

Tartu University  
Faculty of Science and Technology  
Institute of Technology

Kirill Rodionov

**INTUIT-VLNCE: Autonomous Navigation through Vision-and-Language**

Master's thesis (30 EAP)  
Robotics and Computer Engineering

Supervisor:

Kallol Roy

Tartu 2023

# Resümee/Abstract

## **INTUIT-VLNCE: Autonoomne Navigatsioon kasutades Nägemust-ja-Keelt**

Käesoleva lõputöö eesmärk oli ilmse intuitsiooniga Kehastunud Agendi arendus. Kehastunud Agent peab navigeerima läbi pideva siseruumi, kasutades antud Loomuliku Keele instruksiooni ja egotsentrilist nägemust, nagu on kirjeldatud Nägemus-ja-Keel ülesannes. Lõputöö pakkub v"alja luua ilmset intuitsiooni: Agent ennustab ette nii tegu, mida on vaja hetkseisul sooritada, kui ka tegusid, mida kavatsetakse sooritada tulevikus.

Agendi poliis oli trennitud samamoodi, nagu trennimine toimus LAW-VLNCE projektis [1]. Hindamine näitas negatiivseid tulemusi pärast pakutud meetodi rakendamist.

**CERCS:** P176 Tehisintellekt; T125 Automatiseerimine, robotika, control engineering;

**Märksõnad:** robotika, loomulik keel, kehastunud agent, autonoomia

## **INTUIT-VLNCE: Autonomous Navigation through Vision-and-Language**

The aim of this thesis was to developed an Embodied Agent with explicit intuition capable of navigating indoor Continuous Environments based on provided Natural Language instruction and Agent's egocentric vision as part of a Vision-and-Language Navigation task. The thesis proposes creating explicit intuition by making an Agent predict not only an action to perform at a given time, but also predicting actions for the future.

An Agent's policy was trained mimicking the training procedure from LAW-VLNCE project [1]. Evaluations showed negative results after implementing proposed method.

**CERCS:** P176 Artificial intelligence; T125 Automation, robotics, control engineering

**Keywords:** robotics, natural language, embodied agent, autonomy

# Contents

|                                                   |           |
|---------------------------------------------------|-----------|
| <b>Resümee/Abstract</b>                           | <b>2</b>  |
| <b>List of Figures</b>                            | <b>5</b>  |
| <b>List of Tables</b>                             | <b>7</b>  |
| <b>Abbreviations. Constants. Generic Terms</b>    | <b>8</b>  |
| <b>1 Introduction</b>                             | <b>10</b> |
| 1.0.1 Research Hypothesis . . . . .               | 11        |
| 1.0.2 Problem Description . . . . .               | 11        |
| <b>2 Background</b>                               | <b>13</b> |
| 2.1 Preliminaries . . . . .                       | 13        |
| 2.2 Model of Environment . . . . .                | 13        |
| 2.3 Scene Datasets . . . . .                      | 15        |
| 2.4 Instruction and Path datasets . . . . .       | 17        |
| <b>3 Methodology</b>                              | <b>19</b> |
| 3.1 R2R_VLNCE_v1-3 . . . . .                      | 19        |
| 3.2 VLN-CE . . . . .                              | 21        |
| 3.3 Task description . . . . .                    | 22        |
| 3.4 Policy Development . . . . .                  | 23        |
| 3.4.1 Inflection Weights . . . . .                | 23        |
| 3.4.2 Progress Monitor . . . . .                  | 23        |
| 3.4.3 Dataset Aggregation . . . . .               | 25        |
| 3.5 Cross-Modal Attention Model . . . . .         | 26        |
| 3.6 LAW-VLNCE . . . . .                           | 27        |
| 3.7 Evaluation Metrics . . . . .                  | 27        |
| 3.8 Proposed Method . . . . .                     | 29        |
| 3.9 Design of Experiments: INTUIT-VLNCE . . . . . | 30        |
| 3.10 Training and Evaluation . . . . .            | 32        |
| 3.10.1 Main Training Phase . . . . .              | 33        |
| 3.10.2 Fine tuning Phase . . . . .                | 34        |
| <b>4 Experimental Results</b>                     | <b>36</b> |
| 4.1 Results in the Main Training Phase . . . . .  | 36        |
| 4.2 Results in the Fine tuning Phase . . . . .    | 38        |

|          |                                                                                 |           |
|----------|---------------------------------------------------------------------------------|-----------|
| <b>5</b> | <b>Analysis and Discussion</b>                                                  | <b>41</b> |
| 5.1      | Analysis of Results in the Main Training Phase . . . . .                        | 41        |
| 5.2      | Analysis of Results in the Fine Tuning Phase . . . . .                          | 41        |
| 5.3      | Discussion of the Results . . . . .                                             | 42        |
| <b>6</b> | <b>Future work</b>                                                              | <b>43</b> |
| <b>7</b> | <b>Conclusion</b>                                                               | <b>44</b> |
|          | <b>Conclusion</b>                                                               | <b>44</b> |
|          | <b>Bibliography</b>                                                             | <b>46</b> |
|          | <b>MATTERPORT END USER LICENSE AGREEMENT FOR ACADEMIC USE OF<br/>MODEL DATA</b> | <b>50</b> |
|          | <b>Non-exclusive license</b>                                                    | <b>56</b> |



# List of Figures

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |    |
|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1.1 | Dynamic textual generative model for dialogue . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | 10 |
| 1.2 | Promotional material for GitHub copilot [2] . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | 11 |
| 2.1 | Renders of two houses used in the development of FollowNet. Notice the green grid placed on the floor, where nodes depict possible locations and edges - possible rotations (as well as navigable paths) of an Agent. [3] . . . . .                                                                                                                                                                                                                                                                                                              | 14 |
| 2.2 | Left: an example nav-graph (in blue) of single floor in a Matterport3D Simulator. Upper-Right: a sample view from an Agent’s camera; blue discs indicate available nodes to travel to. Lower-Right: an example vln instruction from a Room-to-Room (R2R) dataset. [4] Image taken from Matterport3D Simulator GitHub repository [5]. . . . .                                                                                                                                                                                                     | 14 |
| 2.3 | Action Space variants in Continuous Environment . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 15 |
| 2.4 | Examples of synthesised scenes . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 16 |
| 2.5 | Environmental Dropout, where all chairs are masked-out from the view [6]. . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | 16 |
| 2.6 | Examples of data contained within Matterport3D dataset. Image taken from Matterport3D GitHub.io project page [7]. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                        | 17 |
| 2.7 | An example of an instruction from ALFRED being carried out in AI2-THOR simulator as depicted with 6 captioned framed. [8] . . . . .                                                                                                                                                                                                                                                                                                                                                                                                              | 18 |
| 3.1 | A scheme for Agent policy training stage 1: training data generation . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | 24 |
| 3.2 | A simplified scheme representing the general architecture of Cross-Modal Attention Model taken from the original paper [9] . . . . .                                                                                                                                                                                                                                                                                                                                                                                                             | 27 |
| 3.3 | Difference between a path created with a sensor, which navigates towards the goal position in a shortest way possible (red), and a path which follows intermediate waypoints, which are placed in respect to the NL instruction (blue). Additionally, several Agents are spread throughout a scene with coloured arrows showing the direction, in which the Agent would move listening to commands of each sensor. Image is taken from [10]. . . . .                                                                                             | 28 |
| 3.4 | An example of how an instruction contains information about the future based on a VLN application with RxR and Matterport3D Simulator. The image is a top down view on a scene segment with an Agent (black triangle). There are also coloured visualisations of directions an agent have and will have viewed; the colour corresponds to a slice of an instruction. While being in a green zone, the Agent can already have an intuition about moving and observing the scene at cyan and blue zones (at least). Image taken from [11]. . . . . | 35 |
| 4.1 | Loss values at each batch in the Main Training Phase . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | 36 |
| 4.2 | Main Training Phase SR and OS metrics . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | 37 |

|     |                                                                                                                                                                  |    |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 4.3 | The remaining metrics per evaluated checkpoint from the first phase of training.<br>The purple vertical line marks the best performing checkpoint (43). . . . .  | 38 |
| 4.4 | Loss values at each batch in the Fine tuning Phase . . . . .                                                                                                     | 39 |
| 4.5 | Fine tuning Phase SR and OS metrics . . . . .                                                                                                                    | 39 |
| 4.6 | The remaining metrics per evaluated checkpoint from the fine tuning phase of<br>training. The purple vertical line marks the best-performing checkpoint (5). . . | 40 |

# List of Tables

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |    |
|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 3.1 | A structure of a $\{split\}.json.gz$ file's episode, which is a datapoint in R2R_VLNCE dataset. [12]                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 20 |
| 3.2 | A structure of values inside $\{split\}_{gt}.json.gz$ file from R2R_VLNCE dataset. Example is given based on the resulted dictionary from key '1'. [12]                                                                                                                                                                                                                                                                                                                                                                                                   | 20 |
| 4.1 | Main Training Phase best checkpoint metrics                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 37 |
| 4.2 | Fine tuning Phase best checkpoint metrics                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | 40 |
| 5.1 | Comparison between our best checkpoint at 1st phase and an equivalent model from VLN-CE paper [9]. We skip metrics, that are not reported for the alternative model. Our model performs slightly worse, than the alternative. This may attributed to the incomplete training set our model used. We cannot compare with a similar model from LAW_VLNCE paper, because they only report results after fine-tuning, [1].                                                                                                                                    | 41 |
| 5.2 | Comparison between our best model and two models from a preceding paper [1]. <code>goal</code> is a CMA with PM and IW and supervised with a greedy goal-oriented sensor, trained under the same conditions as we did. Essentially an equivalent to the best performing CMA model from VLN-CE [9]. <code>LAW_pano</code> is the same, but supervised with LAW sensor,i.e. our model without our modifications. Average steps taken metric is skipped, because it was not reported in the original paper [1]. Our model is under-performing in both cases. | 42 |

# List of abbreviations, constants etc.

**AGI** - Artificial General Intelligence

**AI** - Artificial Intelligence

**ALFRED** - Action Learning From Realistic Environments and Directives

**BPO** - Business Process Outsourcing

**Dagger** - Dataset Aggregation

**DNN** - Deep Neural Network

**GloVe** - Global Vectors (for Word Representation)

**GPS** -Global Positioning System

**GRU** Gated Recurrent Unit

**HPC** High Performance Computing

**LAW** - Language-Aligned Waypoint

**LLM** - Large Language Models

**LMDB** - Lightning Memory-Mapped Database

**LSTM** - Long Short-Term Memory

**Nav-graph** - navigational graph

**ML** - Machine Learning

**NL** - Natural Language

**NN** - Neural Network

**PC** - Personal Computer

**R2R** - Room-to-Room

**R4R** - Room-for-Room

**ResNet** - Residual Network

**RGB-D** - Red Green Blue - Depth, refers to channels of an image

**RNN** - Recurrent Neural Network

**RxR** - Room-Across-Room

**TUIT** - Tartu University Institute of Technolog

**VLN** - Visual-and-Language Navigation

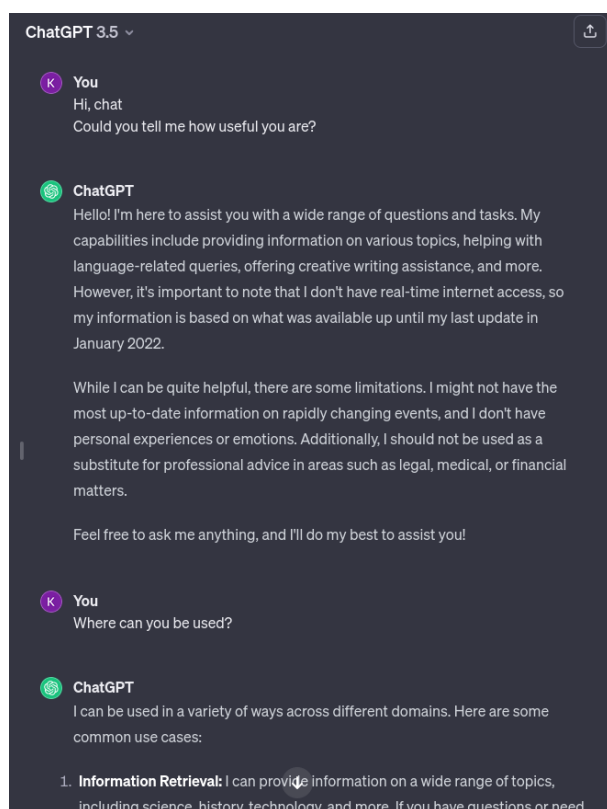
**VLN-CE** - Visual-and-Language Navigation in Continuous Environments

**ToT** - Tree-of-Thought

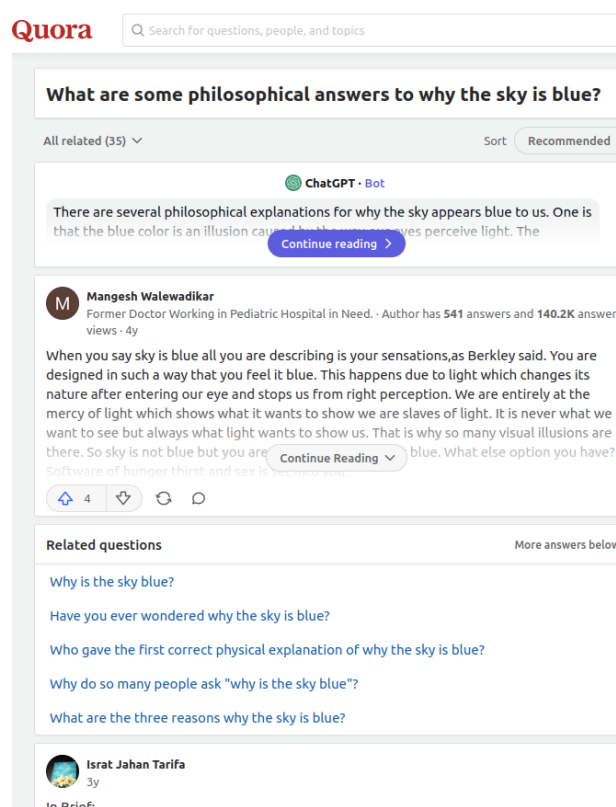
**WPN** - Waypoint Prediction Network

# 1 Introduction

We are in exciting times where AI is pervasive in our everyday lives. State-of-the-art machine learning models can now tackle more and more complex tasks, from playing Go to solving mathematical problems. Among all the disciplines, Natural Language Processing (NLP) shows huge promise, especially with the arrival of ChatGPT. Applications such as text generation, summarization, and sentiment analysis are now widely used in media, and advertisement [13]. The multi-modal applications of integrating images and text are ubiquitous that find applications in providing captions to images. Machine learning models Midjourney [14], Stable Diffusion [15] and DALL-E [16] are now capable of generating images based on text prompts. Chatbots/Virtual assistants work on the information retrieval paradigm and answer queries in real-time used in applications ranging from health care to business process outsourcing (BPO). Large Language Models (LLMs) based on the transformer neural networks introduced by OpenAI changed the AI landscape. LLMs trained on large number of text tokens in an auto-



(a) A small interaction between the author and ChatGPT 3.5 [17]



(b) popular website for submitting and answering questions (Quora) now hosts a ChatGPT based bot to provide an additional answer automatically [18]

Figure 1.1: Dynamic textual generative model for dialogue

regressive manner started showing artificial general intelligence (AGI) capabilities in solving very complex tasks of automated theorem prover. The divide between synthetic intelligence and natural intelligence is slowly blurring and thus pointing us to unknown terrain. Examples of the capabilities of interactive chatbots for question answering and generating code are shown in Figure 1.1 and Figure 1.2 respectively.

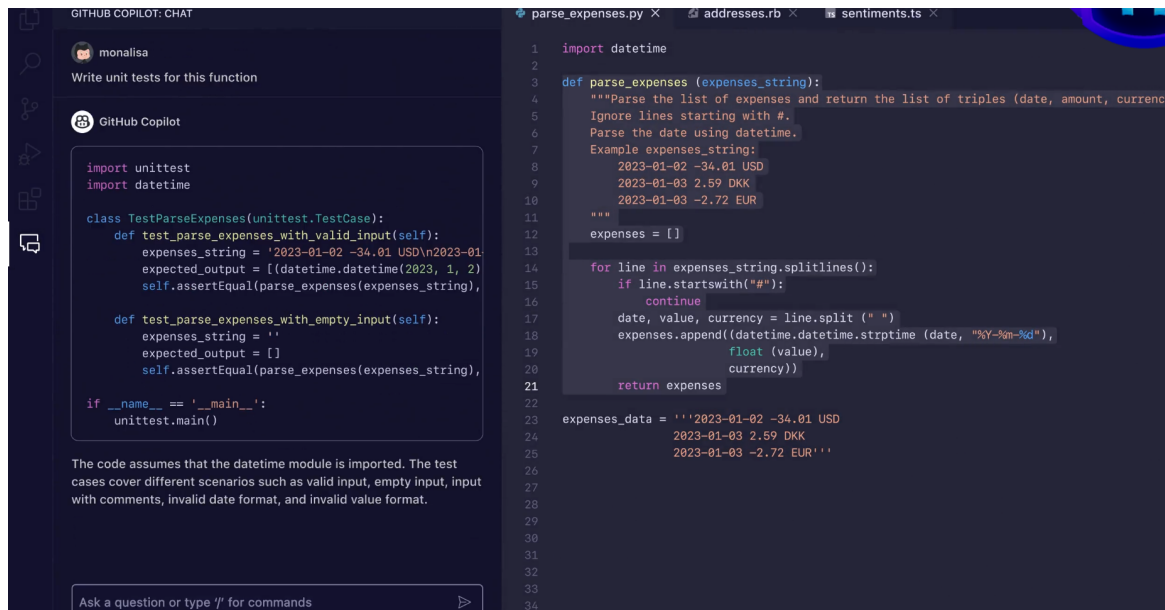


Figure 1.2: Promotional material for GitHub copilot [2]

In this thesis, we have investigated the instruction-tuned robot navigation problem. Though there are standard navigation algorithms (e.g. shorted path), the use of natural language as a prior knowledge for the navigation task is a less studied area. Natural language adds logic (common sense) and planning through syntax and semantics while solving the navigation task. Text instructions "Go through the corridor and into the kitchen and bring me a spoon from the coffee table" - provides the strategy for the optimal path from source to destination. The augmented visual intelligence through the Visual-and-Language Navigation (VLN) task, gives the situational awareness to the robot to navigate its surroundings based on a given instruction. This thesis investigates the following hypothesis

### 1.0.1 Research Hypothesis

Natural language provides an informed prior (through syntax and semantics) to search for an optimal path from source to destination

### 1.0.2 Problem Description

An Agent (robot) located in an unknown indoor environment without any prior knowledge of its location is given a text command to navigate. The agent is also given the additional ability to egocentrically visually perceive its environment. Based on the instruction set and perception, the agent optimally chooses its actions sequentially. The agent takes a series of local actions to achieve its global objective of reaching the destination through a minimum cost (measure) path. The local decisions to choose from a set of actions are function text commands and visual images. The agent comprehends its surroundings from the multimodal data of text and visual

data and improvises its local decisions accordingly. We restrict our navigation task to indoor environments for simplicity.



