

**Project Acronym:** STAR  
**Grant Agreement number:** 956573 (H2020-ICT-2020-1 – Research and Innovation Action)  
**Project Full Title:** Safe and Trusted Human Centric Artificial Intelligence in Future Manufacturing Lines  
**Project Coordinator:** Netcompany-Intrasoft



Funded by the Horizon 2020  
Framework Programme of the  
European Union

## DELIVERABLE

### D5.8 – Reinforcement Learning Techniques for AMRs-Final version

<b>Dissemination level</b>	PU -Public
<b>Type of Document</b>	Report
<b>Contractual date of delivery</b>	31/03/2023
<b>Deliverable Leader</b>	THALES SIX GTS FRANCE
<b>Status - version, date</b>	V1.0, 17/07/2023
<b>WP / Task responsible</b>	WP5/T5.4
<b>Keywords:</b>	Reinforcement Learning, Simulation, Mobile robots, Fleet management

*This document is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 956573. It is the property of the STAR consortium and shall not be distributed or reproduced without the formal approval of the STAR Management Committee. The content of this report reflects only the authors' view. The European Commission is not responsible for any use that may be made of the information it contains.*

## Executive Summary

This deliverable describes the final version of our demonstrator of a Reinforcement Learning (RL) based Automatic Mobile Robots (AMRs) Fleet Management System (FMS). This document is an update of the previous D5.7, with new sections and updated parts.

The use of such technology comes from the need to adapt to an always-evolving environment, in particular because of human presence. By leveraging video analytics deployed in the factory and developed in T5.3, the FMS will be able to take into account areas to avoid when dispatching AMRs, at a planning level. AMRs will still embed integrated collision avoidance mechanisms for last second avoidance. Our focus is to improve collision avoidance with increased anticipation: the avoidance should now be seen as a part of planning level and not only to a reactive level anymore. This is a light shift from our previous vision of our work, described in D5.7.

In our work, simulation of the factory is key. It is an essential component for the following reasons:

- It allows to produce an annotated data set to train Machine Learning Techniques to anticipate the evolution of human presence in the factory.
- It allows demonstrating our solution on a large-scale scenario, which are not available in the scope of the STAR project.
- It brings technical primitives such as pathfinding that can be used as a high-level action in the RL FMS.
- It allows to train a Reinforcement Learning agent to act as an FMS.

<b>Deliverable Leader:</b>	Bertrand Duqueroie (THA)
<b>Contributors:</b>	Christos Emmanouilidis (RUG) Alexandre Kazmierowski (THALES)
<b>Reviewers:</b>	Irene Zattarin (GFT) Spyros Theodoropoulos (UPRC)
<b>Approved by:</b>	Charalampos Ipektsidis (INTRA)

<b>Document History</b>			
<b>Version</b>	<b>Date</b>	<b>Contributor(s)</b>	<b>Description</b>
0.1	20/06/23	Bertrand Duqueroie (THA)	Draft
0.2	29/06/23	Bertrand Duqueroie (THA) Alexandre Kazmierowski (THA)	Draft
0.3	03/07/23	Christos Emmanouilidis (RUG)	Ready for internal review
0.4	05/07/23	Spyros Theodoropoulos (UPRC)	Internal Review 1
0.5	06/07/23	Irene Zattarin (GFT)	Internal Review 2
0.6	07/07/23	Christos Emmanouilidis (RUG)	Corrections in section 4 following internal reviews
0.7	10/07/23	Bertrand Duqueroie (THA)	Corrections following internal reviews
0.8	13/07/23	Bertrand Duqueroie (THA)	Ready for submission
1.0	17/07/23	Charalampos Ipektsidis (INTRA)	QA-ed version to be submitted

# Table of Contents

<b>EXECUTIVE SUMMARY .....</b>	<b>2</b>
<b>TABLE OF CONTENTS.....</b>	<b>4</b>
<b>TABLE OF FIGURES.....</b>	<b>5</b>
<b>LIST OF TABLES.....</b>	<b>6</b>
<b>DEFINITIONS, ACRONYMS AND ABBREVIATIONS .....</b>	<b>7</b>
<b>1 INTRODUCTION.....</b>	<b>8</b>
1.1 APPROACH .....	8
1.2 OVERVIEW .....	9
<b>2 HEATMAP CONCEPT .....</b>	<b>10</b>
2.1 HEATMAP DESCRIPTION .....	10
<b>3 SIMULATION CAPABILITIES .....</b>	<b>13</b>
3.1 GENERIC SIMULATION ENGINE .....	13
3.1.1 <i>Virtual workers</i> .....	14
3.1.2 <i>Equipment</i> .....	14
3.1.3 <i>Satisfying needs &amp; Scripts: the relation between virtual actors and equipment</i> .....	15
3.1.4 <i>Logging</i> .....	15
3.2 HEATMAP SIMULATED SAMPLES .....	15
3.3 GENERALISATION AND REALISM .....	18
3.4 PROCEDURAL GENERATION (NEW).....	20
<b>4 HEATMAP FORECASTER.....</b>	<b>23</b>
4.1 HUMAN TRAJECTORY PREDICTION .....	23
4.1.1 <i>Physics-based methods</i> .....	24
4.1.2 <i>Planning-based methods</i> .....	25
4.1.3 <i>Pattern-based methods</i> .....	25
4.2 CONVOLUTIONAL NEURAL NETWORK – BASED APPROACHES .....	25
4.2.1 <i>How the problem is posed</i> .....	25
4.2.2 <i>CNN structure</i> .....	26
4.2.3 <i>Performance Assessment</i> .....	26
4.2.4 <i>Results</i> .....	26
4.3 GRAPH NEURAL NETWORK– BASED APPROACHES .....	28
4.3.1 <i>How the problem is posed for a graph representation</i> .....	28
4.3.2 <i>The Graph network</i> .....	30
4.3.3 <i>Model training and performance metrics</i> .....	31
4.3.4 <i>Results</i> .....	32
4.3.5 <i>Learnable Edge Weights</i> .....	32
4.3.6 <i>Model Performance Over Time</i> .....	34
4.3.7 <i>Contextual Node Features</i> .....	35
<b>5 AMR FLEET CONTROLLER .....</b>	<b>38</b>
5.1 OBJECTIVES OF THE FLEET.....	38
5.2 PATHFINDING.....	38
5.3 RL PATH OPTIMISATION.....	40
<b>6 NEXT STEPS AND CONCLUSION .....</b>	<b>46</b>
<b>REFERENCES .....</b>	<b>47</b>

# Table of Figures

FIGURE 1: OVERVIEW OF THE APPROACH ..... 9

FIGURE 2: TOY EXAMPLE .....10

FIGURE 3: 2D GRID STRUCTURE .....11

FIGURE 4: FROM GRID TO HEATMAP.....11

FIGURE 5: HEATMAP RAW LOG.....12

FIGURE 6: SE-STAR SIMULATION TOOL PRESENTATION .....13

FIGURE 7: FACTORY SIMPLE EXAMPLE .....15

FIGURE 8: SIMULATED WORKERS.....16

FIGURE 9: GEOMETRIC LAYOUT .....17

FIGURE 10: LAYOUT CONFIGURATION FILE .....18

FIGURE 11: FROM WORKER SIMULATION TO AN AVERAGE HEATMAP .....18

FIGURE 12: GENERALISATION .....19

FIGURE 13: EXAMPLES OF WAVE FUNCTION COLLAPSE USAGES .....20

FIGURE 14: FACTORY LAYOUTS GENERATED WITH WAVE FUNCTION COLLAPSE .....20

FIGURE 15: DFKI REAL ENVIRONMENT .....21

FIGURE 16: TRAINING WORKFLOW .....21

FIGURE 17: SINGLE AND MULTI-MODEL PHYSICS-BASED PREDICTION .....25

FIGURE 18: PREDICTION HEATMAP FOR THE NEXT SECOND 50 WORKERS; THRESHOLD SET TO 0.6.....27

FIGURE 19: VISUAL REPRESENTATION OF AN EXAMPLE GRID-TO-GRAPH CONVERSION OF A 3x3 GRID.....29

FIGURE 20: VISUALISATION OF HOW SEQUENCES OF GRAPH INPUTS ARE LINKED WITH ACTUAL OUTPUTS .....30

FIGURE 21: MODEL’S ARCHITECTURE.....31

FIGURE 22: HISTOGRAM OF LEARNED EDGE WEIGHTS SHOWING THE DISTRIBUTION OF THE LEARNED WEIGHTS. ....33

FIGURE 23: THE BALANCED ACCURACY OVER TIME FOR DIFFERENT THRESHOLDS. ....34

FIGURE 24: PREDICTED HEATMAPS FOR DIFFERENT TIME PERIODS .....35

FIGURE 25: PREDICTED OCCUPANCY GRIDS FOR DIFFERENT THRESHOLDS AND TIME PERIODS.....35

FIGURE 26: AREA TO AVOID (THE HOTTER THE COLOUR, THE MORE IT NEEDS TO BE AVOIDED).....38

FIGURE 27: DISTANCE FIELDS TO SEVERAL DESTINATIONS .....39

FIGURE 28: ADDING FIELD DISTANCE TO RL INPUT .....40

FIGURE 29: THE REINFORCEMENT LEARNING LOOP.....40

FIGURE 30: USING RL IN THE STRATEGIC PART .....42

FIGURE 31: RL FOR PATH FOLLOWING .....42

FIGURE 32: RL ROBOT GUIDANCE .....43

FIGURE 33: RL GUIDANCE FIRST RESULTS .....43

FIGURE 34: REPLACING PATH GUIDANCE BY DISTANCE MAP GUIDANCE .....44

FIGURE 35: USE OF DISTANCE MAP .....45

FIGURE 36: LEARNING CURVES .....45

FIGURE 37: HUMAN DIGITAL TWIN FRAMEWORK .....46

## List of Tables

TABLE 1: CNN-BASED HEATMAP OVERALL PREDICTIVE PERFORMANCE FOR DIFFERENT THRESHOLDS .....	27
TABLE 2: CNN-BASED HEATMAP PREDICTIVE PERFORMANCE FOR DIFFERENT TIME HORIZONS (50 WORKERS).....	28
TABLE 3: CNN-BASED HEATMAP PREDICTIVE PERFORMANCE FOR DIFFERENT TIME HORIZONS (100 WORKERS) .....	28
TABLE 4: CLASSIFICATION RESULTS COMPARING LEARNABLE WEIGHTS WITH AN UNWEIGHTED GRAPH .....	33
TABLE 5: REGRESSION RESULTS WITH DIFFERENT CONTEXTUAL INFORMATION.....	36
TABLE 6: CLASSIFICATION RESULTS WITH DIFFERENT CONTEXTUAL INFORMATION .....	36

## Definitions, Acronyms and Abbreviations

Acronym/ Abbreviation	Title
<b>AI</b>	Artificial Intelligence
<b>AMR</b>	Automatic Mobile Robot
<b>CNN</b>	Convolutional Neural Network
<b>FMS</b>	Fleet Management System
<b>FN</b>	False Negatives
<b>FP</b>	False Positives
<b>GCN</b>	Graph Convolutional Network
<b>GCNN</b>	Graph Convolutional Neural Network
<b>GNN</b>	Graph Neural Networks
<b>GRUs</b>	Gated Recurrent Units
<b>ML</b>	Machine Learning
<b>MSE</b>	Mean Squared Error
<b>ORCA</b>	Optimal Reciprocal Collision Avoidance
<b>ReLU</b>	Rectified Linear Activation Unit
<b>RL</b>	Reinforcement Learning
<b>RMSE</b>	Root Mean Squared Error
<b>RNN</b>	Recurrent Neural Network
<b>RVO</b>	Reciprocal Velocity Obstacle
<b>SSO</b>	Service Oriented Object
<b>STGNN</b>	Spatio Temporal Graph Neural Network
<b>TGNN</b>	Temporal Graph Neural Network
<b>TN</b>	True Negatives
<b>TP</b>	True Positives
<b>VCA</b>	Video Content Analytics
<b>WP</b>	Work Package

# 1 Introduction

## 1.1 Approach

The STAR project will bring new capabilities for dynamic and adaptive Automatic Mobile Robots Fleet management. This will allow the use of such kinds of robots in a changing environment, without the need for time consuming configuration.

