

1 Supplementary Material

1.1 Contributions

This section expands on our main contributions, detailing each technical component and its role in the overall system.

- **A preference-driven reward learning framework** that uses Plackett-Luce loss to align ranked action sets with human visual intuition, based on responses to binary feasibility queries (e.g., "Can the robot turn left?", "Can the robot accelerate?") from egocentric camera views. The resulting reward model is deployed in two downstream settings: (i) training a goal-conditioned offline policy, and (ii) augmenting a model predictive control (MPC) planner with the learned reward as a cost term—demonstrating strong generalization across diverse indoor and outdoor environments in both learning-based and classical navigation frameworks.
- **An action-conditioned visual feature aggregation mechanism** that uses a binary mask of the predicted robot path, generated via homography, to identify spatially relevant regions in the image. This mask is processed through a CNN to produce a weighting map that modulates visual features, allowing the model to aggregate highly relevant image information based on the intended action.
- **Extensive real-world evaluation** on the Clearpath Husky platform, showcasing HALO’s ability to generalize across diverse indoor and outdoor environments not encountered during training. Our reward-model-based policies operate in real time at approximately 50 Hz on a laptop (Nvidia 3060Ti GPU, Intel Core i7 CPU), and consistently outperforms state of the art vision based navigation methods rewards—achieving an 33.3% improvement in success rate, 14.9% reduction in normalized trajectory length, and 36.6% lower Fréchet distance to expert trajectories.
- **Annotated user preference dataset for sub-optimal trajectories** will be released along with the code for the paper.

1.2 Results and Analysis

This section expands the main results with additional evaluations in diverse environments and provides further qualitative analysis and visualizations. We will also define the metrics used.

1.2.1 Metrics

Metric	Definition	Description
Success Rate	$\frac{N_{\text{success}}}{N_{\text{total}}}$	Fraction of episodes where the robot reaches the goal without collisions. $N_{\text{total}}=10$
Normalized Trajectory Length	$\frac{L_{\text{actual}}}{L_{\text{expert}}}$	Ratio of the executed trajectory length to the expert trajectory length. Lower is better.
Fréchet Distance to Expert Trajectory	$\inf_{\alpha, \beta} \max_{t \in [0, 1]} \ T_r(\alpha(t)) - T_e(\beta(t))\ $	Measures the maximum distance between points on the robot trajectory T_r and expert trajectory T_e under continuous, non-decreasing time reparameterizations. This metric captures both spatial proximity and temporal alignment. Lower is better.

Table 1: Evaluation metrics used for assessing navigation performance.

All metrics are averaged over both the successful (reaching the goal) and unsuccessful trials (collision/ not reaching the goal)

1.2.2 Further Results

Scenario	Method	Success Rate (%) \uparrow	Normalized Traj. Length \downarrow (m)	Fréchet Distance \downarrow (m)
Scenario 1	DWA (†) [1]	80	1.043	1.677
	VANP [2]	60	1.205	1.216
	HER + TD3-BC (†) [3]	0	0.420	12.651
	HER + IQL (†) [4]	50	1.118	3.665
	HALO + TD3-BC	10	0.565	6.758
	HALO + IQL	80	1.049	1.140
	HALO + DWA	70	1.212	0.892
Scenario 2	DWA (†) [1]	100	1.157	0.678
	VANP [2]	60	1.294	1.682
	HER + TD3-BC (†) [3]	0	0.213	10.724
	HER + IQL (†) [4]	10	1.271	3.851
	HALO + TD3-BC	70	1.269	1.040
	HALO + IQL	90	1.030	0.759
	HALO + DWA	70	1.306	1.573
Scenario 3	DWA (†) [1]	0	0.205	6.783
	HER + TD3-BC (†) [3]	0	0.265	7.062
	HER + IQL (†) [4]	0	0.412	5.324
	HALO + TD3-BC	10	0.481	4.022
	HALO + IQL	80	1.136	1.263
	HALO + DWA	40	0.687	4.179

Table 2: Performance comparison of different navigation methods across five scenarios. **Methods with a † have the benefit of using LiDAR**