## 2 Background

In this section, we briefly describe the concepts, environmental models, and the datasets used.

### 2.1 Preliminaries

**Agent** - the thing or the person solving the task; a robot in our case

**Oracle** - an expert algorithm which is able to solve a given task perfectly, what an agent will try to mimic in imitation learning

**Embodiment** - a quality of having a physical body located somewhere in an environment, able to interact with a physical environment

### 2.2 Model of Environment

To train our Agents effectively to navigate, we need environments for an Agent to navigate. The simulated environments bypass the difficulty of training the Agent in real-life environments. Training in real-life environments with actual buildings, roads, spaces, and robot are expensive, time consuming and hazardous. Environment models are broadly split into two categories based on the type of navigable space: Discrete and Continuous.

**Discrete Environments.** Here environment is represented as a navigational graph (nav-graph), where each node is a possible place for an Agent to visit and each edge represents a navigable path between places. Discrete environment gives a high-level abstraction and reasoning for the agent’s navigation path. Moving to a new node on the nav-graph, our Agent receives a new image with respect to the reached node. The image can be a whole panorama, i.e. all possible views from the node, or it can be a single view from the Agent’s heading direction. For example in FollowNet project [3], a 3D indoor house is encoded (represented) as a 2D regular grid, where at each time-step the agent can either step straight into the forward facing node or turn 90° in either left or right and receives an image corresponding to the Agent’s position and rotation. In Matterport3D Simulator the node alignment are distributed throughout the floor with average separation of 2.25m. The edges (navigable paths) connect the nodes within 5m of each other with respect to serious obstacles such as walls. The Agent can pan and elevate the camera and move to a next connected node caught in the camera’s field of view. After an action is executed the Agent receives a new image and a set of navigable nodes [4]. The discrete nature of the navigation space gives a perfect localization effect. The agent does not have a chance to overshoot a node nor can it collide with an obstacle by accident. The quantization viewpoints make the observed gathered data to be sparse [9]. Training an agent on a discrete environment and then transferring the trained agent into the low-level real world comes with a penalty. Anderson et al. reported that the accuracy (of the agent) went from 55.9% in simulations to 22.5%

in the real world, without any prior knowledge [19].

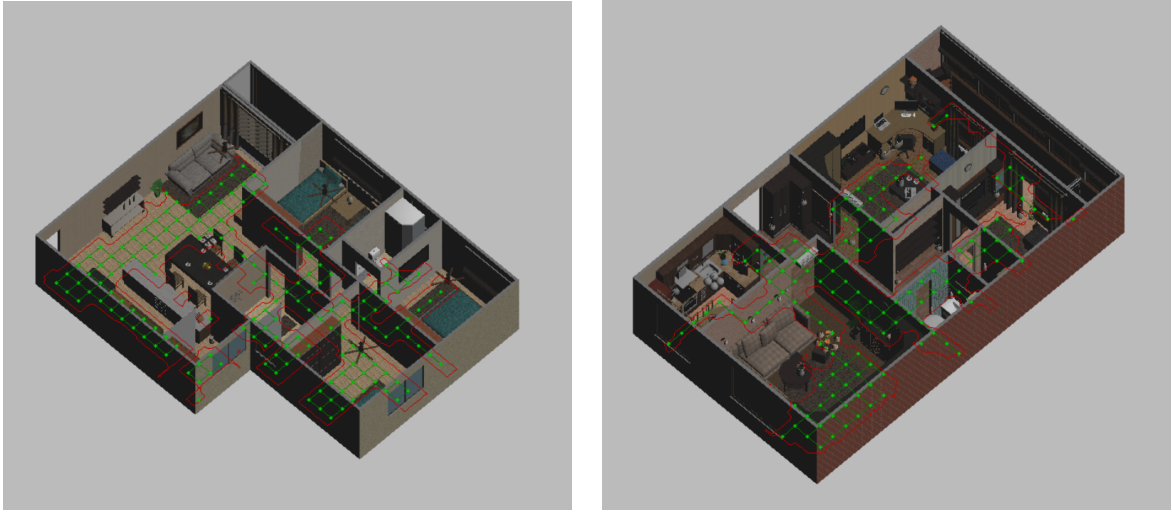


Figure 2.1: Renders of two houses used in the development of FollowNet. Notice the green grid placed on the floor, where nodes depict possible locations and edges - possible rotations (as well as navigable paths) of an Agent. [3]

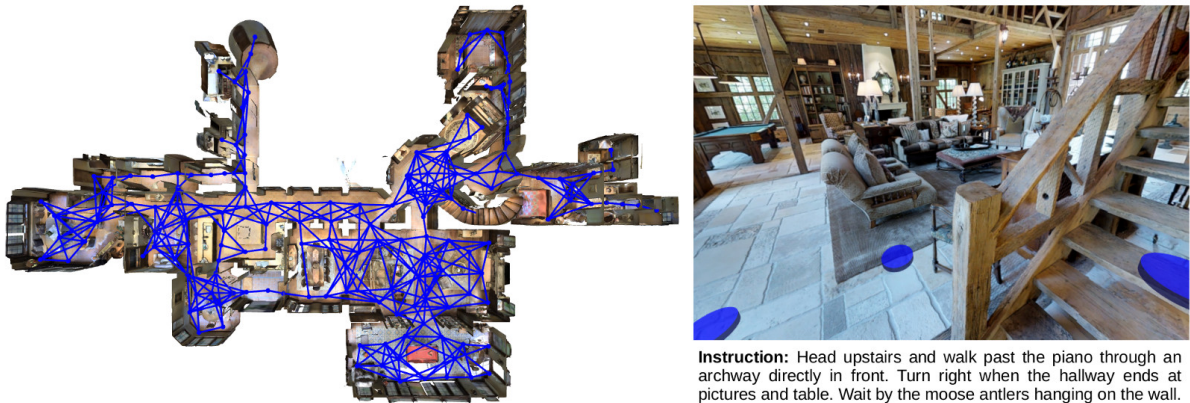


Figure 2.2: Left: an example nav-graph (in blue) of single floor in a Matterport3D Simulator. Upper-Right: a sample view from an Agent's camera; blue discs indicate available nodes to travel to. Lower-Right: an example vln instruction from a Room-to-Room (R2R) dataset. [4] Image taken from Matterport3D Simulator GitHub repository [5].

**Continuous Environment.** In a continuous space there are no obvious edges/graphs to traverse and exploit. Agent's location is unrestricted to nodes set, and we get data from the environment by infinitesimal movement of the Agent. Data here are more voluminous and fine-grained from the fine-grained sequences of actions as reported by Krantz and Wijmans et al. Continuous environment for the same scene and instruction+paths and collected datasets are shown in [9]. A removal of nav-graph abstraction provides a richer selection of Agent's action space definition. Krantz and Gokaslan et al. investigated 6 action space for a Waypoint Prediction Network [20] as shown in Figure 2.3. 6 action space variants for Waypoint Prediction Network (WPN) in a continuous environment. At each times-tep the Agent receives a panoramic RGB-D observation as a set of 12 images from 12 evenly spaced angles. The Agent can select one of these

angles as its new heading. Additionally, it chooses an offset for a heading along with the step distance [20].

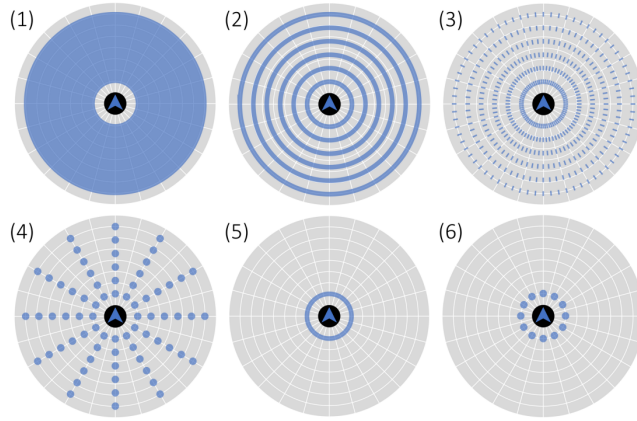


Figure 2.3:

1. Offset and step distance are real numbers, meaning an Agent is able to step anywhere within a continuous toroid.
2. Heading offset remains to be of continuous nature, but the step distance is made discrete.
3. Both the heading offset and step distance are discrete.
4. Step distance is discrete, but heading offset is fixed to  $0^\circ$ , resulting in only 12 angles an Agent can turn to.
5. Step distance is fixed to a constant value, whereas heading offset is continuous.
6. Both the step distance and heading offset are constant.

[20]

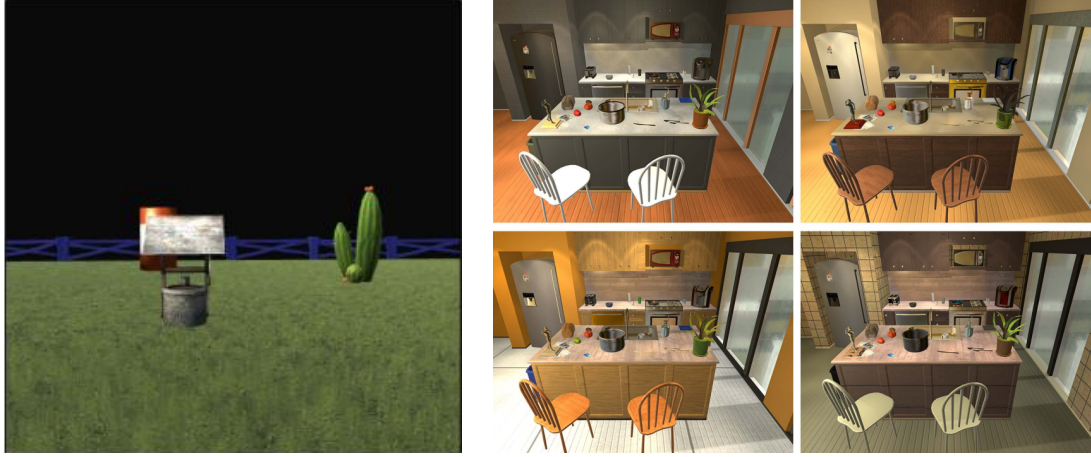
Examples of Continuous Space simulators: AI2-THOR - a simulation platform of near photo-realistic 3D indoor scenes, which uses front-end Python API and back-end Unity [21]. Another case is Habitat software stack, which contains a low-level Habitat-Sim and high level Habitat python library [22].

## 2.3 Scene Datasets

For the Agent to navigate in the indoor environment we would need to know the floor layout, furniture locations, color of walls, etc. Such an independent layout is referred to as a scene. A scene can be as small as a single room or it can be a whole building. Of course, we would like an entire dataset of various scenes for our training. There are available open-source scene datasets which can be separated into two categories: synthesised and real-life photos / real world reconstructions.

**Synthesised Scenes.** Synthesised scenes utilise work of 3D artists to manually create models and textures for each object type. This draws a lot of similarities with how scenes are created in video games. Synthesised scenes have wide appearance varying from simplistic to near photo-realistic. Objects possess high level of interactability: they can easily be manipulated, replaced

and even have their textures (materials, colours) switched - which gives an enormous degree of variance for data augmentation [21]. The downside is, naturally, not having a 100% visuals of the real world. This may cause trouble when transferring Agents from simulations into reality. AI2-THOR framework operates with a selection of synthesised scene datasets, which were specifically created for it: iTHOR, RoboTHOR, ProcTHOR and ArchitecTHOR [21].



(a) A render from a Lani dataset [23]. Notice that it is an outdoor scene

(b) AI2-THOR is able to randomise materials of objects in the scene, quickly augmenting data. [21]

Figure 2.4: Examples of synthesised scenes

**Real-world Reconstructions.** In Real-world Reconstructions scenes are based on the photos (RGB-D preferred) of real-world buildings. 3D meshes are then generated to match the existing counterpart. Such scenes undoubtedly represent reality, although the appearance is dependent on the quality of the photos used for reconstruction. The objects in the scene do not possess the same level of interactability as with the case of fully synthesised items, yet they still can provide a source of physical collision to an Agent. There are ways to augment such datasets. Tan et al. have introduced data augmentation technique called Environmental Dropout, where instances from a particular class of objects are masked off from a visual observation, as shown in Figure 2.5 [6]. Environmental Dropout introduces more variance among the scenes, so a trainable model could generalize more. An example of a photo-reconstructed scene dataset is

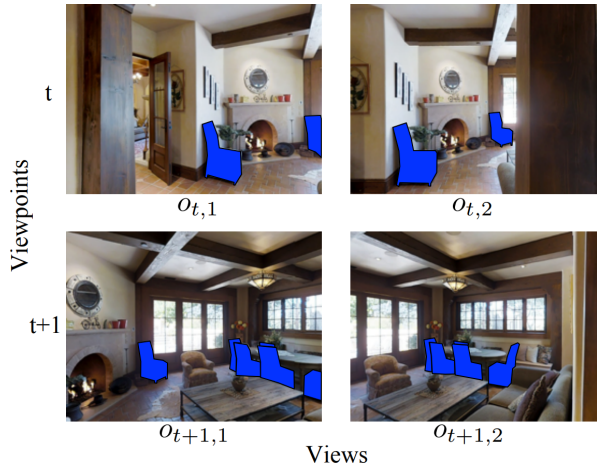


Figure 2.5: Environmental Dropout, where all chairs are masked-out from the view [6].



Matterport3D [24], which lay the basis for Matterport3D Simulator [4]. The dataset contains 194000 RGB-D images from 10800 panoramas. The images were taken using a Matterport camera mounted on a tripod inside 90 different buildings (in their indoor entirety) such as private homes, offices and churches. The dataset also provides textured mesh reconstructions of buildings documented (24727520 textured triangles). Additionally, captured objects and building elements possess instance-level semantic annotations from a list of 40 possible labels [24]

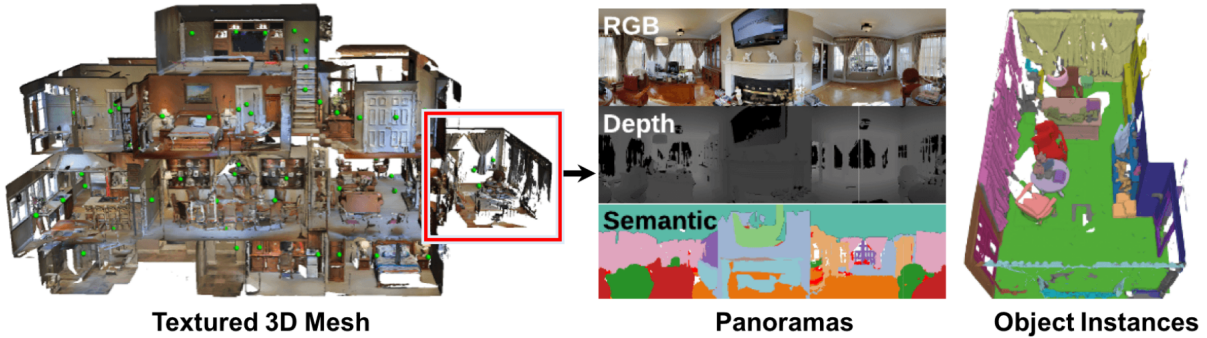


Figure 2.6: Examples of data contained within Matterport3D dataset. Image taken from Matterport3D GitHub.io project page [7].

## 2.4 Instruction and Path datasets

In the VLN task, language instruction is given to an Agent for navigation. Each instruction has a ground truth path, which the model learns to predict during the training. A dataset with input as language instruction and output as path is available for public use. Besides Matterport3D Simulator, a Room-to-Room (R2R) dataset was released to train models. The dataset contains 7189 paths, which traverse no less than 4 nor more than 6 edges in the related nav-graph, while also being at least 5m long (in terms of a continuous environment). Each path has 3 different navigation instruction, written by actual humans, resulting in 21567 instructions of an average word length of 29 words. An example of an instruction from R2R can be seen in Figure 2.2 (Lower-Right corner). Even though, the original R2R dataset was created for a Discrete Environment (Matterport3D Simulator), the research group has suggested the same dataset can be applicable in a Continuous Environment [4]. As part of a VLN-CE project, aimed at porting VLN task from Discrete Environment of Matterport3D Simulator into the realm of Continuous Environment created through Habitat, the R2R dataset was transferred into R2R\_VLNCE. In total 77% of original R2R paths were deemed navigable in Continuous Environment and were successfully transferred [9].