Modern factories rely more and more on mobile robots to perform logistic tasks. They are efficient means to move goods from one place to another within the factory. However, they currently require a specific configuration to be used safely within an environment with human workers. Before installing a Robot Fleet, a digital map of the environment needs to be created. Possible paths for the robots are designed once and for all. This kind of solution is adapted when the factory layout and working processes are fixed.

STAR is developing new technologies relying on Artificial Intelligence and simulation to bring the adaptive capabilities needed to use Robot Fleet in wider types of factory environments. Our solution consists in:

1. Creating an updated digital view of the environment, thanks to low-cost cameras deployed in the factory and advanced Machine Learning to analyse the situation
2. Anticipating human movements within the factory, thanks to Machine Learning trained on a huge set of factory data, created by simulation.
3. Optimising “on the fly” Robot Fleet commands to adapt to the current layout of the factory and human workers’ behaviors, thanks to Machine Learning trained by trials and errors, within simulation (Reinforcement Learning).

To keep the cost of our solution low, we rely on a few standard cameras deployed in the factory instead of adding expensive sensors embedded on the robots. Moreover, the occlusion that is faced by embedded sensors will not limit our global situation awareness. Having a clear picture of where the obstacles are (that could be temporarily left on possible paths), and analyzing human presence is mandatory to optimise efficiently robot fleet behavior.

The anticipation of human presence within the factory is a challenge on its own. Machine Learning is key to extrapolate the current situation to the near future. Like always in Machine Learning, training is crucial. Thanks to our simulation capabilities, we can simulate a huge variety of factory layouts and working processes that will encompass the situations encountered, when our solution will be deployed in a real factory.

Finally, yet importantly, we compute optimised Robot Fleet commands, sending the more suitable robot with the most efficient and safest path. Constraints on the fleet can be hard, because of the dynamic environment and because of the need to avoid as much as possible interfering with human movements. Moreover, optimised commands must be very fast to compute, to adapt to the ever-evolving situation within the factory. Once again, Machine Learning, and in particular Reinforcement Learning, is the solution we develop. This kind of solutions have been made famous thanks to impressive results where AI beats professional human players at different games such as Chess, Go, or StarCraft2. In the solution we develop the AI trains itself in an interactive and fast simulation through a very large number of runs. Thanks to this huge training, it will be able to cope with the large diversity of real situations.

## 1.2 Overview

The goal of Task 5.4 is to produce a controller for the whole fleet of robots, using new AI approaches and leveraging simulation capabilities.

Since the simulation must represent the key aspect of the environment, we defined a data format called Heatmap that captures all the key aspects of the factory at any point in time. This data representation can also be seamlessly filled with information coming from video analytics plugged to real cameras of a real factory. This HeatMap makes the link with the previous STAR Task 5.3 and described in deliverable D5.5 “Visual Scene Analysis for Safety Zones Detection”.

Therefore, the next section will focus on describing the Heatmap data structure.

Then section 3 will focus to the simulation capabilities available in our T5.4 and how it is used to generate many different situations/scenarios, which can be used both as a learning data set for ML or as environment for RL.

Section 4 focuses on a first use of simulation and AI for the specific task of short-term forecasting of the Heatmap, which will be used as input to the RL part, improving the anticipation capabilities of the whole developed system.

Finally in section 5, the RL optimisation is described, with the goal to improve robots/human cohabitation.

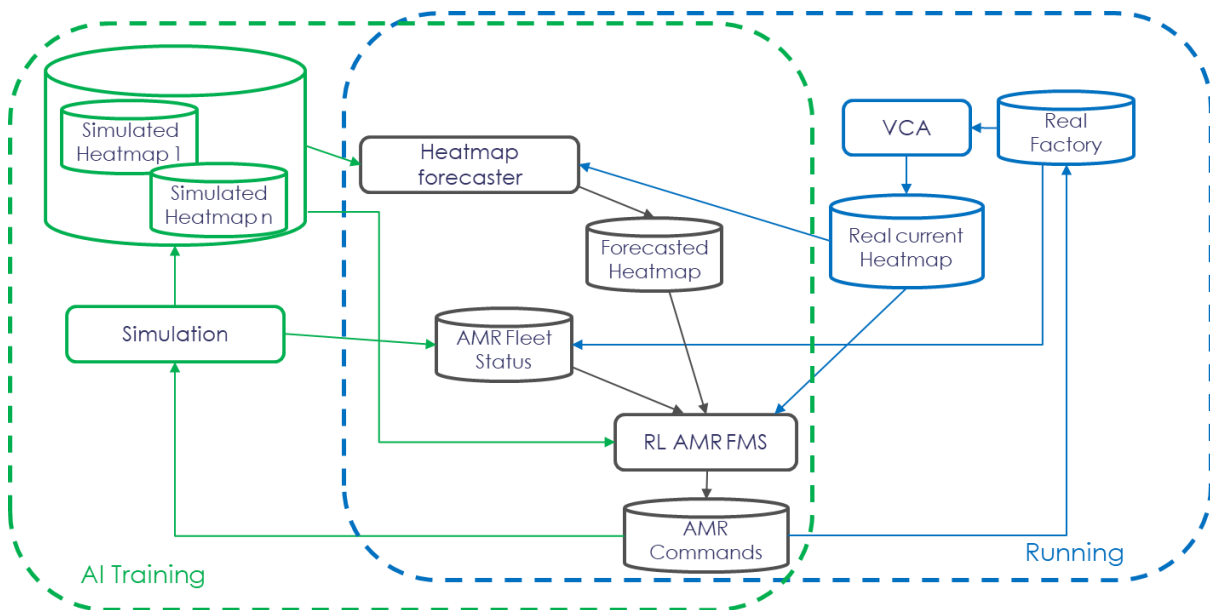


Figure 1: Overview of the approach

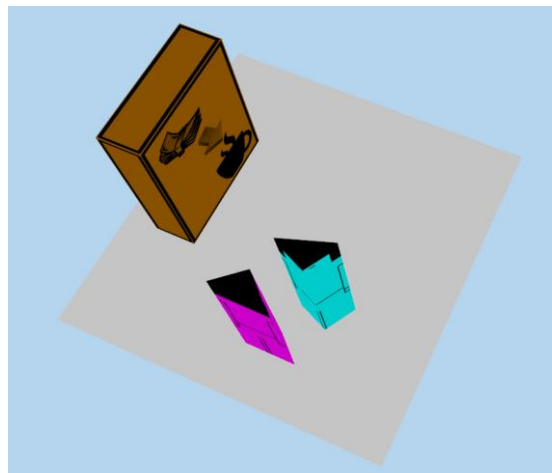
## 2 HeatMap Concept

The Heatmap is a data representation that aggregates all extracted information from all cameras in a factory (thanks to Machine Learning tools from T.5.3). It is a spatial representation of the whole factory, describing where people and obstacle are. It is used as input by Reinforcement Learning that will perform AMRs Fleet control: computing optimised paths for all AMRs, depending on the current need of the fleet, and depending on the current positions of humans and obstacles.

The Heatmap forecasting component (ML based, trained with simulated data set) enhances the RL anticipation capabilities, allowing it to predict where humans will be a few minutes later, when the AMRs will advance, following the orders coming from the RL Fleet controller.

### 2.1 HeatMap Description

To describe this data format, let's take a toy example, where two humans and an obstacle are contained in a small square environment.



*Figure 2: Toy example*

A Heatmap can be created to describe this environment, thanks to camera analytics. Note that the positions of the cameras are not relevant. We suppose that cameras can convert their output (localisation of human and object in their field of view) to a global coordinates system. If several cameras' field of view overlap, their results should be merged when producing the Heatmap. If there are some blind spots due to a lack of camera or to visual occlusion, then we may have empty data on some parts of the Heatmap. It is also possible as an extension (out of the scope of the STAR project) to deal with other types of localisation systems.

The Heatmap is basically a regular grid which contains, in each of its cells, what is contained inside.

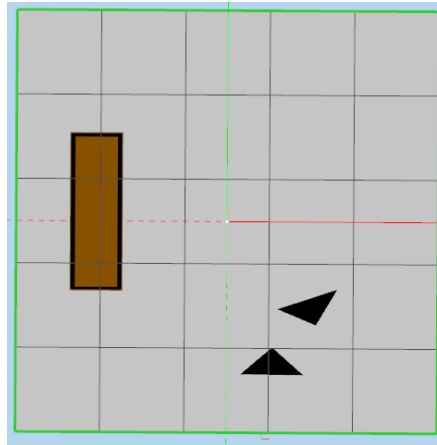


Figure 3: 2D Grid structure

From the illustration above we can note that it is a 2D representation. The 4 meters square environment is divided in 25 cells (0.8m size).

Since obstacles and humans have a width (for instance, we approximate human with a disk of 0.3m radius), they can be present in several cells at once, and a cell can contain several things at once, as you can see in the following illustration.

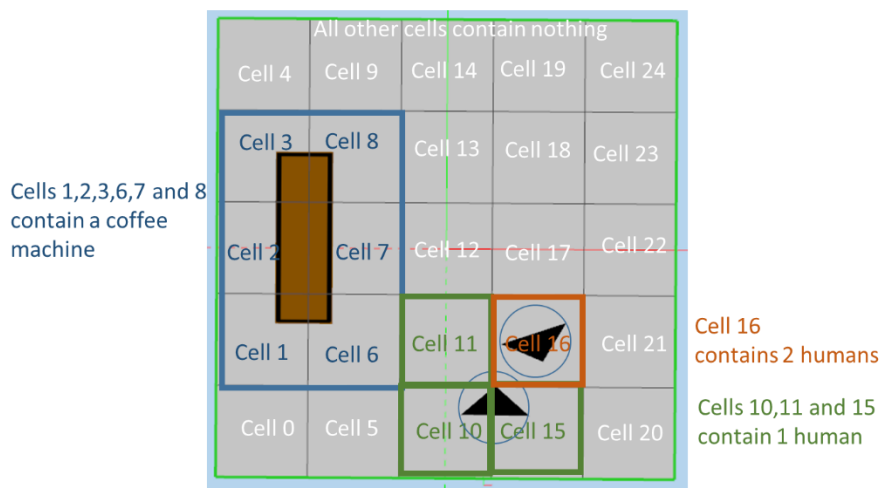


Figure 4: From grid to Heatmap

The Heatmap format will progress in its definition during the project, but a first version is to list each cell from the bottom left to the upper right, and for all of them count how many humans are present. If some obstacles overlap with a cell, they should also be listed.

