2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

StereoNavNet: Learning to Navigate using Stereo Cameras with Auxiliary Occupancy Voxels

Hongyu Li[1], Taşkın Padır[2], and Huaizu Jiang[3]

***Abstract*— Visual navigation has received significant attention recently. Most of the prior works focus on predicting navigation actions based on semantic features extracted from visual encoders. However, these approaches often rely on large datasets and exhibit limited generalizability. In contrast, our approach draws inspiration from traditional navigation planners that operate on geometric representations, such as occupancy maps. We propose StereoNavNet (SNN), a novel visual navigation approach employing a modular learning framework comprising perception and policy modules. Within the perception module, we estimate an auxiliary 3D voxel occupancy grid from stereo RGB images and extract geometric features from it. These features, along with user-defined goals, are utilized by the policy module to predict navigation actions. Through extensive empirical evaluation, we demonstrate that SNN outperforms baseline approaches in terms of success rates, success weighted by path length, and navigation error. Furthermore, SNN exhibits better generalizability, characterized by maintaining leading performance when navigating across previously unseen environments.**

I. Introduction

The intelligent navigation capability is essential for a robot to be integrated into our daily lives. During the navigation, a robot needs to execute a sequence of actions to reach the desired goal, which may be a spatial coordinate, a specific object, or even a description in natural language [1, 2]. It is a challenging problem in robotics as the robot needs to move swiftly while effectively perceiving and avoiding unforeseeable obstacles. Furthermore, the accurate perception and avoidance of obstacles often rely on data from expensive sensors.

To extract scene semantics and improve affordability, visual navigation works focus on developing solutions that rely solely on low-cost vision sensors [3–10], for instance, RGB cameras. These approaches utilize deep learning-based visual encoders [11–14] to extract a feature vector from visual input, which implicitly encapsulates semantic information such as obstacles and can be later used for obstacle avoidance. We refer to this class of methods as *semantic* (Fig. 1 (a)). However, these semantic methods typically require large datasets or a long time for training and are susceptible to domain gaps when deployed in unseen environments.

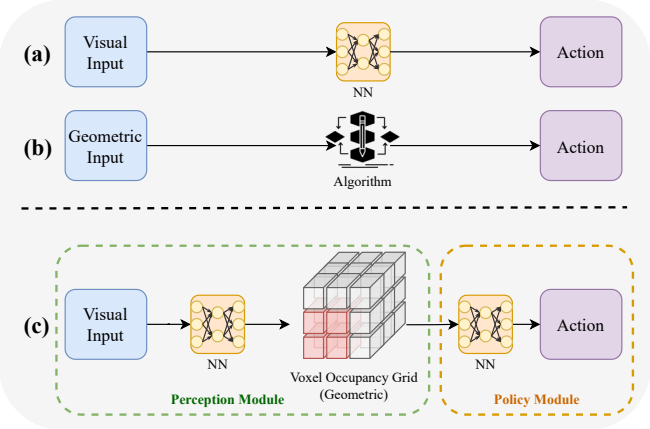

Fig. 1: **A high-level comparison.** Unlike the conventional visual navigation approach (a), where visual input is typically encoded into semantic features using a visual encoder for subsequent action prediction, we introduce a novel visual navigation network (c) inspired by traditional navigation approaches (b). We extract an auxiliary voxel occupancy grid from semantic features using a neural network (NN) and derive geometric obstacle features from it, which our policy network is conditioned on.

Traditional navigation planners, such as Dynamic Window Approach (DWA) [15] and Time Elastic Bands (TEB) [16, 17], operate on geometric 2D/3D representations like a grid map or voxel occupancy grid. In these representations, free spaces and obstacles are clearly differentiated, often as zeros and ones in the 2D/3D grid (zeros mean free space). Therefore, we refer to this paradigm as *geometric* (Fig. 1 (b)). Geometric planners exhibit appealing attributes compared with their semantic counterparts. For instance, such methods rarely suffer from the domain gap between various scene appearances as they do not directly take vision sensory data as input, which is of particular interest in deploying an autonomous robot in practice. However, they have limited learning capacity and require accurate geometric information, which is often challenging to obtain during real-world deployment.

Can we combine the best of the two paradigms? Deep neural networks are suitable for processing raw sensory data, *e.g.*, RGB images. At the same time, a navigation policy learned from geometric scene representations, instead of visual input directly, may lead to higher learning efficiency

This research is supported by the National Science Foundation under Award Number IIS-2310254. Taşkın Padır holds concurrent appointments as a Professor of Electrical and Computer Engineering at Northeastern University and as an Amazon Scholar. This paper describes work performed at Northeastern University and is not associated with Amazon.

[1]Department of Computer Science, Brown University, Providence, RI. email: hli230@cs.brown.edu
[2]Institute for Experiential Robotics, Northeastern University, Boston, MA. email: t.padir@northeastern.edu
[3]Khoury College of Computer Sciences, Northeastern University, Boston, MA. email: h.jiang@northeastern.edu

and improved robustness to the domain gap [18, 19]. Several recent studies [19–24] have explored *hybrid* approaches, leveraging both semantic and geometric features. These approaches mostly follow the sim-to-real paradigm, relying on *ground-truth depth maps* provided by simulated RGB-D sensors. Nevertheless, Bansal et al. [25] and Gervet et al. [24] observed that policies trained using this paradigm often encounter significant domain gaps in real-world deployment, primarily due to noisy depth measurements.

In this paper, we introduce StereoNavNet (SNN), a hybrid end-to-end visual navigation approach that encapsulates two modules: a perception module and a policy module, as shown in Fig. 1 (c). *Without having access to any ground-truth depth information*, the perception module takes raw sensory data from stereo RGB cameras as input and outputs a voxel occupancy grid, which is a pure geometric representation of the scene. It is then fed into the policy module, which produces a velocity command.