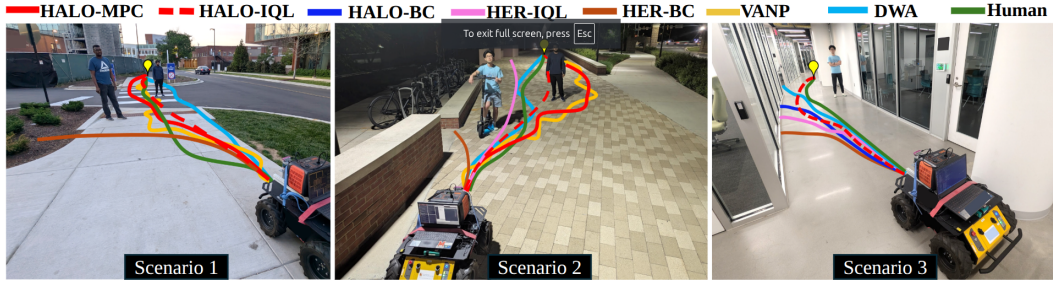


Figure 1: Comparison across three diverse real-world navigation scenarios (outdoors, low light, indoors). Each row shows the trajectory taken by different methods—DWA, VANP, HER + IQL/BC, and HALO-based policies—highlighting differences in social compliance, obstacle avoidance, and goal-reaching behavior.

Scenario 1 depicts an outdoor environment featuring pedestrians and a crosswalk. In this setting, a human would naturally prefer to avoid pedestrians and navigate along the crosswalk to reach the goal. The MPC planner equipped with the HALO reward model (HALO + MPC) mirrors this human-like behavior, producing smoother, more socially compliant trajectories. As a result, it achieves a lower Fréchet distance and a higher success rate. In contrast, the DWA planner opts for a direct path that ignores the sidewalk and pedestrian context, largely due to its reliance on LiDAR-only perception. While VANP demonstrates the ability to avoid pedestrians, it exhibits jerky motion—particularly within the crosswalk region—attributed to inconsistent action predictions. Similar to the HALO + MPC method, the IQL policy trained with the HALO model (HALO + IQL) also demonstrates social compliance by achieving a low Fréchet distance, although not as low as HALO + MPC. It achieves a success rate comparable to the DWA method, but without relying on LiDAR. The IQL policy trained using the hand-engineered reward (HER + IQL) achieved moderate success rates but did not perform as well as its HALO counterpart. The behavior cloning policy trained with HALO rewards (HALO + BC) was able to successfully avoid obstacles but failed to reach the goal. The BC policy trained using hand-engineered rewards (HER + BC) became stuck in a slight left-hand turn, ultimately driving itself into the bushes.

Scenario 2 features pedestrians and pavement regions under low-light conditions. While the DWA planner avoids pedestrians and proceeds directly toward the goal, both VANP and the MPC planner

with the HALO reward model (HALO + MPC) adopt a more socially aware approach by maintaining a large buffer radius around pedestrians—even if it requires adjusting their heading back toward the goal after passing. This behavior reflects the socially compliant navigation induced by the HALO reward model in the MPC-based planner. The IQL and behavior cloning (BC) policies trained with the HALO reward model both achieved relatively high success rates but demonstrated less social compliance. They tended to take more direct paths toward the goal, often passing closer to pedestrians compared to HALO + MPC. The policies trained with the hand-engineered reward (HER) consistently failed to reach the goal. Specifically, the behavior cloning policy trained with the hand-engineered reward (HER + BC) became stuck heading toward the curb.

Scenario 3 takes place in an indoor hallway environment with glass walls and pedestrians. LiDAR-based approaches like DWA struggle in this setting, failing to detect glass walls and resulting in collisions. Similarly, IQL and behavior cloning (BC) methods trained with hand-engineered rewards fail to generalize to this unseen scenario. In contrast, the BC policy trained using the HALO reward demonstrates relatively better performance, indicating improved generalization. Notably, the IQL policy trained with the HALO model exhibits human-like behavior, successfully recognizing both glass walls and pedestrians and navigating the hallway effectively to reach the goal.