A first version of Heatmap that corresponds to the above example is given here:

```

1 0
2 0 coffee1
3 0 coffee1
4 0 coffee1
5 0
6 0
7 0 coffee1
8 0 coffee1
9 0 coffee1
10 0
11 1
12 1
13 0
14 0
15 0
16 1
17 2
18 0
19 0
20 0
21 0
22 0
23 0
24 0
25 0

```

*Figure 5: Heatmap raw log*

Note that the cell 0 is written on line 1 of the file (there is an offset of one between the file row number and the cell index).

The number starting each line refers to human presence. "coffee1" describes the fact that the obstacle on the left has been identified as a coffee machine by video analytics. Several objects can overlap on the same cell at the same time, in which case they will be all listed separated by a space. Most of the obstacles do not need to be identified correctly. However, it can be useful to identify working stations, which mean knowing which object can be a destination for human or AMR and where those are located.

A Heatmap captures the information at a specific point in time, and thus, must be timestamped.

Later in the project, Heatmap could contain some more advanced information such as a "probability" of presence, or an average density.

The format by itself may evolve to be more easily processed by STAR components, including the framework used to pass messages between components.

## 3 Simulation capabilities

### 3.1 Generic Simulation Engine

To take into account the inherent dynamics of human worker flows in the factory, the use of crowd simulation software is very helpful. Several tools exist on the market for this purpose such as Oasys MassMotion [REF-01], Onhys [REF-02], or Bentley Legion [REF-03]. While those tools bring more or less easy to use layout design capabilities, pedestrian movement computation and key performance computation such as crowd density, the definition of worker behaviors can be tricky. Moreover, they are not optimised for AI training. In particular, they may lack capabilities to run faster than real time, which is needed to record huge learning data sets of Heatmaps, or for the use as learning environment for RL.

For the STAR project, a new generation of Synthetic Environment engine is used. This engine is based on innovative technologies in the Synthetic Environment field and is developed by Thales. SE-Star, its name, is highly scalable and is able to compute complex and adaptive behaviours as well as low level navigation and environment interactions. With such an engine, it is possible to populate complex critical infrastructure, for instance a subway station or a whole airport, with a realistic crowd of passengers exhibiting context specific decisions and optimising their own motivations. In particular, a factory with workers' movement can be simulated, thanks to STAR T5.4 developments.

SE-Star solution uses four concepts:

- Virtual actors (workers) who move inside the environment (factory);
- Equipment of the environment (such as working station);
- Needs to be satisfied, to relate virtual actors and equipment (such as business process) & Scripting (with randomness)
- Logging

SE-Star is illustrated in Figure 6.

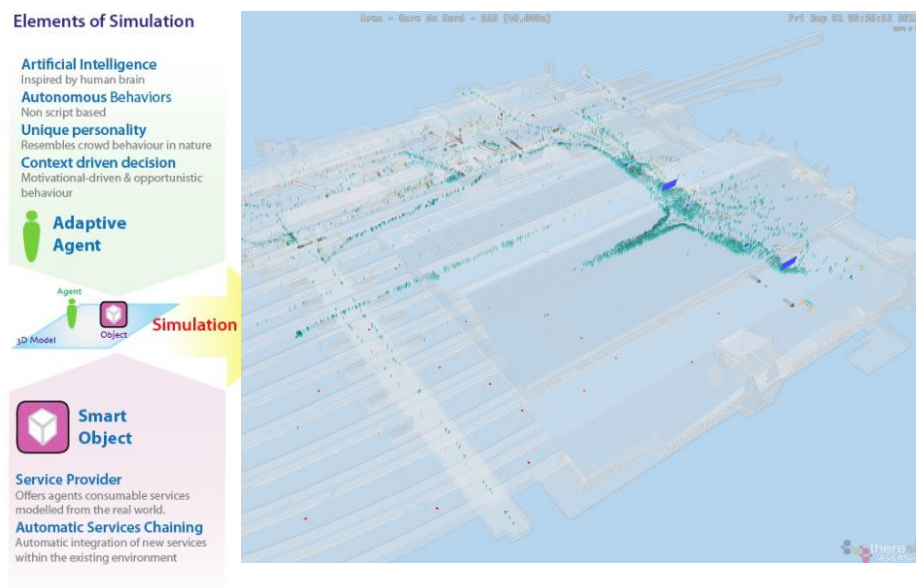


Figure 6: SE-Star simulation tool presentation

### 3.1.1 Virtual workers

The main functionality of SE-Star is to simulate humans as “**virtual actors**”. Thanks to advanced behavioural models, SE-Star provides a very sophisticated set of capabilities in human simulation. Indeed, each virtual actor is a truly situated agent (cf. Animat Approach [REF-04]), meaning it is able to:

- **perceive** its own immediate surrounding environment through different sensors with their precisions and perceptual aliasing: virtual eyes with occultation, virtual ears with hearing illusions and virtual nose.
- **take its own decisions** according to its perceptions and its own internal variables, character traits and motivations.
- **act** in its environment in order to achieve its decisions and satisfy its own needs.

Those high-level capabilities rely on low-level ones, such as pathfinding and collision-avoidance:

- **pathfinding** allows virtual actors to reach the destination they have in mind (such as a group of working station of a precise type). This is a wide scientific domain [REF-10], leading to specific challenges when computation speed is required. Several patents have been submitted on the solution used in SE-Star for pathfinding. In particular, one derived from vector field computation is very useful in the STAR context. This comes from the fact that it relies on a regular grid as input graph to model the environment. Such grid allows to easily make the link with the heatmap concept. Current Heatmap or an average one can be transformed has “costs” for pathfinding. Since this approach can be used as high-level actions for FMS RL, illustrations are given in section 5.2 of this document.
- **collision-avoidance** is also well-studied [REF-06][REF-07], with more or less realistic models. The model used in SE-Star is an approach derivate from RVO/ORCA [REF-08], which allows a good compromise between realism and computation load. Collision-avoidance mechanisms of SE-Star could also be used as part of the controller for the AMR. However, those AMR already embed such mechanisms, with more dedicated solution for robots.

### 3.1.2 Equipment

Besides virtual actors, all the equipment and devices are modelled independently from them. The logic and behaviour of these objects (equipment/device) are encapsulated as services. These objects are named Service Oriented Object (SOO). The set of SOOs constitute a complete world model for this simulation. They can also be used by the virtual actor when they are seeking for a way to satisfy their needs. For instance, when a virtual actor wants to take a coffee, all SOOs coffee machine will provide him the service « coffee ». It will just need to choose the right one, based on proximity and how long the queue may be in front of it. The description of services is done through generic configuration files and include the fact that they may require dependencies: “using” working station of type 1 may require to use a working station of type 2 beforehand. SE-Star is able to represent and simulate a great variety of equipment and devices that can be deployed in any kind of area, or factory layout.

### 3.1.3 Satisfying needs & Scripts: the relation between virtual actors and equipment

Once virtual actors and equipment have been defined, the next step is to connect these two kinds of entities by exploiting the services offered by the objects in order to satisfy the needs of the agents. Here a compromise between AI autonomous realistic behaviors and a more scripted one, derived from behaviours tree [REF-09][REF-10] is used. It allows to define business processes, merged with behaviors that are more natural, and still let emerge a large variety of situations thanks to some random parameters.

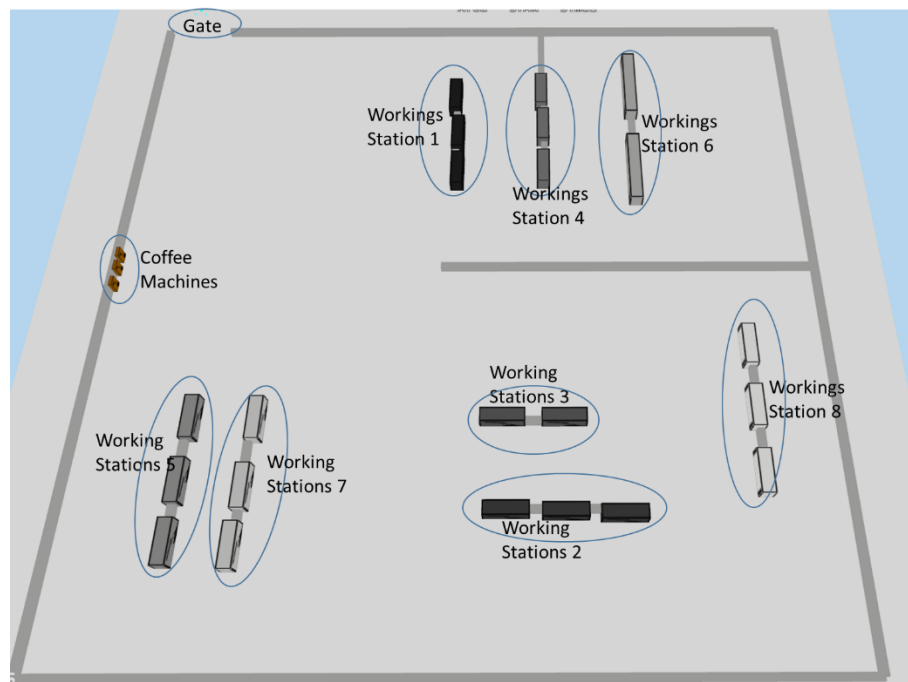
### 3.1.4 Logging

Depending on the use case, different outputs of SE-Star can be useful. In particular, in the context of AI training, such as in the STAR project, the capability to reproduce the outputs of sensors deployed in the environment is key. As such, SE-Star can for instance compute the number of persons and their positions in the field of view of cameras deployed in the simulated infrastructure. It can also record the usage of equipment, such as working stations. Thus, it can easily provide an annotated learning data set, through log files or in a Redis database in case of large volume. In the STAR project, the output of camera is processed by T5.4 VCA to produce a Heatmap, so SE-Star logging capabilities are mostly used to generate plausible Heatmap.

## 3.2 HeatMap Simulated Samples

Several factory-like environments have been simulated in the project. A simple one in particular is taken here to give a more explicit example. This is obviously a fictional factory.

