Université de Montréal

Leveraging Self-Supervision for Visual Embodied Navigation with Neuralized Potential Fields

par

Miguel Angel Saavedra Ruiz

Département de mathématiques et de statistique
Faculté des arts et des sciences

Mémoire présenté en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en Discipline

May 9, 2023

Université de Montréal

Faculté des arts et des sciences

Ce mémoire intitulé

Leveraging Self-Supervision for Visual Embodied Navigation with Neuralized Potential Fields

présenté par

Miguel Angel Saavedra Ruiz

a été évalué par un jury composé des personnes suivantes :

Aaron Courville

(président-rapporteur)

Liam Paull

(directeur de recherche)

Guy Wolf

(membre du jury)

Résumé

Une tâche fondamentale en robotique consiste à naviguer entre deux endroits. En particulier, la navigation dans le monde réel nécessite une planification à long terme à l'aide d'images RVB (RGB) en haute dimension, ce qui constitue un défi considérable pour les approches d'apprentissage de bout-en-bout. Les méthodes semi-paramétriques actuelles parviennent plutôt à atteindre des objectifs éloignés en combinant des modèles paramétriques avec une mémoire topologique de l'environnement, souvent représentée sous forme d'un graphe ayant pour nœuds des images précédemment vues. Cependant, l'utilisation de ces graphes implique généralement l'ajustement d'heuristiques d'élagage afin d'éviter les arêtes superflues, limiter la mémoire requise et permettre des recherches raisonnablement rapides dans le graphe.

Dans cet ouvrage, nous montrons comment les approches de bout-en-bout basées sur l'apprentissage auto-supervisé peuvent exceller dans des tâches de navigation à long terme. Nous présentons initialement Duckie-Former (DF), une approche de bout-en-bout pour la navigation visuelle dans des environnements routiers. En utilisant un Vision Transformer (ViT) pré-entraîné avec une méthode auto-supervisée, nous nous inspirons des champs de potentiels afin de dériver une stratégie de navigation utilisant en entrée un masque de segmentation d'image de faible résolution[1]. DF est évalué dans des tâches de navigation de suivi de voie et d'évitement d'obstacles. Nous présentons ensuite notre deuxième approche intitulée One-4-All (O4A). O4A utilise l'apprentissage auto-supervisé et l'apprentissage de variétés afin de créer un pipeline de navigation de bout-en-bout sans graphe permettant de spécifier l'objectif à l'aide d'une image. La navigation est réalisée en minimisant de manière vorace une fonction de potentiel définie de manière continue dans l'espace latent O4A[2].

Les deux systèmes sont entraînés sans interagir avec le simulateur ou le robot sur des séquences d'exploration de données RVB et de contrôles non experts. Ils ne nécessitent aucune mesure de profondeur ou de pose. L'évaluation est effectuée dans des environnements simulés et réels en utilisant un robot à entraînement différentiel.

Mots clés: navigation visuelle, apprentissage auto-supervisé, champs de potentiel, apprentissage de variétés, robotique

[1]Page de projet pour Duckie-Former: `https://sachamorin.github.io/dino/`
[2]Page de projet pour One-4-All: `https://montrealrobotics.ca/o4a/`

Abstract

A fundamental task in robotics is to navigate between two locations. Particularly, real-world navigation can require long-horizon planning using high-dimensional RGB images, which poses a substantial challenge for end-to-end learning-based approaches. Current semi-parametric methods instead achieve long-horizon navigation by combining learned modules with a topological memory of the environment, often represented as a graph over previously collected images. However, using these graphs in practice typically involves tuning various pruning heuristics to prevent spurious edges, limit runtime memory usage, and allow reasonably fast graph queries.

In this work, we show how end-to-end approaches trained through Self-Supervised Learning can excel in long-horizon navigation tasks. We initially present Duckie-Former (DF), an end-to-end approach for visual servoing in road-like environments. Using a Vision Transformer (ViT) pretrained with a self-supervised method, we derive a potential-fields-like navigation strategy based on a coarse image segmentation model[3]. DF is assessed in the navigation tasks of lane-following and obstacle avoidance. Subsequently, we introduce our second approach called One-4-All (O4A). O4A leverages SSL and manifold learning to create a graph-free, end-to-end navigation pipeline whose goal is specified as an image. Navigation is achieved by greedily minimizing a potential function defined continuously over the O4A latent space. O4A is evaluated in complex indoor environments[4].

Both systems are trained offline on non-expert exploration sequences of RGB data and controls, and do not require any depth or pose measurements. Assessment is performed in simulated and real-world environments using a differential-drive robot.

Keywords: visual navigation, self-supervised learning, potential fields, manifold learning, robotics

[3]Project page for Duckie-Former: `https://sachamorin.github.io/dino/`
[4]Project page for One-4-All: `https://montrealrobotics.ca/o4a/`

Contents

List of tables

List of figures

List of Acronyms and Abbreviations

A^*	A-Star Algorithm
APFs	Artificial Potential Fields
AWA*	Anytime Weighted A-Star Algorithm
BYOL	Bootstrap Your Own Latent
CFT	Collision-Free Trajectories
CNNs	Convolutional Neural Networks
D^*	Dymamic A-Star Algorithm
DF	Duckie-Former
DiNO	Self-Distillation with No labels
DL	Deep Learning
DTG	Distance to Goal

EMA	Exponential Moving Average
GC-BC	Goal Conditioned Behavioral Cloning
HER	Hindsight Experience Replay
HPF	Harmonic Potential Function
Hybrid A^*	Hybrid-State A-Star Algorithm
InvDM	Inverse Dynamics Models
IoU	Intersection Over Union
IsoMap	Isometric Mapping
LTM	Learning to Map
LTVN	Lifelong Topological Visual Navigation
MPC	Model Predictive Control
NRNS	No RL, No Simulator
O4A	One-4-All

PCA	Principal Component Analysis
PRM	Probabilistic Roadmap
RL	Reinforcement Learning
RNNs	Recurrent Neural Networks
RPP	Randomized Potential Planner
RRT	Rapidly Exploring Random Tree
SimSiam	Simple Siamese
SLAM	Simultaneous Localization and Mapping
SMT	Scene Memory Transformer
SPL	Success weighted by Path-Length
SPTM	Semi-Parametric Topological Memory
SR	Success Rate
SSL	Self-Supervised Learning

SSR	*Soft* Success Rate
TO	Trajectory Optimization
UAF	Uncertainty Aware Functions
ViNG	Visual Navigation with Goals
ViTs	Vision Transformers

Remerciements

I dedicate this work to all those who have supported me, not only through academic guidance but also with their day-to-day counsel.

To start with, I thank my advisor, Liam Paull, for granting me the opportunity to conduct research within his group. I would like to express my sincere appreciation for the exceptional guidance and mentorship you have provided me with. Your willingness to listen to even the most trivial of my ideas and your unwavering encouragement to experiment with robots have been incredibly valuable. Above all, I am grateful for our conversations and how you always pushed me to approach interesting problems and design solutions for them, rather than solely seeking out problems for preconceived solutions. Your guidance has been invaluable in helping me become a better scientist. Thank you.

To Mila and DIRO for their financial support, which made it possible for me to pursue my studies. Furthermore, I would like to express my gratitude for your exceptional work facilities, which were critical in carrying out this work, as well as for fostering a lively community where ideas can be freely exchanged.

To Sacha Morin, who not only made invaluable contributions to this work but also became a friend. Thank you for engaging in countless discussions with me and constantly challenging my ideas. You have helped me to develop a more critical attitude towards new ideas, as well as increasing my confidence in presenting them.

To Victor Romero-Cano for introducing me to the fascinating world of robotics and research. Despite the distance, you have consistently been a source of support, always encouraging me to become a better researcher in robotics. I truly appreciate all the help and guidance you have provided me with and I acknowledge that I would not be where I am today without your valuable advice.

To all the members of the Robotics and Embodied AI Lab (REAL). I am grateful for the stimulating discussions, the enjoyable lunch sessions, and the guidance provided, even on the most trivial of matters. In particular, I want to thank Steven Parkison, Kaustubh Mani and Ali Harakeh. Their insightful comments and discussions have been instrumental in shaping my ideas and improving the quality of this work.

To my friends, especially Ana Pinto and Gustavo Salazar, for their support during my academic journey. I am deeply grateful for their willingness to always lend me their valuable time to read countless reports of mine and for cheering me up at all times.

To my parents, sister, grandparents and uncle. Thank you for being my cornerstone and always support my ideas and goals, even during challenging times. I am truly grateful for their constant presence in my life and could not have pursued my dreams without their unwavering support. Particularly, I would like to thank God for the countless blessings bestowed upon me. This work is for all of you.

Last but not least, to my beloved life partner, Lilibeth. Thank you for your unrivalled patience and support. You have always been there for me, believing in my work even more than I. This work is for you with all my love and appreciation.

Chapter 1

Introduction

Over the past decade, Deep Learning (DL) has significantly contributed to improving the state of the art in several computer vision tasks. These advances are powered by learning-based architectures specifically designed to work within images. Examples of such models include Convolutional Neural Networks (CNNs) [**90, 67, 94**] and Vision Transformers (ViTs) [**40, 93**], which have replaced hand-crafted feature extractors [**95, 12, 119, 106**] with a set of trainable parameters [**57**]. Moreover, the ability of learning-based methods to learn directly from raw images makes them appealing to the robotics community, particularly as perception modules [**50**]. Applications of DL within robotics can be seen in trajectory forecasting [**22**], control [**5**] and most notably, navigation [**124, 127, 26, 151**], which is particularly relevant to this work.

Navigation for robotics can be defined as the ability of a robot to move from one location to another [**131**]. This problem is characterized by a robot's ability to identify the most efficient and feasible path between a start pose (position and orientation) and a goal pose in a given environment [**88**]. The standard approach involves first piloting the robot within the environment to build a map containing metric information (i.e., metric map), often using a range sensor, and then using this representation for planning [**139**]. However, the memory complexity of these classic approaches scales poorly with the size of the environment, and they do not exploit semantic information nor visual cues [**128**].

Alternatively, learning-based methods for navigation also dubbed *experiential learning* [**92**] are widely use to overcome these limitations. Unlike traditional approaches, experiential learning operates directly with high-dimensional data (e.g., images) and reasons about non-geometric concepts in a scene. Furthermore, these methods are more intuitive to use for non-expert users as they allow for goal positions to be specified using images of places or objects rather than coordinates in a metric map [**26**].

Although learning-based approaches offer a sound solution to robot navigation, they are not free of limitations. Firstly, Deep Learning methods can suffer from sample inefficiency

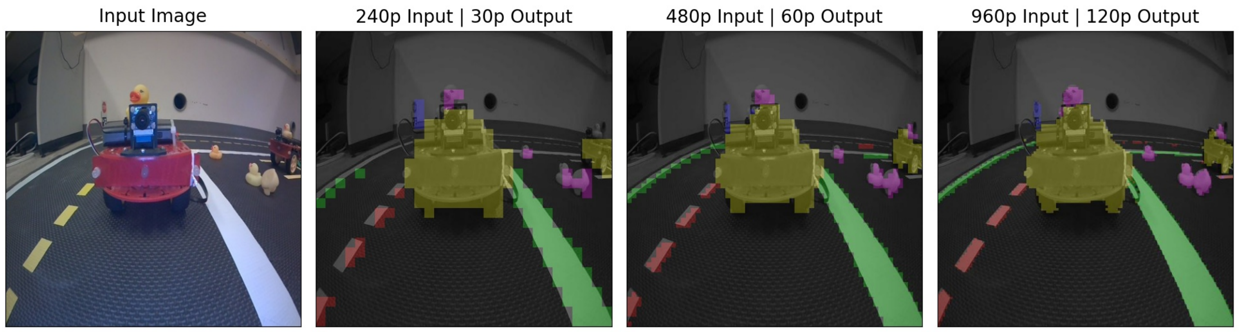


Fig. 1.1. Semantic segmentation predictions obtained with Duckie-Former at different input resolutions. The original input image is divided into patches (e.g., 240p input produces 30 patches at the output. The letter *p* stands for pixels) and a ViT-based model performs segmentation on these patches. We demonstrate how to use these predictions for navigating a Duckietown environment [**108**] in Chapter 3.

as they typically require a substantial number of annotated images to produce estimators with good generalization capabilities [**57, 41, 126**]. Therefore, deployment in novel visual environments is likely to need an expensive data annotation procedure [**158**]. Secondly, DL models have been characterized by a "depth race" with architectures of increasing size [**64**], which are of limited use for embodied agents requiring high inference performance on resource-constrained hardware.

To address the first limitation, Self-Supervised Learning (SSL) has been successfully used to train models in the absence of annotated data. The idea is to design clever *pretext tasks* to extract valuable information from the data. Examples include unshuffling image patches [**38, 103**], predicting image rotations [**52**] or reconstructing masked input [**107**]. Not surprisingly, experiential learning has successfully extrapolated ideas from SSL to robot navigation. A broad variety of methods leverage SSL-based objectives to learn a global controller that maps images directly to robot actions [**149, 31, 102, 8, 25, 98**]. However, mapping images directly to actions has been shown to have poor navigation capabilities over long-horizon tasks [**127, 128**]. Indeed, research in neuroscience demonstrated how humans and animals achieve long-horizon navigation by using not only motor controls but also a "memory map" of the environment [**104, 53, 47**].

Drawing inspiration from neuroscience to perform long-horizon navigation, *topological memory* representations [**85**] are used to divide the navigation problem into two parts. First, the memory representation is used to produce a globally coherent navigation plan, which is then followed waypoint-by-waypoint using a learned or classical local controller [**8**]. Approaches that incorporate both memory and learning-based components are referred to as *semi-parametric*, while approaches that rely solely on learning are known as *fully-parametric*.

While semi-parametric methods have proven to be effective for image-based navigation both indoors [**124, 151, 26, 81**] and outdoors [**127, 129**], they still encounter memory issues. This is a result of the topological memory typically being encoded as a graph whose nodes represent visited states (i.e., images) and edges traversability. As the environment size increases, more nodes and edges are added to the graph, increasing the memory requirements.

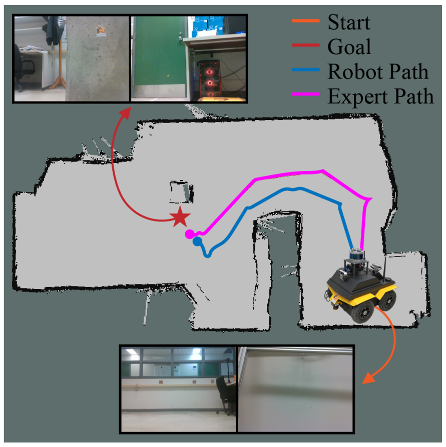
In addition, spurious connections in the graph can impede navigation performance as they may represent non-feasible transitions in the physical world, leading to failure modes in the global planning stage. Although the literature offers partial solutions to these limitations by pruning the graph with hand-crafted heuristics [**151, 127**], these add complexity to the problem and generally require tuning for each environment.

To address the aforesaid limitations, this work demonstrates how SSL can be used to create a neural memory representation for robot navigation within end-to-end navigation approaches. Hence, this work first proposes **Duckie-Former (DF)**, a fully-parametric approach for monocular visual servoing[1] using few annotated images. DF works by training an instance segmentation model based on ViTs to perform "coarse" semantic segmentation at the 8×8 patch level, as shown in Figure 1.1. The segmentation mask is subsequently used as a repulsor potential in a potential-fields-based navigation and control strategy [**32**][2]. Sample inefficiency is addressed using standard data augmentation techniques as well as pretrained weights from a leading self-supervised method called DiNO [**21**]. As for the computational aspect, it is shown how ViTs can be used to predict labels at different resolutions, allowing a compromise between prediction granularity, inference speed and memory footprint in real-life robotic applications.

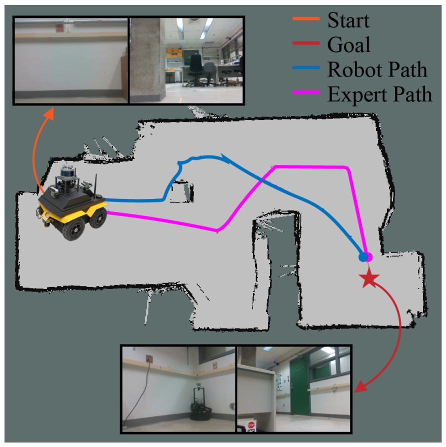
This approach is further extended and we propose **One-4-All (O4A)**, an end-to-end fully-parametric method for image-goal navigation[3]. O4A is trained offline using non-expert exploration sequences of RGB data and controls. We first rely on SSL to identify neighbouring RGB observations. Armed with this notion of connectivity, we compute a graph to derive a manifold learning [**137, 157**] objective for our planning module, which we dub the **geodesic regressor**. The geodesic regressor will learn to predict shortest path lengths between pairs of RGB images and in that sense, encodes the geometry of the environment and acts as our memory module [**60, 51, 68, 46**]. While we do compute a temporary graph during training, we discard it for navigation, and found that it does not require the graph pruning heuristics of existing semi-parametric methods. Intuitively, we trade a potentially high number of nodes and edges in a graph for a fixed number of learnable parameters, thus mitigating the memory limitations of semi-parametric approaches. Inference is also improved: graph queries are replaced with efficient forward passes in a neural network.

Similar to Duckie-Former, O4A draws inspiration from potential fields planning [**32**] for navigation and uses the output of our geodesic regressor as an attractor in a potential

[1]Visual servoing refers to controlling the robot's motion using only information extracted from vision sensor.
[2]This work was published at the *19th Conference on Robotics and Vision (CRV)* [**121**]
[3]This work is under review at the *International Conference on Robotics and Automation (IROS) 2023* [**101**]



(a) First trajectory.



(b) Second trajectory.

Fig. 1.2. Navigation paths produced by O4A (Robot, blue) and by human teleoperation (Expert, magenta) in a 4.65m x 9.10m laboratory. When prompted with a goal image (red star), the robot uses its current front and back RGB observations (orange) to navigate towards the goal by minimizing a neural potential function via gradient descent. The paths were captured using information from a ViCON system, which was not available to the agent.

function. This allows us to frame navigation as a minimization problem, with the global minima located at the goal image. We show how this navigation approach enables the robot to perform long-horizon navigation and succeed even in geometrically complex environments.

To summarize, the main contributions of this work are:

- **Duckie-Former**
  1. Using a ViT pretrained with a label-free self-supervised method, we successfully derive a potential-fields-based function by training a coarse image segmentation model using only 70 training images;
  2. We show how the same model can be used to predict labels at different resolutions, allowing a compromise between prediction granularity, inference speed and memory footprint;
- **One-4-All**
  1. An offline self-supervised training procedure using non-expert exploration sequences of RGB data and controls, without any depth or pose measurements;
  2. A graph-free, end-to-end navigation pipeline that avoids tuning graph pruning heuristics;
  3. A potential fields-based planner that avoids local minima and reaches long-horizon goals, thanks to a geodesic attractor trained with a manifold learning objective;
  4. An interpretable system that recovers the geometry of the environment in its latent space, even in the absence of any pose information.

We perform experimental validation of Duckie-Former in two visual servoing tasks using the Duckietown environment [**108**]. Similarly, we show that One-4-All achieves state-of-the-art indoor navigation in 8 simulated environments. We further provide a real-world evaluation using the Jackal UGV platform.

Chapter 2

Background

In this section, we first provide an overview of the preliminary background necessary to comprehend the visual navigation problems addressed in this work. We begin by defining the motion planning problem in Chapter 2.1.1 and provide an overview of classic techniques. In Chapter 2.1.2, we delve into visual navigation tasks and their variations. Then, we review representation learning in Chapter 2.1.3, including neural architectures for image processing and their application in robotics. We conclude the preliminaries part of this section discussing two relevant subjects for the development of this work: self-supervised learning in Chapter 2.1.4 and manifold learning in Chapter 2.1.5. Afterwards, we present the related work in Chapter 2.2, which is divided into two parts: First we discuss applications of classic robot navigation in Chapter 2.2.1 and we conclude this section presenting learning based approaches for navigation in Chapter 2.2.2

2.1. Preliminaries

2.1.1. Motion Planning

Motion planning in the context of robotics is defined by LaValle [**88**] as "designing algorithms that generate useful motions by processing complicated geometric models". In other words, how to plan a sequence of motions to navigate an agent from a starting position q_I to a goal position q_G in a given environment [**87**]. The motion planning stack fulfills a critical role in any robotics pipeline as it is responsible for producing an efficient yet feasible path to reach a goal. For instance, in 2005, the *Stanley* robot won the DARPA challenge by navigating approximately 150 miles over rugged desert roads using only onboard sensors [**140**]. One of the chief ingredients for the success of Stanley was its optimization-based lateral motion planer, whose role was primarily obstacle avoidance.

To formalize the idea of motion planning, we need to define the configuration space $\mathcal{C}$. This is defined as as the set of rigid body transformations that could be applied to a robot

Family	Description	Algorithm
Sampling Based	This family of methods avoid an explicit construction of $\mathcal{C}_{\texttt{obs}}$ and instead probes the configuration space $\mathcal{C}$ with a sampling scheme. Then, all samples that are also felt into $\mathcal{C}_{\texttt{obs}}$ are rejected with a collision detection module [**88, 43**]. A navigation path is retrieved by connecting samples between q_I and q_G.	RPP [**10**] PRM [**78**] RRT [**89**] RRT* [**77**] Informed RRT [**49**]
Graph Based	These methods are mainly searching algorithms over a discrete state space. Giving a discrete representation of the configuration space $\mathcal{C}$ (e.g., occupancy grid, lattice, graph), planning is formulated as a searching problem over the state space with starting position q_I and goal q_G [**56**]. The result of this search is a path that visit different states in $\mathcal{C}$ and arrives to the goal.	Dijkstra [**37**] A* [**62**] D* [**132**] AWA* [**69**] Hybrid A* [**39**]
Optimization Based	These methods frame motion planning as an optimization problem subject to different constrained variables [**56**]. More importantly, this family defines a function over the configuration space $\mathcal{C}$ whose global maxima or minima is located at the goal location q_G. The function can be arbitrarily expressive an account for obstacles in the environment as-well as robot morphology [**32**].	APFs [**79, 54, 32**] HPF [**80**] TO [**71**] UAF [**156**]

Table 2.1. Motion planning algorithms.

[**88**]. Typically, $\mathcal{C}$ is an n dimensional manifold (Chapter 2.1.5) where n represents the degrees of freedom of the robot. The set of configurations that cause the robot to collide can be defined as $\mathcal{C}_{\texttt{obs}}$. Equipped with these two notions, motion planning aims to produce a path between q_I and q_G in the the obstacle-free configuration space $\mathcal{C}_{\texttt{free}} = \mathcal{C} \setminus \mathcal{C}_{\texttt{obs}}$ [**55**].

However, it is well-known that the motion planning problem is NP-complete [**19**]. As a result, significant research efforts have been focused on developing planning algorithms that are computationally tractable and can provide approximate solutions to this problem [**131**]. Classic motion planning methods can be categorized into three principal families: graph-based, sampling-based, and optimization-based [**55, 56**]. In Table 2.1, we provide an overview of these methods.

It is worth noticing that each one of these methods has its own limitations and none is the ultimate solution to the planning problem. For instance, sampling based methods assume prior knowledge of the environment to sample the configuration space $\mathcal{C}$ of the robot [**88**]. Graph based methods are highly sensitive to spurious connections in the graph i.e., edges representing unfeasible transitions in our physical world such as traversing through

Navigation Task	Goal Specification	Task Description
Point-Goal Navigation	Coordinate in the environment	The agent has to navigate to a goal coordinate (x, y, z) specified relative to the agent [**100, 8**].
Object-Goal Navigation	Object class	The agent has to navigate to an specific object within the environment (e.g., TV, chair) [**24, 159**].
Area-goal Navigation	Place Category	The agent has to navigate to an area category (e.g., kitchen, bathroom) in the environment [**153**].
Image-goal Navigation	Image	The agent has to navigate to a goal specified with an image [**124, 127, 151**].
Exploration	Maximize map coverage	The agent has to maximize its coverage over the free space of the environment [**23, 123, 51**].
Image-text goal Navigation	Navigation instructions specified with natural language	The agent has to navigate to a goal solely specified as a series of natural language instructions (e.g., navigate to the red building next to the fire hydrant) [**130, 1**].

