arXiv:2503.03921v2 [cs.RO] 26 Jun 2025

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

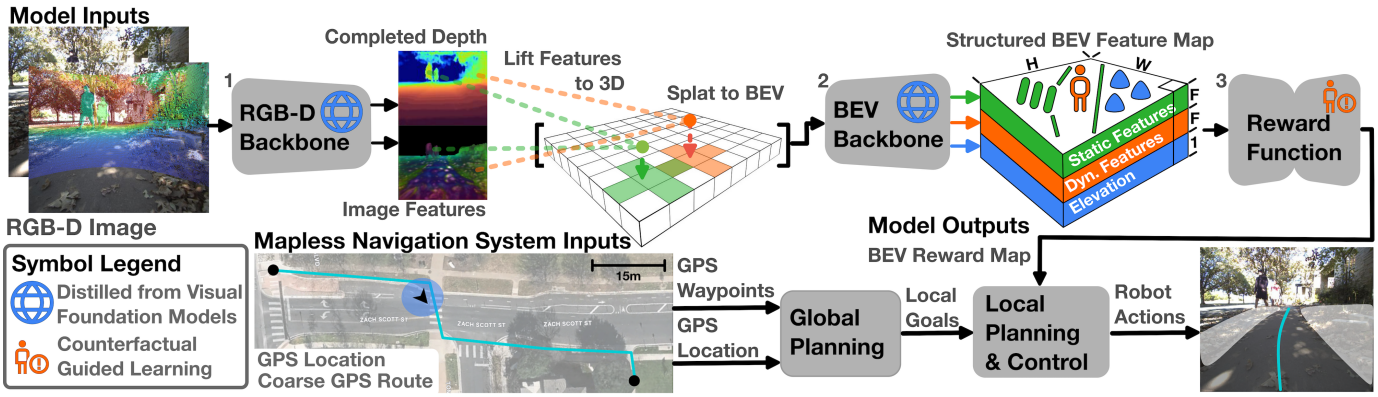

Fig. 1: CRESTE learns a perceptual encoder[1,2] and reward function[3] to predict structured bird's eye view (BEV) feature and reward maps for navigation. Our model inherits generalization and robustness from visual foundation models and learns expert-aligned rewards using our counterfactual-guided learning framework. We integrate CRESTE in a modular navigation system that uses coarse GPS guidance and rewards to reach navigation goals safely.

***Abstract*—We introduce CRESTE, a scalable learning-based mapless navigation framework to address the open-world generalization and robustness challenges of outdoor urban navigation. Key to achieving this is learning perceptual representations that generalize to open-set factors (e.g. novel semantic classes, terrains, dynamic entities) and inferring expert-aligned navigation costs from limited demonstrations. CRESTE addresses both these issues, introducing 1) a visual foundation model (VFM) distillation objective for learning open-set structured bird's-eye-view perceptual representations, and 2) counterfactual inverse reinforcement learning (IRL), a novel active learning formulation that uses counterfactual trajectory demonstrations to reason about the most important cues when inferring navigation costs. We evaluate CRESTE on the task of kilometer-scale mapless navigation in a variety of city, offroad, and residential environments and find that it outperforms all state-of-the-art approaches with 70% fewer human interventions, including a 2-kilometer mission in an unseen environment with just 1 intervention; showcasing its robustness and effectiveness for long-horizon mapless navigation. Videos and additional materials can be found on the project page: `https://amrl.cs.utexas.edu/creste`.**

I. Introduction

Mapless navigation is the task of reaching user-specified goals without high-definition (HD) maps and precise navigation waypoints. Mapless approaches plan routes using egocentric sensor observations (e.g. RGB images, point clouds, GPS), coarse waypoints from public routing services, and satellite imagery. These solutions demonstrate promise as a scalable alternative to conventional map-centric approaches: enabling generalization to unforeseen factors (e.g. curb ramps and foliage) and dynamic entities (e.g. pedestrians and strollers) while reducing map maintenance overhead and reliance on pre-defined routes.

Traditional geometric-only mapless solutions [3, 43] exhibit robust generalization when it is only necessary to consider costs for geometric factors like static obstacles. However, open-world navigation demands perception systems that perceive an open set of factors unknown apriori, ranging from terrain preferences (grass vs. concrete) to semantic cues (crosswalks and crossing signs). Furthermore, it requires the ability to identify the most salient features in the scene, and how they influence the choice of paths.

Learning-based approaches are a scalable alternative that considers factors beyond geometry, but must overcome data scarcity, robustness, and generalization challenges to achieve similar reliability. These approaches 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. While single-factor [20, 44, 43] and multi-factor [21, 33] methods learn representations that consider relevant navigation factors (e.g. geometry, terrain, semantics), they rely on a hand-curated list of semantic classes and terrains that limit generalization to unseen classes. End-to-end methods [34, 35, 36] alleviate this by jointly learning the representation and policy from expert demonstrations, but are prone to overfitting without large-scale robot datasets. While recent works [45, 22] demonstrate that pre-trained LLMs and VLMs can reason about expert-aligned behavior without large-scale robot datasets, we empirically demonstrate that they are poorly attuned to urban navigation, leading to brittle zero-shot transfer in complex scenes.

We address the aforementioned limitations with CRESTE,

Counterfactuals for Reward Enhancement with Structured Embeddings, a novel approach that learns open-set representations and navigation costs/rewards for mapless urban navigation. To learn open-set representations, CRESTE combines prior work on bird's eye view (BEV) representation learning [21] and visual foundation models (VFMs) [25, 30] to learn BEV map representations with inherited real-world robustness and open-set semantic knowledge. We unify priors from multiple foundation models by distilling image features from Dinov2 [25] and refining these features in BEV using SAM2 [30] instance labels. CRESTE infers expert-aligned navigation costs without large-scale demonstrations by leveraging counterfactuals to guide learning, where counterfactuals hold all other variables constant except for the path taken from the start to the end goal. Unlike conventional preference learning, our proposed counterfactual inverse reinforcement learning (IRL) objective explicitly minimizes rewards along paths that exhibit undesirable behavior (e.g. veering off crosswalks, driving off curb ramps, etc.), enabling operators to correct robot behavior by providing offline counterfactual feedback. We summarize the main contributions of our work as follows:

- **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 counterfactual IRL formulation for reward alignment using counterfactual and expert demonstrations.

We demonstrate our approach's effectiveness through real-world kilometer-scale navigation experiments in urban environments with off-road terrain, elevated walkways, sidewalks, intersections, and other challenging conditions. In total, we evaluate in **6** distinct seen and unseen geographic areas and significantly outperform 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. Underlying these methods are two key challenges: learning robust perceptual representations that encode an open set of navigation factors, and learning planning modules that can identify and reason about the most important factors to plan safe paths.

Single [14, 20, 39] and multi-factor [9, 21, 33] approaches learn perceptual representations grounded in the planning horizon with elevation, terrain, and semantic factors, and rely on expert tuning to balance the relative costs of each factor. While this enables joint reasoning about multiple factors, it assumes the full set of semantic classes and terrains are known apriori, generalizing poorly to unexpected semantic classes, terrains, or other factors not seen during training. Furthermore, they require expert tuning of complex multi-objective cost functions for each environment.

