Synthetic vs. Real Training Data for Visual Navigation

Lauri Suomela, Sasanka Kuruppu Arachchige, German F. Torres, Harry Edelman, Joni-Kristian Kämäräinen
Tampere University

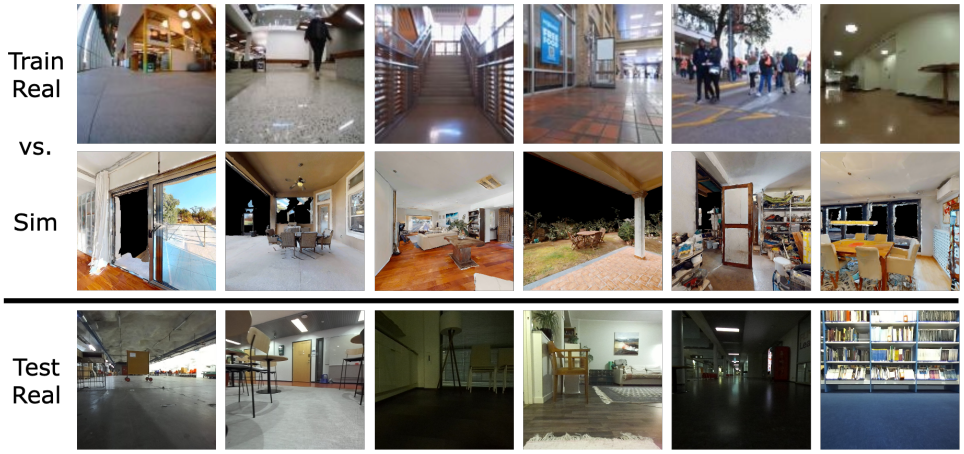

Figure 1: Training a vision-based navigation policy with real-world data vs. synthetic data from a simulator. Real-world experiments demonstrate that despite the shortcomings of synthetic data, by leveraging pretrained visual representations and large-scale on-policy data collection, the simulator-trained policy can reach a performance comparable to the policy trained on real-world data.

Abstract: This paper investigates how the performance of visual navigation policies trained in simulation compares to policies trained with real-world data. Performance degradation of simulator-trained policies is often significant when they are evaluated in the real world. However, despite this well-known sim-to-real gap, we demonstrate that simulator-trained policies can match the performance of their real-world-trained counterparts. Central to our approach is a navigation policy architecture that bridges the sim-to-real appearance gap by leveraging pretrained visual representations and runs real-time on robot hardware. Evaluations on a wheeled mobile robot show that the proposed policy, when trained in simulation, outperforms its real-world-trained version by 31% and the prior state-of-the-art methods by 50% in navigation success rate. Policy generalization is verified by deploying the same model onboard a drone. Our results highlight the importance of diverse image encoder pretraining for sim-to-real generalization, and identify on-policy learning as a key advantage of simulated training over training with real data. We make the code and model checkpoints available at `lasuomela.github.io/faint`.

Keywords: Robot learning, Visual Navigation, Sim2Real

1 Introduction

Recently, learning-based visual navigation methods have received attention as potential replacements for traditional sense-plan-act approaches that leverage geometric environment representations. Navigation systems with learned components have many advantages, for example, being able

to utilize semantic information to infer traversability [1, 2] and guide exploration [3, 4], and incorporate other task specifications in addition to metric coordinates [5]. The primary robot learning paradigms involve imitation learning (IL) from real-world robot datasets [1, 6, 7, 8, 9, 10], and reinforcement learning (RL) or imitation from scripted [11, 12] or human experts [13, 4] in simulation [14, 15, 16, 17]. The performance of learned robot policies heavily depends on the quality, quantity, and diversity of training data [18]. Real and synthetic data collection each have distinct strengths and weaknesses with respect to these factors. Real-world data collection is labor-intensive and platform-specific, yet it reduces domain shifts at policy deployment, while synthetic data generation offers greater scalability and flexibility but faces the *simulation-to-real* (sim2real) gap [19], limiting generalization to real environments.

Our work investigates how navigation performance differs between policies trained with real-world data and those trained entirely in simulation. Although prior works [20, 21, 4, 22] have assessed the sim2real gap by evaluating simulation-trained policies in simulated and real environments, they have not directly compared these policies with those learned from real-world data. We address this gap by evaluating whether simulation-only training can achieve competitive performance compared to real-world data training.

Previously proposed navigation policies have been tailored for either synthetic or real-world data. To facilitate our experiments, we introduce a novel policy architecture for visual topological navigation: **F**ast **A**ppearance-**I**nvariant **N**avigation **T**ransformer (FAINT). FAINT can be trained on either real or simulated data, is lightweight enough for deployment on resource-constrained robot hardware, and demonstrates robust sim-to-real transfer. We deployed sim and real-data-trained versions of the policy on a wheeled mobile robot and a drone, and performed more than 50 hours of evaluation in challenging real-world indoor environments.

Our findings demonstrate that a navigation policy trained entirely in simulation can perform on par with—or even outperform—those trained on real-world data. Simulation-trained FAINT surpasses its real-world-trained counterpart by 31 points and the previous state-of-the-art by 50 points in navigation success rate. Despite being trained with fully synthetic data, the policy can successfully adapt to unseen environment conditions and different robot platforms. Based on these results, we identify scalable data generation and the ability to perform on-policy learning as key advantages of simulation over real-world data.

2 Related work

Learning from real-world data. The main challenges for learning navigation from real-world demonstrations are related to generalization across different unseen environments and embodiments. Most works approach the problem with methods based on goal-conditioned imitation learning [23]. Kahn et al. [1] train a navigation model in one of the first works to demonstrate generalization to novel environments. Shah et al. [7, 8, 9] achieve generalization across robot platforms by training topological navigation models with data collected from multiple different robots. Suomela et al. [24] improve upon this line of work by reducing the dependency on robot-originated training data through the use of place recognition models [25]. We build on the architectures proposed by Shah et al. [8] and Suomela et al. [24], but find that these methods are not well-suited for sim2real transfer. To overcome this limitation, we introduce key architectural modifications that are critical for improving sim2real generalization.

Learning from synthetic data. Simulation offers scalability and tailorability, making it an attractive option for training navigation policies. However, the sim2real gap [19] remains a major challenge for real-world deployment [4]. While policies trained on raw RGB inputs with heavy domain randomization have shown some success, their generalization remains limited [26, 27, 28]. Training policies on sensor abstractions such as depth images [11, 29], segmentation masks [3, 17], and feature points [30] has shown better results. Recently, feature maps from *pre-trained visual representation* (PVR) models have proven effective for bridging the sim2real gap. Ehsani et al. [12, 31, 32, 33] use

a frozen SigLIP [34] encoder for navigation and manipulation tasks, while Silwal et al. [22] demonstrate successful sim2real transfer in image-goal navigation using a VC-1 [35] encoder. However, these large PVR's have low inference speed, making them unsuitable for real-time deployment on resource-constrained robots. In this work, we show that even smaller, distilled PVR models [36] enable sim2real transfer while being suitable for on-robot execution.

