Université de Montréal

Deep Reinforcement Learning for Multi-Modal Embodied Navigation

par

Martin Weiss

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

Orientation mathématiques appliquées

December 22, 2020

Université de Montréal

Faculté des études supérieures et postdoctorales

Ce mémoire intitulé

Deep Reinforcement Learning for Multi-Modal Embodied Navigation

présenté par

Martin Weiss

a été évalué par un jury composé des personnes suivantes :

Guillaume Rabusseau
(président-rapporteur)

Chris J. Pal
(directeur de recherche)

Gauthier Gidel
(membre du jury)

Résumé

Ce travail se concentre sur une tâche de micro-navigation en plein air où le but est de naviguer vers une adresse de rue spécifiée en utilisant plusieurs modalités (par exemple, images, texte de scène et GPS). La tâche de micro-navigation extérieure s'avère etre un défi important pour de nombreuses personnes malvoyantes, ce que nous démontrons à travers des entretiens et des études de marché, et nous limitons notre définition des problèmes à leurs besoins. Nous expérimentons d'abord avec un monde en grille partiellement observable (Grid-Street et Grid City) contenant des maisons, des numéros de rue et des régions navigables. Ensuite, nous introduisons le Environnement de Trottoir pour la Navigation Visuelle (ETNV), qui contient des images panoramiques avec des boîtes englobantes pour les numéros de maison, les portes et les panneaux de nom de rue, et des formulations pour plusieurs tâches de navigation. Dans SEVN, nous formons un modèle de politique pour fusionner des observations multimodales sous la forme d'images à résolution variable, de texte visible et de données GPS simulées afin de naviguer vers une porte d'objectif. Nous entraînons ce modèle en utilisant l'algorithme d'apprentissage par renforcement, Proximal Policy Optimization (PPO). Nous espérons que cette thèse fournira une base pour d'autres recherches sur la création d'agents pouvant aider les membres de la communauté des gens malvoyantes à naviguer le monde.

Mots-Clés: *Navigation Incarnée*, *Les Réseaux de Neurones*, *Apprentissage par Renforcement*, *Représentations Multimodales*, *La Technologie d'Assistance*, *Aveugles et malvoyants*

Abstract

This work focuses on an Outdoor Micro-Navigation (OMN) task in which the goal is to navigate to a specified street address using multiple modalities including images, scene-text, and GPS. This task is a significant challenge to many Blind and Visually Impaired (BVI) people, which we demonstrate through interviews and market research. To investigate the feasibility of solving this task with Deep Reinforcement Learning (DRL), we first introduce two partially observable grid-worlds, Grid-Street and Grid City, containing houses, street numbers, and navigable regions. In these environments, we train an agent to find specific houses using local observations under a variety of training procedures. We parameterize our agent with a neural network and train using reinforcement learning methods. Next, we introduce the Sidewalk Environment for Visual Navigation (SEVN), which contains panoramic images with labels for house numbers, doors, and street name signs, and formulations for several navigation tasks. In SEVN, we train another neural network model using Proximal Policy Optimization (PPO) to fuse multi-modal observations in the form of variable resolution images, visible text, and simulated GPS data, and to use this representation to navigate to goal doors. Our best model used all available modalities and was able to navigate to over 100 goals with an 85% success rate. We found that models with access to only a subset of these modalities performed significantly worse, supporting the need for a multi-modal approach to the OMN task. We hope that this thesis provides a foundation for further research into the creation of agents to assist members of the BVI community to safely navigate.

Keywords: *Embodied Navigation*, *Neural Networks*, *Reinforcement Learning*, *Multimodal Representations*, *Assistive Technology*, *Blind and Visually Impaired*

Contents

List of tables

List of figures

Liste des sigles et des abréviations

BISM: Blind Industries and Services of Maryland. 21
BVI: Blind and Visually Impaired. 7, 13, 21–24, 40, 67–69, 81

CNN: Convolutional Neural Network. 22, 59, 65

DQN: Deep Q-Network. 29
DRL: Deep Reinforcement Learning. 7, 22, 54, 63, 67

FOV: Field of View. 56–59
FPS: Frames Per Second. 54, 56

MDP: Markov Decision Process. 26, 29

OCR: Optical Character Recognition. 22, 38
OMN: Outdoor Micro-Navigation. 7, 12, 15, 21, 25, 26, 43, 49, 52, 53, 67–69, 83

POMDP: Partially Observable Markov Decision Process. 26, 29
PPO: Proximal Policy Optimization. 7, 12, 25, 29, 43, 46, 47, 54, 59, 67, 85

RL: Reinforcement Learning. 22, 25, 26, 29, 33, 40, 41, 54, 63, 67

SEVN: Sidewalk Environment for Visual Navigation. 7, 67, 68
SLAM: Simultaneous Localization and Mapping. 26, 55

VQA: Visual Question Answering. 21, 22, 39

Acknowledgements / Remerciements

This research was supported by NSERC, PROMPT, CIFAR, MITACS, ElementAI and Humanware.

I would like to thank all my collaborators and co-authors who made this work possible. In particular, I would like to thank Roger Girgis, who contributed his time and effort in the very early days of these works, who has experience developing tools for the blind and visually impaired, and who contributed to the development and application of the reinforcement and representation learning methods. I am so grateful to have had someone like Roger working with me throughout this process. Next, all of the post-doctoral researchers who helped guide this work, Florian Golemo, Margaux Luck, Samira Kahou, Felipe Codevilla, and Joseph Paul Cohen. I'd also like to thank Maxime Chevalier-Bois for her work developing the MiniGrid framework used in the development of the Grid City and Grid Street environments. And of course, thanks to Simon Chamorro who made invaluable contributions to the Sidewalk Environment for Visual Navigation during his internship at Mila.

I would like to thank everyone who helped me to join Mila, in chronological order: Matthew Brightman, for challenging me; Jessica Thompson and Matthew Leavitt for discussing their research and welcoming me to their reading group; David Krueger for replying to my emails and introducing me around; and finally, Joseph Paul Cohen and Yoshua Bengio, who accepted me as an intern to work on a fascinating medical representation learning project.

Next, I would also like to thank François Boutrouille for sharing his expertise with assistive technology for the blind and visually impaired. Additionally, I would like to thank the team at HumanWare for their endless patience and positive attitude while we learned about problems affecting the blind and visually impaired. I would also like to give a special thanks to Maryse Legault for her help with the interview process that motivated this work.

I'd also like to thank all of the professors who educated me, guided me through this process, and answered so many of my questions: Doina Precup, Liam Paull, Simon Lacoste-Julien, Aaron Courville, and Derek Nowrouzezahrai. Finally, I am very grateful to Chris Pal whose support and guidance made this work possible.

Chapter 1

Introduction

In this thesis, we propose an automated system to solve Outdoor Micro-Navigation (OMN) problems of fewer than 50 meters terminating at a unique street address. This task is motivated by the Blind and Visually Impaired (BVI) community, with whom we conducted user interviews to establish the significance of this problem. The core of our solution is a deep neural network trained with reinforcement learning that has access to inputs typically available on a smartphone. We evaluate our proposed solution in simulation environments.

Some BVI people require assistance from sighted people to perform everyday tasks like reading mail, finding objects, and navigating to new places [**119**]. Organizations like the U.S. National Foundation for the Blind and Canadian National Institute for the Blind provide programs to support the BVI community. Some of these programs teach Braille, cane travel, and cooking; more broadly, they instill a sense that students will be able to live a full, productive, independent life after losing their vision [**83**]. To complete the Blind Industries and Services of Maryland (BISM) program, students must find their way around an unfamiliar city, build something using power tools, and cook a formal meal for six. Still, many tasks become more challenging without sight.

BVI people often use smartphones as assistive tools that enable more autonomy and flexibility. Some apps like VizWiz Social [**14**] allow BVI users to connect with sighted volunteers or friends for asynchronous image captioning and question answering. While smartphone mediated question answering can provide significant benefits to many BVI people, interviews with BVI people suggest that they are uncomfortable over-relying on social connections for support due to high perceived social costs [**15**]. Research in psychology has shown that people are often reluctant to ask for help if they believe it may incur costs to themselves or the person asked, or if they are afraid of being judged incompetent [**34**]. While this fear may be remedied through user interface design, an automated Visual Question Answering (VQA) solution remains desirable. Automatic Visual Question Answering (VQA) is an active field of research, and datasets created by VizWiz Social have proven useful for VQA researchers. However, this technology has not yet fully matured as a reliable and

commercially available assistive technology. We discuss the state of the VQA field briefly in Section 2.3.3.

For complex tasks requiring multi-step reasoning, asynchronous remote support and VQA models are ill-suited. Initiatives like BeMyEyes and Aira rely instead on live video calls with volunteers and paid employees to provide a wider range of services. These applications are better suited for tasks like reading the mail (with multiple pages of text and complex layouts) or navigation. Still, it can be difficult to trust the competence and discretion of volunteers, and for-profit services can be quite expensive. In both cases, significant internet bandwidth is required which may constrain the availability of service in some areas. The goal of this work is to lay the foundation for an alternative to existing human-in-the-loop services for on-device navigation for the BVI.

Many mobile applications for the BVI community have already been developed, such as BlindTool [**28**], BlindSight [**67**], SeeingAI [**30**] and EnvisionAI [**60**], which provide on-device object identification, text extraction, and scene description for the BVI community. The computer vision functionality in these apps takes in images and returns a label, object segmentations, or text recognitions. These functionalities are generally implemented by a multi-layer Convolutional Neural Network (CNN). This type of architecture has several characteristics that suit them to vision tasks, including sparse interactions, parameter sharing and equivariant representations [**42**]. Still, the most accurate of these models tend to be computationally intensive, leading to a challenging trade-off between user-privacy, battery usage, and execution time. Further, they require large amounts of training data, and great care must be taken if they are to be used outside of the training data distribution. The latter problem becomes very significant when applications are to be deployed in the real-world.

In contrast to static computer vision tasks like Optical Character Recognition (OCR) or pixel-wise semantic object segmentation, our problem setting presents the significant challenge of making multi-step predictions in a partially observable environment. While navigation tasks have a broad range of possible formulations, we focus on the task where an agent must move around an environment to find a unique target location specified by natural language. Deep Reinforcement Learning (DRL) methods address this problem by jointly learning to extract useful feature representations and to execute policies using these representations. This strategy can also be applied to settings with multiple types of inputs.

Though fast progress is being made on DRL methods, the field is still in its adolescence. RL algorithms require large amounts of experience to learn policies, and are prone to overfitting and instability. This can make the transfer of policies learned in a simulator to reality quite difficult. A large volume of work have been done on this subject, with strategies like using intermediate representations [**81**] (e.g., policies which only see semantic segmentation masks) and domain randomization to improve transfer [**110**]. We provide a background on RL

in Section 2.1, which describes some of the many ways researchers are approaching these problems. We focus specifically on multi-goal navigation strategies relevant to our setting.

Next, we describe the design process undergone at the beginning of this project to identify and understand some problems facing the BVI community. We describe this process in detail for several reasons: 1) to defend the novelty of our approach; 2) to promote the existence of some important open problems facing the BVI; and 3) to validate the needs of the BVI and understand the problem to be solved.

1.1. Design Process

Our goal with this design process was to verify the existence and significance of approachable problems affecting the BVI using a combination of market research and interviews with BVI people and orientation and mobility specialists. Figure 1.1 shows what we consider an approachable problem. This process revealed three candidate problems for the BVI, namely intersection crossing, obstacle avoidance and navigating to addresses. In each of the following sections, we illustrate our design process by discussing how it shaped this study and our approach to the finding addresses problem. We hope that this case study can, for future creators of assistive technology, serve as an illustration of how to begin an applied research effort.

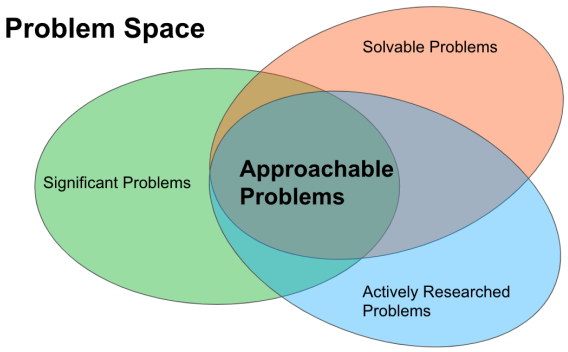


Fig. 1.1. Problem space. We define approachable problems as those which are useful to the BVI community, can be solved with guidance from a sighted person via a device or in-person, and those related to active research areas of computer science. It is risky to work on problems that are not at least somewhat well-studied, or that you can not situate within a well-studied domain.

Interviews: We conducted several interviews with BVI people, experts in the field of accessibility, and developers of mobile apps for the BVI. We sought to understand the problems BVI people encounter, how they solve them today, and which solutions experts in the field are already investigating. We learned, with regards to navigation, that BVI people typically use white canes or guide dogs to avoid obstacles, and navigation tools like Google Maps and Blindsquare for route planning. For challenging visual tasks they use

services like BeMyEyes or Aira which are human-in-the-loop systems that send images or videos to assistants who can cost over $1/minute. We paid special attention to tasks which induced high willingness to pay for real-time human assistance because this indicates they are significant problems that may be solvable with machine learning methods. An example interview focused on navigation is contained in the supplemental material Appendix A.

Market Research: The goal of our market research was to find out which commercial solutions exist, which worked best, and why. We were also interested in the challenges faced by developers. We began by compiling a dataset of several dozen available products and services for the BVI community. While creating this list of services (available in Appendix B), we created an informal ontology of provided features and recorded user feedback. We selected the most relevant services from this list, contacted the developers, and were able to conduct interviews with 12 of those. By doing so, we learned that many assistive technologies are made by individuals and relatively small companies (like Aira.io). Though their attitudes varied, the developers were generally very open to talking to us. The majority of the companies we emailed were focused on providing solutions to text summarization, vision, and navigation tasks. However, we did not see any applications providing an automated service for address finding.

Approachable Problems: Our interviews and market research revealed three significant problems which are not yet fully resolved by available services or methods. These problems were constrained to sidewalk navigation. **Intersection crossing** happens when a person arrives at an intersection and wants to cross it. To safely accomplish this task the user must be aware of the spatial configuration of the intersection, the intersection's signalling pattern, the correct orientation to follow while crossing, (i.e., is there a crosswalk, underground passage, pedestrian bridge?), and the time when it is safe to begin crossing. **Obstacle avoidance** happens when a person is walking on a sidewalk or crosswalk and is confronted by obstacles such as fire hydrants, parking meters, road work, stairs, etc. Today, this use case is primarily solved with traditional assistive tools, such as guide-dogs or canes. Other options like the BeAware app use a combination of beacons planted in the environment and bluetooth. **Outdoor micro-navigation** happens when a person has a unique destination such as a restaurant, physician's office, or public transit station. The most difficult part of this task is the last few meters of navigation. BVI people often use smartphone apps like Google Maps that leverage GPS, but this only partially solves the problem. Once near the desired location, BVI people usually have to consult a scene description application or nearby people to make sure they are in the right place. However, objects of interest such as building entrances, ramps, stairs, house numbers, bus stops, and subway entrances are often missing.

After investigating those problems, we decided to pursue the outdoor micro-navigation problem because other groups [**90, 35**] had provided significant treatment for the intersection crossing problem and the obstacle detection problem already has workable solutions in the

form of white canes and guide dogs. Furthermore, the OMN address finding problem seemed to have significant individual and commercial interest with limited prior work.

1.2. Problem Statement

Given the early stage of the technologies available for solving the OMN problem, we decided to propose and evaluate a solution in simulation. Before embarking on the construction of this simulator, we performed a literature review summarized here and in Chapter 2 that showed that there were no existing simulators suitable to evaluating models for our problem setting. Therefore, the two primary problems to overcome were (1) the construction of a simulator allowing us to investigate and train an RL agent to accomplish OMN tasks, and (2) the construction, training, and assessment of models on these tasks.

Construct an Outdoor Micro-Navigation Simulator: The pedestrian navigation system is responsible for making a sequence of decisions to safely find their goal. In order to navigate to an address in a way similar to sighted humans, we determined that the model would need access to multi-modal inputs including images, scene-text, and gps. Further, we would need an outdoor environment containing a realistic data distribution. This environment could contain computer generated graphics or real-world imagery, but ideally the state transitions, goal distributions, and environmental structure would closely mirror that of the real world setting.

Visually realistic simulators for autonomous driving at first appear to be a useful setting in which to train our agent, but the sidewalk and houses lack necessary details for our task [**37, 103, 95**]. Real-world datasets that contain a realistic distribution of street names, house numbers, and buildings are almost exclusively captured from vehicle-mounted cameras on public roads [**126, 29**]. Other research into visual navigation has also generated many simulators [**21, 121, 107**], but these are limited to indoor domains. Some specialized datasets for street crossing and localization exist, but they are static and contain images only from intersections [**33, 90**]. Though many adjacent RL simulators exist, we determine in Section 2.4 via a literature review that no suitable environment exists that contains the necessary resolution, density, and modality of data for training RL agents for the desired task. Chapter 3 proposes several simple grid-world environments and initial experiments, while Chapter 4 explores an environment using real image data.

Learn a Multi-Modal Navigation Policy: Reinforcement learning methods [**108**] are often used for navigation tasks. The specifics of these environments vary widely and tend to require different algorithms. For example in settings where the agent's goal is specified at the beginning of an episode, methods like Universal Value Function Approximators (UVFA) [**99**] and successor representations [**6, 66, 13**] can be effective. Deep Q-Networks [**79, 51, 128**] with modifications like experience replay [**79, 3, 20, 97**], or policy improvement methods like PPO [**101**] can be effective in settings with high dimensional observations. The vision and

language navigation setting often features an agent that follows natural language instructions to achieve a specified goal [**2, 22, 52**]. These agents often leverage prior information [**124**], multi-modal sensor data [**71**], or expert demonstrations [**25**] to achieve the goal.

Learning a robust navigation policy is a difficult task. Classical computer vision methods like Simultaneous Localization and Mapping (SLAM) have traditionally been used in autonomous robotics to create maps and plot the agent's location on that map. These methods can also be used to detect important task-relevant objects, like obstacles or roads or goals. SLAM is often used in conjunction with planning and control algorithms to achieve autonomous navigation. Learned models, on the other hand, do not use a rigid pre-defined algorithms for achieving their goals. Instead, they learn a representation of the world, and the effect their actions have on the world, from experience.