R2R dataset overtime has encounter several routes of development. R2R-EnvDrop was introduced, where along with Environmental Dropout exercised onto Matterport3D new paths were generated with corresponding instructions using automated speaker model [6]. Over 14000 new trajectory+instruction were created in such a manner. Later, these additions were ported to R2R\_VLNCE as well with minor pruning (109, 0.07%) [12]. A concern is raised for R2R dataset, that all paths are goal-oriented, i.e they follow the shortest possible. This is an aspect a model can learn to exploit, without learning language understanding and the instructions [25]. To address this, two datasets have spawned with longer and indirect navigation trajectories: Room-for-Room (R4R) [26] and Room-Across-Room (RxR) [27]. The latter was

also ported into a Continuous Environment [28]. an alternative take, the R2R datapoints are fractured into sub-instructions between each node pair along the paths creating a Fine-grained R2R dataset [29]. It is implied in a study by Raychaudhuri et al. that Fine-grained R2R is also applicable in Habitat’s environment [1]. One disadvantage of R2R datasets is their lack of object manipulation vocabulary. Since the Matterport3D scenes are static, an Agent cannot move or otherwise utilise encountered objects. This makes R2R family datasets usable only for the sake of learning navigation, i.e. learning to interpret instruction relating to Agent’s re-location. In contrast to the non-interactive instructions of R2R dataset, AI2-THOR 2.0 based Action Learning From Realistic Environments and Directives (ALFRED) utilises its dynamic simulator and provides instructions aimed at manipulating objects. Verbs such as *pick*, *open*, *close*, *cut* and *clean* are given to an Agent to execute. The dataset provides 120 scenes, 25743 language directives each describing one of the 8 055 action sequences, also referred to as expert demonstrations. These demonstrations can be deterministically replayed in the simulator [8].



Figure 2.7: An example of an instruction from ALFRED being carried out in AI2-THOR simulator as depicted with 6 captioned framed. [8]

## 3 Methodology

In this thesis we tackle the VLN task through implementations based on VLN-CE project with R2R\_VLNCE. Using a Continuous Environment brings the implementation closer to real-time environment. We start with simple navigation through shorter instructions and not considering other object manipulation.

### 3.1 R2R\_VLNCE\_v1-3

The latest version v1-3 of R2R\_VLNCE was released on February 3<sup>rd</sup>, 2022. The dataset is fractured into 6 splits and a file for word embeddings (50d GloVe [30]). Each split contains datapoints referred to as episodes and additional data regarding vocabulary used in instructions. Every split possesses a subset of 90 scenes from Matterport3D. Such subsets tell in how many different scenes (buildings) episodes within a split take place. These subsets may or may not overlap with each other. The splits are [12]:

- **train**: 10819 episodes, 61 scenes (overlaps with val\_seen, envdrop and joint\_train\_envdrop)
- **val\_seen**: 778 episodes, 53 scenes (both the episodes and scenes overlap with train, envdrop and joint\_train\_envdrop)
- **envdrop**: episodes generated for R2R-EnvDrop, 146304 episodes, 60 scenes (overlaps with train val\_seen, and joint\_train\_envdrop)
- **joint\_train\_envdrop**: fuse of train and envdrop splits, 157232<sup>1</sup>, 61 scenes
- **val\_unseen**: 1839 episodes, 11 scenes (no overlapping)
- **test**: used for VLN-CE Challenge [32], 3408 episodes, 18 scenes (no overlapping)

Each split consists of two *.json* files:  $\{split\}.json.gz$  and  $\{split\}_gt.json.gz$ , where *split* is the respective split's name. Files  $\{split\}.json.gz$  have two keys 'instruction\_vocab', which contains information for instruction processing, and 'episodes', which gives an array of individual episodes. Episodes are also dictionaries themselves. Let us examine how an episode's data is structured in Table 3.1.

In contrast, the structure of a  $\{split\}_gt.json.gz$  is much more simpler. Every key is an 'episode\_id' which ties the data between the two *.json* files together. To each key corresponds another dictionary of 3 keys, which can be studied in Table 3.2. One thing worth noting is that data from  $\{split\}_gt.json.gz$  file is not used in training. Instead, it is used in a few performance measurements.

---

<sup>1</sup>this number is used in the configuration file [31], which uses this split for training, however simple math of  $10819 + 146304$  yields 157123

| Key name         | Value type                                                                           | Example                                                                                                                              | Comment                                                                                                                                                                                                                                                                                                                                              |
|------------------|--------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 'episode_id'     | int                                                                                  | 1                                                                                                                                    | Identification number for the episode for ease of reference                                                                                                                                                                                                                                                                                          |
| 'trajectory_id'  | int                                                                                  | 4                                                                                                                                    | Identification number for the trajectory of the path for ease of reference.<br>Reminder that a single trajectory can be used in multiple episodes with different instructions                                                                                                                                                                        |
| 'scene_id'       | str                                                                                  | 'mp3d/7y3sRwLe3Va/7y3sRwLe3Va.glb'                                                                                                   | A reference to a 3D mesh file of a scene (building) from Matterport3D dataset, where the episode takes place                                                                                                                                                                                                                                         |
| 'instruction'    | dictionary with two keys: 'instruction_text' and 'instruction_tokens'                | { 'instruction_text': 'Go around the right side...', 'instruction_tokens': [982, 141, 2202, ..., 0, 0, 0] }                          | The instruction for an Agent to perform during this episode. 'instruction_text' gives a human-readable text of the instruction, whereas 'instruction_tokens' provides an array of machine-interpretable values                                                                                                                                       |
| 'start_position' | an array of 3 float                                                                  | [-16.267200469970703, 0.1518409252166748, 0.7207760214805603]                                                                        | The global X, Y, Z coordinates in the scene, from where an embodied Agent begins the episodes, also the starting point of the trajectory                                                                                                                                                                                                             |
| 'start_rotation' | an array of 4 float                                                                  | [0.0, 0.0007963267107332633, 0.0, 0.9999996829318346]                                                                                | Global rotation of an embodied Agent at the beginning of the episode, represented in quaternion                                                                                                                                                                                                                                                      |
| 'goals'          | an array of dictionaries, each dictionary contains two keys: 'position' and 'radius' | { 'position': [-12.337400436401367, 0.1518409252166748, 4.213699817657471], 'radius': 3.0 }                                          | There is usually only a single goal, therefore the array usually has a single dictionary. A goal is represented by a point in a scene.<br>Key 'position' is the global X, Y, Z coordinates of the point in the scene. Key 'radius' represents how close an Agent must come to the point for the goal to be considered reached, i.e. episode finished |
| 'reference_path' | an array, where each element is an array of 3 float                                  | [[[-16.267200469970703, 0.1518409252166748, 0.7207760214805603], ..., [-13.907199859619140, 0.1518409252166748, 4.2282099723815920]] | This is the sequence nodes of a nav-graph, which are visited by a Discrete Environment Oracle when it goes towards the goal.<br>In other words - ground truth for Discrete Environment Agent in an analogous episode.<br>Each node in the sequence is represented with its X, Y, Z global coordinates in the scene                                   |
| 'info'           | a dictionary                                                                         | { 'geodesic_distance': 6.425291538238525 }                                                                                           | Additional information, which can be used by simulator's sensors                                                                                                                                                                                                                                                                                     |

Table 3.1: A structure of a  $\{split\}.json.gz$  file's episode, which is a datapoint in R2R\_VLNCE dataset. [12]

| Key name        | Value type                                          | Example                                                                                                                              | Comment                                                                                                        |
|-----------------|-----------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|
| 'actions'       | an array of ints                                    | [2, 2, 2, ..., 0]                                                                                                                    | Action sequence of an Agent to perform in order to get from starting position to goal in shortest path         |
| 'forward_steps' | int                                                 | 27                                                                                                                                   | Number of "FORWARD" actions which an Agent takes throughout the shortest path from start to goal               |
| 'locations'     | an array, where each element is an array of 3 float | [[[-16.267200469970703, 0.1518409252166748, 0.7207760214805603], ..., [-12.644463539123535, 0.1518409252166748, 4.2241311073303220]] | A sequence of points, which an Agent reaches after each action throughout the shortest path from start to goal |

Table 3.2: A structure of values inside  $\{split\}.gt.json.gz$  file from R2R\_VLNCE dataset. Example is given based on the resulted dictionary from key '1'. [12]



## 3.2 VLN-CE

Visual-and-Language Navigation in Continuous Environments (VLN-CE) is one of the projects spearheading the idea of moving away from training VLN Agents in Discrete Environments for the sake of better representing reality to an Agent with the use of Continuous 3D Environments. It was released in spring of 2020 along with a related paper by Krantz and Wijmans, et al [9]. At the time of this thesis writing, the VLN-CE project provides:

- two Instruction and Path datasets with their corresponding challenges:
  - R2R\_VLNCE dataset [12] and VLN-CE Challenge [32]
  - RxR\_VLNCE dataset and RxR-Habitat Challenge [28]
- a GitHub repository [33] with starting code containing:
  - 3 Neural Network (NN) models for Agent’s decision making, implemented with PyTorch [34]
  - algorithms for training, evaluation and inference of VLN Agents using provided datasets, written in Python 3.6 [35]
  - a multitude of configuration files with specific algorithm and NN model settings

The project’s code is built on top of Habitat software stack, more specifically on version 0.1.7. Developed by Facebook Research AI, by the time of this thesis writing, Habitat 3.0 (i.e. version 0.3.0) has already been released in late October of 2023. Habitat stack itself consists of two repositories: low-level Habitat-Sim [36] and high-level Habitat Lab [37]. Most development happens by extending *habitat* package from Habitat Lab, which addresses Habitat-Sim as the core simulator [37]. We take a note, that the code inside the mentioned repositories is quite convoluted, with a considerable amount of vague classes/abstractions, with little to no documentation (at least as far version 0.1.7 is concerned). Habitat Lab does provide an *examples* package, yet a potential developer must manually go through thousands lines of code to study the applications and ways of improving the stack. The body of an Agent is physically represented by a 1.5m tall cylinder with a 0.2m diameter. The body has a single, forward-mounted RGB-D camera, with a  $256 \times 256$  pixel resolution and  $90^\circ$  horizontal field-of-view [9]. The body emulates a ground-based, zero-turning radius robot analogous to LoCoBot [38]. Initially, the Agents movement is constrained with 4 low-level actions: *move-forward* 0.25m, *turn-left* or *turn-right*  $15^\circ$  or *stop* action, which is invoked when an Agent considers the goal position to be reached [9]. Such actions keep the action space simple and concise enough, while also being replicable in real-life. If necessity arises, the action space model can be changed, since the underlying Habitat-Sim is quite flexible with movement definitions. Another aspect of movement is the absence of noise, meaning when an Agent attempts to *move-forward*, it will always move 0.25m (if there is no collision occurring of course). [9]

### 3.3 Task description

An Embodied Agent is given an episode from R2R\_VLNCE\_v1-3 dataset [12] to accomplish. The episode places the Agent inside a simulated scene of Continuous Environment, which is one of the virtually reconstructed buildings documented in Matterport3D [24]. The Agent inside the scene is described by a current state, consisting of Agent’s current position and rotation within the environment. The episode specifies the starting state of the Agent, as well as goal position with a radius, which creates a sphere where the Agent must reach at the end of the episode. The episode also provides a natural language (NL) instruction describing how to arrive at the goal. For the Agent, the environment is only partially observable: the Agent is unaware of its exact state in the scene, neither is it aware of the goal location; the Agent can perceive the scene through an egocentrically mounted RGB-D camera. The Embodied Agent can move through the scene with a set of 4 basic actions: *move-forward* 0.25m, *turn-left* or *turn-right*  $15^\circ$  or *stop* action. The actions are encoded with integers, such that: 0 stands for *stop*, 1 - *move-forward*, 2 - *turn-left*, 3 - *turn-right*. The scene is static. The Agent’s body can collide with the objects in the scene, but cannot move, manipulate or damage them. The Agent possesses a navigation algorithm, referred to as policy, which it uses as a decision-making tool for reaching the goal. Throughout an episode, at each time-step the Agent receives a set of observations: an RGB-D image from its camera in regards to Agent’s current state; an NL instruction remaining constant throughout the episode. The Agent can use the observations and knowledge of its previous actions to feed them to the policy to make optical decision to choose best action (one out of 4 available) for reaching its goal. The whole navigation is represented as a sequence of decision-making and action tacking. The episode terminates either when the maximum number of time-steps is reached or when the Agent executes a *stop* action, meaning that the Agent considers having arrived at the goal.

## 3.4 Policy Development

The crucial aspect of VLN Embodied Agent creation is the development of Agent’s policy. VLN-CE project proposes using Machine Learning (ML) as the main method and Deep Neural Networks (DNN) as the implementation basis. ML models require training and evaluation procedures, which the VLN-CE project provides as previously mentioned. The project utilizes Imitation Learning, where the ML model learns to imitate an Oracle - an algorithm able to make the best decisions at each time step by the use of data unavailable to the Agent. The training procedure are divided into two stages: training data generation and model updating. To see the description of the first stage, along with the classes in use, please look at Figure 3.1.

The code for the second training stage (model updating) is located in the same Trainer Class. Previously created LMDB is read in batches of per episode grouped data points. The loss value is calculated as a sum of losses throughout every time-step in an episode recorded. *Agent’s sensor observations* along with their corresponding *previous actions* are fed to the policy to get predictions. Predictions are compared with Oracle sensor’s created *correct next actions* and with cross-entropy loss function the model receives iterative updates (specifically `torch.nn.functional.cross_entropy` [40] is used).

This is the very basic training procedure. VLN-CE offers a few additional optional improvements:

### 3.4.1 Inflection Weights

The model can focus on mimicking long repeated sequences of the same action and miss a necessary change of operation. For example, the model outputs a constant stream of *move-forward* actions and fails to make a turn in time. To promote timely decision making Inflection Weighting was introduced. The aim is to increase the learning impact of a time-step, when inflection took place along the correct path (i.e.  $action_{t-1} \neq action_t$ ). Such time-steps, named inflection points, during loss calculation are multiplied by an additional weight  $N/n_i$ , where  $N$  is the total number of time-steps present in the whole set of correct paths and  $n_i$  is the total number of inflection points, meaning the Inflection Weight is an inverse frequency of inflection points in the dataset [41]. Fortunately, VLN-CE project provides a pre-computed Inflection Weight to go along with their dataset.

### 3.4.2 Progress Monitor

An accomplished episode is the episode, where the Agent issued a *stop* action within the success radius of the goal point. We encourage such behavior in our model. One way is to explicitly supervise the model with a normalised distance from the Agent to the goal. The normalised distance is measured as 0 at the starting point and becomes closer to 1 the closer the Agent gets towards the goal. Normalised distance can be a negative number if the Agent moves away from the goal further than the starting point. The supervision is performed as an auxiliary loss. The model is modified to also estimate the mentioned distance. At each time step the real and estimated normalised distances are put in a mean squared error, summed up across an episode, and used alongside the cross-entropy error resulting from supervising action decision [42].

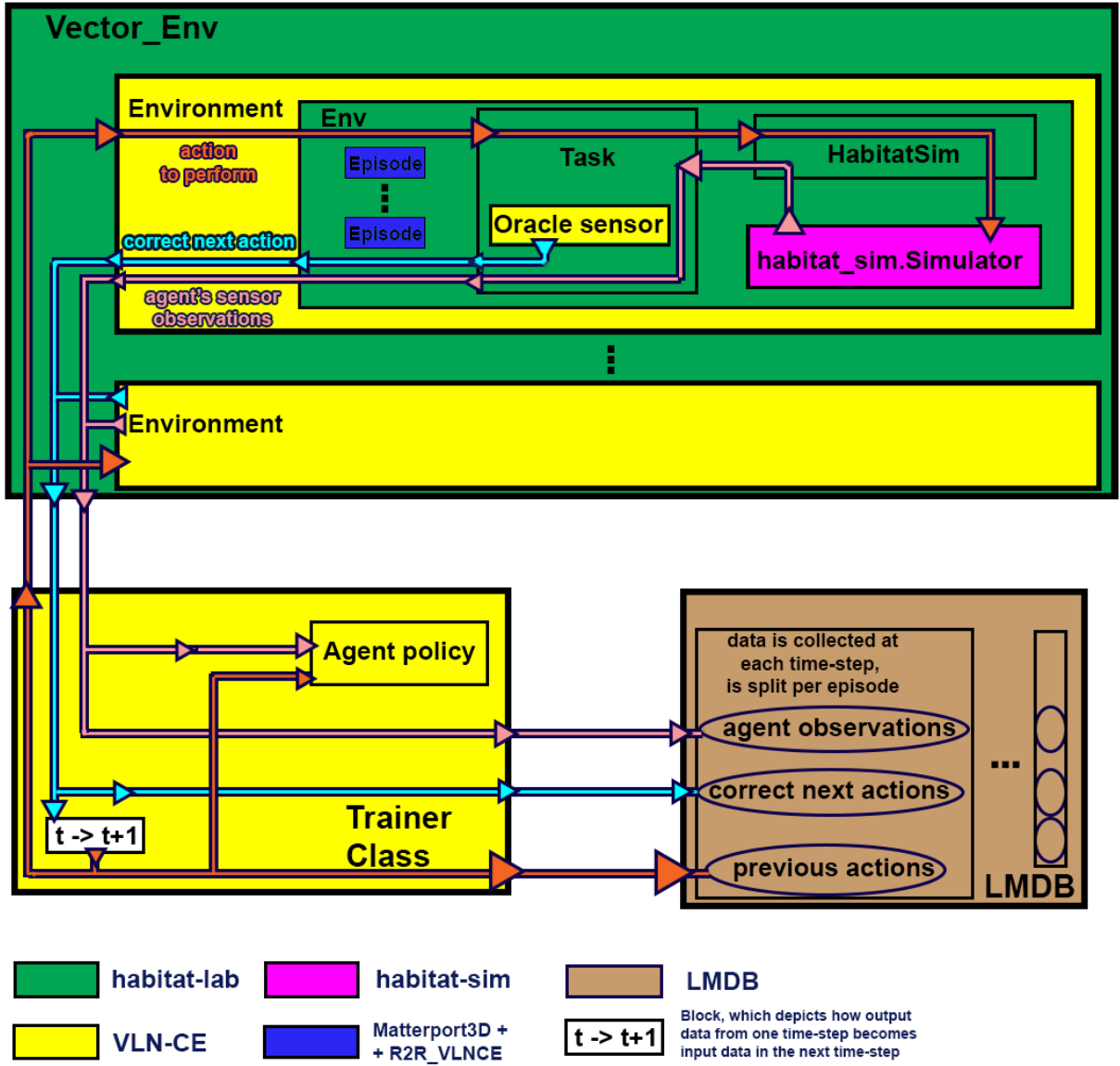


Figure 3.1: A scheme for Agent policy training stage 1: training data generation. The scheme depicts the data flow and the main class objects involved (multiple other class instances are left unmentioned for the sake of clarity). Class objects are represented with rectangles and their background colour shows to which project/repository/library they belong. The dataflow is shown with coloured lines and arrows(triangles) give output/input direction.

