OVSegDT: Segmenting Transformer for Open-Vocabulary Object Goal Navigation

Supplementary Material

Appendix

This appendix provides additional analyses, implementation details, and experimental results that complement the main paper.

- Sec. A describes the hyperparameters used for the policy model and algorithms.
- Sec. B presents an analysis of the employed training strategies.
- Sec. C examines the role of the entropy threshold in the training strategies.
- Sec. D analyzes the confidence threshold of the segmentation model and its impact on the navigation performance of OVSegDT.
- Sec. E provides a breakdown and analysis of the individual training loss components.
- Sec. F presents a statistical analysis of the main experimental results.
- Sec. G evaluates the impact of observation components on navigation performance.
- Sec. H analyzes the relationship between segmentation error and navigation quality.
- Sec. I reports performance metrics, including inference speed and GPU memory usage.
- Sec. J presents results and insights from real-world robot experiments.
- Sec. K provides an extended overview of related work on entropy-guided adaptive learning methods.

A. Hyperparameters

Tab. A lists the key hyperparameters used for the policy model.

Table A. Policy Model Hyperparameters

Component	Parameter	Value
Visual encoder	Backbone Network	SigLIP
	Fusion Type	Concatenation
	Use Visual Query	True
	Use Residual Connections	True
Transformer	Base Model	LLaMA
	Number of Layers	4
	Number of Attention Heads	8
	Hidden Dimension Size	512
	MLP Hidden Dimension Size	1024
	Max Context Length	100
	Shuffle Position IDs	True
Training parameters	Learning Rate	2.5×10^{-4}
	Parallel environments	40

Table B. Proximal Policy Optimization (PPO) Hyperparameters

Component	Parameter	Value
PPO	Clip Parameter (ϵ)	0.2
	PPO Epochs	1
	Mini-batches	2
	Value Loss Coefficient	0.5
	Entropy Coefficient	0.01
	Learning Rate (LR)	2.5×10^{-4}
	Adam Epsilon (ϵ)	1×10^{-5}
	Max Gradient Norm	0.2
	Steps per Update	100
	Use GAE	True
	Discount Factor (γ)	0.99
	GAE Lambda (λ / τ)	0.95
	Linear Clip Decay	False
	Linear LR Decay	True
	Reward Window Size	50
	Normalized Advantage	False

Table C. Agent and Observation Configuration Parameters

Component	Parameter	Value
NavMesh	Agent Max Climb	0.1
	Cell Height	0.05
Environment	Turn Angle (degrees)	30
Agent	Height	1.41
	Radius	0.17
	RGB Sensor Width (pixels)	360
	RGB Sensor Height (pixels)	640
	Horizontal FOV (degrees)	42
	RGB Sensor Position (x, y, z)	[0, 1.31, 0]

Tab. B lists the key hyperparameters used for the Proximal Policy Optimization (PPO) algorithm.

Tab. C details the configuration parameters for the agent and its observations within the simulation environment.

B. Analysis of Training Strategies

As shown in Fig. A, the naïve PPO agent makes little progress. However, Success Rate (SR) alone doesn't reveal *how* the agent reaches its goal. To address this, Fig. B includes the average number of *collisions*, a useful proxy for safety since it's directly tied to the PPO reward—fewer collisions suggest the agent is genuinely learning to navigate rather than memorizing paths. While DAgger achieves near-perfect SR on training scenes, it averages over 50 collisions per episode, indicating strong overfitting. The DAgger+PPO combined loss reduces collisions initially, but because the PPO signal doesn't fully outweigh the DAgger loss, the agent regresses to unsafe behavior, leading to a drop in validation performance.

The EarlySwitcher highlights the sensitivity of manual objective scheduling. Gradually shifting from imitation to reinforcement between 40M and 60M steps reduces colli-

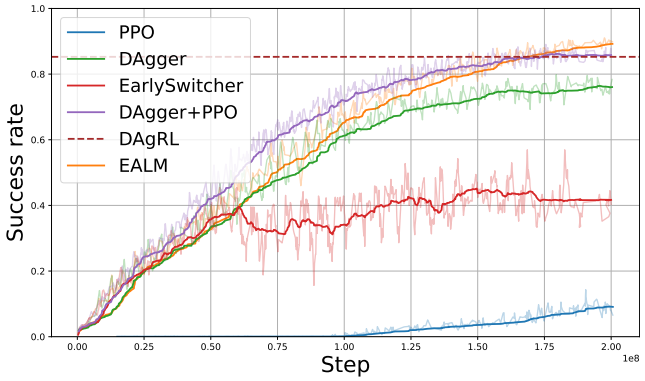

Figure A. Training curves of **Success Rate** (higher is better) for the considered switching strategies. Our entropy-adaptive EALM reaches top performance with the fewest samples.