Sim2Real investigations. Substantial amounts of work have been put into studying and quantifying the sim2real gap by comparing policies' performances in simulation and the real world [20, 37, 15, 4]. What we identify as still missing is a real-world comparison of the performance of navigation models trained with synthetic and real datasets. The work most similar to ours is the investigation by Silwal et al. [22], which examines the performance of manipulation policies trained with real and simulated data in real-world settings. They find that simulated policies trained with few-shot imitation learning exhibit poor sim2real transfer and underperform compared to real-data policies. However, they also train an image-goal navigation policy using large-scale reinforcement learning in simulation, which performs well in real-world environments. Notably, they do not compare the image-goal policy to a model trained on real-world data, likely because there are no suitable real-world datasets. In contrast, our investigation focuses on visual topological navigation, a task for which real-world datasets are available. This allows us to directly compare the policies trained on real and synthetic data for the same task.

3 Methods

In this Section, we present the definition of the navigation task (Sec. 3.1), outline the architecture of the model used in the experiments (Sec. 3.2), and describe the datasets and learning approach for synthetic (Sec. 3.3) and real-world (Sec. 3.4) data.

3.1 Problem formulation

Topological visual navigation. We perform the investigation in the context of topological image-goal navigation [38, 7, 24] because it is a navigation task with some of the most extensive real-world datasets available. Topological approaches divide a navigation route into a set of intermediate subgoals $\{s_0, s_1, \ldots, s_n\}$, each with an associated image observation. These subgoals comprise a topological map $\mathcal{M}$ that is created from images collected prior to robot deployment. During navigation, at each time step t a *subgoal selection policy* π_s finds the next subgoal s_t along the route to the final navigation goal, and returns the corresponding subgoal image S_t:

$$S_t = \pi_s(O_t, \mathcal{M}) \tag{1}$$

where O_t is the current observation image. Given S_t and a sequence of P recent observations $\mathbf{O}_t = \{O_{t-P+1}, \ldots, O_t\}$, a *goal-reaching policy* π_g then produces a sequence of H robot control commands $\mathbf{a}_t = \{a_t, \ldots, a_{t+H-1}\}$ towards the subgoal:

$$\mathbf{a}_t = \pi_g(\mathbf{O}_t, S_t). \tag{2}$$

We adopt the subgoal selection method from Suomela et al. [24], and perform the selection with place recognition [25] models that can be trained with large-scale datasets from *e.g.* Google Streetview [39]. This lets us focus on the goal-reaching policies which are more tightly tied to the robot embodiment and for which the training data is more scarce.

Action space. The goal-reaching policy is parametrized by predicting trajectory *waypoints* relative to the robot coordinate frame, as it allows embodiment-agnostic control and direct comparison to relevant prior methods [7, 8, 24]. Each waypoint $a \in \mathbf{a}_t$ is a pose $a = [x, y, \theta] \in SE(2)$ where (x, y) is the position and θ the orientation. During deployment, a simple PD-controller estimates velocity commands from the waypoints.

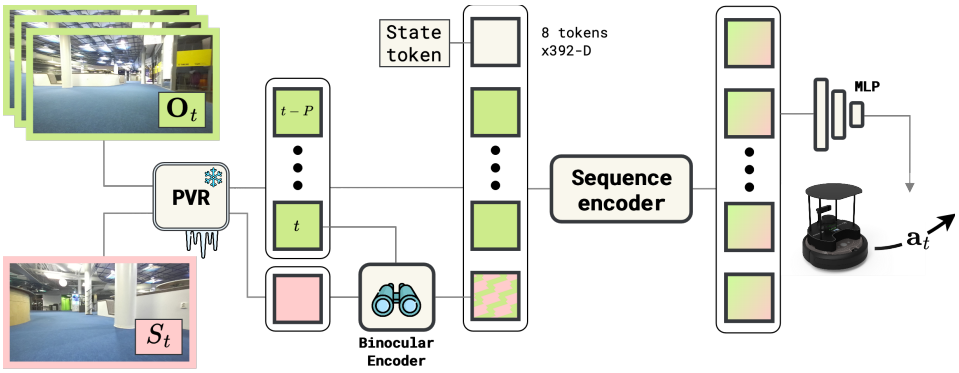

Figure 2: **Model architecture.** FAINT implements the goal-reaching policy $\mathbf{a}_t = \pi_g(\mathbf{O}_t, S_t)$. Observation and subgoal images are encoded with a frozen PVR, and a binocular encoder refines the goal tokens by conditioning on the latest observation. A sequence encoder with a predictor head then produces the actions $\mathbf{a}_t$. Subgoals S_t are obtained from a separate subgoal selection policy π_s.

3.2 FAINT model architecture

We extend the ViNT architecture [8] with two key modifications: integration of *pretrained visual representations* (PVR's) and a novel *binocular goal encoder*. FAINT has just 12M parameters—half the size of prior models [9]—enabling real-time inference on resource-constrained hardware. An overview of the architecture is shown in Figure 2. We describe each component in detail below.

Pretrained visual representation. As a key to bridging the sim-to-real appearance gap, we leverage image encoders pretrained on a diverse visual tasks. We adopt the $5M$-parameter *Tiny CDDSV* variant of the Theia encoder [36], which distills representations from CLIP [40], DiNOv2 [41], Depth Anything [42], Segment Anything [43], and ViT [44]. Despite its small size, it demonstrates robust sim-to-real generalization. The weights are frozen during training to avoid overfitting.

Binocular encoder. Previous work [8, 45] has shown that conditioning the goal image on the current observation significantly improves navigation performance, for instance by facilitating estimation of the relative pose between the robot and the goal [46] However, standard methods that concatenate the observation and goal images along the channel dimension are incompatible with pretrained image encoders. Inspired by the binocular vision architecture of Weinzaepfel et al. [47], we utilize a transformer decoder to extract correspondences between the encoded observation and goal tokens. Based on an ablation study (see Appendix C) we choose an architecture where the decoder alternates between self-attention on the goal tokens and cross-attention on the observation tokens. A key strength of this approach is its ability to learn navigation-relevant cues directly from arbitrary, frozen pretrained embeddings. As illustrated in Fig. 3, our binocular encoder identifies matches between image features despite being trained end-to-end with the rest of the policy, without explicit supervision. We use 4 transformer layers with 4 attention heads.