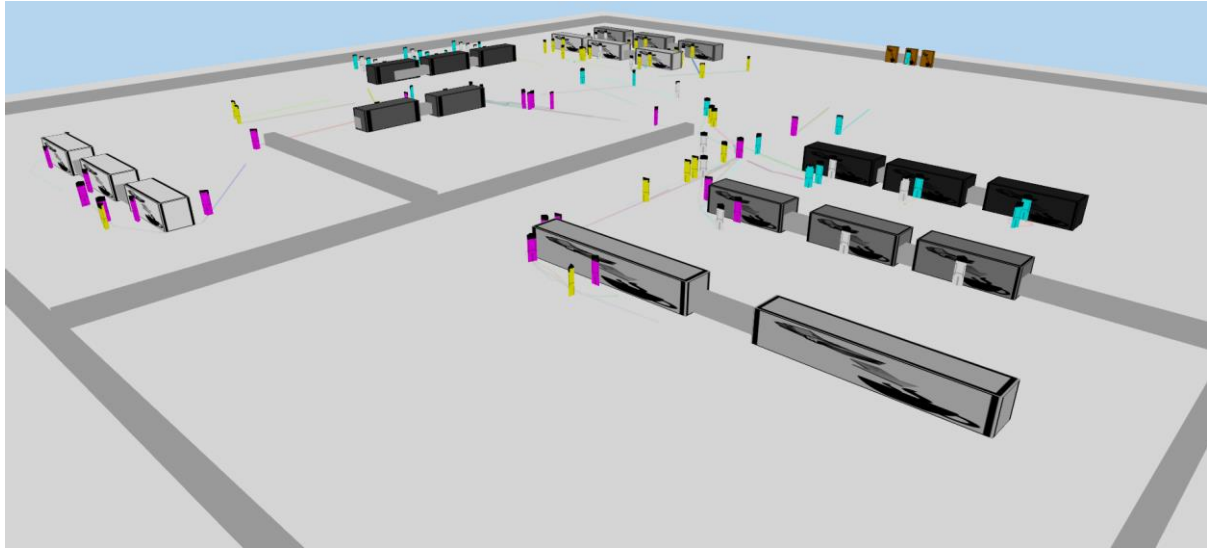
This sample factory is a square of 100 meters wide, with 8 types of working stations, with between 2 or 3 working stations of each type. Three coffee machines are also present, as you can see below in Figure 7.



*Figure 7: Factory simple example*

This fictive factory will then be populated by 50 to 100 workers among 4 profiles (Engineer1, Engineer2, Engineer3 and Engineer4).

For each profile, a description of their working activity, consisting in switching between working stations of different types is provided below.



*Figure 8: Simulated workers*

Engineer1 (displayed in light blue/green in Figure 8):

1. Go to working station of type 1, then
2. Go to working station of type 2, then
3. Go to working station of type 3.

Engineer2 (displayed in light yellow):

1. Go to working station of type 5, then
2. Go to working station of type 6, then
3. Go to working station of type 7, then
4. Go to working station of type 8.

Engineer3 (displayed in light purple):

1. Go to working station of type 4, then
2. Go to working station of type 3, then
3. Go to working station of type 6, then
4. Go to working station of type 8.

Engineer4 (displayed in light grey):

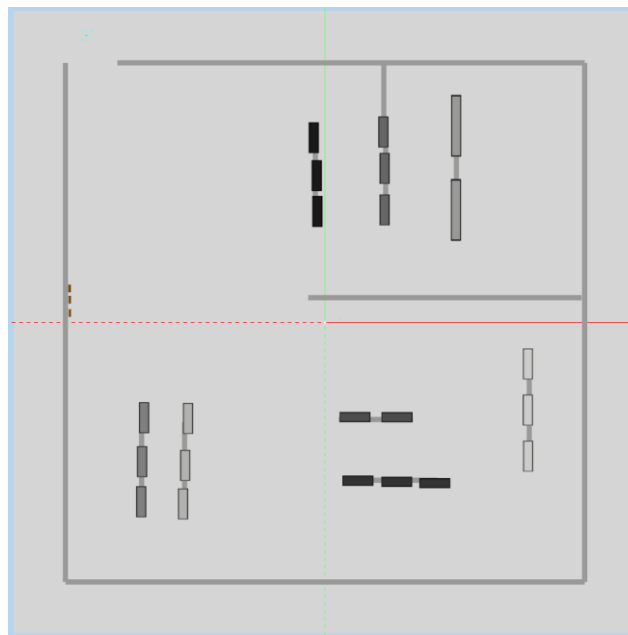
1. Go to working station of type 7, then
2. Go to working station of type 2, then
3. Go to working station of type 4, then
4. Go to working station of type 5, then
5. Go to working station of type 1.

All working stations of a given type are equivalent. Workers will choose the fastest to reach (distance based) by default. If the targeted working station is already occupied, they may select another one, or queue up.

All workers appear initially in the top left area. Before completing a work cycle, they may decide to go take a coffee first (probability of 0.1, which is obviously non-realistic, but enough to create randomness during business processes execution).

Working at station of type 1, 2, 4, 5, 7 and 8 last in average 30 seconds. Working at station of type 3 and 6 last in average 10 seconds. Those timings are set in the configuration file and can be adapted to match a more realistic scenario.

More precise information about the layout of the factory is given below.



*Figure 9: Geometric layout*

The outer walls of the factory correspond to a 100 meters width square (included in a 120 meters width square world). The coordinate system is displayed in red (x axis), and green (y axis), centered in the middle.

A configuration file (Objects.xml) describes the position of all objects (walls, working stations, coffee machines) their orientation, and to some extends their size. An overview is given below:

```
<SmartObjects>
<Object model="Wall" pos="(0.00,-50.00,0.00)" scale="(100.00,1.00,1.00)" yprDegree="(0.00,0.00,0.00)" /><!--bottom wall, 100m long; defaultSize=(1.0,1.0,1.0)meters, but we scale it along x axis to reach 100m -->
<Object model="Wall" pos="(50.00,0.00,0.00)" scale="(100.00,1.00,1.00)" yprDegree="(90.00,0.00,0.00)" /><!--left wall, 100m long, with a 90° rotation -->
<Object model="Wall" pos="(-50.00,0.00,0.00)" scale="(100.00,1.00,1.00)" yprDegree="(90.00,0.00,0.00)" />
<Object model="Wall" pos="(5.00,50.00,0.00)" scale="(90.00,1.00,1.00)" yprDegree="(0.00,0.00,0.00)" /><!--top wall, 90m long to let people enter -->
<Object model="Coffee Machine" name="coffee1" pos="(49.14,4.38,0.00)" scale="(0.5,1.5,2.0)meters" yprDegree="(0.00,0.00,0.00)" /><!--coffee machines size = (0.5,1.5,2.0)meters -->
<Object model="Coffee Machine" name="coffee2" pos="(49.20,6.53,0.00)" scale="(0.5,1.5,2.0)meters" yprDegree="(0.00,0.00,0.00)" />
<Object model="Coffee Machine" name="coffee3" pos="(49.15,1.79,-0.00)" scale="(0.5,1.5,2.0)meters" yprDegree="(0.00,0.00,0.00)" />
<Object model="WorkingStation1" name="WS1A" pos="(-2.17,35.59,0.00)" scale="(2.0,6.0,2.0)meters" yprDegree="(0.00,0.00,0.00)" /><!--working station (with the darkest color) : size=(2.0,6.0,2.0)meters -->
<Object model="WorkingStation1" name="WS1B" pos="(-1.60,28.28,0.00)" scale="(2.0,6.0,2.0)meters" yprDegree="(0.00,0.00,0.00)" />
<Object model="WorkingStation2" name="WS2A" pos="(6.34,-30.50,0.00)" scale="(2.0,6.0,2.0)meters" yprDegree="(90.00,0.00,0.00)" />
<Object model="WorkingStation2" name="WS2B" pos="(13.81,-30.71,-0.00)" scale="(2.0,6.0,2.0)meters" yprDegree="(90.00,0.00,0.00)" />
<Object model="WorkingStation2" name="WS2C" pos="(21.14,-30.97,-0.00)" scale="(2.0,6.0,2.0)meters" yprDegree="(90.00,0.00,0.00)" />
<Object model="WorkingStation3" name="WS3A" pos="(5.77,-18.24,0.00)" scale="(2.0,6.0,2.0)meters" yprDegree="(90.00,0.00,0.00)" />
<Object model="WorkingStation3" name="WS3B" pos="(13.86,-18.27,0.00)" scale="(2.0,6.0,2.0)meters" yprDegree="(90.00,0.00,0.00)" />
<Object model="WorkingStation4" name="WS4A" pos="(11.46,21.68,0.00)" scale="(2.0,6.0,2.0)meters" yprDegree="(0.00,0.00,0.00)" />
<Object model="WorkingStation4" name="WS4B" pos="(11.47,29.67,0.00)" scale="(2.0,6.0,2.0)meters" yprDegree="(0.00,0.00,0.00)" />
<Object model="WorkingStation4" name="WS4C" pos="(11.23,36.69,0.00)" scale="(2.0,6.0,2.0)meters" yprDegree="(0.00,0.00,0.00)" />
<Object model="WorkingStation5" name="WS5A" pos="(34.86,-18.35,0.00)" scale="(2.0,6.0,2.0)meters" yprDegree="(0.00,0.00,0.00)" />
<Object model="WorkingStation5" name="WS5B" pos="(35.24,-26.82,-0.00)" scale="(2.0,6.0,2.0)meters" yprDegree="(0.00,0.00,0.00)" />
<Object model="WorkingStation5" name="WS5C" pos="(35.41,-34.54,0.00)" scale="(2.0,6.0,2.0)meters" yprDegree="(0.00,0.00,0.00)" />
<Object model="WorkingStation6" name="WS6A" pos="(25.20,21.67,0.00)" scale="(2.0,12.0,2.0)meters" yprDegree="(0.00,0.00,0.00)" /><!--working station, scaled on its y axis to be wider: realize=(2.0,12.0,2.0)meters -->
<Object model="WorkingStation6" name="WS6B" pos="(25.25,37.68,0.00)" scale="(2.0,12.0,2.0)meters" yprDegree="(0.00,0.00,0.00)" />
<Object model="Wall" pos="(23.08,4.77,0.00)" scale="(52.67,1.00,1.00)" yprDegree="(0.00,0.00,0.00)" /><!--middle wall, just to break the flow of people -->
<Object model="WorkingStation7" name="WS7A" pos="(26.40,-18.50,0.00)" scale="(2.0,6.0,2.0)meters" yprDegree="(0.00,0.00,0.00)" />
<Object model="WorkingStation7" name="WS7B" pos="(26.92,-27.54,0.00)" scale="(2.0,6.0,2.0)meters" yprDegree="(0.00,0.00,0.00)" />
<Object model="WorkingStation8" name="WS8A" pos="(39.08,-8.00,0.00)" scale="(2.0,6.0,2.0)meters" yprDegree="(0.00,0.00,0.00)" /><!--working station (with the lightest color) -->
<Object model="WorkingStation8" name="WS8B" pos="(39.06,-16.89,-0.00)" scale="(2.0,6.0,2.0)meters" yprDegree="(0.00,0.00,0.00)" />
<Object model="WorkingStation8" name="WS8C" pos="(39.11,-25.77,-0.00)" scale="(2.0,6.0,2.0)meters" yprDegree="(0.00,0.00,0.00)" />
<Object model="Wall" pos="(11.87,28.99,0.20)" scale="(14.28,1.00,1.00)" yprDegree="(90.00,0.00,0.00)" /><!--some walls to fill the gaps between working stations -->
<Object model="Wall" pos="(11.67,28.33,0.20)" scale="(14.28,1.00,1.00)" yprDegree="(90.00,0.00,0.00)" />
<Object model="Wall" pos="(25.30,30.03,0.20)" scale="(14.28,1.00,1.00)" yprDegree="(90.00,0.00,0.00)" />
<Object model="Wall" pos="(27.05,-26.36,0.20)" scale="(14.28,1.00,1.00)" yprDegree="(90.00,0.00,0.00)" />
<Object model="Wall" pos="(39.33,-16.83,0.20)" scale="(14.28,1.00,1.00)" yprDegree="(90.00,0.00,0.00)" />
<Object model="Wall" pos="(14.56,-30.40,0.20)" scale="(14.28,1.00,1.00)" yprDegree="(180.00,0.00,0.00)" />
<Object model="Wall" pos="(9.79,-18.70,0.20)" scale="(14.28,1.00,1.00)" yprDegree="(180.00,0.00,0.00)" />
<Object model="Wall" pos="(35.33,-26.80,0.20)" scale="(14.28,1.00,1.00)" yprDegree="(90.00,0.00,0.00)" />
<Object model="Wall" pos="(11.31,43.20,0.20)" scale="(14.28,1.00,1.00)" yprDegree="(90.00,0.00,0.00)" />
<Object model="Gate" pos="(45.84,55.29,0.20)" scale="(14.28,1.00,1.00)" yprDegree="(0.00,0.00,0.00)" /><!--workers are created by this object -->
</SmartObjects>
```

Figure 10: Layout configuration file

Note that this kind of information will not be accessible in a real (not simulated) environment. It is the purpose of the Heatmap to provide almost the same information. This means that neither the Heatmap forecaster component nor the RL Optimiser will take this "Objects.xml" as input. It is given just as background information.