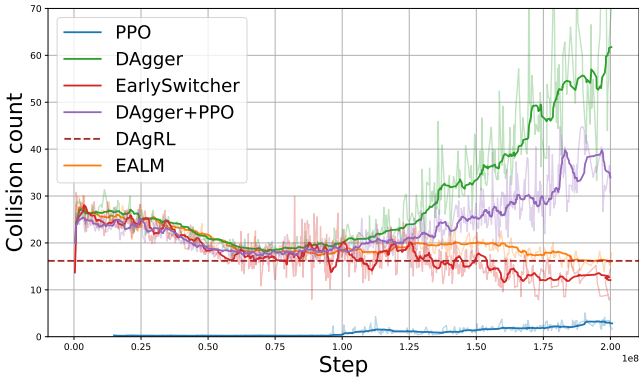

Figure B. Training curves of **Collision Count** (lower is better). Collisions serve as a safety proxy; EALM steadily reduces collisions whereas other methods eventually over-fit and regress.

sions (to about 26), but it takes over 150M steps for the policy to regain the SR lost during the transition. Identifying the optimal switching point would require extensive hyperparameter tuning. EALM avoids this issue by *continuously* adjusting the balance between imitation learning and reinforcement learning based on policy entropy. When the agent is uncertain, imitation dominates; as confidence grows, reinforcement naturally takes over. This adaptive approach combines the strengths of both methods—EALM converges twice as fast as the best baseline while achieving the fewest collisions.

C. Analysis of the Entropy Threshold of the Training Strategies

Objective. The efficacy of EALM relies on defining an appropriate "confidence" window for the policy. The hyperparameters H_{low} and H_{high} determine the entropy range over which the agent transitions from Imitation Learning (IL) to Reinforcement Learning (RL). This section analyzes how shifting this window—specifically the lower bound H_{low}—affects training dynamics and final performance.

Experimental Setup. We compare three configurations for the lower entropy bound: $H_{low} \in \{0.30, 0.35, 0.40\}$, while keeping the upper bound fixed at $H_{high} = 0.75$. The upper bound H_{high} represents the entropy level at which the policy begins to accept partial RL signals. The lower bound H_{low} represents the point of maximum confidence where the IL signal is fully deactivated ($\lambda_{PPO} = 1$).

Dynamics Analysis. Fig. F illustrates the impact of these thresholds on three key metrics:

- **PPO Ratio (Transition Speed).** The center plot (Fig. D) shows the evolution of the mixing coefficient λ_t (PPO weight) over time. A lower H_{low} (e.g., 0.30, blue curve) requires the agent to achieve higher certainty (lower entropy) before fully switching to PPO. Consequently, the transition to pure RL is slower compared to the 0.40 setting (green curve), which switches earliest.
- **Success Rate (Performance).** The left plot (Fig. C) reveals that switching too early ($H_{low} = 0.40$) is detrimental. The green curve exhibits a dip in success rate mid-training because the agent stops imitating the expert before it has fully mastered the necessary navigation primitives. Conversely, switching too late ($H_{low} = 0.30$, blue curve) delays the benefits of RL exploration, slowing down the mastery of complex scenarios.
- **Collision Count (Safety).** The right plot (Fig. E) demonstrates the safety trade-off. Prolonged imitation ($H_{low} = 0.30$) leads to higher collision rates in the intermediate phase because the agent overfits to expert trajectories without receiving negative RL rewards for collisions. The balanced configuration ($H_{low} = 0.35$, orange curve) achieves the optimal compromise, minimizing collisions while maximizing success.

Conclusion. The configuration ($H_{low} = 0.35, H_{high} = 0.75$) yields the best performance. This empirically validates our heuristic derived in the main text, balancing the need for stabilizing demonstrations with the necessity of safety-critical reinforcement learning.

D. Analysis of the Confidence Threshold of the Segmentation Model on Navigation Performance of OVSegDT

In our experiments with predicted segmentation, we use a pretrained YOLOE model based on YOLOv8-L. We analyze how the confidence threshold of the YOLOE model for each category affects the percentage of episodes with successful navigation to objects of that category. Fig. G,