Table 2.2. Visual navigation tasks.

an obstacle [**56**]. Assume a graph $\mathcal{G} = (V, E)$ of the environment, composed of vertices V and edges E, where $E = E_{\texttt{true}} \cup E_{\texttt{spurious}}$. If the number of spurious edges dominates the graph, i.e., $E_{\texttt{spurious}} >> E_{\texttt{true}}$, graph based methods are doomed to fail [**19**]. Indeed, graph search methods will exploit these spurious edges in the graph, yielding unfeasible navigation paths. Lastly, optimization-based methods are subject to local minima issues and designing the objective function is a non-trivial task [**32**].

This work leverages optimization methods for navigation and takes inspiration from Artificial Potential Fields (APFs). The objective function is composed of an attractive potential p^+ towards the goal and repulsive potentials p^- from previously visited states. Navigation is performed by following the local force produced by the gradient of the potential function: $\mathcal{P} = p^+ + p^-$. It is important to acknowledge that, in the context of traditional APFs, repulsive potentials are typically defined around obstacles in the environment. However, when such obstacles are not explicitly represented, as is the case in this work, repulsive potentials can be defined over previously visited states instead to avoid local minima.

In O4A, we derive a potential function from neural networks, such that it avoids the local minima issue on potential fields, and attains a global minimum at the goal q_G. Notably, the only prior knowledge we assume over the environment is a set of RGB images collected offline, which we use to train our neural potential function for embodied navigation.

2.1.2. Visual Navigation

The development of more expressive Deep Learning (DL) architectures have propelled research on visual navigation for embodied agents. These approaches rely on a vision-based sensor, such as a camera, depth camera, or event-based camera, to perceive the surroundings [**92**]. Moreover, They can directly operate on high-dimensional image data. The use of vision modalities in robot navigation has enabled a shift in the way goals are defined, allowing for images to be used as goals instead of traditional map coordinates (x, y, z) [**44**]. For instance, a navigation goal can be specified as the image of a place within the environment (e.g., living room) or an object of interest (e.g., a plant) [**126**]. Visual navigation also allows for more intuitive use by non-expert users, potentially increasing its impact on society. In Table 2.2, we provide an overview of various visual navigation tasks, as well as recent trends in the field that leverage multi-modal information (e.g., natural language instructions and images) for navigation [**2**].

In this work we tackle specifically the problem of **image goal navigation**. We only use RGB image data and assume no access to depth information to train our models. A more detail description of visual-based methods for embodied navigation is presented in Chapter 2.2

2.1.3. Representation Learning

Representation learning aims to extract meaningful representations from raw data such that these can be seamlessly used in downstream tasks like classification or regression [**15**]. In its more basic form, DL provides a way to learn these representations by composing affine transformations of the form $f(x, W, b) = x^T W + b$ with non-linear activation functions. The matrix W and vector b are composed of learnable parameters optimized via gradient descent or some variation of it [**120, 134, 57**]. Input data is represented by x.

Learning from raw data paved the way to design specialized DL architectures that operate directly on the image domain. There are two prominent neural architectures to learn representations from images: Convolutional Neural Networks (CNNs) [**90**] and Vision Transformers (ViTs) [**40**]. These architectures, along with example applications in the field of robotics, are presented below.

2.1.3.1. Convolutional Neural Networks. These are a specialized type of deep neural architecture tailored to efficiently operate in grid-like topologies, such as images [**90**], time series [**144**] or volumetric data [**165**]. The key component of CNNs are the convolutional layers. These layers apply a set of learnable filters to the input data to extract relevant features also called feature maps [**67**]. Moreover, CNNs can efficiently operate with very high-dimensional data in oppose to fully connected linear layers. This is an effect of weight sharing in CNNs as the same matrix of parameters (i.e., filter), is used to produce a single feature map. By concatenating a series of convolutional layers with non-linearities and pooling layers, CNNs can capture hierarchical representations of the input data [**57**].

Example applications of CNNs can be found for different computer vision tasks. For instance, object classification requires the model to determine which objects are in an image [**84, 67, 94**]. In object detection, the model not only determines the class of the objects but also their positions on the image with a bounding box [**115, 113, 122**]. Ultimately, in instance segmentation the task involves partitioning the image into multiple regions, where each region is labeled with a specific object class, thereby providing pixel-level classification [**66, 27, 6**]. The latter is of particular relevance to this work as Duckie-Former derives its potential function from an instance segmentation model.

2.1.3.2. Transformers and Vision Transformers. ViTs are an adaptation of the transformer architecture originally proposed to address sequence-to-sequence tasks in natural language processing [**146**]. The transformer architecture follows an encoder-decoder structure and is based on a self-attention mechanism. Self-attention allows the model to selectively focus on different parts of the input sequence when making predictions. To achieve this, the input sequence is decomposed into tokens (i.e., words), which are mapped to queries Q, keys K and values V using linear layers. Attention is computed using

$$\text{Attention}(Q, K, V) = \text{Softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad \text{with} \quad \text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{2.1.1}$$

where d_k is the dimension of the keys. Intuitively, attention updates token encodings with a weighted combination of all other token encodings. To further increase the expressivity of the model, transformers use multiple attention heads over the input sequence, which are finally combined with a linear layer. Both encoder and decoder employ self-attention, with the decoder incorporating the output of the encoder and its own input to generate the final output sequence. Advantages of transformers over standard Recurrent Neural Networks (RNNs) [**70**] are their ability to handle long-range dependencies with ease and also, its operations are easily parallelized in compute software [**48**].

Vision Transformers [**40**] adapt the transformer encoder to directly operate with images and compete with CNNs. Echoing the original transformer formulation for natural language processing [**146**], ViTs decompose images ("sentences") into small image patches ("visual words"), typically of size 8 × 8 or 16 × 16. ViTs learn vector encodings for each patch via self-attention. A single transformer layer can therefore learn dependencies between any two patches in the input image. Different improvements over the original ViT have been proposed, such as non-fixed input sequence [**35**], hierarchical feature maps [**93**], and attention windows [**29**].

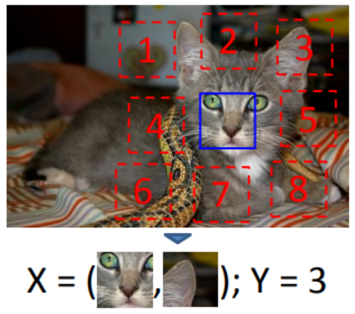
Demonstrating the versatility of ViTs for image processing, it has been successfully applied for various tasks, including instance segmentation. For this task, ViT-based architectures normally leverage an encoder-decoder structure [**7**]. The idea is to reconstruct the segmented image from patch encodings using up-sampling operators to predict labels in pixel space [**133, 163, 155, 111, 136**]. In this work, we exploit a pre-trained ViT and fine-tuned it over few annotated images where segmentation labels are predicted at the patch level.

2.1.3.3. Applications in Robotics. Not surprisingly, both CNNs and ViTs have been widely adopted by the robotics community as perception modules [**50**]. For instance, Wong *et al.* [**152**] exploit a CNNs-based segmentation module to perform pose estimation on objects. The approach leverages a segmentation mask of objects in the scene to retrieve precomputed meshes of these models. Then, the pose is estimated by aligning the meshes with the current objects in sight. In [**161, 136**], a convolutional segmentation backbone is jointly used to localize a robot and generate a segmentation map of the environment using Simultaneous Localization and Mapping (SLAM). Object tracking for self-driving applications have also benefitted from these neural architectures. Wang *et al.* [**148**] proposed PointTrackNet, a neural architecture for generating a bird's-eye-view segmentation from raw point clouds. This segmentation is then delivered into a filtering pipeline to track other vehicles on the road.

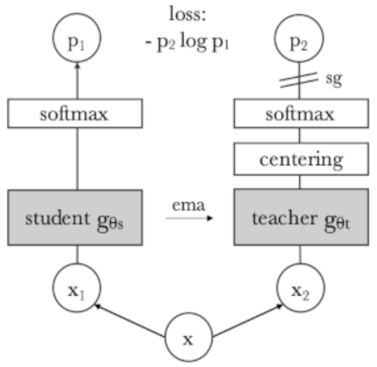
An example with multi-camera images is shown by Zhou *et al.* [**164**]. The authors employ multiple cameras and their pose information to learn a bird's-eye-view map segmentation of the environment with ViTs. Object goal navigation is addressed in [**81**] by first creating a graph with segmented objects within the environment. Subsequently, a ViT is used to retrieve the best next navigation waypoint giving an object goal query. In Chapter 3, we showcase the derivation of a potential-fields-based function for obstacle avoidance and lane following using an instance segmentation model based on ViTs. The key insight is that performing segmentation at different patch resolutions enables a balance between prediction granularity and inference speed. Therefore, allowing the deployment of these models in resource-constrained embodied applications.

2.1.4. Self-Supervised Learning

Self-Supervised Learning (SSL) is a subset of unsupervised learning which aims to learn generic and expressive data representations in the absence of labels. The potential of SSL lies in the possibility to pretrain deep architectures on massive unlabeled datasets [**64**]. A first class of SSL algorithms relies on designing clever pretext tasks that will extract meaningful



(a) Self-supervised pretext task.



(b) DINO architecture.

Fig. 2.1. A self-supervised pretext task and the DINO architecture a) Example of pretext task where the objective is to understand the spatial context of an object in an image in order to tell the relative position between parts [**38**]. b) DINO is a teacher-student architecture trained only with positive samples. It works by maximizing the similarity between the embeddings of two augmented views of an image produced by teacher and student networks. During training, the weights of the student are updated with gradient descent while the teacher weights are updated with an Exponential Moving Average (EMA) of the student's weights [**21**]. Images taken from the original authors.

representations from data [**38, 103, 107, 52**]. An example pretext task is presented in Figure 2.1a

A second class, more relevant to this report, is that of contrastive learning algorithms. Contrastive learning aims to learn representations in which samples from the same class ("positives") are close to one another and samples from distinct classes ("negatives") are separated. The InfoNCE, is the most commonly used objective function for contrastive learning. It was introduced in [**143**], and is defined as follows:

$$\mathcal{L}_{\text{InfoNCE}} = -\mathbb{E}\left[\log \frac{f(\mathbf{x}, \mathbf{c})}{\sum_{\mathbf{x}' \in X} f(\mathbf{x}', \mathbf{c})}\right] \tag{2.1.2}$$

Here, $f(\mathbf{x}, \mathbf{c})$ is proportional to the ratio of the conditional distribution $p(\mathbf{x}|\mathbf{c})$ over a positive sample $\mathbf{x}$ to the marginal distribution $p(\mathbf{x})$ over negative samples $\mathbf{x}'$. The context vector $\mathbf{c}$ can be intuitively thought of as another positive sample. It is noteworthy that InfoNCE represents a lower bound in mutual information. By optimizing this lower bound, the mutual information between vectors $\mathbf{x}$ and $\mathbf{c}$ can be maximized [**143**]. In the absence of labels, SSL usually relies on the so-called "cross-view" approach, whereby random augmentations of the same image sample are assumed to be positive examples and augmentations from different images are assumed to be negatives [**28**].

Image representations obtained by contrastive methods achieve state-of-the-art performance on vision benchmarks [**28, 65, 21, 64, 20, 162, 9**]. However, the traditional view of contrastive learning using positive and negative pairs was recently challenged by methods like BYOL [**58**], SimSiam [**30**] and DiNO [**21**]. In these works, the authors managed to train visual representations using only positive pairs and/or a teacher-student paradigm. Especially noteworthy for this work is DiNO (Figure 2.1b), where the authors demonstrate how self-supervised ViTs naturally learn explicit information about the semantic segmentation of an image in their attention masks. In Duckie-Former, we exploit a ViT pretrained with DiNO to derive an instance segmentation model for navigation. Likewise, in One-4-All we rely on SSL to train the four neural components of our navigation pipeline.

2.1.5. Manifold Learning

The field of manifold learning provides an invaluable tool for representation learning based on the geometric notion of a manifold. A manifold is a topological space that is locally Euclidean in the vicinity of each of its points. Specifically, for every point on the manifold, there exists a topologically equivalent neighbourhood to an open unit ball in $\mathbb{R}^d$ [**74**]. Consider the earth as an example, locally each point can be represented as a 3D Cartesian coordinate but globally it is a sphere embedded in a 3D space.

Manifold learning for dimensionality reduction assumes data samples lie on a low dimensional manifold $\mathcal{M}$ embedded in a higher dimensional space. The goal is to map samples of $\mathcal{M}$ embedded in a high-dimensional space $\mathbb{R}^n$, to a lower dimensional space $\mathbb{R}^d$ (for $n >> d$), such that intrinsic relations between data samples in $\mathbb{R}^n$ are preserved in $\mathbb{R}^d$ [**15**]. A common instance of manifold learning is Principal Component Analysis (PCA) [**75**]. The PCA algorithm linearly projects high-dimensional data samples onto a set of orthogonal bases that explain the most variance in the data.

To address real-world data manifolds with non-linear correlations, manifold learning typically relies on the nearest neighbour graph approach. The goal is to optimize low dimensional embedding coordinates of the input such that these preserve properties (e.g., distance) of a neighbourhood graph computed with the original high-dimensional samples of $\mathcal{M}$ [**138, 14, 137, 34, 145**]. Moreover, neural networks are also exploited to infer low dimensional representations of data via reconstruction or variational cost functions [**117, 83, 116**].

In O4A, we take inspiration from Isometric Mapping (IsoMap) [**137**] to derive a geodesic regressor to estimate shortest path lengths between input image observations. IsoMap works by first creating a neighbourhood graph where each node represents a data point. The identification of neighbours and their pairwise distances is accomplished through k-nearest neighbours with an appropriate distance metric. Subsequently, a graph search method like Dijkstra [**37**] is employed to determine the shortest path length (geodesic distances) between each pair of nodes in the graph. Lastly, the data points are projected onto a lower dimensional space that preserves pairwise geodesic distances to the greatest degree possible. O4A estimates the neighbourhood graph using a local backbone trained over images with SSL and then produces a geodesic regressor trained on top of the embeddings of this local backbone.

2.2. Related Work

2.2.1. Classic Robot Navigation

Classic motion planning for robotics leverages the family of methods presented in Chapter 2.1.1. As an example, in the 2007 DARPA Urban Challenge, the MIT team used RRT to produce navigation paths [**86, 91**]. They employed RGB-Depth cameras with hand-crafted filters to do line detection on the road. To sample navigation paths, they relied on those road detections, a Velodyne laser scanner for collision detection and a prior map of the environment using a Road Network Definition File (RNDF).

Kala and Warwick [**76**] proposed sampling-based multi-agent planning using RRT where each agent path was smoothed with spline curves. Although appealing, the method was only tested in simulation with ground-truth information of the scene. Likewise in [**72**], the authors proposed an adaptive sampling scheme for PRM whereby samples are biased towards unexplored regions in the free configuration space $\mathcal{C}_{\texttt{free}}$. In this work, we do not rely on sampling based methods for navigation nor depth sensors, and our only prior knowledge of the environment are a set of RGB images collected offline.

Likewise, graph-based methods are exploited to navigate agents in the real-world. Dayoub *et al.* [**36**] relies on a LiDAR sensor to construct a 2D occupancy grid and plan over it by employing the Dijkstra algorithm. In [**42**], the authors proposed to enhance metric maps with semantic features using objects in the scene detected with SIFT [**95**]. Their method combines both semantic and metric features to produce a hybrid map for planning which is queried via graph search. Moreover, range and vision sensors can also be used to create a sparse map of the environment consisting of landmarks, which are queryable for navigation through graph search [**63, 13**].

Another common graph-based representation for planning is based on pose graphs, also called topo-metric. These can exploit RGB-Depth sensory information to construct a graph whose nodes contain pose information of the robot [**139**]. Navigation paths are retrieved from the graph through shortest path search [**141**]. The A* algorithm has equally been used by exploiting structured road maps to navigate the streets of Berlin [**17**] and on the DARPA Urban Challenge [**114**]. In contrast, O4A does not assume access to any map or pose information but instead builds an intermediate *topological graph* where nodes are solely composed of RGB images from previously visited states.

Framing navigation as an optimization problem has given rise to a variety of applications. Guivant *et al.* [**59**] retrieved a robot path by minimizing a cost matrix over the connectivity of local triangular regions of the environment. Likewise, Model Predictive Control (MPC) is commonly used to predict the future trajectory of the agent while optimizing control inputs and morphological constrains [**3, 96**].

To address local minima problems in optimization-based methods like APFs, strategies like random actions [**11**] or genetic algorithms [**142**] have been proposed. Bounini *et al.* [**18**] proposed to add negative potentials to previously visited states in the potential function to prevent local minima. In O4A (Chapter 4), we draw inspiration from Boubini and expand on the neural potential-like function presented in Duckie-Former (Chapter 3). Specifically, we define a latent potential function over images instead of a metric space and include negative potentials to each previously visited state to escape local minima.

As discussed previously, classical approaches often depend on a costly sensor suite to acquire depth information, which is then used to generate a metric map or perform collision checking. The exclusive use of metric representations is insufficient to capture the semantic notions of traversability. For example, in a field, tall grass can be traversed, whereas a wire fence cannot be traversed [**128**]. While metric representations may be augmented with semantic information through "hand-crafted" feature extractors [**95, 12, 119**], these methods are typically tailored to specific scenes and are susceptible to changes in the environment.

In this work, we exclusively employ inexpensive RGB cameras and demonstrate how to learn meaningful semantic representations for navigation using SSL. In Duckie-Former, we show how a neural monocular system can be used to navigate and avoid obstacles in a road while being executed on resource-constrained hardware. Likewise, O4A also works with RGB images but employs front and rear-facing cameras to address partially observability issues in monocular applications, thereby allowing the agent to escape from semantically similar surroundings, such as white walls or corridors in a building.

2.2.2. Learning-Based Robot Navigation

To overcome the limitations of classic approaches presented in Chapter 2.2.1, learning-based methods or experiential learning [**92**] have been used. These approaches have the advantage of being able to reason about the geometry of an environment, as well as the semantic aspects of traversability [**128, 147**]. Furthermore, they are designed to learn from high-dimensional data like images or point clouds, avoiding the need of "hand-crafted" feature extractors.

Reinforcement Learning (RL) agents are a prominent example of experiential learning and have been widely employed for image-based embodied navigation [**99, 149, 112**]. For instance, in GAPLE [**160**], the authors trained an end-to-end RL policy on top of both semantic and depth features extracted by a convolutional backbone. Furthermore, Yang *et al.* [**159**] demonstrated how navigation policies can be improved with semantic scene priors (e.g., an apple being likely inside a fridge). While these approaches provide a sound solution to the navigation problem, RL policies are known to be data-hungry, with some *offline* RL methods requiring up to 30 hours of training data [**126**]. In addition, end-to-end RL methods have limited navigation capabilities over long horizons [**41**].

To overcome the long horizon limitation, semi-parametric approaches use a topological memory encoded as a graph [**85**]. A topological graph aims to encode the topology of the environment, where nodes correspond to spatial locations and edges represent connections between them. This memory representation draws inspiration from research in neuroscience that establishes how human and animals achieve long-horizon navigation by relying on "memory maps" of the environment [**104, 53, 47**].

Various instances in the literature take advantage of a topological memory [**26, 8, 124, 151, 127, 45, 44, 97**]. The topological memory serves as a *global planner* to generate navigation waypoints towards a goal, which are then delivered to a *local policy* to produce locomotion commands for the agent. Hahn *et al.* [**61**] presented a self-supervised approach to construct a topological graph by leveraging SLAM for relative pose estimation. The graph was weighted with a geodesic estimator over RGB-D observations, and an RL policy was used for actuation. In O4A, we rely on a self-supervised objective that leverages one-step temporal distance between samples, thereby eliminating the need for depth or a computationally expensive SLAM pipeline to infer pose targets.

Despite the benefits of semi-parametric approaches, they suffer from two primary limitations. The first is that spurious connections in the graph can adversely affect the planner's ability to derive a feasible plan (e.g., connections that allow warping between obstacles). The second is that these methods do not scale well in terms of memory as the number of vertices and edges in the graph increases with the size of the environment.

These limitations are partially addressed by hand-crafted heuristics. Examples of these include connectivity thresholds to prune the number of spurious edges [**124, 45**], node sparsification strategies [**8, 44**] and lifelong updates to the graph [**151**]. However, these methods end up adding a non-negligible number of parameters that require tuning and are often environment dependant. In contrast, O4A avoids the need for such heuristics by simply training a connectivity classifier on top of a backbone trained to do contrastive learning.

Semi-Parametric Topological Memory (SPTM) [**124**] and Visual Navigation with Goals (ViNG) [**127**] are of particular interest for the assessment of this work. SPTM employs a classifier to determine the connectivity between two temporally close images and creates an unweighted graph based on those. This graph does not accurately reflect the distance between two close or distant samples as it is unweighted, exacerbating the spurious edges issue. Similarly, ViNG uses a classifier to estimate the temporal number of steps between samples and weight the edges of the graph with these estimates. Planning is performed over the graph and a relative pose predictor is paired with a PD controller to navigate waypoints.

Both methods employ pruning and sparsification strategies. Conversely, our method O4A does not prune the graph and directly estimates the action between two samples, allowing us to predict connectivity and local controls simultaneously.

Furthermore, SPTM and ViNG use all training images to create a graph and generate navigation plans using Dijkstra's algorithm [**37**]. However, employing all images during graph construction is indeed doomed to pose memory limitation. Wiyatno *et al.* [**151**], proposed a solution that samples the dataset of observations, but their method requires pose information and tuning of multiple hyper-parameters. In contrast, O4A uses latent codes to weight a training graph, which is then embedded in a shortest path lengths regressor. This approach overcomes the memory constraints associated with graph-based methods and allows the estimation of geodesic distances over seen an unseen images, as opposed to a fixed set defined on a graph.

There has also been research on alternative memory representations for embodied navigation, including the use of semantic [**81, 51, 110**], spatial [**60, 68, 23**] or latent representations [**46**]. For instance, LTM [**51**], proposes a framework to estimate top-down semantic maps outside the field of view of the agent to address long horizon goals. To the best of our knowledge, none of these representations exploit a training graph to obtain an image-based geodesic regressor for navigation.

O4A draws significant inspiration from Plan2Vec [**157**], which is a pure planning method. Plan2vec uses its geodesic regressor (referred to as global metric) as a heuristic for graph search. However, the authors only showcased Plan2vec in simulation environments and did not address robot actuation in their work. In contrast, O4A tackles planning and navigation simultaneously and uses the geodesic regressor in a potential function to frame navigation as a minimization problem [**32**].

Lastly, it is worth noting that several existing methods rely on ground-truth pose information to learn temporal distances or relative pose between samples [**8, 151, 127, 97**]. Both of our methods, DF and O4A, leverage SSL and are trained using only raw RGB image samples and actions. As a result, our methods are not dependent on ground-truth pose information, making them compatible with low-cost camera sensors and readily scalable. In Table 2.3 we present an overview of various learning-based methods on the literature and how O4A differ from those.