Such a *modular* design endows our approach's desired properties for visual navigation. First, it supports flexible training schedules. We begin by training the perception and policy modules independently through modular learning [24, 26–28]. Once both modules are trained to converge, we proceed to optimize the entire pipeline in an end-to-end fashion, which enables the backpropagation of perception errors to the policy end, compensating for the inevitable perception noises. Second, our policy module, conditioning only on geometric features, is more robust to the domain gap between training and testing data.

To confirm the effectiveness of our approach, we perform experiments in the OmniGibson framework [29] built on top of the NVIDIA Isaac Sim™ simulator, which provides realistic rendering and physics simulation. We compare against a semantic approach based on ResNet [11], hybrid baselines using monocular and stereo depth estimation models [30, 31], and a geometric method using ground-truth depth information serving as the upper bound. To evaluate our method, we collect an expert demonstration dataset for PointNav [1] task, consisting of 100 random trajectories across five environments (Fig. 3) using a privileged expert [32–34] equipped with access to global ground-truth information. We assess our method's robustness to the domain gap by validating it across seven novel environments. Our experiments show that our model achieves a higher success rate compared to the semantic approach and generalizes well in previously unseen environments, with a 35% success rate in novel environments versus the semantic baseline's 21%.

In summary, our main contributions are:

1) We propose a novel visual navigation network encapsulating intermediate 3D geometric representation. Our approach is free of ground-truth depth map assumptions and relies purely on RGB images.
2) We present extensive experiment results showing SNN outperforming other baseline methods in terms of success rates, success weighted by path length, and navigation error in both seen and novel environments.

II. Related Works

There are two relevant fields to our paper: 1) visual navigation, the task we are solving, and 2) stereo vision, the approach we use to obtain the intermediate geometric representation.

A. Visual Navigation

Visual navigation is a type of navigation task that utilizes only visual input. Depending on the various goals, different types of visual navigation tasks exist [35]. For example, PointNav focuses on navigating to a spatial coordinate. ObjectNav aims to find a type of object indoors, and ImageNav aims to locate the place where the goal image is taken.

We focus on the PointNav task in this paper since it is fundamental, *i.e.* the obstacle avoidance ability required in this task is also needed by other tasks. In this paper, we focus on extracting the geometry representation for obstacle avoidance. For other semantic navigation tasks, such as the ImageNav or the ObjectNav, it may require fusions with semantic information [36]. Notably, Gervet et al. [24] showed that semantic navigation tasks can be reduced to PointNav by proposing a module learning approach that transforms semantic goals into spatial goal coordinates.

Starting from the early works [3, 4, 6, 37], a dominant paradigm is to predict actions using the semantic features extracted by a visual encoder [11–14]. More recently, efforts have been directed towards refining learning strategies. For instance, Sorokin et al. [34] achieve higher learning efficiency through privileged learning in an abstract world. Kahn et al. [7] control the robot to avoid obstacles by minimizing the predicted human engagement probability. Bansal et al. [25] and Tolani et al. [9] predict waypoints and use an optimal controller to generate actions. Nevertheless, relying solely on visual encoders necessitates large datasets and often results in limited generalizability.

Similar to ours, several works [19–24] also present visual navigation solutions that condition on geometric representations. Notably, DD-PPO [23] has achieved near-perfect performance on PointNav in Habitat-Sim benchmark [35]. However, these works rely on the assumption of ground-truth depth information obtained by RGB-D sensors. Bansal et al. [25] observed that the RGB-D approach has a significantly larger domain gap and worse performance than RGB counterparts due to noisy depth measurements in the real world. Similarly, Gervet et al. [24] found navigation errors in the real world are mainly caused by depth sensor errors. This is primarily due to the limitations in the field of view, object size, material reflections, and lighting conditions [24, 25, 38, 39]. In light of these limitations, we assume no ground-truth depth information is provided and choose the stereo RGB sensor as the research platform.

B. Stereo Vision

Stereo vision has been extensively used in robotics, enabling us to estimate depth by measuring the disparity between stereo images. It relies on the principle of triangulation, mimicking the human visual system. Classical

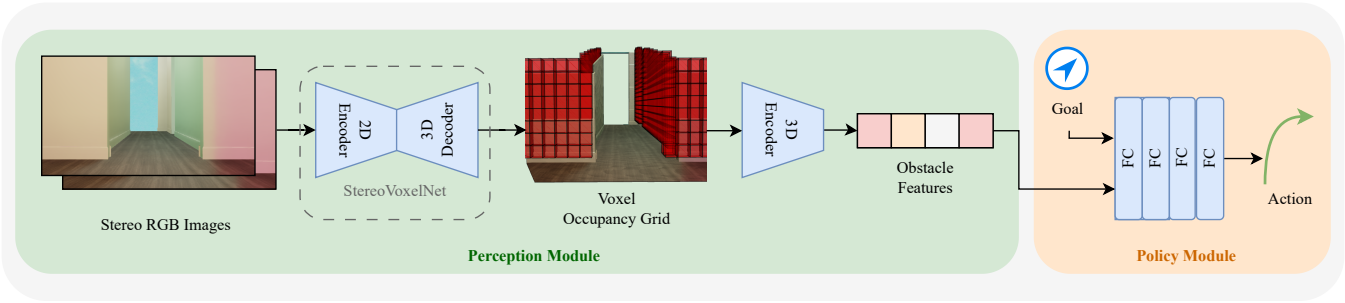

Fig. 2: **The network design of StereoNavNet.** We propose to extract the occupancy features from explicit geometry using the voxel occupancy grid and the curvature samples. The extracted features are used to predict an action using a four-layer MLP.

methods like semi-global matching (SGM) [40] have been widely used in robot navigation [21, 33, 41]. SGM aggregates matching costs over multiple directions and disparities and calculates the disparity for each pixel by minimizing a global energy function. Even though these classical methods have the advantage in speed, they often demonstrate suboptimal performance compared with their learning-based counterparts [42, 43]. Furthermore, it is challenging to integrate the non-differentiable classical methods in an end-to-end learning framework.

