Robotic Navigation in Simulated Urban Environments

An investigation on the effect of uncertainty in the observed environment

Supervisor: Ed Manley

## Abstract

# Setting the stage

## Topic and aim

The topic for this dissertation is robotic navigation in simulated urban environments, with the reason being threefold. Firstly, urban environments contain a lot of different interesting challenges, such as static and dynamic obstacles, and the effect of interaction between, along with the relevance and certainty of their position, to mention a few. The mentioned challenges are to be elaborated below, yet these challenges are important for the growing number of *smart cities.* Secondly, movement in urban environments is surrounded by ethical principles, irrespectively of you being a human or a robot. It is a topic to be discussed further below, however, simulated environments present the ability to address ethical concerns in a save manner. *Unity*, introduced doing the programme of Spatial Data Science and Visualisation, provides the software engine to construct such simulated environments. Thirdly, robotic navigation can be done using a variety of methods, of which one is *reinforcement learning* (RL). RL has gained a lot of attention in recent years, mainly because of advances in deep learning, and RL is an active research area within the faculty of CASA, UCL*. Deep reinforcement learning* (DRL), combing the recent advances in deep learning with RL, has enabled super-human performance in certain tasks, yet DRL isn’t the answer for every challenge.

The aim of this dissertation is to explore how the state-of-the-art DRL method for continuous control task, *Proximal Policy Optimisation*, manages some of the challenges present in urban environments. In particular, how uncertainty about the observed environment affects the performance. The simulated environment is a toy model of a real-life urban environment, yet it is constructed with realistic physical settings, possible by using Unity ML-Agents Toolkit by *Unity*, a simulation engine based on state-of-the-art game developing software.

## Introduction

## Literature review

# Prerequisites

## Notation

Words used interchangeably: Sensors/Pedestrians, Robot/Agent

## Delivery Robots today

Delivery robots are not a thing of the future, they are already deployed in a few pilot cities, even in London (Nichols, 2019). An increasing number of companies are putting they attention on autonomous delivery robots, to meet increasing customer expectations of companies to ride the technology weaves, to enable low-cost-low-emission products. 2019 has so far been an exciting year in this matter, with three of the biggest players (Merrit, 2019), within the field of autonomous delivery robots (ADR), launching different initiatives taking effect doing 2019.  
Starship, founded by two of the Skype co-founders, newest launch is autonomous delivery of food and beverages at George Mason University, Maryland. The partnership is to accommodate the rising need for smart solutions in a high-paced-high-expectation environment, where nutrition sometimes is overlooked (Nichols, 2019). Starship fleet of ADR’s has over 100.000 logged kilometres and more than 25.000 deliveries under the wheels, and been deployed in cities like London, New York and Washington, DC (Merrit, 2019; Nichols, 2019). Starship’s fleet has gained enough experience to surpass the need for any handholding (Nichols, 2019), which entitles them as the leader in the ADR race.

Another two contenders are two well-known giants, Amazon and FedEx, which both revealed their ADR in the first quarter of 2019 aiming at lunching in pilot cities around this time now (FedEx, 2019; Scott 2019).

All three contributions are equally interesting for this project because they appear, as little information is revealed at this stage in the race, to have same specifications (FedEx, 2019; Starship, 2019; Scott, 2019). Them having similar specifications provides guidance on the specifications of the agent, central for the simulations in Unity. For now, it is noted that the ADR’s appears to have a height around 0.5-1.5 metre, a width and depth of 0.5 metre, a total weight (including cargo) of 45-50 kgs and a speed around 5 km/h (FedEx, 2019; Starship, 2019; Scott, 2019).

## Unity – as a simulation engine for research in DRL

Unity is best described as being a multi-functional platform, enabling development of everything from complex high resolution multiplayer games to less complex mobile games over to VR/AR applications, and increasingly as a challenging set-up to conduct research on artificial intelligence (AI) (Juliani et al, 2018; OpenAI, 2018; Sadeghi & Levine; 2016).