Method	Plan2Vec [157]	LTVN [151]	SPTM [124]	InvDM [112]	ViNG [127]	NRNS [61]	LTM [51]	SMT [46]	O4A
Memory Representation	Graph	Graph	Graph	Value function	Graph	Graph	Semantic map	Latent	Neural
Path Planner	Graph search	Graph search	Graph search	End to end	Graph search	Graph search	Goal policy	End to end	Potential function
Training	SSL	Supervised	SSL	RL based	Supervised	Supervised	Supervised	RL based	SSL
Requires Pose	No	Simulator & wheel encoders	No	No	Wheel encoders & co-training	From SLAM	Simulator	Simulator	No
Uses Depth	No	Yes	No	No	No	No	Yes	Yes	No
Pruning Strategy	None	Sampling & probabilistic updates	Defines max. number of edges	None	Edge threshold	None	None	None	None
Action Space	None	Continuous	Discrete	Discrete	Continuous	Continuous	Discrete	Discrete	Discrete
Locomotion	None	iLQR controller	Inverse dynamics	RL	PD controller	Inverse dynamics	RL	RL	Inverse dynamics
Interpretable Representation	Yes	No	No	No	No	No	Yes	No	Yes
Real World Deployment	No	Yes	No	No	Yes	No	No	No	Yes

Table 2.3. Comparison of learning-based methods for visual navigation. It should be noted that this table excludes DF as O4A, our main approach, is an enhanced version of it.

Chapter 3

Duckie-Former

In this section we present our first work called Duckie-Former (DF), a vision-based system for monocular embodied navigation. The main objective of this section is to demonstrate how self-supervised pre-trained models can be used to produce a simple yet effective potential-fields-based controller for navigation. We begin by defining the navigation problem that we are addressing in Chapter 3.1. We then present our approach and its components in Chapter 3.2. Subsequently, we provide an evaluation of our segmentation model in Chapter 3.3.1, and analyze the navigation abilities of the agent in two tasks: lane following and obstacle avoidance in Chapter 3.3.2. We conclude this section discussing limitations of our method in Chapter 3.4, and introduce One-4-All as an enhancement over Duckie-Former.

3.1. Problem Formulation

We consider a differential drive robot driving at a constant linear speed on a road with a continuous action space composed of its angular velocity $\mathcal{A} = \omega$ for a visual servoing navigation task. Particularly, assume two common driving tasks on a road: lane following and obstacle avoidance. Giving prior knowledge of the environment like object-classes in the scene and the color of road lines, we assume we can estimate a segmentation masks over those from raw RGB images.

When the agent is prompted with an RGB image o_t, it has to either stay on the road without drifting from it or avoid obstacles depending on the task at hand. These tasks must be seamlessly achieved using only the current image observation o_t and its segmentation mask z_t.

3.2. Method

In this section, we take inspiration from Artificial Potential Fields and showcase how to navigate and control a robot by simply constructing a repulsive potential around objects of interest in the scene.

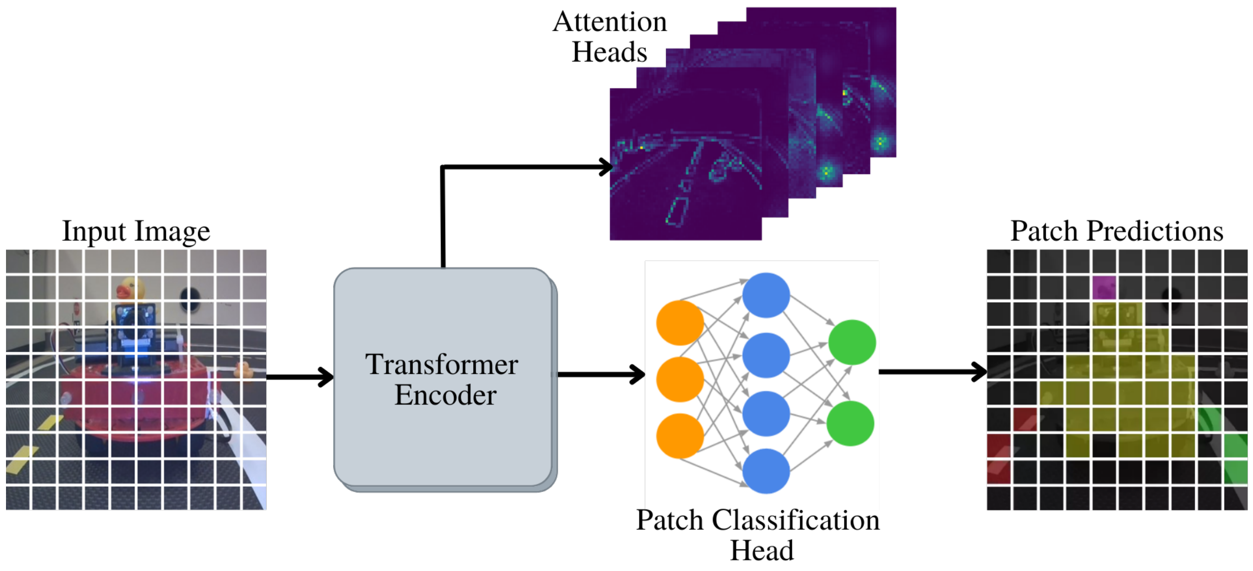


Fig. 3.1. Coarse semantic segmentation using ViTs. We encode image patches using a vision transformer as encoder and then predict a class label for each patch with a shallow fully connected network. We visualize attention heads to assess the effect of training in the transformer encoder.

3.2.1. Coarse Semantic Segmentation with Vision Transformers

Our approach is based on the following hypothesis: an agent can successfully and safely navigate an environment with low resolution segmentation masks. Therefore, we propose training a neural classifier g_ϕ to predict a class label $z_t^i \in \{1, 2, \dots, K\}$ for every image patch $o_t^i \in \mathbb{R}^{p \times p}$ indexed with i. Our classifier is a fully-connected network that we apply over ViT patch encodings $y_t^i = \texttt{DiNO}(o_t)^i \in \mathbb{R}^d$ to predict patch labels (Equation 3.2.1). The coarse segmentation mask $z_t \in \mathbb{R}^{n \times n}$ is obtained by concatenating all patch predictions in a grid (Figure 3.1).

$$z_t^i = g_\phi(y_t^i) \tag{3.2.1}$$

To illustrate the process of obtaining patch-labels, consider a ViT with encodings of dimensionality d, applied to an image with a resolution of 480p[1]. This results in an encoding map with dimensions of $60 \times 60 \times d$ assuming a patch size of 8 pixels ($480/8 = 60$). To obtain

[1]For the rest of this work, p will stand for pixels.

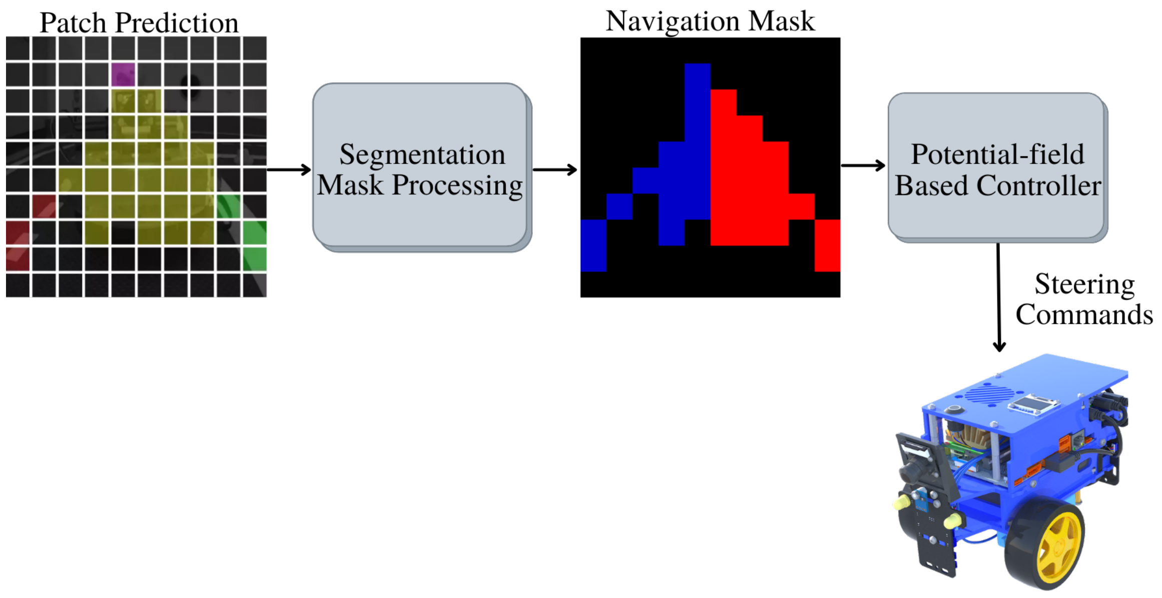

Patch Prediction
Segmentation Mask Processing
Navigation Mask
Potential-field Based Controller
Steering Commands

Fig. 3.2. Potential-fields-based controller for lane following and obstacle avoidance. The coarse segmentation mask is used to compute a left (blue) and right (red) mask which are delivered to a potential-fields-based controller. The controller receives the mask and maps it as a "repulsive" potential to steer away from the half of the image with the most obstacle patches (Equation 3.2.2).

a label for each patch during training, we downsample the original ground-truth mask accordingly using nearest neighbour interpolation. Our simple design avoids up-sampling architectural components and allows us to reuse pretrained weights from any vanilla CNNs/ViTs architectures, such as DiNO [**21**].

As discussed in Chapter 2.1.3, our motivation for using ViTs is twofold: first, we hope to leverage the ability of transformers to learn long-range dependencies in an image, which is an appealing proposition for real-time navigation and control tasks. Second, as with convolutional layers, a trained ViT can run segmentation on images at various resolution, as long as the patch size is the same. This implies that a ViT trained for coarse segmentation of 8×8 patches will yield 30×30 predictions for 240p images, 60×60 predictions for 480p images, 120×120 predictions for 960p images and so on. One can therefore adjust the granularity of the prediction and the associated computational load by simply downscaling or upscaling input images: a welcome flexibility for deployment on embodied agents with hardware constraints. ViTs can be trained and fine-tuned at different resolutions [**40**] and more advanced techniques for improving the performance at different test resolutions is a matter of active research [**155**].

3.2.2. Navigation With a Neural Potential Function

We borrow inspiration from APFs and propose a potential-fields-based strategy to navigate and control the robot on the environment. Specifically, this function consists solely of a negative or “repulsive” potential that enables the agent to avoid obstacles and stay on the road. To this end, we leverage the coarse segmentation mask $z_t \in \mathbb{R}^{n\times n}$ for the current $n \times n$ patches extracted from observation o_t. We then compute a navigation mask $m_t \in \{0,1\}^{n\times n}$ by identifying the pixel patches corresponding to line patches and obstacles. This mask is used to compute a “repulsive” potential pushing the agent to steer away from the half of the image with the most line patches or obstacles. More formally, the angular speed is controlled based on

$$\omega_{t+1} = \omega_t - p^-(U, m_t, \gamma) = \omega_t - \gamma \sum_i^{n^2} \text{vec}(U \odot m_t)^i \tag{3.2.2}$$

where the matrix $U \in \mathbb{R}^{n\times n}$ is a sign mask with -1 values in the first $\frac{n}{2}$ columns (left) and -1 in the lasting columns (right). The parameter γ is a weighting factor, $\text{vec}(\cdot)$ transform a matrix into a vector by stacking its rows and $\odot$ denotes the Hadamarad product. This potential-fields-based controller will reach an equilibrium by keeping the same energy on the right and the left, i.e., by being centered between the white and yellow lines of the road in the absence of obstacles. We present a visualization of the controller in Figure 3.2.

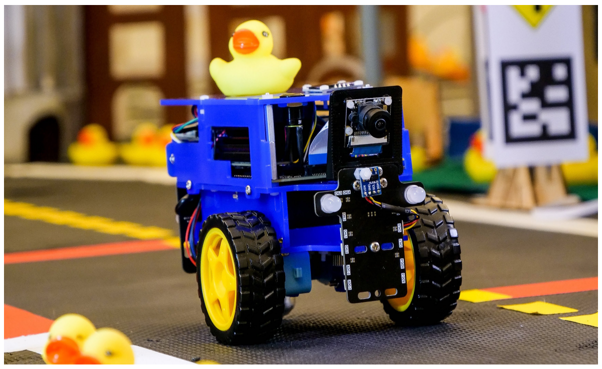
While our navigation strategy is not a one-to-end instance of APFs, it shares similarities with this approach. Specifically, Equation 3.2.2 is designed to guide the robot away from the region with the largest negative potential, denoted as p^-. This objective can be understood as an attempt to optimize (i.e., finding an equilibrium) the value of p^- by steering the robot in the opposite direction of the obstacles.

3.3. Experiments

3.3.1. Image Segmentation

3.3.1.1. Data. Our dataset is composed of RGB images gathered using the on-board camera of our Duckiebot (Figure 3.3a). We collected images by teleoperating the robot in the data collection scene presented in Figure 3.3b. To increase the diversity in the training scene, various objects were added on the road during the episode. A total of 100 images were sampled from the recordings and labeled with 7 classes: Duckiebot, duckie, white lane marking, yellow lane marking, road sign and human hand. We use a 70-15-15 split for training, validation and testing.

3.3.1.2. Implementation Details. We use a standard ViT architecture [**40**] and pretrained weights from DiNO for our perception backbone. Specifically, the ViT-S/8 architecture


(a) DB21J Robot.

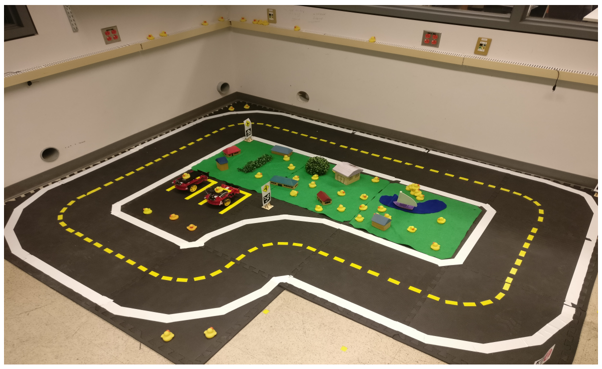
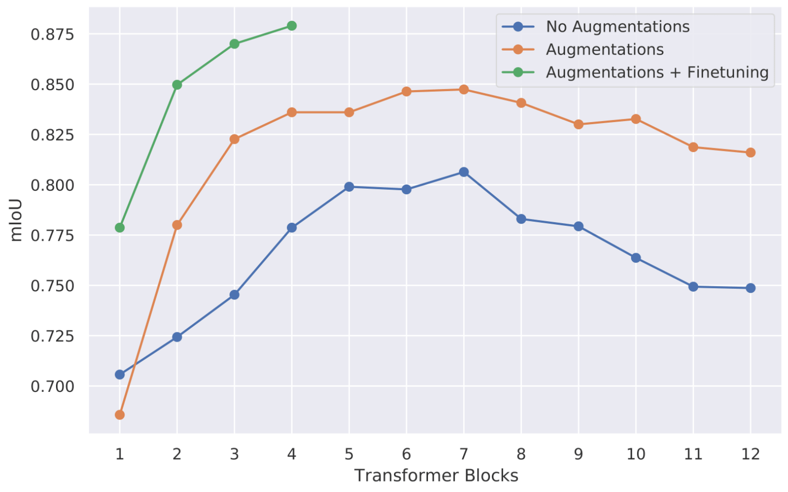

(b) Data collection scene.

Fig. 3.3. Experimental setup for Duckie-Former. a) Example of the differential drive DB21J Duckiebot model used in our experiments. While other sensors are available, we only used the camera for navigation. b) We collected 100 images by teleoperating the robot on the scene and occasionally adding various objects on the road.

consists of 12 transformer blocks—a block includes a self-attention layer followed by 2 fully-connected layers—and takes 8×8 patches as input. As with standard ViTs architectures, the size of the feature map is constant throughout layers and any number of blocks will output predictions at the same resolution. Each patch is encoded as a 384-dimensional vector which we classify using a fully-connected segmentation head. In Figure 3.4, we vary the number of transformer blocks in the backbone as well as the augmentations applied to the training data in order to study segmentation performance over the test set. Encouragingly, we find that using only a few transformer blocks is sufficient to achieve good performance. We further observe that metrics plateau or even deteriorate with a backbone deeper than 5 blocks. Moreover, standard data augmentations and fine-tuning the backbone still appear necessary to maximize performance. The bigger part of the fine-tuning improvement is observed in the road sign class, which was not heavily featured in our dataset[2].

For training, we use 480p images, a batch size of 1 image (3600 8×8 patches), and train for 200 epochs. The validation set is used for checkpointing and the model with the best balanced validation accuracy is retained. When training the segmentation head only, we use the Adam optimizer with a learning rate of $1e^{-3}$. For finetuning the backbone, we continue training for 200 epochs using the AdamW optimizer with a learning rate of 1^{e-4}. We use standard data augmentations where mentioned: random crops, flips, shifts, scales, rotations, color jittering and Gaussian blur. Reported Intersection Over Union (IoU) and Accuracy (Acc) scores are with respect to the downscaled ground-truth masks, e.g., 480p predictions are benchmarked against the 60x60 interpolated masks to reflect patch-level performance.

[2]All the architectures were implemented using Pytorch (`https://pytorch.org/`).



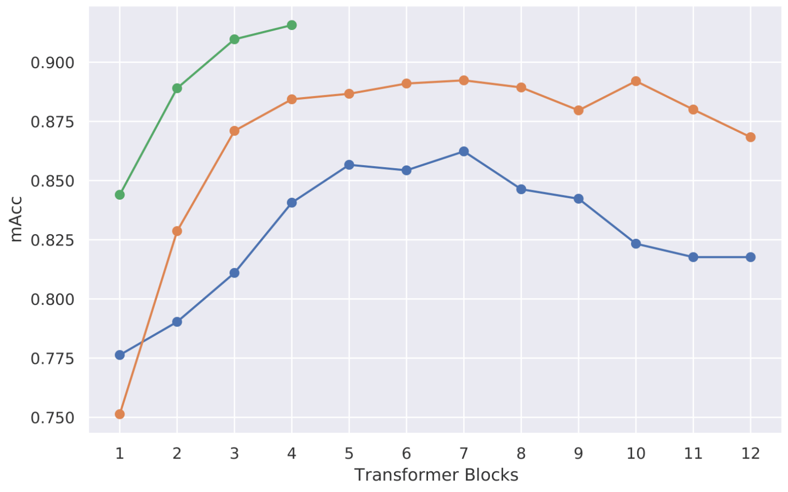


Fig. 3.4. Intersection Over Union (IoU) $\frac{1}{N}\sum_j(\frac{\texttt{Prediction Area Overlap}}{\texttt{True Area Union}})_j$ and Accuracy (Acc) $\frac{1}{N}\sum_j(\frac{\texttt{Correct Sample Prediction}}{\texttt{Total Samples}})_j$ of various segmentation models on our Duckietown segmentation test dataset. Metrics are averaged across 3 seeds, hence the "m" on IoU and Acc. The DiNO architecture consists of 12 transformer blocks: we therefore probe the intermediary patch representations by training a segmentation head at various depths using a ViT backbone of b blocks. While raw patch representations (**"No Augmentations"**) perform reasonably well, adding standard image augmentations (**"Augmentations"**) is hugely beneficial, despite the self-supervised DiNO pretraining. In both the "No Augmentations" and "Augmentations" setup, the ViT backbone parameters are frozen and we only train the segmentation head. As expected, unfreezing the backbone and continuing training (**"Augmentations + Finetuning"**) increases both performance metrics and is particularly beneficial for the 1-block and 2-block backbones. We could not finetune backbones with more than 4 transformer blocks due to hardware constraints.

3.3.1.3. Inference Results. In Table 3.1, we study the inference speed and quality of the finetuned 1-block and 3-block ViTs segmentation models from the previous section. For comparison, we also study backbones built with the first layers of a DiNO-pretrained ResNet-50 CNN architecture to perform the same task. We benchmark the same models at different

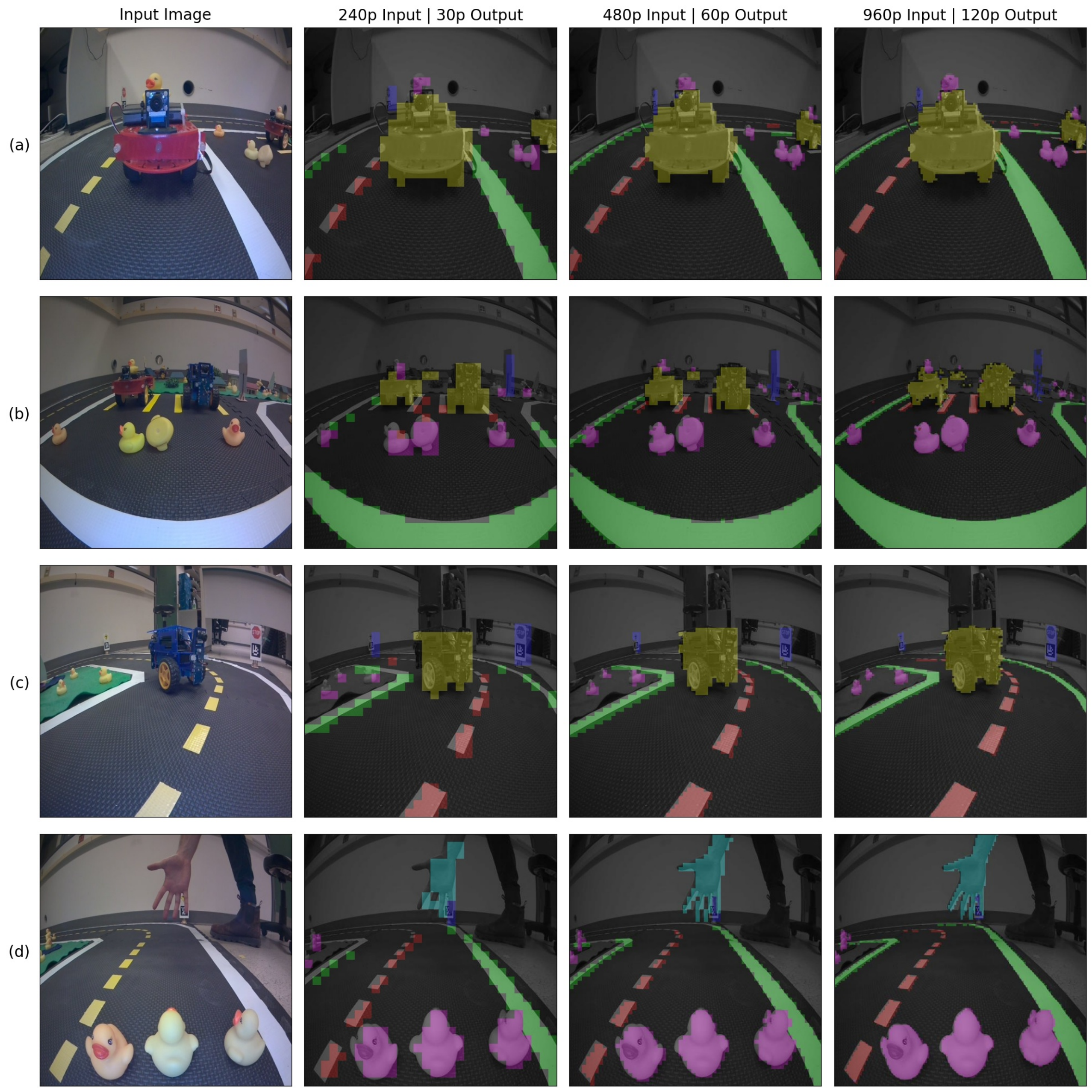


Fig. 3.5. Predictions of the **same 3-block ViT** at different resolutions. While the model was trained in the 480p regime, it performs well on downscaled (240p) or upscaled (960p) images. The 240p predictions are visually coarse, but accurate for nearby objects or large distant ones. We show in Chapter 3.3.2.1 and 3.3.2.2 how 240p and 480p predictions can be used for navigating a Duckietown environment. The 960p predictions are shown for illustrative purposes and are too slow for real-time navigation.

inference resolutions and find that ViTs perform relatively well even on downscaled images. Importantly, all models can run inference at a reasonable framerate on GPU for 240p or 480p input. CPU inference at 240p would even be conceivable. All benchmarks were run on a *11th Gen Intel Core i7-11800H @ 2.30GHz × 16* CPU and a *NVIDIA GeForce RTX 3050* Laptop GPU. Contrary to the memory efficient CNNs, the ViT 960p resolution could not fit into GPU memory due to the unwieldy amount of patches.