End-to-end methods jointly learn a representation and policy using either behavior cloning (BC), RL, or IRL. BC methods [15, 35] learn task-specific representations for reasoning about how to mimic the expert policy. These approaches scale effectively given large-scale datasets with expert demonstrations, but are prone to overfitting otherwise. RL [11] and IRL [47] methods also mimic the expert policy, but design handcrafted reward functions (e.g. minimize vibrations [12] and likelihood of interventions [13]) or preference rankings [40, 7] to guide representation and policy learning. These methods efficiently learn expert-aligned behaviors at the cost of careful reward tuning or preference enumeration.

An emerging class of methods [22, 45] leverages 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 steer VLMs/LLMs to reason about the most important navigation factors for navigation tasks.

The aforementioned works either rely on large-scale robot datasets for open-set generalization or complex reward design to learn expert-aligned behavior. However, constructing large robot datasets and expressive multi-term objectives is prohibitively difficult for open-world settings. In contrast, CRESTE learns open-set representations by directly distilling knowledge from VFMs, and expert-aligned reward functions by using counterfactuals as a unified language for conveying navigation constraints. Unlike counterfactual-based methods like VRL-PAP [39], our counterfactuals specify arbitrary navigation constraints beyond terrains and can be readily obtained offline, enabling a data flywheel for continuous improvement with additional deployments. Importantly, our reward learning approach is distinct from preference-learning methods, as counterfactuals are a more specific form of feedback than preferences, which can be between any two trajectories.

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:

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

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. 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.

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 $\Theta : \mathcal{O} \rightarrow \mathcal{T}$ that maps observations o_t to a joint embedding space $\mathcal{T}$ 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 $T : \mathcal{X} \rightarrow \mathcal{T}$ 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 function that combines individual cost terms.

We assume the operator has an underlying true cost function $H : \mathcal{T} \rightarrow \mathbb{R}^{0+}$ 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

CRESTE is a modular approach with two key components that can be trained end-to-end: 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 $y_{\text{bev, t}}$ and outputs a BEV scalar reward map $y_{\text{reward, t}}$. In the remainder of this section, we describe the following: 1) Sec. IV-A - The CRESTE model architecture and 2) Sec. IV-B - The CRESTE training procedure, where Sec. IV-B1 presents our VFM distillation objective for Θ and Sec. IV-B2 presents counterfactual IRL and our active reward learning framework for learning r_ϕ.

A. CRESTE *Model Architecture*

1) Perceptual Encoder Model Architecture: Our 25.5M parameter perceptual encoder Θ draws inspiration from the TerrainNet [21] backbone, which trains a RGB-D encoder f_{rgbd} to predict a latent feature map z_{rgbd} using an EfficientNet-B0 [42] encoder and a completed depth map y_{depth} using a depth completion head $f_{\text{depth}}(z_{\text{rgbd}})$. Like TerrainNet, we train a lift-splat module $f_{\text{splat}}(z_{\text{rgbd}}, y_{\text{depth}})$ to lift latent features to 3D and "splat" them to an unstructured BEV feature map $z_{\text{bev, splat}}$. Finally, we pass $z_{\text{bev, splat}}$ to a BEV inpainting backbone f_{bev} that uses a shared U-Net [32] encoder and separate decoders to predict a structured feature map consisting of separate semantic and elevation layers.

Building on TerrainNet, we make two key architectural modifications. Our first modification, semantic decoder f_{semantic}, promotes learning semantic and geometric-aware features by regressing image features from Dinov2 [25] using latent image features z_{rgbd}. This design is analogous to model distillation [29] and allows our RGB-D encoder f_{rgbd} to inherit properties from VFMs like robustness to perceptual aliasing and open-set semantic understanding.

Our second modification stems from the observation that Dinov2 features alone lack the entity understanding needed to backproject and inpaint features in heavily occluded urban scenes with noisy depth predictions. Thus, we supplement f_{bev} with two panoptic map decoders $f_{\text{bev,static}}$ and $f_{\text{bev,dynamic}}$ to ensure that predicted BEV feature maps y_{bev} are consistent with BEV instance maps from SegmentAnythingv2 (SAM2) [30]. We use Supervised Contrastive Loss [16] to optimize y_{bev} such that features belonging to the same instance are closer in embedding space than features belonging to different instances. Combined with f_{semantic}, our modifications synergistically unify the strengths of Dinov2 and SAM2 to learn open-set semantic, geometric, and instance-aware representations grounded in the local planning horizon. Altogether, our structured BEV feature map y_{bev} consists of three layers stacked along the channel dimension: 1) Static panoptic feature map $y_{\text{bev, static}}$, 2) Dynamic panoptic feature map $y_{\text{bev, dynamic}}$, and 3) Elevation map $y_{\text{bev, elev}}$.

2) Reward Function Model Architecture: To ensure our reward function enforces spatial invariance and considers multi-scale features, we implement r_ϕ using a 0.5M parameter Multi-Scale Fully Convolutional Network (MS FCN) first used by Wulfmeier et al. [47]. We supervise r_ϕ using our counterfactual IRL loss, which we describe along with our training procedure for Θ in the next section.

B. CRESTE *Training Procedure*

We train CRESTE by first optimizing Θ and freezing the parameters before training r_ϕ using our counterfactual-based