RL algorithms, or agents, learn by interacting with their environment. Their tasks often take the form of sequential decision making problems, formalized by Markov Decision Processes (MDPs). Generally, the goal of these algorithms is to maximize a reward, which may be received from the environment or intrinsically determined. In this thesis, we focus on discrete Partially Observable Markov Decision Processes (POMDPs), which are a challenging setting that closely models the types of problems faced by agents in the real-world.

1.3. Thesis Statement

This work introduces a formulation for an Outdoor Micro-Navigation (OMN) task, and a novel method for performing this task. To specify the task, we develop and release code for several grid-world environments and a high-dimensional environment using real-world data collected from a sidewalk perspective. To perform the task, we use reinforcement learning and neural network methods to model the environment and execute the navigation task. By combining three observation modalities (images, scene-text, and GPS), we see a significant improvement over models utilizing only a subset of these modalities.

1.4. Statement of Contributions

The main contributions of this thesis are as follows:

- A review of reinforcement learning, multi-goal navigation methods, computer vision literature, and vision and language navigation environments (chapter 2).
- Grid-world environments for RL that simulate sidewalk navigation (chapter 3).
- A visually realistic deep reinforcement learning simulator for pedestrian navigation (chapter 4).

Certain aspects of this thesis are taken from works that are in preparation for publication or have been published. In particular, the introduction includes modified parts of [**119**] and [**118**]. The majority of Chapter 4 is reproduced from [**118**]. The author of this thesis was

the lead author of all these works, and the collaborators acknowledge the use of these works in this thesis.

Chapter 2

Background and Related Work

This chapter builds up the notation, background, and related literature for the Reinforcement Learning and computer vision methods that are used and extended in chapters 3 and 4. The multi-disciplinary nature of the problem setting requires a description of modern reinforcement learning techniques (see Section 2.1), which includes a formalization of MDPs, POMDPs, Deep Q-Network (DQN) models, and Proximal Policy Optimization (PPO). These last two subsections describe two perspectives on the central problem of RL, namely that given an observation we may either predict action values, or policies. These approaches are often called value-based and policy-based methods. In Section 2.2 we focus on methods developed to learn flexible models, capable of achieving different goals at run-time. Some of these methods focus on explicitly disentangling the state and goal representations, as in Universal Value Function Approximators [**99**] and the Successor Features [**32**] line of work. Others such as Hindsight Experience Replay [**3**] focus on improving sample efficiently and learning in a domain where rewards are sparse and binary. Section 2.3 discusses the history of computer vision, building up to the three key domains relevant to the address finding task: object detection, scene-text recognition, and visual question answering. In Section 2.4, we summarize the literature on vision and language navigation simulators.

2.1. Reinforcement Learning (RL)

Reinforcement Learning (RL) is the study of how to behave in order to maximize rewards. In RL parlance, we aim to learn a policy for getting large rewards given some interactions with the environment. There are many kinds of environments that RL algorithms may be applied to, including spatial, video game, economic, and text. The pre-requisites for using RL methods are simple: a state space, a reward function, an action space, and a transition function. As described in [**108**], a Markov Decision Process (MDP) is a 5-tuple:

$$(S,A,T,R,\gamma)$$

where S is a finite set of states, A is a finite set of actions, T is a transition matrix ($S \times A \times S$) defining the probability that taking action $a \in A$ from state $s \in S$ will lead to $s' \in S$, here we refer to the reward taken at a specific timestep as R_t, $R(s,a)$ is the expected immediate reward generated by taking action a in s, and $\gamma \in [0,1]$ is the discount factor. Because the experiments in Chapters 3 and 4 contain deterministic actions (i.e., there is no action noise), we present a simplified version of the classic reinforcement learning equations.

The challenge with MDPs is to maximize the sum of future discounted rewards,

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... + \gamma^{T-1} R_T = \sum_{k=0}^{T} \gamma^k R_{t+k+1}$$

where T is the last timestep in the episode. We only address finite trajectories here because navigation tasks have an episodic nature; it is natural to think of them as a finite trajectory of states and actions ending at a goal state. We say that a policy π maximizes the reward if $\pi(s)$ yields the action which maximizes the sum of future rewards $\forall s \in S$.

In reinforcement learning it sometimes makes sense to incorporate a predictive model of the environment into the agent. We call these approaches model-based. In the model-based setting, we either assume knowledge of environmental dynamics like transition and reward functions or estimate them. Values can then be computed using methods like Monte Carlo tree-search or dynamic programming. The alternative approach is model-free reinforcement learning which learns value functions directly from the sampled trajectories. We discuss some of each type of algorithm in later sections.

In order to compare policies we find it useful to employ state-value functions $v_\pi(s)$ which describe the expected value of the discounted return for a given policy π starting in some state s:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) \left(r + \gamma v_\pi(s')\right)$$

where $\pi(a|s)$ is the probability of selecting action a while in s under the policy π, and r is the reward for transitioning from s to s'. For environments with large state spaces, we typically model value functions using a function approximator $V(s,\theta)$, such as a linear combination of features or a neural network parameterized by θ.

It is also useful to define the action-value function, also known as the Q-function $q_\pi(s,a)$, as the value of taking an action a from state s as determined by the policy π:

$$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)(r + \gamma v_\pi(s'))$$

where $p(s',r|s,a)$ is the probability of transitioning from s to s' and receiving reward r when taking the action a from state s. The Q-function may also be represented by a function approximator parameterized by some θ, as is discussed at length in Part II of [**108**].

In MDPs, we assume perfect information about the state of the environment. This is a reasonable assumption for games like chess and go, but an unreasonable one for poker, Starcraft, and many embodied tasks. To approach this sort of problem, we follow [**51**] to define a Partially Observable Markov Decision Processes (POMDP) as a 7-tuple extension of the MDP, (S,A,T,R,γ, Z, O), where Z defines a latent observation space with observations drawn $o \in Z$ and O is a set of observation probabilities conditioned on the state $s \in S$. For navigation tasks, these observations generally take the form of sensor readings from cameras, depth sensors, or ultrasonics.

2.1.1. Deep Q-Networks (DQNs)

A landmark work from 2013 demonstrated that one RL algorithm can solve multiple Atari games from just pixels and rewards [**79**]. This work also showed that Atari game screens with sufficient pre-processing become extremely high dimensional MDPs where each state is comprised of four game-screen images resized to 84 by 84 pixels and converted to 256 level gray-scale. This results in an intractable $256^{84\times84\times4}$ possible states. Clearly, it is infeasible to sample each state-action pair, so to solve the problem of learning the Q-values the authors passed these states to a deep convolution neural network function approximator, call this parameterization θ, that output a Q-value for each action in the Atari action space. Because Q-values take on real values we can view this as a regression task and use a squared loss:

$$L(s, a \mid \theta) = \Big[R(s, a) + \max_{a'} \Big(q_\pi(s', a' \mid \theta)\Big) - q_\pi(s, a \mid \theta)\Big]^2$$

where $q_\pi(s, a|\theta)$ is our prediction and $R(s, a) + max_{a'} q_\pi(s', a'|\theta)$ is our target. This target is not subject to optimization itself using the loss gradient, but provides the optimal action and return for our learning signal.

It is important to note that another major contribution of this paper was experience replay, which combats problems associated with correlated data and non-stationary distributions by sampling past experiences to determine updates. There is a long history of combining Q-learning with experience replay and simple neural networks [**70**] and more recent work has showed that this technique can scale to high-dimensional state spaces [**79**]. Hindsight Experience Replay (HER) [**3**] is a recent development that enables RL algorithms to learn from sparse reward signals by retroactively imagining failed trajectories as successful ones. Policies trained using HER take as input a current state and a goal state like UVFA [**99**], but suffer an additional requirement that every state must satisfy some goal and does not specify a universal goal representation.

DQNs were later applied to POMDPs in a new environment named Flickering Pong, a modification of Atari Pong where the screen is either fully revealed or fully obscured with a probability $p = 0.5$ [**51**]. This work also introduced Deep Recurrent Q-Networks (DRQN), a model that adds a recurrent Long-Short Term Memory (LSTM) module at the end of the

standard convolutional DQN model architecture. The proposed model recurrently integrated arbitrarily long histories of observations to estimate a policy that was more robust to partial-observability than DQN. When DQN and DRQN were compared on MDP environments their performance was roughly equivalent.

Further work extended the DRQN model to take as input actions as well as the observation history [**127**]. The action-based Deep Recurrent Q-Network (ADQRN) model encodes actions via a fully connected layer, while in parallel passing the image observation through three convolutional layers. The model then passes these embedded action-observation pairs to an LSTM and fully connected layers to compute Q-values identical to the DQN model.

2.1.2. Proximal Policy Optimization (PPO)

In contrast to these value-based methods, PPO is a policy gradient method, which means that instead of using a value function, we directly optimize the policy [**101**]. It is also an example of an actor-critic architecture, where we have two networks: an actor which controls the policy, and a critic which measures how good the policy's actions are (essentially approximating a Q-value).

We say that the probability ratio between the old and new policy is:

$$r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} \tag{2.1.1}$$

and the objective function we would like to maximize is

$$J(\theta) = \mathbb{E}[r(\theta)\hat{A}_{\theta_{old}}(s,a)] \tag{2.1.2}$$

where $\hat{A}$ is the advantage function $A(s,a) = q(s,a) - v(s)$. Updating our parameters using this object function can lead to policy instability and large changes in the distribution of data generated by the environment because there is no limitation on the distance between the old and new parameters θ_{old} and θ.

PPO solves this problem by using a clipped surrogate objective function [**101**]. This is a constraint on the way we train the actor, requiring the policy updates to be small, specifically within the bounds of some hyper-parameter, ϵ. Resulting in:

$$J(\theta) = \mathbb{E}\Big[\min(r(\theta)\hat{A}_{\theta_{old}}(s,a), \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{\theta_{old}}(s,a))\Big] \tag{2.1.3}$$

A somewhat related concept for slowly modifying a policy is curriculum learning. The idea is to start small and learn the easy aspects of a task, then gradually increase the difficulty level [**8**]. Taking for example a navigation task, in curriculum learning we want our agent to accurately navigate from a start position to any goal location in the environment. However, if the initial task is too difficult, and a positive reward is only given upon completion of the task, the agent may fail to learn. We would have to train the simulator for a very long time before a random policy would ever find any gradient from which it may update its policy.

Furthermore, a uniform random sampling of goal locations will on average yield goals that are quite distant from the start location, and nearby goals would be infrequently sampled.

2.2. Methods of Multi-Goal Navigation

There are many ways to tailor RL methods for the multi-goal setting. Much work has been done to learn models which can be tailored at runtime to achieve different tasks. A naive solution would be to add world knowledge to the model, but this will restrict the space of policies which may be learned and make it more difficult for the model to perform well to different tasks or the real world. Instead, Universal Value Function Approximators [**99**] proposes to learn state and goal embeddings which enables their method to better generalize over goals. Automatically discovering and encapsulating the environmental dynamics is perhaps an even more attractive option, and one which we discuss in the sub-section on Successor Representations (Section 2.2.2). Playing back experiences that are relevant to the agent's level of knowledge is another way to more efficiently use interaction with the environment, and to learn to achieve goals in a sparse reward setting. We discuss this approach in more depth in our sub-section on Hindsight Experience Replay (Section 2.2.3). The next sections will delve into these topics and lay the groundwork for future work.

2.2.1. Universal Value Function Approximators (UVFA)

Approximating value functions, the expected sum of discounted future rewards, is a core concept in reinforcement learning. We often try to estimate the value function for a state s using some model with parameters, $v_\theta(s)$. This function can be extended to generalise over goals, $v_\theta(s, g)$. [**99**] formalized this concept and developed an efficient technique for supervised learning of UVFAs by factoring observed values into embedding vectors for states and goals, then learning a mapping from s and g to these factored embeddings.

One observation from this work was that there are many combinations of states and goals, and while an agent usually will only see a small subset of these combinations we would like to generalize across all combinations. A contribution of this paper was to propose a novel architecture wherein they decompose this regression problem into two stages. They view the data as a sparse matrix where each row contains data from a state and each column contains data for each goal. First, the authors find a low-rank factorization of the matrix into state embeddings $\phi(s)$ and goal embeddings $\psi(g)$. In the second step, the authors learn a non-linear mapping from states s to state embeddings $\phi(s)$ and from goal g to goal embeddings $\psi(g)$. This approach learned UVFAs an order of magnitude faster than naive regression in their experiments.

In the navigation setting it is natural to generalize over goals. We can clearly see that two goals should be similar if they are spatially close. For example, if we take two specific

gates at an airport as our target locations, we can see that the ability to successfully navigate from our home to one of these gates should help when navigating to the other. The next chapter will develop several environments that enable us to further explore this intuition.

2.2.2. Successor Representations (SRs)

First proposed in [**32**], successor representations (SR) factor the value function into a predictive representation of the environment and a reward function. This representation of the environmental dynamics is generally a vector of expected discounted future state occupancies. The reward associated with the transition (s, a, s') can then be represented as the dot product of a vector of features of the transition with a vector of weights, i.e.

$$R(s, a, s') = \phi(s, a, s')^\top \mathbf{w}$$

where $\phi(s, a, s') \in \mathbf{R}^d$ are features of (s, a, s') and $\mathbf{w} \in R^d$ are weights. Now we follow [**6**] and simplify the notation by letting $\phi_t = \phi(s_t, a_t, s_{t+1})$, and specify by $\mathrm{E}^\pi[\cdot]$ the expected value of a random variable given that the agent follows policy π, and t is any time step. Here, we also refer to rewards sampled at timestep t from R as r_t. Then we can define our action-value function as,

$$\begin{aligned}
q^\pi(s, a) &= \mathrm{E}^\pi \left[r_{t+1} + \gamma r_{t+2} + \ldots | S_t = s, A_t = a \right] \\
&= \mathrm{E}^\pi \left[\boldsymbol{\phi}_{t+1}^\top \mathbf{w} + \gamma \boldsymbol{\phi}_{t+2}^\top \mathbf{w} + \ldots | S_t = s, A_t = a \right] \\
&= \mathrm{E}^\pi \left[\sum_{i=t}^{\infty} \gamma^{i-t} \boldsymbol{\phi}_{i+1} | S_t = s, A_t = a \right]^\top \mathbf{w} \\
&= \boldsymbol{\psi}^\pi(s, a)^\top \mathbf{w}
\end{aligned}$$

Where the ith component of $\boldsymbol{\psi}^\pi(s, a)$ is equal to the discounted future sum of $\boldsymbol{\psi}_i$ when following policy π starting with (s, a).

The SR paradigm was more recently extended to operate on high-dimensional observations using a neural network architecture [**66**]. This approach, which we will call Successor Features (SF), provides increased sensitivity to changes in the reward function and the ability to extract bottleneck states, or candidate sub-goals, that improve exploration. This work was theoretically grounded by [**6**]'s extension of the Bellman Policy Improvement theorem [**7**] to the multi-task setting with several proposed models that combine SFs and Generalized Policy Improvement (GPI). Most recently, [**13**] combined UVFAs and GPI to create Universal Successor Feature Approximators (USFA) which generalize over both goals and policies.

2.2.3. Hindsight Experience Replay (HER)

In sparse-reward settings like our navigation task, we find reinforcement learning algorithms struggle to learn policies [**3**]. A common solution is to engineer shaped reward functions that

reflect the task, but also guide policy optimization. The goal of Hindsight Experience Replay (HER) is to develop an algorithm that can learn directly from sparse reward signals, which it accomplishes by retroactively imagining failed trajectories as successful ones.

HER follows UVFA [**99**] by training policies which take as input the current state and a goal state. After experiencing some episode $(s_0, s_1, \ldots, s_T)$, we store every transition, $s_t \rightarrow s_{t+1}$, not only with the original goal but also with a subset of other goals. The authors note that the goal being pursued changes the agent's actions, but not the environment dynamics, and therefore we can replay each trajectory with an arbitrary goal assuming the use of an off-policy RL algorithm like DQN [**79**].

HER improves sample efficiency in this setting, and makes learning possible when the reward is sparse and binary. The authors show that training an agent to perform multiple tasks can be easier than training it to perform one task. Applications of HER may be seen as an implicit curriculum as the goals used for replay naturally shift from ones which are simple to achieve even by a random agent to more difficult ones.

The authors assume that "given a state s we can easily find a goal g which is satisfied in this state" [**99**]. This requires a total map from goals to states which may not be easy to define for some environments. The assumption made by HER does not hold in navigation tasks, that each state visited can be viewed as a goal state. Navigation tasks naturally have a sparse and binary reward, either you find your target location (or a location within the equivalence class of your target) or you do not, and not all locations are desirable targets.

Recent work proposed an extension of HER that uses language to represent goals [**20**]. In their framework, whenever an agent finishes an episode, a teacher gives advice in natural language to the agent based on the episode. The agent takes the advice to form a new experience with a corresponding reward, alleviating the sparse reward problem. To explicitly apply this method to POMDPs, the authors ask the teacher to give advice based on the history of states and actions during the episode, i.e., $g_{alt} = T(s_0, a_0, ..., s_T)$. While language provides a compact and universal representation for the goals it can sometimes be expensive to obtain or unsuited to the task.

Visual Hindsight Experience Replay extends HER to a range of robotic and visual tasks by using a Hallucinatory Generative Adversarial Network (HALGAN) to minimally alter images along failed trajectories such that it appears to approach and achieve the goal [**97**]. This method is best suited to tasks where we do not have an exact specification of the goal during execution, and where many states may not easily map to goals. HALGAN is trained using near goal images, where the relative location of the agent to the goal is known.

The network architecture for the generative model used in this work was an improved Wasserstein Auxiliary Classifier GAN (W-ACGAN) [**85, 45**]. They selected this architecture because W-ACGANs can create realistic looking outputs and have the ability to condition the generated images on a desired class. Another requirement for this generative model is the

ability to consistently hallucinate the goal in the same absolute position throughout the failed trajectory. The authors of that work note that because UVFAs do not extend to the visual setting where the goal location is unknown (and must be identified within the environment), the agent's policy in their work is solely condition on its state.

2.3. Computer Vision and Perception

One of the earliest computer vision projects was created by Seymour Papert in 1966 and given to a group of students [**86**]. The goals of this project included basic foreground and background segmentation, analysing scenes with simple non-overlapping objects of distinct uniform color and texture, with future work to extend the system to more complex objects. Over 50 years later, we continue to see rapid and significant improvements in computer vision applications and models.

By the late 1960's and early 1970's, some of the foundational techniques for computer vision were already being developed including the Sobel-Feldman operator [**55**] and Prewitt operator [**88**] for edge detection. These operators are based on convolving the image with a small filter which yields a gradient approximation for horizontal and vertical changes. While these operators may result in crude approximations, and can show poor performance in high-frequency regions, they are very fast to compute. Further, they laid the foundation for new convolutional and learning-based methods that would follow.