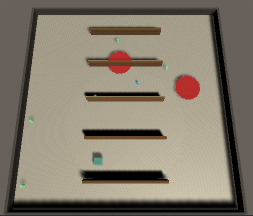
With increasing complexity, sometimes in an exponential manner, comes a need for low latency and distributed computing. Irrespectively of the increasing complexity being a derivative of a high-resolution 3D game consisting of, potentially, thousands of players with widespread interaction patterns, or research on algorithms that roughly needs 18 million video frames to surpass human performance (Hessel et al, 2017), satisfies Unity the need (Juliani, 2019). The underlying engine runs in de-synchronized fashion, supporting simulations at run times at least 100 times faster than real time. This is possible while still maintain physics and frame rendering. A final appealing feature of the computational side is the ability to run concurrent training session, internally in one application as well as externally, enabling the possibility to utilise cloud computing for further increased computational power.

The diversity in the application areas implies that Unity is a serious candidate for modelling the complex dynamics of urban environments. The most appealing feature is the ability to replicate real-life physical complexity, and thereby enabling realistic movement patterns as well as interaction between objects. This is an important factor with an underlying interest in enabling the results to generalise to the real-world, as higher similarity between the environment within the results are obtained and the environment in which the results are deployed, increases the likelihood for generalisation. Another feature of Unity, which makes it appealing for modelling urban dynamics, is the possibility to model complex social interaction. Unity comes with two options to add layers of social interaction to the environment. The *NavMesh* class provides the ability to add AI agents to the environment, useful for spatial queries, as pathfinding. As described in detail later, this class is utilised to model pedestrian behaviour in the constructed urban environment for this paper. NavMesh agents can interact with other NavMesh agents, as well as avoid other moving obstacles, enabling a layer of social interaction with relative ease.   
A more challenging, and perhaps are more interesting way to add social interaction to the environment is by utilising the ML-Agents toolkit, a central part of this paper. The ML-Agents toolkit bridges sophisticated machine learning methods with the graphical interface and complex physical engine of the traditional Unity application, enabling a new setting to push the boundaries for DRL research (Juliani, 2018). The toolkit allows researcher to utilise pre-defined algorithms, based on TensorFlow, or define them themselves, via a Python API. In the light of the NavMesh class, the toolkit puts the control of the interaction in the hands of the researcher, which carries a certain responsibility. It requires the researcher to exhibit a greater understanding of the task and modelling at hand, and so reduces the possibility of headless simulation – limiting the risk of another black box appearance.  
With the introduction of ML-agents, can Unity partly be regarded as the new kid in the class of software’s usable for Agent-Based modelling (ABM). The ML-Agents toolkit makes Unity an appealing contender to the traditional software’s used for ABM, by the fact that the scripting languages are C# and Python. The use of C# and Python implies low latency along with a wide variety of options for further data processing, through open source libraries, and user support, from the enormous communities surrounding the two languages.

Consider the ending of this section, if it should be changed/extended a bit.

### The environment

The environment explored in this paper is seen in figure 1, and it is intended to simulate areas of a city which are less trafficated with cars, and more dominated by pedestrians, formally known as pedestrian streets. An example hereof could be the area around Carnaby Street in Soho, London. Simulating a network of pedestrian streets is justified by the current state at which ADR’s are today, i.e. still being in an early stage and facing challenges in the unpredictability of the real world (Nichols, 2019).

Figure 1 –   
  
*The environment contains five moving pedestrians (light green), two high-density areas (red), the target (dark green square) and the five static obstacles.*

Developing an environment, formally known as a *Scene* in Unity, to facilitate the possibilities within the ML-agents toolkit requires some basic objects; an actual *environment* to explore, an *academy*, an *agent* and a *target*. With the presence of these objects one gets something similar to the environment seen in figure 2, which on top includes walls to prevent the agent from falling of the surface – yet these are not strictly needed, as to be elaborated on shortly when digging into the set-up of the *agent* component. Figure 2 is an example of how everything could be organised within a scene.