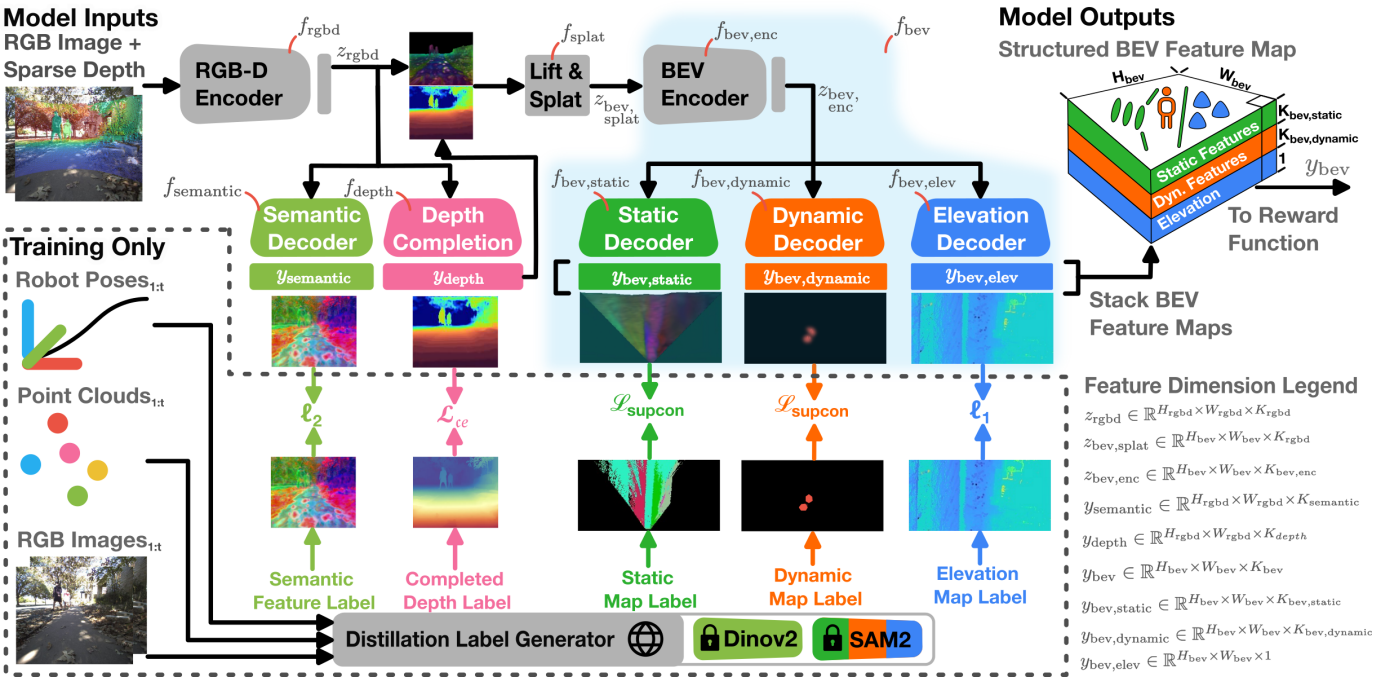

Fig. 2: Model architecture and training procedure for the CRESTE perceptual encoder Θ. Using a RGB and sparse depth image, f_{rgbd} extracts image features z_{rgbd} and performs depth completion. Next, we lift and splat z_{rgbd} to an unstructured BEV feature map before predicting continuous static panoptic, dynamic panoptic, and elevation features. Finally, we stack the predicted features to construct a structured BEV feature map for our learned reward function. We supervise Θ using semantic feature maps, completed depth, and BEV map labels generated by our SegmentAnythingv2 [17] and Dinov2 [25]-powered distillation label generator. We define the dimensions for important feature maps in the Feature Dimension Legend on the bottom right.

active reward learning framework. 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 our training and label generation procedure for supervising Θ and Sec. IV-B2 defines our counterfactual IRL objective $\mathcal{L}_{\text{IRL}}$ and active reward learning framework for training r_ϕ.

1) Training the Perceptual Encoder Θ: We jointly train f_{rgbd} via backpropagation from f_{semantic} and f_{depth}. The semantic decoder head f_{semantic} is supervised via an MSE loss that minimizes the error between the predicted and ground truth feature maps y_{semantic} and $\hat{y}_{\text{semantic}}$. The depth completion head f_{depth} is supervised using a cross-entropy classification loss $\mathcal{L}_{\mathcal{CE}}$, where the ground truth depth $\hat{y}_{\text{depth}}$ is uniformly discretized into bins. Empirically, this captures depth discontinuities better than regression [31]. Altogether, the training objective for the RGB-D backbone is:

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

where α_1 and α_2 are tunable hyperparameters.

We supervise f_{bev} with three losses, one for each BEV decoder head, using ground truth static instance maps $\hat{y}_{\text{bev, static}}$, dynamic instance maps $\hat{y}_{\text{bev, dynamic}}$, and elevation maps $\hat{y}_{\text{bev, elev}}$. We use Supervised Contrastive Loss [16] ($\mathcal{L}_{\text{contrastive}}$) for training $f_{\text{bev, static}}$ and $f_{\text{bev, dynamic}}$ to predict continuous feature maps from discrete instance labels $\hat{y}_{\text{bev, static}}$ and $\hat{y}_{\text{bev, dynamic}}$. 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:

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

where β_1, β_2, and β_3 are tunable hyperparameters. We refer readers to Appendix Sec. X-B for additional training details.

To scalably obtain training labels for Θ, we design a distillation label generation module that leverages VFMs to automatically generate training labels. As shown in Fig. 2, our label generator only requires sequential SE(3) robot poses and synchronized RGB–point cloud pairs ($o_{\text{rgb, 1:t}}$, $o_{\text{cloud, 1:t}}$) to generate the training labels described next.

i) Label Generation for RGB-D Encoder f_{semantic}. We generate training labels for the semantic decoder f_{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 depth completion labels $\hat{y}_{\text{depth}}$ by projecting $o_{\text{cloud, t}}$ to the image and applying bilateral filtering [28] for edge-aware inpainting, producing a dense depth map that respects object boundaries.

ii) Label Generation for BEV Inpainting Backbone f_{bev}. To train f_{bev}, we first generate $\hat{y}_{\text{bev, dynamic}}$ by prompting SAM2 with bounding boxes of common dynamic classes (e.g., vehicles, pedestrians), backproject the masks to 3D using $o_{\text{cloud, t}}$, and apply DBSCAN [6] clustering at multiple

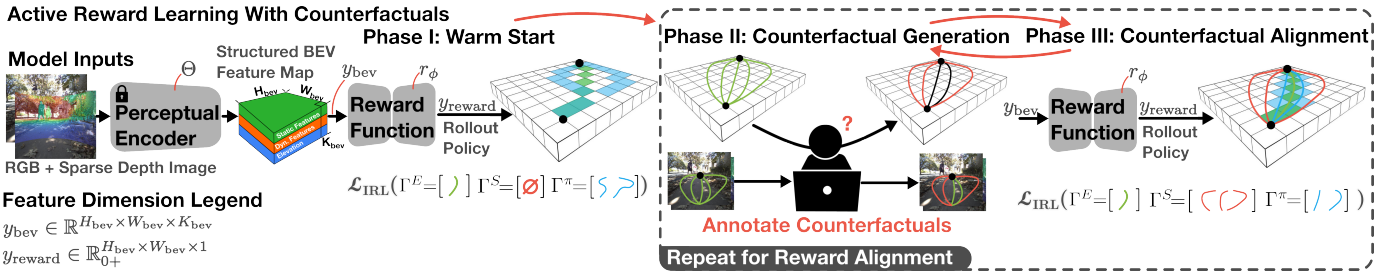

Fig. 3: Framework for Active Reward Learning with Counterfactuals. Before reward learning, we train and freeze the perceptual encoder Θ and use the output BEV feature map y_{bev} with r_ϕ to predict BEV reward maps y_{reward}. In phase I, we train r_ϕ using our counterfactual IRL objective $\mathcal{L}_{\text{IRL}}$ 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.

density thresholds. The dynamic BEV map retains clusters with sufficient IoU overlap with the SAM2 instance labels. This procedure mirrors prior works [26] that leverage SAM2 for learning instance-aware 3D representations. To generate $\hat{y}_{\text{bev, static}}$, we query SAM2 with a grid of points, filter out overlaps with dynamic masks to isolate static segments (generating dynamic masks as before), and apply a greedy IoU-based merging strategy across frames to maintain instance consistency across frames. The merged static masks are then projected and accumulated in BEV using known robot poses. We refer readers to the Appendix Sec. X-A for our algorithmic formulation describing the merging procedure. Lastly, we generate $\hat{y}_{\text{bev, elev}}$ by accumulating static 3D points across sequential frames using robot poses (generating static masks 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.

2) Training the Reward Function r_ϕ: After freezing Θ, we train r_ϕ using our counterfactual-based reward learning objective $\mathcal{L}_{\text{IRL}}$, where counterfactuals hold all other variables constant with the expert except for the path taken to the end goal. $\mathcal{L}_{\text{IRL}}$ optimizes the reward function such that the likelihood of counterfactual trajectories is minimized and the likelihood of expert trajectories is maximized, which we observe improves the sample efficiency, expert alignment, and interpretability of the learned rewards.