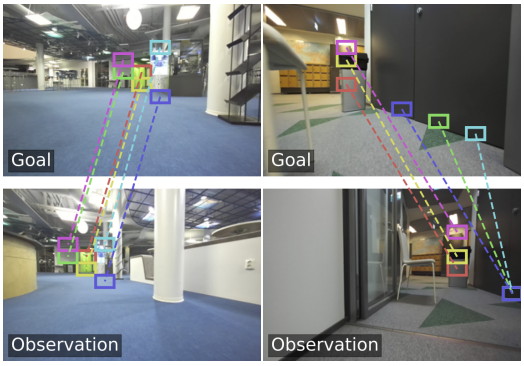

Figure 3: Implicit correspondences of the six highest attention values in the binocular encoder's first cross-attention layer.

Sequence encoder. The sequence of observation and goal tokens is processed by a transformer encoder with non-causal self-attention. Before input into the sequence encoder, the patch tokens of each image are compressed into a one-dimensional vector with a 2D convolution layer followed by flattening, similar to [35]. A learnable state token is added to the sequence, and its corresponding token from the sequence encoder output is passed to an MLP predictor head. This produces the final output, a sequence of waypoints $\mathbf{a}_t$. The sequence encoder consists of 4 layers with 4 attention heads each.

3.3 Learning to navigate in simulation

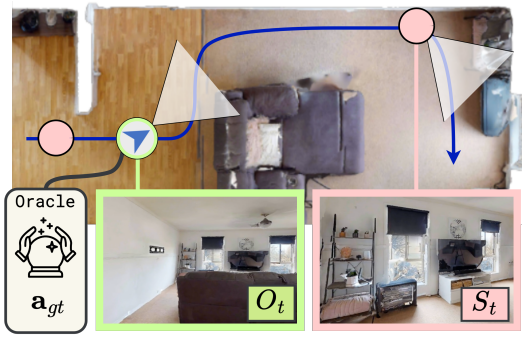

Figure 4: Training data collected from the simulator - oracle actions $\mathbf{a}_{gt}$ that control the agent, agent observation O_t, and subgoal image S_t.

Learning goal-reaching from a shortest-path oracle. We train a learning-based agent to mimic a scripted oracle agent that follows the shortest paths between the episode start and goal locations. The oracle has privileged access to the simulator state and utilizes a proportional controller for path tracking. The student agent is trained as if the oracle was navigating a sequence of subgoals $s \in \mathcal{M}$ along the shortest path. The agent predicts the oracle actions $\mathbf{a}_{gt}$ while only having access to the $P = 6$ latest egocentric camera observations $\mathbf{O}_t$, and the subgoal image S_t captured at the next subgoal pose s_t. The predicted actions are trained to minimize the *mean squared error* loss $\mathcal{L} = MSE(\mathbf{a}_t, \mathbf{a}_{gt})$. The set of subgoals $\mathcal{M}$ is randomly sampled along the shortest path so that for each consecutive subgoal the geodesic distance $d(s_n, s_{n+1}) \in [d_{min}, d_{max}]$, where d_{min}, d_{max} are the minimum and maximum subgoal separation. The purpose of this randomization is to increase the diversity of the subgoals. The subgoal at the time step t is chosen from $\mathcal{M}$ based on the distance from the position of the agent. At each time step, the oracle agent is rolled out for $H = 5$ steps to acquire future actions $\mathbf{a}_{gt}$ for each pair of observation and subgoal. Each triplet $(\mathbf{O}_t, S_t, \mathbf{a}_{gt})$, illustrated in Fig. 4, is saved to disk for use as training data for the student agent.

Data distribution. Naively imitating the oracle agent leads to poor performance, as shown in Sec. 4.4. During deployment, compounding prediction errors lead to covariate shift between training data and actual observations [48]. If a policy has only been trained with shortest-path trajectories, it might not be able to recover back after ending in a state outside the shortest-path distribution. Thus, we employ *DAgger* [49] to diversify the state distribution of the training data. During training data collection, the simulated agent executes the student policy action instead of $\mathbf{a}_{gt}$ with probability $p(\mathbf{a}_t) = \beta^r$, where β is a decay coefficient and r is the number of the current training round. The student action is only executed if it does not lead to a collision.

Simulator setup. Training was carried out in the Habitat [50] simulator with the train split of the HM3D environments and the PointNav route dataset [51]. The dataset consists of 800 individual indoor scenes with $10,000$ routes per scene. Training routes were sampled with the following criteria: *1)* The goal is reachable by a robot with the specified radius. *2)* The agent successfully reaches the goal in 500 steps. *3)* No collisions occur during navigation. The movement of the agent was simulated using kinematic control [15], and the agent was 'teleported' between poses acquired by Euler integration of the commanded velocities. Please refer to Appendix A.1 for the full details about simulator setup and training.

3.4 Learning from real-world data

The real-world-data version of FAINT was trained with the publicly available topological navigation datasets, specifically RECON [52], GoStanford [53], SACSoN [54], SCAND [55], and Tartan-Drive [56]. The synthetic part of GoStanford was omitted to avoid mixing the real and synthetic data. Trajectories, when sampled at 4 Hz, have $\sim 1.2M$ image frames. We follow the training procedure described by Shah et al. [8] with the difference that we omit the temporal distance prediction. To produce training data, pairs of observation and goal images are sampled from the dataset trajectories. After sampling a sequence of observations $\mathbf{O}_t$, a goal image S_t is picked randomly from the same trajectory, $[l_{min}, \ldots, l_{max}]$ frames in the future from t, similar to hindsight relabeling [57]. The H future poses $\mathbf{a}_{gt}$ relative to the current pose of the robot are then used as action labels for training. To enable learning across data from heterogeneous robots, the waypoints are normalized by the average waypoint distance of each dataset [7]. Where applicable, we used the same parameter values as with simulated training. Please see Appendix A.2 for the complete training details.

(a) Open space (b) Tight maneuver (c) Clutter

Figure 5: Example segments from different types of test routes.

4 Experiments

We conducted real-world navigation tests in various indoor environments in experiments designed to answer the following questions.

- **Q1**: How well can a policy perform when trained in simulation instead of with real-world data?
- **Q2**: How does FAINT trained with synthetic data compare to the previous state-of-the-art?
- **Q3**: How do different architectural choices affect the policy's sim-to-real generalization?
- **Q4**: Does the deployment embodiment have to closely match the simulated training embodiment?