Model	Parameters	Input Resolution	Output Resolution	Patches	CPU Inference (Im./sec) ↑	GPU Inference (Im./sec) ↑	GPU RAM (GB) ↓	mIoU ↑	mAcc ↑
ViT (1 block)	2.2M	240x240	30x30	900	**58**	**168**	**0.10**	0.71	0.79
		480x480	60x60	3,600	6	47	0.75	**0.78**	**0.85**
		960x960	120x120	14,400	0.5	OOM	OOM	0.75	0.83
ViT (3 block)	5.8M	240x240	30x30	900	**23**	**131**	**0.11**	0.76	0.84
		480x480	60x60	3,600	2	29	0.93	**0.86**	**0.90**
		960x960	120x120	14,400	0.2	OOM	OOM	0.84	0.89
CNN (24 layers)	1.6M	240x240	30x30	900	**61**	**214**	**0.03**	0.67	0.77
		480x480	60x60	3,600	12	103	0.08	**0.79**	**0.85**
		960x960	120x120	14,400	4	50	0.16	0.71	0.80
CNN (32 layers) +ConvTranspose	7.1M	240x240	30x30	900	**40**	**183**	**0.06**	0.66	0.75
		480x480	60x60	3,600	9	78	0.16	**0.82**	**0.88**
		960x960	120x120	14,400	3	38	0.30	0.75	0.83

Table 3.1. Quantitative assessment of segmentation quality and inference speed in Duckie-Former. "OOM" indicate models that exhaust their available memory during inference and Im./sec stands for number images processed per second. Although the 3-block ViT demonstrated the best segmentation results over the test set, CNNs have superior inference speed and are unaffected by OOM issues as they do not rely on the attention mechanism. The attention mechanism possesses a time and space complexity that scales quadratically with the number of input tokens, therefore, yielding performance issues with high-resolution images.

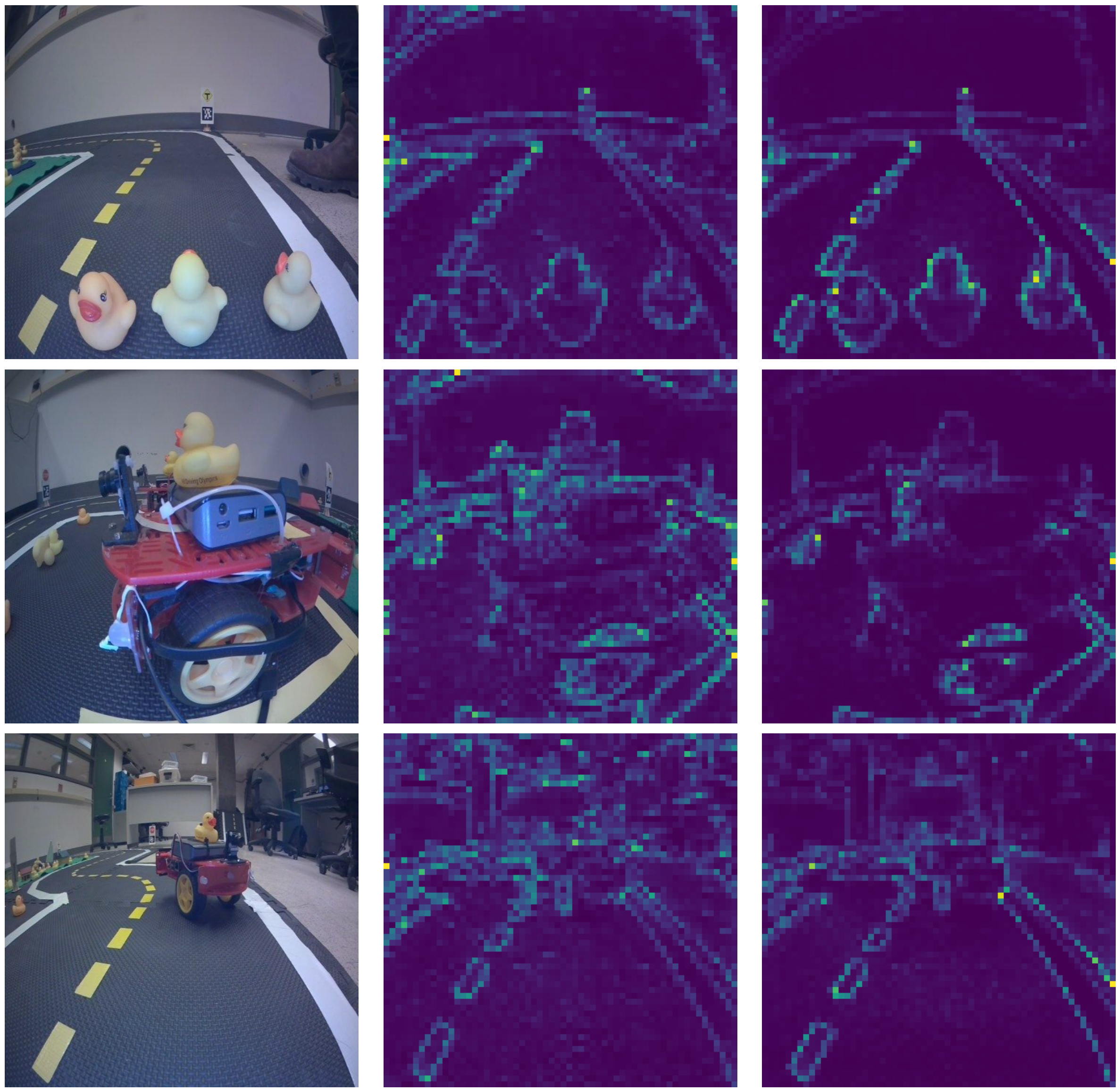

(a) Input Image. **(b)** DiNO. **(c)** Finetuned.

Fig. 3.6. Segmentation masks of the 3-block ViT pretrained with DiNO compared with a finetuned version of itself. Results demonstrate how the attention masks of the pretrained model attend to a general variety of objects in the scene like the desk in row three. Interestingly, the attention masks of the finetuned model attend primarily to the objects that belong to one of the classes in our dataset, ignoring objects that are not relevant to our task.

Figure 3.5 provides further visualization of the predictions made by our ViT and how prediction granularity can be enhanced with higher-resolution images while using the same ViT model. Moreover, Figure 3.6 compares the attention masks of the pretrained 3-block model to the finetuned one. Interestingly, the finetuned model learns to focus predominantly on the objects that are relevant to our task, even when trained with only a few annotated images.

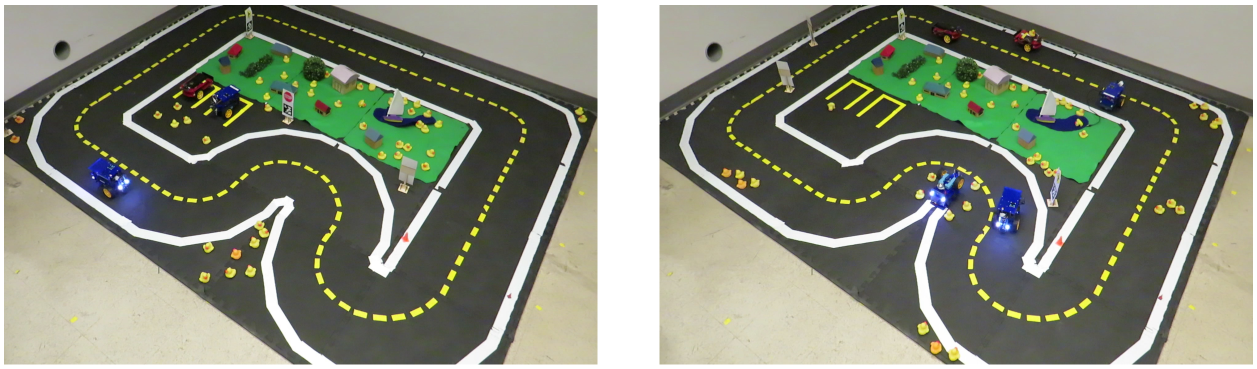

(a) Lane Following scene. **(b)** Obstacle avoidance scene.

Fig. 3.7. Visual navigation scenes used to assess DF. a) The robot is tasked with completing laps without crossing the yellow line (left) or white line (right). We benchmarked both the outer and inner loops. b) The robot needs to complete laps without colliding with on-road obstacles. For this experiment, the controller ignores the yellow line predictions and the robot can navigate both white lanes to avoid obstacles. In total, 4 duckiebots, 2 signs and 4 groups of duckies must be avoided during a loop.

3.3.2. Navigation

In this section, we assess the performance of the trained 1-block and 3-blocks ViTs from Table 3.1 in two different monocular visual servoing tasks: lane following and obstacles avoidance. The robot is controlled using the potential-fields-based controller presented in Chapter 3.2.2. It receives as input the segmentation output produced by a trained ViT and outputs steering commands for the robot while maintaining a fixed linear velocity as shown in Figure 3.2. For the lane-following task, the goal is to maintain the vehicle centered in a lane while in the obstacle avoidance task, the agent can use both lanes to navigate and avoid obstacles[3].

The scenes for both experiments are modified from the original data collection scene. In Figure 3.7a, we show the driving environment used for lane-following where an additional U-turn is added to measure the robustness of the driving agent. The same driving scene is reused for the obstacle avoidance track but different objects are added on the road (Figure 3.7b).

3.3.2.1. Lane Following. The objective of this task is to navigate the vehicle on the road (without obstacles) and maintain it centered between the yellow and white lines. To accomplish this, we leverage the segmentation mask and negative potential presented in Chapter 3.2.

Each model is evaluated by navigating the agent for five loops (2 outer loops and 3 inner loops) and report the number of minor and major infractions. A minor infraction occurs when the robot steps over either the white or yellow line. A major infraction is defined as any event

[3]For demonstrative videos of our method see `https://sachamorin.github.io/dino/`

Model	**Input Resolution**	**Outer Loop**	**Inner Loop**	**Minor Infractions ↓**	**Major Infractions ↓**
Baseline	480p	✓		7	**0**
			✓	11	2
ViT 1 block	240p	✓		**1**	**0**
			✓	**3**	**0**
	480p	✓		**1**	**0**
			✓	4	1
ViT 3 blocks	240p	✓		**1**	**0**
			✓	4	3
	480p	✓		**1**	**0**
			✓	8	**0**

Table 3.2. Lane following results obtained with DF.

requiring human intervention to put the agent back on track. Both models are evaluated at 240p and 480p input resolutions and compared against the standard lane following system implemented in Duckietown, which consists of HSV filters alongside a histogram filter for state estimation [**139**].

The results of the lane following evaluation are reported in Table 3.2. Both ViTs perform equally well in the outer loop, however, the inner loop proves more challenging. The best performing model was with 1-block at 240p with a total of three minor infractions and zero major infractions. We hypothesize this good performance is owed to the high throughput of the model, which allows for better reaction time in the controller. Surprisingly, the high-capacity 3-block model reports a higher number of infractions. The baseline model is the worst-performing whose result is a likely consequence of the HSV filters producing false positives line detections when white or yellow objects are placed in the scene (see Figure 3.7a).

These results demonstrate that even though higher capacity models perform better than low capacity ones with respect to test segmentation metrics, more variables should be considered for real-world deployment. Additionally, while the coarse segmentation predictions (see Figure 3.5 column 2) lower the resolution to increase inference speed, they still appear to hold enough information to accurately navigate the agent within the environment using our potentials-based method.

3.3.2.2. Obstacle Avoidance. The objective for the obstacles avoidance experiment is to navigate the agent through the scene in Figure 3.7b while avoiding on-road duckies, signs and Duckiebots. This task was designed to validate the performance of the agent within a more challenging environment, and make use of predictions for all classes. We define the area between both white lines as being drivable, i.e., the agent is allowed to cross the yellow

ViT Blocks	Input Resolution	Minor Infractions ↓	Major Infractions ↓
1	240p	2	**0**
	480p	**1**	2
3	240p	**1**	4
	480p	3	**1**

Table 3.3. Obstacles avoidance results obtained with DF.

line to avoid obstacles without penalty. Particularly, the navigation and control strategy is the same used in the previous experiments with the difference that the navigation mask now includes all obstacles to be avoided. Therefore, pixels classified as white line, ducks, signs and duckiebots are actively contributing to the "repulsive" potential. This potential function encourages the agent to drive on the road while avoiding objects placed on it. The assessment is similar to that of Chapter 3.3.2.1 (same number of loops) with the difference that driving over the yellow line is permitted and small contacts with obstacles are considered minor infractions.

We evaluate the same models as in the previous subsection (1-block, 3-block, at 240p and 480p). For this benchmark, we do not have a particular baseline in the Duckietown stack to compare with. The results are presented in Table 3.3 and are consistent with the ones in Table 3.2. The best performing model was once again 1-block with an input resolution of 240p, reporting a total of two minor infractions and zero major ones. The higher-capacity model performed well but did not surpass the 1-block ViT even though the segmentation results produced by this are of superior quality, again suggesting that visual servoing benefits from the higher framerate of the shallow 1-block backbone.

3.4. Discussion

In this Chapter, we have explored the potential benefits of using ViTs pre-trained through SSL for embodied visual navigation. We were able to train an end-to-end model using only 70 annotated images. This demonstrates the effectiveness of our approach, highlighting the benefits of neural-based representations for visual navigation. We also showcases how a simple yet effective potential-fields-based controller can be readily adapted to navigate on a road and perform different navigation manoeuvres on it. In contrast to standard linear probing in SSL, which typically favors large high-capacity models, we found that small models with high-throughput may be more beneficial for embodied applications. Additionally, we highlighted how ViTs can adjust their inference resolution based on available resources and be used in real-world robotics applications, depending on the required precision of the embodied agent.

Nevertheless, our approach operates on 8x8 image patches rather than pixels, and is not well-suited for predicting high-resolution segmentation masks, in which case an encoder-decoder architecture should be preferred. While the low resolution of our predictions may not hinder navigation, it may not be appropriate for safety-critical applications where the consequences of collisions are severe. Furthermore, our approach is inspired by APFs, but it is limited to operate only in a looping road-like scenario where obstacles in the scene and line colors have to be known in advance. In the next chapter, we will build upon the potential-fields-based approach developed in this section and propose a self-supervised fully-parametric approach for image-goal navigation, which we call One-4-All.

Chapter 4

One-4-All

In this section, we present One-4-All (O4A), a self-supervised fully-parametric approach for image-goal navigation. O4A builds upon Duckie-Former (Chapter 3), and it seamlessly achieves navigation by minimizing a neural potential function. Moreover, O4A is completely trained via SSL and does not require any labeled information unlike DF. We begin by defining the image-goal navigation problem in Chapter 4.1. Then, we present O4A in Chapter 4.2 where we discuss the data assumptions in Chapter 4.2.1, the various components of the system in Chapter 4.2.2 and the navigation pipeline in Chapter 4.2.3. We provide an extensive evaluation in Chapter 4.3 in three different settings: a simulated maze environment (Chapter 4.3.1), a 3D photorealistic simulator (Chapter 4.3.2), and a real-world environment (Chapter 4.3.3). Ultimately, we conclude this section discussing O4A and its limitations in Chapter 4.4.

4.1. Problem Formulation

We consider a robot with a discrete action space for an image-goal navigation task [**2**]. The action space is defined as $\mathcal{A} = \{\texttt{STOP}, \texttt{FORWARD}, \texttt{ROTATE\_RIGHT}, \texttt{ROTATE\_LEFT}\}$. Using our knowledge of the robot's geometry and an appropriate exteroceptive onboard sensor (e.g., a front laser scanner), we assume that the set of collision-free actions $\mathcal{A}_{free}$ can be estimated.

When prompted with a goal image o_g, the agent should navigate to the goal location in a partially observable setting using only RGB observations o_t and the $\mathcal{A}_{free}$ estimates. The agent further needs to identify when the goal has been reached by *autonomously* calling the `STOP` action in the vicinity of the goal [**2**].

4.2. Method

4.2.1. Data Trajectories

We aim to achieve image-goal navigation using learned modules parameterized by deep neural networks. For any given environment, we assume that some previously collected observation trajectory $\tau_o = \{o_t\}_{t=1}^{T}$ and corresponding actions $\tau_a = \{a_t\}_{t=1}^{T}$ are available. We consider a single trajectory from a single environment for notational conciseness, but in practice use multiple data trajectories (Figure 4.1) from different environments. We do not require an expert data collection policy and the dataset could be the product of teleoperation, self-exploration or random walks, as long as it sufficiently covers the free space $\mathcal{C}_{\texttt{free}}$ of the environment. As discussed in Chapter 2.2, we tackle navigation in an unsupervised setting and do not assume access to pose estimates for each image observation, which greatly simplifies data collection. Moreover, we do not collect any depth measurement, and only rely on a front laser scanner at runtime for simple collision checking.

4.2.2. System

4.2.2.1. Overview. We illustrate and present an overview of our system in Figure 4.1. We first rely on self-supervised learning (Chapter 2.1.4) to learn an RGB backbone paired with a connectivity head to infer a graph over all images in τ_o. The graph will then be used to derive training objectives for a forward dynamics module and a geodesic regressor. We finally show how to navigate the trained system in Chapter 4.2.3.

4.2.2.2. Local Backbone. The local backbone learns a mapping from raw images to low-dimensional latent codes $h : \mathcal{O}_{RGB} \to \mathcal{X}$. For simplicity, we will denote extracted features as $x = h(o)$. The function h will serve a dual purpose: 1) to extract low-dimensional features in $\mathcal{X} = \mathbb{R}^n$ that will be used as input for other modules, and 2) to learn a **local metric** defined as

$$d_h(x_t, x_s) = \|x_t - x_s\|_2 \tag{4.2.1}$$

between pairs of observations. Given the lack of pose information in the training data, h is trained via self-supervised learning using the siamese loss [**157**]

$$\begin{aligned} \mathcal{L}_h(x_t,\ x_{t+1}, \mathcal{N}) = (m_+ - d_h(x_t, x_{t+1}))^2 \\ + \frac{1}{|\mathcal{N}|} \sum_{x^- \in \mathcal{N}} \max(0, m_- - d_h(x_t, x^-))^2, \end{aligned} \tag{4.2.2}$$

where $m_+,\ m_- \in \mathbb{R}^+,\ m_+ < m_-$ are positive and negative margins, respectively, and $\mathcal{N}$ is a set of random data from τ_o, which are used as so-called negatives. Equation 4.2.2 is

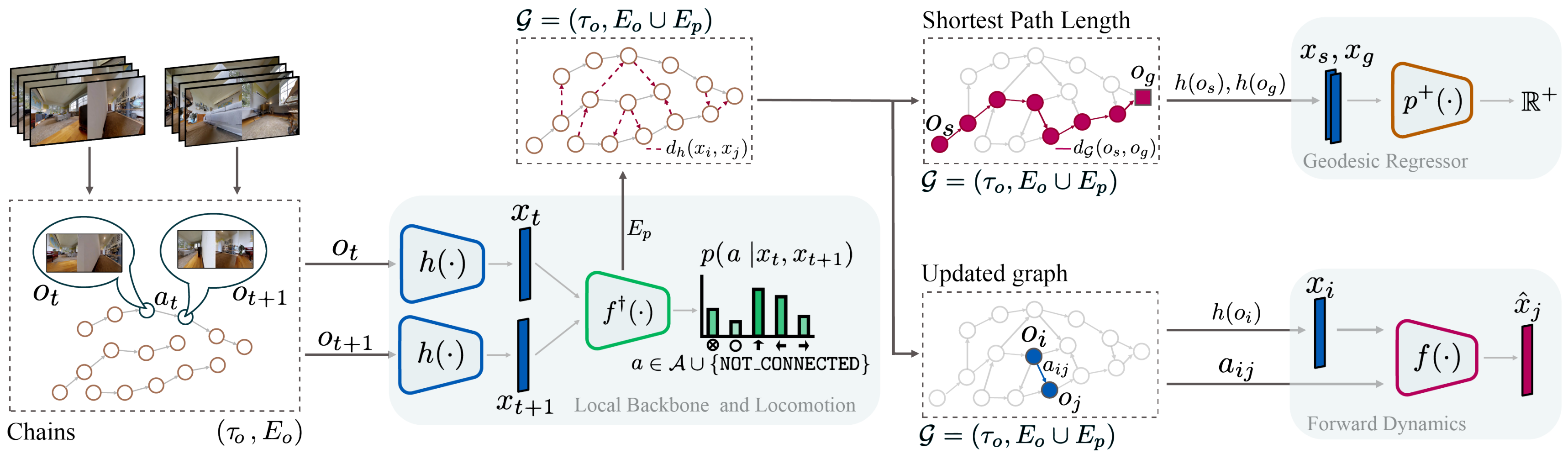


Fig. 4.1. O4A consists of 4 learnable modules for image-goal navigation. Learning is entirely achieved using previously collected RGB observation trajectories $\tau_o = \{o_t\}_{t=1}^{T}$ and corresponding actions $\tau_a = \{a_t\}_{t=1}^{T}$, without pose. The **local backbone** h (left) takes as input RGB images to produce low-dimensional latent codes $x \in \mathcal{X}$. The **locomotion head** $f^\dagger$ (center) uses pairs of latent codes to predict the action required to traverse from one latent code to the other (order matters), or the inability to do so through the `NOT_CONNECTED` output. h and $f^\dagger$ are then used to construct a directed graph $\mathcal{G}$, where nodes represent images and edges represent traversability. The **forward dynamics head** (bottom right) f is trained using edges from $\mathcal{G}$ to predict the next code x_j given the current code x_i and an action $a_{ij} \in \mathcal{A}$. The geometry of the graph $\mathcal{G}$ is embedded in a neural network using a **geodesic regressor** p^+ (top right), which outputs the shortest path length for any pair of codes. Once all the modules are trained, $\mathcal{G}$ can be discarded, and p^+ be used as part of a potential function, as illustrated in Figure 4.2 and detailed in Equation 4.2.5.

an instance of time contrastive learning: consecutive observations (positive pairs), which we know to be indeed close in terms of pose, are encouraged to be at a distance of exactly m_+ in $\mathcal{X}$. Negatives are pushed to be at least at a distance of m_-, reflecting the fact they should not share the same neighbourhood even if the exact distance between them is unknown at this stage. This latest observation motivates the term “local metric” [**157**], since the actual distance d_h is only informative when applied to positive pairs that are close in latent space. It should be stressed that d_h cannot predict how far negative pairs are apart in general, as it tends to saturate around m_- as discussed in [**157**]. Furthermore, in contrast to the coarse segmentation module in Duckie-Former (Chapter 3), the local backbone does not necessitates any annotated training instance.

4.2.2.3. Locomotion Head. The component $f^\dagger : \mathcal{X} \times \mathcal{X} \to \mathcal{A} \cup \{\texttt{NOT\_CONNECTED}\}$ predicts the action needed to travel between two latent codes, or returns the `NOT_CONNECTED` token when the transition is deemed not feasible in a single action. $f^\dagger$ therefore acts as both a loop closing module and an inverse dynamics predictor. It is trained using the standard cross-entropy loss on the actions observed in τ_a. We use the same negatives $\mathcal{N}$ from Equation 4.2.2 to train the `NOT_CONNECTED` class.

