible to identify the root cause.

An emerging class of methods [25, 47] leverage pre-trained factors present in VLMs and LLMs, large foundation models pre-trained on internet-scale data, zero-shot for navigation. These foundation models leverage geographic hints in the form of satellite imagery and topological maps, along with image observations and text instructions to predict safe local waypoints for analytical planning and control methods. While promising, it is not well understood how to best guide VLMs/LLMs to reason about the most important navigation factors for long-horizon navigation tasks.

CRESTE is a hybrid approach that learns perceptual representations by distilling navigation factors from VFMs pre-trained on internet-scale data and leverages IRL to efficiently learn expert-aligned behavior. This stands in contrast to prior work that either learn the representation from expert demonstrations or bespoke datasets, and approaches that extract navigation factors from VLMs/LLMs in a zero-shot manner. This provides a number of benefits when scaling to diverse urban settings: 1) Improving robustness to perceptual aliasing due to lighting, weather, and viewpoint variations; 2) Promoting generalizability to unseen semantic entities; 3) Encoding semantic and entity aware priors without dense human labels. Additionally, CRESTE replaces complex multi-objective reward functions common for IRL and RL methods with a unified counterfactual-based learning objective. Counterfactual demonstrations provide a theoretically principled way to communicate complex navigation operator preferences that would be otherwise difficult to specify from expert demonstrations alone.

III. The Mapless Urban Navigation Problem

We now develop the mapless navigation problem for urban environments. We first formulate the path planning problem in this context in Sec. III-A and then discuss the problem of learning general expert-aligned costs in Sec. III-B. Finally, Sec. III-C highlights key challenges in our problem formulation that this work addresses. In this work, we refer to costs as negated rewards and use them interchangeably.

A. Path Planning for Mapless Navigation

For each timestep, we assume the robot has access to the current observation o_t and pose $x_t \in \mathcal{X}$ in the global frame, where the robot state space $\mathcal{X}$ lies in $SE(2)$ for ground vehicles. The robot must plan a finite trajectory $\Gamma_S = [x_t, ..., x_{t:t+S}]$ from x_t to goal G, either given by the user or obtained from public routing services. Γ_S consists of S current and future states $x \in \mathcal{X}$ which minimize the following objective function:

$$\Gamma_S = \arg_{\Gamma}\min ||x_{t+S} - G|| + \lambda_t \mathcal{J}(\Gamma), \tag{1}$$

where $||x_{t+S} - G||$ is the distance between the final state x_{t+S} and G, and $\mathcal{J}(\Gamma)$ is a cost function for path planning scaled by a relative weight λ_t. In mapless urban environments, the robot pose x_t and goal G may be highly noisy, causing $\mathcal{J}(\Gamma)$'s importance to vary dynamically across time. In this work, we do not address this problem and assume λ_t to be static.

B. General Preference-Aligned Cost Functions

To properly define our cost function $\mathcal{J}(\Gamma)$, we first need to introduce the notion of an observation function $\Theta : \mathcal{O} \to \mathcal{T}$ that maps observations o_t to a joint distribution $\mathcal{T}$ that encapsulates a set of relevant factors for navigation in the world. In general, the sufficient set of factors is unknown but can be approximated for each environment. Let $T : \mathcal{X} \to \mathcal{T}$ be a function that maps a robot pose $x \in \mathcal{X}$ to a feature $\tau \in \mathcal{T}$, where τ captures the relevant set of factors necessary to reason about $\mathcal{J}(\Gamma)$. The relationship and set of relevant factors may consist of geometric, semantic, and social costs as follows:

$$\mathcal{J}(\Gamma) = \sigma(\mathcal{J}_{\text{geometric}}(\Gamma), \mathcal{J}_{\text{semantic}}(\Gamma), \mathcal{J}_{\text{social}}(\Gamma)) \tag{2}$$

where σ is a nonlinear function that combines individual cost terms.

We assume the operator has an underlying true cost function $H : \mathcal{T} \to \mathbb{R}^{0+}$ mapping the sufficient set of factors to scalar real-valued costs based on their preferences. Let $H \in \mathcal{H}$, where $\mathcal{H}$ is the continuous space of underlying cost functions. In general, H is unknown and often depends on the robot embodiment, environment, and task. For mapless navigation, we define this task to be goal-reaching.

C. Open Challenges

In this work, we are concerned with learning both $\Theta(o_t)$ and $\mathcal{J}(\Gamma)$. This is difficult as the sufficient set of factors for navigating diverse, urban environments is unknown and the relationship between factors may be highly nonlinear. Our approach to learning $\Theta(o_t)$ distills features from VFMs, which provide a breadth of factors including but not limited to: geometry, semantics, and entities. This promotes learning a joint distribution $\mathcal{T}$ sufficient for mapless urban navigation. Furthermore, we propose a counterfactual-based framework for learning $\mathcal{J}(\Gamma)$, which becomes important when dealing with complex feature distributions $\mathcal{T}$ and nonlinear relationships between factors σ.

IV. Approach

We now present CRESTE, an end-to-end learning-based system that predicts general-purpose, egocentric reward maps for local path planning in complex, urban environments. Unlike prior works [40, 48] that learn solely from expert demonstrations, our method leverages large visual foundation models (VFMs) to learn robust perceptual representations rich with navigation priors and counterfactual demonstrations to align reward maps more closely with operator preferences.

CRESTE is a modular approach with two key components: 1) A perceptual encoder $\Theta(o_{\text{rgb, t}}, o_{\text{depth, t}})$ that takes the robot's current RGB and sparse depth observation and predicts a completed depth image $y_{\text{depth, t}}$ and structured BEV feature map $y_{\text{bev, t}}$; 2) A reward function $r_\phi(y_{\text{bev, t}})$ that takes $y_{\text{bev, t}}$

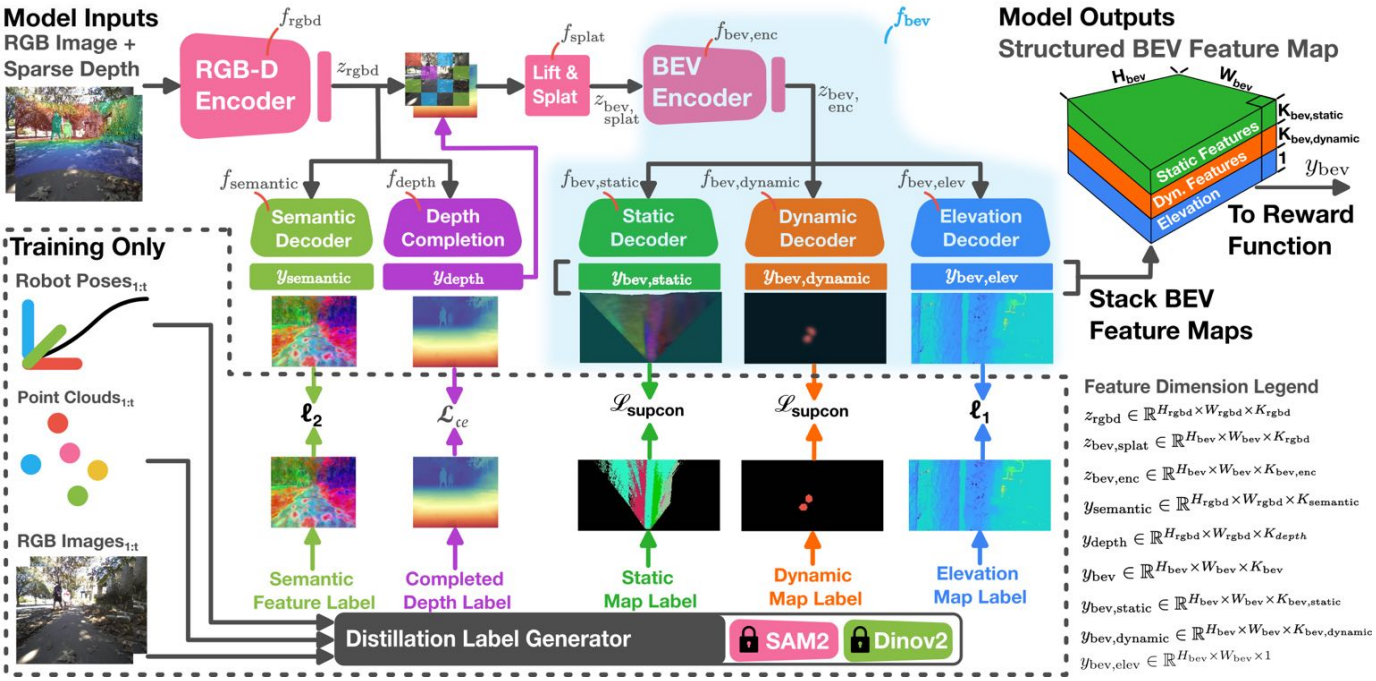

Fig. 2: Training procedure and model architecture for the CRESTE perceptual encoder Θ. Our RGB-D backbone extracts semantic features and performs depth completion from a single RGB and sparse depth image pair. Next, we lift and splat the image features z_{rgbd} to an unstructured BEV feature map before predicting continuous static panoptic, dynamic panoptic, and elevation feature maps. Finally, we stack the predicted BEV features to construct a structured BEV representation for our learned reward function. We supervise Θ using semantic feature maps, completed depth, and BEV map labels generated by our SegmentAnythingv2 [20] and Dinov2 [28]-powered distillation label generator. We define the dimensions for important feature maps in the Feature Dimension Legend on the bottom right.

and outputs a BEV scalar reward map $y_{\text{reward, t}}$. Θ captures key semantic, geometric, and entity factors crucial for urban navigation and r_ϕ jointly reasons about these factors to produce local BEV reward maps for local path planning. Both components are learned from VFMs, expert demonstrations, and offline counterfactual annotations, without relying on dense human labels, exhaustive semantic categories, or large-scale expert datasets.