The main purpose of this stage is to store at each time-step a datapoint, which contain an agent observations (RGB-D image + Instruction); action, performed in a previous time-step; a correct action to perform at this time-step. All datapoints are grouped by episodes and are kept in a chronological order. The generated data is stored in a Lightning Memory-Mapped Database (LMDB) [39] for later use in stage 2.

The program can run several simulations in parallel by creating several Environments, which run synchronously within Vector\_Env class object.

At each timestep the Trainer Class receives *correct next actions* and *agent's sensor observations* for each instance of Environment. The Environments each in turn receive a new action to make an Agent to perform in order to generate a new set of *agent's sensor observations* and *correct next actions*. The *correct next action* in one time-step becomes *action to perform*, as well as *previous action* in the next one. Note, that Agent policy is fed the necessary inputs, however its output in this simple case is not used anywhere.

### 3.4.3 Dataset Aggregation

During training, the Agent executes only the correct actions dictated by the Oracle sensors. Therefore the observations are confined purely to the ground truth path. Thus the Agent is never exposed to the full extent of available positions in the Continuous Environment with their respective observations. In addition, during tests or evaluations, a single mistake can cause the failure of an entire episode. Agent’s policy has never experienced an incorrect action and never learned to recover. To address this the Dataset Aggregation (Dagger) procedure is implemented in Stage 1 of training. The training is carried out in several analogous DA Iterations, where at each iteration we perform both of the described stages; all generated training data is accumulated throughout the DA Iterations; during training data generation there exists a chance (it starts as 0 on the first DA Iteration, but eventually grows to 1), that the *correct next action* is replaced with an action predicted by the Agent policy at the current development state (this is why the policy receives the input data at each time-step). The replaced action is likely to differ from the Oracle’s expertise, therefore the resulting path will deviate from the ground truth path [43]. The pseudo-code of Dagger in VLN-CE is given:

```
#Initialise starting values from configuration file.
 $P \leftarrow [0;1]$ ;
 $N \leftarrow \mathbb{N}^+$ ;
 $policy_0.init()$ ;
 $LMDB.begin()$ ;

#Begin loop of DA Iterations.
for  $n$  in  $0 \dots N$  do
    #Set up exponentially decaying  $\beta$ 
     $\beta \leftarrow P^n$ ;

    #Begin loop of modified training stage 1.
    for each time-step in each episode do
        correct next action  $\leftarrow$  Oracle Sensor output;
        observation  $\leftarrow$  Simulator;
        predicted next action  $\leftarrow policy_n.predict(observation)$ ;

        With likelihood  $P(\beta)$ : action to perform  $\leftarrow$  correct next action;
        else: action to perform  $\leftarrow$  predicted next action;

         $LMDB.add(observation, correct\ next\ action)$ ;
        Simulator  $\leftarrow$  action to perform
    end

    #Perform training stage 2.
     $policy_{n+1} \leftarrow policy_n.update(LMDB.data)$ 
end
```

**Algorithm 1:** Pseudo-code for training procedure when Dagger is involved. DA Iterations can be viewed as entire training runs, which share a single LMDB, that gets continuously updated over time. Meaning that the last policy is being trained on every trajectory created. We can see that the first DA Iteration functions exactly like a standard training procedure due to  $\beta = 0$  (i.e. only the correct actions are performed).

Note that for the sake of clarity operations involving *previous actions* are removed from this pseudo-code.

### 3.5 Cross-Modal Attention Model

After observing the data and the procedures for policy training, we can move on to examining the actual ML model, which will be used as the Agent’s policy. VLN-CE repository provides 3 models. This thesis focuses on the Cross-Modal Attention (CMA) model, which is deemed as the baseline model for VLN-CE Challenge [33]. The model takes as input separately RGB and Depth images, tokenized instruction and previous action ( $a_{t-1}$ ) to predict the next action ( $a_t$ ) as shown in Figure 3.2.

**Vision Encoding.** RGB image (a  $256 \times 256 \times 3$  array) is encoded using ResNet50 [44] pretrained for image classification task based on ImageNet (a huge database of labeled images) [45], which gives the basis for detecting semantic visual features. Depth image (a  $256 \times 256$  array) is also encoded with ResNet50, although modified for point-goal navigation tasks. This task shares similar aspects to our VLN task: an Embodied Agent is placed at a random location in a previously unknown scene (Continuous Environment) and has to navigate to the specified goal using egocentric vision, however instead of an NL instruction the Agent possesses a GPS+Compass sensor, which guides the Agent to the goal [46]. The visual encodings are processed twice to be fed separately to GRU and Visual Attention blocks.

**Instruction Encoding.** The NL instruction string is fractured into an array of fixed size, where each positive integer represents a machine-comprehensible token (as depicted in Table 3.1, ‘instruction’ line). Tokens are then replaced with corresponding GLoVe embeddings [30], which transforms the array into a sequence of 50D vectors. Values within the vectors denote the semantic similarity between the words. These vectors are created on purely text, i.e. based on co-occurrence in a huge text corpus; images, sounds, etc related to the words were not used. The resulting array is then passed onto the bi-directional Long Short-Term Memory [47] encoder.

**GRU.** The model has two Gated Recurrent Units (GRUs, also known as RNN Encoder-Decoder) [48], which grant the model memory aptitude. The first one (left-most on the scheme) is responsible for processing visual information. The second one makes the final decisions based on all previous computations. Note, that in the actual implementation the second GRU does not output the predicted action directly. Instead it outputs 512 features, which are then fed to a Fully Connected layer outputting 4 features (corresponds to the number of possible actions), which in turn flow into a modified Categorical Distribution class from PyTorch [49] in order to finally retrieve the predicted action.

**Attention Blocks.** The model uses Attention to better focus on the task execution at a given time and circumstances. It is implemented as a scaled dot-product attention:

$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$ , where  $Q$ ,  $K$  and  $T$  are vectors (or sets of vectors packed together as matrices),  $d_k$  is the dimension of both vectors  $Q$  and  $K$ . [50]

Instruction Attention takes  $Q$  based on the output of the first GRU,  $K$  and  $V$  are based on instruction encodings. There are separate Visual Attention Blocks for RGB and Depth. A Visual Attention takes  $Q$  based on the Instruction Attention output,  $K$  and  $V$  are based on corresponding visual encodings passed through a 1D Convolutional layer.

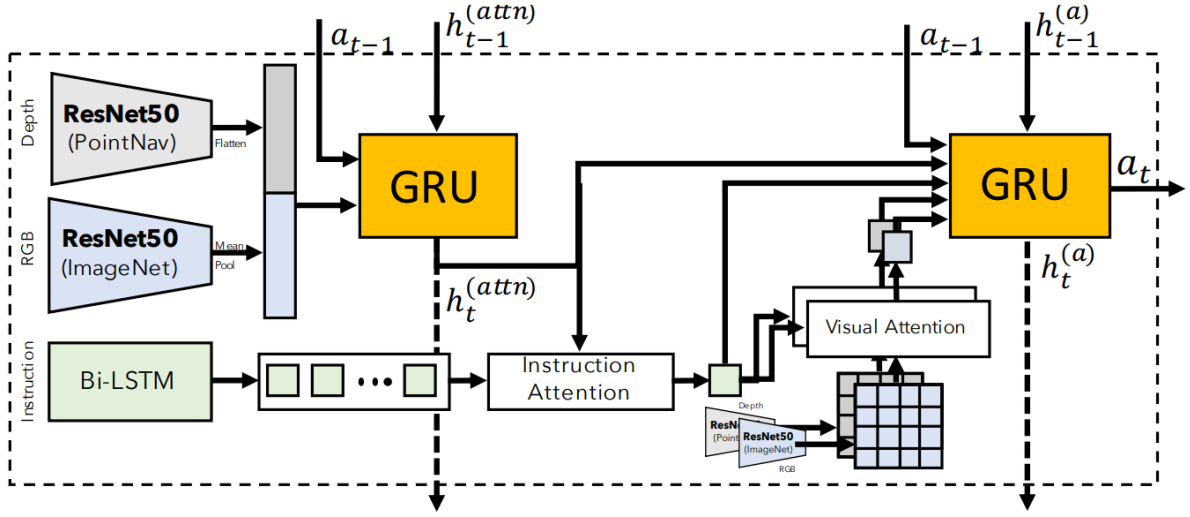


Figure 3.2: A simplified scheme representing the general architecture of Cross-Modal Attention Model. Image taken from the original paper [9]. See section 3.5 for descriptions.

### 3.6 LAW-VLNCE

Supervision of correct actions during training in VLN-CE code is carried out by a Greedy Geodesic Follower [51], which creates the shortest path toward the goal. The shortest path, however, does not always constitute correctly following a given instruction, as depicted in Figure 3.3. Such inconsistencies may be detrimental to policy learning to understand NL. To solve this problem a fork project from VLN-CE was created named LAW-VLNCE, where LAW stands for Language-Aligned Waypoint. It introduced a new sensor for the use as an Oracle. This sensor tries to sequentially lead an Agent along the intermediately placed waypoints, before leading towards the goal. These waypoints are either taken from the ground truth file ( $\{split\}_{gt.json.gz}$ , 'locations' key from the respective episode instance) or from the 'reference path' specified in the episode ( $\{split\}_{json.gz}$ ). Usage of LAW paths in training leads to policy better comprehending NL as hinted by increased performance compared to the standard VLN-CE. [1]

### 3.7 Evaluation Metrics

There are several measurements, that can describe how successful an Agent (or more specifically its policy) at solving VLN tasks. Here is a list of them (the same were used in LAW-VLNCE original paper [1]):

- **Trajectory Length (TL) or Path Length:** length of a path, which the Agent traversed during an episode.
- **Navigation Error (NE):** distance to goal from the Agent at the end of an episode.
- **Success Rate (SR):** rate of an Agent finishing an episode within a specified radius from the goal (3m usually).
- **Oracle Success Rate (OS):** rate of an Agent entering within a specified radius from the goal (3m usually) at any point in time.

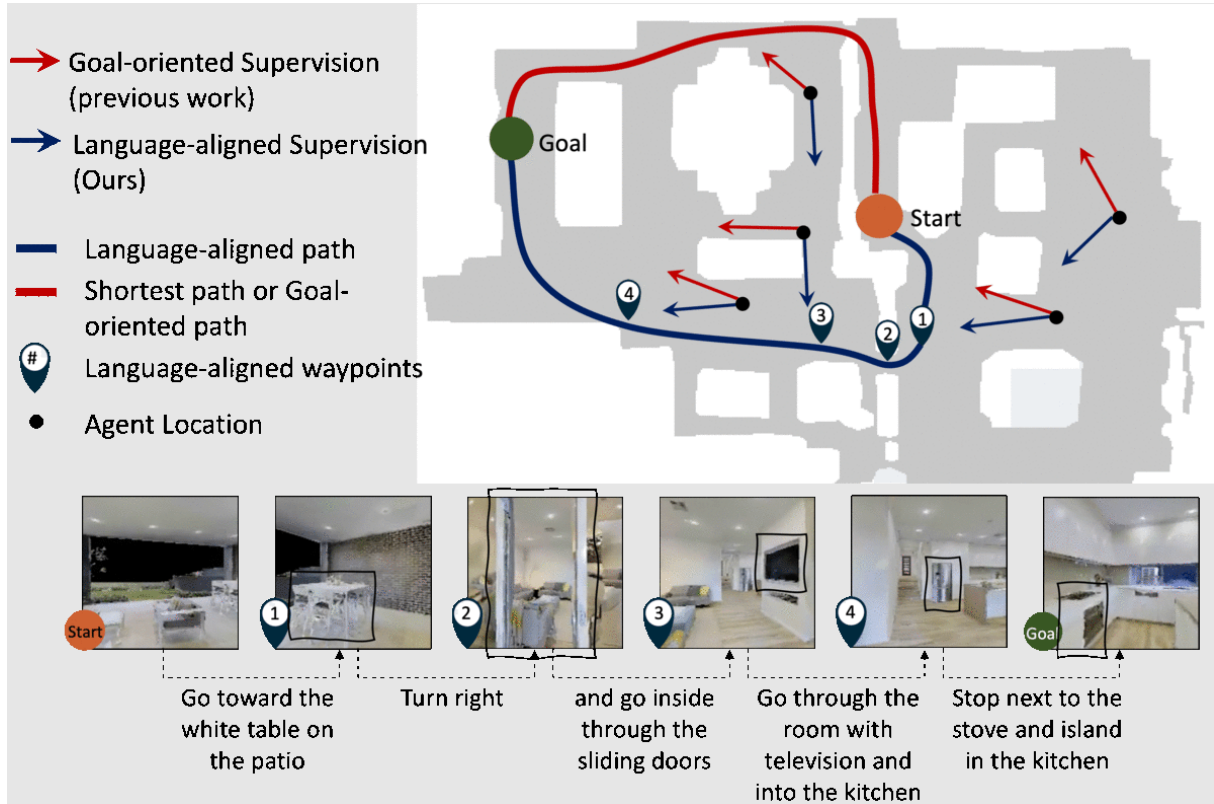


Figure 3.3: Difference between a path created with a sensor, which navigates towards the goal position in a shortest way possible (red), and a path which follows intermediate waypoints, which are placed in respect to the NL instruction (blue). Additionally, several Agents are spread throughout a scene with coloured arrows showing the direction, in which the Agent would move listening to commands of each sensor. Image is taken from [10].



- **Success weighted by inverse Path Length (SPL):** success weighted by trajectory length relative to shortest path trajectory between start and goal. Computed over the whole evaluated set as  $\frac{1}{N} \sum_{i=1}^N S_i \frac{l_i}{\max(p_i, l_i)}$ , where  $N$  is the number of evaluated episodes in the set,  $S_i$  is the binary indicator for success of an episode  $i$ ,  $l_i$  is the shortest path distance from the start to goal in an episode  $i$ ,  $p_i$  is the path Trajectory Length of episode  $i$  [52].
- **Normalized dynamic-time warping (nDTW):** evaluates on a scale from 0 to 1 how the Agent’s trajectory matches with the ground truth, where 1 means both trajectories are identical [53].
- **Success weighted by nDTW (SDTW):** same as nDTW, however it is considered a 0, when the episode was unsuccessful [53].
- **Waypoint Accuracy (WA):** fraction of Language Aligned Waypoints the Agent has correctly visited in its path, the Agent must come close to at least 0.5 meters from a waypoint for it to be considered visited [1].

### 3.8 Proposed Method

We, humans, have developed an intuition for navigating indoors. When a person hears: *”Go down the corridor and enter the third room”* - even without seeing the environment, they can already expect to walk some time forward through the corridor, there will be doors on the side-walls since doors connect rooms, and eventually they are going to turn either left or right to face one of the doors and once again step forward to enter a room. Although we cannot say exactly how far must one go forward or at what angle to turn, we already built up an idea of a possible path. And this idea gets more and more fleshed out as the person starts seeing the corridor with the doors. We are able to create such ideas initially, because we are accustomed to a context, which is present among the indoor environments and interior furniture: a toilet is usually placed in the bathroom, desk lamps reside on a table, corridors lead to doors and doors to other rooms. Another thing, since the whole instruction is provided at once, as shown in the previous example, information about the layout of the environment ahead of us can be extracted from the NL instructions. Moreover, one can imagine their own state within the environment in the future. Vision only bolsters intuition, when, for example, one catches glimpses of other rooms through doorways or corridors.

Similar intuition we would like to create within the policy of an Embodied Agent. And we would like to create it explicitly. We should actively encourage the trainable model to think about the states currently unobservable and its operations in the future. We can implement the development of intuition by upgrading the model to predict more than just an *action to perform*. We’ve already seen such an example with the addition of Progress Monitor, where the model develops intuition of *”how far away the Agent is from the goal”*. Another example can be brought from a separate study by Song, et al [54], where a model from a single depth image simultaneously predicted occluded 3D shapes of observed object, while also giving each object semantic labeling. As a result this model outperformed all other models, which tried to do those two tasks separately. Another perk of having additional explicit outputs is, that we can receive more hints of the model’s reasoning during application.

In the scope of this thesis the explicit intuition will be constructed by updating the model to predict not only one *action to perform* for current time-step, but for several following time-steps

ahead. In other words: at each time-step  $t$  from an input of  $input_t$  the model will output a set  $\{\hat{a}_t, \hat{a}_{t+1}, \dots, \hat{a}_{t+n}\}$ , where  $\hat{a}_t$  is an action, which the model predicts the Agent to perform at the end of time-step  $t$ . By supervising the model predicting actions of near future time-steps, we hypothesize that the model will "look ahead", when making each decision, positively impacting the performance.

Modifying the baseline model was not a big issue, it was a simple change of the output size of fully-connected layer, which comes after the second GRU block. The size was changed from *number\_of\_possible\_actions* (4) to *number\_of\_possible\_actions*  $\times$  *number\_of\_intuition\_steps*, where *intuition\_steps* mean the time-steps for which the model makes predictions. The question now arises: how to create sets of corrected actions at each time-step, including the cases when DAgger is used, and considering possible collisions. We can utilise an already existing Oracle sensor and habitat-sim. Two methods were proposed:

1. At each time-step during training stage 1 the Agent's state (position and rotation in the environment) is saved, then for a number of additional intuition time-steps the Agent follows the commands of the Oracle sensors, records the actions taken and then the Agent is teleported back to the original saved state.