Like many other tasks in computer vision, deep learning-based approaches have outperformed the classical ones by a large margin [44]. Recent learning-based stereo models focus on building an end-to-end neural network that maps the stereo image pairs into disparity maps. Inspired by traditional approaches, cost volume is commonly used to correlate the feature maps from the left and right images. While the use of learning-based approaches is appealing, their high computational cost renders them impractical for real-time execution on an onboard computer during autonomous navigation [39]. To tackle this challenge, we introduced StereoVoxelNet [39] in the prior work, which offers an efficient deep-learning model for 3D perception. Instead of calculating the disparity map and transforming it into the 3D point cloud, StereoVoxelNet directly generates the 3D voxel occupancy grid from the stereo images. We employ this approach in our visual navigation framework for 3D perception, capitalizing on its efficiency and accuracy.

III. Problem Formulation and Assumptions

In this paper, we address the problem of PointNav [1], in which the robot operates in a static environment and has no prior maps or knowledge of the environment. In this problem, the locomotion problem is often abstracted out, *i.e.*, there are no holes or stairs blocking movement, and the robot can movely freely to any location unless blocked by obstacles. The robot relies solely on egocentric visual input from an onboard camera to perceive the environment. Given a user-defined point goal $g \in \mathbb{R}^3$, the agent generates a trajectory τ of actions $\tau = (a_1, a_2, \cdots, a_{|\tau|})$ to navigate the robot towards the goal without any collisions. $|\tau|$ represents the total steps taken to reach the goal. Each action is a continuous velocity pair $a = (v, \omega)$, where v denotes the linear velocity and ω denotes the angular velocity. We also assume the ground-truth odometry is given, following the PointNav task setting [1], so that the relative transformation between the robot and the goal is always accurate and known.

Since we focus on the sensing aspect in this paper instead of the navigation policy, we verify SNN on the simplest navigation agent, which is purely reactive [1] and Markovian. Our method is model-free, which means that our policy module doesn't construct a map of the environment [36] nor store internal states across time steps using recurrent units [45].

We assume the robot "sees" the world using stereo RGB cameras instead of RGB-D cameras, as discussed in Sec. II-A, to mitigate potential navigation failures stemming from reflections, varying lighting conditions, etc., when transferred to real-world scenarios. We choose the stereo setup as it provides more accurate 3D geometry than monocular [46]. In the experiment section, we provide numerical results on utilizing both monocular model [30] and stereo model [31] and demonstrate the efficacy of the stereo approach.

IV. StereoNavNet

Our main contribution is the proposed StereoNavNet (SNN), which extracts an explicit 3D geometry representation from the visual observation and trains a policy network upon it. SNN consists of two modules, the perception module and the policy module, as shown in Fig. 2. The perception module takes in stereo images as input and outputs occupancy voxels. Subsequently, the occupancy feature and the user-defined goal g are fed into the policy module (Sec. IV-B), which generates the velocity actions a.

A. Perception Module

Our main motivation is to leverage a perception module f_p to extract geometric features from the visual observation. We hypothesize that a policy network trained on the geometric feature could lead to higher performance and a narrower domain gap since it does not directly take raw sensory data as input. Specifically, the perception module aims to extract obstacle features h_o from a pair of stereo images (I_L, I_R), such that $h_o = f_p(I_L, I_R)$. To achieve this goal, we first

Fig. 3: **Training scenes**. We deploy a privileged agent in five scenes from OmniGibson [29] to collect an expert demonstration dataset.

extract an intermediate geometry representation. While there are various choices of 2D/3D representation, in this paper, we concentrate on utilizing the voxel occupancy grid $\mathcal{G}$ as the 3D geometry representation.

Extracting voxel occupancy grid. We define a robot-centric grid $\mathcal{G}$,

$$\mathcal{G} = \{0,1\}^{n_x \times n_y \times n_z}, n_x, n_y, n_z \in \mathbb{Z}^+. \tag{1}$$

Here n_x, n_y, n_z represents the number of voxels on x, y, z axis, respectively, which point towards the left, top, and forward directions. Each voxel is a cubic volume with side length l_v and can be either occupied (denoted as 1) or unoccupied (denoted as 0). Therefore, the physical dimension of the voxel grid is a volume of size $(n_x \times l_v, n_y \times l_v, n_z \times l_v)$.

In order to estimate the voxel grid $\mathcal{G}$, we employ our previous work, StereoVoxelNet [39], which consists of a 2D-3D encoder-decoder structure. We refer readers to the original paper for more technical details.

The perception module is optimized through Intersection of Union (IoU) loss $\mathcal{L}_p$,

$$\mathcal{L}_p = 1 - \frac{|\mathcal{G} \cap \widehat{\mathcal{G}}|}{|\mathcal{G} \cup \widehat{\mathcal{G}}|}, \tag{2}$$

where $\mathcal{G}$ is the ground-truth voxel grid, and $\widehat{\mathcal{G}}$ is the estimated voxel grid.

Extracting geometric feature h_o. We extract the obstacle feature vector h_o from the voxel occupancy grid $\mathcal{G}$ using a set of four 3D convolutional layers with ReLU layers in between. We set the size of h_o to be 256 and configure the layers with channels of $[4, 8, 6, 32]$ respectively, each with a kernel size of 4 and a stride of 2.