To understand $\mathcal{L}_{\text{IRL}}$, we formally refer to counterfactuals as suboptimal trajectories and 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. Under this formulation, the Counterfactual IRL objective 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:

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

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:

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

Assuming non-trivial rewards, which can be enforced using gradient regularization techniques [19], 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. 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. Our reward learning framework uses Counterfactual IRL to learn a mapping from BEV features in y_{bev} to their scalar utilities. Our framework trains r_ϕ in multiple phases, iteratively prompting expert operators for counterfactual annotations in underperforming scenarios until the learned rewards are sufficient for planning trajectories similar to expert demonstrations. Fig. 3 illustrates this framework, which we describe next:

Phase I: Warmstart We obtain a base reward function by training r_ϕ using only expert demonstrations. This can be done simply by setting α equal to zero in $\mathcal{L}_{\text{IRL}}$.

Phase II: Synthetic Counterfactual Generation We rollout a policy using the learned rewards from phase I and select training samples needing refinement using the Hausdorff distance between the policy rollouts and expert demonstrations. For samples exceeding a distance threshold, we accumulate RGB observations to BEV and generate candidate trajectories

sharing start/end points with the expert. A human annotator using our counterfactual annotation tool in Fig. 8 to select trajectories that violate their preferences (e.g., collisions, undesirable terrain) as counterfactuals, which are used in phase III to retrain the reward function using $\mathcal{L}_{\text{IRL}}$.

Phase III: Counterfactual Reward Alignment We retrain r_ϕ using $\mathcal{L}_{\text{IRL}}$ using phase II's counterfactual annotations and the original expert demonstrations. We set α to be nonzero to balance their relative importance. We repeat phases II and III until the learned policy converges with expert behavior.

Counterfactual Generation Details. Counterfactual IRL relies on being able to sample counterfactual trajectories that visit a diverse set of states between the start and goal. To do this, we propose a simple method of perturbing the expert trajectory to obtain these counterfactuals. Given the expert trajectory Γ_t^E, we sample a handful of "control" states at regular intervals along Γ_t^E, excluding the start and goal states. We perturb each control state according to a non-zero mean Gaussian distribution centered at each control state $\mathcal{N}(\mu+s, \sigma)$ before planning a kinematically feasible path that reaches the start, goal, and perturbed control states. Practically, we implement this using Hybrid A* [5] - however, any kinematic planner will suffice. We repeat this process twice using positive and negative μ terms to sample paths (~ 10) on both sides of the expert before passing these paths to the operator for counterfactual labeling. For additional implementation details regarding generating alternate trajectories, we refer readers to Appendix Sec. X-C.

Counterfactual IRL Derivation $\mathcal{L}_{\text{IRL}}$. IRL methods [50, 23] 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 relabeled with suboptimal or 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 [40] 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, this is:

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

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:

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

and the policy player maximizes expected return:

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

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:

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

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. X.

1) Global and Local Planning and Controls: Our global planning module plans a global path and identifies the next local subgoal for our local planner to plan local paths. Given a user-specified GPS end goal G_N, we use OpenStreetMap [24] 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 [8] 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 a scalar 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 [27] 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 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.

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?
- (Q_3) How much do counterfactual demonstrations improve urban navigation performance?
- (Q_4) How well does CREStE perform long horizon mapless urban navigation compared to other top state-of-the-art approaches?

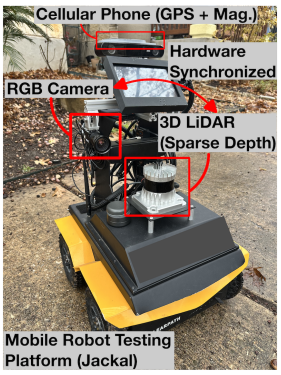

Fig. 6: Mobile robot testing platform for real-world experiments. We annotate the locations of the monocular camera, 3D LiDAR, and cellular phone (not visible) on our testing platform, the Clearpath Jackal.

We investigate $\mathcal{Q}_1$ by comparing CREStE's performance against other methods in unseen environments. Additionally, we evaluate the relative performance between different input modalities and observation encoders for CREStE. 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 [24] 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. Additionally, we annotate 3% of our training dataset with negative counterfactual annotations to train CREStE. Our dataset consists of synchronized image LiDAR observation pairs and ground truth robot poses computed using LeGO-LOAM [38], 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 urban environments where we perform short horizon ($\sim$100m) quantitative experiments. These consist of six challenging seen and unseen environments with trails, road crossings, narrow pathways, different terrains, and obstacle hazards. For locations 1-5, we repeat the same experiment twice for each approach. For location 6, we repeat the same experiment five times for each approach. We evaluate the

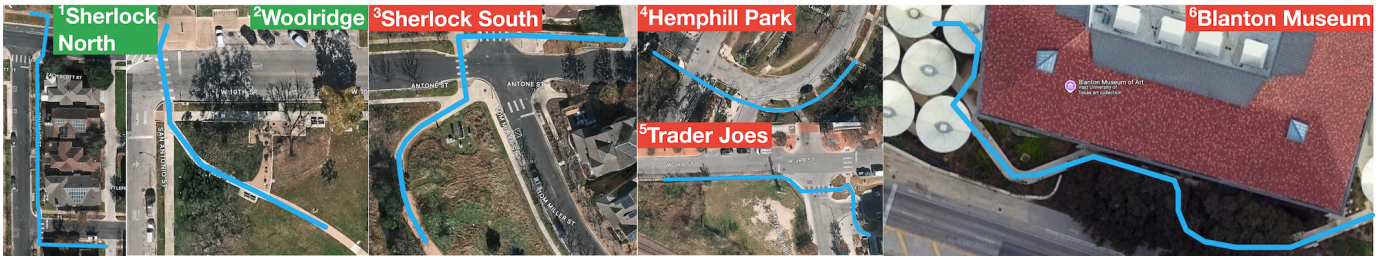

Fig. 4: Satellite image of testing locations for short horizon mapless navigation experiments. We evaluate baselines across 2 seen (green) and 4 unseen (red) urban locations. Our testing locations consist of residential neighborhoods, urban shopping centers, urban parks, and offroad trails. In each location, each baseline must start from an endpoint on the annotated blue trajectory and navigate to the opposite end of the trajectory. We denote each location's ID with a numerical superscript.