**Pros:** easy to implement; habitat-sim takes on all the computations for tracking the Agent's state and collisions; can utilise noisy actuation, if necessity arises.

**Cons:** every additional intuition time-step will generate new Agent observations, wasting computational time.

2. We create another Embodied Agent, which is a copy of the first one, although it is blind, i.e. it lacks camera. We refer to this second Agent as Geist. This method starts out similar to the previous one, but the during the intuition time-steps the movement is performed only by Geist.

**Pros:** No wasted computations for generating unused visual observations; habitat-sim supports multiple Agents; Agents do not collide with each other and are not rendered in camera images.

**Cons:** HabitatSim and Task from habitat-lab do not support multi-Agent operations out-of-the-box.

Out of the two methods, the second one was selected due to the prospect of faster performance. Additionally, the LAW sensor from LAW-VLNCE was chosen for the position as the Oracle sensor.

### 3.9 Design of Experiments: INTUIT-VLNCE

The code for our approach resides in a GitHub repository named INTUIT-VLNCE [55]. It is a fork project of LAW-VLNCE [10] (which it is in turn a fork of VLN-CE [9]). The name INTUIT-VLNCE takes on the naming conventions of previous projects dealing with VLN tasks in a Continuous Environment. The INTUIT part represents our strive to develop explicit intuition, while also containing an easter egg reference to the Author's institute by containing its acronym (Tartu University Institute of Technology - TUIT). This project requires independent installation of habitat-lab [37] and habitat-sim [36] both of versions v0.1.7. The development of the code was performed on the author's PC.

The changes this repository introduces in short are:

- **HabitatSimDual** - a child class of HabitatSim from habitat-lab, which is capable of creating a second Agent without vision (Geist) and is able to issue commands to them.
- **NavigationTaskDual** - a child class of NavigationTask from habitat-lab, able to operate with at least two Agents (was not tested with more than two Agents).
- **VLNCEDaggerIntuitionEnv** - an environment class used for training. When an action is received, the Geist is teleported to the position of the main Agent, then the Geist generates a sequence of correct actions for several time-steps ahead by moving towards the goal listening to the Oracle sensor (LAW sensor to be specific). After the sequence of correct actions is generated, the input action is sent to the main Agent to execute, regardless of what the Geist has generated. This allows to use DAgger, when the Agent must sometimes perform incorrect actions.
- **Intuition Policy** - a modified version of CMA model from the original VLN-CE paper [9], which predicts several actions for the current and future time-steps.
- **DaggerLawIntuitionTrainer** - a trainer class, which is able to utilise all of the additions above, along with DAgger. Based on the trainers from VLN-CE and LAW-VLNCE projects.
- several configuration files for training and evaluation parameters:  
intuition\_config/cma\_pm\_aug.yam, intuition\_config/cma\_pm\_da\_tune.yam,  
vlnce\_task\_aug\_intuit\_pano.yaml and vlnce\_task\_intuit\_pano.yaml.

For every instance of training we use Progress Monitor and Inflection Weights. We calculate loss in batches of episodes, which contain recorded time-steps. The entire loss function can be written down as:

$$\begin{aligned}
Loss(X, Y, \hat{P}^m, P^m) &= Loss_{action}(X, Y) + Loss_{PM}(\hat{P}^m, P^m) \\
Loss_{action}(X, Y) &= -\frac{1}{I} \sum_{i=0}^I \sum_{b=0}^B \frac{1}{\sum_{t=0}^T \omega_{b,t}} \sum_{t=0}^T \omega_{b,t} \log \frac{\exp(x_{t,b,i}, y_{t,b,i})}{\sum_{c=0}^C \exp(x_{t,b,i}, c)} \\
Loss_{PM}(\hat{P}^m, P^m) &= -\frac{\alpha}{N_b} \sum_{b=0, t=0}^{B, T} (\hat{p}_{b,t}^m - p_{b,t}^m)^2 \\
\omega_{b,t} &= \begin{cases} \omega, & \text{inflection point} \\ 1, & \text{otherwise} \end{cases}
\end{aligned}$$

- $T$  - number of time-steps in an episode ( $T_{max}$  number of time-steps of the longest episode in batch)
- $B$  - Batch size, number episode in a batch
- $I$  - number of intuition-steps, how many actions does a model predict
- $C$  - number of possible actions
- $N_b$  - number of inflection points present in an episode
- $X$  - matrix containing unnormalised logits of action predictions of size  $T_{max}, B, I, C$

- $Y$  - matrix containing correct actions of size  $T_{max}, B, I$
- $\hat{P}^m$  - matrix containing PM predictions of size  $T_{max}, B$
- $P^m$  - matrix containing correct PM values of size  $T_{max}, B$
- $\omega_{b,t}$  - inflection weight at time-step  $t$  in an episode  $b$
- $\omega$  - hyper-parameter for weights at inflection points
- $\alpha$  - hyper-parameter for scaling PM loss

For action loss we essentially compute loss for each place of predicted action in intuition action sequence at every time-step in every episode and then averaging them. Additionally, Adam optimiser [56] is utilised in every instance of training just like in the VLN-CE implementation [9].

### 3.10 Training and Evaluation

All training and evaluation were carried out with the assistance of Tartu University’s High Performance Computing (HPC) Center, which provided access to Rocket HPC cluster [57]. HPC services have a monetary cost related to them [58]. Luckily HPC Center provides free credit of 1000 Euros to the students from the University of Tartu each year. The Author was not interested in exceeding this limit, therefore it also affected the selection of the VLN-CE task and R2R dataset. The cluster consists of nodes, which the program must request for the sake of computational resources. Every training or evaluation run has requested one of 6 GPU nodes from the Falcon series. The hardware belonging to those nodes is:

- 2 x Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz (48 cores total)
- 512 GB RAM
- 5TB of local SSD storage
- Infiniband:
  - Falcon 1-3 – 2x 40 Gbps each
  - Falcon 4-6 – 5x 100 Gbps each
- 24x NVIDIA Tesla V100 GPUs:
  - Falcon 1-3 versions have 16 GB of VRAM
  - Falcon 4-6 versions have 32 GB of VRAM

[57]

The training of our policy follows the training performed in the paper for LAW-VLNCE [1]. It is divided into main training phase and fine tuning phase.

### 3.10.1 Main Training Phase

In this phase we initialise the model and train it on the **joint\_train\_envdrop** split of R2R\_VLNCE\_v1.3. During the training PM and IW are used. Oracle sensor is LAW-sensor. At the end of each training epoch, the model's current state is saved as a checkpoint. After the training, all checkpoints would be evaluated on **val\_seen** splits (to see how well the model learned from the training data) and then on **val\_unseen** to measure actual performance. Evaluations will use metrics, discussed in section 3.7 along with the average number of time-steps taken, out of which **ndtw** is considered the most important one. The best performing checkpoint in terms of **ndtw** will be used in the fine tuning phase.

Related configuration and SLURM files:

- *vlnce\_baselines/config/paper\_configs/intuition\_config/cma\_pm\_aug.yaml*
- *habitat\_extensions/config/vlnce\_task\_aug\_intuit\_pano.yaml*
- *trainjob.sh*
- *evaljob.sh*

Important hyperparameters:

- **Maximum time-steps per episode:** 7500 during training (to account for extra steps performed in the VLNCEDaggerIntuitionEnv), 500 during evaluations
- **Success Distance:** 3 meters
- **Number of simultaneous Environments:** 8
- **Number of training Epochs:** 45
- **Batch Size:** 5 (how many episode within a batch for loss calculation)
- **PM  $\alpha$ :** 1.0
- **IW  $\omega$ :** 3.2 (as was calculated in [9])
- **Number of intuition Steps:** 15
- **Learning Rate:**  $2.5e-4$  (as was used in previous papers [9])

Training and evaluation on **val\_seen** took place on Falcon 5 node, whereas evaluation with **val\_unseen** on Falcon 2.

Note: During the training stage 1, the program encountered an error after processing 47% of **joint\_train\_envdrop** episodes. The author considered it to be a minor issue, because a seemingly sufficient number of episodes were processed: 73581 out of declared 157232.

### 3.10.2 Fine tuning Phase

The best performing checkpoint (**nDTW** metric) was selected as a starting ground. During this training LAW-sensor, PM and IW were used again. In addition DAgger is activated as well. In this phase 5000 episodes from **train** split are used. As previously, a checkpoint is saved at the end of every Epoch, and all checkpoints are evaluated identically to the previous phase.

Related configuration and SLURM files:

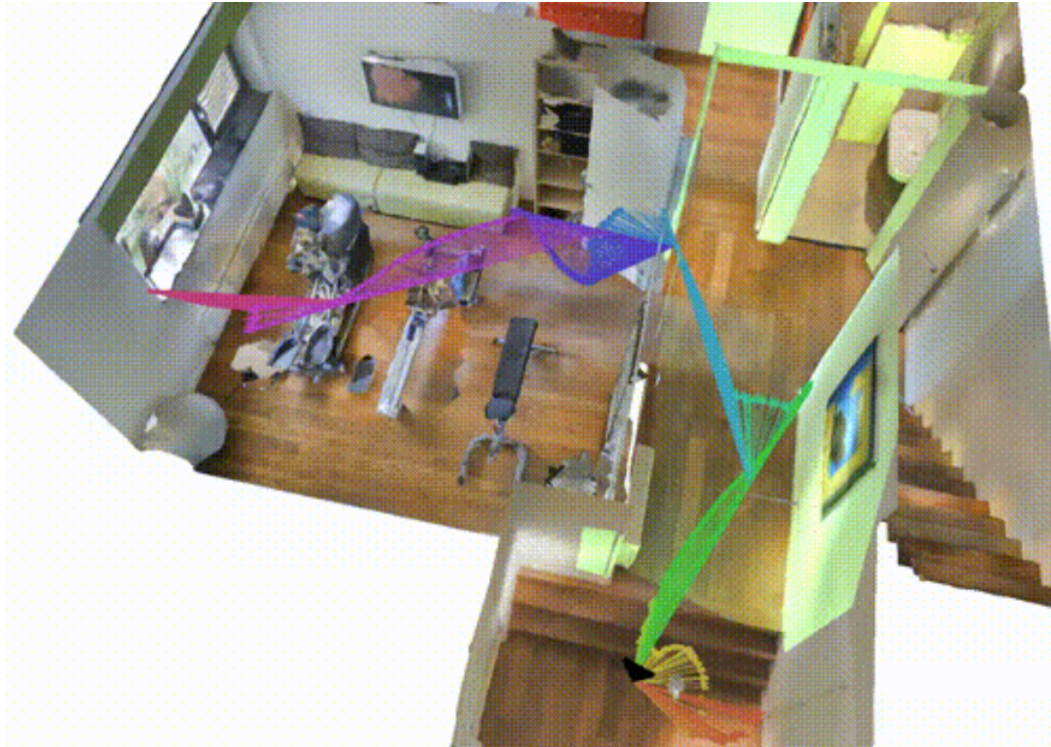
- *vlnce\_baselines/config/paper\_configs/intuition\_config/cma\_pm\_da\_tune.yaml*
- *habitat\_extensions/config/vlnce\_task\_intuit\_pano.yaml*
- *trainDA.sh*
- *evaljobDA\_tune.sh*

Important hyperparameters:

- **Maximum time-steps per episode:** 7500 during training (to account for extra steps performed in the VLNCEDaggerIntuitionEnv), 500 during evaluations
- **Success Distance:** 3 meters
- **Number of simultaneous Environments:** 8
- **Number of DA Iterations:** 10
- **Number of training Epochs per DA Iteration:** 4
- **DA P:** 0.5
- **Batch Size:** 5 (how many episode within a batch for loss calculation)
- **PM  $\alpha$ :** 1.0
- **IW  $\omega$ :** 3.2 (as was calculated in [9])
- **Number of intuition Steps:** 15
- **Learning Rate:** 2.5e-4 (as was used in previous papers [9])

Note: Evaluation on **val\_unseen** was performed in two parts, because it was initially launched with 7500 **Maximum time-steps per episode**, instead of 500.

Falcon 6 node was used for training. Falcon nodes 4, 2 and 1 were used for evaluation on **val\_seen**, 1st and 2nd parts of **val\_unseen** respectively.



Now you are standing in-front of a closed door, turn to your left, you can see two wooden steps, climb the steps and walk forward by crossing a wall painting which is to your right side, you can see open door enter into it. This is a gym room, move forward, walk till the end of the room, you can see a grey colored ball to the corner of the room, stand there, that's your end point.

Figure 3.4: An example of how an instruction contains information about the future based on a VLN application with RxR and Matterport3D Simulator. The image is a top down view on a scene segment with an Agent (black triangle). There are also coloured visualisations of directions an agent have and will have viewed; the colour corresponds to a slice of an instruction. While being in a green zone, the Agent can already have an intuition about moving and observing the scene at cyan and blue zones (at least). Image taken from [11].

## 4 Experimental Results

### 4.1 Results in the Main Training Phase

Each epoch was provided with the same set of 14700 batches. In total 662500 batches were processed overall. The first stage of training took around 55 hours of computing to complete (until the error) and the second - 57.5 hours. The best performing checkpoint is under number 43.

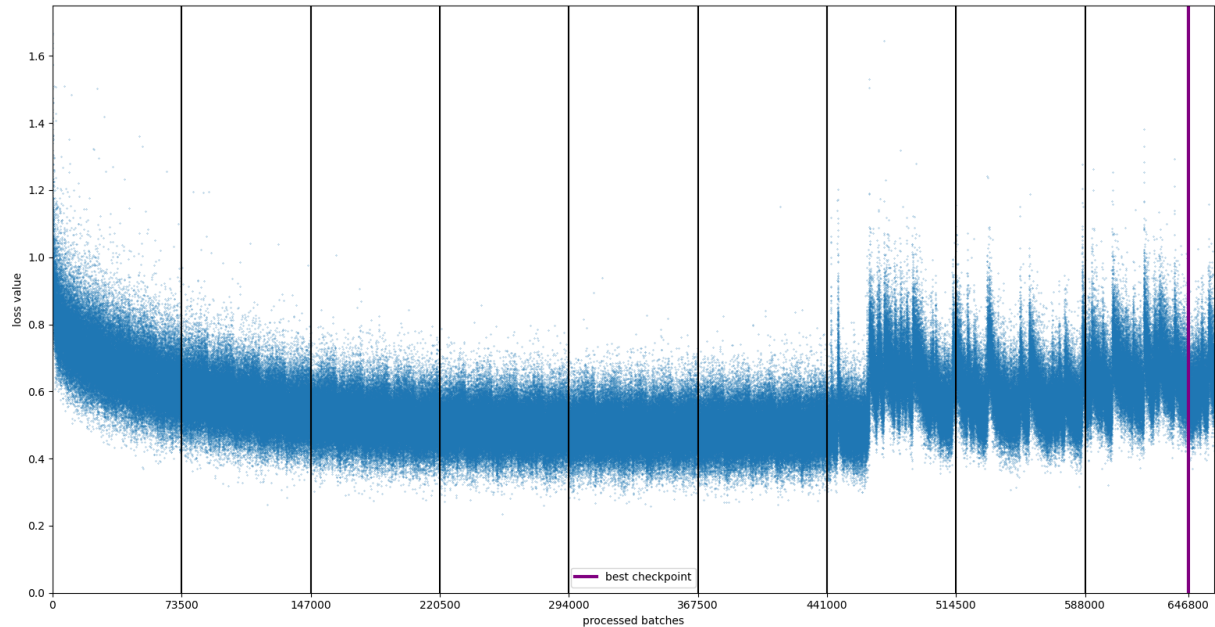


Figure 4.1: Loss values through out the whole Main Training Phase. Black vertical lines denote the end of every 5th Epochs. The purple vertical line represent the end of 43rd Epoch, i.e. the best performing checkpoint.