B. Policy Module

Our policy module f_o takes the extracted obstacle feature h_o from the perception module and a point goal feature h_g as input and predicts a velocity action $\hat{a}$, such that $\hat{a} = f_o(h_o, h_g)$. The point goal g is represented in the form of $(d, \cos(\theta), \sin(\theta))$ (converted from polar coordinate [23]), where d and θ are the distance and the angle relative to the goal respectively. Before passing the goal g into the policy network, we encoded it into a feature vector $h_g \in \mathbb{R}^{16}$ using a non-linear layer.

To maintain simplicity, we implement the policy module as a simple multilayer perceptron (MLP) in accordance with prior works [3, 7, 22]. The MLP is composed of four fully connected (FC) layers, each of which is followed by a batch normalization (BN) layer and a ReLU activation function. At each time, the MLP predicts a velocity pair $a = (v, \omega)$ that is executed by the robot. We optimize the policy module using behavior cloning with the Mean Squared Error (MSE) loss $\mathcal{L}_o$,

$$\mathcal{L}_o = (a - \hat{a})^2, \tag{3}$$

where a is the ground-truth action from the sampled trajectories in the expert demonstrations, and $\hat{a}$ is the predicted action obtained by $\hat{a} = f_o(h_o, h_g)$.

C. Modular Learning

We optimize SNN using modular learning and find this leads to better navigation performance (Tab. III). We first train the perception module, which produces geometric obstacle features h_o. Then, we feed these geometric obstacle features along with goal features h_g into the policy module to predict a velocity action a. Our experiments show that this design leads to better generalizability when navigating in novel environments. Specifically, we optimize the perception module for 150 epochs, followed by optimizing the policy module for 50 epochs. Finally, we conduct joint fine-tuning of both modules for 300 epochs using the overall loss, which is a weighted sum of perception loss and policy loss $\mathcal{L} = \mathcal{L}_p + \alpha \mathcal{L}_o$ and can be expanded as:

$$\mathcal{L} = 1 - \frac{|\mathcal{G} \cap \widehat{\mathcal{G}}|}{|\mathcal{G} \cup \widehat{\mathcal{G}}|} + \alpha(a - \hat{a})^2, \tag{4}$$

where $\hat{a} = f_o(f_p(I_L, I_R), h_g)$. We empirically tune the weight $\alpha = 0.1$.

V. Experiment

We follow the privileged learning framework [32–34] and collect an expert demonstration dataset using the five environments shown at Fig. 3. Concretely, the privileged agent (expert) has access to the ground-truth state of all obstacles and the global occupancy map. We generate optimal paths using the A* algorithm and utilize the PD controller to follow the planned trajectory. The velocity command produced by the controller at every frame is saved as the expert action.

We adopt a subset of the scenes provided by the OmniGibson framework [29]. We utilize five scenes during data collection and thirteen scenes (including eight novel scenes) during testing (Tab. I). To improve the robustness of policies, we apply domain randomizations to the scenes, including

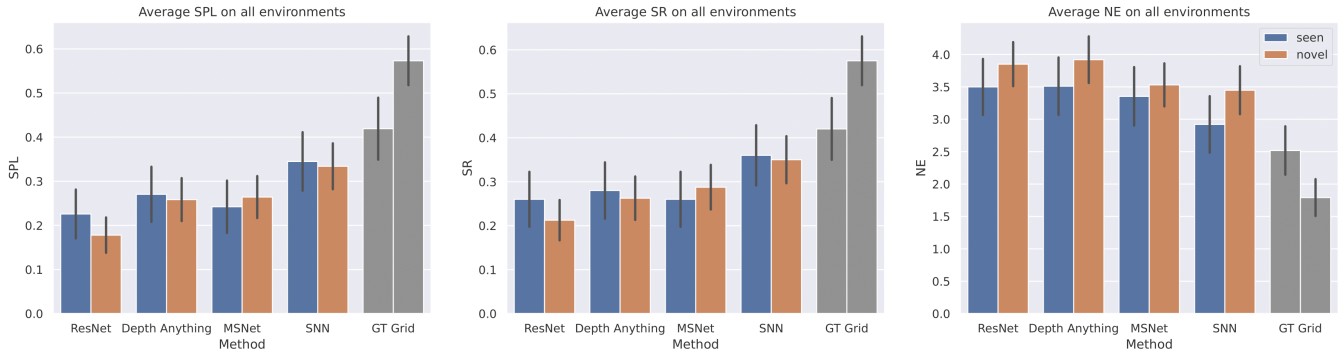

Fig. 4: **Comparison against baseline approaches.** We compare against agents using ResNet, Depth Anything, MobileStereoNet (MSNet), and ground-truth voxel occupancy grid (GT Grid) using three metrics: SPL, SR, and NE. The agent using ground truth only serves as the upper bound for our policy module and is not comparable. The error bars show the standard errors.

dome lights with different RGB colors and intensities. We render and capture the images at the resolution of 672×376.

For each trajectory collection, the initial position of the robot and the goal position are randomly picked in the feasible areas. We randomly sample 100 trajectories in each of the five scenes, creating a dataset with a total of 500 trajectories. The simulation runs at real-time speed, and the collection process typically takes less than 6 hours. The small scale of trajectories collected makes us incomparable with reinforcement learning works like DD-PPO [23], which leverages 80 years of navigation experience.

After collecting the expert demonstration dataset, we train the models using behavior cloning (BC). We implement the neural networks using PyTorch and utilize Adam [47] optimizer with the learning rate at 0.001. We divide the collected demonstration dataset from the five scenes into 80% / 20% split for training and validation and utilize thirteen scenes (including eight novel scenes) for testing purposes. We set the grid size $n_x = n_y = n_z = 64$ to capture details of the scenes while maintaining low computation costs during navigation.