Even if most negatives in $\mathcal{N}$ are true negatives (in the sense that the observations are not connectable with one action step), both h and $f^\dagger$ can be exposed to occasional false negatives during training (e.g., if the same location is visited twice, the induced observations won’t be temporally consecutive and can appear in $\mathcal{N}$). These false negatives in fact correspond to the loop closures that should be discovered by the trained system in the data. Perhaps surprisingly, it turns out that $f^\dagger$ makes “fortuitous mistakes” and does predict a number of loop closures properly, therefore overcoming to some degree the imperfections in its training objective. This property allows self-supervised methods to learn useful data representations in the absence of a clean supervisory signal (for navigation, the relative pose between samples).

4.2.2.4. Graph Construction. Equipped with h and $f^\dagger$, we can now build a **directed graph** $\mathcal{G}$ whose edges are weighted using d_h (Equation 4.2.1). We first treat the collected data as a chain graph with observed edges $E_o = \{(o_t, o_{t+1}) : o_t, o_{t+1} \in \tau_o\}$ and then run pairwise computations to obtain new loop closure edges $E_p = \{(o_t, o_s) : o_t, o_s \in \tau_o,\ f^\dagger(x_t, x_s) \in \mathcal{A}\}$. The final graph is denoted by $\mathcal{G} = (\tau_o, E_o \cup E_p)$. No additional post-processing of the graph is required, contrary to existing methods [**124, 151, 127, 44**] which can require tuning numerous hyper-parameters to curate nodes and edges.

4.2.2.5. Forward Dynamics Head. The forward dynamics head is denoted by $f : \mathcal{X} \times \mathcal{A} \to \mathcal{X}$ and trained using edges/transitions from $\mathcal{G}$. For any edge (o_t, o_s) in $\mathcal{G}$ during training, the module is trained with the mean squared error loss to approximate the function $(x_t, f^\dagger(x_t, x_s)) \mapsto x_s$, using the locomotion head $f^\dagger$ to provide an input action even if none was observed. f will therefore benefit from additional transitions in E_p that were not initially observed in E_o. The above is an instance of semi-supervised learning called co-training [**16**], in which the functions h and $f^\dagger$ are used to label unseen transitions in the training set, thus enhancing the supervisory signal that is employed to train f.

4.2.2.6. Geodesic Regressor. The final component and core navigation module $p^+ : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ learns to predict the shortest path lengths on $\mathcal{G}$. We denote these distances as $d_\mathcal{G}(o_t, o_g)$ and compute them with Dijkstra's algorithm. $d_\mathcal{G}$ is defined over observation pairs from the discrete vertex set of $\mathcal{G}$. We aim to extend it over the continuous latent space $\mathcal{X}$ to predict shortest path lengths for any pair of images at runtime. The training loss of the geodesic regressor is

$$\mathcal{L}_{p^+}(o_s, o_t) = (p^+(x_s, x_t) - d_\mathcal{G}(o_s, o_t))^2. \tag{4.2.3}$$

Interpreting observations as samples from a manifold embedded in the high-dimensional RGB space, the backbone h learns an embedding with locally Euclidean neighbourhoods (d_h), which are chained together by the graph search to compute the geodesic (intrinsic) distance over the entire manifold (Chapter 2.1.5). Equation 4.2.3 in fact corresponds to a manifold learning objective [**157, 137**], and we will show the O4A training results in interpretable visualizations of the environment in Figure 4.7.

Once all the components have been trained, $\mathcal{G}$ can be discarded and is not required for deploying the system. Indeed, both f and p^+ will provide all the required information for image-goal navigation, as we will detail in Chapter 4.2.3. In fact, the geodesic regressor p^+ can be interpreted as encoding the geometry of $\mathcal{G}$, thereby trading a potentially high number of nodes and edges for a fixed number of learnable parameters.

4.2.2.7. Multiple Environment Setting. When k environments are considered, we train both h and $f^\dagger$ on the entire data. To provide a more challenging task for the model, we sample negatives $\mathcal{N}$ from either the same environment or a different one. h and $f^\dagger$ can then be used to close loops and compute a set of graphs $\{\mathcal{G}_i\}_{i=1}^k$, one per environment. The forward dynamics f are then trained using transitions from all the graphs. Finally, each $\mathcal{G}_i$ is used to train a geodesic regressor p_i^+. In summary, h, $f^\dagger$ and f are shared across environments while p_i^+ is environment-specific.

4.2.3. Navigation

In this section, we discuss how to deploy O4A for navigation. This approach iterates over the potential function derived for DF in Chapter 3.2.2. Particularly, we plan motions over the agent configuration space $\mathcal{C}$ by defining A) an attractive potential around the goal, and B) repulsive potentials around obstacles, allowing the agent to minimize the total potential function via gradient descent to reach the goal while avoiding obstacles.

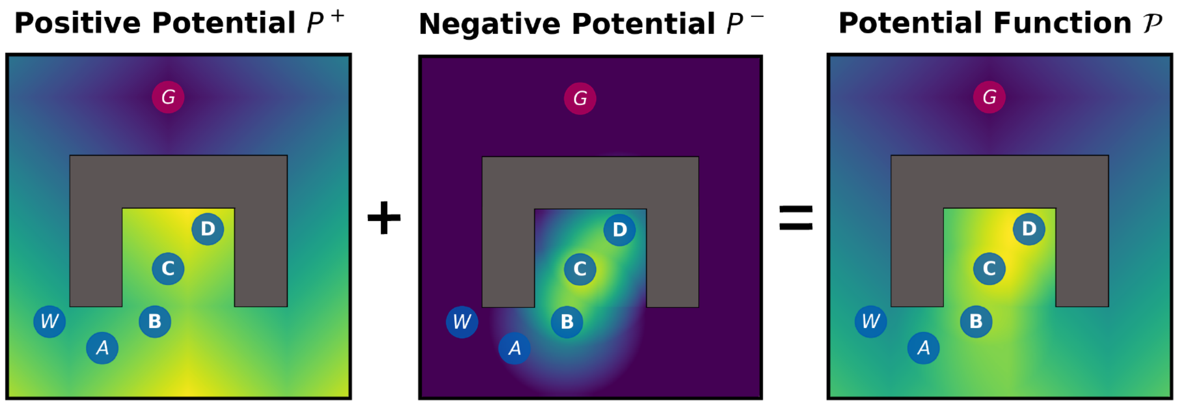


Fig. 4.2. An illustration of our potential function $\mathcal{P}$ (Equation 4.2.5). Darker colors indicate lower potential cost. (Left) Geodesic attractor, which reflects the geodesic distance to the goal G. (Center) Visited state repulsors (B, C, D). (Right) Total potential function $\mathcal{P}$, which can be minimized by the agent at A by picking an appropriate waypoint W.

As with APFs, O4A will navigate by minimizing an attractor located at at the goal. Since actual agent and goal states are unobserved, the potential computations occur over the latent space $\mathcal{X}$, i.e. the extracted latent features of the agent and goal RGB observations. As the attractor, we use the geodesic regressor p^+ which estimates the geodesic distance to the goal. Critically, this attractor factors in the environment geometry and can, for example, drive an agent out of a dead end to reach a goal that is close in terms of Euclidean distance, but far geodesically (c.f. Figure 4.2). This property is somewhat reminiscent of navigation functions in APFs literature as discussed in Chapter 2.1.1.

In practice, we found that minimizing p^+ alone did not suffice to successfully navigate. The agent would often end up thrashing between two poses due to a local minimum in the attractor landscape, which can occur due to learning errors and the discrete action space. We therefore define a latent repulsor function—similar to Duckie-Former and Bounini *et al.* [**18**]—which is only active in a certain radius $m_r \in \mathbb{R}^+$:

$$p^-(x_t, x_s) = \max(0, m_r - d_h(x_t, x_s)). \tag{4.2.4}$$

We use p^- to drive the agent away from previously visited states, the representations of which are saved in a buffer $\mathcal{B}$. By combining repulsors around observations in $\mathcal{B}$ and the geodesic attractor p^+, we obtain a total potential function of

$$\mathcal{P}(x_t, x_g, \mathcal{B}) = p^+(x_t, x_g) + \sum_{x \in \mathcal{B}} p^-(x_t, x) \tag{4.2.5}$$

where x_t and x_g represent the encodings of the current and goal RGB images, respectively.

The detailed navigation procedure is presented in Algorithm 1. During navigation, our agent greedily minimizes $\mathcal{P}$ by finding the best candidate action using forward dynamics

Algorithm 1 Navigation algorithm for One-4-All (O4A).

```
**Input**: env, goal image $o_g \in \mathcal{O}_{RGB}$, thresh $\in \mathbb{R}^+$, backbone $h$, forward dynamics $f$, collision detection function $\gamma$, potential function $\mathcal{P}$
$o$, scan $\leftarrow$ env.initialize()                              ▷ RGB and scan
$x \leftarrow h(o)$
$x_g \leftarrow h(o_g)$
$\mathcal{B} \leftarrow \{x\}$
while $d_h(x, x_g) >$ thresh do
    $\mathcal{A}_{free} \leftarrow \gamma($scan$)$
    $a^* \leftarrow \arg\min_{a \in \mathcal{A}_{free}} \mathcal{P}(f(x,a), x_g, \mathcal{B})$      ▷ Eq. 4.2.5
    $o$, scan $\leftarrow$ env.step(a$^*$)                            ▷ RGB and scan
    $x \leftarrow h(o)$
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{x\}$                              ▷ Update visited states
end while
env.step(STOP)
```

over the set $\mathcal{A}_{free}$ estimated by a collision detection function γ. This stands in contrast to APFs, since we blacklist collision-inducing actions instead of explicitly modeling repulsors around obstacles. In practice, since the agent rotates in place, we suppose that only the FORWARD action can cause collisions, which greatly simplifies the collision detection γ: we simply define a scan collision box in front of the robot based on its geometry.

4.3. Experiments

We assess our approach in both simulated and real-world environments. Our first simulated experiments are conducted within maze-like environments. The discrete action space of the agent consists of the movements $\mathcal{A} = \{\texttt{STOP}, \uparrow, \rightarrow, \downarrow, \leftarrow, \nearrow, \searrow, \swarrow, \nwarrow\}$ and its discrete configuration space is solely determined by a tuple of indices (i, j) on the maze's grid. The observation is an RGB top-down view of the environment with a resolution of 64×64 pixels. In the second block of experiments, our simulated and real-world experiments use a differential drive robot with two on-board RGB cameras, one facing forward and the other facing backward, each with a field of view of 90°. Each image has a resolution of 96×96 pixels. Consistent with [**61**], the robot moves FORWARD by 0.25m and ROTATE by 15°.

4.3.1. Simulation in Maze Environments

This section presents preliminary experiments of O4A over a simple maze-like environment. The goal is to demonstrate that a neural potential function can be indeed derived

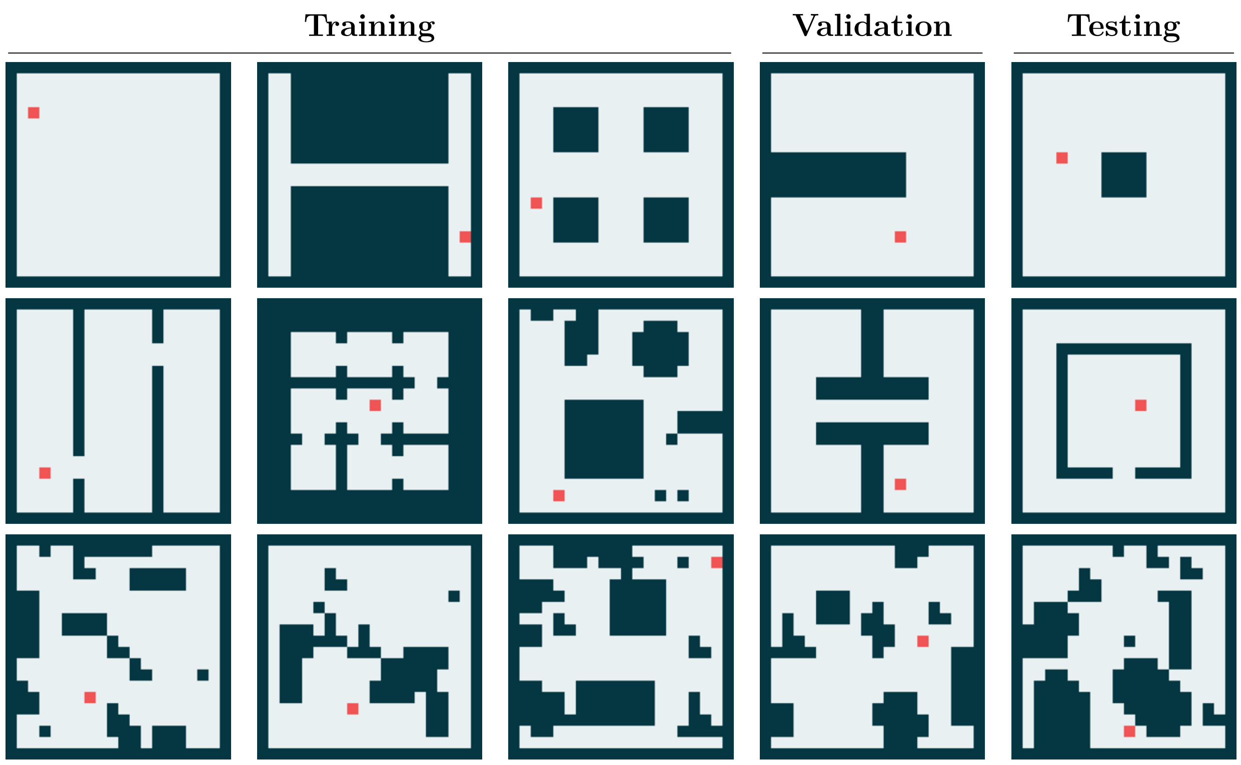


Fig. 4.3. Top-down view of maze environments used for training, validation and testing in O4A. The red square represents the agent at a random position within each environment. The first 3 columns show the training environments, fourth column validation and fifth column testing. All the environments share similar semantic features but with varying topologies with the purpose to assess the navigation capabilities of O4A over novel environments

with the methodology presented in Chapter 4.2. Furthermore, we show how the two principal components (obtained with PCA) of the last layer in the geodesic regressor p^+ exhibit consistency with the topology of the environments.

4.3.1.1. Data. These experiments are conducted using a maze simulator called Mazelab [**166**]. A total of 15 environments were used: 9 for training, 3 for validation, and 3 for testing. All environments have a 20×20 grid layout and we collect a total of 5,000 data points by navigating random waypoints (resulting in very suboptimal trajectories). Data points are split into a training set (50%) and a validation set (50%). Evaluation metrics are only reported over the three unseen test environments, and model checkpoints are obtained using the three validation environments. A layout of all the environments is presented in Figure 4.3.

4.3.1.2. Implementation details. All our maze models are trained using a batch size of 256. The local backbone uses 6 2D convolutions with batch norm and ReLU non-linearities, followed by 3 linear layers. The locomotion head is composed of 4 linear layers. Both the local backbone and locomotion heads are jointly trained for 100,000 gradient steps, using

Test Environment	**SR↑**	**SSR↑**	**SPL↑**
Omaze	0.96	0.96	0.77
Room	**0.99**	**0.99**	**0.84**
Obstacles	0.87	0.87	0.72
Average	0.94	0.94	0.78

Table 4.1. Navigation performance over 3 unseen test environments for O4A. We find that the neural potential function proposed in O4A is capable to navigate towards the goal and effectively call `STOP`. Indeed our method achieves a high Success Rate (SR), *Soft* Success Rate (SSR) and Success weighted by Path-Length (SPL) across the three unseen environments, highlighting the effectiveness of our SSL approach. As expected, the worst performance is achieved in the *obstacles* environments (row three, column five in Figure 4.3) as it is remarkably the one with the most complicated topology.

$m_+ = 1$ and $m_- = 2$. The forward dynamics model has 4 linear layers and is trained for 45,000 steps. The geodesic regressor has 6 linear layers, and the same architecture is used for all environments, with training also carried out for 10,000 steps for each regressor. All linear layers are followed by ReLU non-linearities. The models are trained using the Adam optimizer [**82**] with learning rate $5e^{-4}$. Navigation is performed with $|\mathcal{B}| = 20$, $m_r = 1.0$ and `thresh` $= 0.5$ throughout environments and experiments. At runtime, we simulate a range measurement to perform collision checking as detailed in Chapter 4.2.3[1].

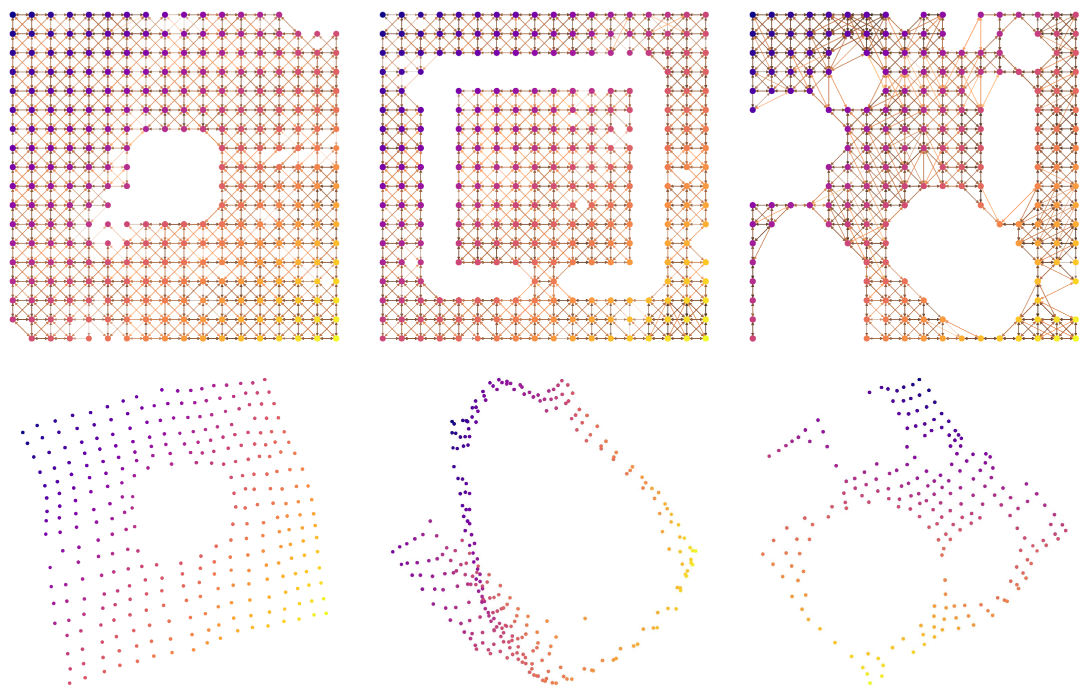
4.3.1.3. Evaluation. Navigation performance is assessed by sampling different starting and goal position over the three test environments (last column Figure 4.3). For each environment, 500 trajectories with a maximum episode length of 250 are sampled and the results are averaged to provide an overall measure of performance. We assess navigation performance by measuring the Success Rate (SR) and Success weighted by Path-Length (SPL) [**2**] defined as:

$$\frac{1}{N}\sum_{i=1}^{N} S_i \frac{l_i}{max(p_i, l_i)} \tag{4.3.1}$$

where l_i is the shortest path distance between the start and goal in trajectory i, p_i represents the length of the path actually taken for the agent and S_i is a binary indicator of success for current trajectory i. For maze experiments, a navigation trial is considered successful if the agent comes to a `STOP` within the exact goal position. We equally use a *Soft* Success Rate (SSR), where a trial is successful if the agent passes through the goal without calling `STOP` at any point during navigation.

4.3.1.4. Results. Table 4.1 presents quantitative results of our proposed method, O4A, for the maze environments. By training a geodesic regressor p^+ on top of a self-supervised

[1]All neural models in O4A were implemented with PyTorch (`https://pytorch.org/`)

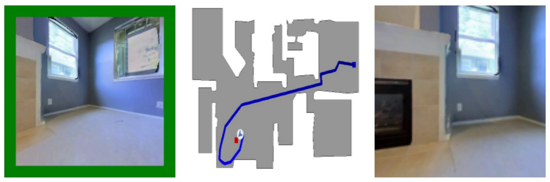


(a) Omaze. **(b)** Room. **(c)** Obstacles.

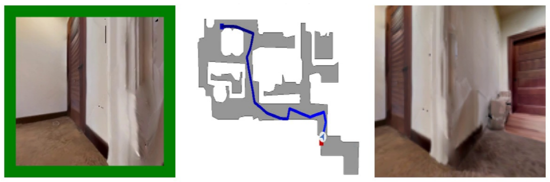
Fig. 4.4. (First row) O4A graph connectivity over test environments. Points correspond to the location of RGB observations and are colored by the sum of their x and y coordinates. The graphs are free of egregious spurious edges, which allows to train effective geodesic regressors before discarding them. (Second row) 2 principal components (PCA) of the last layer in the geodesic regressor p^+ with the same coloring scheme. The unsupervised latent geometry is consistent with the environment's geometry, and some topological features (e.g., the obstacle "holes") are evident in the latent space, even if the training of O4A never used pose information.

local backbone h, O4A demonstrates generalization capabilities over new environments and produce feasible navigation paths. This suggests that meaningful "local" representations can be obtained from SSL objectives and be further mapped to a more globally coherent representation using a nearest neighbour graph approach inspired from manifold learning (Figure 4.4). Our approach achieves an average Success Rate of 0.94 across the three test environments, demonstrating its navigation and generalization competences.

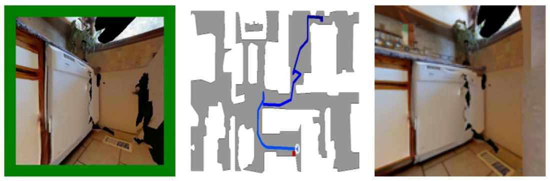
Furthermore, we emphasize the effectiveness of our `STOP` strategy discussed in Chapter 4.2.3, as both SR and SSR are identical across the three test environments. Therefore, indicating that the agent is capable of autonomously calling `STOP` upon reaching the goal. The average SPL of our approach was 0.78, which intuitively indicates that the trajectories produced by our method have a path length 22% longer than the optimal path. We note the forward dynamics of our method was equally capable of generalizing over these unseen environments without any fine-tuning. Ultimately, in Figure 4.4 we present the graph connectivity derived by O4A and the two principal components of the last layer in the geodesic regressor p^+. Interestingly, a linear projection of O4A latent codes results in a latent geometry that exhibits consistency with the ground-truth topology of the environments, demonstrating it is indeed possible to embed the approximate topology of the environment in a neural network via SSL and manifold learning.



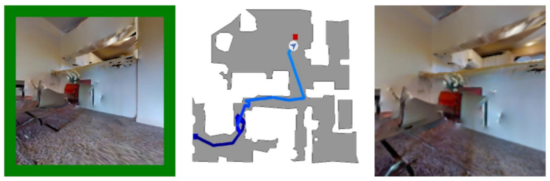
(a) Aloha.



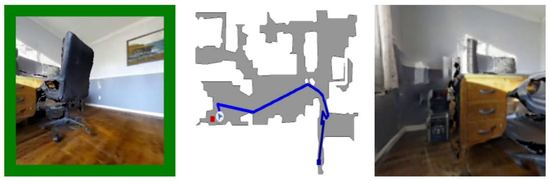
(b) Annawan.



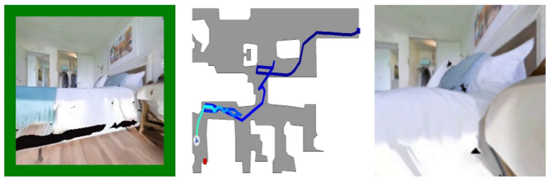
(c) Cantwell.