Machine learning-based methods for computer vision have gone through many iterations. Many of these use neural network models, and attempts to describe algorithms for motion detection mimicking insect visual systems using hidden units extends at least as far back as the first International Congress on Cybernetics in 1956 [**50**]. In the 1980's, an ambitious work attempted to integrate the fields of computation, psychophysics, physiology, and biophysics [**61**]. They described the advances of object and motion detectors, the recovery of 3-dimensional structure from motion, and how these models may explain the behavior of animal visual cortices. In the early 2000's, Viola and Jones published a method for face detection using Haar-like features [**115**], while David Lowe published the Scale-Invariant Feature Transform (SIFT) algorithm for extracting useful image features. These methods rely heavily on hand-crafted features. In fact, the dominant image classification methods relied on hand-crafted features until 2012, when AlexNet [**65**], a deep convolutional neural network with 60 million parameters trained exclusively on RGB data outperformed all other methods in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC2012) [**96**]. This breakthrough in computer vision set the stage for modern era of computer vision with many tasks and deep neural network models.

In the following sections we discuss the sub-fields of computer vision relevant to the perceptual aspects of the sidewalk navigation task. In particular, we discuss object detection (Section 2.3.1), scene text recognition (Section 2.3.2), and visual question answering (Section

2.3.3). Each is a significant sub-field of computer vision, and due to voluminous research done in these areas we can only provide a short treatment of their state. We focus on recent works which optimize deep neural network models and have generally shown strong performance. Still, it is important to have a sense of what models are used for sensory perception and the data structures for scene representation.

2.3.1. Object Detection

Unlike image classification tasks which simply require the model to output a label, object detection tasks additionally require the model to indicate a localization in the form of a bounding box composed of x and y pixel coordinates of the top-left corner, the width, and the height of the box. Detecting objects like house numbers, doors and street name signs is necessary in order to complete the navigation task of finding the entrance to a specific address; ideally, solutions will be efficient enough to run in real-time on a mobile phone while performant enough to feel reliable to users. Modern solutions to this sort of problem tend to fall into two main categories: single-shot and two-stage methods.

The two-stage methods refer to the serial computation of region proposals, followed by a classification of each proposed region, where regions are output if the classifier's feature response is large enough. These methods trade a high cost in computation for high performance. Three models published in 2014 and 2015 popularized this style of model. The first, R-CNN [**41**], used selective search [**113**] to generate a fixed size number of regions, warped these regions into squares, used a CNN to extract features, then fed these features into a Support Vector Machine to predict the presence of an object. Fast R-CNN [**40**], and Faster R-CNN [**94**] made significant efficiency improvements to the R-CNN pipeline, replacing the fixed selective search algorithm by a learned (faster) approach which did not rely on a fixed number of regions and proposing a novel Region of Interest pooling layer.

One-shot object detectors like YOLO [**92**], SSD [**73**], SqueezeDet [**120**] and DetectNet [**109**] are faster and more suitable for mobile. They require only a single pass through the neural network using fixed grid detectors that allow the prediction of all bounding boxes at the same time. Those models are composed of a body network, usually pre-trained on a large image classification dataset like ImageNet, that acts as a feature extractor and a head network that detects objects. However, those networks can be too large for mobile deployment. Instead, it will be preferred to use an architecture specifically designed for mobile like SqueezeNet [**56**] that uses fire modules, SEP-NET [**68**] that uses filter convolutions and pattern residual blocks, MobileNet V1 [**54**] that is based on depth-wise separable convolutions or MobileNetV2 [**98**] that adds linear bottleneck and expansion convolution to MobileNetV1[**54**]. SDDLite [**98**] has also been proposed as an object detection model and uses separable convolutions.

Both categories of object detection model can benefit from multi-task prediction [**125**], including joint object detection, scene classification and semantic segmentation. Extensions to PASCAL VOC 2010 detection with pixel-wise semantic categories, showed improvements enabled the development of models that showed that adding a simple contextual feature produces nearly as much improvement as more sophisticated methods [**80**]. Other extensions to this task include 3-D bounding box prediction from RGB-D [**69**] and RGB images [**24**].

Armed with a plethora of vision models, we now turn to object detection datasets. An examination of the classes covered in general object detection benchmarks datasets like ImageNet [**96**] reveal that they lack those object classes necessary for pedestrian navigation. Much work has focused on pedestrian detection [**36**] and surveillance [**49**], and datasets for autonomous vehicles like ApolloScape [**117**] provide some classes of interest like sidewalks, bridges, walls, fences, traffic signs and signals. However, these classes are not comprehensive or fine-grained enough to achieve our tasks of interest and all these images are taken from a raise perspective, not that of a pedestrian. For house numbers, there is also the Street View House Number dataset, but the images are tightly cropped for text recognition tasks, not text detection [**84**].

Different problems require models with different trade-offs between per-class accuracy, speed, and size. High accuracy models like Yolo [**92**] are too large and slow for real-time usage on mobile devices. Models specially designed for use on mobile devices, like MobileNet [**54, 98**] and SSD [**73**] tend to be less accurate, but faster and smaller. Though these models sometimes outperform human performance in the supervised learning setting, they continue to struggle in a variety of situations which can expose high false-positive rates, difficulty with small or out-of-distribution objects, and fine-grained classification.

Therefore, we decided to label in our Sidewalk Environment for Visual Navigation (Chapter 4) some of the relevant object classes for sidewalk navigation that are missing in other datasets, e.g., doors, house numbers, and street name signs.

2.3.2. Scene Text Recognition

Optical Character Recognition (OCR) is the task of recognizing letters in images, thereby converting this visual information into machine-encoded text. The classic setting focused primarily on recognizing text in images of printed paper records with varying fonts. In contrast, scene text detection and recognition is a much more difficult version of this task which may require the model to overcome occlusion, curved and oriented text, as well as numerous distractors.

Addresses usually contain a street name and house number that manifest as visible text in the environment. This text, as well as other scene text like business names, are useful landmarks for localization. Extracting incidental scene text can be quite difficult because it appears in the scene without the user having taken any specific prior action to cause

its appearance or improve its positioning in the frame. Datasets of annotated images with real text [**116, 89**] and synthetically overlayed text [**57, 46**] have resulted in useful models, but modest end-to-end performance on the ICDAR 2019 Large-Scale Street View Text with Partial Labelling challenge indicates that this problem remains open [**72, 129, 5**].

Fig. 2.1. Scene text detections. A cropped image taken with the Vuze+ camera from the SEVN dataset proposed in Chapter 4. Text and house numbers are clearly visible, with rectangular bounding boxes for each house number. We also provide hand-annotated word recognitions associated with each bounding box detection.

Models which perform well in this setting are usually composed of an object detection model followed by an Optical Character Recognition (OCR) model, though there is growing interest in single-stage methods [**19**]. Focusing on the OCR models in not incidental scene text extraction tasks, [**4**] recently highlighted the fact that many of the OCR pipelines like CRNN [**104**], Rosetta [**12**] and STAR-Net [**74**] were incomparable due to differences in their evaluation settings and proposed a uniform four-part modular pipeline for OCR. Furthermore, they show that those models struggle with the recognition of unusual fonts, vertical text, special characters, occluded text, and low resolution images which often appears to be the characteristics of the extracting incidental scene text that interest us. Some works have proposed specialized scene text localization methods with the BVI in mind, like [**38**], which focused on objects like license numbers of buses, traffic and store signs.

2.3.3. Visual Question Answering

Visual Question Answering (VQA) is a class of natural language question answering tasks solved by searching and reasoning over the contents of an image. This setting grew from the space of image-captioning, which joins machine translation and vision models [**123**]. When correctly formulated, VQA is believed to require the model to understand certain aspects of the scene. However, when poorly formulated these tasks often require large amounts of

domain-specific knowledge that are outside the scope of the task. Datasets like VQA [**44**], DAQUAR [**75**], and COCO QA [**93**] have enabled the development of VQA algorithms. These datasets generally struggle to find the correct level of difficulty, often requiring general world knowledge beyond that contained in the dataset. They can also fail to capture the long tail of possible questions and images which appear in real world scenarios. Datasets like CLEVR [**58**] and Visual Genome [**64**] are other challenging datasets which propose questions that necessitate reasoning about relationships between objects in images. Very recently, two datasets have been proposed for reasoning over images with text: TextVQA [**105**] and Scene Text Visual Question Answering (ST-VQA) [**11**], but neither was designed taking into account the needs of the blind.

One exceptional dataset was created about 10 years ago: VizWiz [**9**]. The team which developed this dataset distributed an application to the BVI community which enabled them to take an image, ask questions about it, and get timely spoken answers from a sighted person. 31,000 of these questions, answers, and images compose the VizWiz dataset and VizWiz grand challenge [**47**], which introduced notable new difficulties for VQA like images of poor qualities, conversational questions, irrelevant questions due to out-of-focus images.

VQA models are usually composed of two networks [**59**]: one that extracts features from the question using natural language processing methods like Bag-Of-Words or Long Short Term Memory encoders [**53**] and another that extracts features from the image using convolutional neural networks pre-trained models on ImageNet [**96**]. These two sets of features may be combined together in different ways (e.g. concatenation, use of Bayesian models to infer the relationship between the different modalities) to produce an answer. Approaches based on attention can also be used and explore the spatial and logical relation among objects presented in the data. More recently, FiLM [**87**] has also been proposed to solve the VQA task. This model differs from classical VQA model because it relies solely on a very simple feature-wise affine conditioning to use question information to influence the behavior of the visual pipeline to answer questions. However, despite those recent advances the current performance of those models are far from human-level on most real-world tasks, making them unreliable for the BVI community.

2.4. Vision and Language Navigation Environments

Some of the primary contributions of this work are RL environments for multi-modal sidewalk navigation. There is extensive literature on navigation tasks that require the use of both vision and language, which we discuss briefly in this section. Additionally, there is work focusing on outdoor navigation. However, neither area addresses the type of multi-modal learning or provides an appropriate setting to investigate the learning of structures required to perform our task of interest.

Vision and language navigation tasks generally require agents to act in the environment to achieve goals specified by natural language. This setting at the intersection of computer vision, natural language processing, and reinforcement learning has generated many tasks and a common framework for evaluation of embodied navigation agents [**1**]. The task defined in "Vision and Language Navigation" [**2**] provides agents with 21,567 human-generated visually-grounded natural language instructions for reaching target locations in the Matterport3D environment [**21**], an environment which consists of 10,800 panoramic views from 90 building-scale scenes connected by undirected spatial graphs. The success of [**2**] has also motivated the development of systems to generate natural language instructions for navigation [**39**]. Many environments focus on navigating apartment interiors, with continuous [**121**] or discrete [**21**] action spaces and varying levels of interactivity [**62**], while relatively few have investigated vision and language tasks in outdoor settings.

Outdoor RL environments often base their work on panoramic imagery and connectivity graphs from Google Street View. A prominent example of such a work on outdoor navigation is Learning to Navigate in Cities without a Map [**77**]. Outdoor environments support tasks that range from navigating to the vicinity of arbitrary coordinates [**77**] and certain types of locations (e.g. gas stations, high schools) [**16**], to following natural language instructions [**52**]. However, the sparsity of the nodes containing images (averaging 10 meters between connected nodes in [**77**]) and the vehicle-mounted perspective imagery makes these environments unsuitable for pedestrian navigation. Furthermore, such environments do not provide the type of labelled information that is necessary to construct pedestrian navigation tasks (e.g., door numbers, street signs, door annotations). Nor is it possible to provide dense annotations in the StreetLearn Google Street View data [**76**] because the resolution of that imagery is too low (1664 x 832 pixels). As outlined in Section 4.1, SEVN provides the higher resolution imagery that is necessary for agents that find and reason about scene text in order to navigate to specified doors.

It is also important to note that the agent in [**77**] did not have access to high resolution visual features. In that work, panoramic RGB images were cropped to 84×84 pixels covering a 60^o field of view then input to a convolutional model, thereby losing much of the semantic information that humans would use to perform the same task. Also, the goal location was quite large, on the order of an entire block. In Chapter 4 we extract information from high-resolution images and train an agent to navigate using these. Additionally, we focus on navigating to very narrowly defined goal locations.

Chapter 3

Grid World Experiments

In this chapter, we construct two grid world environments and show results from experiments using Proximal Policy Optimization (PPO) to train a neural network to perform simple OMN tasks. We analyze both the ability of this model to fit these environments as well as the model's generalization performance. Several experiments focus on training this agent under a curriculum and under domain randomized environments. This work was largely conducted during the Winter semester of 2018 in the McGill Reinforcement Learning course taught by Doina Precup and Pierre Luc-Bacon. The author of this thesis was the sole author of that work, which received the highest possible mark, but was not submitted for peer-review.

Grid worlds have been long used as a simple spatial abstraction because they are easy to create, understand, and computationally inexpensive to experiment on with RL methods. We were unable to find an environment which adequately suited our tasks of interest, thereby motivating the development of this environment. An example of a related but unsuitable environment is the 4-room environment used to demonstrate many algorithms including UVFA [**99**]. In one popular task defined on this environment, the agent must learn to explore different rooms to find a goal object. In UVFA, the task was slightly modified such that the agent was provided a goal vector specifying which room contained the goal. The agent was trained to perform the task in 3 of the 4 rooms as specified by a goal vector, then tasked with generalizing to goals in the fourth room. Because our task of interest contains significantly more structure (e.g. house numbers, goal distributions, distractors), we felt it inappropriate to benchmark on an environment lacking these key elements.

The purpose of the proposed environments is to create a setting which contains realistic structures that enable human navigation, but where current methods of DRL fall short. Street addresses are a very common method of specifying locations. A fully-specified address usually contains a house number and street name, postal code, city name, state name, and country name, however in our simulation we only model local navigation tasks requiring the specification of the street name and house number.

3.1. Grid Street and Grid City Environments

We propose a Grid Street and Grid City environment developed using the gym-minigrid framework [**26**]. Both environments are static POMDP grid worlds. These POMDPs are static both in the sense that the spatial distribution of houses and the numerical value of addresses are constant between successive training episodes, and that there are no moving objects in the environment. We also report experiments in a domain randomized version of Grid Street with changing spatial and numerical distributions, but as we will see fitting the static environments already provides significant challenges. Though these grid worlds contain many of the characteristic elements of the outdoor micro-navigation task, we do not address some challenges including safe street crossing, following sidewalks and footpaths, and traversing courtyards. The more complex environment proposed in Chapter 4 shares many of the same characteristics, but is modelled using real-world data.

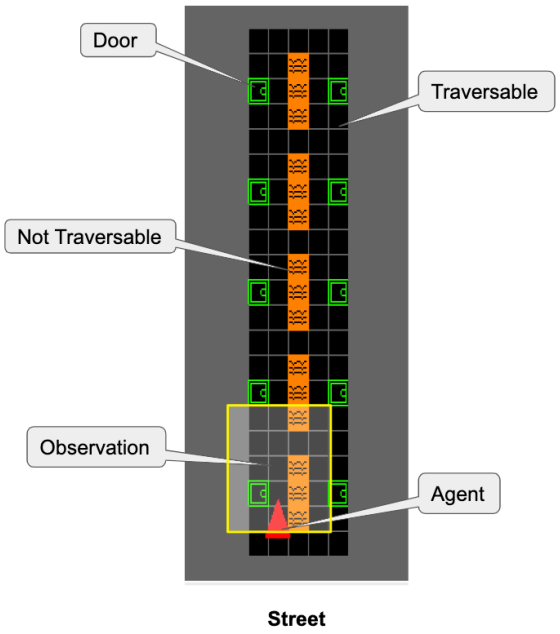


Fig. 3.1. Grid Street. The Grid Street environment has 10 houses that are potential target locations. The agent's observations come as a 5×5 matrix (demarcated by the lighter grey outlined with yellow) at each timestep. The center of the road is lava, and not traversable, meaning that there are no actions from states adjacent to the lava to the lava states.

In Grid Street and Grid City, like the real-world, the agent can only see a small part of the environment at any time. More specifically, while the state-space of the environment S is modeled as a tensor of shape $(X, Y, 3)$, where $X, Y \in \mathbb{N}$, the agent only observes $o \in \Omega$, where the dimension of o is $(5, 5, 3)$. o contains information about objects in a 5×5 grid in front of the agent, where the additional three dimensions represent object type, color, and door number. In this task setting, the first dimension (object type) can only represent three types

of objects: traversable, not traversable, and door. While we have not explicitly modelled occlusions, objects come into and out of view as the agent moves around the environment. Still, this is a much easier task than it would be in the real-world given that the agent has perfect information about the position and identity of objects within its field of view.

The agent's action space is as follows: $A = \{left, right, forward, done\}$. The $left$ and $right$ actions turn the agent by 90°, while the forward action moves the agent 1 tile in the direction they are facing, unless that direction action would transition them off the edge of the environment or onto an untraversable tile. In that case, the action is interpreted as a no-operation, and agent stays in the same location. Upon executing the *done* action, the episode is terminated and a reward is calculated. Here, the rewards are sparse, i.e. the agent only receives positive reward for successfully terminating the trajectory on the correct tile.

At the beginning of each training episode $m \in M$, the agent is provided with a natural language sentence $I_m = w_0, ..., w_l$ as an instruction, where each w_i is a token while the l is the length of the sentence. Here, I_m instructs the agent to navigate to a unique address, for example "go to street 0 house 3". The agent must then move to the unique grid location specified by I_m and select the *done* action. In these environments, the *done* action finishes the trajectory, and if the agent is in the correct tile, returns a positive reward γ^i, where $\gamma = 0.99$ and i is the number of steps the agent took to complete the trajectory. Otherwise the agent receives a zero reward. We experimented with setting the reward to $\gamma^{\frac{i}{k}}$ where i is the number of steps the agent takes and k is the shortest path length, but this did not meaningfully alter the results.

There are many ways that we can modulate the difficulty of these environments. We can start our agent in the same location, or have different starting locations; all experiments start the agent in the same position. We can also provide the agent with a curriculum, asking the agent to navigate to more distant doors as it learns to accomplish easier goals. We will address these notions further in Section 3.3, Experimental Results.

The Grid City shown in Figure 3.2 is an extension of the Grid Street environment that adds street signs at each street corner. Much like a real city, house numbers are even on one side, odd on the other, and ascend (or descend) monotonically as you continue in one direction. For our experiments, we will refer to the Static Grid City when each episode is run on the same environment, and Domain Randomized Grid City when we randomize the location of the house and house numbers at the start of each trajectory. The motivation behind the Domain Randomized Grid City is that if we want our learned policy to work in the real-world, we will want our policy to generalize to different spatial distributions of houses and distributions of house numbers. This technique of domain randomization to improve transfer has a long history in the literature [**110**].