We adopt three metrics to measure navigation performance:

1) Success Rate (SR): portion of successful trials over total runs conducted.
2) Success weighted by Path Length (SPL) [1]: success rate weighted by the actual travel distance compared to optimal path length.
3) Navigation Error (NE): the mean distance to goal when failed (0 when successful).

The qualitative results of navigation performance can be found at Fig. 5 as well as the supplementary video.

A. Comparison against baselines

To evaluate the performance of our proposed method, we conduct experiments in five seen and eight novel environments. To ensure a fair comparison, all of the methods listed below utilize the exact *same policy module* f_o, and the only difference is the perception module f_p. This design follows prior works such as [4, 20, 34], and we only keep the FC layer and remove the LSTM layer for simplicity. We compare our method with four approaches:

1) ResNet: We leverage a ResNet visual encoder pre-trained on ImageNet [48] dataset to produce a semantic obstacle feature h_o from the RGB observation.
2) Depth Anything: We first utilize the current state-of-the-art monocular depth estimation model Depth Anything [30] to estimate a depth map based on the RGB image. Then we pass this depth map into the ResNet encoder to produce the geometric obstacle feature h_o [21, 23, 24].
3) MobileStereoNet: Similar to Depth Anything, we leverage a stereo depth estimation module MobileStereoNet-3D [31] to obtain the depth map and pass it to the ResNet encoder.
4) GT Grid: This approach assumes access to the ground-truth depth map, which serves as a performance upper bound for our experiments. We convert the ground-truth depth map into a voxel grid $\mathcal{G}$, which is the label of our perception module.

In the average performance shown at Fig. 4, our approach outperforms all of the compared approaches except GT Grid. Notably, the GT Grid agent exhibits better performance in novel scenes compared to seen scenes. Two primary factors may contribute to this discrepancy: Firstly, the GT Grid agent benefits from using ground-truth depth information, minimizing any domain gap in perception. Secondly, the randomly selected novel scenes may pose less complexity and difficulty compared to the seen scenes.

SNN achieves SR of 35%, SPL of 34%, and NE of $3.25m$, surpassing MobileStereoNet (28%, 26%, $3.46m$) and Depth Anything (27%, 26%, $3.76m$). These findings suggest that the geometric feature extracted by our approach is better suited for the navigation policy. Additionally, all hybrid approaches (SNN, Depth Anything, and MobileStereoNet) outperform the semantic counterpart (ResNet) by a large margin.

To assess generalizability, we compare the performance

TABLE I: **Per scene numerical results.** We compare our method (SNN) against agents using ResNet (RN), Depth Anything (DA), MobileStereoNet (MS), and GT Grid (GT). The best results are **bolded**, and the second-best results are underscored. GT is listed only for reference but is not comparable.

	Scene	SR (%) ↑					SPL (%) ↑					NE (meters) ↓				
		RN	DA	MS	SNN	GT	RN	DA	MS	SNN	GT	RN	DA	MS	SNN	GT
Seen	Beechwood_0_int	**0.40**	0.00	0.10	0.30	0.60	**0.38**	0.00	0.09	0.30	0.60	**3.10**	6.07	6.17	3.74	1.83
	Beechwood_1_int	0.00	0.20	**0.30**	0.00	0.20	0.00	0.20	**0.29**	0.00	0.20	7.05	5.08	**3.79**	5.68	3.92
	Benevolence_0_int	**0.90**	0.70	0.40	0.80	0.80	**0.75**	0.66	0.33	0.72	0.80	**0.24**	0.38	1.12	0.32	0.39
	Benevolence_1_int	0.00	**0.30**	0.20	**0.30**	0.30	0.00	**0.30**	0.20	**0.30**	0.30	4.39	**2.90**	4.06	3.16	3.67
	Benevolence_2_int	0.00	0.20	0.30	**0.40**	0.20	0.00	0.20	0.30	**0.40**	0.20	2.71	3.13	**1.62**	1.71	2.77
	Seen Average	0.26	0.28	0.26	**0.36**	0.42	0.23	0.27	0.24	**0.34**	0.42	3.50	3.51	3.35	**2.92**	2.52
Novel	Ihlen_0_int	**0.90**	0.50	0.40	0.60	1.00	**0.76**	0.49	0.36	0.59	1.00	**0.40**	2.20	1.60	1.52	0.00
	Ihlen_1_int	0.00	0.00	**0.30**	0.20	0.40	0.00	0.00	**0.29**	0.20	0.40	5.16	4.63	**3.54**	3.74	2.81
	Merom_0_int	0.50	**0.70**	0.40	0.40	0.80	0.41	**0.70**	0.33	0.31	0.80	1.76	**1.49**	3.38	2.31	0.70
	Merom_1_int	0.30	0.30	**0.40**	**0.40**	0.50	0.25	0.30	0.36	**0.37**	0.49	2.99	4.62	**2.62**	3.74	2.46
	Pomaria_0_int	0.00	0.20	0.20	**0.30**	0.60	0.00	0.19	0.20	**0.30**	0.60	5.88	5.37	4.67	**4.29**	1.44
	Pomaria_1_int	0.00	0.10	0.20	**0.40**	0.30	0.00	0.09	0.20	**0.39**	0.30	4.77	4.51	**4.05**	4.45	3.65
	Rs_int	0.00	0.10	**0.30**	**0.30**	0.40	0.00	0.10	**0.30**	**0.30**	0.40	2.65	**2.53**	2.61	2.70	1.35
	grocery_store_cafe	0.00	**0.20**	0.10	**0.20**	0.60	0.00	**0.20**	0.07	**0.20**	0.60	7.20	5.99	5.77	**4.85**	1.92
	Novel Average	0.21	0.26	0.29	**0.35**	0.57	0.18	0.26	0.26	**0.33**	0.57	3.85	3.92	3.53	**3.45**	1.79
	All Average	0.23	0.27	0.28	**0.35**	0.52	0.20	0.26	0.26	**0.34**	0.51	3.72	3.76	3.46	**3.25**	2.07

TABLE II: **Computation cost of each approach.** We quantify the multiply-accumulate (MACs) and model parameters for each approach, alongside assessing the real-time capability by measuring its operational frequency.