1.2.3 Further Qualitative Analysis

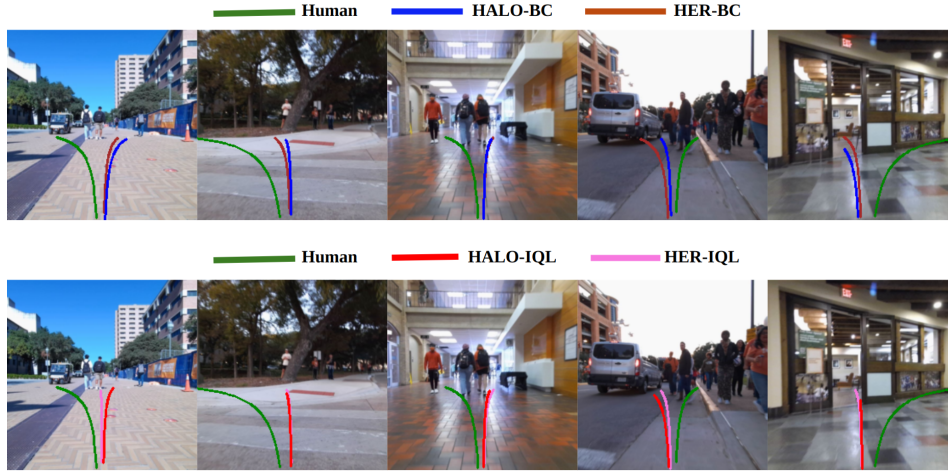


Figure 2: Qualitative Analysis for Behavioral Cloning (BC) and Implicit Q-Learning (IQL) policies for HALO and Hand Engineered Reward (HER)

In Figure 2 **Top**: we qualitatively analyze how the behavior cloning (BC) policies respond to a variety of navigation scenarios by showcasing five egocentric images and comparing the actions selected by the policies to those taken by a human operator. The HALO-BC policies are trained using reward models learned from human preference feedback, while the hand-engineered reward (HER) BC policies are trained using manually defined rewards (see Section 1.6). In all images, the human-selected actions correspond to more higher velocities or sharper turns, while the BC policies tend to produce more conservative actions. The divergence between human and policy behavior may be attributable to the human operator’s access to contextual information beyond the robot’s egocentric view. For example, in the leftmost image, the human executes a sharp left turn, potentially into oncoming traffic. While this action may appear suboptimal based solely on the visible scene, it could reflect additional cues—such as an approaching agent or a clearer path outside the frame. Given only the egocentric input, the actions chosen by the BC policies often appear more cautious and contextually appropriate. Each trajectory represents a three-second prediction horizon; the policies will generate new actions before the current trajectory completes.

The HALO-trained policies generate plausible and socially compliant behaviors across diverse scenarios. In the crosswalk image (second from the left), while the human prepares to turn left, the policies prioritize entering the sidewalk, reflecting goal-directed yet safe behavior. When pedestrians are present, the policies prefer to pass on the right, consistent with social norms. In the second-to-last image—a crowded scenario—the human elects to turn right over a yellow curb and through a dense group of pedestrians. This may be feasible due to the use of a legged robot in the SCAND dataset (see Section ??), but the policies instead select a leftward trajectory, trading crowd avoidance for slight proximity to vehicle lanes. In the rightmost image, the policies successfully infer a trajectory through an open door, demonstrating spatial reasoning using only egocentric input.

1.3 Preference Score Generation

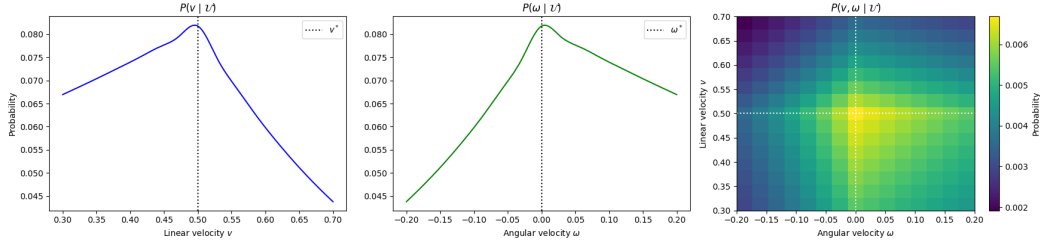


Figure 3: Visualization of the action probability distribution conditioned on user preferences. **Left:** The marginal probability $P(v | \mathcal{U})$ over linear velocities v , showing that actions greater than v^* are strongly penalized. **Middle:** The marginal probability $P(\omega | \mathcal{U})$ over angular velocities ω , indicating a strong preference for left turns. **Right:** The joint distribution $P(v, \omega | \mathcal{U}) = P(v | \mathcal{U}) \cdot P(\omega | \mathcal{U})$, capturing a separable Boltzmann weighting over the discrete action set $\mathcal{A}_{\text{local}}$. Dotted lines mark the reference action (v^*, ω^*) .