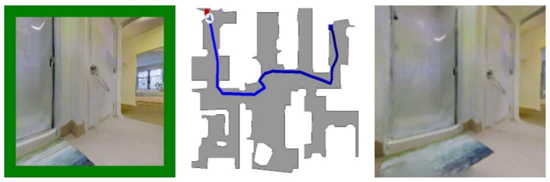
(d) Dunmor.



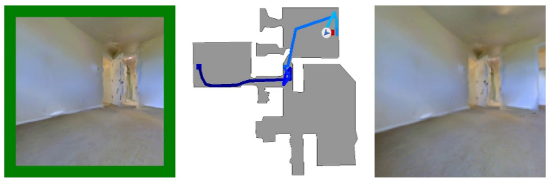
(e) Eastville.



(f) Hambleton.



(g) Nicut.



(h) Sodaville.

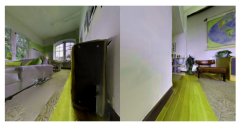
Fig. 4.5. Navigation trajectories obtained with O4A within the Habitat simulator over eight different environments. For each environment we present the following information: the last navigation frame of the front-facing camera (left), a top-down view of the environment (center) and the goal image (right). In the top-down view, the goal position is depicted by a red square, the start position with a blue square and trajectory taken by a blue line. It is noteworthy that the top-down map serves solely for visualization purposes and is not provided to the agent. O4A successfully navigates towards the goal across diverse environments featuring intricate topologies, resulting in navigation paths that are almost optimal (See Table 4.3).

(a) Original.

(b) Brightness and Contrast.

(c) Dropout.

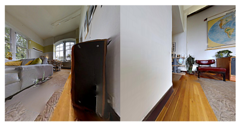
(d) Gaussian Noise.


(e) Hue Saturation.

(f) Color Jitter.

(g) Motion blur.

(h) Change of perspective.


(i) Sharpening.

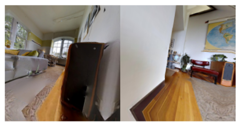

(j) Shift, Scale and Rotate

Fig. 4.6. Image augmentations used during training in O4A to improve the models' performance. These augmentations are used for both Habitat (Chapter 4.3.2) and real-world experiments (Chapter 4.3.3).

4.3.2. Simulation in Habitat Environments

In this section, we assess the performance of O4A in a harder navigation scenario using the 3D photorealistic Habitat simulator [**135**]. We provide an evaluation against competitive baselines on the literature as-well as ablation studies over the different pieces composing our neural potential function (Chapter 4.2.3)[2].

4.3.2.1. Data. We perform our experiments using the Habitat simulator [**125**] with the Gibson dataset [**154**]. We use the Gibson split defined in [**125**] and use eight environments from it: Hambleton ($67m^2$), Annawan ($75m^2$), Nicut ($90m^2$), Dunmor ($90m^2$), Cantwell ($107m^2$), Sodaville ($114m^2$), Aloha ($114m^2$) and Eastville ($121m^2$). A total of 240,000 data points are collected using the same procedure described in Chapter 4.3.1.1 and samples are split into a training set (70%) and a validation set (30%). A top-down view of the different environments is presented in Figure 4.5.

4.3.2.2. Implementation Details. Implementation of our models for Habitat is similar to those defined for maze experiments in Chapter 4.3.1.2. The primary differences are that all the Habitat models are trained with a batch size of 512 and the local backbone uses a ResNet 18 encoder [**67**], followed by 5 1D convolutions to fuse the latent codes of the front and rear facing cameras. Local backbone and locomotion heads are trained for 410,000 gradient steps, using $m_+ = 1$ and $m_- = 10$. Both forward dynamics and geodesic regressor have the same architecture as in maze but are trained for 330,000 steps. All models are trained using the image augmentations shown in Figure 4.6. We use the same optimizer and learning rate as in maze and navigation is performed with $|\mathcal{B}| = 500$, $m_r = 2.5$ and

[2]For further results and videos, see `https://montrealrobotics.ca/o4a/`

`thresh` = 3.5 throughout experiments. During navigation, we use a front-facing laser scan to perform collision checking.

Method	Component	Parameter	Value	Description
SPTM	Retrieval R	l	10	Maximum temporal distance (steps) between two positive samples.
		M	5	Temporal margin (l×M) between positive and negative samples.
	Graph $\mathcal{G}$	$s_{shortcut}$	0.7	Similarity threshold to create visual shortcut in the graph.
		ΔT_l	5	Minimum temporal distance between consecutive samples for visual shortcuts.
		ΔT_w	10	Number of samples in sequence to consider while performing visual shortcuts.
	Navigation	k	5	Number of nearest neighbors used to localize new sample in the graph.
		s_{local}	0.7	Threshold used to perform only local localization.
		s_{reach}	0.7	Reachability threshold to pick next navigation waypoint.
		H_{min}	1	Lower interval in the search window for next navigation waypoint.
		H_{max}	20	Upper interval in the search window for next navigation waypoint.
ViNG	Traversability $\mathcal{T}$	d_{max}	20	Maximum temporal distance (steps) between two samples the model is capable to predict.
	Graph $\mathcal{G}$	$\delta_{sparsify}$	5.0	Sparsification threshold for easily traversable vertices in the graph
O4A	Local Backbone h	m_+	1	Positive margin for siamese loss used to train local backbone.
		m_-	10	Negative margin for siamese loss used to train local backbone.
	Navigation	m_r	2.5	Radius at which the repulsors are activate with respect to current observation.
		`thresh`	3.5	Minimum local metric distance between current observation and goal to call `STOP`.
		$\lvert\mathcal{B}\rvert$	500	Maximum number of previously visited states stored and used for the repulsor function.

Table 4.2. Hyperparameters of baselines and O4A for Habitat experiments. Further detail of the models used and their architectures are given in Chapter A.1.

4.3.2.3. Baselines. To assess the navigation performance of our method, we compare against the following baselines:

- **Random Agent.** An agent sampling actions uniformly from $\mathcal{A} \setminus \{\texttt{STOP}\}$, subject to collision checking. The policy relies on ground-truth pose for oracle stopping.
- **Goal Conditioned Behavioral Cloning (GC-BC)**, adapted from [**33**]. We extended the method with Hindsight Experience Replay (HER) [**4**] to relabel new goals and enhance the training signal for the agent. This policy also relies on ground-truth pose to call `STOP`. We collected a distinct set of expert trajectories for this baseline.
- **SPTM** [**124**]. This method is extended with hard-negative mining as proposed by [**127**] to improve generalization performance. To call `STOP`, we tune a threshold on the reachability estimator between current and goal images.
- **ViNG** [**127**]. The PD controller of the original work is replaced by an oracle Habitat controller to handle discrete actions. We tune a threshold on the timestep predictor to call `STOP`.
- **O4A Without Latent Repulsors**. The repulsors p^- are not used during navigation and the agent is driven towards the goal by greedily minimizing p^+.
- **O4A Without Geodesic Regressor**. We discard the attractor p^+ and only use the latent repulsors p^-.

To ensure a fair comparison, all the baselines have the same capacity and collision checking strategy as O4A. We substitute the neural architectures used in the baselines with our local backbone h and locomotion head $f^\dagger$. Baselines are trained for 415,000 gradient steps and tested with various hyperparameters, the best of which were used for benchmarking. In Table 4.2 we provide a detailed summary of the hyperparameters used for the baselines and our model in these experiments.

4.3.2.4. Evaluation. To assess the navigation performance in the Habitat simulator, we rank the difficulty of each trajectory as proposed by [**26**]. Trajectories are categorized into easy ($1.5-3$m), medium ($3-5$m), and hard ($5-10$m) based on their geodesic distance to the goal. We add an extra category labeled as "very hard" (>10m) to evaluate the agent's ability to plan for long-horizon goals. Each difficulty level is assessed over 1,000 trajectories using a maximum episode length of 500. To ensure a comprehensive assessment, we sample distinct starting positions and goals for each environment and difficulty, resulting in a total of 4,000 trajectories per environment.

Evaluation metrics are the same as the ones discussed for maze experiments in Chapter 4.3.1.3 (i.e., SR, SSR, SPL). However, we add an additional metric to monitor the ratio of Collision-Free Trajectories (CFT) across all difficulties. CFT is calculated by

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{1}(i) \tag{4.3.2}$$

	$\mathcal{G}$	**Oracle Stop**	**Easy (1.5 - 3m)**			**Medium (3 - 5m)**			**Hard (5 - 10m)**			**Very-Hard (>10m)**			**CFT↑**
			SR↑	**SSR↑**	**SPL↑**	**SR↑**	**SSR↑**	**SPL↑**	**SR↑**	**SSR↑**	**SPL↑**	**SR↑**	**SSR↑**	**SPL↑**	
Random	–	✓	0.49	0.49	0.32	0.24	0.24	0.15	0.07	0.07	0.04	0.01	0.01	0.01	0.94
GC-BC [**33**]	–	✓	0.42	0.42	0.25	0.21	0.21	0.10	0.07	0.07	0.03	0.02	0.02	0.01	0.94
SPTM [**124**]	✓	–	0.32	0.51	0.14	0.17	0.28	0.06	0.07	0.12	0.03	0.02	0.03	0.01	0.96
ViNG [**127**]	✓	–	0.29	0.64	0.10	0.19	0.46	0.07	0.11	0.28	0.05	0.06	0.12	0.02	**0.99**
O4A (Ours)	–	–	**0.95**	**0.97**	**0.65**	**0.93**	**0.95**	**0.65**	**0.90**	**0.92**	**0.65**	**0.85**	**0.88**	**0.65**	0.96
O4A w/o p^-	–	–	0.11	0.36	0.11	0.07	0.22	0.07	0.03	0.09	0.03	0.01	0.02	0.01	**0.99**
O4A w/o p^+	–	–	0.45	0.70	0.13	0.27	0.47	0.09	0.13	0.23	0.04	0.04	0.07	0.02	0.90

Table 4.3. Average navigation performance over 8 simulated Gibson environments for O4A and relevant baselines. We further study two additional variants of O4A by ablating terms in the potential function (Equation 4.2.5). We also denote which methods rely on a graph ($\mathcal{G}$) for navigation and oracle stopping (other methods need to call `STOP` autonomously). We find that O4A substantially outperforms baselines, achieving a higher Success Rate (SR), *Soft* Success Rate (SSR), Success weighted by Path-Length (SPL), and a competitive ratio of Collision-Free Trajectories (CFT). A more detailed presentation of these results is provided in Chapter A.2

where the indicator function $\mathbb{1}(\cdot)$ is equal to one if trajectory i is collision-free. A trajectory is deemed collision-free if it does not collide during the experiment.

4.3.2.5. Results. We present quantitative navigation results in Table 4.3. O4A successfully navigates to goals of all difficulty levels and outperforms all considered baselines. The tight gap between SR and SSR showcases the reliability of the `STOP` mechanism based on the local metric d_h. Surprisingly, the O4A's SPL is stable across all difficulties, despite a decreasing success rate. This indicates that the quality of successful paths is in fact slightly better for distant goals. This observation, combined with visual evaluation of episodes, suggests that the O4A attractor provides a clearer signal for distant goals and is noisier when navigating nearby locations. The two O4A ablations confirm that all considered potentials in our potential function $\mathcal{P}$ are essential contributors to success.

While the SPTM and ViNG performance are below those reported in the original papers, they are in line with recent comparable benchmarks in the literature [**151**]. The gap can be partially explained by variations in the experimental design. Neither method considers collisions nor how to stop when the goal is reached. Having `STOP` in the action space increases the challenge, due to the agent's ability to prematurely terminate episodes because of a false positive. We also note that SPTM considered omnidirectional actions, and that the original ViNG results focused on larger-scale problems in relatively open outdoor settings. The GC-BC results demonstrate how learning long-range navigation remains challenging for end-to-end methods.

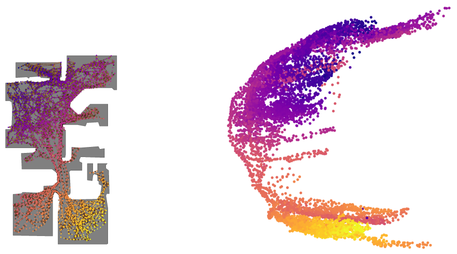
We further explore the O4A graph connectivity used for training and demonstrate how it learns an interpretable latent representation in Figure 4.7. While some latent representations display visual consistency with the topology of the environment, others like Cantwell (c) or Nicut (g) are harder to interpret. We argue this is an effect of the robot's heading not being considered on these plots. The degrees of freedom of a differential drive robot is 3 (x, y, θ), therefore, the configuration space of the robot is a three dimensional manifold which cannot be represented in a two dimensional space. On top of that, PCA is a linear operator that does not capture non-linear relations between the data during the projection

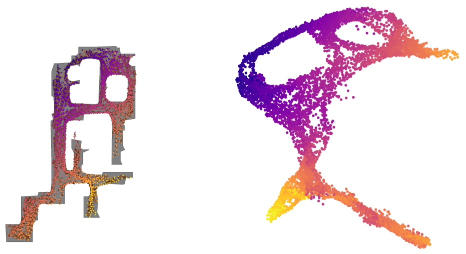
We decided to maintain PCA as our dimensionality reduction technique to highlight how our method is capable of recovering representations that are linearly separable. Furthermore, navigation trajectories over different environments are presented in Figure 4.5, where O4A is capable of effectively navigating a variety of environments with different topologies.

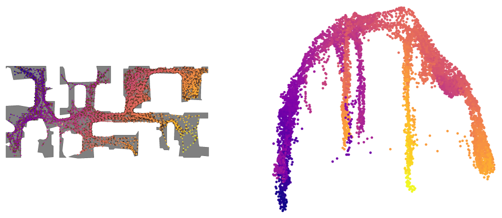
4.3.3. Real-World Environment

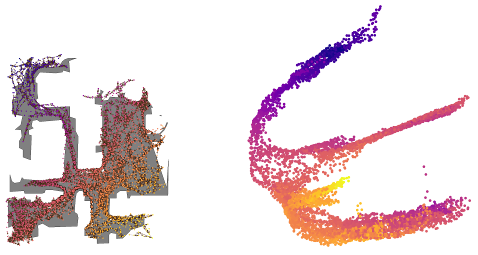
In this section, we assess the navigation capabilities of O4A in a real-world laboratory. Experiments demonstrate O4A can solve real-world indoor navigation tasks[3].

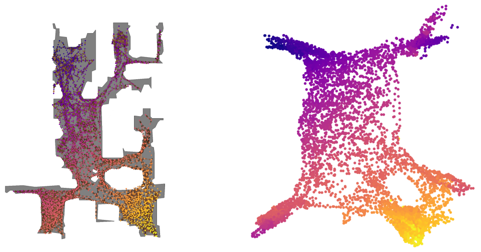
[3]For a demonstrative video, see `https://montrealrobotics.ca/o4a/`

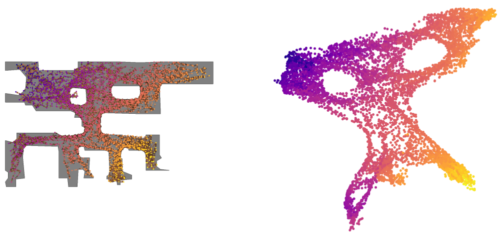

(a) Aloha.

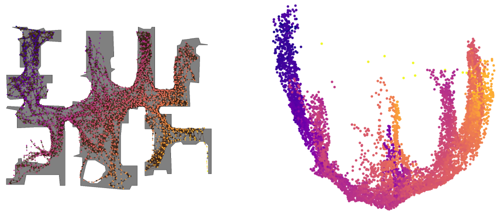

(b) Annawan.


(c) Cantwell.


(d) Dunmor.


(e) Eastville.


(f) Hambleton.


(g) Nicut.

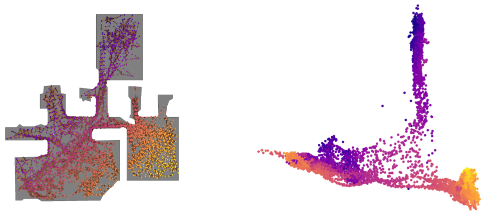

(h) Sodaville.

Fig. 4.7. Graph connectivity and unsupervised latent geometry of the eight Habitat environments. These plots follow the same plotting scheme as Figure 4.4. Points on the top-down view (gray map) correspond to the location of RGB observations and are colored by the sum of their x and y coordinates. The two principal components of the last layer in the geodesic regressor p^+ are colored with the same coloring scheme (located at right of top-down view). Remarkably, the unsupervised latent geometry derived by O4A in: a) Annawan , d) Dunmor , e) Eastville and f) Hambleton exhibit consistency with the topology of the environment even in partially observable settings and without any pose information. Top-down map of the environment and pose information are solely used for visualization purposes.

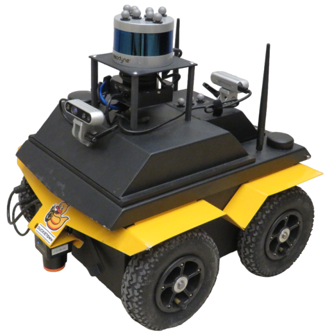


Fig. 4.8. We use the Jackal UGV mobile platform [**118**] using the RGB channels (no depth) of two Realsense D435i cameras with a 90° FOV. We detect forward collisions using a forward-facing Hokuyo laser scanner. We run O4A onboard using an Intel i7-8700 CPU with 32 GB of RAM. We did not require a GPU for navigation.

4.3.3.1. Data. For the real-world experiments, we collected data using the Jackal platform described in Figure 4.8. The experiments are conducted in a 4.65m x 9.1m laboratory. We collected a total of 21,000 RGB image samples via teleoperation and split them as in our Habitat simulation experiments (Chapter 4.3.2.1).

4.3.3.2. Implementation Details. We employ the exact same neural architectures used in the Habitat experiments (Chapter 4.3.2.2). However, all the models used for the real-world experiments are finetuned for 30,000 gradient steps starting from the best checkpoint obtained in the Habitat's experiments. All the image augmentations presented in Figure 4.6 are used during training. We run navigation with $|\mathcal{B}| = 300$, $m_r = 2.5$ and $\texttt{thresh} = 5$. During runtime we use a forward-facing Hokuyo laser scanner for collision-checking.

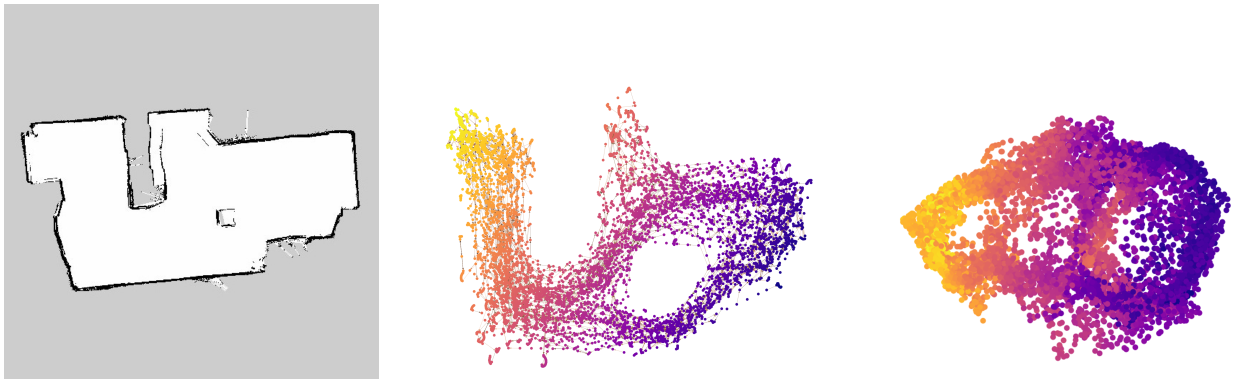
4.3.3.3. Evaluation. For evaluation, we chose 9 interesting episodes (3 starting positions with 3 goal images each) and repeated each 3 times, for a total of 27 runs. The maximum episode length was set to 300 steps, although the robot ended up exceeding 150 steps on only 2 occasions. In addition to SR and SSR (Chapter 4.3.1.3), we evaluate the final Distance to Goal (DTG), the number of `FORWARD` steps, and the number of `ROTATION` steps. For context, we also teleoperated the robot over the same episodes to provide an estimate of human performance.

4.3.3.4. Results. The real-world navigation results are presented in Table 4.4. O4A solves most episodes and achieves an average DTG of under 1m, even if most goals were not visible from the starting location and located up to 9 meters away. The maximal measured DTG was 1.74m. Interestingly, the number of O4A calls to the `FORWARD` action is comparable to human performance, meaning the O4A paths over the plane are competitive. Moreover, the robot collided only one time over all episodes using our collision checking strategy.

	SR ↑	**SSR** ↑	**DTG** ↓	FORWARD ↓	ROT. ↓
O4A	0.74	0.78	0.83	23	53
Human	**0.96**	**1.00**	**0.46**	**20**	**18**

Table 4.4. Average real-world navigation performance for O4A and human teleoperation over 27 episodes. We report the Success Rate (SR), *Soft* Success Rate (SSR), final Distance to Goal (DTG), number of `FORWARD` steps and number of rotation steps (`ROT.`).

Two navigation episodes are shown in Figure 1.2 where navigation paths are gathered using a motion capture ViCON system. In Figure 4.10, we provide snapshots from the front-facing camera during the aforesaid navigation experiments. Remarkably, the agent is capable to autonomously call `STOP` right in front of the goal image for both navigation trials.



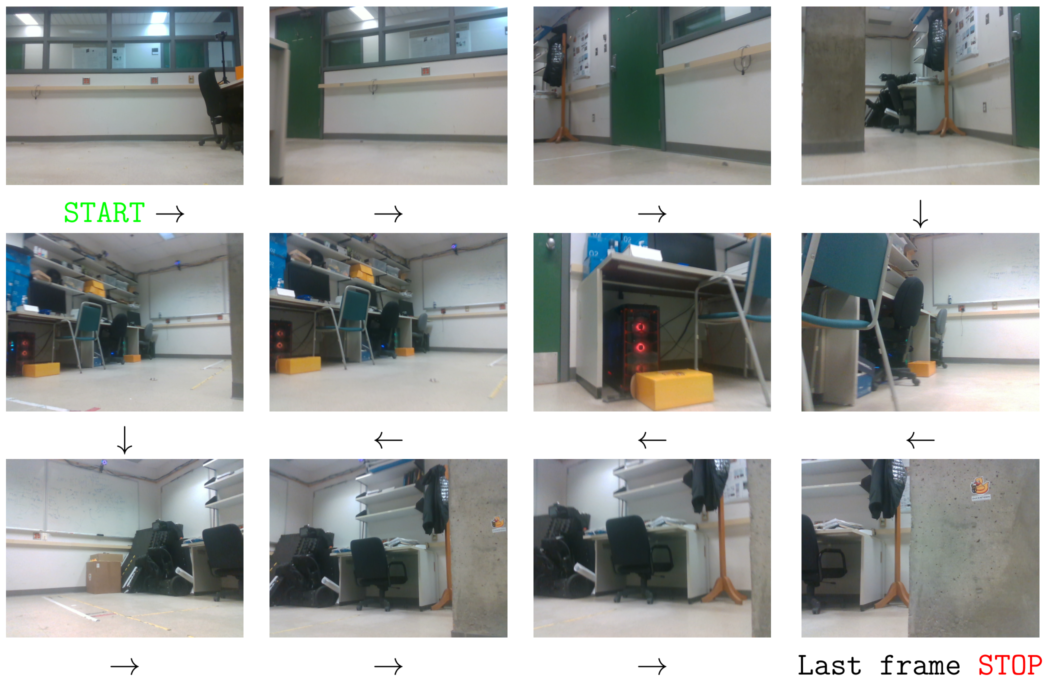
(a) Layout of laboratory. **(b)** Derived graph connectivity. **(c)** Unsupervised latent geometry.