Method	MACs ↓	Params ↓	real-time
ResNet	**1.82G**	11.35M	✓
Depth Anything	91.38G	35.52M	✗
MobileStereoNet	300.34G	13.11M	✗
SNN (Ours)	9.53G	**1.15M**	✓

TABLE III: **Impact of modular learning.** We measure navigation performance and computation costs for policies built upon various grid sizes.

Method	SR ↑	SPL ↑	NE ↓
With modular learning	**0.35**	**0.34**	**3.25**
Without modular learning	0.22	0.22	3.66

gaps between seen and novel scenes. In terms of SR and SPL, SNN experiences a drop of 1%, while ResNet exhibits a more substantial decrease of 5%. This underscores the robustness of SNN in comparison to the semantic approach.

Moreover, we present the computation costs of each approach (Tab. II). We introduce three computation cost metrics: multiply-accumulate (MACs), model parameters (Params), and real-time. MAC represents the number of computational operations performed, with higher MACs typically indicating greater computational power but slower processing frequency. Larger model parameters contribute to increased memory consumption and result in larger weight files. For the last metric, an agent is deemed real-time if it operates at a frequency exceeding 30 Hz on an NVIDIA RTX 4070 GPU. The results show that the other two hybrid approaches have significantly higher computation costs than SNN, resulting in the inability to operate in a real-time manner.

B. Does modular learning help?

We leverage modular learning during our SNN training. Concretely, we first train the perception module using only stereo data; then, we train the policy module using ground-truth perception. Once two modules are trained to converge, we optimize SNN in an end-to-end manner. We find this leads to higher training efficiency and stability, leading to higher ultimate navigation performance. In this experiment, we investigate whether this paradigm improves performance. As a comparison, we train an agent without modular learning by directly learning the outcomes of both the perception module and policy module end-to-end (same as the last step of our modular learning procedure). As shown in Tab. III, without modular learning, the success rate of SNN drops from 35% to 22% (37% relative). Similarly, SPL drops 35%, and NE rises by 13%.

We find that this is mainly due to the challenges of optimizing the perception module. Given a randomly initialized perception module, the policy module could only obtain noisy and meaningless input. As a result, the policy module cannot capture geometric features and further hinders the optimization of the perception module. Beyond improved navigation performance, we envision this modular design to improve sim-to-real transfer in future studies and elaborate in the last section (Sec. VI).

TABLE IV: **Navigation performance with various grid sizes.** We measure navigation performance and computation costs for policies built upon various grid sizes.

Grid Size	SR ↑	SPL ↑	NE ↓	MACs ↓	Params ↓
$64 \times 64 \times 64$	**0.52**	**0.51**	**2.07**	9.53G	1.15M
$16 \times 16 \times 16$	0.39	0.39	2.72	6.22G	1.06M
$4 \times 4 \times 4$	0.22	0.20	3.85	**4.53G**	**0.95M**

C. Does voxel grid resolution matter?

In this experiment, we investigate the effect of lower voxel grid resolution on the navigation policy. While decreasing the resolution of the voxel grid could reduce the costs of computation, especially at the perception stage, it limits the agent's understanding of the environment at lower granularity. We train two agents with a ground-truth occupancy grid