Each action (v, ω) in the local action set $\mathcal{A}_{\text{local}}$ is assigned a probability based on its distance to the reference action (v^*, ω^*) using a separable Boltzmann distribution:

$$P(v, \omega | \mathcal{U}) = \frac{\exp\left(-\frac{|v-v^*|}{\tau_v(v, \mathcal{U})}\right)}{\sum_{v' \in \mathcal{V}} \exp\left(-\frac{|v'-v^*|}{\tau_v(v', \mathcal{U})}\right)} \cdot \frac{\exp\left(-\frac{|\omega-\omega^*|}{\tau_\omega(\omega, \mathcal{U})}\right)}{\sum_{\omega' \in \Omega} \exp\left(-\frac{|\omega'-\omega^*|}{\tau_\omega(\omega', \mathcal{U})}\right)}. \quad (1)$$

The temperature values $\tau_v(\mathcal{U})$ and $\tau_\omega(\mathcal{U})$ are selected adaptively based on user preferences $\mathcal{U} \in \{0, 1\}^4$, which correspond to the four directional queries (acceleration, deceleration, turning right, and turning left). When a directional preference is indicated, the corresponding temperature is set such that the preferred direction receives at least 95% of the total probability mass in its respective marginal distribution. In the absence of preference, the distribution is flattened by assigning a higher temperature. This encourages sharper or more uniform distributions depending on the level of user certainty.

To reflect the user’s perceived desirability of a scene, the final score is scaled by a scalar factor λ , defined as:

$$\lambda(\mathcal{U}, \mathcal{U}_{\text{danger}}) = \begin{cases} -\frac{1}{1 + \sum_{i=1}^4 \mathcal{U}_i}, & \text{if } \mathcal{U}_{\text{danger}} = 1, \\ 1 + \sum_{i=1}^4 \mathcal{U}_i, & \text{otherwise.} \end{cases} \quad (2)$$

The final preference score assigned to an action (v, ω) is therefore:

$$\text{Pref}(v, \omega | \mathcal{U}) = \lambda(\mathcal{U}, \mathcal{U}_{\text{danger}}) \cdot P(v, \omega | \mathcal{U}). \quad (3)$$

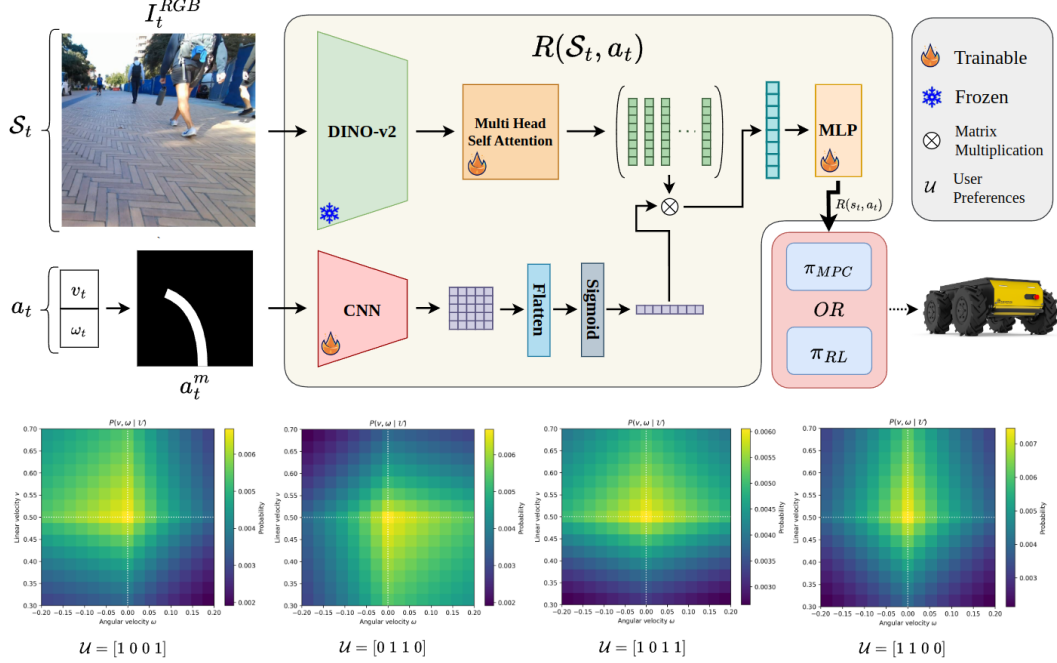


Figure 4: Architecture of the proposed reward model. Given the current observation I_t^{RGB} , a frozen DINO-v2 encoder extracts patch embeddings. Simultaneously, the candidate action $a_t = (v_t, \omega_t)$ is projected into image space via a homography transform to produce a trajectory mask a_t^m , which is passed through a trainable CNN to yield a spatial relevance weighting. This vector modulates the image patch embeddings, focusing on regions relevant to the trajectory. The resulting feature is passed through a trainable MLP to produce the scalar reward $R(s_t, a_t)$. This reward can be used in either a model predictive controller (π_{MPC}) or an offline reinforcement learning policy (π_{RL}). **Bottom:** Boltzmann distributions $P(v, \omega | \mathcal{U})$ generated from binary user preferences \mathcal{U} , used to assign preference scores to actions during reward model training.