4.1 General setup

Hardware. Experiments **Q1.**-**Q3.** were performed on a Turtlebot4 robot with a 110° FOV ZED 2i camera and an Nvidia Jetson Orin AGX, moving at 0.3 m/s. For studying **Q4.**, FAINT was deployed on a custom-built *Agipix*-drone, equipped with a forward-facing 110° FOV USB camera and a Jetson Orin NX, and moving at 0.4 m/s. All computation was performed on board the robots.

Deployment. FAINT was deployed within the PlaceNav [24] framework, which divides navigation into separate goal-reaching and subgoal selection policies. Subgoal selection was performed with a ResNet18 [58] variant of the EigenPlaces [59] place recognition model followed by Bayesian filtering. The subgoal selection and goal-reaching policies run in separate threads, both at 4 Hz. To navigate a route, the robot is first teleoperated to capture map images. A new image is added to the topological map every 5 s, meaning node spacing of $\sim$1.5 m at robot speed of 0.3 m/s. During navigation, the robot follows the sequence of image goals from the beginning to the end of the route.

Evaluation. We evaluated navigation performance by average *success rate* (SR) [5]. Each test route is repeated 3 times with each method and we report the proportion of successful attempts. Similar to [24], a navigation episode is considered successful if the navigation system's subgoal selection module localizes to the last image of the topological map. An episode is unsuccessful if the robot collides with the environment or gets lost in such a way that it cannot return to the test route.

Test environments. The experiments were carried out in various indoor environments including a real apartment, offices, public spaces on a university campus, and a decommissioned nuclear fallout shelter. The tests were limited to indoor environments to reduce the domain gap between the deployment and the simulated training environments. The test routes had different features relevant to navigation, illustrated in Figures 5 and 6. The lengths of the test routes ranged from 5 to 25 meters. For more details about the test route types and route sampling procedure, see Appendix B.1

4.2 Synthetic vs. real training data

To answer **Q1.**, we trained versions of FAINT with different amounts of real and synthetic data, described in detail in Appendix A, and compared the models' performances in real-world experiments.

Results. Table 1 shows the experiment results divided by route category. When training with the same number of samples, real-world data produces better navigation performance than the synthetic data. On the 'open space'-routes, the performance is similar, but on the more challenging routes

Table 1: **FAINT** trained with **real vs. simulated** data, *success rates* (↑) over 3 repetitions of 7 routes per category.

Dataset	Samples	Open space	Tight maneuver	Clutter	Total
		$n = 21$	21	21	63
Real	$1.2M$	0.43	0.52	0.38	0.44
$\text{Sim}_{10\%}$	$1.2M$	0.57	0.05	0.05	0.22
Sim	$12.0M$	**0.86**	**0.62**	**0.76**	**0.75**

the $Sim10\%$ policy often collides as result of cutting too close to obstacles. A 10-fold increase in the amount of synthetically generated data, however, leads to a drastic performance increase. FAINT_{Sim} outperforms the other two by a wide margin in all categories. It is able to perform complex maneuvers to *e.g.* go around obstacles, and reach goal images not within the immediate camera view. Some of the performance gap between $Real$ and Sim may be explained by dataset size—FAINT_{Real} trained on 12M samples might match FAINT_{Sim}. However, scaling simulated data is essentially free, while real-world data collection is very labor-intensive. We suggest on-policy learning as a more insightful explanation. FAINT_{Sim} trained with behavior cloning (see Sec. 4.4) performs poorly despite being trained with $12M$ samples. Interestingly, FAINT_{Real} exhibits similar failure modes, sometimes getting stuck in feedback loops such as spinning in place [48]. This parallel suggests that the behavior cloning's inability to handle compounding errors also affect real-world policies trained without on-policy corrections. DAgger exposes FAINT_{Sim} to wide distribution of states during training, enabling recovery from such situations. This underscores the potential of simulation for robot learning, as large-scale on-policy learning is impractical in the real-world.

4.3 SOTA comparison

Table 2: **SOTA comparison** *success rates* (↑) over 3 repetitions of 4 routes per category.

Method	Open space	Tight maneuver	Clutter	Illumination change	Total
	$n = 12$	12	12	12	48
NoMAD [9]	0.08	0.08	0.25	0.00	0.10
PlaceNav [24]	0.67	0.25	0.33	0.00	0.31
ViNT [8]	**0.92**	0.25	0.42	0.00	0.40
FAINT (ours)	**0.92**	**0.92**	**0.75**	**1.00**	**0.90**

To study **Q2.**, we compared FAINT_{Sim} with topological navigation methods from previous work (see Appendix B.2), trained with real-world data. We extend the experiment setup from Sec. 4.2 by introducing a new route type with illumination change between map collection and deployment. Additionally, we conducted a controlled study on illumination change in which we captured maps on one easy route under regular indoor lighting (220 lx [60]) and dim lighting (25 lx). During testing, illumination was progressively transitioned between these two levels.

Results of the SOTA comparison are shown in Table 2. NoMAD performs surprisingly poorly - we hypothesize the diffusion-based method to be less robust to variation in cameras and embodiments. FAINT outperforms other methods across every route category. The baselines struggle with segments that require sharp turns, often failing to navigate effectively toward the goal. We hypothesize this is caused by the limited receptive fields of their convolutional goal-conditioning modules. In contrast, FAINT 's binocular encoder allows it to associate observation and goal image features even under wide baseline changes (see Fig. 3). The performance differences were the most drastic on routes with illumination change. The baselines struggle with even moderate changes, while FAINT's performance is consistent across illuminations. With controlled illumination change (Fig. 6), both ViNT and FAINT perform well when the deployment illumination is close to the reference, but as the difference grows, ViNT degrades and fails completely. With the 25 lx map, it only succeeds if the deployment illumination is the same. In contrast, FAINT succeeds under all conditions with both references. This robustness is likely due to FAINT 's use of a pre-trained visual encoder, which helps bridge appearance gaps — including those caused by lighting changes.

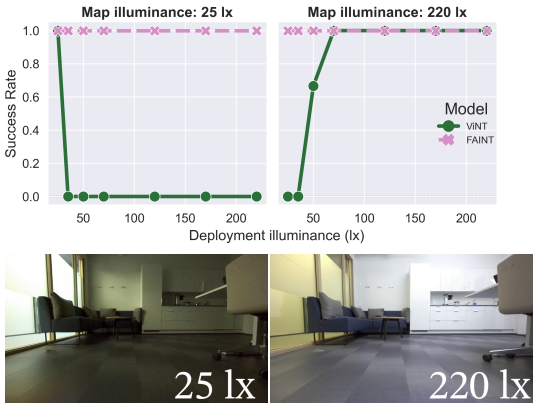

Figure 6: ViNT and FAINT were tested for 3 repetitions under various illumination levels, with two maps captured under 25 lx and 220 lx.