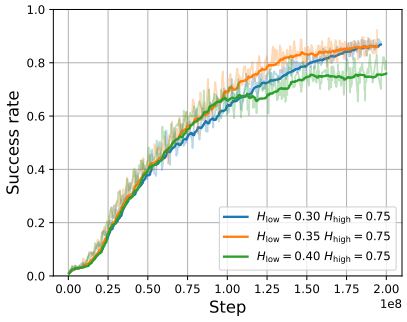

Figure C. Success Rate

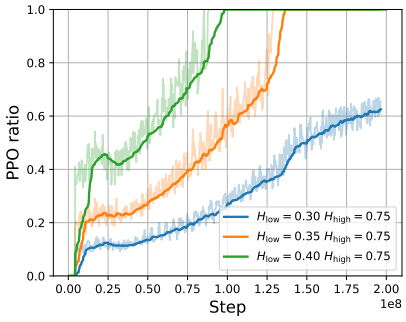

Figure D. PPO Ratio

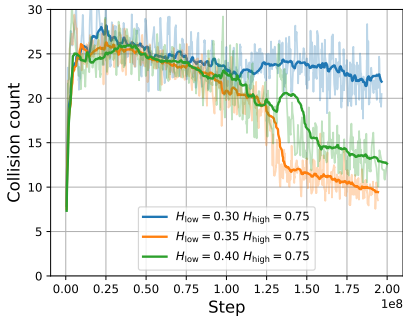

Figure E. Collision Count

Figure F. Ablation study of entropy thresholds in EALM. We vary the lower entropy bound H_{low} while keeping $H_{high} = 0.75$ fixed. The $(H_{low} = 0.35, H_{high} = 0.75)$ configuration achieves optimal balance, reaching the highest success rate while maintaining the lowest collision count.

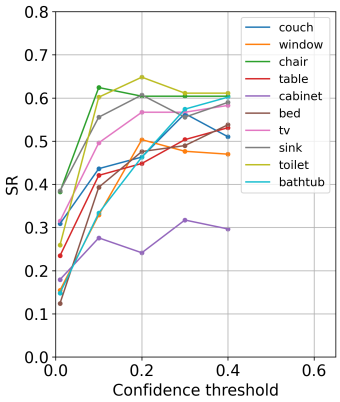

Figure G. Dependence of the success rate (SR) on the confidence threshold of the YOLOE model for different categories from the *val seen* split of HM3D-OVON. The plots are shown for the top 10 categories by number of episodes.

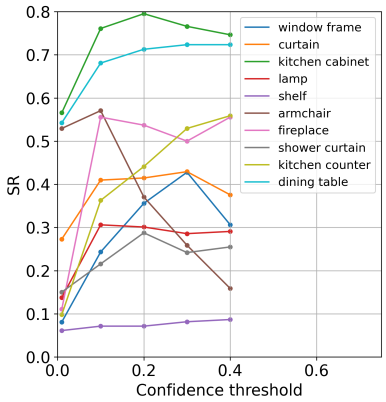

Figure H. Dependence of the success rate (SR) on the confidence threshold of the YOLOE model for different categories from the *val seen synonyms* split of HM3D-OVON. The plots are shown for the top 10 categories by number of episodes.

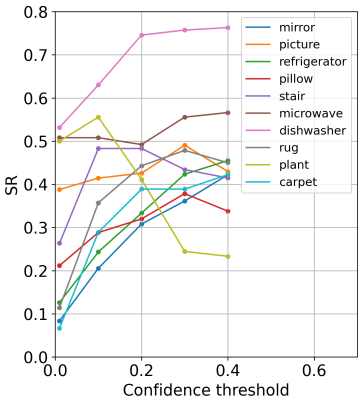

Figure I. Dependence of the success rate (SR) on the confidence threshold of the YOLOE model for different categories from the *val unseen* split of HM3D-OVON. The plots are shown for the top 10 categories by number of episodes.

Fig. H, and Fig. I illustrate the relationship between success rate and different confidence thresholds for the top 10 categories by number of episodes from each split of HM3D-OVON.

The analysis shows that some categories are challenging for the pretrained YOLOE model and require a low confidence threshold for successful recognition (e.g., *plant* from *val unseen* or *armchair* from *val seen synonyms*). On the other hand, some categories require a high confidence threshold to be reliably distinguished from others (e.g., *kitchen counter* from *val seen synonyms* or mirror from *val unseen*). For most categories, the optimal confidence threshold falls within the range of 0.1 to 0.3.

E. Analysis of Training Loss Components

Fig. J visualizes how each loss term evolves over 200 M environment steps. The *PPO actor* loss drops within the first 10 M steps and then asymptotically approaches zero, reflecting rapid policy improvement. The *critic* loss spikes early, stabilizing once value estimates become consistent. During the imitation-heavy phase the *DAgger* loss is low; it rises after ∼160 M steps when EALM hands full control to PPO, confirming that the optimization objective has indeed shifted. Finally, the *segmentation* loss falls quickly and remains an order of magnitude lower than the policy losses, indicating that the auxiliary head is able to reconstruct accurate goal masks throughout training.