*Figure 2 – A scene in Unity containing the components of the ML-Agents toolkit*A close up of a screen

Description automatically generated  
*The scene contains the base elements of the environment.*

#### Environment

#### Academy

to bridge actions and observations from the actual environment to the TensorFlow-based models in Python

#### Agent

to perform the action and collect the observations

#### Target

hej

g

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Table 1. Environment configuration (200.000 steps)** | | | | | | | | |
| Speed | Decision Interval | Time Scale | Training time | Deviation\* | ACR: Mean\*\* | ACR: Std. dev. | AEL: Mean | AEL: Std. Dev. |
| 2 | 5 | 60 | 1650 | 22.2% | 0.663 | 0.65 | 40.32 | 19.41 |
| 1 | 5 | 60 | 1500 | 11.1% | 0.920 | 0.59 | 39.55 | 15.59 |
| 1 | 3 | 60 | 1350 | 0.0% | 0.916 | 0.80 | 101.31 | 198.75 |
| 1 | 7 | 60 | 1750 | 29.6% | 0.896 | 0.63 | 34.07 | 15.86 |
| 1 | 10 | 60 | 1800 | 33.3% | 0.812  **Re-create in Tex** | 0.56 | 26.22 | 9.32 |
| 1 | 5 | 100 | 1350 | 0.0% | 0.908 | 0.62 | 43.03 | 19.58 |
| 1 | 5 | 20 | 2020 | 49.6% | 0.908 | 0.57 | 44.02 | 21.15 |
| \*: Relative to the fastest, \*\*: Mean of converged path | | | | |  |  |  |  |

### Tensorboard

Inspired by (Juliani, 2018b)

## Q-learning

## Trust Region Policy Optimisation

## Proximal Policy Optimisation

## Curriculum Learning

Specifying curriculums in Unity is done in a json file, seen in figure 3, and contains five different parameters

# Exploration/Exploitation trade-off

Coming of configuring the set-up, it is time to explore the environment, address the challenges and hopefully solve the environment.  
When is the environment *solved*? The environment is solved when the agent can cope with the different type of challenges that the environment processes. As mentioned in section X, the environment of this paper processes four main challenges; difficult areas, static obstacles, crowded areas and dynamic obstacles.

The first relevant question to address is, *what is the look of learning*?   
Learning comes with many looks, and a wide range of relevant statistics from training is visualised via TensorBoard. For a description of how each statistic expresses whether learning has occurred, see section X. The most intuitive statistics is the average cumulative reward (ACR) and the average length of an episode (ALE), examples are seen from figure 2.  
The ACR is expected to converge to the maximum obtainable level of reward within a single episode, when learning has taken place and the agent has solved the environment. The ALE should converge to the minimum number of steps needed to obtain the reward. The speed of which the converge takes place is a direct consequence of the complexity of the environment and the degree of assistance provided to the agent, as will be clear doing this section.

*Figure 2 – The simplest version of the environment.*

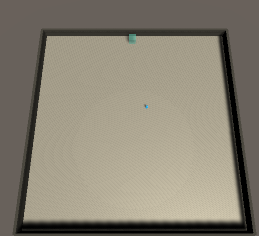
   
*The simple environment contains only a target (darker green square) and the agent (light blue).*

Figure 2 shows the simplest possible environment, namely without any obstacles. The target is here in the upper middle of the environment, yet it changes position every episode. Learning in this environment is straight forward, and this is seen from figure 3. Figure 3 shows how ACR and ALE converges rapidly to their optimal levels, which are obtained after around 75.000 steps in this case.

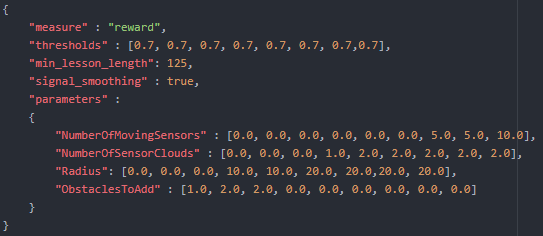
*Figure 3 – ACR and ALE for the environment from fig.2*  
A screenshot of a cell phone

Description automatically generated *The average cumulative reward and average episode length obtained from learning in the simple environment from figure 2.*

The shapes of ACR and ALE seen in figure 3 are ideals, and far from trivial to obtain with the slightest degree of complexity present. The road to meaningful learning requires careful design of the aid provided to the agent to ensure generalisation and so providing the agent with the ability to handle unseen environments, which is one of the aims of this paper. Figure 4 shows the same output, as in figure 3, for the far more complex environment seen in figure 1

*Figure 4 – ACR and ALE for sample environment b*

The first step towards improving learning is to introduce a curriculum, for the agent to learn from, to gradually add complexity to the environment, as the agent learn to cope with novel challenges. An interesting aspect is how the curriculum should be designed, in order to facilitate optimal learning of the agent, and this is almost an area for research itself. The aim of this paper is not to shed light on optimal construction of curriculums, and the curriculum used here is chosen such that it addresses three of the four main challenges equally, namely the static obstacles, the crowded areas and the dynamic obstacles. The difficult areas a separate task, which will be addressed shortly.  
The initial curriculum used in this paper is seen from figure 5, and the reader should revisit section X.Y for a full discussion of each of the parameters.