In the remainder of this section, we describe the following: 1) Sec. IV-A - The CRESTE model architecture, highlighting key innovations for learning the perceptual encoder Θ in Sec. IV-A1 and reward function r_ϕ in Sec. IV-A2, 2) Sec. IV-B - The CRESTE training procedure, where Sec. IV-B1 presents our learning and VFM distillation procedure for Θ and Sec. IV-B2 presents our active reward learning framework for learning r_ϕ from expert and counterfactual demonstrations.

A. CRESTE *Model Architecture*

CRESTE implements Θ and r_ϕ using four major components. Fig. 2 illustrates the first three components that constitute Θ: 1) **RGB-D encoder** (f_{rgbd}) encodes important semantic cues for navigation (e.g. crosswalk markings, objects, terrains) and geometric information (dense depth), 2) **Lift-splat module** (f_{splat}) lifts and grounds image features into a local BEV map, and 3) **BEV Inpainting Backbone** (f_{bev}) completes the scene, predicting a structured BEV feature map y_{bev} with continuous entity-aware semantic and elevation layers. The fourth component is our reward function $r_\phi(y_{\text{bev}})$, which consumes the BEV representation y_{bev} and predicts a scalar reward map y_{reward} for navigation. We define specific details regarding the dimensions of each feature in Fig. 2 and present our architecture and innovations for Θ in Sec. IV-A1 and r_ϕ in Sec. IV-A2.

1) Perceptual Encoder Model Architecture: Our 25.5M-parameter perceptual encoder draws inspiration from the TerrainNet [24] architecture while introducing two key model innovations that improve generalizability to diverse urban scenes. These innovations are our semantic decoder head f_{semantic} and BEV map decoders ($f_{\text{bev,static}}$, $f_{\text{bev,dynamic}}$) that predict continuous panoptic features for static and dynamic entities. We borrow the term "panoptic" [19] from the computer vision community, where it refers to the task of understanding both the semantics and instances in the scene. We present an overview of these components next.

RGB-D Encoder. $f_{\text{rgbd}}(o_{\text{rgb}}, o_{\text{depth}})$ processes the RGB and sparse depth image using an EfficientNet-B0 [44] encoder and outputs a latent feature map z_{rgbd}. We pass z_{rgbd} to two decoder heads: 1) a depth completion head $f_{\text{depth}}(z_{\text{rgbd}})$ that outputs a completed depth map y_{depth} and 2) a semantic decoder head $f_{\text{semantic}}(z_{\text{rgbd}})$ that regresses a semantic feature map y_{semantic}.

Lift-Splat Module. Adopting TerrainNet's soft-quantization approach, $f_{\text{splat}}(z_{\text{rgbd}}, y_{\text{depth}})$ lifts the output of f_{rgbd} to 3D space using y_{depth} and "splats" them onto a BEV grid to produce an unstructured feature map $z_{\text{bev, splat}}$. We learn weights to bilinearly interpolate each image feature to its corresponding cell and 4 neighboring map cells, which simultaneously grounds features to the robot's local planning horizon and corrects depth prediction inconsistencies.

BEV Inpainting Backbone. We implement the final component $f_{\text{bev}}(z_{\text{bev, splat}})$ using a U-Net [36] architecture with a shared BEV map encoder $f_{\text{bev, enc}}$ and three decoder heads, $f_{\text{bev, static}}(z_{\text{bev, enc}})$, $f_{\text{bev, dynamic}}(z_{\text{bev, enc}})$, and $f_{\text{bev, elev}}(z_{\text{bev, enc}})$, where $z_{\text{bev, enc}}$ is the latent feature map produced by $f_{\text{bev, enc}}$. Our three decoder heads predict static panoptic feature maps $y_{\text{bev, static}}$, dynamic panoptic feature maps $y_{\text{bev, dynamic}}$, and elevation maps $y_{\text{bev, elev}}$ respectively. We stack these BEV feature maps along the channel dimension to construct a structured BEV feature map y_{bev} for downstream reward learning.

Design Rationale and Key Innovations. Our architectural innovations aim to address three challenges to perceptual representation learning for mapless urban navigation: 1) Robustness to perceptual aliasing, where representations are lighting, viewpoint, and weather invariant; 2) Generalization beyond the training data to unseen semantic classes and terrains; 3) Panoptic understanding without exhaustively enumerating all possible semantic classes and terrains a priori.

We address challenges 1 and 2 by using f_{semantic} to learn representations that regress to the features produced by large-scale Visual Foundation Models (VFMs) such as Dinov2 [28]. By matching Dinov2's learned representations given the same RGB observations, our model inherits robust semantic and geometric priors. Furthermore, these priors capture high-level semantic cues without restricting the model to a fixed set of semantic classes. As a result, Θ is expected to generalize beyond the classes and terrain types present in the robot's training data.

However, Dinov2 alone lacks the entity-awareness needed to accurately project and inpaint features in heavily occluded urban scenes, particularly when working with noisy and sparse depth inputs. $f_{\text{bev, static}}$ and $f_{\text{bev, dynamic}}$ remedy this by learning a map representation aligned with BEV instance labels derived from SegmentAnythingv2 (SAM2) [20]. This ensures our model can identify and localize individual entities—both static and dynamic—within a local BEV horizon, thus bolstering its capacity to handle complex occlusions and ambiguous depth cues. Working together, our decoder heads synergistically balance learning a perceptual representation that is simultaneously semantic, entity-aware, and geometrically discriminative, all while maintaining strong robustness to perceptual aliasing.

2) Reward Function Model Architecture: We implement r_ϕ using a 0.5M parameter Multi-Scale Fully Convolutional Network (MS FCN) first used by Wulfmeier et al. [49]. We select this model architecture as it enforces spatial invariance and considers features at multiple scales.

B. CRESTE *Training Procedure*

To train CRESTE, we first optimize Θ and freeze the parameters before training r_ϕ using our active reward learning framework with counterfactuals. Altogether, our full learning objective is:

$$\mathcal{L}_{\text{CRESTE}} = \mathcal{L}_{\text{rgbd}} + \mathcal{L}_{\text{bev}} + \mathcal{L}_{\text{IRL}} \tag{3}$$

where $\mathcal{L}_{\text{rgbd}}$ and $\mathcal{L}_{\text{bev}}$ supervise Θ and $\mathcal{L}_{\text{IRL}}$ supervise r_ϕ. Sec. IV-B1 breaks down the objective terms $\mathcal{L}_{\text{rgbd}}$ and $\mathcal{L}_{\text{bev}}$ and how we leverage VFMs to generate training labels. Sec. IV-B2 defines our counterfactual-based IRL objective $\mathcal{L}_{\text{IRL}}$, procedure for obtaining counterfactual annotations, and theoretical derivation for $\mathcal{L}_{\text{IRL}}$.

1) Training the Perceptual Encoder Θ: To train Θ, we propose a distillation label generation module that leverages large VFMs to generate training labels. Our labels enable us to learn representations that are robust to perceptual aliasing, spatially grounded, and rich in navigation priors. As shown in Fig. 2, we generate semantic feature maps $\hat{y}_{\text{semantic}}$ and completed depth labels $\hat{y}_{\text{depth}}$ to distill semantic and geometric priors to f_{rgbd}. Then, we generate BEV map labels for static and dynamic entities ($\hat{y}_{\text{bev, static}}$, $\hat{y}_{\text{bev, dynamic}}$) and BEV elevation map labels $\hat{y}_{\text{bev, elev}}$ to distill entity priors and enforce robustness to lifting artifacts when training f_{bev}. Next, we present our distillation label generator and training procedure for f_{rgbd} and f_{bev}.

Distillation Label Generator. Fig. 2 visually depicts the inputs and outputs for our label generator. Using sequential SE(3) robot poses and synchronized RGB–point cloud pairs ($o_{\text{rgb, 1:t}}$, $o_{\text{cloud, 1:t}}$), our Distillation Label Generator produces five training labels for supervising Θ. Below, we detail the label generation procedure for a single timestep t, separating it into two parts: (i) labels supervising f_{rgbd}, and (ii) labels supervising f_{bev}.

i) Generating Training Labels for the RGB-D Encoder f_{rgbd}. We generate $\hat{y}_{\text{semantic}}$ by passing $o_{\text{rgb, t}}$ through a frozen Dinov2 encoder and bilinearly interpolating the spatial dimension of our output feature map to match the spatial resolution of y_{semantic}. We generate $\hat{y}_{\text{depth}}$ by projecting $o_{\text{cloud, t}}$ to the image and applying bilateral filtering [31] for edge-aware inpainting, producing a dense depth map that respects object boundaries.

ii) Generating Training Labels for the BEV Inpainting Backbone f_{bev}. To generate $\hat{y}_{\text{bev, dynamic}}$, we prompt SAM2 with bounding box detections from a set of commonly encountered dynamic categories (e.g. vehicles, pedestrians) to obtain dynamic entity labels. We follow the approach in Osep. et al. [29], backprojecting dynamic labels to 3D using the corresponding point cloud $o_{\text{cloud, t}}$ and clustering the points at multiple density thresholds using DBSCAN [7]. We retain the DBSCAN clusters that exceed a minimum intersection-over-union (IoU) overlap with the entity-labeled clusters and project the matched points to the dynamic entity BEV map.