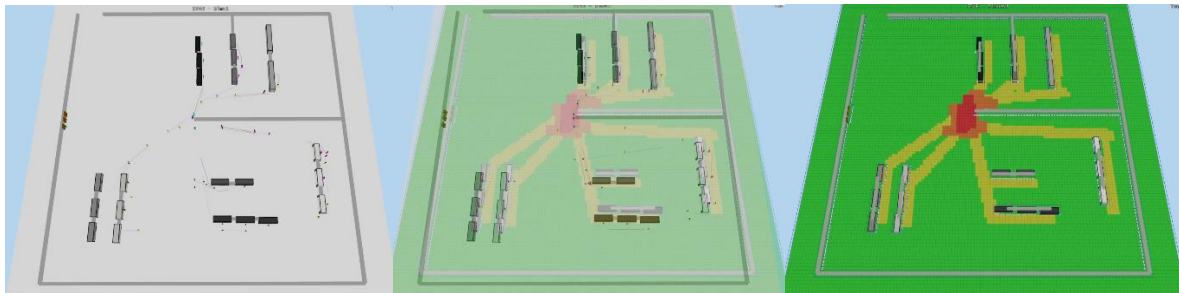
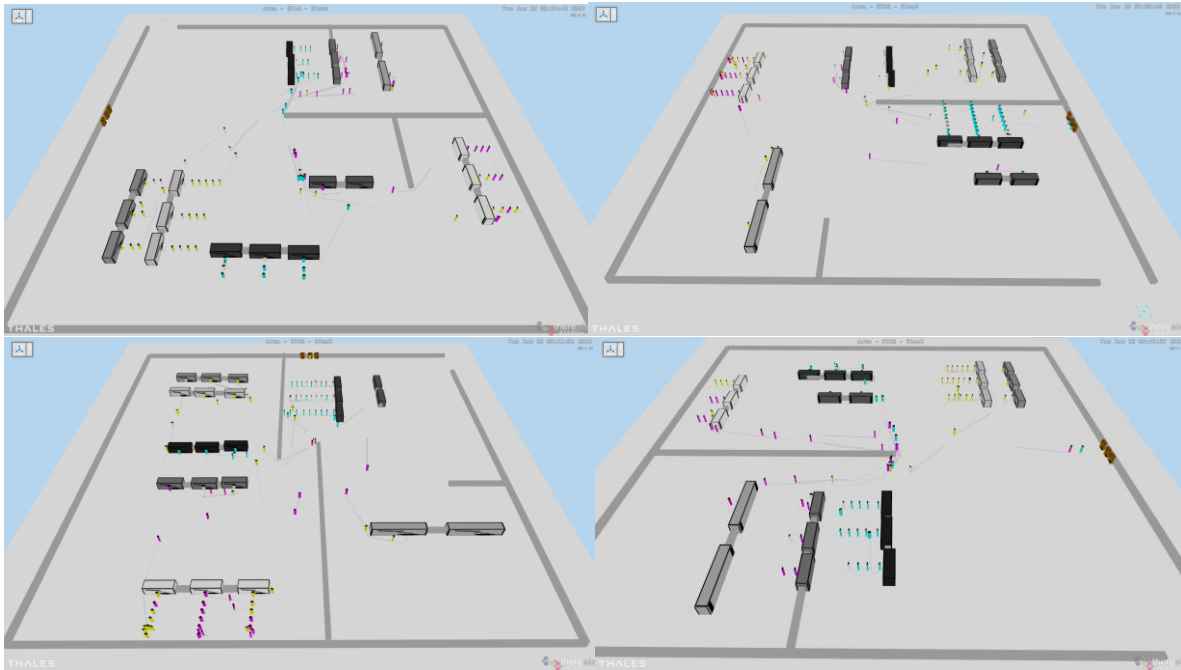


Figure 11: From worker simulation to an average Heatmap

Both Heatmap forecaster component and the RL Optimiser will take as input the current HeatMap and all the previous ones (from t=0 to t=current time). To simplify the AI processing, it is possible to make some average temporal computation of Heatmaps. For instance, Figure 11 show what looks like an average Heatmap, by displaying its cells with different colors (from red to green, passing through yellow), depending on the average crowding level: red is the area where, on average, there is the more chance to find workers.

### 3.3 Generalisation and Realism

Another benefit of the SE-Star simulator is its capability to set a new environment with low effort. In particular, thanks to new development in the STAR project, we can easily change the factory layout, the proportion of workers population, or the worker activities schemes. Thus, any ML AI using the outputs of the simulator for training (as dataset or as RL environment) will easily face many different situations. This helps ensuring the AI will encounter in training a situation similar to the one it will face during its exploitation, when deployed in a real environment.



*Figure 12: Generalisation*

To give some order of magnitude, SE-Star can simulate 2 hours of factory scenario, populated with 50 to 100 workers, in roughly 1 minute. During the run, it records the current Heatmaps each second, in a format similar to what will be produced by the VCA of Task 5.3 (described in D5.6). Depending on the Heatmap resolution used, the size of the generated data of a 2 hours scenario can vary from 250MB (1m Heatmap resolution) to more than 1GB (0.5m resolution).

One may ask if, despite the advance capabilities in human behaviours modelling of SE-Star, the simulated data produced are good enough to train an AI that will still perform good enough later in the real world. This general issue with AI training with simulated data, sometimes known as “reality gap”, is here moderated by the Heatmap approach. Seeing the factory through the Heatmap is like seeing a picture through a very low-resolution camera, with only a few colours. The Heatmap can then be seen as a lens that make difficult to perceive the difference between simulated data and real one. For instance, the position of human workers are not described with coordinates  $(x,y)$ , but instead through the cells of the Heatmap. The Heatmap creates a discretisation of the environment, depending on its resolution, that smooths the simulated data in such a way that it becomes hard to see the difference with real data. Of course, this comes with a decrease in precision. However, the final goal of the AMRs Fleet Optimiser is to avoid, during planning of robots’ movements, the areas where there are chances to find workers. As soon as a worker (defined by a point with a radius) overlaps with a cell, this cell is tagged as containing the worker. Rounding up cell occupancy and having less precision is not an issue in this context. Same apply to objects in the factory (cf. Figure 4). AMRs Optimiser can also use a minimal distance to obstacles (to the cell containing an obstacle to be precise) when performing pathfinding.

### 3.4 Procedural Generation (New)

When dealing with learning, the question of generalisation comes fast. Since we want a solution that can be adapted quickly to a new environment, it is necessary to learn in the training process (in particular for our Reinforcement Learning part) over a diverse range of environments. In our context, a large variety of environments can be directly interpreted as a large number of factory layouts. Configuring a few factory layouts manually is easy. However, scaling on the number of layouts requires another approach.

To do so, we used Wave Function Collapse<sup>1</sup> that generates bitmaps that are locally similar to an input bitmap.

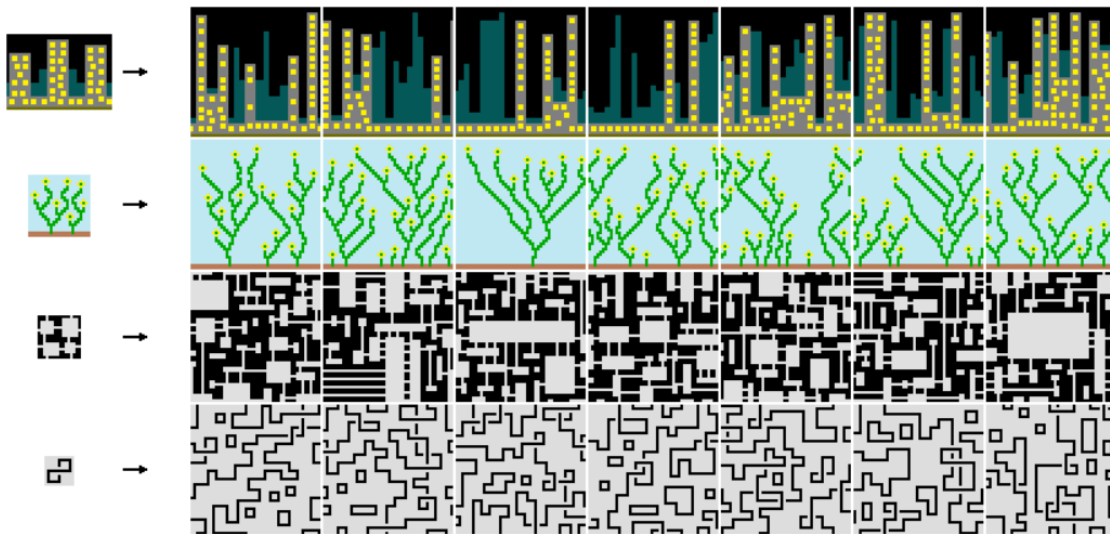


Figure 13: Examples of Wave Function Collapse usages

Using this technology from a bitmap that represents a sample of factory, we are then able to generate automatically random factory layouts.