Performance trends on the HM3D-OVON *train* split are shown in Fig. K. Both **Success Rate** and **SPL** increase

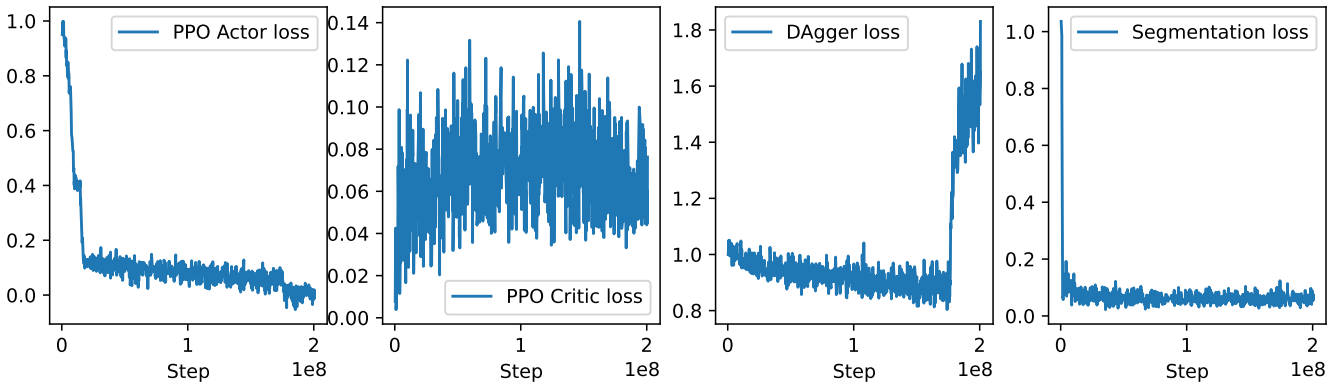

Figure J. Training losses for OVSegDT over 200 M environment steps. **Left to right:** (*i*) PPO actor loss; (*ii*) PPO critic loss; (*iii*) DAgger (behavior-cloning) loss; (*iv*) Segmentation loss.

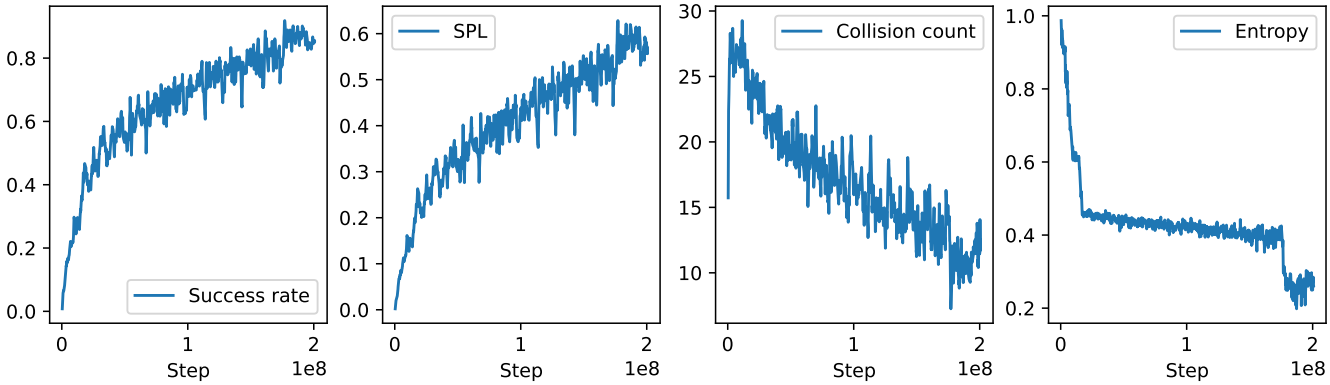

Figure K. Metrics on the HM3D-OVON *train* split during training. **Left to right:** (*i*) Success Rate (SR); (*ii*) Success weighted by Path Length (SPL); (*iii*) Collision count; (*iv*) Policy entropy.

Table D. Statistical comparison of different switching strategies on HM3D-OVON benchmark.