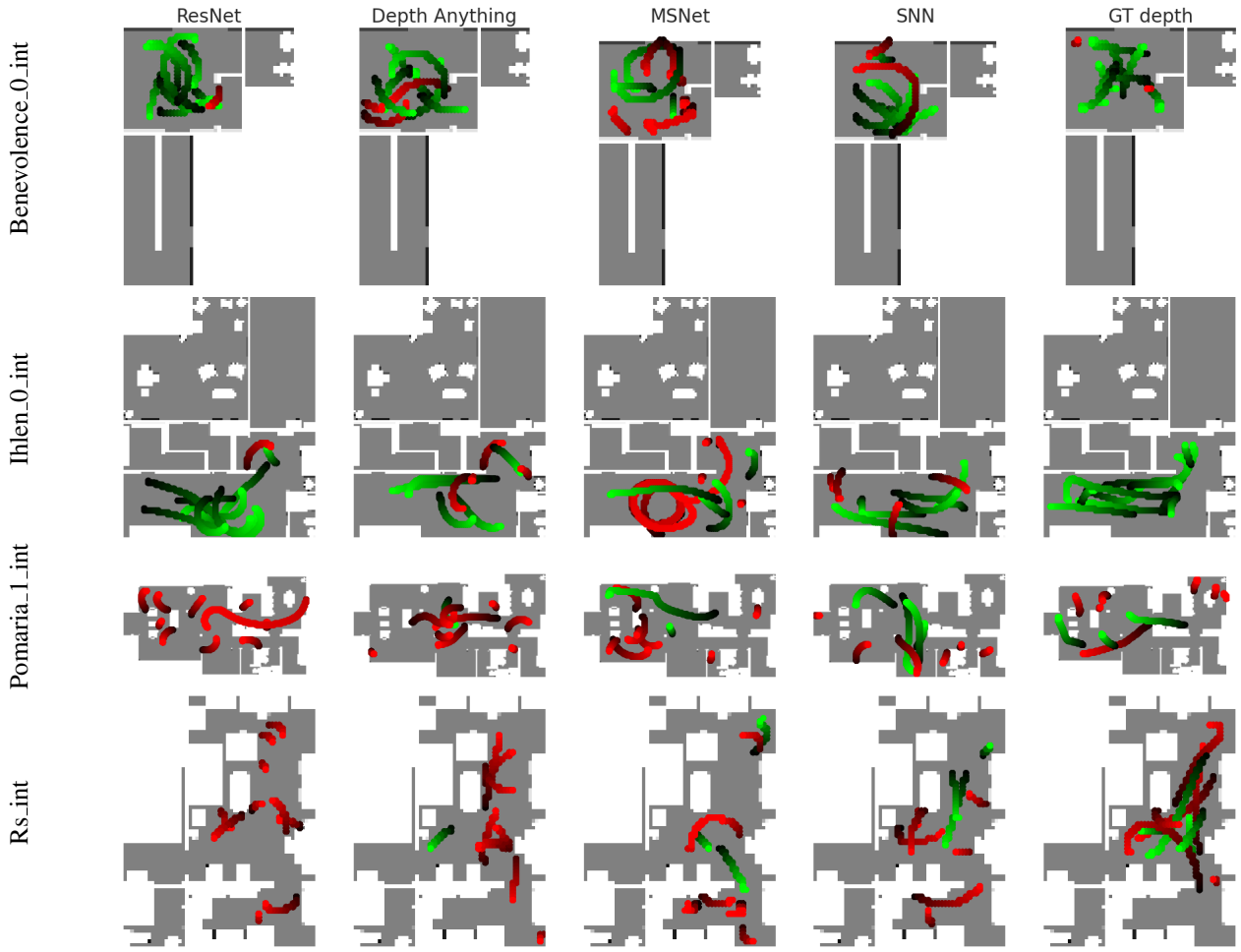

Fig. 5: **Qualitative results**. We present the visualizations of navigation experiments from four scenes. The top environment is seen in the demonstration dataset, and the rest are novel environments. We label the successful trials in green and the failures in red. The agents start from the light color and move towards the dark color.

resolution of $16 \times 16 \times 16$ and $4 \times 4 \times 4$ as their input, which is 4 and 16 times smaller than our original policy trained on $64 \times 64 \times 64$ grid and present the results at Tab. IV

From the results, we can observe that the navigation performance drops greatly from 52% success rate to 39% success rate (reduced by 25%) when the grid size is smaller by four times. At the same time, the computation cost (MACs) reduces by 35%, having a slightly larger reduction than the performance. The success rate further reduces to 22% when using the grid of $4 \times 4 \times 4$, and SPL and NE share a similar pattern. These findings indicate that our approach could perceive the environment in coarser-grained time-sensitive scenarios with the tradeoff of weaker navigation performance.

VI. Conclusion

This paper contributed SNN, a novel visual navigation network encapsulating an auxiliary voxel occupancy grid extracted from an RGB stereo camera. We presented empirical evidence showing that SNN outperforms the semantic and hybrid baselines with better SPL, SR, and NE metrics. We further showed the better generalizability demonstrated by SNN compared to the semantic approach and higher computation efficiency compared to other hybrid approaches. We performed ablation studies to confirm the effectiveness of modular learning and the effect of various voxel grid sizes. Our work shows the possibility of utilizing an auxiliary 3D representation for visual navigation that improves performance and efficiency.

Limitations. In this work, we examine the generalizability of all approaches by testing them in simulated novel environments. However, their transferability in the real world is still in question. Nevertheless, recent works have shown great potential to achieve sim-to-real transfer using a photorealistic simulator [49] or modular learning [24], like the design we employed in this work. Another limitation is the lack of implicit (LSTM) or explicit memory (mapping) of the environment. These techniques could lead to better navigation performance, but they are out of the scope of this work.

Future works. In future works, we are interested in investigating scaling up the perception module. Thanks to the modular design, our perception module could be pre-trained

on large-scale real-world stereo datasets. These datasets are cheaper and have a larger scale than those obtained through robot navigation in the real world and potentially lead to policies with a narrower sim-to-real gap [39]. We are also interested in achieving planning capability using differentiable planners [37].

REFERENCES

[1] P. Anderson *et al.*, "On Evaluation of Embodied Navigation Agents," Jul. 2018, arXiv:1807.06757 [cs].

[2] S. Tellex *et al.*, "Robots That Use Language," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 25–55, 2020.

[3] D. Gandhi *et al.*, "Learning to fly by crashing," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 3948–3955, iSSN: 2153-0866.

[4] A. Loquercio *et al.*, "DroNet: Learning to Fly by Driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, Apr. 2018.

[5] A. Kendall *et al.*, "Learning to Drive in a Day," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 8248–8254, iSSN: 2577-087X.

[6] Y. Pan *et al.*, "Agile Autonomous Driving using End-to-End Deep Imitation Learning," Aug. 2019, arXiv:1709.07174 [cs].

[7] G. Kahn *et al.*, "LaND: Learning to Navigate From Disengagements," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1872–1879, Apr. 2021.

[8] H. Nguyen *et al.*, "Motion Primitives-based Navigation Planning using Deep Collision Prediction," in *2022 International Conference on Robotics and Automation (ICRA)*, May 2022, pp. 9660–9667.

[9] V. Tolani *et al.*, "Visual Navigation Among Humans With Optimal Control as a Supervisor," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2288–2295, Apr. 2021.

[10] D. Shah *et al.*, "ViNG: Learning Open-World Navigation with Visual Goals," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 13 215–13 222, iSSN: 2577-087X.

[11] K. He *et al.*, "Deep Residual Learning for Image Recognition," 2016, pp. 770–778.

[12] C. Yu *et al.*, "BiSeNet: Bilateral Segmentation Network for Real-time Semantic Segmentation," 2018, pp. 325–341.

[13] M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in *Proceedings of the 36th International Conference on Machine Learning*. PMLR, May 2019, pp. 6105–6114, iSSN: 2640-3498.