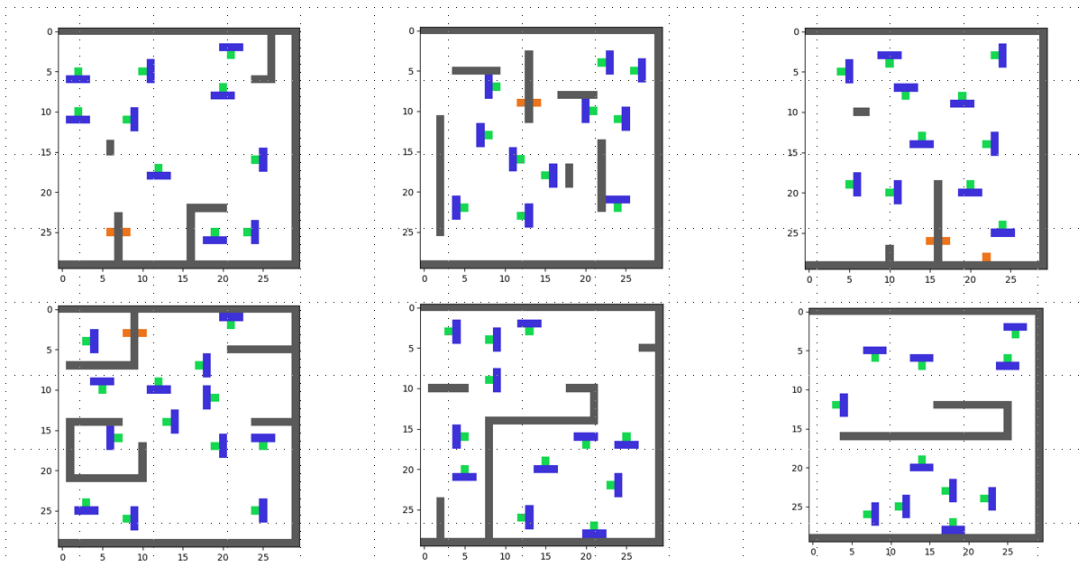


Figure 14: Factory layouts generated with Wave Function Collapse

<sup>1</sup> <https://github.com/mxgmn/WaveFunctionCollapse>



The overall training workflow is the following:

1. Creation of sample pictures for Wave Function Collapse
2. Specification of constraints for Wave Function Collapse
3. Generation of factory layouts as pictures
4. Generation of simulation configuration files from generated pictures
5. Creation of generic configuration files for workers behaviours (independent of factory layout)
6. Runs of simulation to generates HeatMaps
7. Use of recorded HeatMaps for HeatMap Forecaster (see section 4)
8. Use of recorded HeatMaps to compute average HeatMap
9. Transform average HeatMap as "cost map" for path-finding
10. Use of recorded HeatMaps for RL training
  - a. Workers do not avoid robots, since their movements have been recorded without the presence of the robot.
  - b. Forecasted HeatMap can be emulated by using each for each time  $t$  the heatmap of  $t+n$ .
  - c. Do curriculum learning: use different number different runs of simulations with different number of workers to change the complexity for RL agent during training

## 4 HeatMap Forecaster

### 4.1 Human Trajectory Prediction

For AMR socially-aware navigation, it is of interest to learn to predict human occupancy everywhere in the shared workspace, given historical records of occupancy sequences. In this setting, the workspace can be seen as an occupancy grid. The historical records can be records of occupancy in the grid, showing movable (robots, humans) and immovable entities (e.g., walls, obstacles, production machines etc.). The goal is, given a sequence of occupancy grid instances, to project the future sequence of occupancy instances, at least as far as human presence is concerned. This is clearly a spatiotemporal problem i.e., predicting human presence in a grid cell (spatial dimension) at a specific point in time in the future (temporal dimension). Furthermore, a (shared) workspace is a complex setting with a high degree of spatiotemporal relationships. In a workspace, many workers are all simultaneously working on their own (different) tasks and interacting with each other within a limited physical space. A shared workspace also adds cobots into the mix and thereby further increases the complexity of the setting.

While there is substantial research on socially aware robot navigation for outdoor spaces, there is limited work targeting the problem of human movement prediction in industrial workspaces. Models suitable to capture the complex spatial and temporal dependencies in the data are, for example, convolutional neural networks (CNN) or recurrent neural networks (RNN). Such models can be levered to take the spatiotemporal information into account and are frequently used in the literature for human movement predictions 47 [REF-11](Rudenko et al., 2020). Alternatively, human movement can be predicted using a graph-based approach. By representing the problem using graphs, the spatial dependencies can directly be captured by the topology of a graph. Such graph neural networks (GNN) use the concept of message passing, meaning that each node representation is updated with the message received from its connected neighbours [REF-12]. A GNN can be combined with RNNs or CNNs, where the GNN captures the spatial dynamics and the RNNs or CNNs models the temporal dependency. Such spatiotemporal graph neural networks are with success applied to traffic speed forecasting, COVID-19 forecasting and trajectory prediction [REF-13]. However, in the context of shared workspaces, graph-based spatiotemporal models have not been applied for human presence prediction. Given that graph neural networks (GNN) have performed well in other spatiotemporal problems e.g., traffic forecasting [REF-14] and pedestrian trajectory prediction [REF-15], the reported work here explores also their applicability, alongside that of CNNs to predict human presence in the context of shared human – robot workspaces.

A widening range of applications increasingly require methods for predicting the future movement of humans within spaces of interest. Distinguishing between indoor and outdoor spaces, the interest here focuses on indoor spaces, as appropriate for the deployment environment of the human – mobile robot co-existence scenarios in the STAR project. In a typical trajectory forecasting problem, given a time sequence of human movement trajectories, the requirement is to derive estimates of the likely human presence in the next few time instances. Obviously, the uncertainty regarding any such predictions will increase at each time step, so predictions will need to be continuously updated, at each such time step. The problem of human movement prediction is a broader one, that includes topics such as intention recognition, and even involve breaking down the action of human movement to movement primitives, especially if the interest is in action recognition, and not just in movement detection and forecasting. However, the focus here is merely on the trajectory of

the human movement, and not on action or movement primitives. The basic premise of the trajectory prediction in this context is the availability of a sufficiently robust and reliable indoors human detection mechanism. The perception elements that feed into the detection are immaterial for the forecasting: perception could be fed through multiple modalities, including laser, camera-based or positioning mechanisms. In the context of the STAR project, it is based on the video analytics component.

A standard Kalman Filter – based approach from estimation theory can be applied to this problem [REF-16]. The approach can be extended to multiple simultaneous people tracking, as relevant for the STAR project needs, which can employ state-space representation methods, with the state dynamics being represented as a Markov chain [REF-17]. This can extend to growing projected trajectories and employing the Minimum Description Length (MDL) principle for making a selection among the candidate trajectories [REF-18]. Although the ideas have been applied for outdoors tracking, the principle also holds for indoors cases [REF-19].

Considering that human movement follows physics-based rules, methods based on physics of movement can also be applied, which can be extended to handle cases when different human trajectories cross each other, by introducing a repulsion factor to avoid them falling on each other's way [REF-20]. Physics of motion and spatio-temporal variables can be included in the model representation.

However, human movement in working spaces is more likely to involve at least some types of patterns, compared to random human movement, and therefore methods have been applied which are relevant to shorter- or longer-term patterns, employing for example Long-Short Term Memory Networks to capture such patterns and predict future trajectories [REF-21]. When considering the Heatmap-based approach adopted in STAR, the importance of spatial and temporal proximity in determining the future state of heatmap grid cells regarding the human presence, implies that models which explicitly seek to exploit such proximity, such as Convolutional Neural Networks, are appropriate candidate methods and for this reason have been applied also for human movement prediction [REF-22].

Furthermore, the context of the movement patterns, as reflected on trajectories planning, could be applicable. However, such methods pre-suppose some planning and destination assumptions and although likely to offer accuracy advantages, can also be more restrictive in their applicability [REF-22]. Instead, the movement patterns might be learnable through historical records of movements and occupancy, implying that a model might be able to build data-driven internal representations to account for the movement patterns.

Considering the range of options for the forecaster, a shortlist of candidate approaches follows.

#### 4.1.1 Physics-based methods

This can consider a set of explicitly defined dynamical models of human motion. These can be partly data-driven, observing the dynamics of human motion as observed in the available data. These can be fed into a recursive Bayesian filter to produce projected outcomes for each cell in the grid for a given time window in the future. The basic form is:

$$s'_t = f(s_t, u_t, t) + w_t$$

where  $u_t$  is the (unknown) control input and  $w_t$  is the process noise. Two variants of this model are single-model and multi-model methods, as different humans are likely to follow different dynamic patterns of movement:

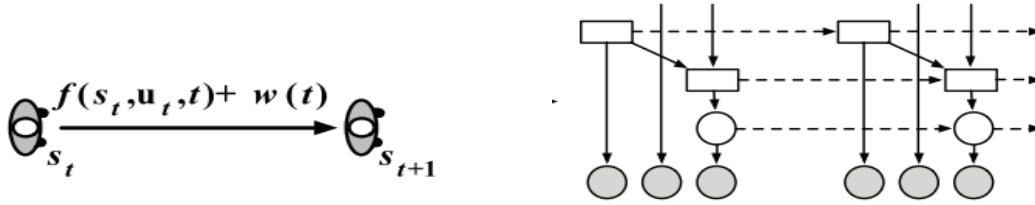


Figure 17: single and multi-model physics-based prediction

However, in the present case, posing the problem in such terms (linked to a control systems’ viewpoint) would present difficulties, especially considering the high – level of autonomy and unpredictability of all involved entities.

### 4.1.2 Planning-based methods

These are not likely to be further explored, as they imply assumptions, which although can improve predictive accuracy, they will also restrict the applicability of the approach, as any prediction will have to take into account the robot planning, which would necessitate a very tight integration of the planning for each individual robot and the heatmap forecasting, which goes against the need for individual components of the solution to be highly modular.

### 4.1.3 Pattern-based methods

These can follow sequential and non-sequential patterns. Due to the obvious relevance of time and spatial proximity, the sequential patterns are preferred, where the state at the next time instance  $s_t$  in each cell of the grid is a function  $f$  of the state of this and neighbouring cell’s states in the time horizon preceding the current time instance,  $s_{t-n:t}$  :

$$s_t + I = f (s_{t-n:t})$$

The problem can be specifically posed in a way such that the specific individual’s movement is not explicitly considered but instead the workspace can be seen as an occupancy grid where human presence is anonymised. The interest is then to investigate the extent to which future human movement prediction could be achieved on the basis of a historical record of sequences of human presence in workspaces. If such data-driven learning is feasible, the predictive models might as well have developed internal representations that capture also the movement patterns and therefore implicitly only the intentions of the human movement [REF-24].

## 4.2 Convolutional Neural Network – Based Approaches

### 4.2.1 How the problem is posed

Given a space that is partitioned into a grid of cells  $P = \{P_i, i = 1, \dots, N\}$ , with  $N$  being the total number of cells in the grid, for each cell in the grid, and for each second  $t$ , a value  $C^t(P_i) = 1 (0)$ , if the cell is (not) occupied.

The problem is to determine if, given a known sequence of such past values in time it is possible to estimate future values. The proposed model is based on the CNN architecture, considering that CNNs can be applicable to spatio-temporal data. To explore this problem, a simplification assumption is made to consider a time window of three seconds in the time

series of the occupancy grid. This assumption is based on the intuitive idea that workers have to go through the neighbouring cells to move from one cell to another one. The time window could also be set via estimating the average speed of a worker, during movements, but this can be seen as a further enhancement for the future. In a grid with  $N$  cells, the proposed strategy is based on the representation of each cell as a vector in  $R_n$  where  $n$  is the number of features obtained from the information of the cell. Details are provided in [REF-23].

#### 4.2.2 CNN structure

To explore the predictive performance which can be obtained by training CNN models with such data, experiments involving different values of the hyperparameters of CNNs have been performed. The CNN structure has three convolutional layers with 128, 64 and 32 neurons, with corresponding kernel sizes of 5, 2, and 1 each, and Rectified Linear Units (ReLU) as activation functions. A MaxPooling layer is applied after each convolution. The selected dropout rate to compensate for potential overfit was set at 0.2. The training has been performed for 100 epochs with a learning rate of 0.0001 (adaptive nonetheless, involving ADAM training).

#### 4.2.3 Performance Assessment

The performance of the methods is evaluated by the Root Mean Square Error (RMSE) for the regression outcomes of the model. When converting the model output to occupancy, Accuracy, Recall and Precision, where involved, as appropriate for classification.