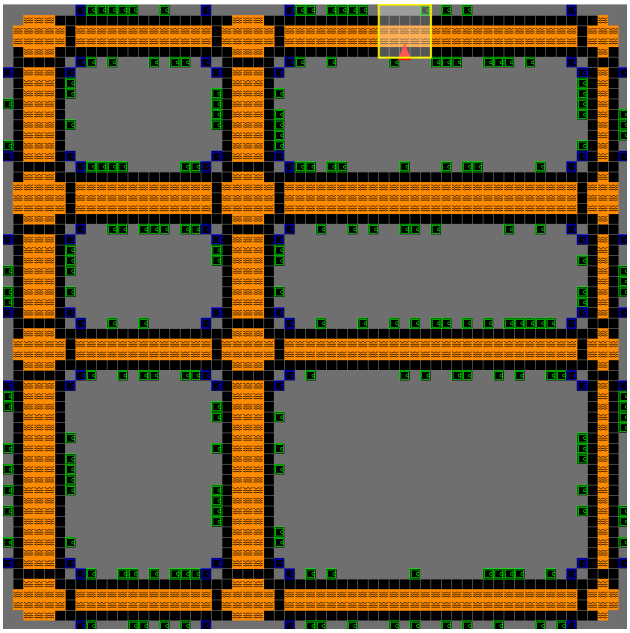


Fig. 3.2. Grid City. Here we see four horizontal streets and three vertical streets in the Grid City environment. We can vary this parameter as well as the spatial distribution of houses and the distribution of their numbers.

3.2. Model and Training

We apply a neural network model similar to that described in [**63**]. The model has both an actor and critic head parameterized by two fully-connected layers with a tanh non-linearity; the critic (v_θ) estimates a value for the state, while the actor (π_θ) outputs a softmax distribution [**10**] over actions. These heads share a trunk f_θ which takes in two inputs at each timestep t: the observation o_t and episodic instruction I. We train this model with PPO, which has achieved state-of-the-art performance across a wide range of challenging tasks and was required little hyper-parameter tuning in our experiments. Though PPO is most commonly known as an algorithm for learning continuous policies, it can also work in discrete settings by simply replacing the Gaussian distribution policy with a softmax distributed one.

We parameterize f_θ as a neural network with a 3-layer convolutional neural network ψ embedding o_t, and a Gated Recurrent Neural Network (GRU) [**27**] ϕ which embeds I. We then concatenate these two equivalently sized embeddings $s = \psi(o_t) \oplus \phi(I)$ and feed s into both the actor and critic networks. The actor network outputs a stochastic softmax distributed policy that is sampled during rollouts $a_t \sim \pi_\theta(s_t)$.

We train this model using Proximal Policy Optimization as described in Section 2.1.2. We initialize the network f_θ and begin doing rollouts of the policy in the environment. The

score function we are maximizing is similar to that in equation 2.1.2:

$$J(\theta) = \mathbb{E}[r(\theta)\hat{A}_{\theta_{old}}(s_t, a_t)]$$

where $\hat{A}$ is the estimated advantage function $A(s_t, a_t) = q_\pi(s_t,a_t) - v_\pi(s_t)$. Advantage can be defined as a way to measure the improvement we can get by taking some action when we are in a particular state $q_\pi(s_t, a_t)$ as compared to the average general action at that state $v(s)$. We weight the advantage of the old policy by the ratio $r(\theta)$ as we have defined in equation 2.1.1:

$$r(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$$

An $r(\theta) > 1$ means that the action a_t is more likely in the current policy than the previous one. Again, following [**101**] we clip the policy updates, requiring that $\theta_{old} - \theta < \epsilon$ with epsilon set to 0.2 in our experiments. This limits the rate of change in the policy to reduce instability during training.

We have not opted to use a recurrent policy network, i.e., this network has no memory. While one may expect performance on navigation tasks should improve with memory resulting from localization and pattern identification, we opted instead to experiment with models operating on stacked frames of experience. This strategy has the benefit of ensuring that relevant information from the environment is available to the policy network, while avoiding the challenges of training recurrent neural networks. During our hyperparameter search, we tried models with framestacks containing 1, 8 and 16 frames.

In the time since this work was conducted, several works have been published at top-tier machine learning venues proposing improvements on policy-based methods for reinforcement learning. Notably [**112**], published at NeurIPS 2019, proposed a method for solving sparse reward tasks using shaped rewards and a modified version of PPO. In contrast, our work forgoes the use of any reward shaping. Instead, we focus exclusively on only the most strict version of this task: a positive reward is received only when the agent has knowingly completed its task (indicated by the *done* action). We set the magnitude of the sparse reward as the ratio of the length of the path taken to the optimal path.

The choice to forego reward shaping here was made for several reasons. First, sparse rewards are more reliable to implement on a physical platform than dense rewards. Works experimenting with real-world training of RL agents include VICE-RAQ [**106**] which learns a reward function on high-dimensional observations. Other methods have been proposed which specify goals using images of the goal, and then train a goal classifier on this data [**122, 114**]. Because we would like to develop methods which may be transferred and fine-tuned in the real-world, we opted to experiment with a sparse reward setting here. Second, it is relatively "easy" to detect if you're in a goal state by reading visible house numbers and matching against the navigation instruction. However, determining if the model is making progress

toward the goal is more difficult. The model needs to determine whether it is on the correct street, and whether the house numbers are ascending or descending towards the target. Only then would the model have some sort of unreliable but dense reward signal. As a result of these challenges, we prefer to experiment here with sparse rewards.

3.3. Experimental Results

We begin by describing our experiments using PPO to explore the Static Grid Street and Grid City environments. We then formalise our curriculum learning approach to these environments as well as investigate the effect of framestacking. We then describe our hyper-parameter search and close with the challenging domain randomized Grid City environment.

Static Grid Street and City: In Figure 3.3, we show the results of training 5 agents using random seeds and PPO with no memory (framestack = 1) for 1 million frames. On the left, we show results on the Static Grid Street environment, and on the right we show results on Static Grid City. The PPO algorithm starts with a random policy that yields near zero returns. This is expected, given that these environments produce sparse episodic rewards. As training progresses its average episodic reward gradually increases to about 0.2. The maximum average return over the trajectories in one update batch was 0.39, while the average reward for acting optimally in this setting is about 0.84.

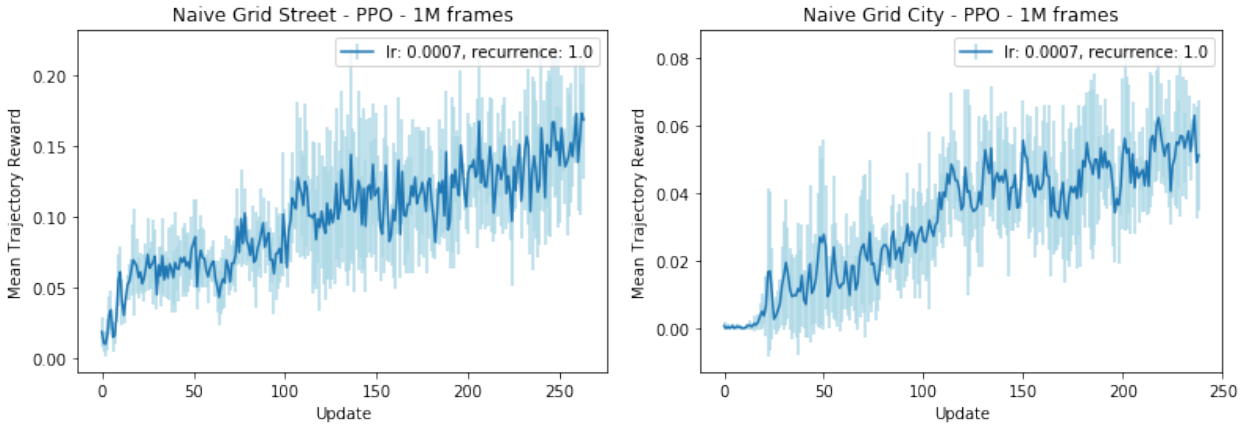


Fig. 3.3. Static Environments. **Left** shows the trajectory rewards during training on the Static Grid Street. **Right** shows the trajectory rewards during training on the Static Grid City. **Both** indicate the mean episodic discounted reward (G_t) with a dark blue line, while the light blue indicates standard deviation. The X-axis indicates the number of parameter updates that have taken place; in our experiments we had 32 processes performing rollouts of a policy in the environment with the maximum number of actions in a trajectory set to 128. Both curves show the results of training our model with PPO in a static environment without a curriculum for 1M frames of experience across 5 trials. As indicated in the legend, the model used a Learning Rate (lr) of $7e^{-4}$ and had no frame stacking (recurrence = 1).

As we can see in Figure 3.3 (Right), the model in the larger environment does not yield strong results. This model had to contend with over 50 goals, with optimal trajectories requiring over a dozen actions. Still, it was still able to achieve about 10% of its optimal reward within 1 million frames of experience. Next, we present a curriculum learning approach, and note that this naive approach to Static Grid City is equivalent to a curriculum setting with maximal difficulty.

Curriculum Learning Approach: To simplify this challenge, we re-frame this OMN navigation task as a curriculum of tasks $C = \langle M_1, \ldots, M_N \rangle$, where N is the number of target addresses. Each task M_n where $0 < n \leq N$ requires the agent to navigate to a different location specified by an instruction I. We propose a partial ordering on the elements of C: ascending by the length of the shortest traversable path from the starting location to the goal. We also propose a set of training tasks, $T = \{M_1, \ldots, M_k\}$ where $1 \leq k \leq N$, which starts out containing only M_1. We sample training tasks uniformly $t \sim \mathcal{U}(M_1, M_k)$.

When we train our agent, we sample training tasks and perform rollouts of our current policy π_θ in parallel across 32 processes until we have accrued 4096 frames of experience (128 frames $\times$ 32 processes) then update parameters as described in the previous section. We continue this process until the mean success rate is greater than 95% across all episodes in a parameter update batch. At this point, we add the next task in C to T. Figure 3.4 shows the results of the curriculum learned model on the Static Grid City environment. We see that the model is able to quickly achieve a very high mean trajectory reward while learning to navigate to new goals.

Hyper Parameter Search: We ran a hyper-parameter search over the Static Grid Street and Static Grid City environments. We ran 20 experiments with 3 trials each out to 1,000,000 frames, searching over two parameters: the learning rate was sampled from a log-uniform distribution between 10^{-5} and $10^{-1.5}$ while the number of frames to stack (equivalently referred to as recurrence) took on the values of 1, 8, and 16. The results of this search are shown in Appendix C.

It appears that the best learning rates were on the order of 10^{-3} and that frame-stacking did not have much effect. Perhaps a stronger prior or auxiliary tasks that encourages the model to use house numbers as location features would differentiate frame-stacking approaches. In Figure 3.5 (right), we provide additional evidence for this conclusion through another experiment. In it, we keep the learning rate constant while varying the number of frames in the stack with no appreciable change in performance.

Long-Running Curriculum Experiment: We were surprised to find the extent to which PPO struggled to learn the structure of the static grid environments. One cause may have been that the preferred actions in the starting region (going right/left only) along the starting street differ significantly from the ones that are needed in the later regions (going up/down only). Figure 3.5(left) hints at this possible issue. In particular, the first 7 levels

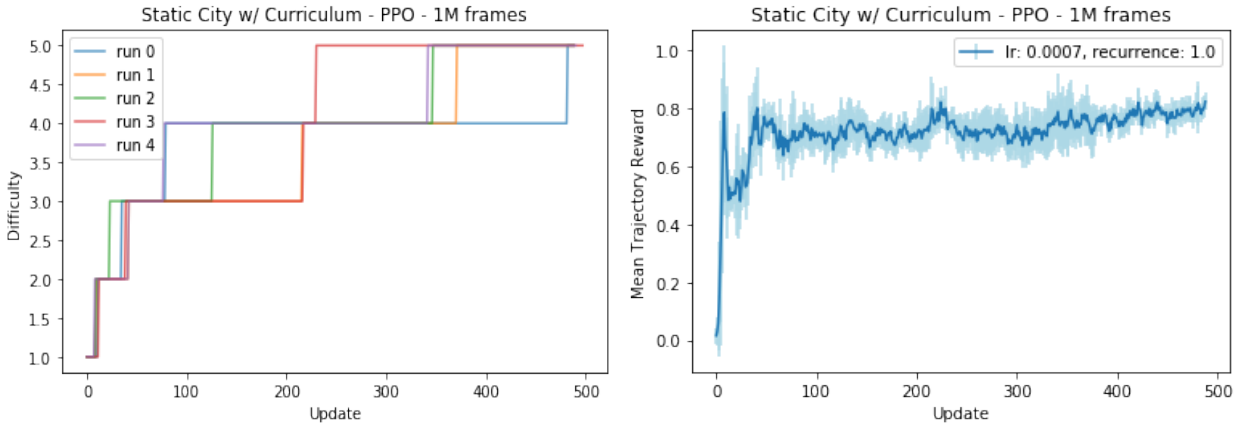


Fig. 3.4. Static Environments with a Curriculum. **Left** shows the level of difficulty of the curriculum during training on the Static Grid City for 5 runs. The difficulty level increases when the success rate is above 95%, which happens quite often for nearby houses and less frequently as the task grows more challenging. **Right** shows the trajectory rewards during training on the Static Grid City under this curriculum. **Both** show the results of training our model with PPO for 1 million frames of experience on the Static Grid City. The X-axis indicates that nearly twice the number of parameter updates have taken place as compared to the previous experiment using the same number of frames of experience. This is because the episodes are shorter on average, and we update parameters after all 32 processes have performed a rollout.

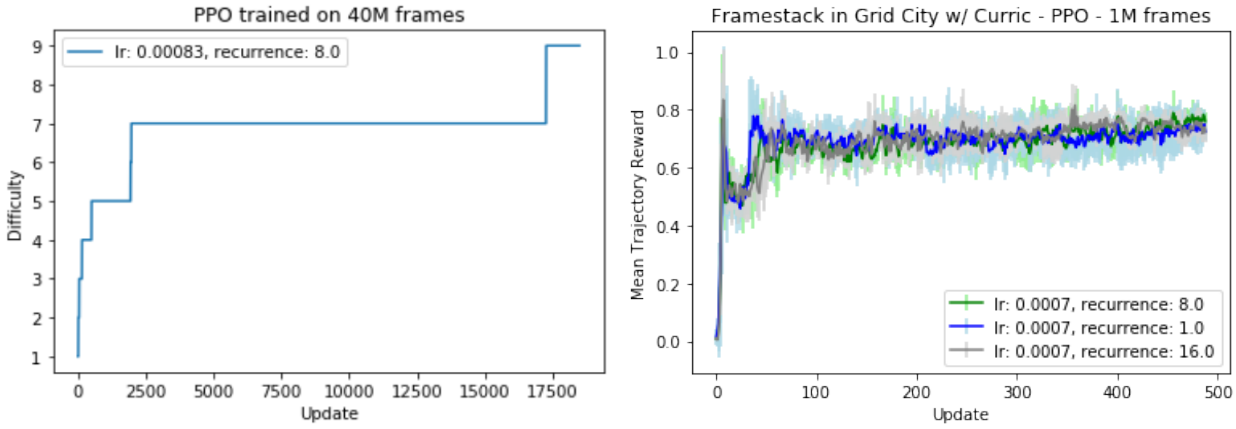


Fig. 3.5. Long-Running Curriculum Experiment and Framestacking. **Left** shows the result of training a single agent on the Curriculum Static Grid City for 40 million frames of experience. As before, the difficulty level increases when the success rate is above 95%. Given the long-running nature of this model, we opted for a lower learning rate in the hope that it would yield more stability. **Right** shows the trajectory rewards during training on the Curriculum Static Grid City for models with varying amount of recurrence. The plot suggests that memory is not a factor limiting our models performance.

involve navigating to goals which are placed along a horizontal line, while the next levels require the agent to cross an intersection and begin going vertically; i.e., the agent starts

in the middle of a block with doors to their left and right, but at difficulty 7 must learn to round a corner onto a new street.

Another explanation is that PPO is an on-policy method being applied to a sparse reward setting. While it is clear from the curriculum learning experiments that a random policy can sometimes achieve the goal, it is possible that some task or tasks at difficulty 7 are much more difficult to achieve. It's possible that a method equipped with an experience replay buffer may make better use of rare successful trajectories. Indubitably, models with access to shaped rewards (increasing as it approached the target) would perform much better.

Domain Randomized Environments: In the Domain Randomized Grid City environment we modify both the spatial distribution of houses and the address ranges for each training episode. This domain randomization makes the task much more difficult to learn, but should also makes the algorithm more robust to distributional shift. As seen in Figure 3.6, the agent struggles to learn navigate to even the nearest goal. In contrast with the static curriculum environment where the agent was able to achieve a success rate above 95% when navigating to 5 different goals after 1 million frames of training, in this domain randomized environment the agent never achieves a success rate above the low 80%'s, and thereby never moves on to the next level of difficulty.

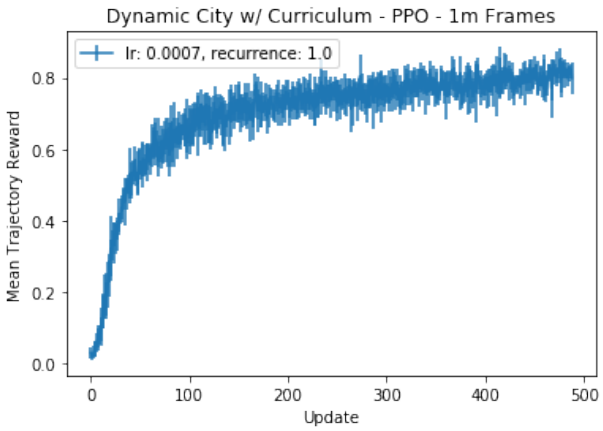


Fig. 3.6. Domain Randomized Grid City. In the domain randomized setting our agent never achieved a 95% success rate even on difficulty 1. It was much more difficult for the agent to learn the structure of the environment under this condition. To solve such an environment, we may require additional model biases and structure, conditioning information, or training procedures to solve such an environment.

There are many possible explanations for this method's poor performance under a domain randomized setting. Most simply, the state space we are optimizing in becomes much larger and may require more time to optimize as a result. As we will explore further in the next section, the model appears to be learning to map its target to a specific navigation policy without learning general concepts. While this type of learning can still lead to useful behavior,

its limitations can be shown when testing on settings outside of the training distribution. We will explore this concept further in the following chapter.