Fig. 4.9. a) Top-down view of the real-world laboratory used in our experiments (captured with ROS GMapping [**150**]). b) Training graph derived with our local backbone, where each point corresponds to an image sample and are colored based on the sum of their x and y coordinates (as in Figure 4.4). c) unsupervised latent geometry obtained by the 2 first principal components of the last layer in the geodesic regressor.

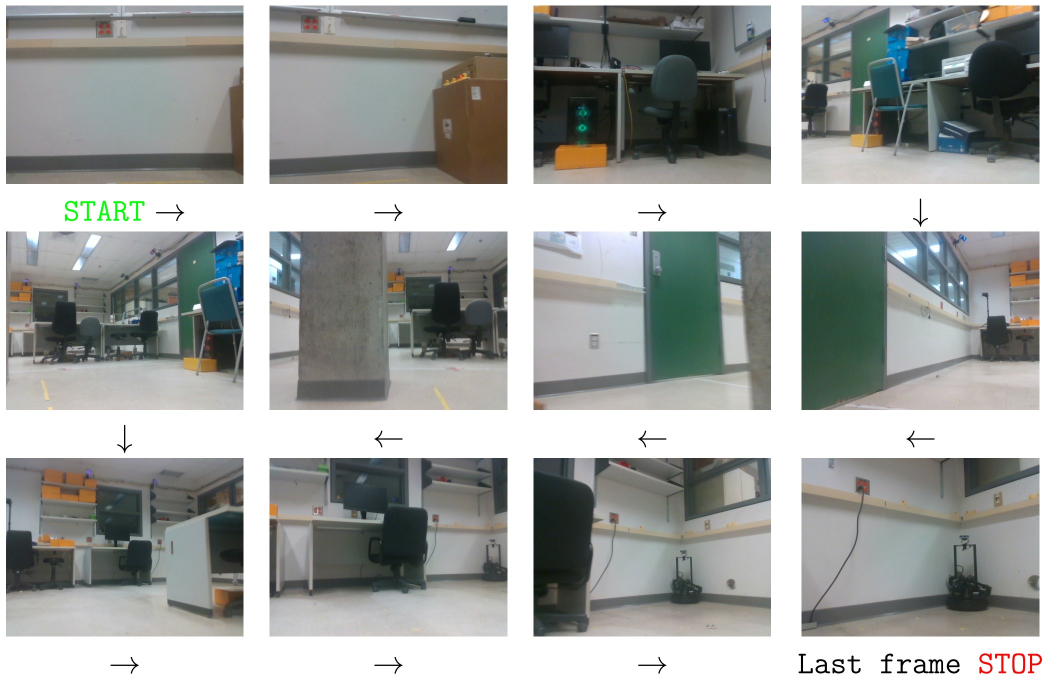
The unsupervised latent geometry derived by O4A and the environment itself are shown in Figure 4.9. Overall, O4A is capable to discover correlations between the data, as similar colors are clustered together. However, the shape of the plot is not as interpretable as those presented in simulation experiments. The visual complexity of the environment, dimensionality of the configuration space's manifold, and the dimensionality reduction technique may be contributing to this issue, as discussed in Chapter 4.3.2.4.

4.4. Discussion

In this chapter, we presented O4A, an end-to-end fully-parametric approach for image-goal navigation tasks. We demonstrated how neural networks can be trained through SSL



(a) First robot trajectory (path in Figure 1.2a).



(b) Second robot trajectory (path in Figure 1.2b).

Fig. 4.10. Images from the front-view camera during two real-world successful navigation trials. a) The goal image corresponds to a pillar with a sticker of a duck as shown in the last navigation frame. b) The goal image corresponds to the turtlebot depicted in the last navigation frame. Navigation sequences start on the top-left (START) and finish at the bottom-right (STOP) by following the arrows.

and manifold learning to encode the approximate topology of the environment. Additionally, we showed how a geodesic regressor can be used in a neural potential function to guide the agent towards the goal and avoid local minima in the optimization landscape. Importantly, our method does not require any annotated images or pose information, making it potentially scalable to large-scale datasets [**130**].

While our final trained system is graph-free, we still require learning a geodesic regressor for each environment to encode the geometry (in the same way current approaches need to build an environment-specific graph). Generalizing geodesic regressors across environments is a promising area of research, since it could allow to completely skip the graph building stage in new settings. Moreover, another limitation is that our method needs substantial coverage over the free configuration space of the environment to learn an informative geodesic regressor. Overall, O4A demonstrated navigation paths that were close to optimal in simulated and real-world environments and surpassed the performance of competitive baselines in the literature.

Chapter 5

Conclusion

We proposed two end-to-end fully-parametric approaches for robot navigation using only RGB images and a navigation strategy inspired by APFs. The aim was to demonstrate the usefulness of neural architectures as memory representations for robot navigation.

Our first approach (Duckie-Former), demonstrated how a ViT pre-trained with SSL can be used to derive a reactive potential for navigation in road-like environments. The second approach, named O4A, extended DF for image-goal navigation by leveraging SSL and manifold learning. We derived a neural potential function capable to drive the agent towards the goal via gradient descent, while escaping local minima in the optimization landscape. Both methods succeeded in real-world navigation tasks, demonstrating their utility beyond simulation environments and did not require any pose information.

In the primary contribution of this work, One-4-All, we addressed the data labeling limitations of Duckie-Former and trained all models through SSL. Additionally, we tackled limitations of graph-based navigation strategies and replaced graph search queries with an inexpensive neural network forward pass. We also highlight how O4A managed to recover a latent representation that is coherent with the intrinsic topology of the environments. Using a neural regressor as a memory representation allows for long-horizon navigation in a fully-parametric manner, which could potentially be scaled to generalize across new environments.

While O4A can navigate indoor environments from only RGB images, a new geodesic regressor has to be learned for each new environment. Consequently, research on a conditioned geodesic regressor that generalizes across environments represents a promising area of research. Furthermore, a limitation of our approach is its discrete action space, which affects the total number of configurations the robot can reach within an environment. An effective way to implement continuous actions while maintaining a tractable way to minimize the neural potential function is an interesting research idea. Finally, a promising research question is how to ground foundation models [**136, 109, 105, 130, 73**] with our unsupervised latent geometry to enhance the navigation abilities of embodied agents.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog *et al.* : Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[2] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva *et al.* : On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018.

[3] Sterling J Anderson, Sisir B. Karumanchi et Karl Iagnemma : Constraint-based planning and control for safe, semi-autonomous operation of vehicles. *In 2012 IEEE Intelligent Vehicles Symposium*, pages 383–388, 2012.

[4] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel et Wojciech Zaremba : Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[5] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray *et al.* : Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[6] Vijay Badrinarayanan, Alex Kendall et Roberto Cipolla : Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017.

[7] Vijay Badrinarayanan, Alex Kendall et Roberto Cipolla : Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[8] Somil Bansal, Varun Tolani, Saurabh Gupta, Jitendra Malik et Claire Tomlin : Combining optimal control and learning for visual navigation in novel environments. *In Conference on Robot Learning*, pages 420–429. PMLR, 2020.

[9] Adrien Bardes, Jean Ponce et Yann LeCun : VICReg: Variance-invariance-covariance regularization for self-supervised learning. *In International Conference on Learning Representations*, 2022.

[10] Jerome Barraquand et Jean-Claude Latombe : Robot motion planning: A distributed representation approach. *International Journal of Robotic Research - IJRR*, 10:628–649, 12 1991.

[11] Jerome Barraquand et Jean-Claude Latombe : Robot motion planning: A distributed representation approach. *International Journal of Robotic Research - IJRR*, 10:628–649, 12 1991.

[12] Herbert Bay, Tinne Tuytelaars et Luc Van Gool : Surf: Speeded up robust features. *Lecture notes in computer science*, 3951:404–417, 2006.

[13] Maximilian Beinhofer, Jörg Müller et Wolfram Burgard : Effective landmark placement for accurate and reliable mobile robot navigation. *Robotics and Autonomous Systems*, 61(10):1060–1069, 2013. Selected Papers from the 5th European Conference on Mobile Robots (ECMR 2011).

[14] Mikhail Belkin et Partha Niyogi : Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[15] Yoshua Bengio, Aaron Courville et Pascal Vincent : Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[16] Avrim Blum et Tom Mitchell : Combining labeled and unlabeled data with co-training. *In Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, page 92–100, New York, NY, USA, 1998. Association for Computing Machinery.

[17] Zahra Boroujeni, Daniel Goehring, Fritz Ulbrich, Daniel Neumann et Raul Rojas : Flexible unit a-star trajectory planning for autonomous vehicles on structured road maps. *In 2017 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pages 7–12, 2017.

[18] Farid Bounini, Denis Gingras, Herve Pollart et Dominique Gruyer : Modified artificial potential field method for online path planning applications. *In 2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 180–185, 2017.

[19] John F. Canny : *The Complexity of Robot Motion Planning.* MIT Press, Cambridge, MA, USA, 1988.

[20] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski et Armand Joulin : Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924, 2020.

[21] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski et Armand Joulin : Emerging properties in self-supervised vision transformers, 2021.

[22] Yuning Chai, Benjamin Sapp, Mayank Bansal et Dragomir Anguelov : Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *In* Leslie Pack Kaelbling, Danica Kragic et Komei Sugiura, éditeurs : *Proceedings of the Conference on Robot Learning*, volume 100 de *Proceedings of Machine Learning Research*, pages 86–99. PMLR, 30 Oct–01 Nov 2020.

[23] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta et Ruslan Salakhutdinov : Learning to explore using active neural slam. *In International Conference on Learning Representations (ICLR)*, 2020.

[24] Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta et Russ R Salakhutdinov : Object goal navigation using goal-oriented semantic exploration. *Advances in Neural Information Processing Systems*, 33:4247–4258, 2020.

[25] Devendra Singh Chaplot, Helen Jiang, Saurabh Gupta et Abhinav Gupta : Semantic curiosity for active visual learning. *In Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pages 309–326. Springer, 2020.

[26] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta et Saurabh Gupta : Neural topological slam for visual navigation. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12875–12884, 2020.

[27] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy et Alan L Yuille : Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834—848, April 2018.

[28] Ting CHEN, Simon KORNBLITH, Mohammad NOROUZI et Geoffrey HINTON : A simple framework for contrastive learning of visual representations. *In International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[29] Wuyang CHEN, Xianzhi DU, Fan YANG, Lucas BEYER, Xiaohua ZHAI, Tsung-Yi LIN, Huizhong CHEN, Jing LI, Xiaodan SONG, Zhangyang WANG *et al.* : A simple single-scale vision transformer for object localization and instance segmentation. *arXiv preprint arXiv:2112.09747*, 2021.

[30] Xinlei CHEN et Kaiming HE : Exploring simple siamese representation learning. *In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15750–15758, 2021.

[31] Hao-Tien Lewis CHIANG, Aleksandra FAUST, Marek FISER et Anthony FRANCIS : Learning navigation behaviors end-to-end with autorl. *IEEE Robotics and Automation Letters*, 4(2):2007–2014, 2019.

[32] Howie CHOSET, Kevin M LYNCH, Seth HUTCHINSON, George A KANTOR et Wolfram BURGARD : *Principles of robot motion: theory, algorithms, and implementations.* MIT press, 2005.

[33] Felipe CODEVILLA, Matthias MIILLER, Antonio LÓPEZ, Vladlen KOLTUN et Alexey DOSOVITSKIY : End-to-end driving via conditional imitation learning. *In 2018 IEEE International Conference on Robotics and Automation (ICRA)*, page 1–9. IEEE Press, 2018.

[34] R. R. COIFMAN, S. LAFON, A. B. LEE, M. MAGGIONI, B. NADLER, F. WARNER et S. W. ZUCKER : Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the National Academy of Sciences*, 102(21):7426–7431, 2005.

[35] Zihang DAI, Zhilin YANG, Yiming YANG, Jaime CARBONELL, Quoc LE et Ruslan SALAKHUTDINOV : Transformer-XL: Attentive language models beyond a fixed-length context. *In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy, juillet 2019. Association for Computational Linguistics.

[36] Feras DAYOUB, Timothy MORRIS, Ben UPCROFT et Peter CORKE : Vision-only autonomous navigation using topometric maps. *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1923–1929, 2013.

[37] Edsger W DIJKSTRA : A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[38] Carl DOERSCH, Abhinav GUPTA et Alexei A EFROS : Unsupervised visual representation learning by context prediction. *In Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015.

[39] Dmitri DOLGOV, Sebastian THRUN, Michael MONTEMERLO et James DIEBEL : Practical search techniques in path planning for autonomous driving. *AAAI Workshop - Technical Report*, 01 2008.

[40] Alexey DOSOVITSKIY, Lucas BEYER, Alexander KOLESNIKOV, Dirk WEISSENBORN, Xiaohua ZHAI, Thomas UNTERTHINER, Mostafa DEHGHANI, Matthias MINDERER, Georg HEIGOLD, Sylvain GELLY, Jakob USZKOREIT et Neil HOULSBY : An image is worth 16x16 words: Transformers for image recognition at scale. *In International Conference on Learning Representations*, 2021.

[41] Gabriel DULAC-ARNOLD, Nir LEVINE, Daniel J. MANKOWITZ, Jerry LI, Cosmin PADURARU, Sven GOWAL et Todd HESTER : Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis. *Mach. Learn.*, 110(9):2419–2468, sep 2021.

[42] Staffan EKVALL, Danica KRAGIC et Patric JENSFELT : Object detection and mapping for service robot tasks. *Robotica*, 25:175–187, 2007.

[43] Mohamed ELBANHAWI et Milan SIMIC : Sampling-based robot motion planning: A review. *IEEE Access*, 2:56–77, 2014.

[44] Scott EMMONS, Ajay JAIN, Misha LASKIN, Thanard KURUTACH, Pieter ABBEEL et Deepak PATHAK : Sparse graphical memory for robust planning. *Advances in Neural Information Processing Systems*, 33:5251–5262, 2020.

[45] Ben EYSENBACH, Russ R SALAKHUTDINOV et Sergey LEVINE : Search on the replay buffer: Bridging planning and reinforcement learning. *In Advances in Neural Information Processing Systems*, volume 32, 2019.

[46] Kuan FANG, Fei-Fei LI, Silvio SAVARESE et Alexander TOSHEV : Scene memory transformer for embodied agents in long time horizon tasks. *In CVPR*, 2019.

[47] Patrick FOO, William WARREN, Andrew DUCHON et Michael TARR : Do humans integrate routes into a cognitive map? map- versus landmark-based navigation of novel shortcuts. *Journal of experimental psychology. Learning, memory, and cognition*, 31:195–215, 04 2005.

[48] Andrea GALASSI, Marco LIPPI et Paolo TORRONI : Attention in natural language processing. *IEEE transactions on neural networks and learning systems*, 32(10):4291–4308, 2020.

[49] Jonathan D GAMMELL, Siddhartha S SRINIVASA et Timothy D BARFOOT : Informed rrt: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. *In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2997–3004. IEEE, 2014.

[50] Andreas GEIGER, Philip LENZ, Christoph STILLER et Raquel URTASUN : Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32:1231–1237, 2013.

[51] Georgios GEORGAKIS, Bernadette BUCHER, Karl SCHMECKPEPER, Siddharth SINGH et Kostas DANIILIDIS : Learning to map for active semantic goal navigation. *In International Conference on Learning Representations*, 2022.

[52] Spyros GIDARIS, Praveer SINGH et Nikos KOMODAKIS : Unsupervised representation learning by predicting image rotations. *In International Conference on Learning Representations*, 2018.

[53] Sabine GILLNER et Hanspeter A. MALLOT : Navigation and acquisition of spatial knowledge in a virtual maze. *Journal of Cognitive Neuroscience*, 10(4):445–463, 1998.

[54] D. GIRARDEAU : *Fonction décision-commande d'un robot manipulateur destiné à tailler automatiquement la vigne*. Université de Bordeaux I, 1983.

[55] Chad L. GOERZEN, Zhaodan KONG et Bernard METTLER : A survey of motion planning algorithms from the perspective of autonomous uav guidance. *Journal of Intelligent and Robotic Systems*, 57:65–100, 2010.

[56] David GONZÁLEZ, Joshué PÉREZ, Vicente MILANÉS et Fawzi NASHASHIBI : A review of motion planning techniques for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 17(4):1135–1145, 2016.

[57] Ian J. GOODFELLOW, Yoshua BENGIO et Aaron COURVILLE : *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. `http://www.deeplearningbook.org`.

[58] Jean-Bastien GRILL, Florian STRUB, Florent ALTCHÉ, Corentin TALLEC, Pierre RICHEMOND, Elena BUCHATSKAYA, Carl DOERSCH, Bernardo AVILA PIRES, Zhaohan GUO, Mohammad GHESHLAGHI AZAR, Bilal PIOT, koray KAVUKCUOGLU, Remi MUNOS et Michal VALKO : Bootstrap your own latent - a new approach to self-supervised learning. *In* H. LAROCHELLE, M. RANZATO, R. HADSELL, M. F. BALCAN et H. LIN, éditeurs : *Advances in Neural Information Processing Systems*, volume 33, pages 21271–21284. Curran Associates, Inc., 2020.

[59] Jose GUIVANT, Eduardo NEBOT, Juan NIETO et Favio MASSON : Navigation and mapping in large unstructured environments. *I. J. Robotic Res.*, 23:449–472, 04 2004.

[60] Saurabh GUPTA, James DAVIDSON, Sergey LEVINE, Rahul SUKTHANKAR et Jitendra MALIK : Cognitive mapping and planning for visual navigation. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2616–2625, 2017.

[61] Meera HAHN, Devendra Singh CHAPLOT, Shubham TULSIANI, Mustafa MUKADAM, James M REHG et Abhinav GUPTA : No rl, no simulation: Learning to navigate without navigating. *Advances in Neural Information Processing Systems*, 34:26661–26673, 2021.

[62] Peter E. HART, Nils J. NILSSON et Bertram RAPHAEL : A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[63] J.B. HAYET, F. LERASLE et M. DEVY : A visual landmark framework for indoor mobile robot navigation. *In Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 4, pages 3942–3947 vol.4, 2002.

[64] Kaiming HE, Xinlei CHEN, Saining XIE, Yanghao LI, Piotr DOLLÁR et Ross GIRSHICK : Masked autoencoders are scalable vision learners. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022.

[65] Kaiming HE, Haoqi FAN, Yuxin WU, Saining XIE et Ross GIRSHICK : Momentum contrast for unsupervised visual representation learning. *In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[66] Kaiming HE, Georgia GKIOXARI, Piotr DOLLÁR et Ross GIRSHICK : Mask r-cnn. *In Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[67] Kaiming HE, Xiangyu ZHANG, Shaoqing REN et Jian SUN : Deep residual learning for image recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[68] Joao F. HENRIQUES et Andrea VEDALDI : Mapnet: An allocentric spatial memory for mapping environments. *In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8476–8484, 2018.

[69] Tobias HESSE, Daniel HESS et Thomas SATTEL : Motion planning for passenger vehicles - force field trajectory optimization for automated driving. *In* M. A. KARIM, K. LEE, H. LING, D. MAROUDAS et T. M. SOBH, éditeurs : *15th IASTED International COnference on Robotics and Applications*, IASTED Technology Conferences, November 2010.

[70] Sepp HOCHREITER et Jürgen SCHMIDHUBER : Long Short-Term Memory. *Neural Computation*, 9(8): 1735–1780, 11 1997.

[71] Taylor HOWELL, Simon CLEAC'H, Sumeet SINGH, Pete FLORENCE, Zac MANCHESTER et Vikas SINDHWANI : Trajectory optimization with optimization-based dynamics. *IEEE Robotics and Automation Letters*, 7:6750–6757, 07 2022.

[72] D. HSU et Zheng SUN : Adaptively combining multiple sampling strategies for probabilistic roadmap planning. *In IEEE Conference on Robotics, Automation and Mechatronics, 2004.*, volume 2, pages 774–779 vol.2, 2004.

[73] Chenguang HUANG, Oier MEES, Andy ZENG et Wolfram BURGARD : Visual language maps for robot navigation. *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, London, UK, 2023.

[74] Xiaoming HUO, Xuelei Sherry NI et Andrew K SMITH : A survey of manifold-based learning methods. *Recent advances in data mining of enterprise data*, pages 691–745, 2007.

[75] Ian T JOLLIFFE : *Principal Component Analysis*. Springer-Verlag New York, Inc, New York, 2nd édition, 2002. [Cited by 2790] (647.34/year).

[76] Rahul KALA et Kevin WARWICK : Planning of multiple autonomous vehicles using rrt. *In 2011 IEEE 10th International Conference on Cybernetic Intelligent Systems (CIS)*, pages 20–25, 2011.

[77] Sertac KARAMAN et Emilio FRAZZOLI : Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[78] L.E. KAVRAKI, P. SVESTKA, J.-C. LATOMBE et M.H. OVERMARS : Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[79] O. KHATIB : Real-time obstacle avoidance for manipulators and mobile robots. *In Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 500–505, 1985.

[80] Jin-Oh KIM et Pradeep K. KHOSLA : Real-time obstacle avoidance using harmonic potential functions. *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 790–796 vol.1, 1991.

[81] Nuri KIM, Obin KWON, Hwiyeon YOO, Yunho CHOI, Jeongho PARK et Songhwai OH : Topological semantic graph memory for image-goal navigation. *In 6th Annual Conference on Robot Learning*, 2022.

[82] Diederik KINGMA et Jimmy BA : Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[83] Diederik P KINGMA et Max WELLING : Auto-encoding variational bayes. *In 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[84] Alex KRIZHEVSKY, Ilya SUTSKEVER et Geoffrey E HINTON : Imagenet classification with deep convolutional neural networks. *In* F. PEREIRA, C.J. BURGES, L. BOTTOU et K.Q. WEINBERGER, éditeurs : *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[85] Benjamin J. KUIPERS et Todd S. LEVITT : Navigation and mapping in large scale space. *AI Magazine*, 9(2):25, Jun. 1988.

[86] Yoshiaki KUWATA, Justin TEO, Gaston FIORE, Sertac KARAMAN, Emilio FRAZZOLI et Jonathan P. HOW : Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems Technology*, 17(5):1105–1118, 2009.

[87] Jean-Claude LATOMBE : *Robot Motion Planning*. Kluwer Academic Publishers, USA, 1991.

[88] S. M. LAVALLE : *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[89] Steven M. LAVALLE : Rapidly-exploring random trees : a new tool for path planning. *The annual research report*, 1998.

[90] Yann LECUN, Bernhard BOSER, John S DENKER, Donnie HENDERSON, Richard E HOWARD, Wayne HUBBARD et Lawrence D JACKEL : Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[91] John LEONARD, Jonathan HOW, Seth TELLER, Mitch BERGER, Stefan CAMPBELL, Gaston FIORE, Luke FLETCHER, Emilio FRAZZOLI, Albert HUANG, Sertac KARAMAN, Olivier KOCH, Yoshiaki KUWATA, David MOORE, Edwin OLSON, Steve PETERS, Justin TEO, Robert TRUAX, Matthew WALTER, David BARRETT, Alexander EPSTEIN, Keoni MAHELONI, Katy MOYER, Troy JONES, Ryan BUCKLEY, Matthew ANTONE, Robert GALEJS, Siddhartha KRISHNAMURTHY et Jonathan WILLIAMS : *A Perception-Driven Autonomous Urban Vehicle*, pages 163–230. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[92] Sergey LEVINE et Dhruv SHAH : Learning robotic navigation from experience: principles, methods and recent results. *Philos. Trans. R. Soc. Lond. B Biol. Sci.*, 378(1869):20210447, janvier 2023.