#### 4.2.4 Results

In this section the results obtained from the application of the CNN approach are presented and discussed. The CNN model directly produced outputs are real numbers between 0 and 1, which can loosely be seen as belief or likelihood that a grid cell will be occupied, with higher values indicating higher such belief or likelihood. If this needs to be interpreted as a binary outcome (occupied or not), then network output can be passed through a threshold unit to decide whether the output of the model has to be interpreted as occupied or unoccupied cell. Specifically, for values higher than this threshold, the model will classify the cell as occupied. Different performance metrics, such as the values of root mean squared (RMSE), accuracy, recall and precision offer a different insight into the predictive performance. To analyse this influence, the CNN training has been performed with different thresholds. For visual interpretability of the results, heatmaps representing the occupancy likelihood, as predicted by the model, have been created. One example is shown in Figure 18. Intuitively, for low thresholds more predictions are interpreted by the model as presence than with high thresholds. Therefore, low thresholds lead to higher True Positives (TPs), while high thresholds lead to higher False Negatives (FN)s. Therefore, recall drops down significantly for thresholds 0.4, 0.5, 0.6 and 0.7 but increases for 0.2. However, lower thresholds of 0.2 and 0.3 also lead to a high RMSE and increased False Positives (FPs). For higher thresholds such as 0.6 and 0.7, the opposite happens. For these values, the probabilities of neighbouring cells are too low to be interpreted as presence and TP decrease over time. For instance, predictions obtained with threshold 0.6 one second ahead (Figure 18) are very similar to the ones obtained with threshold 0.3. Nevertheless, the number of cells predicted as occupied after 15 seconds with threshold 0.6 is significantly lower than the number of cells predicted as occupied after 15 seconds with threshold 0.3. This is because the probabilities of being occupied are too low to be interpreted as presence and the number of cells predicted as occupied decreases over time, leading to error propagation. Predictions beyond 15 seconds for the 0.6 threshold are not presented as not perceptible differences to the values obtained with 15 seconds have

been found. It can be deduced that setting the threshold too high results in a lower rate of correct predictions of occupied cells (TP) for long time intervals.

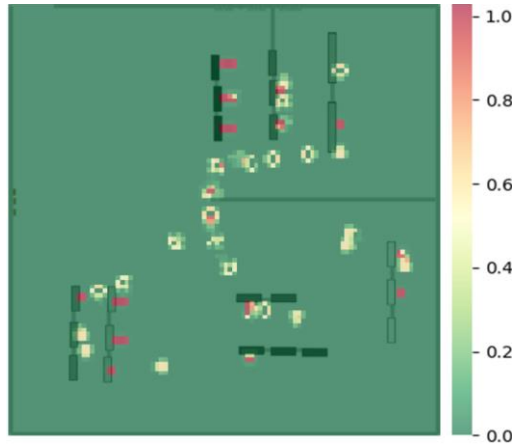


Figure 18: Prediction Heatmap for the next second 50 workers; threshold set to 0.6

A summary of the results against a time horizon of 60 seconds with different thresholds is presented in Table 1. The recall values in Table 2 are higher for the experiments with 100 persons than with 50 persons because the number of TPs is greater for experiments with 100 workers. However, the number of FP predicted for experiments with 100 workers is also high, which results in a decrease in overall accuracy and precision. In more detail, Table 3 shows the results with time horizons 1, 15 and 30 in workspace one with different thresholds for the cases of 50 and 100 workers respectively. It can be observed that accuracy of 0.99 and RMSE of 0.08 have been obtained with one step ahead predictions, but both values get decrease when projecting further ahead into the future, due to prediction error accumulation. When the occupancy detection sensitivity is increased, by lowering the classification threshold, the higher is the number of cells that are predicted as occupied. Safety of workers should be a high priority in Human-Robot Co-existence Spaces (HRCS) and this justifies tolerance for a higher rate of false positives, to err on the side of safety. However, this also results in more cautious and therefore potentially sub-optimal path planning for the robot fleet. False Negatives (FN) measures the number of cells that are occupied, but they are predicted as not occupied. If FN increases, it could lead to less secure workspaces than increasing FP. Thus, a low precision can be interpreted as a lower risk than a low recall.

Table 1: CNN-based heatmap overall predictive performance for different thresholds

Number of workers	Threshold	Average RMSE	Average accuracy	Average recall	Average precision
50	0.2	0.566	0.599	0.896	0.077
	0.3	0.333	0.867	0.650	0.125
	0.4	0.239	0.926	0.468	0.170
	0.5	0.184	0.957	0.338	0.221
	0.6	0.124	0.984	0.261	0.742
	0.7	0.123	0.984	0.246	0.744
100	0.2	0.576	0.583	0.919	0.130
	0.3	0.367	0.839	0.774	0.206
	0.4	0.245	0.924	0.654	0.314
	0.5	0.198	0.952	0.525	0.418
	0.6	0.147	0.978	0.476	0.855
	0.7	0.153	0.976	0.414	0.856

*Table 2: CNN-based heatmap predictive performance for different time horizons (50 workers)*

Second	RMSE	Accuracy	Recall	Precision
1	0.080	0.989	0.856	0.648
15	0.385	0.809	0.830	0.077
30	0.613	0.584	0.957	0.041
60	0.842	0.267	1.0	0.026

*Table 3: CNN-based heatmap predictive performance for different time horizons (100 workers)*

Second	RMSE	Accuracy	Recall	Precision
1	0.123	0.974	0.803	0.747
15	0.385	0.809	0.837	0.147
30	0.614	0.581	0.919	0.073
60	0.865	0.227	0.997	0.054

Tables 2 and 3 show the predictive performance for 1, 15, 30, and 60 seconds overall heatmap predictions. The conclusions drawn from this is that relatively reliable prediction is obtained for a few seconds ahead. However, already at 10 seconds and more, the predictions start to become less actionable, and for longer time horizons not particularly useful anymore.

## 4.3 Graph Neural Network– Based Approaches

### 4.3.1 How the problem is posed for a graph representation

The simulation data has a grid structure where each grid cell contains information about the number of humans present and the presence and the type of obstacle(s) in a grid cell. This grid structure must be converted to a graph to be used in a graph-based neural network. This works adopts a node classification approach similar to a traffic speed forecasting problem analysed in [REF-25]. Here each node represents a specific grid cell and the aim is to predict the attribute(s) of that specific node e.g., in this work the human presence in that cell. Using the node classification approach, the graph structure of workspace  $G$  can be formalised as follows:

$$G(V, E, X_{v(t)})$$

where  $V$  is the set of nodes,  $E$  is the set of edges and  $X_{v(t)}$  is the set of node features at time  $t$ . More specifically, each node in  $V$  corresponds to a grid cell in the data. For example, cell (0,0) is node 1, cell (0, 5) is node 6 and cell (1,0) is node 101. The dataset used has  $N = |V| = 10,000$  total nodes. Furthermore,  $X_{v(t)}$  is a set that contains a vector of node features for each node at time  $t$ . Each vector of node features contains information about the human occupancy and additional contextual information (if a wall, coffee machine or workbench is present). Furthermore, the adjacency matrix is a  $N \times N$  matrix that stores the connectivity information of all nodes:

$$A_{i,j} = \begin{cases} 1, & (i,j) \in E \\ 0, & \text{otherwise} \end{cases}$$

To construct the graph from the grid, it is determined that each node is connected to itself and all its first-degree neighbouring nodes including the diagonals. As the data has a time interval of 1 second, connecting only the first-degree neighbours would be sufficient as it is unlikely that workers would travel a larger distance in 1 second. Furthermore, since workers can move in both directions between nodes the graph is bidirectional. Moreover, each edge has two weights since the actual direction of movement is important. These weights influence the effect of the node features in the graph convolution. Varying edge weights, the node features of different nodes are not equally contributing. The higher the edge weight, the greater the influence of these node features in the graph convolution. The edge weights are introduced as learnable parameters in the model. After each step in training, the edge weights will be updated. Hereby, the model would be able to capture underlying spatial dynamics in the workspace. By learning the edge weight through overall network optimization, the trained model implicitly accounts for these different patterns. The weights are initialized by setting them equal to  $1/8$ , where the value 8 corresponds to the average number of edges per node.

To create a sparser graph, the nodes (and their connecting edges) that do not see any human presence during the entire simulation are omitted from the graph. This results in a graph with significantly fewer nodes and edges. Since these nodes were never visited during the entire simulation, it is reasonable to assume that these are unlikely to be visited in the future. In the studied scenarios, the sparser graph did not negatively impact the predictive capabilities of the STGNN, however, it reduced the computational complexity significantly. The entire process of converting the grid to the graph is visualised in Figure 19, where the dotted lines represent nodes and edges that are omitted.

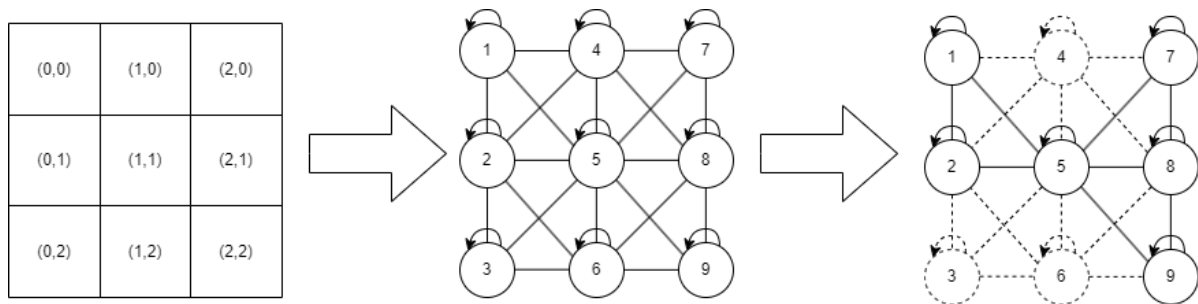


Figure 19: Visual representation of an example grid-to-graph conversion of a 3x3 grid.

The squares are the grid cells, the circles are nodes and its connection are the bidirectional edges. Lastly, the dotted edges and nodes show how non-populated nodes are omitted from the graph.

Following the outlined procedure, 6147 nodes are omitted from the graph. This results in a total of 3846 nodes in the final graph with 31,510 edges, meaning that each node has on average 8.17 edges. Without omitting these nodes, the graph would have 10000 nodes with 88,804 edges. Thanks to the omission of unvisited cells, the graph has become significantly sparser and has a reduced computational complexity.

The constructed graph is a static graph with a temporal signal, meaning that the graph structure remains the same with only the node features changing over time. For training, the time series of graphs are be sliced into smaller sequences. The model uses a historical time series of length  $q$  to predict a time series of length  $p$  i.e., the model predicts the next  $p$  graph representations by using the  $q$  previous observations. To evaluate the performance of the

model, the predictions are compared with the targets i.e., the graph representations from  $t+1$  up to and including  $t+p$ . The sequences of input data are created so the data can be shuffled and split into a test/train set to limit the effects of overfitting. These sequences of node features, combined with the connectivity information serve as the input data for the graph-based model. The procedure is illustrated in figure 20, where  $t$  denotes the time,  $q$  is the length of the historical input time series and  $p$  is the length of the predictor.