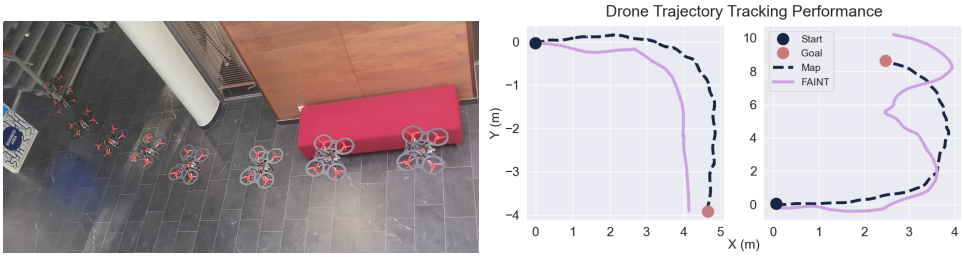

Figure 7: Drone trajectory relative to $\mathcal{M}$ when controlled by FAINT. The drone successfully reached the goal on the left trajectory ($RMSE$ 1.07 m), but missed it on the right ($RMSE$ 1.42 m).

4.4 Simulation-to-real generalization

To address **Q3.** we analyze model design choices' effects on sim2real generalization. We trained models with $12M$ samples from simulation with either behavior cloning (BC) or 10 rounds of DAgger. Evaluation was performed both in simulation on the 2500 routes of the HM3D Val split [51], and in real environments on the Turtlebot. The real-world evaluation followed the protocol from Sec. 4.1, with each method tested over 3 repetitions across 10 routes. See Appendix B.3 for details on the simulator evaluation procedure.

Table 3: **Simulation-to-real** experiment *success rates* (↑). Note that ViNT does not allow freezing the encoder.

Method	Encoder type	Encoder frozen	Mode	Real $n = 30$	Sim 2500
FAINT	Theia CDDSV	✓	BC	0.23	0.87
	Theia CDDSV	✓	DAgger	**0.80**	**0.91**
	Theia CDIV	✓	DAgger	0.60	**0.91**
	EfficientNet-B0	✓	DAgger	0.13	0.79
ViNT [8]	EfficientNet-B0	✘	DAgger	0.40	0.87

Results in Table 3 highlight the role of image encoder pretraining and on-policy data for sim2real transfer. The frozen EfficientNet [61] trained for classification on ImageNet transfers poorly, performing even worse than ViNT with an unfrozen encoder. The two Theia [36] variants distill representations from large models trained for diverse visual tasks, and enable strong real-world performance even without fine-tuning. Models trained with BC and DAgger perform similarly in simulation but differ greatly in real-world performance. We attribute this to the stronger prediction error compounding caused by the sim2real gap and higher non-determinism of the real world. DAgger exposes the policy to a wider state distribution during training, drastically improving real-world performance.

4.5 Cross-embodiment generalization

To study **Q4.**, we trained FAINT with a simulated wheeled robot embodiment and deployed to a real drone without any modifications. The policy controlled the drone's forward velocity, up to 0.4 m/s, and yaw rate at a fixed elevation of $\sim$1.5 m. We tested drone navigation on two 'open space'-type routes. Fig. 7 shows the drone trajectories and tracking metrics compared to the reference routes. These preliminary results indicate that the deployment embodiment does not have to be strictly similar to the one used in training. We leave more thorough analysis of cross-embodiment generalization to future work.

5 Conclusion

This work demonstrated that a simulation-trained visual navigation policy can reach performance comparable to policies trained with real data. We proposed a novel navigation policy architecture that is similar to the previous state-of-the-art, but introduces key modifications that make it suitable for training with both real and synthetic data. Comparison of synthetic and real data trained versions of the policy show that on-policy learning is a major advantage of simulated training, providing robustness to covariate shift. The findings suggest combining off-policy real-world datasets with on-policy corrections from simulation as an interesting avenue for future work.

Limitations

Due to the simulated training environments being limited to indoor scenes and practical considerations related to the test robot platforms, the test domain of our experiments was limited to indoor environments. Additional research will be required to verify the findings in other settings.

The success rates reported in the paper do not represent the methods' absolute performance across all imaginable navigation scenarios. Instead, they measure the relative ranking of the methods on routes where at least one of the methods succeeds.

Acknowledgments

The authors wish to acknowledge CSC – IT Center for Science, Finland, for generous computational resources. We thank the Technology Innovation Institute for financially supporting this work.

References

[1] G. Kahn, P. Abbeel, and S. Levine. BADGR: An Autonomous Self-Supervised Learning-Based Navigation System. *IEEE Robotics and Automation Letters*, 6(2):1312–1319, Apr. 2021. ISSN 2377-3766. 2

[2] M. G. Castro, S. Triest, W. Wang, J. M. Gregory, F. Sanchez, J. G. Rogers, and S. Scherer. How Does It Feel? Self-Supervised Costmap Learning for Off-Road Vehicle Traversability. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 931–938, May 2023. 2

[3] A. Mousavian, A. Toshev, M. Fišer, J. Košecká, A. Wahid, and J. Davidson. Visual Representations for Semantic Target Driven Navigation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8846–8852, May 2019. 2

[4] T. Gervet, S. Chintala, D. Batra, J. Malik, and D. S. Chaplot. Navigating to objects in the real world. *Science Robotics*, 8(79):eadf6991, June 2023. 2, 3

[5] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, and A. R. Zamir. On Evaluation of Embodied Navigation Agents. *arXiv:1807.06757 [cs]*, July 2018. 2, 6, 16

[6] D. Shah, B. Eysenbach, G. Kahn, N. Rhinehart, and S. Levine. ViNG: Learning Open-World Navigation with Visual Goals. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13215–13222. IEEE Press, 2021. 2

[7] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine. GNM: A General Navigation Model to Drive Any Robot. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7226–7233, May 2023. 2, 3, 5, 16

[8] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine. ViNT: A Large-Scale, Multi-Task Visual Navigation Backbone with Cross-Robot Generalization. In *Conference on Robot Learning*. PMLR, Aug. 2023. 2, 3, 4, 5, 7, 8, 16

[9] A. Sridhar, D. Shah, C. Glossop, and S. Levine. NoMaD: Goal Masked Diffusion Policies for Navigation and Exploration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 63–70, May 2024. 2, 4, 7, 16

[10] J. H. Yang, C. Glossop, A. Bhorkar, D. Shah, Q. Vuong, C. Finn, D. Sadigh, and S. Levine. Pushing the Limits of Cross-Embodiment Learning for Manipulation and Navigation. In *Robotics: Science and Systems XX*, volume 20, July 2024. ISBN 979-8-9902848-0-7. 2