This formulation penalizes unsafe actions flagged by the user and emphasizes confident, feasible alternatives based on their directional input.

1.4 Implementation Details

1.4.1 Model Architecture

The overall architecture of our reward model is illustrated in Figure 4. In this section, we provide additional details on the architectural components, input representations, and the specific hyperparameters used during training.

Reward Model. The reward model takes as input a visual observation $I_t^{RGB} \in \mathbb{R}^{3 \times H \times W}$ and a candidate action $a_t = (v_t, \omega_t) \in \mathbb{R}^2$. The image is passed through a frozen DINO-v2 encoder to extract N_p patch embeddings of dimension d_e , denoted as $F_t \in \mathbb{R}^{N_p \times d_e}$, which are refined via N_{sa} Transformer layers with h heads to produce $F_t^{sa} \in \mathbb{R}^{N_p \times d_e}$. In parallel, the action is projected into image space as a binary mask $a_t^m \in \mathbb{R}^{1 \times H_m \times W_m}$, which is processed by a lightweight CNN composed of stacked convolutional and pooling layers with a final sigmoid activation. This produces a spatial relevance map $W_t \in \mathbb{R}^{\sqrt{N_p} \times \sqrt{N_p}}$ which is flattened and applied as weights over the patch embeddings to yield a context vector $f_t \in \mathbb{R}^{d_e}$. This vector is then fed into an MLP to predict a scalar reward $R_t \in \mathbb{R}$.

This combination allows the self-attention layers to learn task-relevant visual patterns across the scene, while the action-conditioned weighting focuses the model on spatial regions relevant to the intended trajectory.

Policy Network. The policy network shares the same visual encoder and MLP structure as the reward model but replaces trajectory-based attention with uniform aggregation. Specifically, patch embeddings F_t are averaged to obtain a global context vector, which is passed through an MLP to predict the parameters of a Gaussian distribution over actions. During deployment, actions are sampled from this distribution; during training, gradients are propagated using the reparameterization trick. Q and V networks follow the same structure as the reward model and the policy respectively but output a single scalar value instead of action parameters.

1.4.2 Model Parameters

Symbol	Definition	Value
H, W	Input image height and width	224, 224
H_m, W_m	Mask resolution after homography projection	32, 32
N_p	Number of image patches (from DINO-v2)	256
d_e	Embedding dimension of each patch	384
f_t	Aggregated feature vector from weighted patches	\mathbb{R}^{384}
R_t	Scalar reward output	\mathbb{R}
n_v, n_ω	Number of discrete velocity and angular bins	5, 5
	Dropout rate for MLPs	0.1

Table 3: Definitions and values of key model dimensions and constants used throughout the architecture.

Trajectory Mask CNN. To convert the binary trajectory mask a_t^m into a spatial weighting map W_t , we use a lightweight convolutional neural network with the following structure:

- Conv2D(1, 4, kernel_size=3, stride=1, padding=1), ReLU
- MaxPool2D(kernel_size=2, stride=2)
- Conv2D(4, 8, kernel_size=3, stride=1, padding=1), ReLU
- Conv2D(8, 4, kernel_size=3, stride=1, padding=1)
- Conv2D(4, 1, kernel_size=3, stride=1, padding=1), Sigmoid

1.5 Training Methodology

1.5.1 Reward Model Training

The reward model is trained using the Plackett-Luce (PL) loss, which aligns predicted scalar rewards with human-labeled preference rankings over discrete action sets. We apply a 20% validation split and employ early stopping based on validation loss. In addition to the PL loss, we incorporate three auxiliary loss terms to regularize the reward predictions:

- **Reward Diversity Regularization:** Encourages the model to assign more distinct rewards to dissimilar actions. This is achieved by penalizing pairs of actions whose reward differences are too small relative to their distances in action space, using a symmetric hinge-based loss:

$$\mathcal{L}_{\text{div}} = \mathbb{E}_{(a_i, a_j)} \left[\max \left(0, \|a_i - a_j\| - \frac{1}{c} |r_i - r_j| \right) + \max (0, c|r_i - r_j| - \|a_i - a_j\|) \right]$$

where a_i, a_j are actions and r_i, r_j are their predicted rewards. The constant c defines the desired proportionality between action and reward differences.