[14] M. Sandler *et al.*, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," 2018, pp. 4510–4520.

[15] D. Fox *et al.*, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, Mar. 1997.

[16] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, May 1993, pp. 802–807 vol.2.

[17] C. Rösmann *et al.*, "Kinodynamic trajectory optimization and control for car-like robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 5681–5686.

[18] W. B. Shen *et al.*, "Situational Fusion of Visual Representation for Visual Navigation," 2019, pp. 2881–2890.

[19] J. Wasserman *et al.*, "Last-Mile Embodied Visual Navigation," in *Proceedings of The 6th Conference on Robot Learning*. PMLR, Mar. 2023, pp. 666–678, iSSN: 2640-3498.

[20] S. Kareer *et al.*, "ViNL: Visual Navigation and Locomotion Over Obstacles," Jan. 2023, arXiv:2210.14791 [cs].

[21] M. Seo *et al.*, "Learning to Walk by Steering: Perceptive Quadrupedal Locomotion in Dynamic Environments," Feb. 2023, arXiv:2209.09233 [cs].

[22] N. U. Akmandor *et al.*, "Deep Reinforcement Learning based Robot Navigation in Dynamic Environments using Occupancy Values of Motion Primitives," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2022, pp. 11 687–11 694.

[23] E. Wijmans *et al.*, "DD-PPO: Learning Near-Perfect PointGoal Navigators from 2.5 Billion Frames," Jan. 2020, arXiv:1911.00357 [cs].

[24] T. Gervet *et al.*, "Navigating to objects in the real world," *Science Robotics*, vol. 8, no. 79, p. eadf6991, Jun. 2023, publisher: American Association for the Advancement of Science.

[25] S. Bansal *et al.*, "Combining Optimal Control and Learning for Visual Navigation in Novel Environments," in *Proceedings of the Conference on Robot Learning*. PMLR, May 2020, pp. 420–429, iSSN: 2640-3498.

[26] D. S. Chaplot *et al.*, "Object Goal Navigation using Goal-Oriented Semantic Exploration," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 4247–4258.

[27] W. Hu *et al.*, "Modular Neural Network Policies for Learning In-Flight Object Catching with a Robot Hand-Arm System," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2023, pp. 944–951, arXiv:2312.13987 [cs].

[28] H. Li *et al.*, "ViHOPE: Visuotactile In-Hand Object 6D Pose Estimation With Shape Completion," *IEEE Robotics and Automation Letters*, vol. 8, no. 11, pp. 6963–6970, Nov. 2023.

[29] C. Li *et al.*, "BEHAVIOR-1K: A Benchmark for Embodied AI with 1,000 Everyday Activities and Realistic Simulation," in *Proceedings of The 6th Conference on Robot Learning*. PMLR, Mar. 2023, pp. 80–93, iSSN: 2640-3498.

[30] L. Yang *et al.*, "Depth Anything: Unleashing the Power of Large-Scale Unlabeled Data," Jan. 2024, arXiv:2401.10891 [cs].

[31] F. Shamsafar *et al.*, "MobileStereoNet: Towards Lightweight Deep Networks for Stereo Matching," 2022, pp. 2417–2426.

[32] D. Chen *et al.*, "Learning by Cheating," in *Proceedings of the Conference on Robot Learning*. PMLR, May 2020, pp. 66–75.

[33] A. Loquercio *et al.*, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, p. eabg5810, 2021.

[34] M. Sorokin *et al.*, "Learning to Navigate Sidewalks in Outdoor Environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3906–3913, Apr. 2022.

[35] M. Savva *et al.*, "Habitat: A Platform for Embodied AI Research," 2019, pp. 9339–9347.

[36] S. Gupta *et al.*, "Cognitive Mapping and Planning for Visual Navigation," 2017, pp. 2616–2625.

[37] L. Zhao *et al.*, "E(2)-equivariant graph planning for navigation," *IEEE Robotics and Automation Letters*, vol. 9, no. 4, 2024.

[38] B. H. Wang *et al.*, "Detecting and Mapping Trees in Unstructured Environments with a Stereo Camera and Pseudo-Lidar," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 14 120–14 126, iSSN: 2577-087X.

[39] H. Li *et al.*, "Stereovoxelnet: Real-time obstacle detection based on occupancy voxels from a stereo camera using deep neural networks," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[40] H. Hirschmuller, "Accurate and efficient stereo processing by semi-global matching and mutual information," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2, Jun. 2005, pp. 807–814 vol. 2, iSSN: 1063-6919.

[41] T. Eppenberger *et al.*, "Leveraging Stereo-Camera Data for Real-Time Dynamic Obstacle Detection and Tracking," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 10 528–10 535, iSSN: 2153-0866.

[42] B. Liu *et al.*, "Local Similarity Pattern and Cost Self-Reassembling for Deep Stereo Matching Networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 2, pp. 1647–1655, Jun. 2022, number: 2.

[43] G. Xu *et al.*, "Attention Concatenation Volume for Accurate and Efficient Stereo Matching," 2022, pp. 12 981–12 990.

[44] J. Žbontar and Y. LeCun, "Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches," *Journal of Machine Learning Research*, vol. 17, no. 65, pp. 1–32, 2016.

[45] H. Sak *et al.*, "Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition," Feb. 2014, arXiv:1402.1128 [cs, stat].

[46] Y. Wang *et al.*, "Pseudo-LiDAR From Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving," 2019, pp. 8445–8453.

[47] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Jan. 2017, arXiv:1412.6980 [cs].

[48] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[49] K. Ehsani *et al.*, "Imitating Shortest Paths in Simulation Enables Effective Navigation and Manipulation in the Real World," Dec. 2023, arXiv:2312.02976 [cs].