[11] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza. Learning high-speed flight in the wild. *Science Robotics*, 6(59), Oct. 2021. 2

[12] K. Ehsani, T. Gupta, R. Hendrix, J. Salvador, L. Weihs, K.-H. Zeng, K. P. Singh, Y. Kim, W. Han, A. Herrasti, R. Krishna, D. Schwenk, E. VanderBilt, and A. Kembhavi. SPOC: Imitating Shortest Paths in Simulation Enables Effective Navigation and Manipulation in the Real World. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16238–16250, 2024. 2

[13] R. Ramrakhya, E. Undersander, D. Batra, and A. Das. Habitat-Web: Learning Embodied Object-Search Strategies from Human Demonstrations at Scale. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5163–5173, June 2022. ISSN: 2575-7075. 2

[14] D. Hoeller, L. Wellhausen, F. Farshidian, and M. Hutter. Learning a State Representation and Navigation in Cluttered and Dynamic Environments. *IEEE Robotics and Automation Letters*, 6(3):5081–5088, July 2021. ISSN 2377-3766. 2

[15] J. Truong, M. Rudolph, N. H. Yokoyama, S. Chernova, D. Batra, and A. Rai. Rethinking Sim2Real: Lower Fidelity Simulation Leads to Higher Sim2Real Transfer in Navigation. In *Conference on Robot Learning*, pages 859–870. PMLR, Mar. 2023. 2, 3, 5

[16] M. Kulkarni and K. Alexis. Reinforcement Learning for Collision-free Flight Exploiting Deep Collision Encoding. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15781–15788, May 2024. 2

[17] I. Geles, L. Bauersfeld, A. Romero, J. Xing, and D. Scaramuzza. Demonstrating Agile Flight from Pixels without State Estimation. In *Robotics: Science and Systems XX*, July 2024. ISBN 979-8-9902848-0-7. 2

[18] Y. Hu, F. Lin, P. Sheng, C. Wen, J. You, and Y. Gao. Data Scaling Laws in Imitation Learning for Robotic Manipulation. In *1st Workshop on X-Embodiment Robot Learning*, Nov. 2024. 2

[19] S. Höfer, K. Bekris, A. Handa, J. C. Gamboa, M. Mozifian, F. Golemo, C. Atkeson, D. Fox, K. Goldberg, J. Leonard, C. Karen Liu, J. Peters, S. Song, P. Welinder, and M. White. Sim2Real in Robotics and Automation: Applications and Challenges. *IEEE Transactions on Automation Science and Engineering*, 18(2):398–400, Apr. 2021. ISSN 1558-3783. 2

[20] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, and D. Batra. Sim2Real Predictivity: Does Evaluation in Simulation Predict Real-World Performance? *IEEE Robotics and Automation Letters*, 5(4):6670–6677, Oct. 2020. ISSN 2377-3766. 2, 3

[21] A. Sadek, G. Bono, B. Chidlovskii, and C. Wolf. An in-depth experimental study of sensor usage and visual reasoning of robots navigating in real environments. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 9425–9431, May 2022. 2

[22] S. Silwal, K. Yadav, T. Wu, J. Vakil, A. Majumdar, S. Arnaud, C. Chen, V.-P. Berges, D. Batra, A. Rajeswaran, M. Kalakrishnan, F. Meier, and O. Maksymets. What Do We Learn from a Large-Scale Study of Pre-Trained Visual Representations in Sim and Real Environments? In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 17515–17521, May 2024. 2, 3

[23] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy. End-to-End Driving Via Conditional Imitation Learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4693–4700, May 2018. 2

[24] L. Suomela, J. Kalliola, H. Edelman, and J.-K. Kämäräinen. PlaceNav: Topological Navigation through Place Recognition. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5205–5213, May 2024. 2, 3, 6, 7, 15, 16

[25] C. Masone and B. Caputo. A Survey on Deep Visual Place Recognition. *IEEE Access*, 9: 19516–19547, 2021. ISSN 2169-3536. 2, 3

[26] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3357–3364, May 2017. 2

[27] F. Sadeghi and S. Levine. CAD2RL: Real Single-Image Flight Without a Single Real Image. In *Robotics: Science and Systems XIII*, July 2017. ISBN 978-0-9923747-3-0. 2

[28] X. Meng, N. Ratliff, Y. Xiang, and D. Fox. Scaling Local Control to Large-Scale Topological Navigation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 672–678, May 2020. ISSN: 2577-087X. 2

[29] J. Truong, A. Zitkovich, S. Chernova, D. Batra, T. Zhang, J. Tan, and W. Yu. IndoorSim-to-OutdoorReal: Learning to Navigate Outdoors Without Any Outdoor Experience. *IEEE Robotics and Automation Letters*, 9(5):4798–4805, May 2024. ISSN 2377-3766. 2

[30] E. Kaufmann, A. Loquercio, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza. Deep Drone Acrobatics. In *Robotics: Science and Systems XVI*, volume 16, July 2020. ISBN 978-0-9923747-6-1. 2

[31] K.-H. Zeng, Z. Zhang, K. Ehsani, R. Hendrix, J. Salvador, A. Herrasti, R. Girshick, A. Kembhavi, and L. Weihs. PoliFormer: Scaling On-Policy RL with Transformers Results in Masterful Navigators. In *Conference on Robot Learning*. PMLR, Sept. 2024. 2

[32] J. Hu, R. Hendrix, A. Farhadi, A. Kembhavi, R. Martín-Martín, P. Stone, K.-H. Zeng, and K. Ehsani. FLaRe: Achieving Masterful and Adaptive Robot Policies with Large-Scale Reinforcement Learning Fine-Tuning. In *1st Workshop on X-Embodiment Robot Learning*, Nov. 2024. 2

[33] A. Eftekhar, L. Weihs, R. Hendrix, E. Caglar, J. Salvador, A. Herrasti, W. Han, E. VanderBil, A. Kembhavi, A. Farhadi, R. Krishna, K. Ehsani, and K.-H. Zeng. The One RING: a Robotic Indoor Navigation Generalist, Dec. 2024. arXiv:2412.14401 [cs]. 2

[34] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer. Sigmoid Loss for Language Image Pre-Training. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11941–11952, Oct. 2023. ISSN: 2380-7504. 3

[35] A. Majumdar, K. Yadav, S. Arnaud, Y. J. Ma, C. Chen, S. Silwal, A. Jain, V.-P. Berges, T. Wu, J. Vakil, P. Abbeel, J. Malik, D. Batra, Y. Lin, O. Maksymets, A. Rajeswaran, and F. Meier. Where are we in the search for an artificial visual cortex for embodied intelligence? In *Advances in Neural Information Processing Systems*, volume 37, pages 655–677, May 2024. 3, 4