H_0 Hypothesis	Metric	Split	p-value	Conclusion
No significant evidence that DAgger+PPO outperforms DAgger.	collisions	val seen	0.004	Significant difference
No significant evidence that DAgger+PPO outperforms DAgger.	collisions	val unseen	0.03	Significant difference
No significant evidence that EarlySwitcher outperforms DAgger+PPO.	collisions	val seen	0.008	Significant difference
No significant evidence that EarlySwitcher outperforms DAgger+PPO.	collisions	val unseen	0.01	Significant difference
No significant evidence that EALM outperforms DagRL.	SR	val seen	0.001	Significant difference
No significant evidence that EarlySwitcher outperforms DAgger+PPO.	SR	val unseen	0.03	Significant difference
No significant evidence that EarlySwitcher outperforms DAgger+PPO.	SPL	val unseen	0.002	Significant difference

Table E. Statistical comparison of different observation types and training objectives on Val Unseen split of HM3D-OVON benchmark.

H_0 Hypothesis	Metric	Split	p-value	Conclusion
No significant evidence that OVSegDT+Goal Mask+r^{sem}+$\mathcal{L}_{\text{seg}}$ outperforms OVSegDT+Goal Mask+r^{sem}.	SR	val unseen	0.0003	Significant difference
No significant evidence that OVSegDT+Goal Mask+r^{sem} +$\mathcal{L}_{\text{seg}}$ outperforms OVSegDT+Goal Mask+r^{sem}.	SPL	val unseen	0.0001	Significant difference
No significant evidence that OVSegDT+Goal Mask+r^{sem}+$\mathcal{L}_{\text{seg}}$ outperforms OVSegDT+Goal Mask+$\mathcal{L}_{\text{seg}}$.	SR	val unseen	10^{-6}	Significant difference
No significant evidence that OVSegDT+Goal Mask+r^{sem} +$\mathcal{L}_{\text{seg}}$ outperforms OVSegDT+Goal Mask+$\mathcal{L}_{\text{seg}}$.	SPL	val unseen	10^{-6}	Significant difference
No significant evidence that OVSegDT+Goal Mask+r^{sem}+$\mathcal{L}_{\text{seg}}$ outperforms OVSegDT.	collisions	val unseen	0.02	Significant difference

monotonically, while the **collision count** falls to fewer than ten per episode - evidence that the agent learns safer, shorter paths as training progresses. The **entropy** curve drops sharply during the first imitation-dominated stage, slowly decreases while EALM mixes objectives, and exhibits a second rough decline when the algorithm switches to a PPO-only loss near the end of training. Importantly, this late entropy reduction *does not* introduce instability: all task metrics continue to improve smoothly, demonstrating that EALM's automatic transition preserves training stability and leads to a confident yet robust final policy.

Table F. Statistical comparison of different strategies of adaptation to predicted segmentation for OVSegDT method on Val Unseen split of HM3D-OVON benchmark.

H_0 Hypothesis	Metric	Split	p-value	Conclusion
No significant evidence that OVSegDT+YOLOE calibrated outperforms OVSegDT+YOLOE.	SR	val unseen	$3 \cdot 10^{-5}$	Significant difference
No significant evidence that OVSegDT+YOLOE calibrated outperforms OVSegDT+YOLOE.	SPL	val unseen	0.0006	Significant difference
No significant evidence that OVSegDT+YOLOE calibrated+Predicted mask finetune outperforms OVSegDT+YOLOE calibrated.	SR	val unseen	10^{-5}	Significant difference
No significant evidence that OVSegDT+YOLOE calibrated+Predicted mask finetune outperforms OVSegDT+YOLOE calibrated.	SPL	val unseen	10^{-4}	Significant difference

F. Statistical Analysis

We analyze the statistical significance of the impact of different components of our method. Each algorithm is run three times, and we perform the one-sided t-test for unpaired samples to determine the level of statistical significance. Tab. D, Tab. E, and Tab. F present the comparison results and corresponding conclusions. Thus, the conclusions presented in the main text are supported by the statistical significance of the results.

G. Analysis of Observation Components Impact on Navigation Performance

The OVSegDT method uses two types of goal prompts at inference time: a textual instruction and a visual cue in the form of a binary mask of the target object. We analyze the impact of each of these components during inference in Tab. G. The observation setting Goal Mask+Text Goal corresponds to the original version of OVSegDT. When using only the Goal Mask at inference time, we replace the part of the observation corresponding to the text instruction with a dummy zero input. In this case, we observe a drop in navigation performance. In the case of navigation using only the text goal, we always feed a zero mask to the model during inference. This leads to a complete breakdown of OVSegDT's navigation, highlighting the importance of high-quality masks for successful navigation. Thus, although binary goal masks are a key component of the observation, the textual instruction also contributes to improving navigation quality in OVSegDT.

Table G. Analysis of the impact of the goal mask and textual instruction on navigation performance on Val Unseen of HM3D-OVON benchmark.