[93] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin et Baining Guo : Swin transformer: Hierarchical vision transformer using shifted windows. *In Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–1022, 2021.

[94] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell et Saining Xie : A convnet for the 2020s. *arXiv preprint arXiv:2201.03545*, 2022.

[95] David G Lowe : Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004.

[96] David Madås, Mohsen Nosratinia, Mansour Keshavarz, Peter Sundström, Rolland Philippsen, Andreas Eidehall et Karl-Magnus Dahlén : On path planning methods for automotive collision avoidance. *In 2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 931–937, 2013.

[97] Xiangyun Meng, Nathan Ratliff, Yu Xiang et Dieter Fox : Scaling local control to large-scale topological navigation. *In International Conference on Robotics and Automation (ICRA)*, pages 672–678. IEEE, 2020.

[98] Lina Mezghan, Sainbayar Sukhbaatar, Thibaut Lavril, Oleksandr Maksymets, Dhruv Batra, Piotr Bojanowski et Karteek Alahari : Memory-augmented reinforcement learning for image-goal navigation. *In 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3316–3323. IEEE, 2022.

[99] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andy Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, Dharshan Kumaran et Raia Hadsell : Learning to navigate in complex environments. *In International Conference on Learning Representations*, 2017.

[100] Dmytro Mishkin, Alexey Dosovitskiy et Vladlen Koltun : Benchmarking Classic and Learned Navigation in Complex 3D Environments. *arXiv*, janvier 2019.

[101] Sacha Morin, Miguel Saavedra-Ruiz et Liam Paull : One-4-all: Neural potential fields for embodied navigation. *arXiv preprint arXiv:2303.04011*, 2023.

[102] Arsalan Mousavian, Alexander Toshev, Marek Fišer, Jana Košecká, Ayzaan Wahid et James Davidson : Visual representations for semantic target driven navigation. *In 2019 International Conference on Robotics and Automation (ICRA)*, pages 8846–8852, 2019.

[103] Mehdi Noroozi et Paolo Favaro : Unsupervised learning of visual representations by solving jigsaw puzzles. *In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI*, pages 69–84. Springer, 2016.

[104] John O'Keefe et Lynn Nadel : *The Hippocampus as a Cognitive Map.* Oxford University Press, 1978.

[105] OpenAI : Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.

[106] Santosh Panda et Chandra Panda : A review on image classification using bag of features approach. *International Journal of Computer Sciences and Engineering*, 7:538–542, 06 2019.

[107] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell et Alexei A Efros : Context encoders: Feature learning by inpainting. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[108] Liam Paull, Jacopo Tani, Heejin Ahn, Javier Alonso-Mora, Luca Carlone, Michal Cap, Yu Fan Chen, Changhyun Choi, Jeff Dusek, Yajun Fang *et al.* : Duckietown: an open, inexpensive and flexible platform for autonomy education and research. *In 2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1497–1504. IEEE, 2017.

[109] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark *et al.* : Learning transferable visual models from natural language supervision. *In International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[110] Santhosh K. Ramakrishnan, Devendra Singh Chaplot, Ziad Al-Halah, Jitendra Malik et Kristen Grauman : Poni: Potential functions for objectgoal navigation with interaction-free learning. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18868–18878, 2022.

[111] René Ranftl, Alexey Bochkovskiy et Vladlen Koltun : Vision transformers for dense prediction. *In Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12179–12188, 2021.

[112] Zhenhuan Rao, Yuechen Wu, Zifei Yang, Wei Zhang, Shijian Lu, Weizhi Lu et ZhengJun Zha : Visual navigation with multiple goals based on deep reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, PP:1–11, 03 2021.

[113] Joseph Redmon, Santosh Divvala, Ross Girshick et Ali Farhadi : You only look once: Unified, real-time object detection. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[114] Charles Reinholtz, Dennis Hong, Al Wicks, Andrew Bacha, Cheryl Bauman, Ruel Faruque, Michael Fleming, Chris Terwelp, Thomas Alberi, David Anderson, Stephen Cacciola, Patrick Currier, Aaron Dalton, Jesse Farmer, Jesse Hurdus, Shawn Kimmel, Peter King, Andrew Taylor, David Van Covern et Mike Webster : *Odin: Team VictorTango's Entry in the DARPA Urban Challenge*, pages 125–162. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[115] Shaoqing Ren, Kaiming He, Ross Girshick et Jian Sun : Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[116] Danilo Rezende et Shakir Mohamed : Variational inference with normalizing flows. *In International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[117] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot et Yoshua Bengio : Contractive auto-encoders: Explicit invariance during feature extraction. *In Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 833–840, Madison, WI, USA, 2011. Omnipress.

[118] Clearpath Robotics : Jackal small unmanned ground vehicle. `https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/`, accessed 2023-04-20.

[119] Ethan Rublee, Vincent Rabaud, Kurt Konolige et Gary Bradski : Orb: An efficient alternative to sift or surf. *In 2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

[120] David E. Rumelhart, Geoffrey E. Hinton et Ronald J. Williams : Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[121] Miguel* Saavedra-Ruiz, Sacha Morin* et Liam Paull : Monocular robot navigation with self-supervised pretrained vision transformers. *In 2022 19th Conference on Robots and Vision (CRV)*, pages 197–204, 2022.

[122] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov et Liang-Chieh Chen : Mobilenetv2: Inverted residuals and linear bottlenecks. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[123] Ziad Al-Halah Santhosh Kumar Ramakrishnan et Kristen Grauman : Occupancy anticipation for efficient exploration and navigation. *In Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[124] Nikolay Savinov, Alexey Dosovitskiy et Vladlen Koltun : Semi-parametric topological memory for navigation. *In International Conference on Learning Representations*, 2018.

[125] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh et Dhruv Batra : Habitat: A Platform for Embodied AI Research. *In Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.

[126] Dhruv Shah, Arjun Bhorkar, Hrishit Leen, Ilya Kostrikov, Nicholas Rhinehart et Sergey Levine : Offline reinforcement learning for visual navigation. *In 6th Annual Conference on Robot Learning*, 2022.

[127] Dhruv Shah, Benjamin Eysenbach, Gregory Kahn, Nicholas Rhinehart et Sergey Levine : Ving: Learning open-world navigation with visual goals. *In 2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13215–13222. IEEE, 2021.

[128] Dhruv Shah, Benjamin Eysenbach, Nicholas Rhinehart et Sergey Levine : Rapid Exploration for Open-World Navigation with Latent Goal Models. *In 5th Annual Conference on Robot Learning*, 2021.

[129] Dhruv Shah et Sergey Levine : ViKiNG: Vision-Based Kilometer-Scale Navigation with Geographic Hints. *In Proceedings of Robotics: Science and Systems*, 2022.

[130] Dhruv Shah, Blazej Osinski, Brian Ichter et Sergey Levine : Robotic Navigation with Large Pre-Trained Models of Language, Vision, and Action. *In CoRR*, 2022.

[131] Roland Siegwart, Illah R. Nourbakhsh et Davide Scaramuzza : *Introduction to Autonomous Mobile Robots*. The MIT Press, 2nd édition, 2011.

[132] A. Stentz : Optimal and efficient path planning for partially-known environments. *In Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 3310–3317 vol.4, 1994.

[133] Robin Strudel, Ricardo Garcia, Ivan Laptev et Cordelia Schmid : Segmenter: Transformer for semantic segmentation. *In Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7262–7272, 2021.

[134] Shiliang Sun, Zehui Cao, Han Zhu et Jing Zhao : A survey of optimization methods from a machine learning perspective. *IEEE transactions on cybernetics*, 50(8):3668–3681, 2019.

[135] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva et Dhruv Batra : Habitat 2.0: Training home assistants to rearrange their habitat. *In Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[136] Chongben Tao, Zhen Gao, Jinli Yan, Chunguang Li et Guozeng Cui : Indoor 3d semantic robot vslam based on mask regional convolutional neural network. *IEEE Access*, 8:52906–52916, 2020.

[137] Joshua B. Tenenbaum, Vin de Silva et John C. Langford : A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[138] Joshua B. Tenenbaum, Vin de Silva et John C. Langford : A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[139] Sebastian Thrun, Wolfram Burgard et Dieter Fox : *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[140] Sebastian THRUN, Mike MONTEMERLO, Hendrik DAHLKAMP, David STAVENS, Andrei ARON, James DIEBEL, Philip FONG, John GALE, Morgan HALPENNY, Gabriel HOFFMANN, Kenny LAU, Celia OAKLEY, Mark PALATUCCI, Vaughan PRATT, Pascal STANG, Sven STROHBAND, Cedric DUPONT, Lars-Erik JENDROSSEK, Christian KOELEN, Charles MARKEY, Carlo RUMMEL, Joe van NIEKERK, Eric JENSEN, Philippe ALESSANDRINI, Gary BRADSKI, Bob DAVIES, Scott ETTINGER, Adrian KAEHLER, Ara NEFIAN et Pamela MAHONEY : *Stanley: The Robot That Won the DARPA Grand Challenge*, pages 1–43. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[141] Bo TIAN, Vui Ann SHIM, Miaolong YUAN, Chithra SRINIVASAN, Huajin TANG et Haizhou LI : Rgb-d based cognitive map building and navigation. *In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1562–1567, 2013.

[142] P. VADAKKEPAT, Kay Chen TAN et Wang MING-LIANG : Evolutionary artificial potential fields and their application in real time robot path planning. *In Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, volume 1, pages 256–263 vol.1, 2000.

[143] Aaron VAN DEN OORD, Yazhe LI et Oriol VINYALS : Representation Learning with Contrastive Predictive Coding. *arXiv e-prints*, page arXiv:1807.03748, juillet 2018.

[144] Aäron van den OORD, Sander DIELEMAN, Heiga ZEN, Karen SIMONYAN, Oriol VINYALS, Alexander GRAVES, Nal KALCHBRENNER, Andrew SENIOR et Koray KAVUKCUOGLU : Wavenet: A generative model for raw audio. *In Arxiv*, 2016.

[145] Laurens van der MAATEN et Geoffrey HINTON : Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[146] Ashish VASWANI, Noam SHAZEER, Niki PARMAR, Jakob USZKOREIT, Llion JONES, Aidan N GOMEZ, Łukasz KAISER et Illia POLOSUKHIN : Attention is all you need. *In Advances in neural information processing systems*, pages 5998–6008, 2017.

[147] Jiankun WANG, Tianyi ZHANG, Nachuan MA, Zhaoting LI, Han MA, Fei MENG et Max Q.-H. MENG : A survey of learning-based robot motion planning. *IET Cyber-Systems and Robotics*, 3(4):302–314, 2021.

[148] Sukai WANG, Yuxiang SUN, Chengju LIU et Ming LIU : Pointtracknet: An end-to-end network for 3-d object detection and tracking from point clouds. *IEEE Robotics and Automation Letters*, 5(2):3206–3212, 2020.

[149] Erik WIJMANS, Abhishek KADIAN, Ari S. MORCOS, Stefan LEE, Irfan ESSA, Devi PARIKH, Manolis SAVVA et Dhruv BATRA : Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *In International Conference on Learning Representations*, 2019.

[150] ROS WIKI : Gmapping - ROS wiki. `https://wiki.ros.org/gmapping`, 2023. [Online; accessed 24-April-2023].

[151] Rey Reza WIYATNO, Anqi XU et Liam PAULL : Lifelong topological visual navigation. *IEEE Robotics and Automation Letters*, 7(4):9271–9278, 2022.

[152] Jay M WONG, Vincent KEE, Tiffany LE, Syler WAGNER, Gian-Luca MARIOTTINI, Abraham SCHNEIDER, Lei HAMILTON, Rahul CHIPALKATTY, Mitchell HEBERT, David MS JOHNSON *et al.* : Segicp: Integrated deep semantic segmentation and pose estimation. *In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5784–5789. IEEE, 2017.

[153] Yi WU, Yuxin WU, Aviv TAMAR, Stuart RUSSELL, Georgia GKIOXARI et Yuandong TIAN : Bayesian relational memory for semantic visual navigation. *In Proceedings of the IEEE/CVF international conference on computer vision*, pages 2769–2779, 2019.

[154] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik et Silvio Savarese : Gibson env: Real-world perception for embodied agents. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9068–9079, 2018.

[155] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez et Ping Luo : Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in Neural Information Processing Systems*, 34, 2021.

[156] Wenda Xu, Jia Pan, Junqing Wei et John M. Dolan : Motion planning under uncertainty for on-road autonomous driving. *In 2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2507–2512, 2014.

[157] Ge Yang, Amy Zhang, Ari Morcos, Joelle Pineau, Pieter Abbeel et Roberto Calandra : Plan2vec: Unsupervised representation learning by latent plans. *In Learning for Dynamics and Control*, pages 935–946. PMLR, 2020.

[158] Heng Yang, Wei Dong, Luca Carlone et Vladlen Koltun : Self-supervised geometric perception. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14350–14361, 2021.

[159] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta et Roozbeh Mottaghi : Visual semantic navigation using scene priors. *In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. ICLR, 2019.

[160] Xin Ye, Zhe Lin, Joon-Young Lee, Jianming Zhang, Shibin Zheng et Yezhou Yang : Gaple: Generalizable approaching policy learning for robotic object searching in indoor environment. *IEEE Robotics and Automation Letters*, 4(4):4003–4010, 2019.

[161] Chao Yu, Zuxin Liu, Xin-Jun Liu, Fugui Xie, Yi Yang, Qi Wei et Qiao Fei : Ds-slam: A semantic visual slam towards dynamic environments. *In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1168–1174. IEEE, 2018.

[162] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun et Stéphane Deny : Barlow twins: Self-supervised learning via redundancy reduction. *In International Conference on Machine Learning*, pages 12310–12320. PMLR, 2021.

[163] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr *et al.* : Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. *In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6881–6890, 2021.

[164] Brady Zhou et Philipp Krähenbühl : Cross-view transformers for real-time map-view semantic segmentation. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13760–13769, June 2022.

[165] Yin Zhou et Oncel Tuzel : Voxelnet: End-to-end learning for point cloud based 3d object detection. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4490–4499, 2018.

[166] Xingdong Zuo : mazelab: A customizable framework to create maze and gridworld environments. `https://github.com/zuoxingdong/mazelab`, 2018.

Appendix A

Additional results

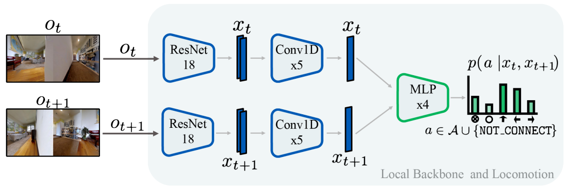
A.1. Network Architectures

We present the network architectures used in our Habitat and real-world experiments in Figure A.1. The architecture used for our GC-BC agent is the same used for O4A's local backbone and connectivity head but without the `NOT_CONNECTED` token.

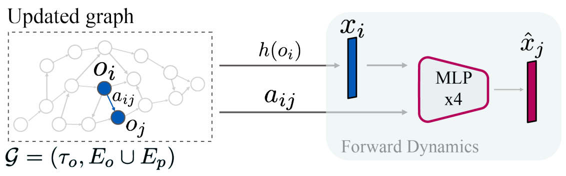
A.2. Extended Navigation Results in Habitat environments

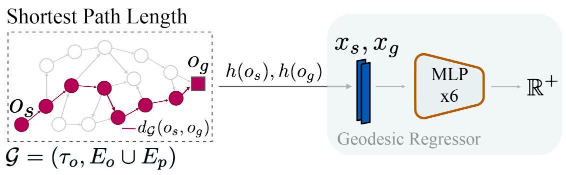
Extended navigation results in Habitat environments are shown in Figure A.2. Each row presents a metric (SR, SSR, SPL and DTG) and columns (from left to right) show the results over different difficulties. Overall, O4A (blue circle) surpass all the baselines and achieves good navigation performance throughout environments without using a graph for navigation[1].

[1]Demonstrative videos with results can be seen in `https://montrealrobotics.ca/o4a/`


o_t
o_{t+1}
ResNet 18
Conv1D x5
x_t
x_{t+1}
MLP x4
$p(a\,|x_t, x_{t+1})$
$a \in \mathcal{A} \cup \{\texttt{NOT\_CONNECT}\}$
Local Backbone and Locomotion

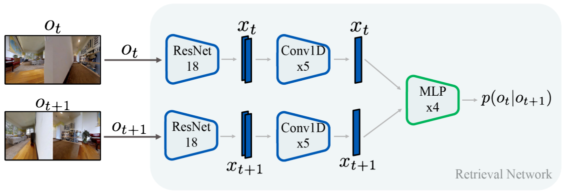
(a) O4A - Local backbone and connectivity head.

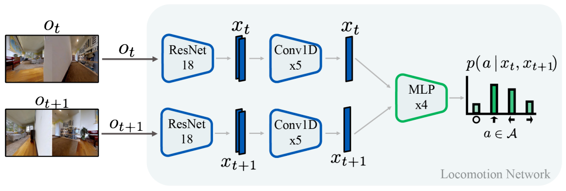




(b) O4A - Forward Dynamics.

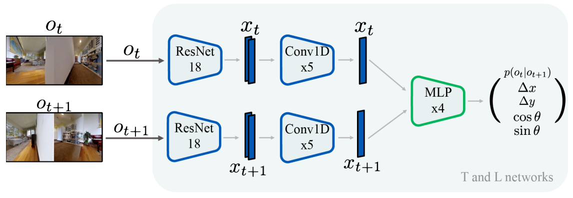
(c) O4A - Geodesic Regressor.





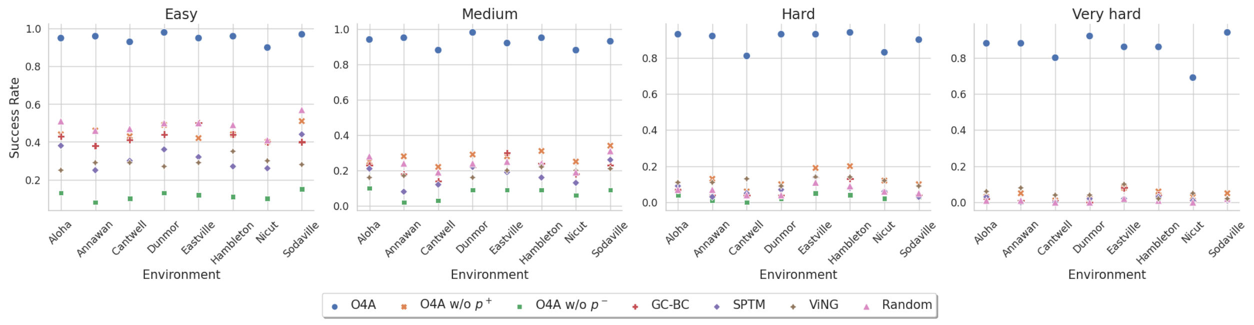
(d) SPTM - Retrieval Network.

(e) SPTM - Locomotion Network.

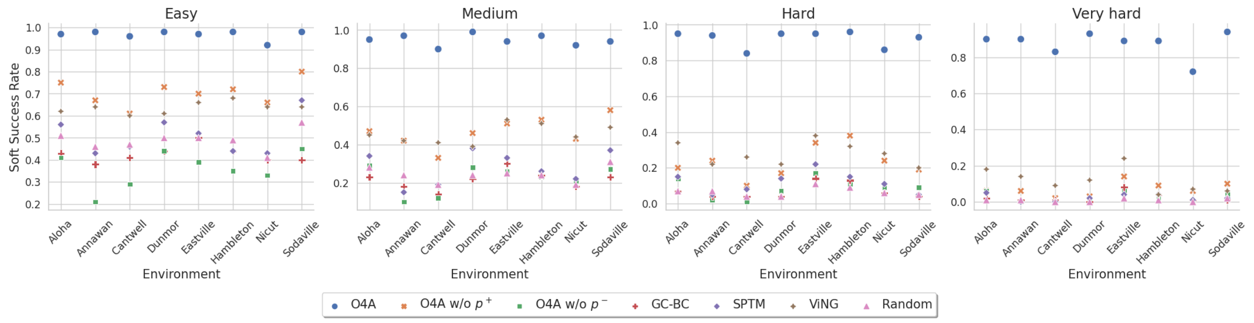

o_t
o_{t+1}
ResNet 18
Conv1D x5
x_t
x_{t+1}
MLP x4
$\begin{pmatrix} p(o_t|o_{t+1}) \\ \Delta x \\ \Delta y \\ \cos\theta \\ \sin\theta \end{pmatrix}$
T and L networks

(f) ViNG - T and L Networks.

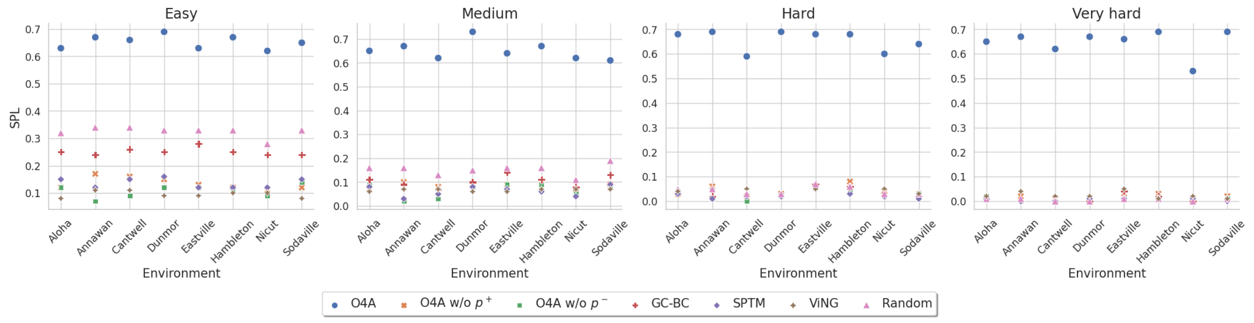
Fig. A.1. Neural architectures used in Habitat (Chapter 4.3.2) and real-world experiments (Chapter 4.3.3). MLP stands for multi-layer perceptron and Conv1D for 1-dimensional convolutions.



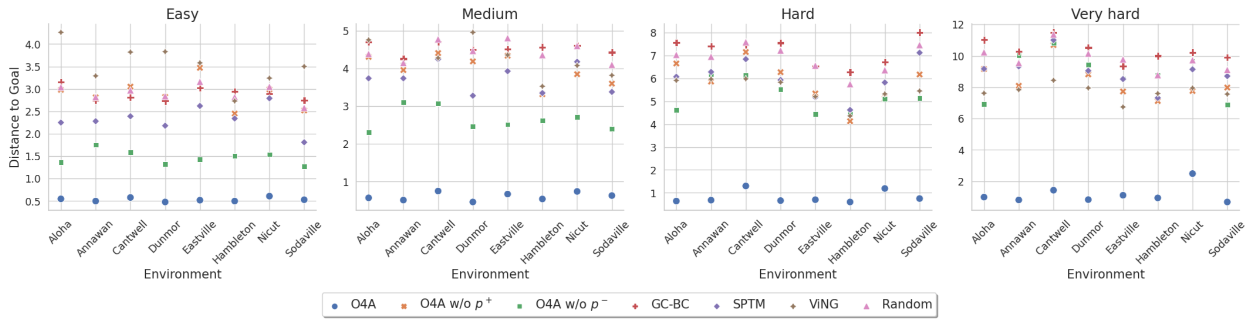
(a) Success Rate for Habitat experiments (higher is better).



(b) *Soft* Success Rate for Habitat experiments (higher is better).



(c) Success weighted by Path-Length for Habitat experiments (higher is better).



(d) Distance to Goal for Habitat experiments (lower is better).

Fig. A.2. Detailed results obtained per environment and difficulties with O4A. (a) Success Rate. (b) *Soft* Success Rate. (c) Success weighted by Path-Length and (d) Distance to Goal.