[36] J. Shang, K. Schmeckpeper, B. B. May, M. V. Minniti, T. Kelestemur, D. Watkins, and L. Herlant. Theia: Distilling Diverse Vision Foundation Models for Robot Learning. In *Conference on Robot Learning*. PMLR, Sept. 2024. 3, 4, 8, 16

[37] A. Sadek, G. Bono, B. Chidlovskii, A. Baskurt, and C. Wolf. Multi-Object Navigation in real environments using hybrid policies. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4085–4091, May 2023. 3

[38] N. Savinov, A. Dosovitskiy, and V. Koltun. Semi-parametric topological memory for navigation. In *International Conference on Learning Representations (ICLR)*, 2018. 3

[39] G. Berton, C. Masone, and B. Caputo. Rethinking Visual Geo-localization for Large-Scale Applications. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4868–4878, June 2022. ISBN 978-1-6654-6946-3. 3

[40] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In M. Meila and T. Zhang, editors, *International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 2021. 4

[41] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. HAZIZA, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. 4

[42] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao. Depth Anything: Unleashing the Power of Large-Scale Unlabeled Data. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10371–10381, 2024. 4

[43] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollar, and R. Girshick. Segment Anything. In *2023 IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023. 4

[44] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*, 2021. 4

[45] X. Sun, P. Chen, J. Fan, J. Chen, T. Li, and M. Tan. FGPrompt: Fine-grained Goal Prompting for Image-goal Navigation. *Advances in Neural Information Processing Systems*, 36:12054–12073, Dec. 2023. 4

[46] G. Bono, L. Antsfeld, B. Chidlovskii, P. Weinzaepfel, and C. Wolf. End-to-End (Instance)-Image Goal Navigation through Correspondence as an Emergent Phenomenon. In *International Conference on Learning Representations (ICLR)*, Oct. 2023. 4

[47] P. Weinzaepfel, V. Leroy, T. Lucas, R. Brégier, Y. Cabon, V. Arora, L. Antsfeld, B. Chidlovskii, G. Csurka, and J. Revaud. CroCo: Self-Supervised Pre-training for 3D Vision Tasks by Cross-View Completion. In *Advances in Neural Information Processing Systems*, volume 34, Oct. 2022. 4, 16

[48] J. Spencer, S. Choudhury, A. Venkatraman, B. Ziebart, and J. A. Bagnell. Feedback in Imitation Learning: The Three Regimes of Covariate Shift, Feb. 2021. arXiv:2102.02872 [cs, stat]. 5, 7

[49] S. Ross, G. Gordon, and D. Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 627–635. JMLR Workshop and Conference Proceedings, June 2011. 5

[50] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A Platform for Embodied AI Research. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9338–9346, Seoul, Korea (South), Oct. 2019. IEEE. ISBN 978-1-72814-803-8. 5

[51] S. K. Ramakrishnan, A. Gokaslan, E. Wijmans, O. Maksymets, A. Clegg, J. Turner, E. Undersander, W. Galuba, A. Westbury, A. Chang, M. Savva, Y. Zhao, and D. Batra. Habitat-Matterport 3D Dataset (HM3D): 1000 Large-scale 3D Environments for Embodied AI. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 1, Dec. 2021. 5, 8, 16

[52] D. Shah, B. Eysenbach, N. Rhinehart, and S. Levine. Rapid Exploration for Open-World Navigation with Latent Goal Models. In *Conference on Robot Learning*, pages 674–684. PMLR, Jan. 2022. ISSN: 2640-3498. 5

[53] N. Hirose, F. Xia, R. Martín-Martín, A. Sadeghian, and S. Savarese. Deep Visual MPC-Policy Learning for Navigation. *IEEE Robotics and Automation Letters*, 4(4):3184–3191, Oct. 2019. ISSN 2377-3766. 5

[54] N. Hirose, D. Shah, A. Sridhar, and S. Levine. SACSoN: Scalable Autonomous Control for Social Navigation. *IEEE Robotics and Automation Letters*, 9(1):49–56, Jan. 2024. ISSN 2377-3766. 5

[55] H. Karnan, A. Nair, X. Xiao, G. Warnell, S. Pirk, A. Toshev, J. Hart, J. Biswas, and P. Stone. Socially CompliAnt Navigation Dataset (SCAND): A Large-Scale Dataset of Demonstrations for Social Navigation. *IEEE Robotics and Automation Letters*, 7(4):11807–11814, Oct. 2022. ISSN 2377-3766. 5

[56] S. Triest, M. Sivaprakasam, S. J. Wang, W. Wang, A. M. Johnson, and S. Scherer. TartanDrive: A Large-Scale Dataset for Learning Off-Road Dynamics Models. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2546–2552, May 2022. 5

[57] D. Ghosh, A. Gupta, A. Reddy, J. Fu, C. M. Devin, B. Eysenbach, and S. Levine. Learning to Reach Goals via Iterated Supervised Learning. In *International Conference on Learning Representations (ICLR)*, 2021. 5

[58] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 6

[59] G. Berton, G. Trivigno, B. Caputo, and C. Masone. EigenPlaces: Training Viewpoint Robust Models for Visual Place Recognition. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11080–11090, 2023. 6

[60] D. L. DiLaura, G. R. Steffy, K. W. Houser, and R. G. Mistrick. *The lighting handbook: reference and application*. Illuminating Engineering Society of North America, New York, NY, 10th ed. edition, 2011. ISBN 978-0-87995-241-9. 7

[61] M. Tan and Q. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *International Conference on Machine Learning (ICML)*, pages 6105–6114. PMLR, May 2019. 8

[62] R. Doshi, H. R. Walke, O. Mees, S. Dasari, and S. Levine. Scaling Cross-Embodied Learning: One Policy for Manipulation, Navigation, Locomotion and Aviation. In *Conference on Robot Learning*. PMLR, Sept. 2024. 15

A Training Details

A.1 Synthetic data

Simulator setup. Table A.1.1 presents the simulator setup details. The robot radius was set to 0.1 m, which is slightly less than the radius of the Turtlebot4 platform, 0.18 m. In path planning the oracle maintained an additional margin of 0.2 m to any obstacles. We found that this produces sufficient collision avoidance behavior without discarding too many routes due to narrow segments. The simulated camera's *field of view* (FOV) and aspect ratio were matched with the deployment camera.

Parameter	Value
Simulator step	0.25 s
Agent radius	0.1 m
d_{min}, d_{max}	0.5 m, 3.0 m
Camera FoV	$110°$
Camera resolution	224×126

Table A.1.1: Simulator setup.