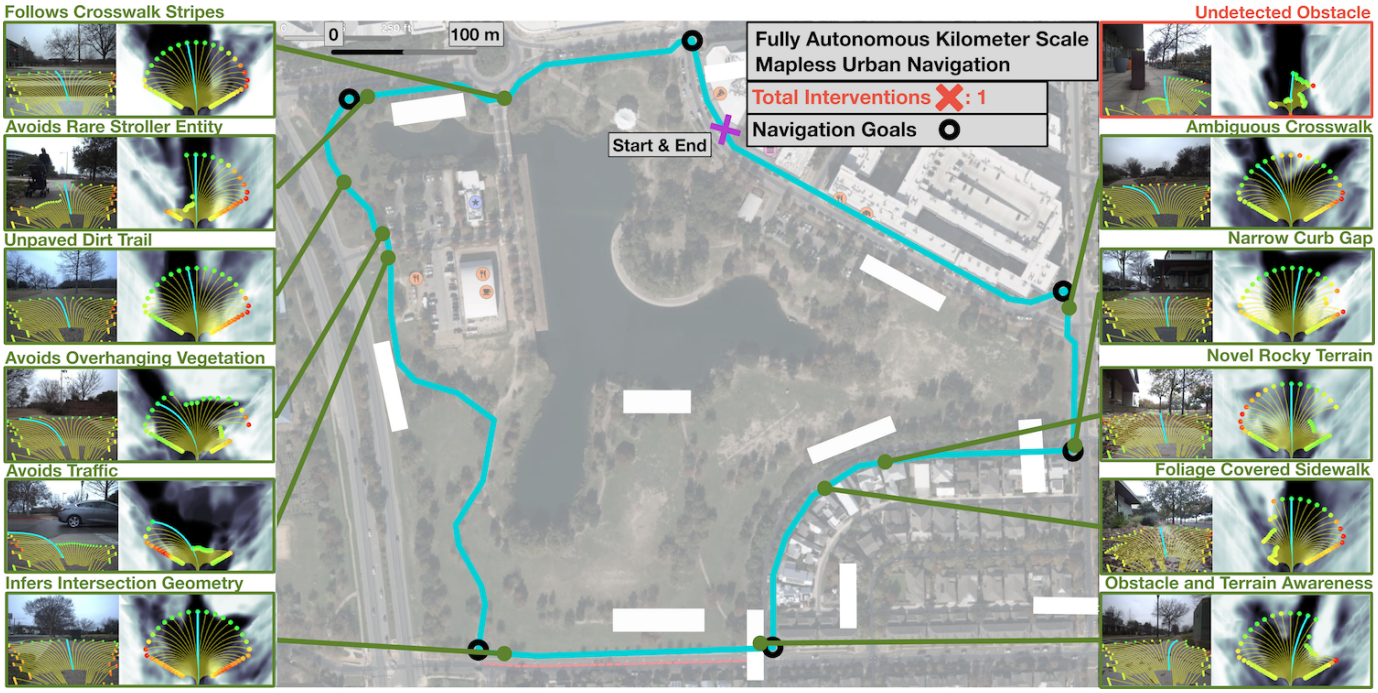

Fig. 5: Satellite image of our 2 kilometer long-horizon testing area, with examples with front view RGB image observations and CRESTE's predicted BEV costmap (converted from the predicted BEV reward map). We annotate successful examples in green with a brief description of the situation. We annotate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

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 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 were 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 [22], perform all computations using onboard compute to fairly evaluate real-time performance. Next, we describe our evaluation baselines.

For questions 1 and 3, we compare CRESTE against four existing navigation baselines and six variations of the

	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 [20]	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 [35]	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 [22]	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 - 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 - 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 - st	19.09	96.29	6.10	17.80	92.90	3.03	12.88	94.65	2.04	22.01	92.85	7.83	16.32	93.42	2.16
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 comparing CRESTE against existing navigation baselines 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.

	Out of Distribution		
Location	Blanton Museum		
Method	AST ↓	%S↑	NIR ↓
CRESTE-RGB-FROZEN	17.14±2.51	97.91±2.59	1.69±0.96
CRESTE-RGB	17.25±3.83	97.38±2.43	1.38±0.90
CRESTE-STEREO	17.72±1.32	97.43±1.88	1.07±0.49
CRESTE (ours)	**13.81±2.38**	**98.36±1.55**	**0.76±0.48**

TABLE II: Quantitative evaluation comparing the impact of different input modalities and observation encoders on CRESTE. All baselines are evaluated on the same experiment area. We compute the mean and standard deviation over 5 trials and bold the best performing method for each metric in each location. We annotate each metric with an up or down arrow to indicate if higher or lower numbers are better.

CRESTE architecture. We first describe the existing baselines: 1) *ViNT* [35] - a foundation navigation model trained on goal-guided navigation 2) *PACER+Geometric* [20] (PACER+G) - a multi-factor perception baseline that considers learned terrain and geometric costs in a dynamic window [8] (DWA) style approach 3) *Geometric-Only* - a DWA style approach that only considers geometric costs 4) *PIVOT* [22] - a VLM-based navigation method that uses the latest version of GPT4o-mini [2] to select local goals from image observations.

We test six variations of CRESTE to evaluate the effect of different observation encoders and sensor modalities. These consist of 1) CRESTE-RGB, the same RGB encoder but using only monocular RGB images, 2) CRESTE-STEREO, the same RGB encoder with a stereo processing backbone that fuses features from stereo RGB images, 3) CRESTE-RGB-Frozen, the same depth prediction head but with a frozen Dinov2 [25] encoder that only uses monocular RGB images. The remaining variations ablate our methodological contributions: 4) CRESTE-cfs, our model trained without counterfactual demonstrations, but otherwise identical to the original, 5) CRESTE-st, our model trained without the BEV inpainting backbone, but otherwise identical to the original, 6) CRESTE-cfs-st, our model trained without counterfactual demonstrations or the BEV inpainting backbone. For architectures 5 and 6, we directly pass the unstructured BEV feature map $z_{\text{bev, splat}}$ 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. For additional details regarding the CRESTE architecture variations, we refer readers to Appendix Sec. X-D2. Next, we will describe high-level implementation details for each non-CRESTE baseline.

Location	Mueller Loop				
Method	AST ↓	%S↑	NIR ↓	Dist. (m) ↑	Total Int. ↓
PACER+G [20]	15.8	61.53	3.56	1345.07	48
CRESTE (ours)	**12.94**	**99.45**	**0.052**	**1919.44**	**1**

TABLE III: 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.

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, Table II, and Table III respectively, and analyze these results to answer $\mathcal{Q}_1$ - $\mathcal{Q}_4$ 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 Ap-

pendix Sec. X-D. Quantitatively, PACER+G, the next best approach, requires two times more interventions than CRESTE in seen environments and five times more interventions in unseen environments. Furthermore, we find that PIVOT, our VLM-based navigation baseline, performs poorly relative to other methods, corroborating our claim that VLMs and LLMs are not well 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. From these findings, we conclude that CRESTE is remarkably more generalizable for mapless urban navigation than existing approaches.

In Table II, our approach performs most favorably using RGB and LiDAR inputs, followed by stereo RGB and monocular RGB respectively. This occurs because LiDAR offers robustness to severe lighting artifacts like lens flares that affect costmap prediction quality. We present qualitative examples of this in Fig. 10. Interestingly, we find that distilled Dinov2 features perform favorably compared to frozen Dinov2 features. Prior works corroborate this observation [48, 49] and verify that Dinov2 features contain positional embedding artifacts that disrupt multi-view consistency. We postulate that our distillation objective enables our RGB-D encoder f_{rgbd} to focus on learning artifact-free features while the semantic decoder f_{semantic} learns how to reconstruct these artifacts. For additional analysis comparing these architectural variations, we refer readers to Appendix Sec. X-D1.

2) Evaluating the Importance of Structured BEV Perceptual Representations.: We assess the impact of structured perceptual representations by evaluating CRESTE-cfs-st and CRESTE-st against CRESTE in Table I, where the only difference is whether structured representations and counterfactuals are used. Without structured representations, CRESTE-st incurs 28% more interventions on average across all environments. Performance further deteriorates without structured representations or counterfactuals, incurring 41% more interventions on average across all environments. We hypothesize that 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,

4) Evaluating Performance on Kilometer-Scale Mapless Navigation.: Table III 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

While CRESTE inherits some viewpoint-invariance from VFMs, it does not allow reward maps to be conditioned on embodiment-specific constraints, such as the traversability differences between quadrupeds and wheeled robots. An interesting future direction is applying existing research on cross-embodiment conditioning [35] to our architecture. Furthermore, CRESTE infers reward maps using observations from a single timestep, limiting multi-step horizon reasoning tasks, such as recovering from dead ends or negotiating paths in constricted environments. Extending CRESTE with memory and counterfactual IRL for reasoning about object dynamics are promising directions for addressing these issues. In addition, reward learning with counterfactuals is a potential bottleneck, as it requires humans for counterfactual labeling. Automated counterfactual generation with VLMs/LLMs is another promising direction to further improve scalability. Lastly, our counterfactual IRL objective can further 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.