To generate $\hat{y}_{\text{bev, static}}$, we first obtain static entity labels by prompting SAM2 with a grid of query points (generating dynamic entity labels via bounding box queries as before), mask

out overlapping regions between grid-queried and dynamic labels to isolate the static masks in each frame, and then apply an iterative greedy merging strategy that fuses overlapping masks (above an IoU threshold) across consecutive frames — treating unmatched IDs as new entities. Finally, we project and accumulate these entity-consistent multi-frame labels in BEV using known robot poses to produce the final static entity BEV map. To see our algorithmic formulation describing this procedure, please refer to the Appendix.

To generate $\hat{y}_{\text{bev, elev}}$, we accumulate static entity-labeled 3D points across sequential frames using robot poses (generating static entity labels as before), assign each 3D point to a grid cell, and compute each cell's minimum elevation by averaging the N lowest 3D points that fall into that cell.

Training the RGB-D Encoder f_{rgbd}. We jointly train f_{rgbd} via backpropagation from f_{semantic} and f_{depth}. First, the semantic decoder head f_{semantic} is supervised via an L2/MSE loss that minimizes the error between y_{semantic} and $\hat{y}_{\text{semantic}}$. Second, the depth completion head f_{depth} is supervised using a cross-entropy classification loss $\mathcal{L}_{\mathcal{CE}}$, where $\hat{y}_{\text{depth}}$ is uniformly discretized into bins. Empirically, this captures depth discontinuities better than regression [34]. Altogether the training objective for the RGB-D backbone is:

$$\mathcal{L}_{\text{rgbd}} = \alpha_1 l_2(y_{\text{semantic}}, \hat{y}_{\text{semantic}}) + \alpha_2 \mathcal{L}_{\mathcal{CE}}(y_{\text{depth}}, \hat{y}_{\text{depth}}) \quad (4)$$

where α_1 and α_2 are tunable hyperparameters.

Training the BEV Inpainting Backbone f_{bev}. We train f_{bev} with three losses, one for each BEV decoder head, that leverage $\hat{y}_{\text{bev, static}}$, $\hat{y}_{\text{bev, dynamic}}$, and $\hat{y}_{\text{bev, elev}}$ to distill entity priors, mitigate projection artifacts from noisy depth predictions, and teach our model to reason about occluded areas. We train $f_{\text{bev, static}}$ and $f_{\text{bev, dynamic}}$ with Supervised Contrastive Loss [18] ($\mathcal{L}_{\text{contrastive}}$) using BEV map labels $\hat{y}_{\text{bev, static}}$ and $\hat{y}_{\text{bev, dynamic}}$. $\mathcal{L}_{\text{contrastive}}$ optimizes the embedding space such that features of the same entity remain close while repelling those from different entities. This is key to learning continuous feature maps from discrete BEV map labels. We train $f_{\text{bev, elev}}$ to predict $\hat{y}_{\text{bev, elev}}$ using l_1 regression loss. Our full training objective for f_{bev} is:

$$\begin{aligned}\mathcal{L}_{\text{bev}} = \beta_1 \mathcal{L}_{\text{contrastive}}(y_{\text{static}}, \hat{y}_{\text{static}}) + \\ \beta_2 \mathcal{L}_{\text{contrastive}}(y_{\text{dynamic}}, \hat{y}_{\text{dynamic}}) + \beta_3 l_1(y_{\text{elev}}, \hat{y}_{\text{elev}})\end{aligned} \quad (5)$$

where β_1, β_2, and β_3 are tunable hyperparameters. Trained together, $\mathcal{L}_{\text{rgbd}}$ and $\mathcal{L}_{\text{bev}}$ guide the network to learn robust semantic, geometric, and entity-level representations, even for regions that are partially or fully hidden from direct view. For specific hyperparameter settings, please refer to Appendix Sec. IX-A.

2) Training the Reward Function r_ϕ: Even with perceptual representations that encode relevant features for navigation, prior works [8, 15] demonstrate that is difficult to identify the most salient features and how they influence operator preferences. This problem is exacerbated for real-world deployments, where limited expert demonstrations further reduce generalizability. To improve policy learning under limited expert demonstrations, we propose an active learning framework that supplements expert demonstrations with counterfactual annotations obtained via offline operator feedback. Our approach teaches policies to learn rewards that simultaneously penalize suboptimal behavior while rewarding expert-aligned behavior, thereby improving sample efficiency, generalizability, and interpretability.

More formally, suboptimal trajectories enable learning more discriminative reward functions that map scene features to their scalar utilities. Let us define a state-action visitation distribution to be the discounted probability of reaching a state s under a policy π and taking action a: $\rho^\pi(s,a) = (1-\gamma)\sum \gamma^t p(s_t = s, a_t = a|\pi)$. We define each state s to be an xy location on the local BEV grid and our actions a along the 8 connected grid. Overall, our objective for reward learning, referred to as Counterfactual IRL, is simple: For each BEV observation y_{bev}, given state-actions sampled from the expert's visitation distribution ρ^E and state-actions sampled from the suboptimal visitation distribution ρ^S; we obtain a reward function and policy by solving the following optimization problem:

$$\begin{aligned}&\min_{\pi} \max_{\phi} \mathbb{E}_{\rho^E}[r_\phi(s,a,y_{bev})] \\ &- (\alpha \mathbb{E}_{\rho^S}[r_\phi(s,a,y_{bev})] + (1-\alpha)\mathbb{E}_{\rho^\pi}[r_\phi(s,a,y_{bev})])\end{aligned} \quad (6)$$

where ρ_π denotes current policy visitation and α is a tunable hyperparameter that balances the relative importance between suboptimal and expert demonstrations. Using the relationship defined in Eq. 10, we can rewrite Eq. 6 in terms of the state-action visitation distribution to obtain our Counterfactual IRL training objective:

$$\mathcal{L}_{\text{IRL}} = \rho^E(s,a) - (\alpha \rho^S(s,a) + (1-\alpha)\mathbb{E}_\pi[\rho^\pi(s,a)])r_\phi \quad (7)$$

Intuitively, the above objective learns a reward function such that the difference between the expert's return and agent policy's return is minimized while ensuring suboptimal counterfactuals have a low return. Assuming non-trivial rewards, which can be enforced using gradient regularization techniques [22], our objective effectively leverages expert and counterfactual demonstrations to learn more discriminative rewards. Next, we describe our active learning framework for learning r_ϕ from counterfactuals and how we generate these counterfactual annotations. We conclude by deriving the reward learning objective using the Bradley-Terry model of preferences and show connections to Inverse Reinforcement Learning (IRL).

Active Reward Learning from Counterfactuals. We introduce an active reward learning framework that leverages counterfactual demonstrations to teach our reward function r_ϕ to map features from our BEV feature map y_{bev} to operator-aligned scalar rewards. Fig. 3 illustrates this framework, which consists of three phases that we shall describe next:

Phase I: Warmstart We first set α equal to zero in $\mathcal{L}_{\text{IRL}}$, effectively training r_ϕ using only expert demonstrations. This provides a base policy to use for the next phase.

Phase II: Synthetic Counterfactual Generation For each training sample, we infer rewards using r_ϕ from Phase I and

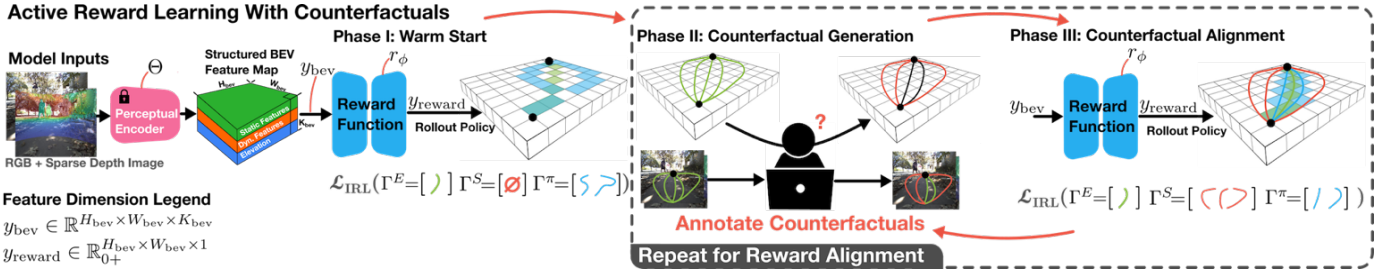

Fig. 3: Framework for Active Reward Learning with Counterfactuals. Before reward learning, we train and freeze the perceptual encoder Θ and use the output BEV feature map y_{bev} with r_ϕ to predict BEV reward maps y_{reward}. In phase I, we train r_ϕ using our counterfactual IRL objective $\mathcal{L}_{\text{IRL}}$ using only expert demonstrations Γ^E. In phase II, we plan goal-reaching paths using y_{reward} and identify samples that align poorly with human preferences. We generate alternate trajectories for these samples and query the operator to select counterfactual demonstrations Γ^S using o_{rgb} for context. In phase III, we retrain r_ϕ using $\mathcal{L}_{\text{IRL}}$, this time with Γ^E and Γ^S. We repeat phases II and III to iteratively improve r_ϕ until it is aligned with human preferences.

plan trajectories using the same start and goal as the expert trajectory. For samples where the learned and expert policies diverge, we generate alternate trajectories from the start to the goal and prompt the operator to identify which alternate trajectories are counterfactuals (i.e exhibit unwanted behavior).

Phase III: Counterfactual Reward Alignment We retrain r_ϕ using $\mathcal{L}_{\text{IRL}}$ with expert and counterfactual annotations. We set α to be nonzero to balance the relative importance of expert and counterfactual demonstrations. We repeat phases II and III using the current r_ϕ until r_ϕ aligns with operator preferences.