Training. Table A.1.2 presents the training details. All the models were trained for 10 rounds of DAgger. The exact number of training samples and training epochs varied by experiment. The image encoder was frozen during training. All models were trained on a single node, with distributed data parallelism (DDP) in both data collection and model training. Training takes 24 h when collecting 10,000 trajectories with $1.2M$ examples per DAgger round.

Parameter	Value
Past observations (P)	6
Waypoints predicted (H)	5
Training method	10 rounds of DAgger
DAgger decay coeff (β)	0.8
Batch size	512
Loss function	Mean squared error (MSE)
Optimizer	AdamW
Initial learning rate	2×10^{-4}
Learning rate schedule	Cosine decay
GPUs	4 × Tesla V100
Parallel simulations per GPU	16
Training time	24 h w/ $1.2M$ images per round
Image resolution	224×224
Image augmentations	Color jitter, posterization

Table A.1.2: Simulated training details.

We trained two different versions of FAINT with synthetic data:

FAINT$_{\text{Sim 10\%}}$: Trained for 10 rounds of DAgger with 3 epochs per round with total dataset size of $1.2M$ samples. The dataset size is equal to the dataset used for training FAINT$_{\text{Real}}$. Training followed the procedure described in Sec. 3.3.

FAINT$_{\text{Sim}}$: Trained for 10 rounds of DAgger, with only 1 epoch per round. Total dataset size of $12M$ samples, which is $10\times$ more data than FAINT$_{\text{Real}}$ or FAINT$_{\text{Sim 10\%}}$.

A.2 Real-world-data

Table A.2.1 presents the training parameters for the real datasets. The same training setup used with the simulated data (A.1) was used with real-world data training when applicable.

Parameter	Value
Past observations (P)	6
Waypoints predicted (H)	5
Training method	Behavior cloning
Training epochs	30
Dataset size	1.2M images
Batch size	256
Loss function	Mean squared error (MSE)
Optimizer	AdamW
Initial learning rate	5×10^{-4}
Learning rate schedule	4-epoch warmup, cosine decay
$[l_{min}, l_{max}]$	[0, 10]
GPUs	4 × Tesla V100
Training time	48 h
Image resolution	224×224
Image augmentations	Color jitter, posterization

Table A.2.1: Real data training details.

We trained only one version of FAINT with real-world data.

FAINT$_{\text{Real}}$: Trained for 30 epochs with $1.2M$ examples from the datasets described in Sec. 3.4.

B Experiment setup

B.1 Real-world test routes

We performed experiments on several types of test routes, with different characteristics relevant for navigation. In open space routes, there is ample clearance during *e.g.* turns, and the main challenge is to execute the correct action at the correct time. Routes with tight maneuvers contain turns into narrow gaps and $> 90°$ turns such as looping around a pillar. In cluttered scenes, the robot has to perform multiple sequential turns with tight margins in order to avoid collision. Finally, we investigate cases where the scene illumination changes between the collection of reference images and robot deployment, as previous work [24] has reported this to be a difficult task. The lengths of the test routes ranged from 5 to 25 meters.

Selecting the evaluation routes requires careful consideration. Routes that are trivially easy for all methods or those where all methods fail provide little insight into their relative performance. Our focus is on ranking methods rather than assessing absolute performance across all possible navigation routes. Trivial test cases shift methods' average success rates uniformly, without affecting the relative ordering. To create informative test cases, we find routes where at least one method fails and at least one succeeds once. Consequently, different experiments with different methods require different test routes. This means that absolute success rates may vary across experiments and are not directly comparable: main result is in the relative ranking of the methods.

B.2 SOTA baselines

For the SOTA comparison (Sec. 4.3) baselines, we only considered methods that can run onboard the robot, which rules out larger models such as CrossFormer [62]. The baseline models were deployed with the author-provided model checkpoints and using the original subgoal selection methods.

ViNT [8]: An embodiment-agnostic transformer-based navigation model, trained with 80 h of real-world navigation data from various robots. The training datasets are only partly available publicly. The subgoals are chosen by temporal distance prediction.

NoMAD [9]: Same architecture and training data as ViNT, except that the policy head is replaced by a diffusion network enabling multimodal predictions instead of point estimates.

PlaceNav [24]: Utilizes place recognition with Bayesian filtering for subgoal selection. Waypoint prediction is performed with the GNM model [7], also trained with real-world data.

B.3 Simulator evaluation

The simulator evaluation for Sec. 4.4 simulation-to-real experiment was performed on the 2500 routes of the HM3D Val split [51]. An episode was considered successful if the agent arrived within 0.4 m of the goal. If the agent fails to do so within 500 time steps or collides with the environment, the episode was considered a failure

C Binocular encoder ablation experiment.

This appendix presents the results of an investigation into the different methods of conditioning the goal image by the observation image. We trained different versions of FAINT with the same procedure as Sec. 4.4. All versions except the EarlyConv utilize the Theia-Tiny CDDSV encoder [36]. Table C.0.1 reports the success rate (SR) and success weighed by path length (SPL) [5] from simulator evaluation with the HM3D validation set [51].

NoBlock. The goal encoding is not conditioned with the observation. The goal encoding from the PVR is concatenated to the sequence encoder input as is.

EarlyConv. This model is equivalent to ViNT [8]. Utilizes an EfficientNet-B0 encoder, since this conditioning method does not accommodate use of PVR's. The observation and goal images are concatenated channel-wise before input to a 6-channel EfficientNet-B0.

LateConv. The PVR encoded goal and observation feature maps are concatenated channel-wise, and processed by a 3-layer CNN.

CatBlock. The observation and goal tokens are concatenated into a single sequence, which is processed with a transformer encoder with self-attention only. The tokens corresponding to the goal are picked for the output. The transformer has 4 layers with 4 attention heads.

CrossBlock. A transformer that alternates between self-attention on the goal tokens and cross-attention on the observation tokens. The transformer has 4 layers with 4 attention heads.

Table C.0.1: **Binocular encoder** ablation *success rates* (↑) and *success rates weighed by path length* (↑).

Binocular encoder	$\mathbf{SR}_{sim}$	$\mathbf{SPL}_{sim}$
NoBlock	0.75	0.71
LateConv	0.85	0.80
CatBlock	0.84	0.80
EarlyConv	0.87	0.83
CrossBlock	**0.90**	**0.86**

The CrossBlock architecture inspired by [47] achieves the best performance as measured by simulator SR and SPL, and it was chosen as the binocular encoder archictecture for FAINT. Both EarlyConv (ViNT) and CrossBlock (FAINT) were additionally evaluated in real-world experiments of Sec. 4.4.