Another explanation for this result may be that because PPO is an on-policy RL algorithm reward sparsity becomes a more important problem in this setting. Under a random policy goals are rarely reached in any of these environments, and even when a goal is successfully completed the policy will likely undergo many parameter updates before the same goal and environment are sampled again. This suggests that a modified domain randomization algorithm and curriculum may improve performance. For example, an alternative training scheme would be to begin with a single environment and goal, then once that policy achieves a high success rate to add an additional task in a domain randomized environment.

3.4. Conclusion

In this chapter we described two grid world environments, proposed a neural network model, trained it using PPO, and analyzed the results. We found that this method was unable to fit the Static Grid City or Static Grid Street environments after 1 million steps of training. After proposing and running experiments using a curriculum learning approach, our model learned policies capable of navigating with a greater than 95% success rate to 5 different goals after 1 million frames of experience (500 policy updates), and up to 9 goals after 40 million frames of experience. Domain randomized environments were surprisingly challenging, and this model was unable to learn to consistently complete even 1 goal.

The Grid Street and Grid City environments contain many of the challenges present in the real outdoor micro-navigation setting. The environment addresses goals via natural language, organizes house numbers and streets in a plausible way (odd numbers on one side, even numbers on the other, monotonic change along a street), provides partial observations of the environment, and contains a non-uniform distribution of house locations. However, several important aspects of the OMN setting are not covered, for example dynamic obstacles such as pedestrians and motor vehicles, or noise in the observation. Indeed, the problem of accurately converting a sequence of images into the gridworld representation presented here is a difficult one. In the next chapter, we will turn to the utilization of real-world image data to populate an environment and train a navigation model using images, text and simulate GPS information.

In the next chapter, we develop a simulator which uses real-world data to further our investigation of OMN tasks. We also train models with shaped rewards and propose a suite of additional tasks. Additionally, we evaluate our models capacity to generalize outside of its training environment and consider ways to improve this performance as future work.

Chapter 4

A Sidewalk Environment for Visual Navigation

This chapter largely consists of a reproduction of [**118**]:

Martin Weiss, Simon Chamorro, Roger Girgis, Margaux Luck, Samira E. Kahou, Joseph P. Cohen, Derek Nowrouzezahrai, Doina Precup, Florian Golemo, and Chris Pal. "Navigation agents for the visually impaired: A sidewalk simulator and experiments." In *Conference on Robot Learning*, pp. 1314-1327. 2020.

The author of this thesis was the primary author of the paper above. In this chapter, we introduce a novel simulated environment for sidewalk navigation "SEVN" (read "seven"), that supports the development of assistive technology for BVI pedestrian navigation. SEVN contains 4,988 high resolution panoramas with 3,259 labels on house numbers, doors, and street signs (see Figure 4.2 for an example). These panoramas are organized as an undirected graph where each node contains geo-registered spatial coordinates and a 360° image.

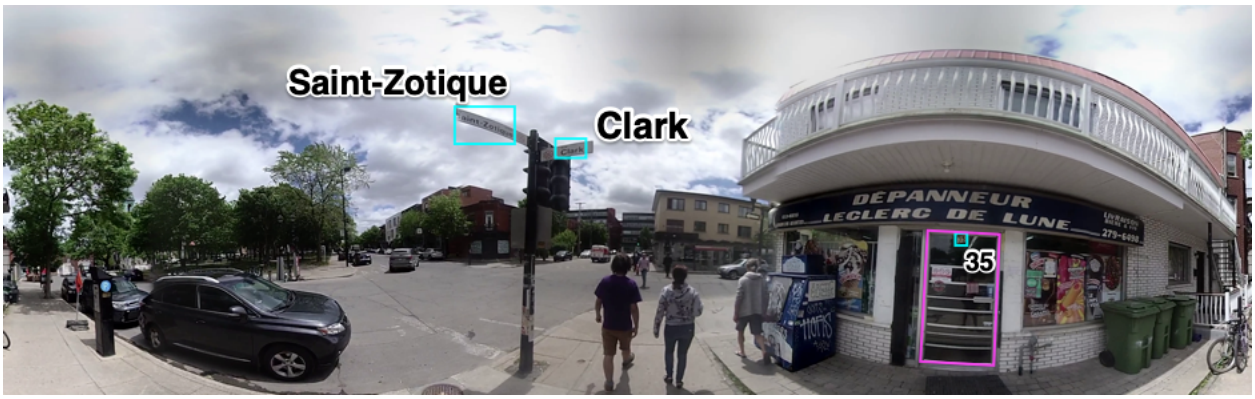


Fig. 4.1. Example stitched equirectangular image. This image was shot on a Vuze+ camera with 8 synchronized Sony FHD image sensors each with a F/2.4 fish-eye lens. The image was stitched using the Vuze VR Studio stitching algorithm, and then hand-annotated with door, house number, and street sign detections and transcriptions.

We define several OMN tasks that may be completed using multi-modal observations in the form of variable resolution images, extracted visible text, and simulated GPS data. Our experiments focus on learning navigation policies that assume access to ground truth text

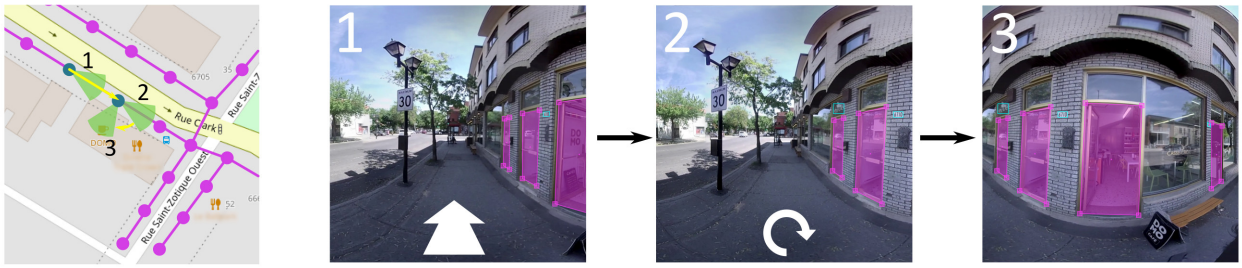


Fig. 4.2. Illustrated example of an agent trajectory. Left) An overhead view of the trajectory. 1) The agent starts on the sidewalk with the goal door in view but out of reach; the agent moves forward. 2) The goal door is beside the agent; the agent turns right. 3) The correct door is visible and fully contained within the frame; the task is complete.

labels, and in this setting our multi-modal fusion model demonstrates strong performance on a street segment navigation task. We hope that the release of this chapter, dataset, and code-base will spark further development of agents that can support members of the BVI community with outdoor navigation[1].

The primary contributions of this chapter are:

- A benchmark dataset containing panoramic images from six kilometers of sidewalks in Little Italy, Montreal, with annotated house numbers, street signs, and doors[2].
- An OpenAI Gym-compatible environment [**17**] for RL agents with multi-resolution real-world imagery, visible text, simulated GPS, and several task settings[3].
- A novel neural architecture for RL trained with PPO [**101**] to fuse images, GPS, and scene text for navigation, with results and ablations [4].

4.1. Overview

This section introduces SEVN, a visually realistic Deep Reinforcement Learning simulator for pedestrian navigation in a typical urban setting. We first describe the process by which we captured and annotated the data in SEVN before discussing the simulator's interface.

The Data were first captured as 360° video using a Vuze+ camera attached to the top of a monopod held slightly above the operator's head. The Vuze+ has four synchronized stereo cameras. Each stereo camera is composed of two image sensors with fisheye lenses that each capture full high definition video (1920x1080) at 30 Frames Per Second. We used the VuzeVR Studio software to stitch together the raw footage from each camera to obtain a 360° stabilized video, from which we extracted 3840 × 1920 pixel equirectangular projections. We then crop and remove the top and bottom sixth of these panoramas resulting in a 3840 × 1280 image which contains about 3.5 times as many pixels as those in StreetLearn [**76**].

[1] `https://mweiss17.github.io/SEVN` [2] `https://github.com/mweiss17/SEVN-data` [3] `https://github.com/mweiss17/SEVN` [4] `https://github.com/mweiss17/SEVN-model`

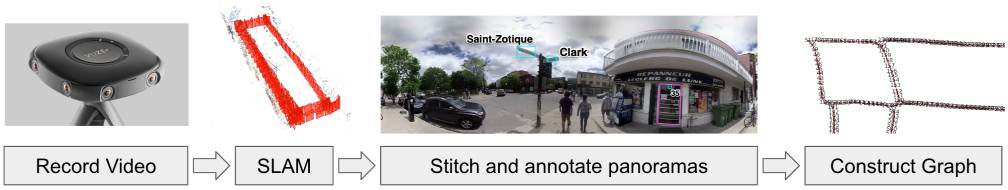


Fig. 4.3. Data processing pipeline. We captured footage using a Vuze+ array of 4 stereo cameras. We then localized this footage with ORB-SLAM2 [**82**] yielding a spatial graph of images and positions. We sparsified the graph, stitched the remaining images into 360° panoramas using VuzeVR Studio, and hand-annotated them.

We used ORB-SLAM2 [**82**], a Simultaneous Localization and Mapping (SLAM) pipeline, to geo-localize the footage. As input to ORB-SLAM2, we provided an undistorted view of the left front facing camera from the Vuze+ and obtained a camera pose for each localized frame (3-D coordinate and quaternion). From these camera poses, we created a 2-D graph with node connectivity determined by spatial location. The recording framerate (30 Hz) resulted in very little distance between nodes - so we sparsified the graph to 1 node per meter. Finally, we manually curated the connectivity of the graph and location of the nodes. Figure 4.3 summarizes the data processing pipeline.

Our dataset is split into a Large and Small dataset, where Small is a block containing several urban zones (e.g. residential, commercial) bounded by St. Laurent, St. Zotique, Baubien, and Clark. Montreal's municipal zoning regulations are described in supplementary section D.2. Figure 4.4 shows the full graph with these splits overlaid on a map. Each split contains street segments and intersections. As part of this work, we released the code to capture, SLAM, and filter the data[5].

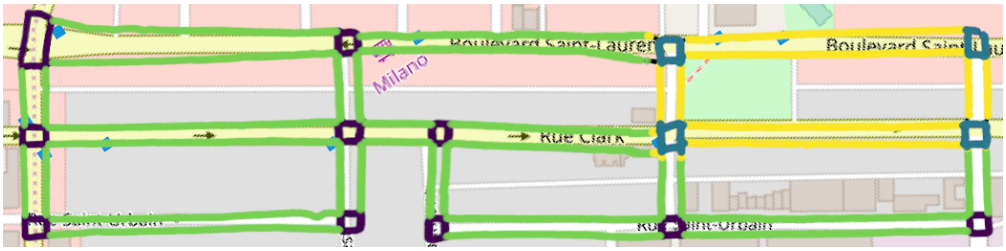


Fig. 4.4. SEVN Spatial Graph super-imposed on an OpenStreetMap Carto (Standard) view of Little Italy, Montreal. The dataset is split into two sets: Large and Small. The Large dataset street segments shown in green and the intersections are shown as purple. Street segments in the Small dataset are shown in yellow and the intersections are shown in teal. Visualizations of the goal locations and municipal zoning are contained in the supplemental material, sections D.2 and D.2.

The Annotations we provide on the full-resolution panoramas are bounding boxes around all visible street name signs and house numbers, and include ground truth text

[5] `https://github.com/mweiss17/SEVN-data`

annotations for both. Doors with clearly associated house numbers are annotated with polygons. To create the annotations, we used the "RectLabel" software. The annotated panoramas are publicly available with blurred faces and licence plates[6]. These annotations and privacy blurs were added and verified by two authors. Table 4.1 shows some global statistics about the dataset, the splits, and the annotations.

Name	# Images	# Streets	Length	# Doors	# Signs	# House Numbers
Large	3,831	8	5,000	1,017(355)	167 (11)	1,307 (367)
Small	1,157	4	1,300	286 (116)	46 (3)	436 (119)
Total	4,988	8	6,300	1,303 (470)	213 (11)	1,743 (485)

Table 4.1. Dataset statistics and number of annotations. We report the total amount of house number, door, and street name sign annotations with the corresponding number of unique physical objects that were annotated in parentheses. Inside the parentheses of the "# Signs" column, we report the unique number of street names visible within the dataset, not the unique number of street name signs. The "# Streets" column shows the number of unique streets where we captured data. The length column represents the length in meters of the region where data was captured.

The SEVN Simulator is based on the OpenAI Gym [**17**] environment. We chose a similar action space to Learning to Navigate in Cities without a Map [**77**] with slight left and right turns (±22.5°), sharp left and right turns (±67.5°), and a forward action that transitions to the neighboring node nearest the agent's current heading. If there is no node within ± 45° of the agent's current heading, then we do not change the agent's position. We also propose two other actions which are used in a subset of the tasks proposed in Section 4.2: read and done. The read action is only used in the CostlyTxt task where the agent incurs a small negative reward in order to access the scene text; the done action is only used in the Sparse task to terminate the trajectory. The agent observes a 140° normalized image cropped from a low-resolution (224×84 px) version of the panorama, created during a pre-processing step from the high-resolution panoramas (3840×1280 px). At this low resolution, most text becomes illegible. Therefore, at each timestep we check if any text which was labelled in the full scale panorama is fully contained in the agent's Field of View (FOV); we encode these labels; and we pass them as observations to the agent (see Section 4.3). An instance of the simulator running with low-resolution imagery can be run at 400-800 FPS on a machine with 2 CPU cores and 2 GB of RAM.

The Oracle was implemented to determine the shortest navigable trajectory between any two poses. A pose is defined as the combination of an agent heading (discritized into 22.5° wedges) and a position (restricted to the set of localized panoramas). Our panorama graph is implemented in NetworkX [**48**], which provides a function to find the shortest path

[6] `https://github.com/mweiss17/SEVN#dataset`

between two nodes (n_1, n_2) in graph G, i.e., $nx.shortest\_path(G, source = n_1, target = n_2)$. For each node we calculated the most efficient way to turn in order to face the next node before emitting a 'forward' action. Once the agent is located in the goal node, we check the direction of the target door and turn until it is entirely contained within the agent's FOV.

4.2. Reinforcement Learning, Rewards & Tasks

In this work we are interested in exploring the use of Reinforcement Learning (RL) techniques for learning agents that have the goal of assisting with navigation tasks. There are many navigation policies which can be learned to assist the BVI, each with its own strengths and weaknesses. This section proposes several reward structures and tasks for learning agents within an RL framework that encourage agents to learn different navigation policies. These are summarized in Table 4.2. **Dense rewards** are provided to the agent at each time-step. We provide a $+1$ reward for each transition taking the agent closer to the goal, and a -1 for each transition that takes the agent further from the goal. For turn actions, we provide a $-.2$ for turns away from the direction which enables a correct transition, and we provide a $+.1$ reward for each turn towards this direction, unless the agent has already inhabited a pose nearer the correct direction in which case the action returns a 0 reward. Within a goal node the correct direction is a heading which fully contains the target door polygon.

Costly Read is a setting where a small negative reward, $-.2$, is provided to the agent after taking the "read" action, returning the labelled text within the FOV. This reward would encourage the agent to only take this action when it provides improved localization. In the context of an expensive scene text recognition model, a policy trained with this reward could be more computationally efficient.

Multi-Goal reward is a setting wherein the agent receives a positive reward, $+1$, for each house number it "sees" during an episode. In this setup, the agent is encouraged to navigate towards regions with many visible house numbers. This behaviour could be useful in assistive navigation systems, helping the BVI person to find house numbers to augment their own navigation ability.

Sparse rewards not only require the agent to complete the entire task before receiving a reward, but also require the agent to emit a "done" action with the target door fully in view. This task is quite challenging, but successful models should generalize better than those trained with dense rewards.

In tasks 1-7, we require that the agent occupy the node closest to the goal door. We identified the nearest panorama to a goal address through a proxy metric: the polygon with the largest door area for a given house number. The success condition also requires the target door to be entirely contained in the agent's field of view. For all tasks, we first select a valid terminal state by uniformly sampling over doors with addresses. Next, we identify the

street segment which contains this address and uniformly sample the agent's start node and direction from this segment.

Tasks 1 to 4 represent an ablation study wherein the agent is trained with different combinations of sensor modalities to determine their relative contributions in a dense reward setting. This task investigates the small-scale sidewalk navigation problem, with trajectories terminating when the agent has navigated to the goal node and turned so that the goal door is fully within view.

Task 5 does not restrict the start and end poses to the same street segment. Instead, the agent must navigate through an intersection to find a goal on another segment.

Tasks 6, 7, and 8 are equivalent to task 1, but that the agent is trained with the costly read reward structure, the sparse reward and a multi-goal reward, respectively.

		Observations			Rewards			
ID	**Task Name**	**Img**	**GPS**	**Txt**	**Dense**	**Costly Read**	**Sparse**	**Multi-Goal**
1	AllObs	✓	✓	✓	✓	·	·	·
2	NoImg	·	✓	✓	✓	·	·	·
3	NoGPS	✓	·	✓	✓	·	·	·
4	ImgOnly	✓	·	·	✓	·	·	·
5	Intersection	✓	✓	✓	✓	·	·	·
6	CostlyTxt	✓	✓	✓	✓	✓	·	·
7	Sparse	✓	✓	✓	·	·	✓	·
8	Explorer	✓	✓	✓	·	·	·	✓

Table 4.2. Rewards & Tasks. The first four tasks examine combinations of observation modalities. In the CostlyTxt task, the agent has access to a "read" action which yields the scene text, but imposes a small negative reward. The Intersection tasks requires the agent to cross static intersections to find goals on other street segments. The Explorer task gives a reward for each unique house number the agent sees. The Sparse task, which is most challenging, only gives the agent a reward once it reaches its target destination and emits a "done" action to terminate the episode.

4.3. Observation Modalities

The agent has access to three types of observation modality and a reward signal. An observation can contain an image, GPS, and visible text. Table 4.3 summarizes the available modalities and formats.

The image is a forward-facing RGB image of shape (3, 84, 84) that contains a 135° FOV which depends on the agent's direction. Note that all tasks can also be run with high resolution images of shape (3, 1280, 1280). The simulated GPS is a 2-dimensional ego-centric vector indicating the relative x and y offset from the goal in meters. Finally, coordinates are scaled to the range $[-1,1]$.