VIII. Conclusion

In this paper, we introduce Counterfactuals for Reward Enhancement with Structured Embeddings (CRESTE), a scalable framework for learning representations and policies that address the open-set generalization and robustness challenges of open-world mapless navigation. 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.

IX. Acknowledgements

This work has taken place in the Autonomous Mobile Robotics Laboratory (AMRL) and Machine Decision-making through Interaction Laboratory (MIDI) at UT Austin. AMRL research is supported in part by NSF (CAREER-2046955, PARTNER-2402650) and ARO (W911NF-24-2-0025). MIDI research is supported in part by NSF (CAREER-2340651, PARTNER-2402650), DARPA (HR00112490431), and ARO (W911NF-24-1-0193). Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

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] 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.

[6] 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.

[7] Bıyık et al. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *The International Journal of Robotics Research*, 41(1):45–67, 2022.

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

[9] 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.

[10] 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.

[11] Noriaki Hirose, Catherine Glossop, Ajay Sridhar, Oier Mees, and Sergey Levine. Lelan: Learning a language-conditioned navigation policy from in-the-wild video. In *8th Annual Conference on Robot Learning*.

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

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

[19] 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.

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

[21] Xiangyun Meng, Nathan Hatch, Alexander Lambert, Anqi Li, Nolan Wagener, Matthew Schmittle, Joonho Lee, Wentao Yuan, Zoey Qiuyu Chen, Samuel Deng, Greg Okopal, Dieter Fox, Byron Boots, and Amirreza Shaban. Terrainnet: Visual modeling of complex terrain for high-speed, off-road navigation. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu, editors, *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi: 10.15607/RSS.2023.XIX.103. URL https://doi.org/10.15607/RSS.2023.XIX.103.

[22] 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. In *International Conference on Machine Learning*, pages 37321–37341. PMLR, 2024.

[23] 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.

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

[25] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research Journal*, pages 1–31, 2024.

[26] 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.

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

[28] 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.

[29] Mike Ranzinger, Greg Heinrich, Jan Kautz, and Pavlo Molchanov. Am-radio: Agglomerative vision foundation model reduce all domains into one. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12490–12500, 2024.

[30] 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.

[31] 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.

[32] 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.

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

[34] Dhruv Shah and Sergey Levine. Viking: Vision-based kilometer-scale navigation with geographic hints. In Kris Hauser, Dylan A. Shell, and Shoudong Huang, editors, *Robotics: Science and Systems XVIII, New York City, NY, USA, June 27 - July 1, 2022*, 2022. doi: 10.15607/RSS.2022.XVIII.019. URL https://doi.org/10.15607/RSS.2022.XVIII.019.

[35] Dhruv Shah, Ajay Sridhar, Nitish Dashora, Kyle Stachowicz, Kevin Black, Noriaki Hirose, and Sergey Levine. Vint: A foundation model for visual navigation. In *7th Annual Conference on Robot Learning*.

[36] 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.

[37] Faranak Shamsafar, Samuel Woerz, Rafia Rahim, and Andreas Zell. Mobilestereonet: Towards lightweight deep networks for stereo matching. In *Proceedings of the ieee/cvf winter conference on applications of computer vision*, pages 2417–2426, 2022.

[38] 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.

[39] Kavan Singh Sikand, Sadegh Rabiee, Adam Uccello, Xuesu Xiao, Garrett Warnell, and Joydeep Biswas. Visual representation learning for preference-aware path planning. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 11303–11309. IEEE, 2022.

[40] Harshit Sikchi, Akanksha Saran, Wonjoon Goo, and Scott Niekum. A ranking game for imitation learning. *Transactions on Machine Learning Research*.

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

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

[43] 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.

[44] 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.

[45] 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. *CoRR*, 2024.

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

[47] 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.

[48] Jiawei Yang, Boris Ivanovic, Or Litany, Xinshuo Weng, Seung Wook Kim, Boyi Li, Tong Che, Danfei Xu, Sanja Fidler, Marco Pavone, et al. Emernerf: Emergent spatial-temporal scene decomposition via self-supervision. In *The Twelfth International Conference on Learning Representations*.

[49] Jiawei Yang, Katie Z Luo, Jiefeng Li, Congyue Deng, Leonidas Guibas, Dilip Krishnan, Kilian Q Weinberger, Yonglong Tian, and Yue Wang. Denoising vision transformers. In *European Conference on Computer Vision*, pages 453–469. Springer, 2024.

[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.

X. Appendix

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

A. Perceptual Encoder Label Generation Details

In this section, we provide additional details regarding generating BEV instance maps for training the perceptual encoder Θ. We observe that the default SAM2 model settings perform well for generating instance labels without significant hyperparameter tuning. When merging sequential SAM2 instance labels to generate static BEV instance maps, it is important to ensure that instance IDs remain temporally consistent when merging cross-view instances. We ensure this by greedily merging BEV instances between frames with the highest intersection over union (IoU). Instances that do not exceed an IoU of 0.2 are treated as unique instances and are merged into the aggregated BEV instance map as such. We repeat this procedure from the first to the last observation in the horizon.

When generating dynamic BEV instance maps, we back-project the image instance labels to 3D and cluster the point cloud using DBSCAN with three density thresholds (0.1, 0.2, 0.3) and require a minimum of (5, 3 5) points in each cluster, respectively. We observe that these thresholds work well across environments for identifying clusters within a 12.8m sensing range. To match SAM2 instance labels to cluster IDs, we use an IoU threshold of 0.2 to determine if the label and cluster IDs should be matched. All unmatched clusters are discarded and the matched clusters are used to construct the dynamic BEV instance map.

B. 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 IV and Table V, 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 [16] 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 III.

Fig. 7: Qualitative Example of context images provided to PIVOT [22], 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.

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 [19] $\alpha_{reg} = 1.0$. 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 [35]: 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 [22]: 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:

> 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 [20]: 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.

C. 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.

TABLE IV: Architecture Hyperparameters for CRESTE.

Hyperparameter	Value
RGB-D Encoder (f_{rgbd})	
Base Architecture	EfficientNet-B0 [42]
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 [10]
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 [41]
Base Reward Architecture	MultiScale-FCN [46]
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

In total, we generate 10 alternate trajectories for each training sample. On average, we find that providing just 1-5 negative counterfactual annotations for 3% of the training dataset 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. We use a fixed set of criteria for annotating counterfactuals, including but not limited to: driving on undesirable terrains, irrecoverable failures like driving off curbs, and unsafe behaviors like colliding into objects and veering off crosswalks. Using our tool, a single human annotator is able to annotate the entire training dataset in approximately 2 hours.