Method	Observation	SR	SPL	Collisions
OVSegDT	Goal Mask+Text Goal	**44.7**$\pm$0.4	**20.6** $\pm$ 0.2	45.4$\pm$0.6
OVSegDT	Goal Mask	12.6 $\pm$ 0.6	4.5 $\pm$ 0.2	70.0$\pm$0.5
OVSegDT	Text Goal	0.1 $\pm$ 0.07	0.01 $\pm$ 0.02	68.0$\pm$0.2

H. Analysis of Segmentation Error and Navigation Quality

We analyze how the adaptation methods we introduced for applying OVSegDT to the predicted segmentation affect segmentation and navigation quality. To evaluate segmentation quality, we use the false positive rate (FPR) and false negative rate (FNR) metrics. A detailed definition of these metrics can be found in Sec. H.2. As shown in Tab. H, the main source of errors is false negative predictions. Introducing confidence-threshold calibration for YOLOE reduces the FPR and thus improves navigation quality. Finetuning on predicted masks improves navigation quality by reducing the false negative rate, i.e. the model learns to find trajectories in which YOLOE can successfully segment the target objects. Finally, we assess the robustness of the confidence-calibration process to photometric augmentations. A detailed description of the augmentations used is provided in Sec. H.1. Owing to the pre-trained YOLOE, the segmentation performance remains stable under these perturbations (see Tab. H). Calibration is introduced to mitigate errors arising from synonym overlap within the navigation vocabulary and to address the real-to-simulation domain gap, particularly for classes such as plant, stairs, curtains.

Table H. Analysis of the segmentation errors source and the navigation performance on Val Unseen of HM3D-OVON benchmark.

Segmentation method	Predicted mask fine-tune	SR, %	FPR, %	FNR, %
YOLOE	✗	28.8$\pm$0.4	11.9$\pm$0.3	68.0$\pm$0.3
YOLOE calib.	✗	35.7$\pm$0.4	**5.5** $\pm$ 0.3	68.5$\pm$0.3
YOLOE calib.	✓	**44.7**$\pm$0.4	7.2 $\pm$ 0.2	**61.3**$\pm$0.3
YOLOE calib. (image aug.)	✓	42.2$\pm$0.7	7.3 $\pm$ 0.2	63.6$\pm$0.3

H.1. Random Lighting Augmentation

We apply a random lighting augmentation module to each input image $\mathbf{I} \in [0,1]^{C\times H\times W}$ to evaluate navigation robustness under controlled photometric perturbations. The augmentation operates per-image and uses the hyperparameters listed in Tab. I.

Color Jitter. Brightness, contrast, saturation, and hue are randomly perturbed independently. For brightness, contrast, and saturation we sample

$$\alpha_p \sim \mathcal{U}(1-p,\ 1+p), \quad p \in \{p_{\text{bright}}, p_{\text{contrast}}, p_{\text{satur}}\}. \quad (1)$$

For hue we sample

$$\delta_h \sim \mathcal{U}(-p_{\text{hue}}, p_{\text{hue}}). \quad (2)$$

Table I. Hyperparameters used in the Random Lighting Augmentation.

Parameter	Value
p_{bright}	0.5
p_{contrast}	0.5
p_{satur}	0.3
p_{hue}	0.1
$\gamma_{\min}^{dist}$	0.6
$\gamma_{\max}^{dist}$	1.4
p_{shadow}	0.5
$s_{\min}$	0.4
$s_{\max}$	0.8

The sequential jitter transformation applied to each image first adjusts its brightness, then adjusts its contrast, followed by a saturation adjustment, and finally applies a hue shift. Each of these operations uses independently sampled random factors for brightness, contrast, saturation, and hue.

Gamma Correction. To introduce nonlinear photometric distortions, we sample

$$\gamma^{dist} \sim \mathcal{U}(\gamma_{\min}^{dist}, \gamma_{\max}^{dist}), \tag{3}$$

and adjust each image by raising its pixel values to this exponent.

Random Shadow. With probability p_{shadow}, a rectangular region $R = [x_1, x_2] \times [y_1, y_2]$ is darkened by a multiplicative factor

$$s \sim \mathcal{U}(s_{\min},\ s_{\max}). \tag{4}$$

This yields:

$$\mathbf{I}(x,y) \leftarrow \begin{cases} s \cdot \mathbf{I}(x,y), & (x,y) \in R, \\ \mathbf{I}(x,y), & \text{otherwise.} \end{cases}$$

Output. All transformed images are clipped to $[0, 1]$ and reassembled into a batch. This augmentation introduces illumination changes, contrast shifts, nonlinear intensity warping, and localized shadows, enabling systematic stress-testing of the segmentation model and therefore OVSegDT navigation under diverse lighting conditions.