Generating Counterfactual Annotations. To obtain counterfactual annotations for a training sample t in our dataset, we first require the expert trajectory Γ^E_t, $o_{\text{rgb, t}}$, and $o_{\text{depth, t}}$ for the sample. We uniformly sample a handful of "control" states along Γ^E_t, excluding the start and goal states. Then, we randomly perturb these control states before planning a kinematically feasible path from the start and goal states such that it reaches all of the perturbed control states. Practically, we implement this using Hybrid A* [6] - however, any kinematic planner will suffice. We generate a handful of alternate trajectories in this manner and prompt the operator to identify the suboptimal trajectories given $o_{\text{rgb, t}}$ and $o_{\text{depth, t}}$. We use the selected suboptimal trajectories to retrain r_ϕ in our active reward learning framework. For additional implementation details regarding generating alternate trajectories, we refer readers to Appendix Sec. IX-B.

Counterfactual IRL Derivation $\mathcal{L}_{\text{IRL}}$. In this section, we derive $\mathcal{L}_{\text{IRL}}$, our IRL objective that leverages expert and counterfactual demonstrations. For context, IRL methods [50, 26] allow for learning reward functions r_ϕ, parameterized by ϕ, given expert demonstrations. However, they provide no mechanism to incorporate suboptimal trajectories. Suboptimal trajectories are easy to obtain and enable a data flywheel for navigation; the BEV observations obtained from expert runs can simply be relabelled with suboptimal of unsafe trajectories offline without any more environmental interactions. Motivated by this idea, we derive $\mathcal{L}_{\text{IRL}}$, a general and principled way to learn from suboptimal and expert trajectories jointly under a single objective.

Our approach builds on the ranking perspective of imitation learning [42] which uses visitation distributions to denote long-term behavior of an agent. We denote $\rho^\pi(s,a)$, $\rho^E(s,a)$, $\rho^S(s,a)$ to be the agent, expert, and suboptimal state-action visitation distributions respectively. We assume the reward function is conditioned on y_{bev} as before, but drop it from the derivation for conciseness. Under this notation the problem of return maximization becomes finding a visitation induced by a policy π that maximizes the expected return given by:

$$\max_\pi J^\pi(r_\phi) = \max_\pi \mathbb{E}_{\rho^\pi(s,a)}[r_\phi(s,a)]. \tag{8}$$

The reward function of the expert should satisfy the ranking $\rho^\pi(s,a) \preceq \rho^E(s,a)$, which implies that the expert's visitation distribution obtains a return that is greater or equal to any other policy's visitation distribution in the environment:

$$\rho^\pi(s,a) \preceq \rho^E(s,a) \implies \mathbb{E}_{\rho^\pi(s,a)}[r_\phi(s,a)] \leq \mathbb{E}_{\rho^E(s,a)}[r_\phi(s,a)]. \tag{9}$$

This property extends to any suboptimal visitations, and as a consequence of linearity of expectations, to any convex combination of the current policy's visitation and any other suboptimal visitation distribution. Mathematically,

$$\alpha\rho^\pi(s,a) + (1-\alpha)\rho^S(s,a) \preceq \rho^E(s,a)\ \forall\alpha \in [0,1]. \tag{10}$$

Thus, given suboptimal visitation distributions, we can create a number of pairwise preferences by choosing a suboptimal visitation and a particular α. We turn to the Bradley-Terry model of preferences to satisfy these pairwise preferences which assumes that preferences are noisy-rational and that the probability of a preference can be expressed as:

$$\begin{aligned} P(\rho^E(s,a) \succeq \alpha\rho^\pi(s,a) + (1-\alpha)\rho^S(s,a)) = \\ \frac{e^{J^E(r_\phi)}}{e^{J^E(r_\phi)} + e^{\alpha J^\pi(r_\phi) + (1-\alpha)J^S(r_\phi)}} \\ = \frac{1}{1 + e^{\alpha(J^S(r_\phi) - J^E(r_\phi)) + (1-\alpha)(J^\pi(r_\phi) - J^E(r_\phi))}}. \end{aligned} \tag{11}$$

Finding a reward function implies maximizing the likelihood of observed preferences while the policy optimizes the learned reward function. Since, the convex combination holds for all values of $\alpha \in [0,1]$, we consider optimizing against the worst-case to obtain the following two-player

counterfactual IRL objective, where the reward player learns to satisfy rankings against the worst-possible α:

$$\max_{\phi} \min_{\alpha} P\left(\rho^E(s,a) \succeq \alpha\rho^\pi(s,a) + (1-\alpha)\rho^S(s,a)\right) \quad (12)$$

and the policy player maximizes expected return:

$$\max_{\pi} J^\pi(r_\phi). \quad (13)$$

In practice, optimizing for worst-case α for each BEV scene y_{bev} can quickly make solving the optimization objective challenging due to the large number of scenes we train the reward function on. We make two mild approximations that we observed to make learning more efficient and tractable: First, we replace the worst-case α with a fixed α, and second, we consider maximizing a pointwise monotonic transformation to the Bradley Terry loss function that directly maximizes $\alpha(J^S(r_\phi) - J^E(r_\phi)) + (1-\alpha)(J^\pi(r_\phi) - J^E(r_\phi))$ instead of its sigmoid transformation. With these changes, we can rewrite our practical counterfactual IRL objective as:

$$\min_{\pi} \max_{\phi} \left(J^E(r_\phi) - (\alpha J^S(r_\phi) + (1-\alpha)J^\pi(r_\phi)\right). \quad (14)$$

This objective reveals a deeper connection between apprenticeship learning (Eq 6 [1]) obtained by setting α to 0 and learning from preferences [4] obtained by setting α to 1. The loss function goes beyond the apprenticeship learning objective that only learns from expert by incorporating suboptimal demonstrations. Second, it goes beyond the offline nature of prior algorithms that learn from preferences alone by instead learning a policy that attempts to match expert visitation making use of the suboptimal demonstrations.

V. Implementation Details

In this section, we cover implementation details for our local planning and control modules depicted in Fig. 1. For additional information regarding model training hyperparameters and generating counterfactual annotations, please refer to Appendix Sec. IX.

1) Global and Local Planning and Controls: Our global and local planning modules work together to identify the next local subgoal for local path planning. Given a user-specified GPS end goal G_N, we use OpenStreetMap [27] to obtain a semi-dense sequence of coarse GPS goals $G = [G_1, ..., G_N]$ spaced 10 meters apart. To select the next GPS subgoal, we compute the set of distances $D = \{||g - G_N|| \mid g \in G \cup \{G_t\}\}$, where D contains the distance of the robot G_t from G_N and the distance of each GPS subgoal in G from G_N. From the set of subgoals with a smaller distance to G_N than G_t, we select the farthest subgoal to use as the next subgoal. We project this subgoal on the edge of the local planning horizon, a 6 meter circle around the robot, giving us a carrot for local path planning.

We adopt a DWA [9] style approach to local path planning, where we enumerate a set of constant curvature arcs (31 in our case) from the egocentric robot frame. We compute learned cost for each trajectory by sampling points along each each arc and computing the discounted cost at each point using our predicted reward map. We simply invert and normalize our reward map to the range $[0, 1]$ to convert it to a costmap. Finally, we compute the distance between the last point on the arc with the local carrot to obtain a goal-reaching cost. We multiply the learned and goal-reaching costs by tunable weights before selecting the trajectory with the lowest cost. Finally, we perform 1D time optimal control [30] to generate low-level actions to follow this trajectory. Empirically, we find that sampling 30 points and using a discount factor of 0.95 is sufficiently dense. We find that that tuning the goal-reaching cost to be 1/10th the importance of the learned cost achieves good balance between goal-reaching and adhering to operator preferences.

Fig. 4: Satellite view of testing locations for short horizon mapless navigation experiments. We evaluate each baseline in 3 unseen (red) and 2 seen (green) urban locations. Our testing locations consist of residential neighborhoods, urban shopping centers, urban parks, and offroad trails. In each location, each baseline must start from an endpoint on the annotated blue trajectory and navigate to the opposite end of the trajectory.

VI. Experiments

In this section, we describe our evaluation methodology for CREStE and answer the following questions to understand the importance of our contributions and overall performance on the task of mapless urban navigation.

- (Q_1) How well does CREStE generalize to unseen urban environments for mapless urban navigation?
- (Q_2) How important are structured BEV perceptual representations for downstream policy learning?