We study two kinds of incidental scene text: house numbers and street name signs. At each timestep, we determine whether there are any house number or street name sign bounding boxes within the FOV of the agent. If so, we pass them to the agent as encoded observations. We encode the house numbers as four one-hot vectors of length 10 (since the longest house numbers in our dataset contain 4 digits) flattened into a unique one hot vector of size (40, 1) representing integers between 0-9999. Next, we stack up to three of these vectors to create a vector of shape (120, 1) before passing them to the agent. This enables the agent to see up to three house numbers simultaneously. Street name signs are similarly encoded, but span the 11 street names contained in the dataset.

Observation Type	**Dimensions**	**Description**
Image (low-res)	(3, 84, 84)	Low-resolution RGB image observation with 135° FOV
Image (high-res)	(3, 1280, 1280)	High-resolution RGB image observation with 135° FOV
GPS Coordinates	(4, 1)	Absolute and relative goal coordinates
House Number	(40, 1)	One-hot encoding representing integers between 0-9999
Street Name	(11, 1)	One-hot encoding of street name signs

Table 4.3. Observation modalities and formats. The simulator provides three observation modalities: images, GPS, and text. Images can be provided in low resolution (84×84 px) or high resolution (1280×1280 px). The absolute coordinates of the goal are fed in as two floating point values (x, y) scaled between -1 and 1, while the relative coordinates are computed by taking the difference between the agent's current position and the goal position. Each character in a house number is represented by a $(10, 1)$ one-hot vector; we concatenate four of these together to represent the range of house numbers found in our dataset.

4.4. Method & Architecture

In this section, we describe a multi-modal fusion model that is at its core a CNN trained with PPO [**101**]. To train our model, we chose the PPO algorithm for its reliability without hyperparameter search[7]. For our policy network (see Figure 4.5), we take each modality in its one-hot format (as described in Table 4.3) and tile it into a matrix of size $(1, w, w)$, where w is the width of the image. We then add any necessary padding. Performing this process for each modality and appending all matrices to the input image yields a tensor of dimension $(8, w, w)$. If a model does not have access to a modality, then that matrix (or matrices) are simply filled with zeros. This representation is fed through three convolutional layers with kernel sizes 32, 64, 32, followed by a dense layer of size 256, all with ReLU nonlinearities. This network outputs a probability vector of dimension $(1, y)$, with y being the dimension of the action space. To determine the agent's next action, we sample from this distribution. The critic network contains an additional dense network that converts the combined embeddings

[7] `https://github.com/mweiss17/SEVN-model`

into a single value. The PPO hyperparameters are mostly identical with the recommended parameters for training in Atari gym environments [**17**]; the only difference being that we increase the learning rate (from 2.5e−4 to 3e−4) and the training duration (from 1e6 steps to 2e6). The full set of parameters can be found in the supplementary section D.1.

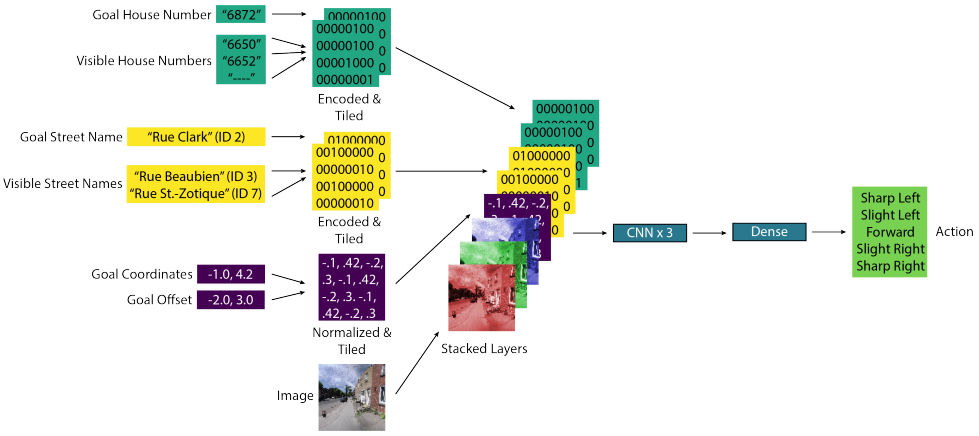


Fig. 4.5. Policy Network Architecture. We show the different input modalities in both their human-readable format and their tiled format. The tiled format is then appended to the RGB image matrix creating an $(8, w, w)$ tensor, where w is the width of the square input image. This tensor is then processed by 3 convolutional layers before being flattened and processed by a dense layer. Finally, this dense layer outputs a vector over the agent's action space, from which we sample the agent's action at that step.

4.5. Experimental Results

This section discusses the performance of our model which navigates to target doors with a mean 74.9% success rate in the sidewalk navigation task. We also ablate this model to investigate the contribution of each observation modality. See section 4.5.2 for ablations with noisy GPS.

4.5.1. Sidewalk Navigation Experiment

Table 4.4 reports the mean and standard deviation of policies after 2M steps of experience on the Small dataset. Results are averaged over 10 seeds. We also report the oracle and random walk performance for comparison. In this setting, the agent has 253 timesteps to navigate to its target location, the length of the longest optimal trajectory in the environment. A random agent successfully completed this task within 253 actions in 5.7% of episodes, while the oracle completed 100 % of tasks in 80.8 steps on average. Two seeds were removed because our analysis indicated that they learned a degenerate solution, exploiting the environment.

As expected, the AllObs model which fuses image data, visible text, and GPS achieved the highest rate of successful navigations. After 2M frames of experience, this model converged to a policy that can navigate to the 116 goals in Small with success achieved in 74.9% of

ID	Task Name	Success Rate	Reward	Trajectory Length
1	AllObs	74.9% (± 7.8%)	43.9 (± 65.9)	113.7 (± 89.7)
2	NoImg	0.8% (± 0.3%)	-24.6 (± 2.3)	251.0 (± 22.0)
3	NoGPS	58.2% (± 3.2%)	32.8 (± 78.3)	147.6 (± 96.6)
4	ImgOnly	57.%(± 2.8%)	32.5 (± 79.8)	149.2 (± 96.6)
-	Oracle	100% (± 0.0%)	58.1 (± 14.2)	80.8 (± 25.1)
-	Random	5.7% (± 2.6%)	-43.2 (± 4.9)	248.9 (± 15.1)

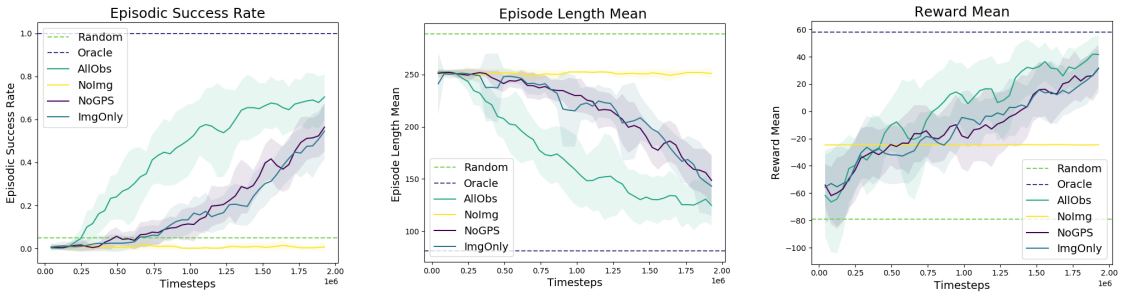
Table 4.4. PPO baseline performance on the Small dataset for the door finding tasks. The maximum trajectory length was set to 253, the number of actions required by the oracle to achieve the longest task. All metrics from training 10 seeds (except for AllObs where we removed two outlier policies) for 2M frames, then averaging 1000 test episodes per seed. The standard deviation reported here is across seeds and evaluation episodes. The two bottom lines of the table corresponds to the mean optimal path length (as calculated by the oracle) and random mean reward for comparison.

trajectories, with the best performing model achieving an 85% success rate. The performance of the AllObs model also improved much more quickly than any other model. However, some trajectories still failed to achieve the goal resulting in a mean trajectory length nearly 40% higher than the oracle. Upon inspection of the trained policies, we observed that the agent sometimes tried to make a forward action when blocked by a wall or street. The agent also sometimes repeated the same incorrect action, turning right and left. Still, the agent often exhibited "intelligent" behaviour, turning to follow the sidewalk and looking for house numbers.

Every model with access to image data had above 55% success rate on average, whereas the model without access to images (NoImg) learned a degenerate policy that performed even worse than a random walk. The poor performance of the NoImg model indicates the importance of even down-scaled 84×84 pixel images for sidewalk navigation. Judging from the moderately reduced performance of ImgOnly and NoGPS, the AllObs model seems not to rely solely on GPS or image data, but can use a combination of all input modalities to achieve superior performance. The next best performing agent, ImgOnly, performed much better, completing the task in more than half of episodes. Adding the visible text modality (i.e., NoGPS model) seems to further improve performance when compared to the ImgOnly model. Figure 4.6 shows the PPO training metrics for task 1 to 4 on Small.

4.5.2. Noisy GPS Experiment

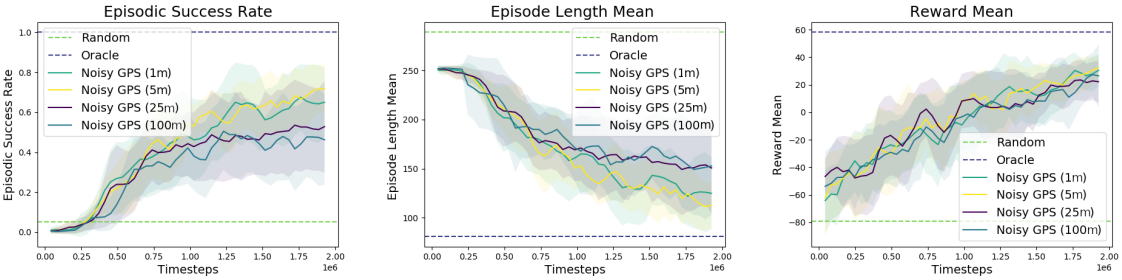
Interested by the effect of noisy GPS sensors on navigation performance, we created a setting where the agent had access to all types of observation modality, but we could add varying levels of Gaussian noise to the GPS sensor. To construct this setting, on the first timestep when we calculate the goal position by sampling the coordinates of a panorama



(a) Success Rate **(b)** Trajectory length **(c)** Trajectory rewards

Fig. 4.6. Training curves over 2M frames of experience in the Small dataset. We report the mean and standard deviation over 10 seeds of several metrics smoothed with a moving average on 10 episodes. The oracle and random walk performances are also provided for comparison. (**a**) shows the proportion of episodes which terminate at the target door. (**b**) shows the length of an episode, which terminates after 253 actions or finding the target door. (**c**) shows the mean dense reward as described in subsection 4.2.

which satisfies the goal conditions, we also sample a Gaussian noise ($\mu = 0, \sigma = x$) where x is the amount of noise in meters. Every time we compute a GPS-based goal offset, we take the agent's current position and add Gaussian Noise ($\mu = 0, \sigma = x$) then subtract the noisy goal coordinates. Results shown in Figure 4.7.



(a) Success Rate **(b)** Trajectory length **(c)** Trajectory rewards

Fig. 4.7. GPS Ablation Study. **All**: We report the mean and standard deviation over 10 seeds of several metrics smoothed with a moving average on 10 training episodes – similar to that in Figure 4.6. (**a**) shows the proportion of episodes which terminate at the target door. (**b**) shows the length of an episode, which terminates after 253 actions or finding the target door. (**c**) shows the mean total trajectory reward.

These results, on the street segment task, indicate that GPS readings with large amounts of noise (25, 100) unsurprisingly lead to worse performance and higher variance in our model. This result is consistent with Figure 4.6, and provides evidence that a model with a highly noisy GPS sensor performs as well as a model trained without a GPS sensor (NoGPS). Smaller amounts of noise seem to have little impact on performance, and possibly even increase

performance. However, the instability of DRL algorithms can make it difficult to recognize small changes to performance.

4.5.3. Generalization Experiment

We hoped that our policy trained to perform navigation to goal doors would generalize to other regions. This, sadly, does not appear to be the case. However, the trained policies are still valuable within the training region. Further, we believe that given sufficient data these policies may generalize. Also, the model proposed in section 4.5 was not specifically designed for generalization. In table 4.5 we present results of the fusion model with all modalities evaluated outside of the training data.

Evaluation	Trained on Large	Trained on Small
Success Rate on Large	21.1% (± 3.7%)	1.0% (± 0.3%)
Success Rate on Small	1.0% (± 0.4%)	74.9% (± 7.8%)
Rewards on Large	13.3 (± 70.4)	-134.7 (± 110.9)
Rewards on Small	-71.1 (± 92.9)	43.9 (± 65.9)
Episode Length on Large	213.5 (± 78.9)	250.5 (± 24.5)
Episode Length on Small	250.6 (± -24.2)	113.7 (± 89.7)

Table 4.5. Generalization Results: This is an extension of table 4.4, with the fusion model trained on the Small dataset remaining the same. We also evaluated this model on the Large set (all data except Small). We also evaluated the same model trained for 20M frames of experience on Large.

We can see that the results of the model trained on the Small dataset and evaluated on the Small dataset performed well, with the same 74.9% success rate seen in table 4.4. When evaluated on the rest of the dataset, this model failed with a success rate of only 1.4%. Similarly poor performances are seen for the model trained on the (much larger) Large dataset with 20M frames of experience.

4.6. Discussion and Future Work

The SEVN dataset and simulator is an RL environment for sidewalk navigation. The codebase and dataset are open and extensible, providing a framework for future research into BVI navigation. As with most machine learning models, deploying RL models trained in this environment into the real-world presents significant challenges. Techniques like domain randomization[**110**] can be used to improve model generalization, but our use of real-world imagery presents a challenge for this approach. One alternative to improve performance would be to gather additional imagery and annotations with better hardware and more sophisticated methods. This could result in higher resolution, more precisely located images, with fewer stitching artifacts common in panoramic images. Alternative methods for embedding scene text as input for the policy model may also yield improved performance.

Besides these approaches, there are several more ways we can imagine to modify and improve this model. Curiosity driven approaches to RL have been explored since at least the early 1990's [**100**], and there is ongoing research using intrinsic rewards to improve the performance of PPO in sparse reward settings [**18, 23**]. Task-specific rewards may be difficult or undesirable to define in high-dimensional settings, which motivated the development of the Random Network Distillation reward in [**18**]. In that work, PPO was modified to have two additional neural network heads: a fixed and randomly initialized target head $f : \mathcal{O} \to \mathbf{R}^k$ and a predictor head $\hat{f} : \mathcal{O} \to \mathbf{R}^k$, where $\mathcal{O}$ is a high dimensional observation. Here, $\hat{f}$ is trained to mimic the output of the randomly initialized network by minimizing MSE $\left\|\hat{f}(x, \theta) - f(x)\right\|^2$ with respect to the parameters of $\hat{f}$ and where x is an input observation. However, in our case with access to low-dimensional inputs in the form of house numbers, it is possible to define an alternative observation prediction task.

A simple option would be to add an auxiliary head f that predicts the observable house numbers in the next observation hn_{t+1} given the current observation x_t, i.e., $f(x_t) \to hn_{t+1}$, where x is the tiled multimodal input shown in Figure 4.5. This, however, would likely be a challenging problem given that there are many thousands of possible house numbers, that there can be several house numbers per observation (or none), and that scene text detection and recognition in the wild remains a very challenging area [**91**]. Supporting this, in 2013 Goodfellow trained CNNs to predict multi-digit house numbers on an internal Google dataset with tens of millions of labelled examples [**43**]. Because these images did not contain bounding boxes, the resulting accuracy of this method was only 91%, much lower than the performance of methods on the publicly available cropped StreetView House Number [**84**] which has an error rate of about 1% [**31**]. Still, training vision models on wide-angle images to detect oriented or partially obscured text is a significant challenge.

To make things easier, we may also propose auxiliary tasks aimed at inducing the model to learn the useful structures in the environment. At each timestep, we may have the head f perform a binary classification task given some relational question q and relevant house numbers binary encoded h_1 and h_2. For example, an important spatial question q would be this: "Are h_1 and h_2 on the same side of the street?", or, "If we were to continue 'forward' would we be moving 'up' the street?". Though we may train the model to learn these structures using a simple binary cross entropy loss, there remains the problem of identifying a model capable of numerical reasoning and extrapolation.

One candidate is the model from Neural Arithmetic Logic Units (NALU) [**111**] which proposed a learnable neural network architecture for numerical reasoning and extrapolation. NALUs are composed of Neural Accumulators (NAC) which are a special case of a linear layer whose transformation W consists of -1's, 0's, and 1's. This means that its outputs are constrained to be additions or subtractions instead of free transformations. The transformation

matrix in a NAC unit is composed of an element-wise multiplication:

$$W = tanh(\hat{\mathbf{W}}) \odot \sigma(\hat{\mathbf{M}})$$

where σ is the sigmoid function $\sigma(x) = \frac{1}{1+e^{-}x}$ and tanh is the hyperbolic tangent function which can be viewed as a rescaled sigmoid $tanh(x) = 2\sigma(x) - 1$. The range of σ is $[0, 1]$ (biasing towards 0 and 1) and tanh ranges between $[-1, 1]$ (biasing towards -1 and 1). Importantly, the use of these activation functions biases the weights towards -1's, 0's, and 1's during optimization. The output, or accumulation vector $\mathbf{a}$, is then computed as:

$$\mathbf{a} = \mathbf{W}\mathbf{x}$$

The efficacy of this bias on learning is demonstrated in several tasks; the most relevant being a gridworld navigation task [**102**]. In the gridworld experiment, the task is to navigate to a goal after a specific number of timesteps T. If the agent arrives at any timestep besides T, then it receives no reward. They train an LSTM model [**53**] with the on-policy RL algorithm A3C [**78**] on the output of a CNN applied to a 56×56 pixel image of the environment. In one version, T is concatenated to the output of the CNN, while in another it is additionally passed through an NAC unit before being passed back into the LSTM. The latter version performed significantly better when extrapolating to regions of T greater than those it had been trained on. The results on that modifying the activation functions and network architecture to preserve numerical reasoning can improve navigation performance even in the context high-resolution observations. Perhaps, modifying our architecture and introducing some auxiliary tasks may enable better extrapolation and navigation.

Notwithstanding this chapter's contributions, there remain many challenges ahead for those who wish to create assistive pedestrian navigation systems. One major challenge is to improve the efficiency of these systems, enabling deployment to edge devices while maintaining accuracy and speed. Large-scale object detection, optical character recognition, and neural networks in general remain computationally intensive despite recent advances. Another major challenge is to design a communication strategy for these assistive technologies that respects the user's level of vision, comfort, and safety.