TABLE V: 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 [18]
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
GPU Training Hours	10 Hours
GPUS Used	3 Nvidia H100 GPUs
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 [18]
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
GPU Training Hours	24 Hours
GPUS Used	3 Nvidia H100 GPUs
Reward Function Training (r_ϕ)	
Batch Size	30
Optimizer	AdamW [18]
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	4.0
GPU Training Hours	2 Hours
GPUS Used	3 Nvidia H100 GPUs

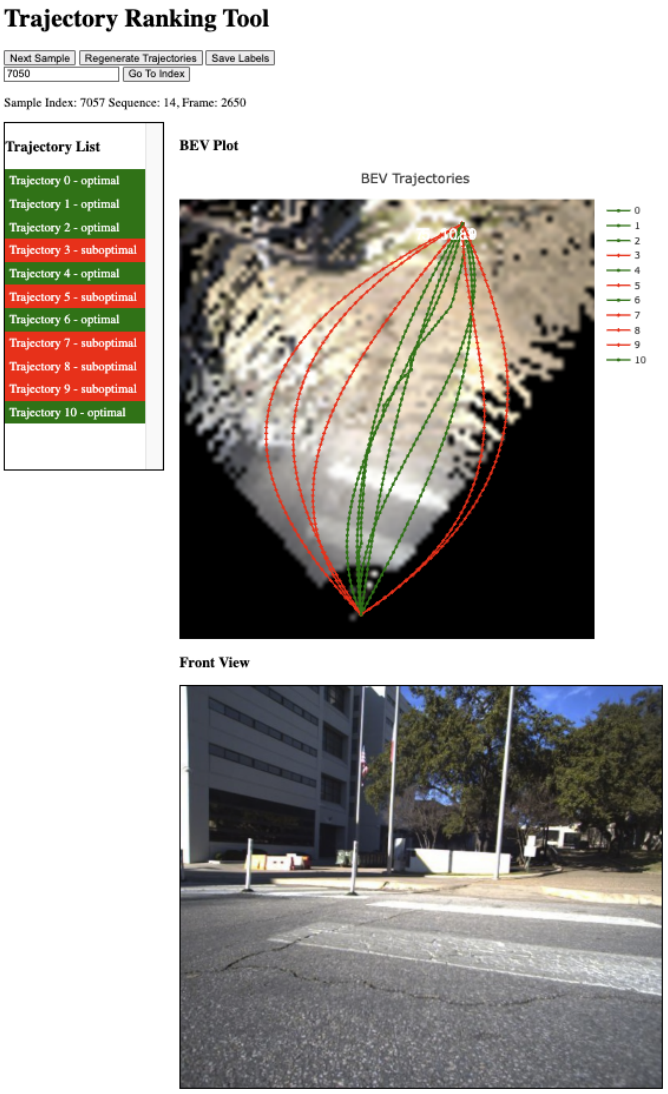

Fig. 8: Counterfactual annotation tool used for labelling counterfactuals for training the reward function r_ϕ. We provide the front view RGB image and BEV RGB image to the human annotator for context. In the image below, the trajectories annotated in red are counterfactuals and the trajectories in green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

D. Qualitative Short Horizon Experiment Analysis

In this section, we present a qualitative analysis of the short horizon experiments for each location. For each experiment area illustrated in Fig. 4, we compare the planned path for each learned baseline in approximately the same location. We will first compare CRESTE against existing navigation baselines in Sec. X-D1 before comparing different variations of the CRESTE architecture in Sec. X-D2.

1) Comparing CRESTE *Against Existing Baselines:* 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) [20] - The input BEV image and predicted BEV cost map where darker regions correspond to lower cost, 3) PIVOT [22] - 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 [35] - The front view image annotated with the local path waypoints predicted by ViNT.

Analyzing each model, we find that PACER+G struggles to 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 infers the costs incorrectly. PIVOT can select the correct local waypoint in scenes with a large margin for error, but struggles at dealing with curb cuts (Hemphill Park, Sherlock North) and narrow sidewalks (Trader Joes). 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 results in failure. ViNT can suc-

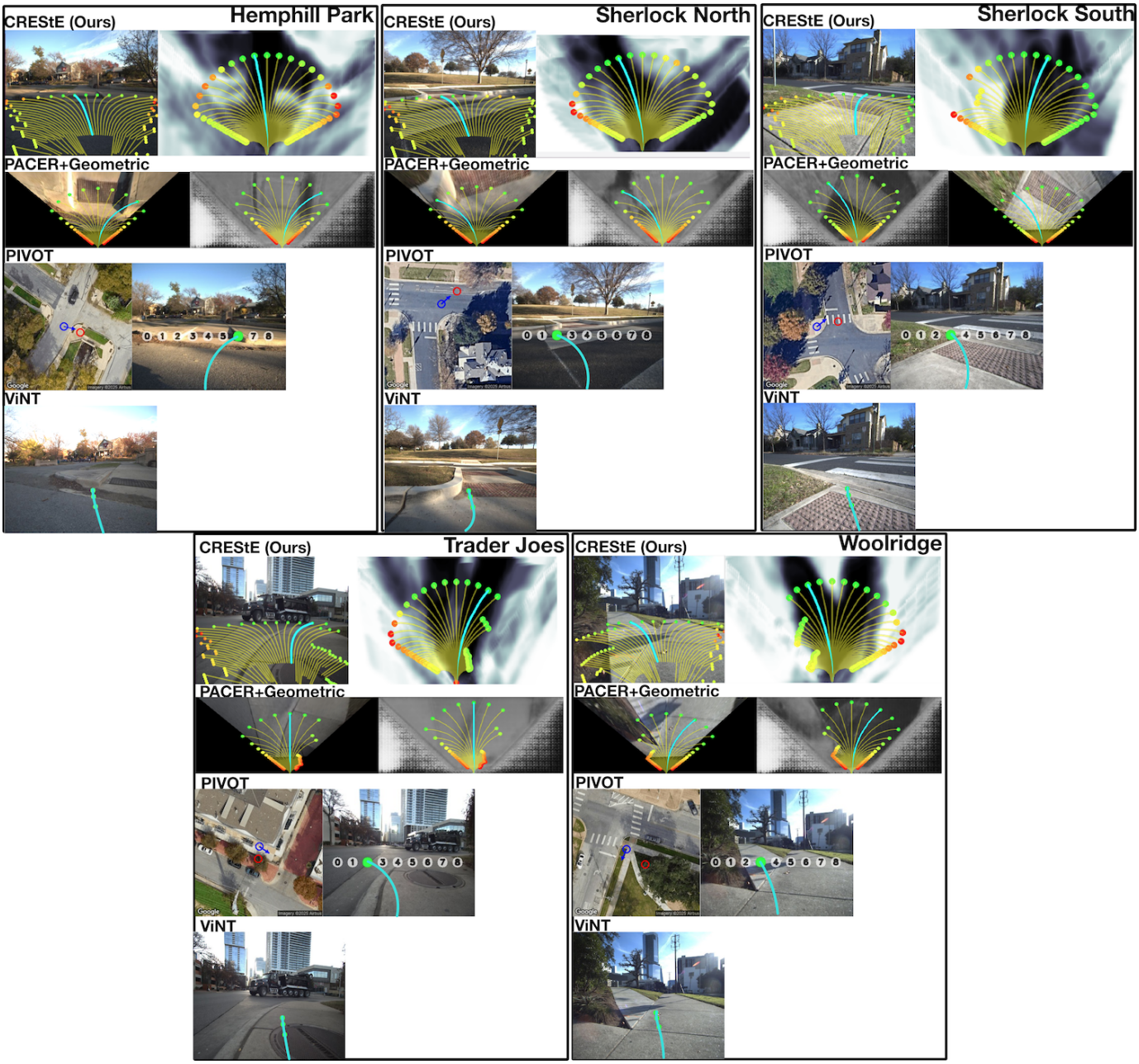

Fig. 9: Qualitative comparison of local paths planned by different learned baselines in different geoegraphic locations: Hemphill Park, Sherlock North, Sherlock South, Trader joes, Woolridge. We visualize each chosen path either in bird's eye view (BEV) or on the front view RGB image in aqua blue. For PACER+Geometric [20], we provide the BEV image used by the model. For PIVOT [22], we show the annotated satellite image and front view image used to prompt the VLM. For ViNT [35], we front view RGB image given as input. For more information regarding each baseline, we refer readers to Appendix Sec. X-D1.