The majority of loss was contributed purely by action loss. Auxiliary loss was contained within the borders of  $[0.000489; 0.698321]$  and average value of 0.00507. Thus also depicting auxiliary loss (and therefore action loss as well) independently does not seem necessary.



| Checkpoint<br>number | Val_Seen |      |      |       |       |      |      |      |        | Val_Unseen |      |      |       |       |      |      |      |        |
|----------------------|----------|------|------|-------|-------|------|------|------|--------|------------|------|------|-------|-------|------|------|------|--------|
|                      | SR↑      | OS↑  | SPL↑ | nDTW↑ | sDTW↑ | WA↑  | NE↓  | TL↓  | Steps↓ | SR↑        | OS↑  | SPL↑ | nDTW↑ | sDTW↑ | WA↑  | NE↓  | TL↓  | Steps↓ |
| 43                   | 0.28     | 0.36 | 0.26 | 0.49  | 0.24  | 0.45 | 8.25 | 8.37 | 68.7   | 0.22       | 0.28 | 0.21 | 0.46  | 0.19  | 0.39 | 8.44 | 7.80 | 69.92  |

Table 4.1: Main Training Phase best checkpoint metrics

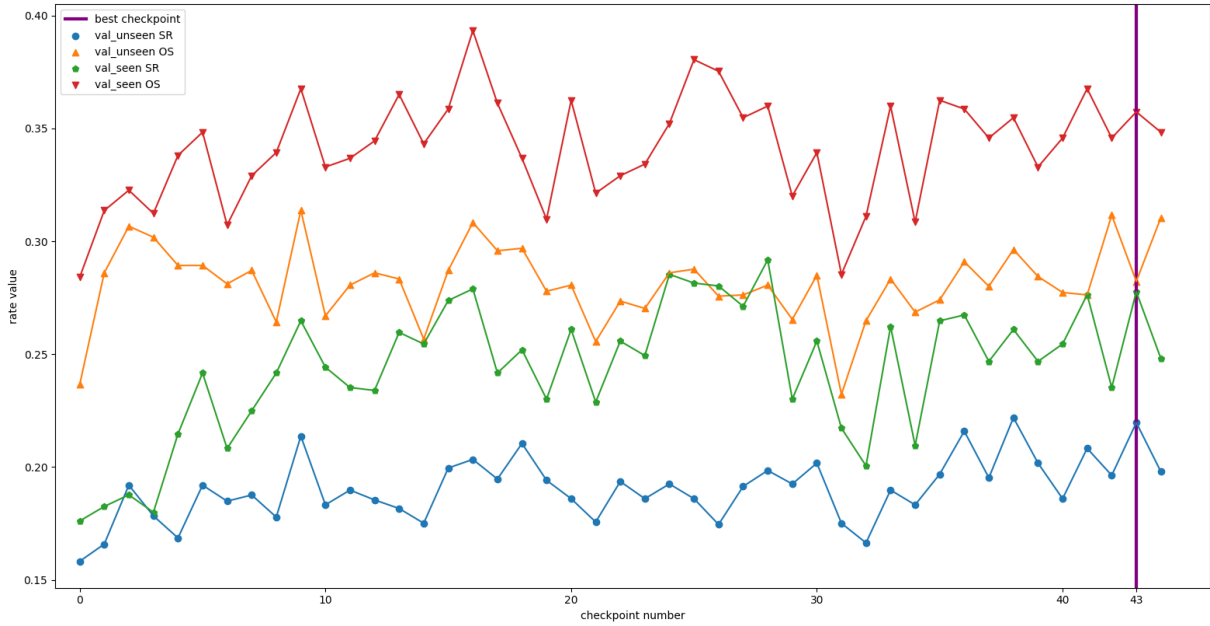
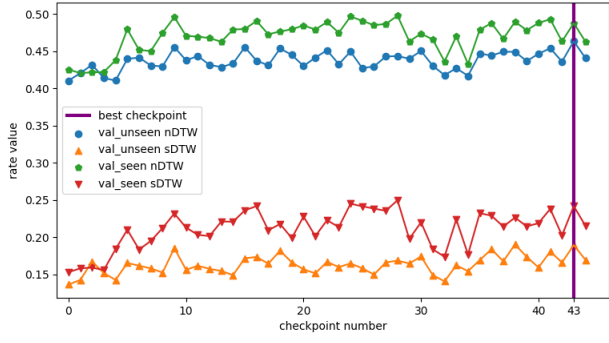
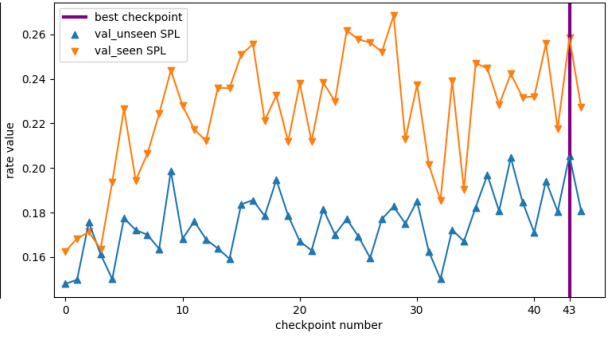


Figure 4.2: Results for **Success Rate (SR)** and **Oracle Success Rate (OS)** metrics received from the evaluations. The bigger rate is considered the better one. By comparing SR to OS on the same evaluation set, we can see how often has an Agent reached the goal, but did not issue a *stop* action in time.

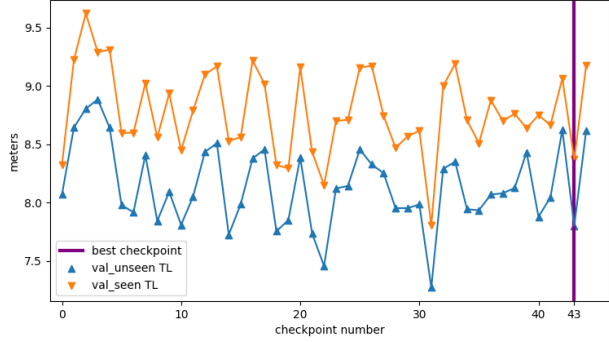
The purple vertical line marks the best performing checkpoint (43).



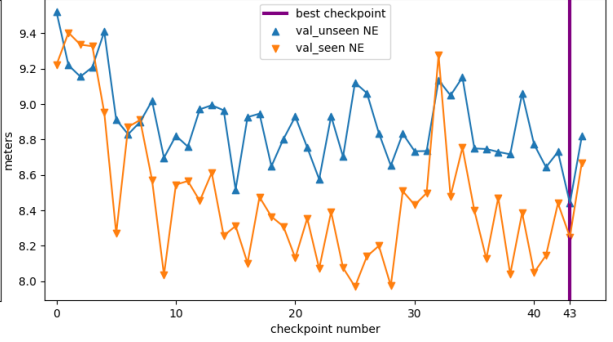
(a) Average **nDTW** and **sDTW**.  
The bigger value is considered the better one.



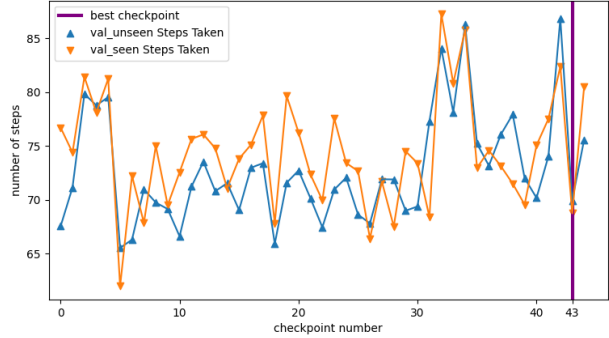
(b) Success weighted by inverse **Path Length (SPL)**.  
The bigger rate is considered the better one.



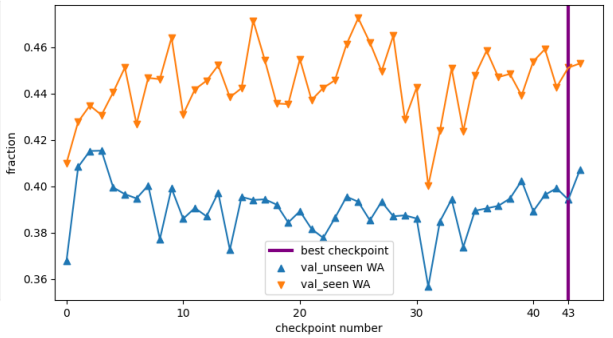
(c) Average **Trajectory Length (TL)**.  
The smaller value is considered the better one.



(d) Average **Navigation Error (NE)**.  
The smaller value is considered the better one.



(e) Average number of time-steps taken.  
The smaller number is considered the better one.



(f) Average **Waypoint Accuracy (WA)**.  
The bigger value is considered the better one.

Figure 4.3: The remaining metrics per evaluated checkpoint from the first phase of training. The purple vertical line marks the best performing checkpoint (43).

## 4.2 Results in the Fine tuning Phase

Each DA Iteration generated 1000 additional batches of training episodes. In total 220000 batches were processed. The whole program took 33 hours of computing to complete. The best performing checkpoint is under number 5.

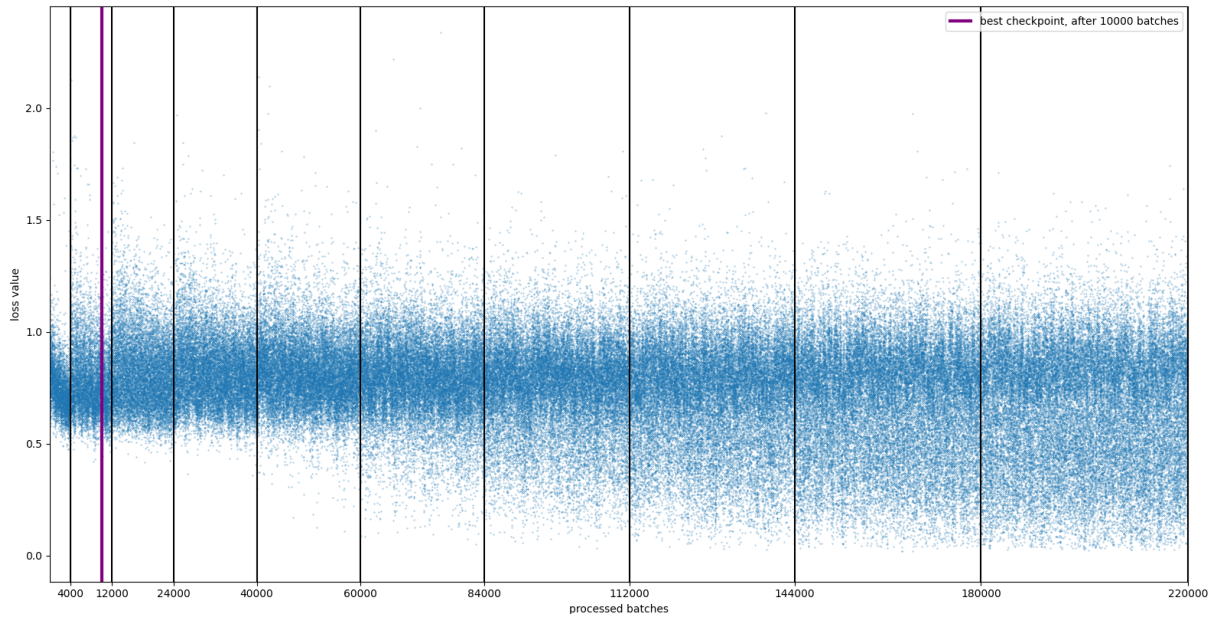


Figure 4.4: Loss values throughout the whole Fine tuning Phase. Black vertical lines denote the end of every DA Iteration. The purple vertical line represents the end of the 2nd epoch during the 2nd DA Iteration, i.e. the best-performing checkpoint.

The majority of the loss was contributed purely by action loss. Auxiliary loss was contained within the borders of  $[0.0; 0.8104]$  and average value of 0.0125. Thus also depicting auxiliary loss (and therefore action loss as well) independently does not seem necessary.

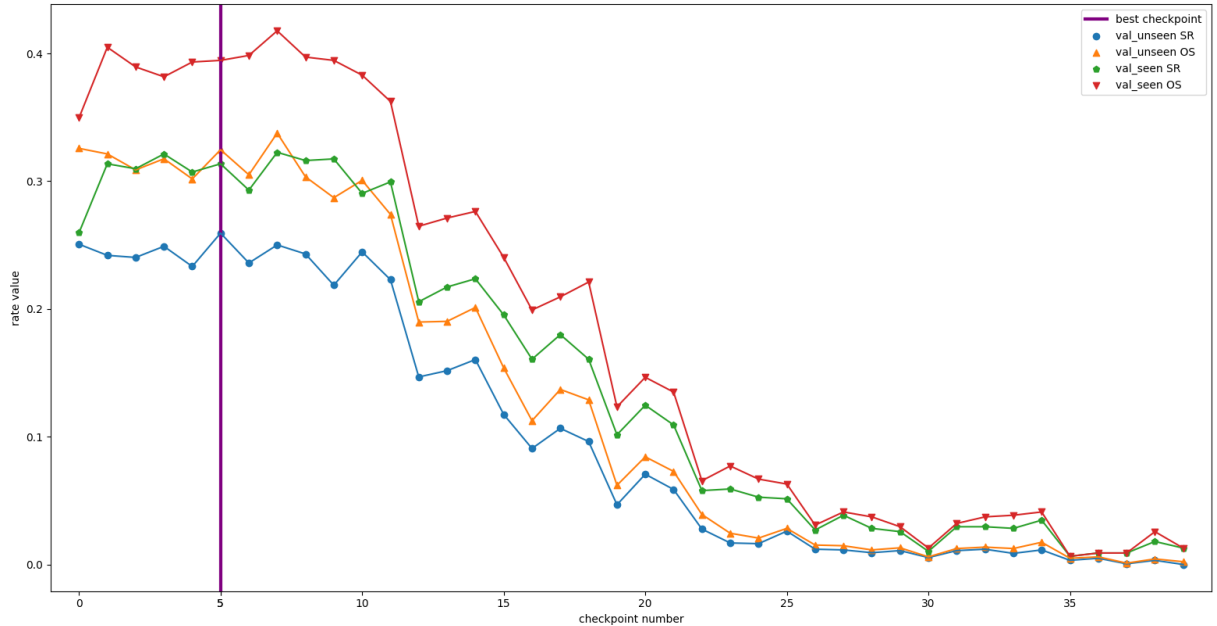
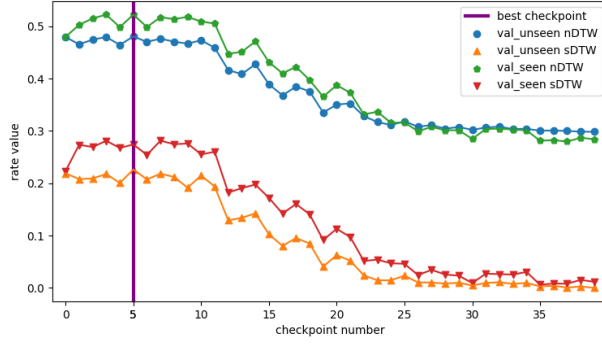


Figure 4.5: Results for **Success Rate (SR)** and **Oracle Success Rate (OS)** metrics received from the evaluations. The bigger rate is considered the better one. By comparing SR to OS on the same evaluation set, we can see how often has an Agent reached the goal, but did not issue a *stop* action in time.

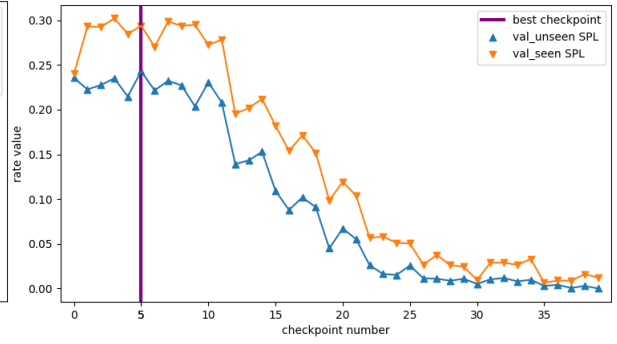
The purple vertical line marks the best performing checkpoint (5).

| Checkpoint<br>number | Val.Seen |      |      |       |       |      |      |      | Steps↓ | Val.Unseen |      |      |       |       |      |      |       |
|----------------------|----------|------|------|-------|-------|------|------|------|--------|------------|------|------|-------|-------|------|------|-------|
|                      | SR↑      | OS↑  | SPL↑ | nDTW↑ | sDTW↑ | WA↑  | NE↓  | TL↓  |        | SR↑        | OS↑  | SPL↑ | nDTW↑ | sDTW↑ | WA↑  | NE↓  | TL↓   |
| 5                    | 0.31     | 0.39 | 0.29 | 0.52  | 0.27  | 0.47 | 7.27 | 8.81 | 88.5   | 0.26       | 0.32 | 0.25 | 0.48  | 0.23  | 0.41 | 8.04 | 8.16  |
|                      |          |      |      |       |       |      |      |      |        |            |      |      |       |       |      |      | 405.0 |

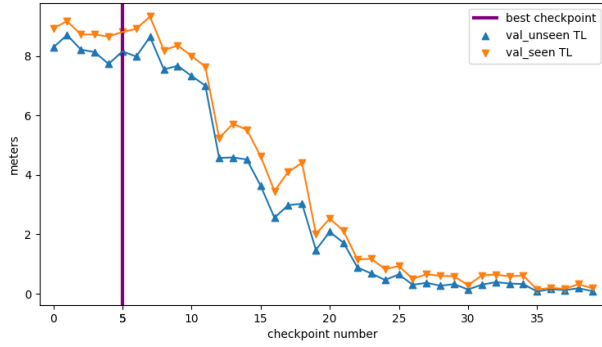
Table 4.2: Fine tuning Phase best checkpoint metrics



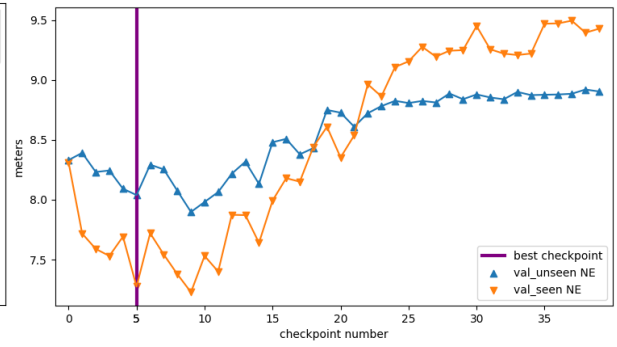
(a) Average **nDTW** and **sDTW**.  
The bigger value is considered the better one.



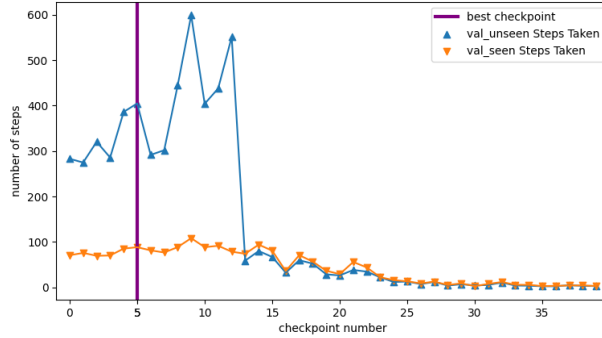
(b) **Success weighted by inverse Path Length (SPL)**.  
The bigger rate is considered the better one.



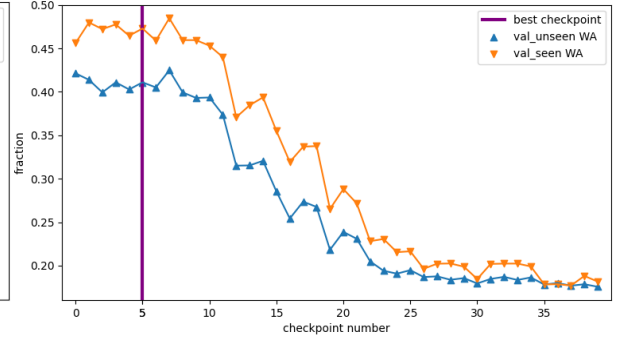
(c) Average **Trajectory Length (TL)**.  
The smaller value is considered the better one.