Chapter 5

Conclusions & Future Work

This research defines the Outdoor Micro-Navigation (OMN) problem and attempts to solve it using modern neural networks and reinforcement learning algorithms. The OMN problem negatively effects people in the BVI community, to the extent that many are willing to pay $1 per minute for remote support from a sighted human. Prior works on vision-and-language navigation have focused mainly on indoor navigation [**21**] and large-scale gps-based outdoor navigation [**77**]. We contribute several formulations for the OMN problem in the form of grid world environments and the Sidewalk Environment for Visual Navigation (SEVN). We benchmark modern DRL algorithms on these environments and identify weaknesses and areas for future work.

We showed that we can train RL models on multi-modal data to perform multi-goal navigation tasks. These results, while preliminary, indicate that there is significant value in developing deep neural network models which navigate using scene text. In contrast to other works focusing on outdoor navigation with very large goal regions (on the scale of a city block) and action spaces (image observations are spaced 10 meters apart) [**77**], we produced an environment and results in an outdoor micro-navigation setting which is of significant and immediate use to the BVI community.

While the performance of the model trained in SEVN with the PPO algorithm does not achieve perfect navigation performance, it does indicate that this is a fruitful direction for future work. Some ideas for additional improvements include adding auxiliary tasks incentivizing the model to learn the structure of the environment, or using an experience replay mechanism. Another possible direction is to investigate a more UVFA-like approach to leverage a rich goal representation for generalizing to new tasks. This approach may pair nicely with the curriculum learning setup seen in Chapter 3. In particular, if the model can use the instruction I to infer the rough location of a goal (i.e., further up the street than these other goals), then it may perform better on what would otherwise be more difficult tasks later in the curriculum.

An organizing principle of this work is to develop methods which can solve OMN problems in the real-world. Several major aspects remain open, even after the tasks described in the grid environments and SEVN are fully solved. First, there is the problem of path-planning in irregular and/or complex environment. Figure 5.1 shows an example of the types of paths necessary for completing OMN tasks to addresses in Montreal.

Fig. 5.1. Path Planning. This figure shows an a high-level representation of the path planning problem present in an OMN task the ends off of the sidewalk. Blue rectangles surround each building, purple boxes surround doors, an orange rectangle indicates the street and sidewalk, and purple lines indicate the navigation trajectory that must be executed.

Second is the problem of transferring policies learned in simulation to the real-world. This challenge is quite interesting in this situation (as opposed to others such as robotic manipulation or self-driving cars) due to the proximity of the BVI person to the navigation task. In fact, there are many simple computer vision utilities which can be used to aid them achieve these goals such as optical character recognition and door detection devices. Indeed, BVI people vary widely in the nature of their disability, and as a result any application deployed to solve problems for the BVI should expend special focus on leveraging their ability.

A clear area for improvement beyond Sidewalk Environment for Visual Navigation (SEVN) is to create a full 3-dimensional reconstruction of the environment where we wish to solve the OMN task. A 3-D reconstruction, even a low-polygon count model as shown in 5.2, would enable three interesting directions. First, we may imagine a setting in which we can directly investigate the off-sidewalk navigation problem by generating novel views of the scene. Second, in SEVN one input to our model were natural images. It may be the case

that we will overfit to these images due to the somewhat limited nature of the input imagery. With a full 3-D reconstruction, we could imagine applying domain randomization or training on novel generated views. Third, we could learn a model to map real images to a low-poly space, then use that representation in our planning and execution of navigation tasks. This type of scheme has been used before in the literature by [**81**], to learn a navigation policy on a semantic segmentation (the intermediate representation).

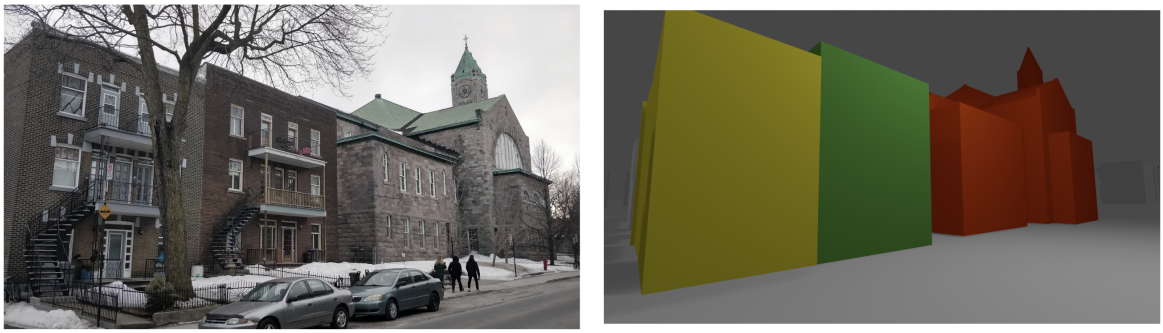


Fig. 5.2. 3-Dimensional Representations. Left: a natural image of a typical street in Montreal, Quebec. Right, an image of a low-poly 3-D model of the same scene.

We hope that the findings of this thesis represent a useful building block for those who aim to improve the lives of BVI people using technology. Though significant effort went into the design process for this work, any conceivable deployed version of technology solving an OMN problem in the real-world for BVI people must undergo much more scrutiny, including extensive user-testing and safety functionality.

References

[1] Peter Anderson, Angel X. Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva et Amir Roshan Zamir : On evaluation of embodied navigation agents. *CoRR*, 2018.

[2] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould et Anton van den Hengel : Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[3] 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. *In Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.

[4] Jeonghun Baek, Geewook Kim, Junyeop Lee, Sungrae Park, Dongyoon Han, Sangdoo Yun, Seong Joon Oh et Hwalsuk Lee : What is wrong with scene text recognition model comparisons? dataset and model analysis. *arXiv preprint arXiv:1904.01906*, 2019.

[5] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun et Hwalsuk Lee : Character region awareness for text detection. *CoRR*, abs/1904.01941, 2019.

[6] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt et David Silver : Successor features for transfer in reinforcement learning. *In Advances in neural information processing systems*, pages 4055–4065, 2017.

[7] Richard Bellman : *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 édition, 1957.

[8] Yoshua Bengio, Jérôme Louradour, Ronan Collobert et Jason Weston : Curriculum learning. *In Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 41–48, New York, NY, USA, 2009. ACM.

[9] Jeffrey P. Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C. Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, Samual White et Tom Yeh : Vizwiz: Nearly real-time answers to visual questions. *In Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 333–342, New York, NY, USA, 2010. ACM.

[10] Christopher M. Bishop : *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[11] Ali Furkan Biten, Ruben Tito, Andres Mafla, Lluis Gomez, Marçal Rusiñol, Ernest Valveny, CV Jawahar et Dimosthenis Karatzas : Scene text visual question answering. *arXiv preprint arXiv:1905.13648*, 2019.

[12] Fedor Borisyuk, Albert Gordo et Viswanath Sivakumar : Rosetta: Large scale system for text detection and recognition in images. *In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 71–79. ACM, 2018.

[13] Diana Borsa, André Barreto, John Quan, Daniel Mankowitz, Rémi Munos, Hado van Hasselt, David Silver et Tom Schaul : Universal successor features approximators. *arXiv preprint arXiv:1812.07626*, 2018.

[14] Erin Brady, Meredith Ringel Morris, Yu Zhong, Samuel White et Jeffrey P. Bigham : Visual challenges in the everyday lives of blind people. *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 2117–2126, New York, NY, USA, 2013. ACM.

[15] Erin L. Brady, Yu Zhong, Meredith Ringel Morris et Jeffrey P. Bigham : Investigating the appropriateness of social network question asking as a resource for blind users. *In Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, page 1225–1236, New York, NY, USA, 2013. Association for Computing Machinery.

[16] Samarth Brahmbhatt et James Hays : Deepnav: Learning to navigate large cities. *CoRR*, 2017.

[17] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang et Wojciech Zaremba : Openai gym, 2016.

[18] Yuri Burda, Harrison Edwards, Amos J. Storkey et Oleg Klimov : Exploration by random network distillation. *CoRR*, abs/1810.12894, 2018.

[19] Michal Bušta, Yash Patel et Jiri Matas : E2e-mlt-an unconstrained end-to-end method for multi-language scene text. *In Asian Conference on Computer Vision*, pages 127–143. Springer, 2018.

[20] Harris Chan, Yuhuai Wu, Jamie Kiros, Sanja Fidler et Jimmy Ba : ACTRCE: augmenting experience via teacher's advice for multi-goal reinforcement learning. *CoRR*, abs/1902.04546, 2019.

[21] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng et Yinda Zhang : Matterport3d: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

[22] Howard Chen, Alane Suhr, Dipendra Kumar Misra, Noah Snavely et Yoav Artzi : Touchdown: Natural language navigation and spatial reasoning in visual street environments. *CoRR*, abs/1811.12354, 2018.

[23] J. Chen et T. Chang : Modified ppo-rnd method for solving sparse reward problem in vizdoom. *In 2019 IEEE Conference on Games (CoG)*, pages 1–4, 2019.

[24] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew G Berneshawi, Huimin Ma, Sanja Fidler et Raquel Urtasun : 3d object proposals for accurate object class detection. *In* C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama et R. Garnett, éditeurs : *Advances in Neural Information Processing Systems 28*, pages 424–432. Curran Associates, Inc., 2015.

[25] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen et Yoshua Bengio : BabyAI: First steps towards grounded language learning with a human in the loop. *In International Conference on Learning Representations*, 2019.

[26] Maxime Chevalier-Boisvert, Lucas Willems et Suman Pal : Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid`, 2018.

[27] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk et Yoshua Bengio : Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[28] Joseph Paul Cohen : BlindTool – A mobile app that gives a "sense of vision" to the blind with deep learning, 2015.

[29] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth et Bernt Schiele : The cityscapes dataset for semantic urban scene understanding. *CoRR*, abs/1604.01685, 2016.

[30] Microsoft Corporation : SeeingAI - Talking Camera for the Blind, 2020.

[31] Ekin Dogus Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan et Quoc V. Le : Autoaugment: Learning augmentation policies from data. *CoRR*, abs/1805.09501, 2018.

[32] Peter Dayan : Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

[33] Harm de Vries, Kurt Shuster, Dhruv Batra, Devi Parikh, Jason Weston et Douwe Kiela : Talk the walk: Navigating new york city through grounded dialogue. *CoRR*, abs/1807.03367, 2018.

[34] Bella M. DePaulo et Jeffrey D. Fisher : The costs of asking for help. *Basic and Applied Social Psychology*, 1(1):23–35, 1980.

[35] Manfred Diaz, Roger Girgis, Thomas Fevens et Jeremy Cooperstock : To veer or not to veer: Learning from experts how to stay within the crosswalk. *In The IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2017.

[36] Piotr Dollár, Christian Wojek, Bernt Schiele et Pietro Perona : Pedestrian detection: An evaluation of the state of the art. *PAMI*, 34, 2012.

[37] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio López et Vladlen Koltun : CARLA: an open urban driving simulator. *CoRR*, 2017.

[38] Lei Fei, Kaiwei Wang, Shufei Lin, Kailun Yang, Ruiqi Cheng et Hao Chen : Scene text detection and recognition system for visually impaired people in real world. *In Target and Background Signatures IV*, volume 10794, page 107940S. International Society for Optics and Photonics, 2018.

[39] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein et Trevor Darrell : Speaker-follower models for vision-and-language navigation. *In Advances in Neural Information Processing Systems*, 2018.

[40] Ross B. Girshick : Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[41] Ross B. Girshick, Jeff Donahue, Trevor Darrell et Jitendra Malik : Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[42] Ian Goodfellow, Yoshua Bengio et Aaron Courville : *Deep Learning*. The MIT Press, 2016.

[43] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud et Vinay Shet : Multi-digit number recognition from street view imagery using deep convolutional neural networks, 2014.

[44] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra et Devi Parikh : Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. *In Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[45] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin et Aaron Courville : Improved Training of Wasserstein GANs. *arXiv e-prints*, page arXiv:1704.00028, Mar 2017.

[46] Ankush Gupta, Andrea Vedaldi et Andrew Zisserman : Synthetic data for text localisation in natural images. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2315–2324, 2016.

[47] Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo et Jeffrey P Bigham : Vizwiz grand challenge: Answering visual questions from blind people. *arXiv preprint arXiv:1802.08218*, 2018.

[48] Aric A. Hagberg, Daniel A. Schult et Pieter J. Swart : Exploring network structure, dynamics, and function using networkx. *In* Gaël Varoquaux, Travis Vaught et Jarrod Millman, éditeurs : *Proceedings of the 7th Python in Science Conference*, Pasadena, CA USA, 2008.

[49] Irtiza Hasan, Shengcai Liao, Jinpeng Li, Saad Ullah Akram et Ling Shao : Pedestrian detection: The elephant in the room, 2020.

[50] Bernhard Hassenstein et W Reichardt : Functional structure of a mechanism of perception of optical movement. *In 1er Congres International de Cybernetique 1956*, pages 797–801. Gauthier-Villars, 1956.

[51] Matthew Hausknecht et Peter Stone : Deep recurrent q-learning for partially observable mdps. *In 2015 AAAI Fall Symposium Series*, 2015.

[52] Karl Moritz Hermann, Mateusz Malinowski, Piotr Mirowski, Andras Banki-Horvath, Keith Anderson et Raia Hadsell : Learning to follow directions in street view. *CoRR*, 2019.

[53] Sepp Hochreiter et Jürgen Schmidhuber : Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[54] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto et Hartwig Adam : Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[55] G. Feldman I. Sobel : A 3 x 3 isotropic gradient operator for image processing. Talk presented at the Stanford Artificial Intelligence Project in 1968.

[56] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally et Kurt Keutzer : Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[57] Max Jaderberg, Karen Simonyan, Andrea Vedaldi et Andrew Zisserman : Synthetic data and artificial neural networks for natural scene text recognition. *CoRR*, 2014.

[58] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick et Ross Girshick : Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910, 2017.

[59] Kushal Kafle et Christopher Kanan : Visual question answering: Datasets, algorithms, and future challenges. *Computer Vision and Image Understanding*, 163:3–20, 2017.

[60] Karthik Kannan Karthik Mahadevan : Envision AI - OCR app that speaks out the visual world, 2020.

[61] Christof Koch et Shimon Ullman : *Shifts in Selective Visual Attention: Towards the Underlying Neural Circuitry*, pages 115–141. Springer Netherlands, Dordrecht, 1987.

[62] Eric Kolve, Roozbeh Mottaghi, Daniel Gordon, Yuke Zhu, Abhinav Gupta et Ali Farhadi : AI2-THOR: an interactive 3d environment for visual AI. *CoRR*, 2017.

[63] Ilya Kostrikov : Pytorch implementations of reinforcement learning algorithms. `https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail`, 2018.

[64] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein et Fei-Fei Li : Visual genome: Connecting language and vision using crowdsourced dense image annotations. *CoRR*, abs/1602.07332, 2016.

[65] Alex Krizhevsky, Ilya Sutskever et Geoffrey E Hinton : Imagenet classification with deep convolutional neural networks. *In* F. Pereira, C. J. C. Burges, L. Bottou et K. Q. Weinberger,

éditeurs : *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[66] Tejas D. Kulkarni, Ardavan Saeedi, Simanta Gautam et Samuel J. Gershman : Deep Successor Reinforcement Learning. *arXiv e-prints*, page arXiv:1606.02396, Jun 2016.

[67] Neuro X Labs : BlindSight uses cutting edge artificial intelligence to see the world as humans do, 2017.

[68] Zhe Li, Xiaoyu Wang, Xutao Lv et Tianbao Yang : Sep-nets: Small and effective pattern networks. *arXiv preprint arXiv:1706.03912*, 2017.

[69] Dahua Lin, Sanja Fidler et Raquel Urtasun : Holistic scene understanding for 3d object detection with rgbd cameras. *In Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[70] Long Ji Lin : Scaling up reinforcement learning for robot control. *In Machine Learning, Proceedings of the Tenth International Conference, University of Massachusetts, Amherst, MA, USA, June 27-29, 1993*, pages 182–189, 1993.

[71] Guan-Horng Liu, Avinash Siravuru, Sai Prabhakar, Manuela Veloso et George Kantor : Learning end-to-end multimodal sensor policies for autonomous navigation. *arXiv preprint arXiv:1705.10422*, 2017.

[72] Jingchao Liu, Xuebo Liu, Jie Sheng, Ding Liang, Xin Li et Qingjie Liu : Pyramid mask text detector. *CoRR*, 2019.

[73] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu et Alexander C Berg : Ssd: Single shot multibox detector. *In European conference on computer vision*, pages 21–37. Springer, 2016.

[74] Wei Liu, Chaofeng Chen, Kwan-Yee K Wong, Zhizhong Su et Junyu Han : Star-net: a spatial attention residue network for scene text recognition. *In BMVC*, volume 2, page 7, 2016.

[75] Mateusz Malinowski et Mario Fritz : A multi-world approach to question answering about real-world scenes based on uncertain input. *In Advances in neural information processing systems*, pages 1682–1690, 2014.

[76] Piotr Mirowski, Andras Banki-Horvath, Keith Anderson, Denis Teplyashin, Karl Moritz Hermann, Mateusz Malinowski, Matthew Koichi Grimes, Karen Simonyan, Koray Kavukcuoglu, Andrew Zisserman et Raia Hadsell : The streetlearn environment and dataset. *CoRR*, 2019.

[77] Piotr Mirowski, Matthew Koichi Grimes, Mateusz Malinowski, Karl Moritz Hermann, Keith Anderson, Denis Teplyashin, Karen Simonyan, Koray Kavukcuoglu, Andrew Zisserman et Raia Hadsell : Learning to navigate in cities without a map. *CoRR*, abs/1804.00168, 2018.

[78] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver et Koray Kavukcuoglu : Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016.

[79] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra et Martin Riedmiller : Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[80] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun et Alan Yuille : The role of context for object detection and semantic segmentation in the wild. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[81] Matthias Müller, Alexey Dosovitskiy, Bernard Ghanem et Vladlen Koltun : Driving policy transfer via modularity and abstraction. *CoRR*, abs/1804.09364, 2018.

[82] Raul Mur-Artal et Juan D. Tardós : Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 2017.

[83] National Foundation for the Blind : National Foundation for the Blind website.

[84] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu et Andrew Ng : Reading digits in natural images with unsupervised feature learning. *NIPS*, 2011.

[85] Augustus Odena, Christopher Olah et Jonathon Shlens : Conditional Image Synthesis With Auxiliary Classifier GANs. *arXiv e-prints*, page arXiv:1610.09585, Oct 2016.