H.2. Segmentation Evaluation Metrics

To evaluate segmentation performance, we compute the False Positive Rate (FPR) and False Negative Rate (FNR) based on the pixel-wise predictions of the model. For each predicted mask pred_mask and its corresponding ground-truth mask gt_mask, we first check if both masks are empty (i.e., no foreground pixels). In this case, we count it as a True Negative (TN). Otherwise, we compute the Intersection over Union (IoU) between the predicted and ground-truth masks. If the IoU exceeds 0.5, the prediction is considered a True Positive (TP). If the IoU is below 0.5 and the predicted mask has foreground pixels, it is counted as a False Positive (FP); if the predicted mask is empty while the ground-truth mask is not, it is counted as a False Negative (FN). Using these counts, FPR and FNR are computed as:

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}, \tag{5}$$

$$\text{FNR} = \frac{\text{FN}}{\text{FN} + \text{TP}}. \tag{6}$$

These metrics provide a quantitative measure of over-segmentation (FPR) and under-segmentation (FNR) errors in the YOLOE model during the navigation process.

I. Performance Analysis

Table J. Analysis of GPU memory usage and inference time of OVSegDT with predicted segmentation.

Segmentation source	GT	YOLOE
GPU Memory Usage (Gb)	3486	4992
Inference time (ms, NVIDIA H100)	13.6 ± 0.9	65.0 ± 14
Inference time (ms, NVIDIA Jetson Orin)	107 ± 5	132 ± 5

The performance analysis demonstrates that OVSegDT remains efficient across both ground-truth (GT) and predicted segmentation settings (see Tab. J). While the use of YOLOE-based predicted segmentation increases GPU memory usage (from 3486 Gb to 4992 Gb) and inference latency on the NVIDIA H100 (from 13.6 ± 0.9 ms to 65.0 ± 14 ms), the impact on the embedded NVIDIA Jetson Orin platform is relatively modest (from 107 ± 5 ms to 132 ± 5 ms). This indicates that incorporating predicted segmentation does not significantly slow down the model in onboard scenarios, which are critical for deployment. Overall, OVSegDT demonstrates strong efficiency in terms of GPU memory usage and maintains near real-time inference performance, making it suitable for real-time navigation policy prediction even when relying on predicted segmentation inputs.

J. Real-World Robot Experiments

We conduct demonstration experiments on a real iRobot Create 3 robot equipped with a ZED X camera, Livox MID 360 LiDAR, and an Nvidia Jetson Orin PC. Since the ZED X has a wide FoV (105°vs. 42°in the HM3D-OVON benchmark), we train a version of OVSegDT with this camera setup to improve the estimation of distances to objects and obstacles during real-robot experiments. Fig. L demonstrates the effectiveness of the OVSegDT strategy for open-vocabulary navigation in two scenarios.

In the first scenario, the robot must distinguish between two targets in front of it: a white box and clothes. The robot's trajectories show that it successfully accomplishes this task and, after detecting the target, moves directly toward it. In the second scenario, the robot must find a white