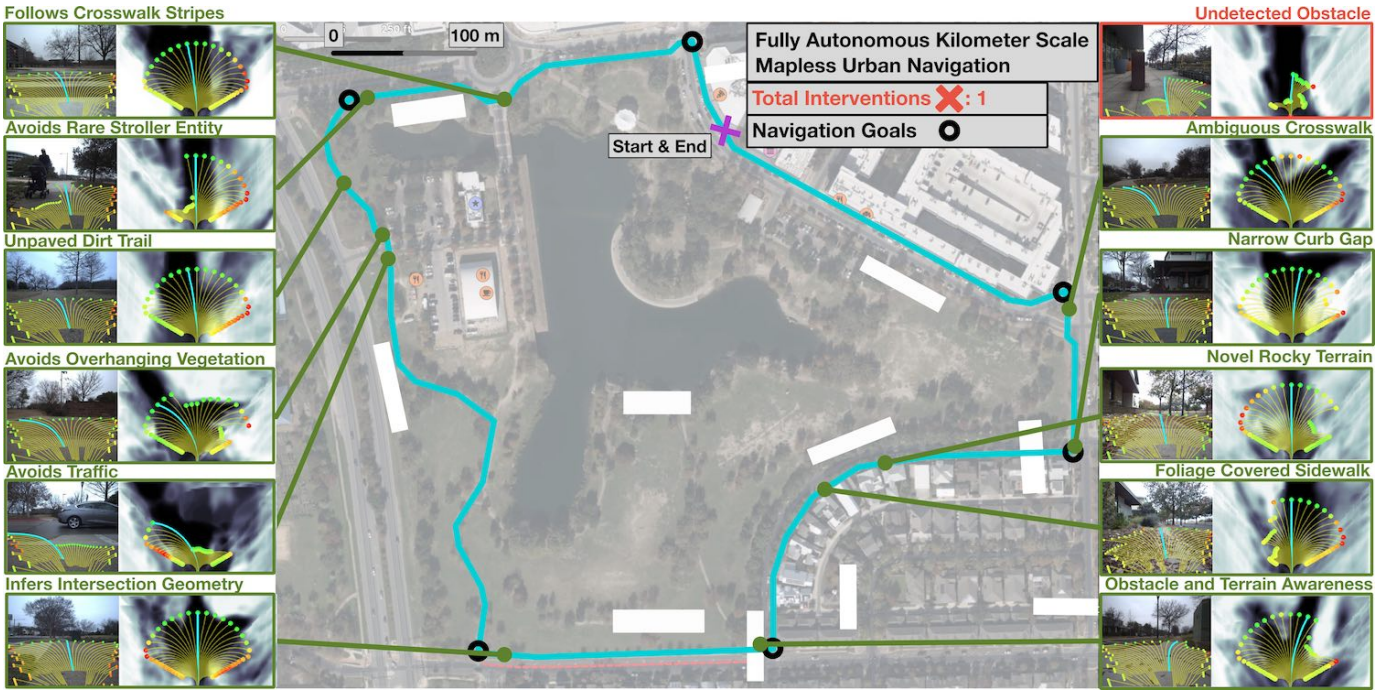

Fig. 5: Satellite image of our 2 kilometer long-horizon testing area, with examples with front view RGB image observations and CRESTE's predicted BEV costmap (converted from the predicted BEV reward map). We annotate successful examples in green with a brief description of the situation. We annotate unsuccessful examples in red, present the observation right before the intervention, and provide a brief description of the cause of failure.

- (Q_3) How much do counterfactual demonstrations improve learned policies in challenging urban scenes?
- (Q_4) How well does CRESTE perform long horizon mapless urban navigation compared to other top state-of-the-art approaches?

Fig. 6: Mobile robot testing platform for real-world experiments. We annotate the locations of the monocular camera, 3D LiDAR, and cellular phone (not visible) on our testing platform, the Clearpath Jackal.

We investigate $\mathcal{Q}_1$ by comparing CRESTE's performance against other methods in unseen environments. To answer $\mathcal{Q}_2$ and $\mathcal{Q}_3$, we conduct ablation studies that isolate the methodological contribution in question. Finally, we evaluate $\mathcal{Q}_4$ by conducting a kilometer-scale experiment comparing CRESTE against the top performing baseline.

A. Robot Testing Platform

We conduct all experiments using a Clearpath Jackal mobile robot. We observe 512×612 RGB images from a 110° field-of-view camera and point cloud observations from a 128-channel Ouster LiDAR. We obtain coarse GPS measurements and magnetometer readings from a cellular smartphone. We use the open source OpenStreetMap [27] routing service to obtain coarse navigation waypoints. Our onboard compute platform has an Intel i7-9700TE 1.80 GHz CPU and Nvidia RTX A2000 GPU. We run CRESTE at 20 Hz alongside our mapless navigation system, which operates at 10Hz.

B. Training Dataset

We collect a robot dataset with 3 hours of expert navigation demonstrations spread across urban parks, downtown centers, residential neighborhoods, and college campuses. Our dataset consists of synchronized image LiDAR observation pairs and ground truth robot poses computed using LeGO-LOAM [41], a LiDAR-based SLAM algorithm. We train all methods on the same dataset for 150 epochs or until convergence.

C. Testing Methodology

We evaluate all questions through physical robot experiments performing the task of mapless urban navigation in urban environments. We present aerial images in Fig. 4 depicting the five urban environments that we perform short horizon ($\sim$100m) quantitative experiments. These consist of five challenging seen and unseen environments with trails, road crossings, narrow pathways, different terrains, and obstacle hazards. For each environment, we repeat the same experiment twice for each approach. Additionally, we evaluate the highest performing methods on a long-horizon quantitative experiment, which we conduct at a 2-kilometer urban trail shown in Fig. 5. We pre-emptively terminate the long-horizon experiment if the baseline is unable to complete the mission within a predefined time. Next, we explain evaluation metrics 1 to 3, which we use for the short-horizon experiments, and

	In Distribution						Out of Distribution								
Location	Sherlock North			Woolridge Square			Sherlock South			Trader Joes			Hemphill Park		
Method	AST ↓	%S↑	NIR ↓	AST ↓	%S↑	NIR ↓	AST ↓	%S↑	NIR ↓	AST ↓	%S↑	NIR ↓	AST ↓	%S↑	NIR ↓
Geometric Only	17.64	61.60	11.21	18.56	86.36	9.50	15.56	92.50	7.80	15.22	94.74	12.05	14.21	95.00	13.21
PACER+G [23]	25.79	96.15	9.42	26.51	90.90	8.83	22.11	95.00	7.67	24.38	87.14	17.93	23.94	92.85	9.47
ViNT [40]	20.38	65.51	10.36	17.85	90.36	7.94	17.06	92.30	7.19	22.10	84.21	18.36	23.92	95.00	10.96
PIVOT [25]	46.56	83.33	26.03	57.36	84.11	20.22	45.41	75.00	21.91	41.77	74.44	40.67	33.85	85.00	28.36
CRESTE - cfs	25.86	96.29	9.20	19.84	90.90	6.12	13.27	92.30	2.41	25.39	91.66	10.49	28.43	83.33	8.57
CRESTE - cfs - st	21.26	96.15	12.99	20.53	91.66	9.79	21.13	92.80	5.65	23.17	93.75	14.87	26.85	94.11	11.11
CRESTE (ours)	**14.01**	**96.60**	**4.60**	**14.70**	**100.0**	**0.00**	**12.60**	**95.23**	**0.86**	**15.28**	**95.65**	**3.73**	**15.42**	**100.0**	**0.00**

TABLE I: Quantitative evaluation for short horizon mapless navigation experiments. In distribution locations are present in the training dataset while out-of-distribution locations are absent from the training data. We bold the best performing method for each metric in each location, and annotate each metric with an up or down arrow to indicate if higher or lower numbers are better. We define the following evaluation metrics in the evaluation metric section of this work and denote their abbreviations as: AST - average subgoal completion time (s), %S - percentage of subgoals reached in mission, NIR - Number of interventions required per 100 meters driven.

Location	Mueller Loop				
Method	AST ↓	%S↑	NIR ↓	Dist. (m) ↑	Total Int. ↓
PACER+G [23]	15.8	61.53	3.56	1345.07	48
CRESTE (ours)	**12.94**	**99.45**	**0.052**	**1919.44**	**1**

TABLE II: Quantitative evaluation for long horizon mapless navigation experiments. All baselines are evaluated on the same 1.9 kilometer urban area. We bold the best performing method for each metric in each location, and annotate each metric with an up or down arrow to indicate if higher or lower numbers are better. We define the following evaluation metrics in the evaluation metric section of this work and denote their abbreviations as: AST - average subgoal completion time (s), %S - percentage of subgoals reached in mission, NIR - Number of interventions required per 100 meters driven, Dist. (m) - total distance driven in meters, Total Int. - total number of interventions required for the entire mission.

evaluation metrics 4 and 5, which we use exclusively for the long horizon experiments.

Our evaluation metrics are defined as follows: 1) *Average Subgoal Completion Time (AST)* - the average time to complete each subgoal where all subgoals are evenly spaced 10 meters apart 2) *Percentage of Subgoals Reached (%S)* - the percentage of subgoals reached by the end of the mission 3) *Normalized Intervention Rate (NIR)* - the number of operator interventions required for every 100 meters driven 4) *Total Distance Driven (Dist. (m))* - the total distance driven before mission failure or completion 5) *Total Interventions (Total Int.)* - the total number of interventions required per mission. During evaluation, we only consider interventions that incurred by the method (e.g. overrides performed to give right of way to vehicular traffic are not considered interventions).

D. Baselines

We supplement all baselines with the same analytical geometric avoidance module to prevent catastrophic collisions. Even with this module, operators preemptively intervene when the baseline deviates from general navigation preferences (e.g. stay on sidewalks, follow crosswalk markings, etc). All baselines, with the exception of PIVOT [25], perform all computations using onboard compute to fairly evaluate real-time performance. Next, we describe our evaluation baselines.

For questions 1 and 4, we compare CRESTE with the following baselines: 1) *ViNT* [40] - a foundation navigation model trained on goal-guided navigation 2) *PACER+Geometric* [23] (PACER+G) - a multi-factor perception baseline that considers learned terrain and geometric costs in a dynamic window [9] (DWA) style approach 3) *Geometric-Only* - a DWA style approach that only considers geometric costs 4) *PIVOT* [25] - a VLM-based navigation method that uses the latest version of GPT4o-mini [2] to select local goals from image observations.

Additionally, we evaluate CRESTE and two modified CRESTE architectures: 1) CRESTE - cfs, the CRESTE model trained without counterfactual demonstrations, but otherwise identical to the original. 2) CRESTE- cfs - st, the CRESTE model trained without counterfactual demonstrations or the BEV inpainting backbone. Instead, we directly pass unstructured BEV feature maps $z_{\text{bev, splat}}$ to the reward function r_ϕ and train the splat module f_{splat} using expert demonstrations. We still freeze the RGB-D backbone f_{rgbd} when training r_ϕ. We provide more detailed analysis in the Appendix and describe high-level implementation details for each baseline next.