[86] Seymour Papert : The summer vision project, 10 1966.

[87] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin et Aaron Courville : Film: Visual reasoning with a general conditioning layer. *In Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[88] J. Prewitt : Object enhancement and extraction. *Picture processing and Psychopictorics*, 1970.

[89] Trung Quy Phan, Palaiahnakote Shivakumara, Shangxuan Tian et Chew Lim Tan : Recognizing text with perspective distortion in natural scenes. *In Proceedings of the IEEE International Conference on Computer Vision*, pages 569–576, 2013.

[90] Noha Radwan, Abhinav Valada et Wolfram Burgard : Multimodal interaction-aware motion prediction for autonomous street crossing. *CoRR*, abs/1808.06887, 2018.

[91] Zobeir Raisi, Mohamed A. Naiel, Paul Fieguth, Steven Wardell et John Zelek : Text detection and recognition in the wild: A review, 2020.

[92] 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.

[93] Mengye Ren, Ryan Kiros et Richard Zemel : Exploring models and data for image question answering. *In Advances in neural information processing systems*, pages 2953–2961, 2015.

[94] Shaoqing Ren, Kaiming He, Ross Girshick et Jian Sun : Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

[95] Guodong Rong, Byung Hyun Shin, Hadi Tabatabaee, Qiang Lu, Steve Lemke, Mārtiņš Možeiko, Eric Boise, Geehoon Uhm, Mark Gerow, Shalin Mehta, Eugene Agafonov, Tae Hyung Kim, Eric Sterner, Keunhae Ushiroda, Michael Reyes, Dmitry Zelenkovsky et Seonman Kim : Lgsvl simulator: A high fidelity simulator for autonomous driving, 2020.

[96] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg et Li Fei-Fei : ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[97] Himanshu Sahni, Toby Buckley, Pieter Abbeel et Ilya Kuzovkin : Visual hindsight experience replay. *arXiv preprint arXiv:1901.11529*, 2019.

[98] 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.

[99] Tom Schaul, Daniel Horgan, Karol Gregor et David Silver : Universal value function approximators. *In International Conference on Machine Learning*, pages 1312–1320, 2015.

[100] J. Schmidhuber : Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010.

[101] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford et Oleg Klimov : Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[102] Santi Seguí, Oriol Pujol et Jordi Vitrià : Learning to count with deep object features. *CoRR*, abs/1505.08082, 2015.

[103] Shital Shah, Debadeepta Dey, Chris Lovett et Ashish Kapoor : Airsim: High-fidelity visual and physical simulation for autonomous vehicles. *In Field and Service Robotics*, 2017.

[104] Baoguang Shi, Xiang Bai et Cong Yao : An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304, 2016.

[105] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh et Marcus Rohrbach : Towards vqa models that can read. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8317–8326, 2019.

[106] Avi Singh, Larry Yang, Kristian Hartikainen, Chelsea Finn et Sergey Levine : End-to-end robotic reinforcement learning without reward engineering. *CoRR*, abs/1904.07854, 2019.

[107] S. Song, S. P. Lichtenberg et J. Xiao : Sun rgb-d: A rgb-d scene understanding benchmark suite. *In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 567–576, June 2015.

[108] Richard S. Sutton et Andrew G. Barto : *Reinforcement Learning: An Introduction*. The MIT Press, second édition, 2018.

[109] A Tao, J Barker et S Sarathy : Detectnet: Deep neural network for object detection in digits. *Parallel Forall*, 4, 2016.

[110] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba et Pieter Abbeel : Domain randomization for transferring deep neural networks from simulation to the real world. *In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017.

[111] Andrew Trask, Felix Hill, Scott E. Reed, Jack W. Rae, Chris Dyer et Phil Blunsom : Neural arithmetic logic units. *CoRR*, abs/1808.00508, 2018.

[112] Alexander Trott, Stephan Zheng, Caiming Xiong et Richard Socher : Keeping your distance: Solving sparse reward tasks using self-balancing shaped rewards, 2019.

[113] J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers et A.W.M. Smeulders : Selective search for object recognition. *International Journal of Computer Vision*, 2013.

[114] Mel Vecerik, Oleg Sushkov, David Barker, Thomas Rothörl, Todd Hester et Jon Scholz : A practical approach to insertion with variable socket position using deep reinforcement learning, 2018.

[115] P. Viola et M. Jones : Rapid object detection using a boosted cascade of simple features. *In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, 2001.

[116] Kai Wang, Boris Babenko et Serge Belongie : End-to-end scene text recognition. *In 2011 International Conference on Computer Vision*, pages 1457–1464. IEEE, 2011.

[117] Peng Wang, Xinyu Huang, Xinjing Cheng, Dingfu Zhou, Qichuan Geng et Ruigang Yang : The apolloscape open dataset for autonomous driving and its application. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[118] Martin Weiss, Simon Chamorro, Roger Girgis, Margaux Luck, Samira E. Kahou, Joseph P. Cohen, Derek Nowrouzezahrai, Doina Precup, Florian Golemo et Chris Pal : Navigation agents for the visually impaired: A sidewalk simulator and experiments, 2019.

[119] Martin Weiss, Margaux Luck, Roger Girgis, Chris Pal et Joseph Paul Cohen : A Survey of Mobile Computing for the Visually Impaired. Rapport technique, Mila, nov 2018.

[120] Bichen Wu, Forrest N Iandola, Peter H Jin et Kurt Keutzer : Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. *In CVPR Workshops*, pages 446–454, 2017.

[121] 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.

[122] Annie Xie, Avi Singh, Sergey Levine et Chelsea Finn : Few-shot goal inference for visuomotor learning and planning, 2018.

[123] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel et Yoshua Bengio : Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044, 2015.

[124] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta et Roozbeh Mottaghi : Visual semantic navigation using scene priors, 2019.

[125] J. Yao, S. Fidler et R. Urtasun : Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. *In 2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 702–709, 2012.

[126] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan et Trevor Darrell : BDD100K: A diverse driving video database with scalable annotation tooling. *CoRR*, 2018.

[127] Pengfei Zhu, Xin Li et Pascal Poupart : On improving deep reinforcement learning for pomdps. *CoRR*, abs/1704.07978, 2017.

[128] Pengfei Zhu, Xin Li, Pascal Poupart et Guanghui Miao : On improving deep reinforcement learning for pomdps. *arXiv preprint arXiv:1704.07978*, 2017.

[129] Yixing Zhu et Jun Du : Textmountain: Accurate scene text detection via instance segmentation. *CoRR*, abs/1811.12786, 2018.

Appendix A

Example Interview with an Accessibility Specialist

Maryse Legault is a visually impaired woman who works as an accessibility specialist at HumanWare, a global leader in assistive technology for people who are blind or have low vision needs. In this interview, we discussed her needs in order to feel comfortable travelling to new locations.

Interviewer: How do you prepare for travelling or going outside?

Maryse: Like sighted people, I need to prepare myself when leaving the home. There are some differences though, for example, I am more comfortable knowing which side of the street the metro station door or the bus stop post is on. If I need to cross a boulevard to get there, knowing if there is a crosswalk, traffic lights, an underground passage or a pedestrian bridge can be very helpful. I also need to know about the direction of traffic and potential obstacles like roadwork, especially if I know that I will not be able to reroute with a map on my phone.

Interviewer: This seems to be a lot of preparation, but I hear that there are mobile phone applications that can be used to help with those difficulties on the ground. Can you give us examples of applications that you use?

Maryse: That's true. When I take the bus I use the Navigation app's vocal instruction to know which street I am on. Sometimes it can be confusing, especially in the US where you can be heading East on West Robert Smith Avenue. This app helps me to get off at the correct stop. When I'm on a sidewalk, I must always ask other pedestrians question to figure out where I am. Like, what is the current house number? I can sometimes ask for help to cross intersections or even more.

Interviewer: Okay, so navigating the city seems to still be very challenging. What would be helpful for you to know to make navigation easier between intersections, and at the intersection?

Maryse: Well, between intersections I would like to know about obstacles to avoid like traffic cones and bicycles. At the intersection, knowing if there is a crosswalk, how many lanes I have to cross, if there are some traffic lights or not and if the light is green would be

very helpful! There is an application that can help with general object detection but there is no feature that lets me select the kind of object I want to detect during navigation. I think I would like an app where I can select a mode which does not overwhelm me with too much information.

Interviewer: It sounds like these apps give you more information than you need and are sometimes more annoying than helpful. I suppose that when you arrive at your destination you will be looking for certain types of objects. Can you tell me more about that?

Maryse: First let's give a bit of context. When I arrive at my destination I first need to check that I am at the right place. I usually do that by asking people around and the information I am asking for is very dependent on my destination. For example, if I am going to a shopping mall the main issue is that it's typically an open space. These are more difficult to understand than streets and sidewalks. It is often difficult for me to walk between parking lots and pedestrian areas. There are very often few or no sidewalks from the bus stop to the shopping center door. So being able to detect those things would be very helpful. Knowing the store's location when arriving at the shopping center would also be a plus. Then I would also need information about what's on the storefront like the store logo, the signs, the opening hours and the house number for example.

Interviewer: Okay, and when you are at the right location is the story over?

Maryse: Well unfortunately not, getting around inside is also very challenging. For example, if I am at a restaurant I need to know if there are any available seats at the tables and how to get to them. I might also need to know how to get to the counter. And I would say even before that finding the entrance or the door of the restaurant is very challenging as dogs are not really able to do that.

Appendix B

Market Research

To understand the challenges of creating assistive technology for the BVI, we performed the following market research. By using online indices of assistive applications for the BVI, we identified several dozen interesting apps. We then exchanged emails and interviewed some, asking questions about their development process, user-testing, and adoption.

Application Name	*URL*	*Status*
EnvisionAI	https://www.letsenvision.com/	Interviewed
SeeingAI	https://www.microsoft.com/en-us/ai/seeing-ai	Interviewed
BlindSight	https://play.google.com/store/apps/details?id=com.neuroxlabs.BlindSight	Interviewed
IDentiFi	https://itunes.apple.com/gb/app/identifi-object-recognition-for-visually-impaired	Interviewed
TalkingGoggles	http://sparklingapps.com/goggles/	Interviewed
Boop Light Detector	http://arii.github.io/boop/	Interviewed
Third Eye	https://thirdeyeglass.com/	Interviewed
Blindly	https://play.google.com/store/apps/details?id=com.aicursor.blindly	Interviewed
EyeCYou	https://play.google.com/store/apps/details?id=com.eyecyoutech.apps.eyecyou	Interviewed
YourEyes	https://play.google.com/store/apps/details?id=org.opencv.sample.opencv_mobilenet	Interviewed
Minerva	https://play.google.com/store/apps/details?id=com.chcepe.minerva	Interviewed
CosyRobo	http://www.cosyrobo.com/	Interviewed
BeMyEyes	https://www.bemyeyes.com/	Interviewed
SmartSight	https://play.google.com/store/apps/details?id=com.esh.smartsight	Emails Exchanged
Color Detector	https://play.google.com/store/apps/details?id=com.keesadens.colordetector	Emails Exchanged
vOICe	https://play.google.com/store/apps/details?id=vOICe.vOICe	Emails Exchanged
Vocal Eyes	http://vocaleyes.ai/	Emails Exchanged
SeeingAssistant Home	https://itunes.apple.com/us/app/seeingassistant-home-lite/id625110066	Emails Exchanged
LightDetector	https://itunes.apple.com/us/app/light-detector/id420929143	Emails Exchanged
ColorVisor	https://itunes.apple.com/us/app/colorvisor/id511093568	Emails Exchanged
ColorSay	https://itunes.apple.com/us/app/colorsay/id605398028	Emails Exchanged
CamFind	https://camfindapp.com/	Emails Exchanged
AIPoly	https://itunes.apple.com/us/app/aipoly-vision/id1069166437	Emails Exchanged
Voiceye	http://www.voiceye.com/eng/	Emails Exchanged
Eye-D	https://eye-d.in/	Emails Exchanged
Looktel	http://www.looktel.com/products	Emails Exchanged
TaptapSee	https://itunes.apple.com/ca/app/taptapsee/id567635020	Emails Exchanged
Object Recognition	https://play.google.com/store/apps/details?id=com.hubble.space.cloud.vision	Emails Exchanged
Bespecular	https://play.google.com/store/apps/details?id=com.bespecular.specular	Not a good fit
ColoredEye	https://itunes.apple.com/us/app/coloredeye/id388886679	Not a good fit
Autour	https://itunes.apple.com/us/app/autour/id887476373	Not a good fit
Voice: OCR	https://apps.apple.com/us/app/voice-take-picture-have-it/id903772588	Not a good fit

Table B.1. Market Research. This table contains the names and website links for assistive technology applications supporting the blind and visually impaired. The status column indicates the manner of interaction with the developers of these applications.

Appendix C

Supplementary Material for the Grid World Experiments

Mean Diff	Max Diff	Learning Rate	Framestack
12.00	23.00	0.005126	1
11.67	21.00	0.003328	8
7.00	10.00	0.005393	16
6.33	9.00	0.006356	8
6.33	9.00	0.001522	16
5.67	8.00	0.008439	1
5.67	7.00	0.009014	8
5.00	6.00	0.008207	16
4.67	6.00	0.006960	16
4.00	4.00	0.000833	8
4.00	4.00	0.000973	8
3.33	4.00	0.000455	16
3.33	4.00	0.000671	1
3.00	3.00	0.000032	1
3.00	3.00	0.000220	16
3.00	3.00	0.000020	1
3.00	3.00	0.000340	16
3.00	3.00	0.000031	1
2.00	3.00	0.026617	1
1.00	1.00	0.000016	1

Table C.1. Hyper-parameter search on Grid City. We report the mean and max difficulty achieved by models in a hyper-parameter search on the Curriculum Grid City environment after 1 million frames of experience. It appears that frame-stacking was not strongly correlated with an increased ability to perform the OMN task, perhaps as a result of the static nature of the environment.

Appendix D

Supplementary Material for the Sidewalk Environment for Visual Navigation

The hyperparameters used to train our PPO model are fairly standard, and we did not find much sensitivity in the policy learned, training efficiency, or final performance. We also provide several additional maps of the SEVN dataset showing goal locations and urban zoning.

D.1. Hyperparameters

Hyperparameter	**Value**
Learning Rate	3×10^{-4}
Number of Steps	2048
Value Loss Coefficient	0.5
Linear LR Decay Schedule	True
Entropy Coefficient	0
Gamma (γ)	0.99
Generalized Advantage Estimation (λ)	0.95
Maximum Gradient Norm	0.5
Number of PPO Epochs	4
Number of PPO Mini-Batches	32
Clipping Parameter	0.2

Table D.1. Hyperparameters. We report the hyper-parameters used while training. These settings are quite similar to those PPO parameters described in [**101**].

D.2. Goal Location and Zoning Maps

The maps shown in the body of the work represent one view on our dataset. Here, we present several other visualizations of this geographic region.

Goal Locations: In SEVN, doors with visually identifiable house numbers are available as goals. The distribution of these goals is dependent on many factors – visibility of the house number, visibility of the door, clear relationship between the two, but most importantly the type and distribution of buildings along the street. In figure D.1, we show the spatial distribution of goals within SEVN.

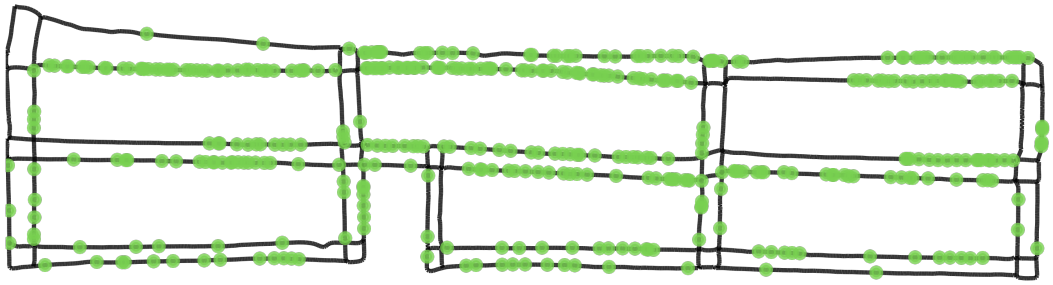


Fig. D.1. Map with Goal Locations in the same orientation as previous maps, with the Small data split being the top right block. We can see that the target locations are well spread out, but still diverse, with some dense areas (often commercial or residential zones as seen in Figure D.2), whereas parks and industrial zones are more sparsely populated with addresses.

Zoning: Arguably the primary factor determining the distribution of goal locations is Montreal's urban zoning regulation. In figure D.2 we show the primary zone determined by the municipality of Montreal around and within the region captured by SEVN.

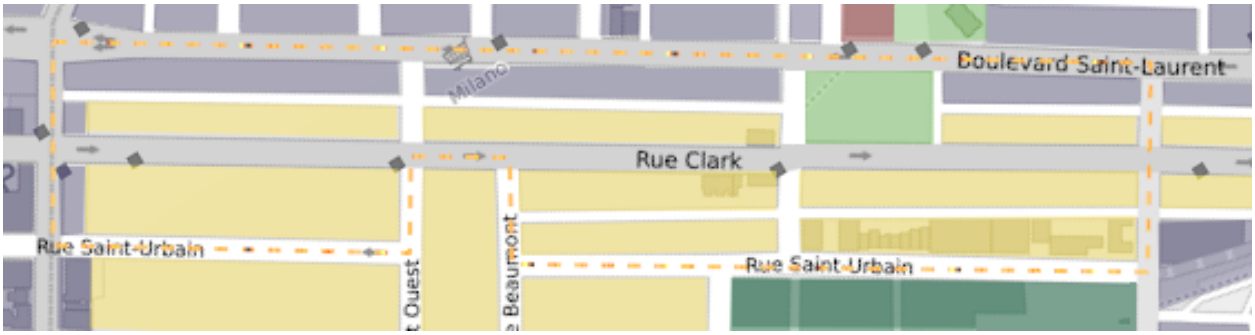


Fig. D.2. Usage and Zoning Map for the region covered by SEVN, outlined by the **dotted orange** line. In this graphic, **purple** indicates commercial zones, **yellow** indicates residential zones, **green** indicates parks and outdoor recreation areas, **dark teal** is industrial, and **red** indicates collective or cultural institutions (in this case, a cathedral). Many areas are mixed, in the sense that they can be commercial and residential. These zones are colored based on their primary purpose. Zoning rules are complex and the details can be viewed on the city of Montreal's website (link). The basis of this map is the OpenStreetMap Carto (Black and White), with zoning data courtesy of the city of Montreal (link).