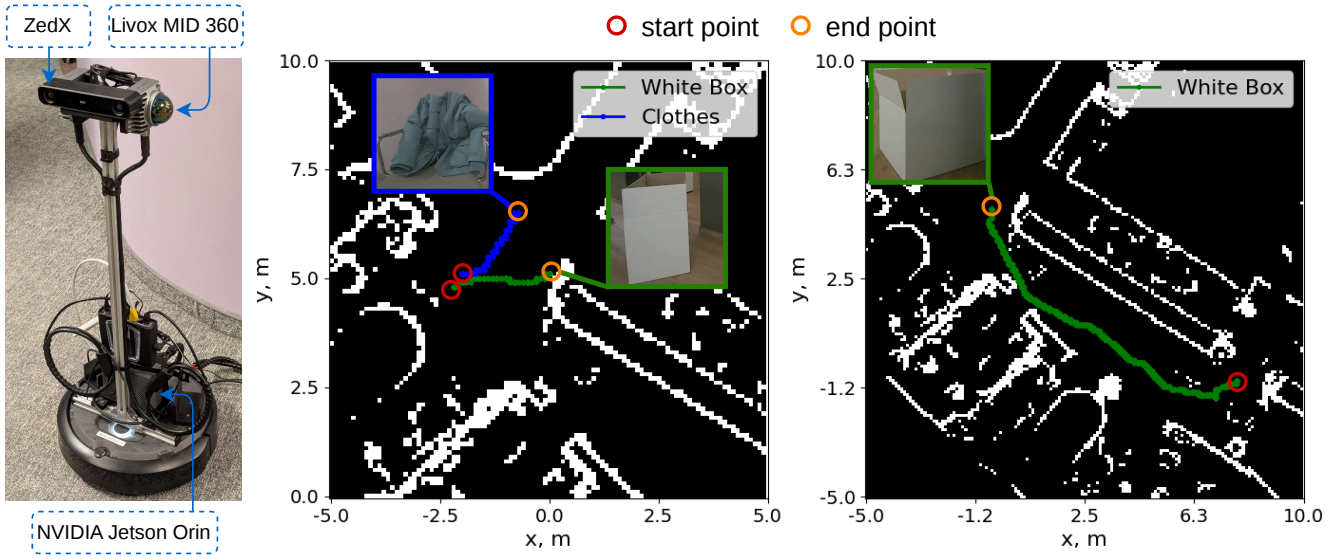

Figure L. **Left:** iRobot Create 3 equipped with a ZED X camera, Livox MID 360 LiDAR, and Nvidia Jetson Orin PC. **Right:** demonstration of the robot's trajectories on the occupancy grid for two scenarios. In the first scenario, the robot must sequentially reach each of two goals specified in natural language. In the second scenario, the robot must find a target object that is outside its field of view while avoiding obstacles.

box initially hidden from its field of view at the start of the episode. It also successfully completes this task. Additionally, when moving along long trajectories, the robot effectively avoids obstacles.

However, it is worth noting that during training, the robot receives a large reward for stopping within a small radius of the target (25 cm), while the penalty for collisions is relatively small. As a result, in real-world experiments, we observe risky behavior near the goal object. Balancing precise goal reaching with safe stopping in front of the target without a depth camera remains a subject for future work.

K. Related Works on Entropy-Guided Adaptive Learning Methods

In this section, we provide an extended overview of related work on entropy-guided adaptive learning methods.

Recent advances in adaptive learning have increasingly leveraged entropy as a guiding signal. EntAugment [5] dynamically adjusts data augmentation intensity based on model entropy to improve generalization, while EA-KD [4] utilizes entropy to re-weight samples during knowledge distillation, prioritizing high-uncertainty instances. In curriculum learning, the READ-C framework [2] utilizes relative entropy for autonomous curriculum design, and in the graph domain, entropy has been used to guide curriculum learning for contrastive tasks [6]. REALM [3] employs robust loss functions scaled by entropy to stabilize test-time adaptation. While our EALM similarly leverages entropy as an adaptive signal, it focuses on dynamically balancing imitation and reinforcement learning objectives during training rather than test-time robustness or curriculum sequencing.

In the realm of policy learning, SAC [1] maximizes policy entropy to encourage exploration. This contrasts with EALM, which uses entropy not as an exploration bonus, but as a scheduling signal to shift between learning paradigms. SafeDAgger [7] introduces a safety policy to reduce queries to expensive reference policies. EALM achieves a similar goal of efficiency but does so through automatic phase transitions guided by policy entropy, eliminating the need for an additional safety network.

Fundamentally, while these approaches use entropy to modulate hyperparameters or sample weights within a single paradigm, EALM employs policy entropy as a switching mechanism between two distinct objectives: imitation learning and reinforcement learning. EALM automatically orchestrates the transition from supervised bootstrapping to autonomous exploration, eliminating the need for manual phase scheduling.

References

[1] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870.

Pmlr, 2018. 7

[2] Muhammed Yusuf Satici, Jianxun Wang, and David L Roberts. Autonomous curriculum design via relative entropy based task modifications. *arXiv preprint arXiv:2502.21166*, 2025. 7

[3] Skyler Seto, Barry-John Theobald, Federico Danieli, Navdeep Jaitly, and Dan Busbridge. Realm: Robust entropy adaptive loss minimization for improved single-sample test-time adaptation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2062–2071, 2024. 7

[4] Chi-Ping Su, Ching-Hsun Tseng, Bin Pu, Lei Zhao, Jiewen Yang, Zhuangzhuang Chen, and Shin-Jye Lee. Ea-kd: Entropy-based adaptive knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 731–740, 2025. 7

[5] Suorong Yang, Furao Shen, and Jian Zhao. Entaugment: Entropy-driven adaptive data augmentation framework for image classification. In *European conference on computer vision*, pages 197–214. Springer, 2024. 7

[6] Chusheng Zeng, Bocheng Wang, Jinghui Yuan, Rong Wang, and Mulin Chen. Multi-task curriculum graph contrastive learning with clustering entropy guidance. *arXiv preprint arXiv:2408.12071*, 2024. 7

[7] Jiakai Zhang and Kyunghyun Cho. Query-efficient imitation learning for end-to-end autonomous driving. *arXiv preprint arXiv:1605.06450*, 2016. 7