Since ViNT is pre-trained for image goal navigation, we are unable to deploy this model in unseen environments where image goals are not known apriori. Thus, we finetune ViNT by freezing the pre-trained ViNT backbone and changing the goal modality encoder to accept 2D xy coordinate goals rather than images. We follow the same model architecture and match the training procedure from the original paper for reaching GPS goals. We reproduce the PACER and PIVOT models faithfully to the best of our abilities as no open-source implementation is available.

E. Quantitative Results and Analysis

We present the quantitative results of the short horizon and long-horizon experiments in Table I and Table II respectively and present our analysis for $\mathcal{Q}_1$ - $\mathcal{Q}_4$ in the following section.

1) Evaluating Generalizability to Unseen Urban Environments.: Comparing CRESTE against all other baselines in Table I, we find that our approach achieves superior performance in all metrics for both seen and unseen environments. We

present qualitative analysis for each learned baseline in Appendix Sec. IX-C. Quantitatively, PACER+G, the next best approach, requires two times more interventions than CRESTE in seen environments and fives times more interventions in unseen environments. Furthermore, we find that PIVOT, our VLM-based navigation baseline, is outperformed by all other methods, corroborating our claim that VLMs and LLMs are not attuned for urban navigation despite containing internet-scale priors. We hypothesize this is because navigation requires identifying which priors are most important for navigation, a task seen rarely by VLMs and LLMs during pre-training. Notably, we achieve an intervention-free traversal in one unseen environment (Hemphill Park), a residential park with diverse terrains, narrow curb gaps, and crosswalk markings. Thus, we conclude that CRESTE is remarkably more generalizable for mapless urban navigation than existing approaches.

2) Evaluating the Importance of Structured BEV Perceptual Representations.: We assess the impact of structured perceptual representations by evaluating CRESTE - cfs - st against CRESTE - cfs in Table I, where the only difference is the structured representation. On average, we observe that structured representations incur 28% fewer interventions in seen environments and 34% fewer interventions in unseen environments. We hypothesize this occurs because structured representations encode higher level features that are more generalizable and easier for policies to reason about, compared to lower level features that capture less generalizable high-frequency information.

3) Evaluating the Importance of Counterfactual Demonstrations.: We compare CRESTE against CRESTE-cfs, the exact same approach but without using counterfactuals to train the reward function. On average, we find in Table I that using counterfactuals reduces the number of interventions by 70% in seen environments and 69% in unseen environments. Both of these baselines distill the same features for navigation from VFMs, thus vindicating our claim that even with sufficiently informative perceptual representations, it is important to leverage our counterfactual-based objective to properly identify and reason about the most salient features for navigation.

4) Evaluating Performance on Kilometer-Scale Mapless Navigation.: Table II compares CRESTE against the top-performing baseline from Table I, PACER+G, on long horizon mapless navigation. Fig. 5 show qualitative examples of CRESTE's predicted BEV costmaps along the route. While PACER+G still drives over a kilometer before timing out, it requires significantly more interventions to do so. We observe that the majority of PACER+G failures that occur result from either not adequately considering geometric factors like curb gaps or predicting poor costmaps due to differences in lighting and weather compared to the conditions from which the training dataset was collected. Contrastingly, CRESTE is robust to perceptual aliasing from lighting and weather variations and predicts accurate costmaps even during failures. From this, we conclude that CRESTE achieves superior performance in long-horizon mapless urban navigation as well.

VII. Limitations and Future Work

CRESTE infers reward maps using observations from a single timestep, discarding past information. This limits our approach in scenes that require temporal reasoning, such as recovering from dead ends or negotiating paths with dynamic entities like pedestrians. Extending CRESTE to consider multiple observations is a promising direction that addresses these issues. Extending CRESTE's counterfactual-based IRL objective to reason about temporal dynamics is another promising direction for improving performance in crowded dynamic environments. Additionally, CRESTE must be manually tuned to balance goal-reaching and reward-map aligned behavior, which poses an issue for urban environments with sporadic global guidance. Extending CRESTE to jointly reason about task-specific goals along with navigation affordances will further improve real-world robustness for mapless urban navigation. While the IRL objective presented in the paper only considers expert and suboptimal demonstrations, this framework can be extended to incorporate preferences following the same derivation from the ranking perspective. This would not only allow the reward to reason about what paths are best but allow is to learn how to discriminate between two suboptimal paths should the need arise. From the perceptual aspect, a promising future direction is to extend CRESTE to learn from large-scale egocentric videos from the internet to increase robustness to semantic features.

VIII. Conclusion

In this paper, we introduce Counterfactuals for Reward Enhancement with Structured Embeddings (CRESTE), the first framework for learning representations and policies for the full mapless navigation problem. CRESTE learns robust, generalizable perceptual representations with internet-scale semantic, geometric, and entity priors by distilling features from multiple visual foundation models. We demonstrate that our perceptual representation encodes a sufficient set of factors for urban navigation, and introduce a novel counterfactual-based loss and active learning framework that teaches policies to hone in on the most important factors and infer how they influence fine-grained navigation behavior. These contributions culminate to form CRESTE, a local path planning module that significantly outperforms state-of-the-art alternatives on the task of long-horizon mapless urban navigation. Through kilometer-scale real-world robot experiments, we demonstrate our approach's effectiveness, gracefully navigating an unseen 2 kilometer urban environment with only a handful of interventions.

References

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal

Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[3] Joydeep Biswas and Manuela Veloso. The 1,000-km challenge: Insights and quantitative and qualitative results. *IEEE Intelligent Systems*, 31(3):86–96, 2016. doi: 10.1109/MIS.2016.53.

[4] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International conference on machine learning*, pages 783–792. PMLR, 2019.

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[6] Dmitri Dolgov, Sebastian Thrun, Michael Montemerlo, and James Diebel. Practical search techniques in path planning for autonomous driving. *Ann Arbor*, 1001 (48105):18–80, 2008.

[7] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[8] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58. PMLR, 2016.

[9] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[10] Nikhil Gosala, Kürsat Petek, Paulo LJ Drews-Jr, Wolfram Burgard, and Abhinav Valada. Skyeye: Self-supervised bird's-eye-view semantic mapping using monocular frontal view images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14901–14910, 2023.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] Noriaki Hirose, Catherine Glossop, Ajay Sridhar, Dhruv Shah, Oier Mees, and Sergey Levine. Lelan: Learning a language-conditioned navigation policy from in-the-wild videos. *arXiv preprint arXiv:2410.03603*, 2024.

[13] Sanghun Jung, JoonHo Lee, Xiangyun Meng, Byron Boots, and Alexander Lambert. V-strong: Visual self-supervised traversability learning for off-road navigation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1766–1773. IEEE, 2024.

[14] 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.

[15] Gregory Kahn, Pieter Abbeel, and Sergey Levine. Land: Learning to navigate from disengagements. *IEEE Robotics and Automation Letters*, 6(2):1872–1879, 2021.

[16] Haresh Karnan, Elvin Yang, Daniel Farkash, Garrett Warnell, Joydeep Biswas, and Peter Stone. Sterling: Self-supervised terrain representation learning from unconstrained robot experience. In *7th Annual Conference on Robot Learning*, 2023.

[17] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. In *2019 international conference on robotics and automation (ICRA)*, pages 8248–8254. IEEE, 2019.

[18] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673, 2020.

[19] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9404–9413, 2019.

[20] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023.

[21] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[22] Yecheng Jason Ma, Andrew Shen, Dinesh Jayaraman, and Osbert Bastani. Smodice: Versatile offline imitation learning via state occupancy matching. *arXiv preprint arXiv:2202.02433*, 1(2):3, 2022.

[23] Luisa Mao, Garrett Warnell, Peter Stone, and Joydeep Biswas. Pacer: Preference-conditioned all-terrain costmap generation. *arXiv preprint arXiv:2410.23488*, 2024.

[24] 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.

[25] Soroush Nasiriany, Fei Xia, Wenhao Yu, Ted Xiao, Jacky Liang, Ishita Dasgupta, Annie Xie, Danny Driess, Ayzaan Wahid, Zhuo Xu, et al. Pivot: Iterative visual prompting elicits actionable knowledge for vlms. *arXiv preprint arXiv:2402.07872*, 2024.

[26] Tianwei Ni, Harshit Sikchi, Yufei Wang, Tejus Gupta, Lisa Lee, and Ben Eysenbach. f-irl: Inverse reinforcement learning via state marginal matching. In *Conference on Robot Learning*, pages 529–551. PMLR, 2021.

[27] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2017.

[28] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fer-

nandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[29] Aljoša Ošep, Tim Meinhardt, Francesco Ferroni, Neehar Peri, Deva Ramanan, and Laura Leal-Taixé. Better call sal: Towards learning to segment anything in lidar. In *European Conference on Computer Vision*, pages 71–90. Springer, 2025.

[30] Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. Routledge, 2018.

[31] Cristiano Premebida, Luis Garrote, Alireza Asvadi, A Pedro Ribeiro, and Urbano Nunes. High-resolution lidar-based depth mapping using bilateral filter. In *2016 IEEE 19th international conference on intelligent transportation systems (ITSC)*, pages 2469–2474. IEEE, 2016.

[32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[33] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[34] Cody Reading, Ali Harakeh, Julia Chae, and Steven L Waslander. Categorical depth distribution network for monocular 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8555–8564, 2021.

[35] Jacques Richalet, André Rault, JL Testud, and J Papon. Model predictive heuristic control. *Automatica (journal of IFAC)*, 14(5):413–428, 1978.

[36] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[37] 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.