- **Focal Regression Regularization:** A weighted mean squared error (MSE) that penalizes large reward errors more heavily, thereby encouraging the model to capture both order (from PL loss) and relative magnitudes:

$$\mathcal{L}_{\text{focal}} = \mathbb{E} \left[(\hat{r} - r)^2 \cdot |\hat{r} - r|^\gamma \right]$$

where $\hat{r} \in \mathbb{R}^Q$ and $r \in \mathbb{R}^Q$ are the predicted and target reward vectors, and γ controls the emphasis on larger errors.

- **L2 Regularization:** A regularization term that penalizes large reward magnitudes to improve numerical stability and prevent overly confident predictions:

$$\mathcal{L}_{L2} = \mathbb{E} [\|\hat{r}\|_2^2]$$

where \hat{r} denotes the vector of predicted rewards.

The final training objective is given by:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{PL}} + \lambda_1 \mathcal{L}_{\text{div}} + \lambda_2 \mathcal{L}_{L2} + \lambda_3 \mathcal{L}_{\text{focal}}$$

The training history of the reward model is shown in figure 5. We employed a two-stage training strategy: an initial warm-up phase with a higher learning rate, followed by a restarted training phase with a lower learning rate. Training was terminated early once signs of overfitting appeared during the restart phase. As noted in the hyperparameter table 4, the cosine-annealing with warm-restarts scheduler was used, which has resulted in spikes in the loss curves.

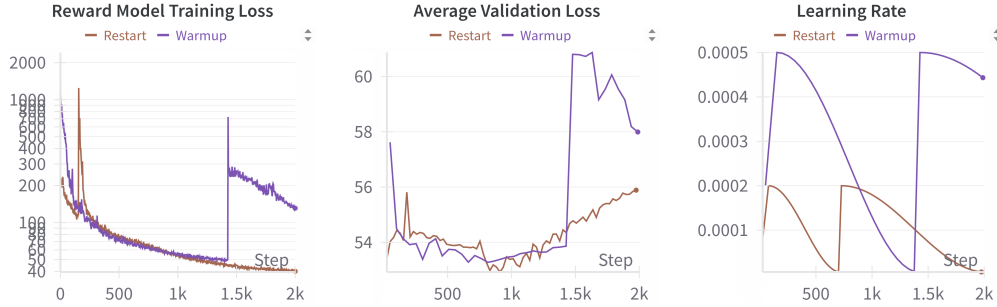


Figure 5: Reward Model Training Loss Curves:

Hyperparameter	Value
Optimizer	AdamW
Learning rate	2×10^{-4}
Weight decay	1×10^{-3}
Batch size	256
Number of epochs	200
Scheduler	Cosine Annealing with Warm Restarts
Warm-up epochs	3
Cosine T_0	25
Cosine T_{mult}	2
Validation split	20%
λ_1 (reward diversity)	2.0
λ_2 (L2 penalty)	0.01
λ_3 (focal regression)	0.05

Table 4: Training hyperparameters used for reward model optimization.

1.5.2 Policy Training

We train goal-conditioned policies using two offline methods: Behavior Cloning (TD3-BC) [5], and Implicit Q-Learning (IQL) [4]. Since the emphasis of our method is the training of the reward model, we leveraged the standard implementation of the clean offline reinforcement learning (CORL) [6] library. The offline RL implementations such as learning rate, loss functions, etc were mostly kept as-is with changes implemented when appropriate, such as including image transformations for the DINOv2 encoder, and providing the trajectory mask to the critic network. In all cases, the

reward signal is provided by our trained preference model. Similar to the reward model, pytorch’s CosineAnnealingWarmRestarts option was used for training the policy as well. This has contributed to the cyclic rise and fall in the loss curves.

The loss curves for the IQL policy trained with our custom reward model are shown in figures 6 and 7. The overlay of the loss shown is a rolling average over 20 steps, while the actual loss values are faded. The overlay for the policy evaluation loss over a holdout set was averaged over only 5 steps, since this evaluation loss was already an average over evaluating the validation set over multiple batches.