(d) Average **Navigation Error (NE)**.  
The smaller value is considered the better one.



(e) Average number of time-steps taken.  
The smaller number is considered the better one.



(f) Average **Waypoint Accuracy (WA)**.  
The bigger value is considered the better one.

Figure 4.6: The remaining metrics per evaluated checkpoint from the fine tuning phase of training. The purple vertical line marks the best-performing checkpoint (5).

## 5 Analysis and Discussion

### 5.1 Analysis of Results in the Main Training Phase

Looking at **nDTW** and **WA** values of the best-performing checkpoint (Table 4.1), we can say, that the Agent follows the correct path only for 46% of it. **OS** shows, that only 28% of the time does an Agent come close to the goal. With the addition of a relatively low score of average time-steps we can conclude, that the majority of episodes failed due to the Agent executing a *stop* action prematurely (i.e. before coming close to the goal). Evaluations on *val\_seen* split do not suggest an overfitting.

| Model            | Val.Seen    |      |             |             |             |             | Val.Unseen  |             |             |       |             |             |
|------------------|-------------|------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------|-------------|-------------|
|                  | SR↑         | OS↑  | SPL↑        | nDTW↑       | NE↓         | TL↓         | SR↑         | OS↑         | SPL↑        | nDTW↑ | NE↓         | TL↓         |
| Main Training    |             |      |             |             |             |             |             |             |             |       |             |             |
| Phase checkpoint | <b>0.28</b> | 0.36 | <b>0.26</b> | <b>0.49</b> | <b>8.25</b> | <b>8.37</b> | 0.22        | 0.28        | 0.21        | 0.46  | 8.44        | 7.80        |
| 43 (Ours)        |             |      |             |             |             |             |             |             |             |       |             |             |
| CMA.PM.Aug [9]   | 0.27        | 0.36 | 0.25        | 0.47        | 8.29        | 8.49        | <b>0.24</b> | <b>0.30</b> | <b>0.22</b> | 0.46  | <b>8.42</b> | <b>7.68</b> |

Table 5.1: Comparison between our best checkpoint at 1st phase and an equivalent model from VLN-CE paper [9]. We skip metrics, that are not reported for the alternative model. Our model performs slightly worse, than the alternative. This may attributed to the incomplete training set our model used. We cannot compare with a similar model from LAW-VLNCE paper, because they only report results after fine-tuning, [1].

### 5.2 Analysis of Results in the Fine Tuning Phase

We can apply the same criticism to this phase’s best checkpoint as we did with the previous one, although it should be pointed that the performance did in fact improve. Another thing to note is the increase in average time-steps taken and average trajectory length. A large average number of time-steps in **val\_unseen** also suggests that an Agent did not issue a *stop* action for a long time in some episodes, possible even hitting the maximum time-step limit.

We can observe from Figures 4.5 and 4.6 that consecutive DA Iterations after the second one harmed the model’s performance. The root of the problem can be that DAgger over-saturates the training data with faulty trajectories created by an inexperienced model. Figure 4.6e hints at the later checkpoints’ behaviour: they terminate episodes very early on, only after a few time-steps.

| Model                               | Val-Seen    |             |             |             |             |             |             |             | Val-Unseen  |             |             |             |             |             |             |             |
|-------------------------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
|                                     | SR↑         | OS↑         | SPL↑        | nDTW↑       | sDTW↑       | WA↑         | NE↓         | TL↓         | SR↑         | OS↑         | SPL↑        | nDTW↑       | sDTW↑       | WA↑         | NE↓         | TL↓         |
| Fine tune<br>checkpoint 5<br>(Ours) | 0.31        | 0.39        | 0.29        | 0.52        | 0.27        | 0.47        | 7.27        | <b>8.81</b> | 0.26        | 0.32        | 0.25        | 0.48        | 0.23        | 0.41        | 8.04        | <b>8.16</b> |
| goal [1]                            | 0.34        | 0.44        | 0.32        | 0.54        | 0.29        | 0.48        | 7.21        | 9.06        | 0.29        | 0.36        | 0.27        | 0.50        | 0.24        | 0.41        | 7.60        | 8.27        |
| LAW_pano [1]                        | <b>0.40</b> | <b>0.49</b> | <b>0.37</b> | <b>0.58</b> | <b>0.35</b> | <b>0.56</b> | <b>6.35</b> | 9.34        | <b>0.35</b> | <b>0.44</b> | <b>0.31</b> | <b>0.54</b> | <b>0.29</b> | <b>0.47</b> | <b>6.83</b> | 8.89        |

Table 5.2: Comparison between our best model and two models from a preceding paper [1]. goal is a CMA with PM and IW and supervised with a greedy goal-oriented sensor, trained under the same conditions as we did. Essentially an equivalent to the best performing CMA model from VLN-CE [9]. LAW\_pano is the same, but supervised with LAW sensor, i.e. our model without our modifications. Average steps taken metric is skipped, because it was not reported in the original paper [1].

Our model is under-performing in both cases.

## 5.3 Discussion of the Results

The results based on our approach hint at the scope of further improvement. The suboptimal results of our experiments due to the incomplete set of training data. The fluctuating loss values in Figures 4.1 and 4.4 can be inferred because of the model’s inability to focus properly. As the loss is calculated based on the average, at each intuition step, the model jumps back and forth: improving predictions at one intuition step, simultaneously damaging some other intuition step, then switching to fixing another and ruining something else yet again. A better loss function would perhaps be a weighted sum, which would give more weight to the predicted actions first in the intuition sequence. It would return to the action loss being dependent on the decision of taking action at the current time-step while relegating the remaining action predictions to the position of auxiliary loss.

$$\text{Current loss} \quad Loss_{action} = -\frac{1}{I} \sum_{i=0}^I l_i$$

$$\text{Newly proposed loss,} \quad Loss_{action} = -\sum_{i=0}^I m_i l_i \quad \text{where } m_i > m_{i_1}$$

## 6 Future work

We plan to use larger models where the interplay of scaling and power-law of natural language will give the optimal policy for navigation. The higher-order reasoning while doing the navigation will be expressed more with a tree-of-thought (ToT) and category theory. We will explore this tree of thought (ToT) with an architectural bias of graph neural networks and more scaling. The other direction is exploiting the model’s improvising capacity while choosing the optimal route during navigation. Improvisations is an emergent phenomenon, while the model has a global structure (a.k.a knowledge of the environments) and will place the local decisions of choosing the decisions accordingly. The present neural attention architecture is shallow and the inference is made based on limited context. We plan to use inductive bias through architectural innovations (tree) on the self-attention block. The other direction is introduced Song, et al [54], an alternative method of evoking explicit intuition is to modify the model to also predict visual observation of the next time-step, as in *”consider what will you see, if taken the action you are thinking of”*. Then as an auxiliary loss calculate the minimum square error between the predicted and an actual image.

## 7 Conclusion

This thesis investigates a non-trivial problem of Visual-and-Language Navigation task by an robot (Agent). An Embodied Agent (robot) navigates from point A to point B by following a given a Natural Language instruction and perceiving the surroundings through an RGB-D camera mounted on top of it. The thesis provides an introduction to the field of autonomous navigation indoors, later focusing on problem simulations in Continuous Environments. The machine learning model maps natural language (NL) instructions along with visual inputs for optimal navigation policy. Natural language through its syntax and semantics provides a high-level reasoning that is hidden in the latent layers of the model. The look-ahead policy aided by natural language (along with vision) allows the agent to have exploration-vs-exploitation while navigating on unknown environment. The thesis gives an idea of the exploration vs exploitation method through the interplay of shortest-path algorithm and natural language. Natural Language because of its inherent ambiguity (and occasionally sloppiness) overrides the standard decisions and follows unconventional routes.



# Acknowledgements

The Author of this thesis would like to express gratitude toward the supervisor of this paper, Kallol Roy, for providing the topic and helping out along the way. Another *thank you* is directed at Jakob Krantz, the maintainer of VLN-CE project [9], who took time answering Author's questions. The University of Tartu HPC Center [57] deserves appreciation for providing students with computational services. Special thanks to the Author's family, who supported him throughout his education path.

Another round of gratitude towards:

- Chang, et al, for creating Matterport3D dataset [24]
- Anderson, et al, for creating R2R dataset [4]
- Krantz, et al, for adapting R2R dataset to Continuous Environments and creating VLN-CE project [9]
- Raychaudhuri, et al, for creating LAW-VLNCE project [1]

Also the Author wishes to thank the University of Tartu for granting the ability to study and the reason of writing this very paper.

# Bibliography

- [1] Raychaudhuri, S., Wani, S., Patel, S., Jain, U., and Chang, A., “Language-aligned way-point (law) supervision for vision-and-language navigation in continuous environments,” in *[Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing]*, 4018–4028 (2021).
- [2] “Promotional material for GitHub Copilot.” Github Features, copilot <https://github.com/features/copilot>. (Accessed: 03 December 2023).
- [3] Shah, P., Fiser, M., Faust, A., Kew, J. C., and Hakkani-Tür, D., “Follownet: Robot navigation by following natural language directions with deep reinforcement learning,” *Third Workshop in Machine Learning in the Planning and Control of Robot Motion at ICRA* **abs/1805.06150** (2018).
- [4] Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., and van den Hengel, A., “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).
- [5] “Matterport3DSimulator/teaser.jpg.” GitHub, user peteanderson80, Matterport3DSimulator <https://github.com/peteanderson80/Matterport3DSimulator/blob/master/teaser.jpg>. (Accessed: 06 December 2023).
- [6] Tan, H., Yu, L., and Bansal, M., “Learning to navigate unseen environments: Back translation with environmental dropout,” in *[Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)]*, Burstein, J., Doran, C., and Solorio, T., eds., 2610–2621, Association for Computational Linguistics, Minneapolis, Minnesota (June 2019).
- [7] “Matterport3D: Learning from RGB-D Data in Indoor Environments.” github.io - Matterport <https://niessner.github.io/Matterport/>. (Accessed: 09 December 2023).
- [8] Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., and Fox, D., “ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks,” in *[The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)]*, (2020).
- [9] Krantz, J., Wijmans, E., Majundar, A., Batra, D., and Lee, S., “Beyond the nav-graph: Vision and language navigation in continuous environments,” in *[European Conference on Computer Vision (ECCV)]*, (2020).

- [10] “LAW-VLNCE github.io page.” Github.io, Language-Aligned Waypoint (LAW) Supervision for Vision-and-Language Navigation in Continuous Environments, EMNLP 2021 <https://3dlg-hcvc.github.io/LAW-VLNCE/>. (Accessed: 19 December 2023).
- [11] “RxR dataset page.” Google AI, Room-Across-Room (RxR) <https://ai.google.com/research/rxr/>. (Accessed: 19 December 2023).
- [12] “R2R-VLNCE dataset page.” GitHub.io, user jacobkrantz, VLNCE data <https://jacobkrantz.github.io/vlnce/data>. (Accessed: 11 December 2023).
- [13] “[Futurism] ”Gizmodo and Kotaku Staff Furious After Owner Announces Move to AI Content” by reddit user frosty\_farralon.” Reddit r/KotakuInAction [https://www.reddit.com/r/KotakuInAction/comments/14ock80/futurism\\_gizmodo\\_and\\_kotaku\\_staff\\_furious\\_after/](https://www.reddit.com/r/KotakuInAction/comments/14ock80/futurism_gizmodo_and_kotaku_staff_furious_after/). (Accessed: 01 December 2023).
- [14] “Midjourney homepage.” <https://www.midjourney.com>. (Accessed: 01 December 2023).
- [15] “Stable Diffusion XL.” stability.ai, models, image <https://stability.ai/stable-diffusion>. (Accessed: 01 December 2023).
- [16] “DALL-E 3.” OpenAI, Research, DALL-E 3 <https://openai.com/dall-e-3>. (Accessed: 01 December 2023).
- [17] “ChatGPT product page.” OpenAI, ChatGPT, Overview <https://openai.com/chatgpt>. (Accessed: 01 December 2023).
- [18] “What are some philosophical answers to why the sky is blue?.” Quora <https://www.quora.com/What-are-some-philosophical-answers-to-why-the-sky-is-blue>. (Accessed: 01 December 2023).
- [19] Anderson, P., Shrivastava, A., Truong, J., Majumdar, A., Parikh, D., Batra, D., and Lee, S., “Sim-to-real transfer for vision-and-language navigation,” in [*Conference on Robot Learning (CoRL)*], (2020).
- [20] Krantz, J., Gokaslan, A., Batra, D., Lee, S., and Maksymets, O., “Waypoint models for instruction-guided navigation in continuous environments,” in [*2021 IEEE/CVF International Conference on Computer Vision (ICCV)*], 15142–15151 (2021).
- [21] “AI2-THOR: An Interactive 3D Environment for Visual AI,” *ArXiv* **abs/1712.05474** (2017).
- [22] Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., Parikh, D., and Batra, D., “Habitat: A Platform for Embodied AI Research,” in [*Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*], (2019).
- [23] Misra, D., Bennett, A., Blukis, V., Niklasson, E., Shatkhin, M., and Artzi, Y., “Mapping instructions to actions in 3D environments with visual goal prediction,” in [*Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*], Riloff,

- E., Chiang, D., Hockenmaier, J., and Tsujii, J., eds., 2667–2678, Association for Computational Linguistics, Brussels, Belgium (Oct.-Nov. 2018).
- [24] Chang, A., Dai, A., Funkhouser, T., Halber, M., Niessner, M., Savva, M., Song, S., Zeng, A., and Zhang, Y., “Matterport3d: Learning from rgb-d data in indoor environments,” *International Conference on 3D Vision (3DV)* (2017).
  - [25] Thomason, J., Gordon, D., and Bisk, Y., “Shifting the baseline: Single modality performance on visual navigation & QA,” in [*Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*], Burstein, J., Doran, C., and Solorio, T., eds., 1977–1983, Association for Computational Linguistics, Minneapolis, Minnesota (June 2019).
  - [26] Jain, V., Magalhaes, G., Ku, A., Vaswani, A., Ie, E., and Baldrige, J., “Stay on the path: Instruction fidelity in vision-and-language navigation,” in [*Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*], Korhonen, A., Traum, D., and Màrquez, L., eds., 1862–1872, Association for Computational Linguistics, Florence, Italy (July 2019).
  - [27] Ku, A., Anderson, P., Patel, R., Ie, E., and Baldrige, J., “Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding,” in [*Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*], 4392–4412 (2020).
  - [28] “RxR-Habitat Competition page.” Google AI, Room-Across-Room (RxR), RxR-Habitat Competition <https://ai.google.com/research/rxr/habitat>. (Accessed: 11 December 2023).
  - [29] Hong, Y., Rodriguez-Opazo, C., Wu, Q., and Gould, S., “Sub-instruction aware vision-and-language navigation,” *arXiv preprint arXiv:2004.02707* (2020).
  - [30] Pennington, J., Socher, R., and Manning, C., “GloVe: Global vectors for word representation,” in [*Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*], Moschitti, A., Pang, B., and Daelemans, W., eds., 1532–1543, Association for Computational Linguistics, Doha, Qatar (Oct. 2014).
  - [31] “Configuration file, which uses joint\_train\_envdrop split of R2R\_VLNCE\_v1-3 for training.” Github, VLN-CE master branch [https://github.com/jacobkrantz/VLN-CE/blob/master/vlnce\\_baselines/config/r2r\\_baselines/cma\\_aug.yaml](https://github.com/jacobkrantz/VLN-CE/blob/master/vlnce_baselines/config/r2r_baselines/cma_aug.yaml). (Accessed: 12 December 2023).
  - [32] “VLN-CE Challenge.” Eval.ai, VLN-CE Challenge organized by VIRT <https://eval.ai/web/challenges/challenge-page/719/overview>. (Accessed: 12 December 2023).
  - [33] “VLN-CE project repository.” Github, VLN-CE <https://github.com/jacobkrantz/VLN-CE/tree/master?tab=readme-ov-file>. (Accessed: 14 December 2023).
  - [34] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M.,

- Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S., “Pytorch: An imperative style, high-performance deep learning library,” in [*Advances in Neural Information Processing Systems 32*], 8024–8035, Curran Associates, Inc. (2019).
- [35] “Python 3.6.” The Python Language Reference (3.6.15) <https://docs.python.org/3.6/reference/index.html>. (Accessed: 17 December 2023).
- [36] “Habitat-Sim v0.1.7 repository.” Github, habitat-sim <https://github.com/facebookresearch/habitat-sim/tree/v0.1.7>. (Accessed: 14 December 2023).
- [37] “Habitat Lab v0.1.7 repository.” Github, habitat-lab <https://github.com/facebookresearch/habitat-lab/tree/v0.1.7>. (Accessed: 14 December 2023).
- [38] “LoCoBot: An Open Source Low Cost Robot.” LoCoBot website homepage <https://locobot-website.netlify.app>. (Accessed: 14 December 2023).
- [39] “Documentation for a universal Python binding for the LMDB ‘Lightning’ Database.” Readthedocs, lmdb, latest release <https://lmdb.readthedocs.io/en/latest/>. (Accessed: 17 December 2023).
- [40] “TORCH.NN.FUNCTIONAL.CROSS\_ENTROPY.” PyTorch documentation, Python API [https://pytorch.org/docs/stable/generated/torch.nn.functional.cross\\_entropy.html](https://pytorch.org/docs/stable/generated/torch.nn.functional.cross_entropy.html). (Accessed: 17 December 2023).
- [41] Wijmans, E., Datta, S., Maksymets, O., Das, A., Gkioxari, G., Lee, S., Essa, I., Parikh, D., and Batra, D., “Embodied Question Answering in Photorealistic Environments with Point Cloud Perception,” in [*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*], (2019).
- [42] Ma, C.-Y., Lu, J., Wu, Z., AlRegib, G., Kira, Z., Socher, R., and Xiong, C., “Self-monitoring navigation agent via auxiliary progress estimation,” in [*Proceedings of the International Conference on Learning Representations (ICLR)*], (2019).
- [43] Ross, S., Gordon, G., and Bagnell, D., “A reduction of imitation learning and structured prediction to no-regret online learning,” in [*Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*], Gordon, G., Dunson, D., and Dudík, M., eds., *Proceedings of Machine Learning Research* **15**, 627–635, PMLR, Fort Lauderdale, FL, USA (11–13 Apr 2011).
- [44] He, K., Zhang, X., Ren, S., and Sun, J., “Deep residual learning for image recognition,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778 (2015).
- [45] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L., “Imagenet: A large-scale hierarchical image database,” in [*2009 IEEE Conference on Computer Vision and Pattern Recognition*], 248–255 (2009).
- [46] Wijmans, E., Kadian, A., Morcos, A., Lee, S., Essa, I., Parikh, D., Savva, M., and Batra, D., “DD-PPO: Learning near-perfect pointgoal navigators from 2.5 billion frames,” in [*International Conference on Learning Representations (ICLR)*], (2020).