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

[39] Dhruv Shah, Benjamin Eysenbach, Gregory Kahn, Nicholas Rhinehart, and Sergey Levine. Ving: Learning open-world navigation with visual goals. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13215–13222. IEEE, 2021.

[40] 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.

[41] Tixiao Shan and Brendan Englot. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765. IEEE, 2018.

[42] Harshit Sikchi, Akanksha Saran, Wonjoon Goo, and Scott Niekum. A ranking game for imitation learning. *arXiv preprint arXiv:2202.03481*, 2022.

[43] Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. *Advances in neural information processing systems*, 29, 2016.

[44] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[45] Sebastian Thrun, Maren Bennewitz, Wolfram Burgard, Armin B Cremers, Frank Dellaert, Dieter Fox, Dirk Hahnel, Charles Rosenberg, Nicholas Roy, Jamieson Schulte, et al. Minerva: A second-generation museum tour-guide robot. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, volume 3. IEEE, 1999.

[46] Kasun Weerakoon, Adarsh Jagan Sathyamoorthy, Utsav Patel, and Dinesh Manocha. Terp: Reliable planning in uneven outdoor environments using deep reinforcement learning. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 9447–9453. IEEE, 2022.

[47] Kasun Weerakoon, Mohamed Elnoor, Gershom Seneviratne, Vignesh Rajagopal, Senthil Hariharan Arul, Jing Liang, Mohamed Khalid M Jaffar, and Dinesh Manocha. Behav: Behavioral rule guided autonomy using vlms for robot navigation in outdoor scenes. *arXiv preprint arXiv:2409.16484*, 2024.

[48] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*, 2015.

[49] Markus Wulfmeier, Dominic Zeng Wang, and Ingmar Posner. Watch this: Scalable cost-function learning for path planning in urban environments. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2089–2095. IEEE, 2016.

[50] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

IX. Appendix

We organize the appendix into the following sections: 1) Details for training CRESTE and other baselines in Sec. IX-A, 2) Details regarding generating counterfactual annotations in Sec. IX-B, and 3) Qualitative analysis for the short horizon experiments in Sec. IX-C.

A. Model Training Details

In this section, we supplement the implementation details for CRESTE and each baseline described in Sec. VI-D.

1) CRESTE*:* We provide specific architecture and training hyperparameters settings in Table III and Table IV, and supplement the CRESTE training procedure in Sec. IV-B1 with additional details for training our observation encoder Θ. We warm-start the RGB-D backbone for 50 epochs or until convergence. We set α_1 and α_2, the loss weights for semantic feature regression and completion, to 1 and 0.5 respectively and keep this fixed for the rest of training. After this, we freeze the RGB-D backbone and train the f_{bev} using the BEV inpainting backbone loss $\mathcal{L}_{\text{bev}}$ for five epochs before unfreezing the RGB-D backbone and training end-to-end for another 45 epochs or until convergence. This enables faster convergence by stabilizing f_{bev} before joint training. We set β_1, β_2, and β_3 to 1, 2, and 3 for $\mathcal{L}_{\text{bev}}$, but find that training remains stable for any reasonable combination of values. Finally, we empirically find that replacing the Supervised Contrastive loss [18] with Cross Entropy loss while training the dynamic panoptic head $f_{\text{bev, dynamic}}$ empirically improves overall performance. Thus, we use this model for our long-horizon navigation experiments presented in Table II.

As mentioned in Sec. IV-B2, we train our reward function r_ϕ in two phases. During both phases, we freeze Θ, thus only training r_ϕ. During initialization (Phase I), we train for 25 epochs setting $\alpha = 0$ only to use expert demonstrations. During finetuning (Phase III), we initialize r_ϕ from scratch train for 50 epochs setting $\alpha = 0.5$ and a reward regularization term [22] $\alpha_{reg} = 1.0$. Empirically, we find that our objective is stable for a range of α values and benefits from a larger reward regularization penalty to encourage more discriminative rewards. In total, we train the perceptual encoder Θ and reward function r_ϕ for a combined 150 epochs.

2) ViNT [40]: To modify ViNT to follow XY goal guidance instead of image goal guidance, we follow the architecture design details in the ViNT paper for adapting the backbone to GPS goal guidance. We only train the XY goal encoder and freeze the remaining model parameters. We sample XY goals from future odometry between 8 to 10 seconds to generate training data.

3) PIVOT [25]: We condition PIVOT using an annotated satellite view image, annotated front view image, and text instructions. As shown in Fig. 7, we annotate a satellite image containing the robot's current position, heading, and next goal GPS coordinates. Fig. 7 also presents the annotated front view RGB image with each numbered circle indicating a potential goal that our VLM must choose from. Finally, we provide the text prompt below:

Fig. 7: Qualitative Example of context images provided to PIVOT [25], our VLM-based navigation baseline. We annotate satellite images with the robot GPS and heading (red) and current GPS goal (blue). Additionally, we annotate the front view image with potential subgoals from which we prompt the model to choose from.

I am a wheeled robot that cannot go over objects. This is the image I'm seeing right now. I have annotated it with numbered circles. Each number represents a general direction I can follow. I have annotated a satellite image with my current location and direction as a red circle and arrow and the goal location as a blue circle. Now you are a five-time world-champion navigation agent and your task is to tell me which circle I should pick for the task of: going forward **X** degrees to the **Y**? Choose the best candidate number. Do NOT choose routes that go through objects AND STAY AS FAR AS POSSIBLE AWAY from untraversable terrains and regions. Skip analysis and provide your answer at the end in a json

file of this form: "points": []

where X is replaced by the degrees from the goal computed using the magnetometer heading and goal GPS location and Y is either left or right depending on the direction of the goal. During testing, we use our geometric obstacle avoidance module to safely reach the current goal selected by PIVOT.

4) PACER [23]: PACER is a terrain-aware model that predicts BEV costmaps from BEV images. It requires pre-enumerating an image context that specifies the preference ordering for terrains. In all environments, we would like the robot to prefer traversing terrains in the following preference order: sidewalk, dirt, grass, rocks. Thus, we condition PACER on this preference order for all environments.

TABLE III: Architecture Hyperparameters for CRESTE.

Hyperparameter	Value
RGB-D Encoder (f_{rgbd})	
Base Architecture	EfficientNet-B0 [44]
Number of Input Channels	4
Input Spatial Resolution	512×612
Output Spatial Resolution	128×153
Output Channel Dimension	256
Semantic Decoder Head (f_{semantic})	
Input Spatial Resolution	128×153
Input Channel Dimension	256
Output Spatial Resolution	128×153
Output Channel Dimension	128
Number of Hidden Layers	4
Depth Completion Head (f_{depth})	
Input Spatial Resolution	128×153
Input Channel Dimension	256
Output Spatial Resolution	128×153
Number of Depth Bins	128
Number of Hidden Layer	2
Depth Discretization Method	Uniform
Number of Intermediate Layers	2
Lift Splat Module (f_{splat})	
Total Input Channel Dimension	288
Input Semantic Channel Dimension	256
Input Depth Channel Dimension	32
Map Cell Resolution	$0.1m \times 0.1m \times 3$
Output Channel Dimension	96
Output Spatial Resolution	256×256
BEV Inpainting Backbone (f_{bev})	
Base Architecture	ResNet18 [11]
Input Channel Dimension	96
Input Spatial Resolution	256×256
Output Static Panoptic Channel Dimension	32
Output Dynamic Panoptic Channel Dimension	32
Output Elevation Channel Dimension	1
Reward Function (r_ϕ)	
Base Policy Architecture	Value Iteration Network [43]
Base Reward Architecture	MultiScale-FCN [48]
Input Spatial Resolution	256×256
Input Channel Dimension	65
Prepool Channel Dimensions	[64, 32]
Skip Connection Channel Dimensions	[32, 16]
Trunk Channel Dimensions	[32, 32]
Postpool Channel Dimensions	[48]
# Future Actions	50
# Actions Per State	8
Discount Factor	0.99

TABLE IV: Training Hyperparameters for CRESTE.

Hyperparameter	Value
RGB-D Backbone Training (f_{rgbd})	
Semantic Loss Weight (α_1)	2.0
Depth Completion Loss Weight (α_2)	0.5
# Training Epochs	50
Batch Size	12
Optimizer	AdamW [21]
Adam β_1	0.9
Adam β_2	0.999
Learning Rate	5×10^{-3}
Learning Rate Scheduler	Exponential
Learning Rate Gamma Decay γ	0.98
BEV Inpainting Backbone Training (f_{bev})	
Static Panoptic Loss Weight (β_1)	1.0
Dynamic Panoptic Loss Weight (β_2)	2.0
Elevation Loss Weight (β_3)	3.0
Warmup Epochs	5
# Training Epochs	50
Batch Size	24
Optimizer	AdamW [21]
Adam β_1	0.9
Adam β_2	0.999
Learning Rate	5×10^{-4}
Learning Rate Scheduler	Exponential
Learning Rate Gamma Decay γ	0.98
Reward Function Training (r_ϕ)	
Batch Size	30
Optimizer	AdamW [21]
Adam β_1	0.9
Adam β_2	0.999
Learning Rate	5×10^{-4}
Learning Rate Scheduler	Exponential
Learning Rate Gamma Decay γ	0.96
Reward Learning Weight $\mathcal{L}_{\text{IRL}}$	1.0
Reward Smoothness Penalty Weight	2.0

B. CRESTE *Counterfactual Generation Details*

We supplement Sec. IV-B2 with additional details regarding the hyperparameters used for generating counterfactual demonstrations. To encourage generating diverse counterfactuals that explore more of the state space, we perturb 3 control points along the expert trajectory according to a non-zero Gaussian distribution. More specifically, for half of the generated trajectories, we sample control points according to a positive mean $\mu = 1$ with standard deviation $\sigma = 0.5$ to perturb these trajectories to one side of the expert trajectory. We repeat this for the other half, setting $\mu = -1$ and $\sigma = 0.5$ to generate trajectories on the opposite side. Additionally, we empirically find that replacing the Hybrid A* path planner by fitting a polynomial line on the control points provides comparable performance gains and is more robust when it is difficult to find a kinematically feasible path between control points.