cessfully maintain course in straight corridors and sidewalks, but struggles to consider factors like curb cuts and terrains. We hypothesize this is because our dataset is not sufficiently large 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 which factors influence the expert to diverge from the straight line path.

2) Comparing Different Variations of CRESTE*:* We evaluate four variations of CRESTE in the same geographic location under adverse lighting conditions in Fig. 10. Our monocular RGB only model, CRESTE-RGB, is identical to the original model but does not process sparse depth inputs. In CRESTE-RGB-FROZEN, we replace the original RGB-D encoder with a frozen Dinov2 encoder (VIT-S14), a 21M parameter pre-trained model, and downsample the input images from 512x612 to 210x308. This is essential for performing real-time inference using the onboard GPU. CRESTE-STEREO uses an EfficientNet-B0 [42] encoder to extract features from a rectified stereo RGB pair and the

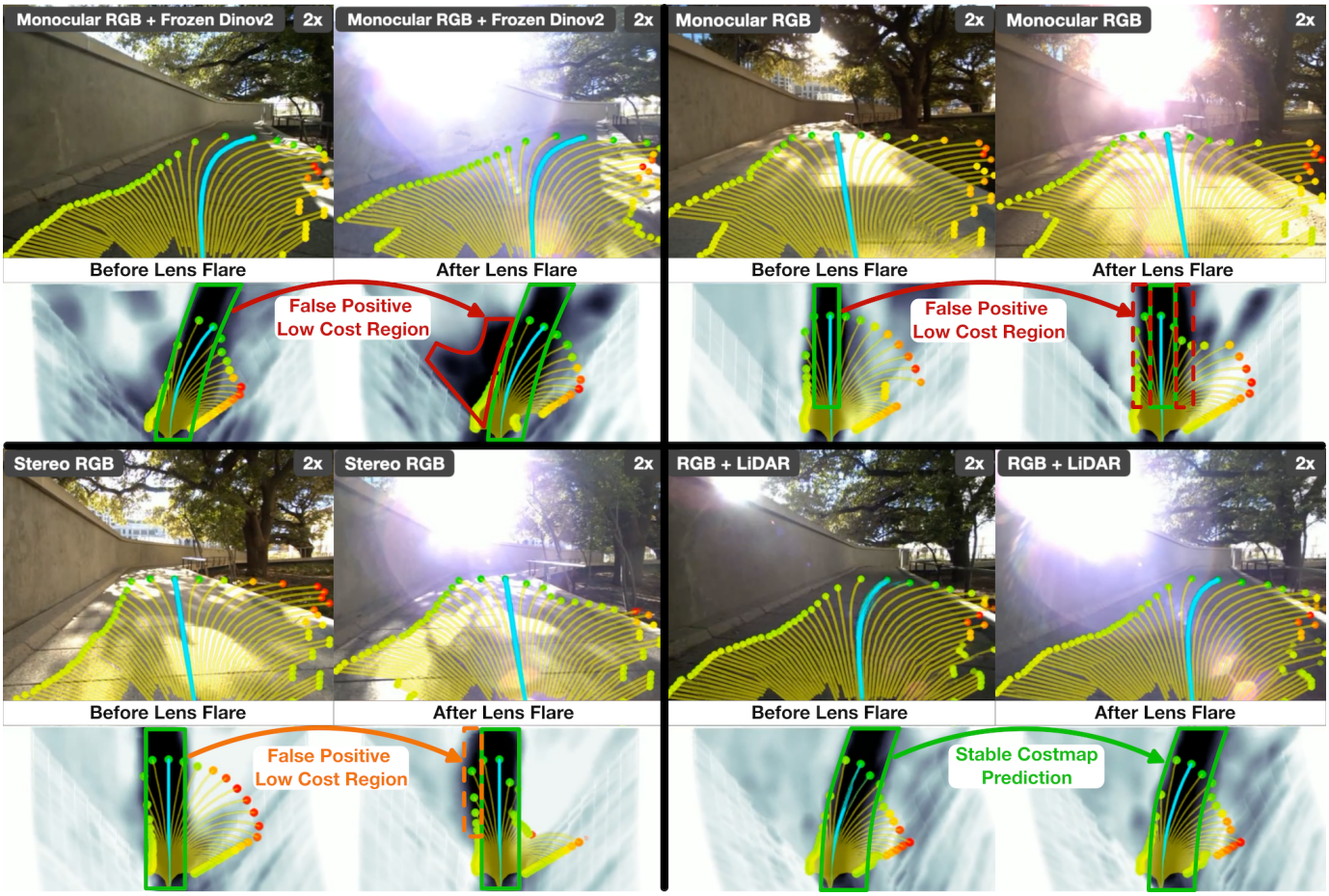

Fig. 10: Qualitative comparison of different CRESTE architecture variations in geographic location 6, Blanton Museum. These variants consist of: 1) CRESTE-RGB-FROZEN (top left), our model but only using monocular RGB inputs and a frozen Dinov2 [25] encoder, 2) CRESTE-RGB (top right), our model but only using monocular RGB inputs with a distilled RGB encoder, 3) CRESTE-STEREO (bottom left), our model but only using stereo RGB inputs and an additional stereo RGB backbone [37] for depth prediction, 4) CRESTE (bottom right), our original model using monocular RGB and sparse depth from LiDAR as inputs. We visualize the predicted bird's eye view (BEV) costmap and front view RGB image with the chosen trajectory in aqua. For more information regarding each baseline, we refer readers to Appendix Sec. X-D2.

MobileStereoNet [37] backbone to classify the depth of each pixel. For CRESTE-STEREO, we backproject image features from the RGB backbone z_{rgb} from the left image only to maintain a consistent field-of-view with other variations.

We find that CRESTE-RGB and CRESTE-RGB-FROZEN are adversely affected by heavy lighting artifacts. While they are robust to most perceptual aliasing, heavy aliasing (e.g. lens flares) degrades costmap quality. As shown in Fig. 10, the monocular RGB only models inflate low cost regions beyond the traversable area under heavy perceptual aliasing. We find that CRESTE-STEREO, our stereo RGB architecture variation, performs more robustly than the monocular RGB only models, but still predicts low cost regions incorrectly at the boundaries between traversable regions. In contrast, CRESTE, which fuses monocular RGB and sparse depth from LiDAR, predicts accurate costmaps even under adverse lighting conditions. We believe this occurs because LiDAR is unaffected by lighting artifacts, allowing the model to compensate for degraded image features.