- [47] Hochreiter, S. and Schmidhuber, J., “Long short-term memory,” *Neural computation* **9**(8), 1735–1780 (1997).
- [48] Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y., “Learning phrase representations using RNN encoder–decoder for statistical machine translation,” in [*Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*], Moschitti, A., Pang, B., and Daelemans, W., eds., 1724–1734, Association for Computational Linguistics, Doha, Qatar (Oct. 2014).
- [49] “Categorical Distribution class.” PyTorch documentation, Python API, PROBABILITY DISTRIBUTIONS <https://pytorch.org/docs/stable/distributions.html#torch.distributions.categorical.Categorical>. (Accessed: 19 December 2023).
- [50] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I., “Attention is all you need,” in [*Advances in Neural Information Processing Systems*], 5998–6008 (2017).
- [51] “Greedy Geodesic Follower source code.” Github, habitat-sim v0.1.7 [https://github.com/facebookresearch/habitat-sim/blob/v0.1.7/habitat\\_sim/nav/greedy\\_geodesic\\_follower.py#L18](https://github.com/facebookresearch/habitat-sim/blob/v0.1.7/habitat_sim/nav/greedy_geodesic_follower.py#L18). (Accessed: 19 December 2023).
- [52] Anderson, P., Chang, A. X., Chaplot, D. S., Dosovitskiy, A., Gupta, S., Koltun, V., Kosecka, J., Malik, J., Mottaghi, R., Savva, M., and Zamir, A. R., “On evaluation of embodied navigation agents,” *ArXiv* **abs/1807.06757** (2018).
- [53] Ilharco, G., Jain, V., Ku, A., Ie, E., and Baldrige, J., “General evaluation for instruction conditioned navigation using dynamic time warping,” *ArXiv* **abs/1907.05446** (2019).
- [54] Song, S., Yu, F., Zeng, A., Chang, A. X., Savva, M., and Funkhouser, T., “Semantic scene completion from a single depth image,” *arXiv preprint arXiv:1611.08974* (2016).
- [55] “INTUIT-VLNCE github repository.” GitHub <https://github.com/SirKrovlood/INTUIT-VLNCE/tree/master>. (Accessed: 20 December 2023).
- [56] Kingma, D. P. and Ba, J., “Adam: A method for stochastic optimization,” *CoRR* **abs/1412.6980** (2014).
- [57] “Rocket HPC cluster.” UT HPC Center, Services / High Performance Computers and Services, Rocket <https://hpc.ut.ee/services/HPC-services/Rocket>. (Accessed: 20 December 2023).
- [58] “HPC cluster prices.” UT HPC Center, Prices <https://hpc.ut.ee/pricing/>. (Accessed: 22 December 2023).

## MATTERPORT END USER LICENSE AGREEMENT FOR ACADEMIC USE OF MODEL DATA

Last updated: Aug 29, 2017 (last edit by Matthias, added signature fields)

### SUMMARY:

Under the following Agreement, Matterport grants You the right to use certain 3D model data. The Agreement provides that:

- The data provided by Matterport is for non-commercial academic use only.
- You must include the Agreement, or a link to it, with any material You publish that contains the data or any information derived from the data, and You must use a “click-wrap” agreement for any distribution of a substantial portion of the data.
- You are prohibited from attempting to identify the address or owner of the models to which the data pertains.
- You are not permitted to use any Matterport software embedded in the data for any purpose other than accessing the data.
- You are prohibited from using the data to create any online experience similar to Matterport’s 3D Showcase.
- Matterport provides the data to You as is. Matterport will have no liability for Your use of the data, and You will indemnify Matterport for such use.

### AGREEMENT:

This Matterport End User License Agreement for Academic Use of Model Data (“**Agreement**”) is between Matterport, Inc. (“**Matterport**”) and you or the entity that you represent (collectively, “**You**”), and governs Your use of the Matterport Dataset (defined below). If You represent an entity, You represent and warrant that You are fully authorized to enter into this Agreement on behalf of such entity.

PLEASE READ THE TERMS OF THIS AGREEMENT CAREFULLY. BY USING THE MATTERPORT DATASET, YOU SIGNIFY YOUR AGREEMENT TO ALL TERMS, CONDITIONS, AND NOTICES CONTAINED OR REFERENCED IN THIS AGREEMENT AND YOU AGREE THAT THIS AGREEMENT IS ENFORCEABLE LIKE ANY WRITTEN CONTRACT SIGNED BY YOU. IF YOU DO NOT AGREE TO THE FOREGOING AND SUCH TERMS, CONDITIONS AND/OR NOTICES, YOU DO NOT HAVE THE RIGHT TO USE THE MATTERPORT DATASET.

Matterport reserves the right to revise any portion of this Agreement in its sole discretion at any time and without prior notice to You by updating this posting, such changes to be effective prospectively. Thus, You should check [this page](#) periodically for changes. If You disagree with any changes to this Agreement, Your sole remedy is to discontinue Your use of the Matterport Dataset. Your continued use of the Matterport Dataset after a change has been posted constitutes Your acceptance of the change thereafter.

### 1. Definitions.

“**3D Data**” means standard digital 3D models, 3D meshes, visual textures associated with 3D meshes, 3D point clouds, or other spatially located information produced by or derived from data gathered with a Matterport 3D Camera.

“**Matterport Dataset**” means Panoramic Data and 3D Data from the models of Matterport or Matterport’s customers that Matterport provides to You hereunder, including without limitation any data format information.

“**Matterport Dataset Derived Information**” means any information or technology derived from the Matterport Dataset, including without limitation any models trained on the Matterport Dataset.

“**Matterport Property**” means: (a) the Matterport Dataset; (b) all technology, software, websites, products and services of Matterport, including any technology of Matterport embodied in the Matterport Dataset; and (c) any and all look and feel, improvements, updates, modifications, translations, copies, compilations and derivative works related to any of the foregoing.

“**Panoramic Data**” means 2D panoramic image data captured by a Matterport 3D Camera regarding visual files, annotations, digital media, or other information that can be displayed in specified locations within a standard 3D model.

## **2. Access; License; Proprietary Rights; Restrictions.**

- 2.1. Access to Matterport Dataset. During the term of this Agreement, Matterport will make available to You, by download or in another manner determined by Matterport, a copy of a Matterport Dataset for a 3D models. The data included in the Matterport Dataset provided to You will be selected by Matterport in its sole discretion. Matterport will have no obligation to provide data for specific models requested by You.
- 2.2. License to Matterport Dataset. Subject to the terms and conditions of this Agreement, Matterport hereby grants You a limited, revocable, royalty-free, non-exclusive license: (a) to use the Matterport Dataset for non-commercial academic purposes only; and (b) to use any software embedded in the Matterport Dataset for the sole purpose of accessing the Matterport Dataset. Matterport has no obligation to provide any development support to You under this Agreement.
- 2.3. Proprietary Rights. Matterport and its licensors owns all right, title and interest (including all associated intellectual property rights) in and to the Matterport Property. Except for the limited license granted in Section 2.2, Matterport reserves all rights in the Matterport Property.
- 2.4. Restrictions.
  - (a) For the avoidance of doubt, except as otherwise specifically agreed by Matterport in writing (email sufficient), You shall not: (i) use or distribute the Matterport Dataset for any non-academic purpose; (ii) create, use or distribute any Matterport Dataset Derived Information for any non-academic purpose; (iii) alter, remove or destroy any attribution, proprietary markings (e.g., copyright and trademark markings) or confidential legends placed upon or contained within the Matterport Dataset; (iv) use the Matterport Dataset to create interactive online experiences similar to Matterport's 3D Showcase; (v) decompile, disassemble, decrypt, extract, reverse engineer, extract or otherwise attempt to derive the source code of any software underlying the Matterport Dataset; (vi) use any Matterport software embedded in the



Matterport Dataset for any purpose other than accessing the Matterport Dataset, or distribute any such software separately from the Matterport Dataset; or (vii) attempt to determine or extract the address, owner, or identity of any locations included in the Matterport Dataset, or any other personal information that may be included in or with the Matterport Dataset. For the avoidance of doubt, the restriction in subsections (i) and (ii) above shall not apply to any distribution of the Matterport Dataset or Matterport Dataset Derived Information to Matterport, or Matterport's use thereof.

- (b) In the event that You publish or distribute the Matterport Dataset or any Matterport Dataset Derived Information, You must include this Agreement, or a hyperlink to this Agreement, with the Matterport Dataset or any Matterport Dataset Derived Information; provided, however, that, if You distribute a substantial portion of the Matterport Dataset – either directly or as part of Matterport Dataset Derived Information – You shall require each recipient thereof to accept the terms and conditions of this Agreement by means of a mutually executed document or a “click-wrap” mechanism that: (i) requires the recipient to affirmatively accept such terms and conditions before receiving access; and (ii) keeps a record of each recipient that accepts such terms and conditions.
  - (c) As a condition of receiving the Matterport Dataset, You agree to provide to Matterport Your name, email address and academic institution, and You consent to Matterport's collection and use of such personal information for Matterport's internal purposes, including outreach to You. Furthermore, when You require recipients to accept the terms and conditions of this Agreement pursuant to Section 2(b), You shall also use reasonable efforts, to the extent permitted under applicable law, to: (a) require such recipients to provide their name, email address and academic institution to You; (b) obtain the recipients' consent for You to collect such information and share it with Matterport, and for Matterport to use it for Matterport's internal and outreach purposes; and (c) promptly provide such collected information to Matterport. You shall comply with all applicable laws and regulations relating to the collection, use and disclosure of personal information in performing Your obligations hereunder.
3. **Term; Termination.** The term of this Agreement and the license granted herein continues until this Agreement is terminated by either party; provided, however, that the license is perpetual with respect to any portion of the Matterport Dataset or any Matterport Dataset Derived Information to the extent contained in material that You published prior to the date of termination of this Agreement. You may terminate this Agreement at any time by discontinuing use of, and deleting all copies of, the Matterport Dataset and any Matterport Dataset Derived Information. This Agreement terminates automatically if You fail to comply with any of the terms of this Agreement. Upon any termination of this Agreement, You must cease use of the Matterport Dataset and any Matterport Dataset Derived Information and destroy all copies of the foregoing; provided, however, that nothing herein will obligate You to delete or remove any portion of the Matterport Dataset or any Matterport Dataset Derived Information from any material that You published prior to the date of termination of this Agreement that did not violate the terms of the Agreement. Sections 2.3, 2.4(a), and 4 through 10 shall survive any expiration or termination of this Agreement.
4. **Disclaimer.** THE MATTERPORT DATASET IS PROVIDED “AS IS.” MATTERPORT MAKES NO WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, AND EXPRESSLY DISCLAIMS, ON BEHALF OF ITSELF AND ITS LICENSORS, ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT. WITHOUT LIMITING THE FOREGOING, MATTERPORT MAKES NO WARRANTY THAT THE MATTERPORT DATASET WILL BE ERROR-FREE OR FREE FROM INTERRUPTIONS OR OTHER FAILURES

OR THAT THE MATTERPORT DATASET WILL MEET YOUR REQUIREMENTS. Some jurisdictions do not allow some of the foregoing exclusions or limitations, so some of these exclusions or limitations may not apply to You.

5. **Indemnification.** Matterport shall not be obligated to indemnify, defend or hold You harmless with respect to any third-party claims arising out or relating to the Matterport Dataset, including without limitation any claims for intellectual property infringement. Nothing in this Agreement shall be deemed an admission that any such claims may arise. You, at Your own expense, will indemnify, defend and hold harmless Matterport, its corporate affiliates and their respective officers, directors, employees, representatives and agents (each a “**Matterport Indemnatee**”) from and against any claim, demand, action, class action, investigation or other proceeding, including but not limited to all damages, losses, liabilities, judgments, costs and expenses (including reasonable attorneys’ fees) arising therefrom (each a “**Claim**”), brought by any third party against a Matterport Indemnatee to the extent that such Claim is based on, or arises out of: (a) a breach, or potential breach, of any of Your obligations under this Agreement; (b) Your use of the Matterport Dataset, including Your use of any Matterport Dataset Derived Data that You may create; or (c) any alleged or actual fraud, gross negligence or willful misconduct of You or Your subcontractors or agents. In the event of a claim in respect of which a Matterport Indemnatee seeks indemnification from You under this Section 5, the Matterport Indemnatee will promptly notify You in writing of the claim, cooperate with You in defending or settling the claim at Your expense, and allow You to control the defense and settlement of the claim, including the selection of attorneys; provided, however, that You shall not settle any claim unless such settlement completely and forever releases the Matterport Indemnatee from all liability with respect to such claim or unless the Matterport Indemnatee consents to such settlement in writing.
6. **Limitation of Liability.** TO THE MAXIMUM EXTENT PERMITTED BY LAW: (a) IN NO EVENT WILL MATTERPORT, ITS AFFILIATES OR LICENSORS BE LIABLE, UNDER ANY CONTRACT, NEGLIGENCE, STRICT LIABILITY OR OTHER LEGAL OR EQUITABLE THEORY FOR ANY INDIRECT, SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES; AND (b) IN NO EVENT SHALL THE TOTAL LIABILITY OF MATTERPORT, ITS AFFILIATES OR ITS LICENSORS UNDER THIS AGREEMENT EXCEED \$50. THESE LIMITATIONS SHALL APPLY NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY REMEDY. You acknowledge and agree that the foregoing limitations of liability are essential elements of the bargain and that in the absence of such limitations, the financial and other terms of this Agreement would be substantially different. Some jurisdictions do not allow the exclusion or limitation of incidental or consequential damages, so these limitations may not apply to You.
7. **Notice to United States Government End Users.** If You are the U.S. Government or if You are a contractor or subcontractor (at any tier) of the U.S. Government and are licensing the Matterport Dataset for use by the U.S. Government or in connection with any contract or other transaction with the U.S. Government, You acknowledge that by installing and using the Matterport Dataset, the Matterport Dataset qualifies as commercial computer software and that any associated documentation qualifies as commercial computer software documentation within the meaning of the applicable acquisition regulations. The terms and conditions of this Agreement are fully applicable to the Government’s use and disclosure of the Matterport Dataset and associated documentation, and shall supersede any conflicting terms or conditions.

8. **Compliance with Law.** You represent and warrant that: (a) You are not located in a country that is subject to a U.S. Government embargo, or that has been designated by the U.S. Government as a “terrorist supporting” country; and (b) You are not listed on any U.S. Government list of prohibited or restricted parties. You may not export, re-export, import, or transfer the Matterport Dataset in violation of any applicable export laws or regulations, and You may not assist or facilitate others in doing any of the foregoing. You acknowledge that it is Your responsibility to comply with any and all applicable export and import laws.
9. **Governing Law; Venue.** This Agreement shall be interpreted in accordance with the laws of the state of California without reference to its conflict of law provisions. Any litigation, suit or other proceeding regarding the rights or obligations of the parties hereunder shall be conducted exclusively before the state and federal courts in and for Santa Clara County, California, and the parties specifically consent to Santa Clara County, California, as the exclusive venue for any such proceeding. This Agreement shall not be governed by the United Nations Convention on Contracts for the International Sale of Goods, the application of which is expressly excluded.
10. **Miscellaneous.** If any provision of this Agreement is held to be unenforceable, such provision shall be reformed to the extent necessary to make it enforceable so as to effect the intent of the parties, and the remainder of this Agreement shall continue in full force and effect. A waiver of any default is not a waiver of any subsequent default. You may not assign or otherwise transfer any of Your rights hereunder without Matterport’s prior written consent, and any such attempt is void. This Agreement is binding upon and is for the benefit of the respective successors and assigns of the parties hereto. The parties acknowledge and agree that a material breach of this Agreement adversely affecting Matterport’s proprietary rights would cause irreparable harm to Matterport for which a remedy at law would be inadequate and that Matterport shall be entitled to injunctive relief in addition to any remedies it may have hereunder or at law. This Agreement is the complete agreement between Matterport and You concerning Your use of the Matterport Dataset, and supersedes any and all prior agreements and representations between Matterport and You related to the same subject matter. Unless otherwise specified herein, all notices and other communications required or permitted to be given or made hereunder shall be in writing and: (a) if to Matterport, delivered personally or sent by pre-paid, first class certified or registered mail, return receipt requested or by overnight courier, to Matterport, Inc., 352 E. Java Drive, Sunnyvale, CA 94089, Attn: Legal Notices; or (b) if to You, by email or by pre-paid, first class certified or registered mail, return receipt requested or by overnight courier to Your mailing address. No amendment of any provision of this Agreement shall be effective unless made in accordance with preliminary paragraphs hereof or set forth in a writing signed by a representative of Matterport and You, and then only to the extent specifically set forth therein.

Before we are able to offer you access to the database, please agree to the aforementioned terms of use.

Requester's name: Kirill Rodionov

Requester's email: kirill.rodionov@ut.ee

PI's name: Kallol Roy

PI's email: kallol.roy@ut.ee

Affiliation: University of Tartu

Requester's signature:  Date: 26.02.2023

# Non-exclusive licence to reproduce thesis and make thesis public

I, Kirill Rodionov

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

**“INTUIT-VLNCE: Autonomous Navigation though Vision-and-Language”**

supervised by Kallol Roy

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.
3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.
4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

*Kirill Rodionov*



23.12.2023