In total, we generate 10 alternate trajectories for each training sample. On average, we find that providing just 1-5 counterfactual annotations for each non-expert-aligned sample significantly improves reward learning for r_ϕ. Fig. 8 presents our counterfactual annotation tool showing qualitative examples of the front view and BEV image observation presented to the human annotator when selecting counterfactual trajectories.

C. *Qualitative Short Horizon Experiment Analysis*

In this section, we present qualitative analysis of the short horizon experiments for each location. For each experiment

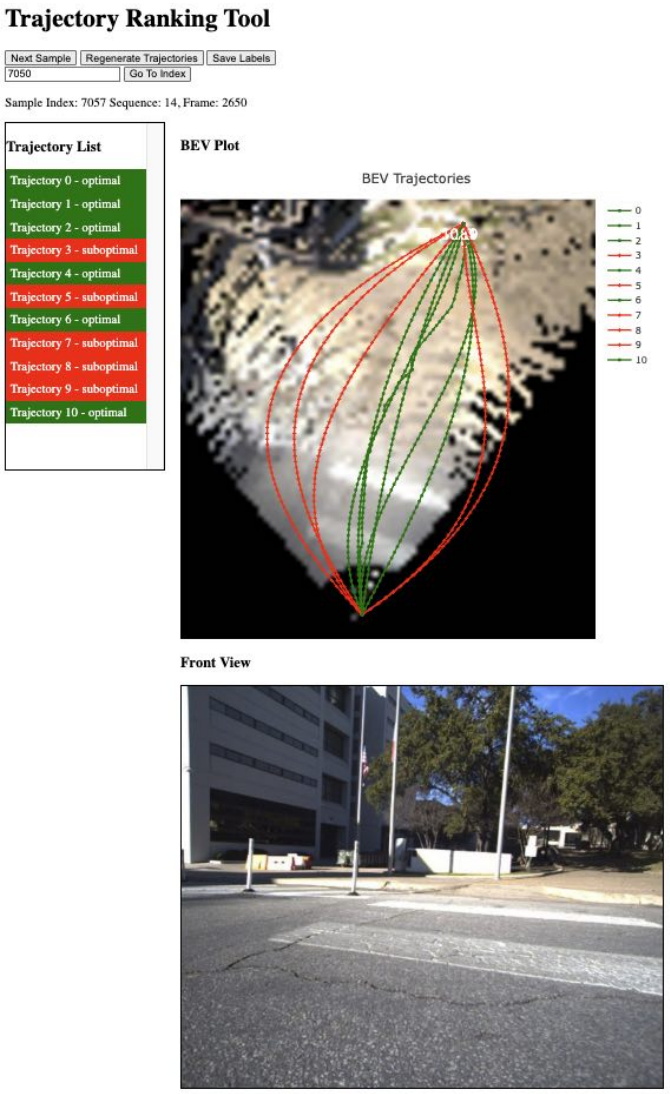

Fig. 8: Counterfactual annotation tool used for labelling counterfactuals for training the reward function r_ϕ. We provide the front view RGB image and BEV RGB image to the human annotator for context. In the image below, the trajectories annotated in red are counterfactuals and the trajectories in green are considered acceptable.

area illustrated in Fig. 4, we compare the planned path for each learned baseline in approximately the same location. Fig. 9 provides additional context to understand the inputs given to each baseline and the planned path, which we describe as follows: 1) CRESTE - The input RGB and sparse depth image (not shown) and predicted BEV cost map where darker regions correspond to low cost, 2) PACER+Geometric (PACER+G) [23] - The input BEV image and predicted BEV cost map where darker regions correspond to lower cost, 3) PIVOT [25] - The annotated satellite image with the robot's current location (blue circle), heading (blue arrow), and goal location (red circle). The front view RGB image annotated with numbered circles for prompting the VLM along with the chosen circle highlighted in green, 4) ViNT [40] - The front view image annotated with the local path waypoints predicted by ViNT.

Analyzing each model, we find that PACER+G struggles to correctly infer the cost for terrains that are underrepresented in the training dataset, such as dome mats. Furthermore, when two terrains look visually similar, such as the sidewalk and road pavement for the Trader joes location, PACER is unable to infer the costs correctly. PIVOT often selects the correct local waypoint in scenes with a large margin for error. However, it struggles with dealing with curb cuts (Hemphill Park, Sherlock North) and narrow sidewalks (Trader Joes). Furthermore, long latency between VLM queries negatively impacts the model's ability to compensate for noisy odometry. This effect is particularly apparent in narrow corridors or sidewalks where even minor deviation from the straight line path result in failure. ViNT is able to successfully maintain course in straight corridors and sidewalks, but often struggles to consider factors like curb cuts and terrains. We hypothesize this is because our dataset is not sufficiently large enough for learning generalizable features from expert demonstrations alone. Furthermore, demonstrations with straight line paths dominate our dataset, making it difficult for behavior cloning methods like ViNT to learn learn which factors influence the expert to diverge from the straight line path.

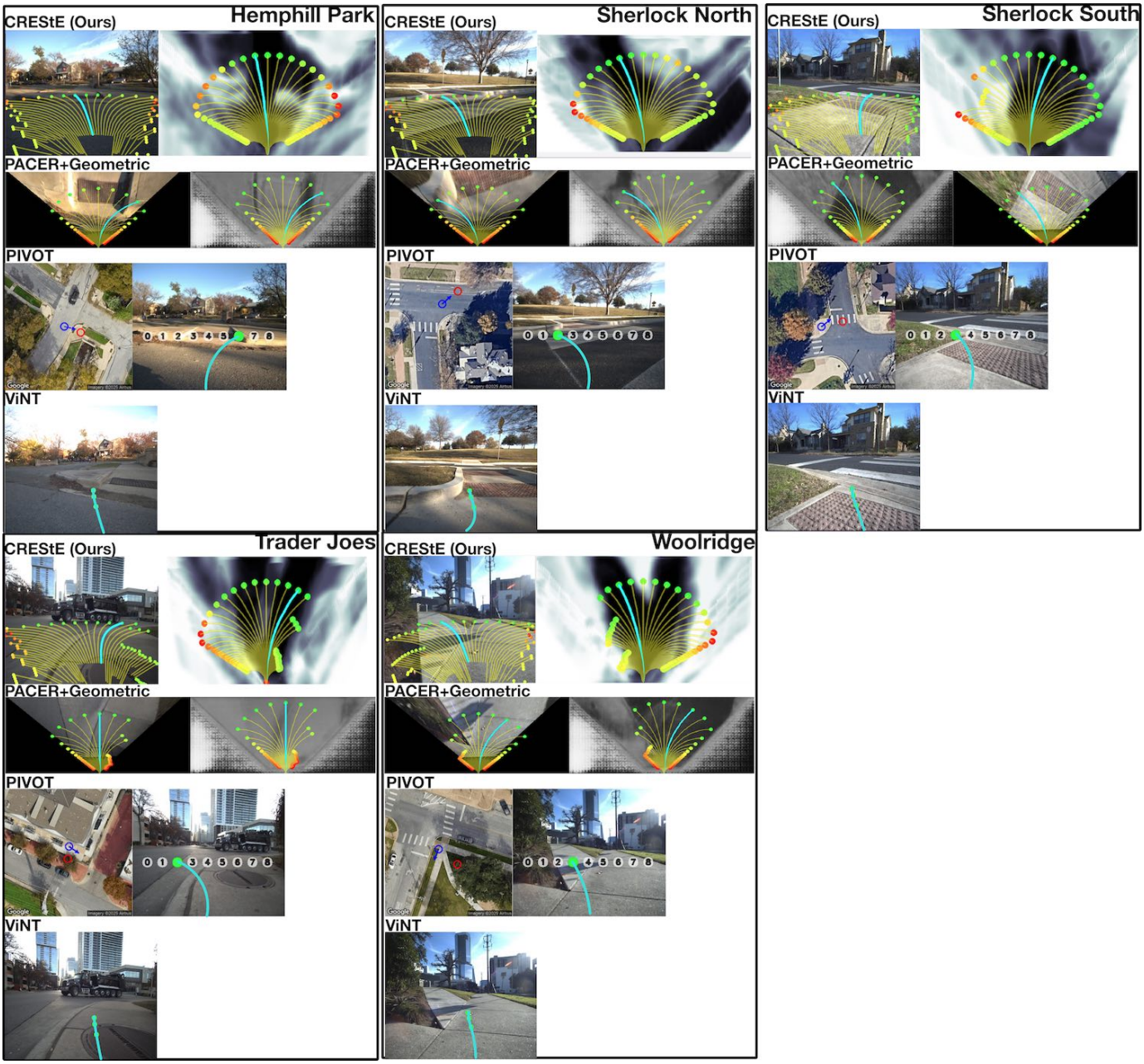

Fig. 9: Qualitative comparison of local paths planned by different learned baselines in different geoegraphic locations: Hemphill Park, Sherlock North, Sherlock South, Trader joes, Woolridge. We visualize each chosen path either in bird's eye view (BEV) or on the front view RGB image as solid green circles. For PACER+Geometric [23], we provide the BEV image used by the model. For PIVOT [25], we show the annotated satellite image and front view image used to prompt the VLM. For ViNT [40], we front view RGB image given as input. For more information regarding each baseline, we refer readers to Sec. IX-C.