*Figure 5 – The initial curriculum used to facilitate improved navigation of the robot.*

Starting out with the parameters specified in the curriculum, it is noticed that there is at least one parameter for each of the obstacles for the robot to avoid. The curriculum is designed such that the obstacles are introduced relative to their degree of complexity. The first challenge is the static obstacles, then the semi-static crowded areas and finally the dynamic pedestrians. Each type of obstacle is introduced over three lessons, and the same threshold to be passed is required. The threshold is at this point not fixed, as it should be set based on the two parameters of the environment, which will be addressed below. Signal smoothing and the minimum number of episodes, in which the threshold is surpassed, are here chosen with the same objective, so sustain generalisation. Signal smoothing is enabled to ensure robustness of the signal. The minimum number of episodes are chosen such that the number ensures robustness, yet not to such a degree that it becomes a burden. The minimum number of episodes should be chosen with the threshold in mind. Too high a threshold compared with too high a requirement on the robustness of the signal, will likely prevent any progress, depending on the complexity of the environment. Choosing too low a threshold and too low a level of the robustness, can imply that progress occurs too soon. A common misconception when working with reinforcement learning is the amount of data needed to sustain meaningful learning, the planning fallacy as of (Irpan, 2018). Figure 6 shows ACR and ALE, along with the changes in lessons[[1]](#footnote-1), for a training session with the initial curriculum.

Figure 6 - *ACR and ALE for first training session using curriculum learning*

As discussed in section X, is the ability of the agent to generalise to unseen environments one of the

Figure 3 illustrates the aim at the end of this section, however, the challenges of the environment will blur these graphs

## Insights:

*Randomization of rewards usually poses a problem for the agent - it doesn't know what to do because it doesn't know what reward to expect.*

*.. the agent can't learn a perfect model of how observations correspond to rewards (i.e., they're "noisy")*.

Source: <https://github.com/Unity-Technologies/ml-agents/issues/1680>

## Stuff for the discussion:

As RL is nothing more than a function mapping, either exact or approximative, it follows sort of trivially that randomness makes the mapping blurry, which is the reason why the agents has a hard time dealing with the sensor clouds.

## Thoughts:

If it turns out that, no matter the actions taken, it seems impossible for the agent to learn avoiding the sensor clouds, presumably because of the randomness in the penalties of the steps (as derivative of above), one idea can be to let the penalty be fixed but use the density to change timescale/speed.

## Policy Evaluation

References:

Juliani, A. et al. (2018a). Unity: A General Platform for Intelligent Agents.

Juliani et al. (2018b), *Using TensorBoard to Observe Training*. Online resource, url: <https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Using-Tensorboard.md>. Visited: 14/06/2019.

Irpan, Alex. (2018). Deep Reinforcement Learning Doesn’t Work Yet. Online resource, URL: <https://www.alexirpan.com/2018/02/14/rl-hard.html>. Visited: 23/07/2019.

Merrit, Tom. (2019). Top 5 Delivery Robots. TechRepublic, URL: <https://www.techrepublic.com/article/top-5-delivery-robots/>.

Nichols, Greg. (2019). Delivery Wars: Amazon’s new delivery robot vs. Starship’s college munchie robot. ZDNet, URL: <https://www.zdnet.com/article/amazons-new-delivery-robot-vs-starships-college-munchie-robot/>.

FedEx. (2019). Delivering the Future: FedEx Unveils Autonomous Delivery Robot. FedEx newsroom, URL: <https://about.van.fedex.com/newsroom/thefuturefedex/>.

Starship. (2019). The Self-Driving Delivery Robot. Starship/About, URL: <https://www.starship.xyz/business/>.

Scott, Sean. (2019). Meet Scout. The Amazon blog; dayone, URL: <https://www.starship.xyz/business/>.

1. Each lesson is an entity in the arrays of the parameters in the curriculum. [↑](#footnote-ref-1)