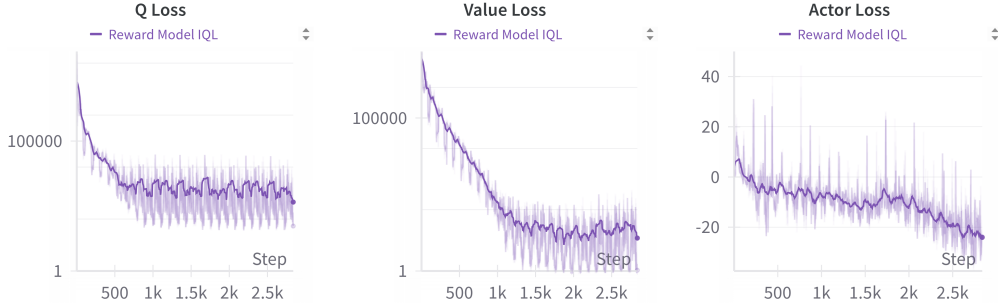


Figure 6: Training loss curves for IQL training

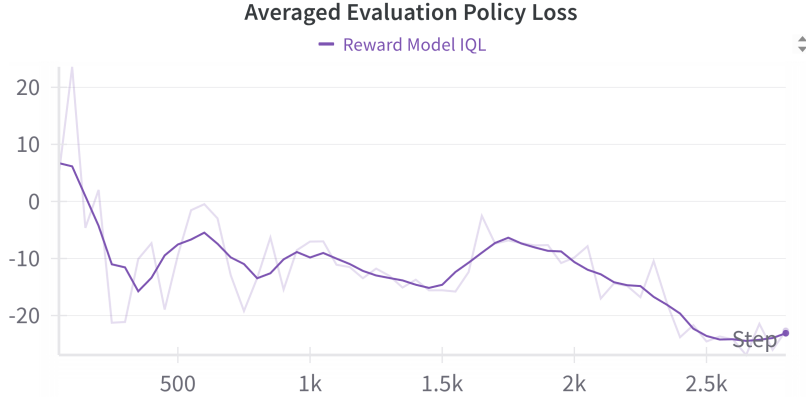


Figure 7: Evaluation curve for IQL training

Similarly, the TD3-BC are shown in figures 8 and 9. The rolling average over 20 steps was overlaid for the critic loss, and the overlay for the validation actor loss was averaged over 5 steps. The learning rate for TD3-BC was kept constant at 0.0001.

1.6 Baselines and Classical Control Details

1.6.1 Hand-Engineered Reward for Offline Comparison Methods

Each action (v, ω) is forward-simulated into a trajectory $\tau_{v, \omega} = \{(x_t, y_t, \theta_t)\}_{t=1}^T$. The comparison methods benefit from access to LiDAR, enabling the use of hand-crafted reward functions that exploit precise geometric cues for obstacle avoidance.

The reward is computed as a weighted sum of three cost components:

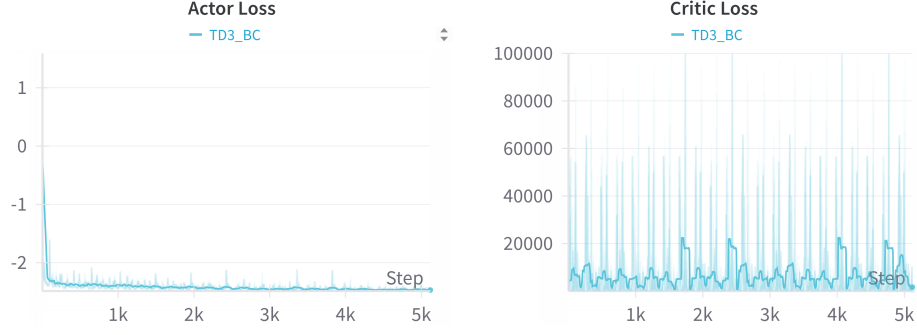


Figure 8: Training loss curves for TD3-BC

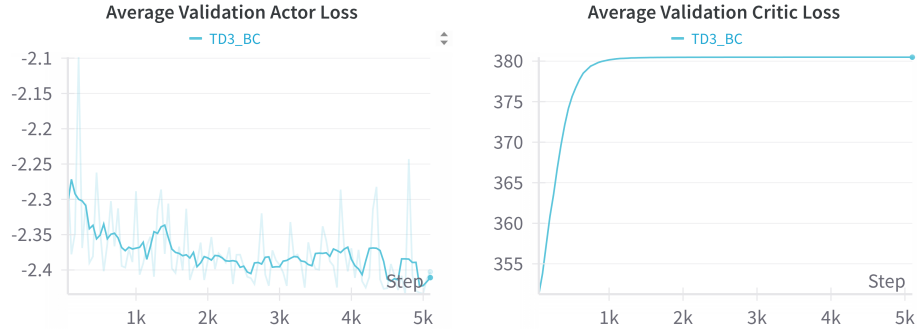


Figure 9: Evaluation loss curves for TD3-BC

$$R(v, \omega) = - \left(\lambda_{\text{goal}} \cdot \underbrace{d_{\text{goal}}(\tau_{v, \omega}, g)}_{\text{goal distance}} + \lambda_{\text{head}} \cdot \underbrace{\theta_{\text{err}}(\tau_{v, \omega}, g)}_{\text{heading error}} + \lambda_{\text{obs}} \cdot \underbrace{c_{\text{obs}}(\tau_{v, \omega})}_{\text{obstacle cost}} + \lambda_{\text{smooth}} \cdot \underbrace{\|a - a_{\text{prev}}\|_2}_{\text{smoothness}} \right)$$

where:

- $d_{\text{goal}}(\tau_{v, \omega}, g)$: Euclidean distance between the final state of the trajectory and the goal position,
- $\theta_{\text{err}}(\tau_{v, \omega}, g)$: absolute difference in heading between the final trajectory angle and the goal orientation,
- $c_{\text{obs}}(\tau_{v, \omega})$: an exponential penalty that emphasizes proximity, assigning higher cost to trajectories near LIDAR-detected obstacles by weighting inversely with distance.
- $\|a - a_{\text{prev}}\|_2$: L2 norm enforcing smoothness relative to the previous action.

We use the following weights, empirically selected based on our offline dataset:

$$\lambda_{\text{goal}} = \frac{1}{5}, \quad \lambda_{\text{head}} = 1.0, \quad \lambda_{\text{obs}} = \frac{1}{8}, \quad \lambda_{\text{smooth}} = 1.0$$

1.6.2 Goal Conditioning with HALO

To enable goal-directed navigation with our learned reward model, we modify the hand-engineered reward by replacing the obstacle penalty with the output of HALO, which captures human preferences, including implicit notions of obstacle avoidance and social compliance:

$$R(v, \omega) = -(\lambda_{\text{goal}} \cdot d_{\text{goal}}(\tau_{v, \omega}, g) + \lambda_{\text{head}} \cdot \theta_{\text{err}}(\tau_{v, \omega}, g)) + \lambda_{\text{halo}} \cdot \underbrace{R_{\text{HALO}}(s, a)}_{\text{learned reward}}$$

This formulation preserves geometric alignment to the goal while incorporating semantics from human-labeled preferences. All existing weights are retained from the hand-engineered formulation, and we set $\lambda_{\text{halo}} = \frac{1}{5}$.

1.6.3 DWA Planner Configuration

We adopt a standard goal-conditioned Dynamic Window Approach (DWA) planner with a 1.0-second planning horizon, 5 Hz control frequency, and 25 uniformly sampled linear and angular velocities. To integrate learned preferences, we simply add the negative of our reward model’s output as a cost term:

$$J(a) = J_{\text{DWA}}(s, a) - \lambda_{\text{halo}} \cdot R_{\text{HALO}}(s, a)$$

Here, $J_{\text{DWA}}(s, a)$ denotes the hand-crafted reward based on goal distance, heading error, and action transition smoothing (excluding obstacle cost), and $R_{\text{HALO}}(s, a)$ is the learned preference-based reward. We set $\lambda_{\text{halo}} = \frac{1}{5}$.

References

- [1] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.
- [2] M. Nazeri, J. Wang, A. Payandeh, and X. Xiao. Vanp: Learning where to see for navigation with self-supervised vision-action pre-training. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2741–2746, 2024. doi:[10.1109/IROS58592.2024.10802451](https://doi.org/10.1109/IROS58592.2024.10802451).
- [3] S. Fujimoto and S. S. Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.
- [4] I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.
- [5] S. Fujimoto and S. S. Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.
- [6] D. Tarasov, A. Nikulin, D. Akimov, V. Kurenkov, and S. Kolesnikov. CORL: Research-oriented deep offline reinforcement learning library. *Advances in Neural Information Processing Systems*, 36:30997–31020, 2023. URL <https://api.semanticscholar.org/CorpusID:252873315>.