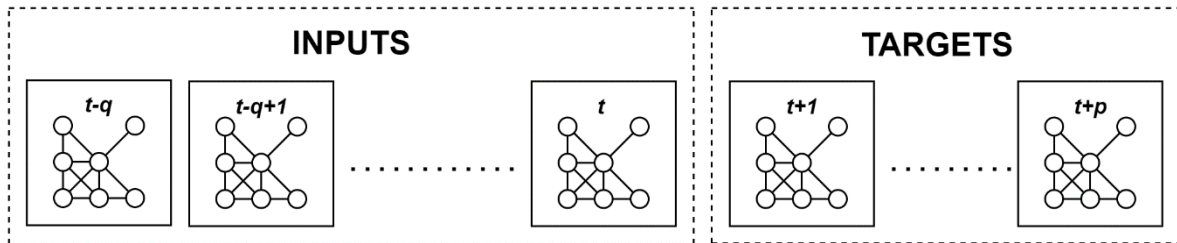


Figure 20: Visualisation of how sequences of graph inputs are linked with actual outputs

### 4.3.2 The Graph network

A spatiotemporal graph neural network (STGNN) is designed to predict the human presence in the workspace. Most of the human movement prediction literature models the agents as nodes and captures the interactions between them. However, in this case the workers are indistinguishable and hard to follow over time, meaning that such an approach would not be suitable. Therefore, a different spatiotemporal approach is used. Specifically, the graph-based model is based on the work of Bai et al [REF-25]. The model by [REF-25] is designed for a traffic forecasting problem which shares a great similarity with the problem posed in this work. Both problems are spatiotemporal problems using node prediction on a static graph with temporal signals. Instead of predicting the traffic speed, the human occupancy at the nodes at a specific time is predicted, given the previous node attributes and topology of the graph. The graph-based model developed by [REF-25] has good long- and short-term prediction capability in the traffic forecasting problem. Therefore, this STGNN architecture can also be levered for the human movement prediction task in this work.

The model captures the spatial dependency by a 2-layered Graph Convolutional Networks GCN on the graph [REF-24]. The 2-layered GCN takes both the first-order and second-order adjacent node attributes into account for the spatial characteristics of the graph. Meaning that information from second degree neighbours, which are not directly connected, are also considered by each node. Contrary to [REF-25], the edge weights in our model are learnable parameters. [REF-25] uses a weighted graph, where all nodes have a specified predetermined weight. These weights are manually set and fixed. These learned edge weights will potentially capture (some of) the underlying spatial dynamics within the data. These edge weights are used as inputs into the GCN but before being used, a Rectified Linear Unit (ReLU) operation is performed on the edge weights. This ensures the stability of the model by making the edge weights non-negative.

Following the GCNs layers, the temporal dependencies are captured by Gated Recurrent Units (GRUs) which learn the short-term trends between the spatial characteristics of the graph. The GRUs determine the human presence at a given time by using the hidden states from the previous moment and the information from the GCN at the current moment as input. It retains the temporal information because of the gated mechanisms. The reset gate controls the degree of irrelevant information to be omitted for the forecast. Whereas the update gate

controls the quantity of information from the previous movement that should be considered for the current state.

Putting both the GCNs and GRUs together yields the Temporal-Graph Convolutional Network T-GCN model as developed by [REF-26]. Each graph representation goes through the GCN layers and is then used as input by the GRU. Besides the input from the GCN, the GRU also considers some of the previous hidden state's information. [REF-25] builds upon this model by feeding the hidden states through an attention model which will learn the importance of the occupancy information at every moment as well as the variation trends of the occupancy states. Finally, a fully connected layer yields the forecasted occupancy data. The final result is a single shot prediction, that predicts the next  $p$  timesteps at once. For all results human occupancy is predicted for up to 40 seconds into the future. Hereby, the interest is to evaluate both how the model predicts human occupancy in the short- and long-term. The model's architecture is visualised in Figure 21.

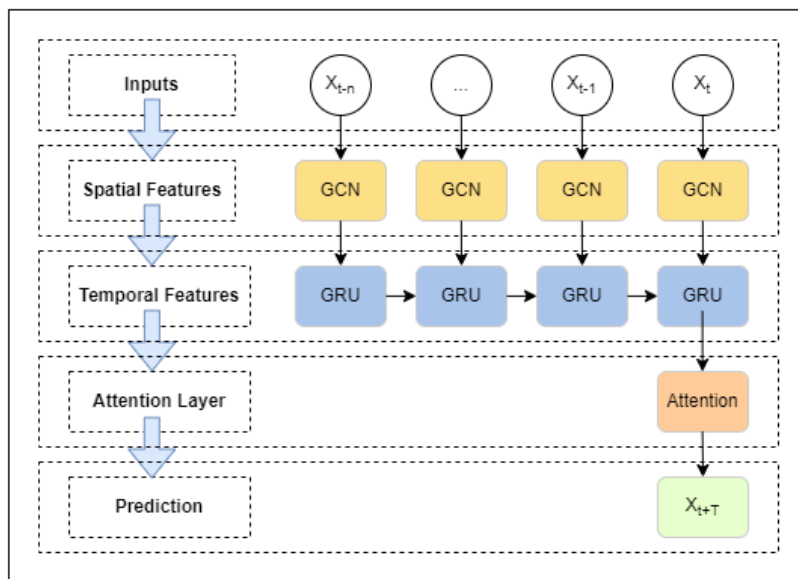


Figure 21: Model's architecture.

### 4.3.3 Model training and performance metrics

The model was implemented using the python library *PyTorch Geometric Temporal* which is created by [REF-27] to handle spatiotemporal graph neural networks. To update the parameters of the network and measure the performance, a mean squared error (MSE) loss function is used:

$$\text{MSE} = \frac{1}{N} \frac{1}{p} \sum_{t=1}^p \sum_{i=1}^N (\hat{y}_i^t - y_i^t)^2$$

where  $N$  is the total number of nodes,  $p$  is number of seconds predicted,  $\hat{y}_i$  denotes the predicted value of node  $i$  and  $y_i$  denoted the actual value of node  $i$ . During the model's training, L2 regularisation is added to limit the effects of overfitting.

The network yields a regression output that using a threshold can be converted into a binary classification. The network’s performance is evaluated using regression and classification metrics. The classification predictions are be evaluated using accuracy, precision, recall, and F-score. Furthermore, as the data is heavily unbalanced due to the vast majority of the nodes being unoccupied, balanced accuracy is also included as an important performance metric. Balanced accuracy provides a better view of the performance of the model as it accounts for the imbalance in the data. Furthermore, the F2-score is be used to evaluate performance. The general F-score formula is shown in formula 7, where the F2-score thus has a  $\beta = 2$ . Whereas the F1-score equally weights precision and recall, any  $\beta > 1$  provides more emphasis on recall than precision.

$$\text{Balanced Accuracy} = \frac{\text{TPR} + \text{TNR}}{2}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

$$F_{\beta} = (1 + \beta^2) \frac{\text{precision} * \text{recall}}{\beta^2 * \text{precision} + \text{recall}}$$

where TP denotes the true positives, TN the true negatives, FP the false positives, FN the false negatives and  $\beta$  is the configuration parameter of the general F-score metric. Lastly, it must be noted that when calculating these performance metrics, the 6147 omitted nodes are not be included. Including the omitted nodes with a value of 0 would inflate the reported assessment metrics, but not accurately reflect the ability of the model to predict human occupancy. Only when plotting the occupancy grids these omitted nodes are added back.

#### 4.3.4 Results

This section presents the results of applying the graph-based model for the human movement prediction. In particular, the section first discusses the effect of adding learnable weights to the model. Next, the performance and behaviour of the model over time is assessed. Finally, the performance of the model with different contextual node information is evaluated. The parameters used to train the model are empirically determined using the validation loss for different configurations. To determine the optimal learning rate and L2 regularisation parameter, a range of values between 0.001 and 0.3 have been tested. The best empirical results were found using a learning rate=0.01 and a L2 regularisation=0.05. Moreover, the model is trained with a batch size=32 for 5 epochs. Furthermore, it is empirically determined that the model is best configured for this specific problem setting using 256 hidden units and an input time series of 5 seconds.

#### 4.3.5 Learnable Edge Weights

The model developed by [REF-25] does not use edge weights as a learnable parameter, therefore as a next we evaluated whether adding the learnable edge weights improves the performance of the model compared to using an unweighted graph. After training, the average edge weight is equal to 0.138. This is close to the edge weight at initialisation, however, that

does not imply that no learning occurred. The edge weight has a standard deviation of 0.142 and the maximum weight is equal to 1.182. The distribution of the edge weights is shown in Figure 22, exhibiting significant variability as a result of the learning process. Due to the ReLU operation on the edge weights, all weights are non-negative. Moreover, it can be observed that the majority of the edge weights have a value close or equal to zero. However, a significant number of the edge weights are larger weights than its value at initialisation. Overall, this indicates that during the training process, the model learns which edges are important and which edges are unimportant for the human movement prediction.

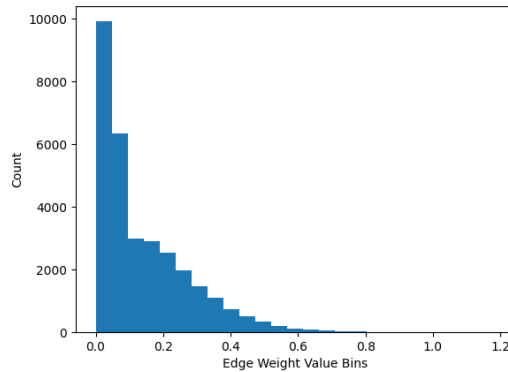


Figure 22: Histogram of learned edge weights showing the distribution of the learned weights.

The performance metrics presented in this section are the average over the 40 predicted time periods. The model with learnable weights has an MSE of 0.4095 whereas the model without learnable weights has an MSE of 0.4501. This is a noticeable difference in performance based on MSE. Furthermore, Table 4 shows the classification performance metrics for different thresholds. The table shows that the model with learnable weights yields a higher balanced accuracy, recall, precision, and F2-score for almost all thresholds. The better performance metrics across all thresholds, indicate that learnable weights could significantly improve the predictive capabilities of the model.

Table 4: Classification results comparing learnable weights with an unweighted graph

Threshold	Balanced Accuracy	Recall	Precision	F2-score
<b>Learnable Weights</b>				
0.00	0.707	0.982	0.156	0.474
0.05	0.832	0.903	0.290	0.630
0.10	0.858	0.805	0.484	0.711
0.15	0.855	0.762	0.603	0.723
0.25	0.842	0.708	0.753	0.716
0.60	0.805	0.616	0.913	0.658

### Unweighted Graph

0.00	0.618	0.982	0.120	0.404
0.05	0.807	0.885	0.254	0.591
0.10	0.842	0.780	0.456	0.683
0.15	0.839	0.729	0.603	0.700
0.25	0.825	0.673	0.761	0.689
0.60	0.791	0.588	0.912	0.633

### 4.3.6 Model Performance Over Time

The previous sections showed the performance averaged over all 40 predicted time points. In contrast, this section will evaluate how the model performance is changing over time. Figure 23 plots the balanced accuracy over time and shows that it decreases with time. The performance in the first few seconds is good, however, the further the predictions are from the input data, the worse the performance becomes. Moreover, the decrease in balanced accuracy begins to stabilise after around 10 seconds. The model appears to be producing steady state prediction for all time periods after 10 seconds.

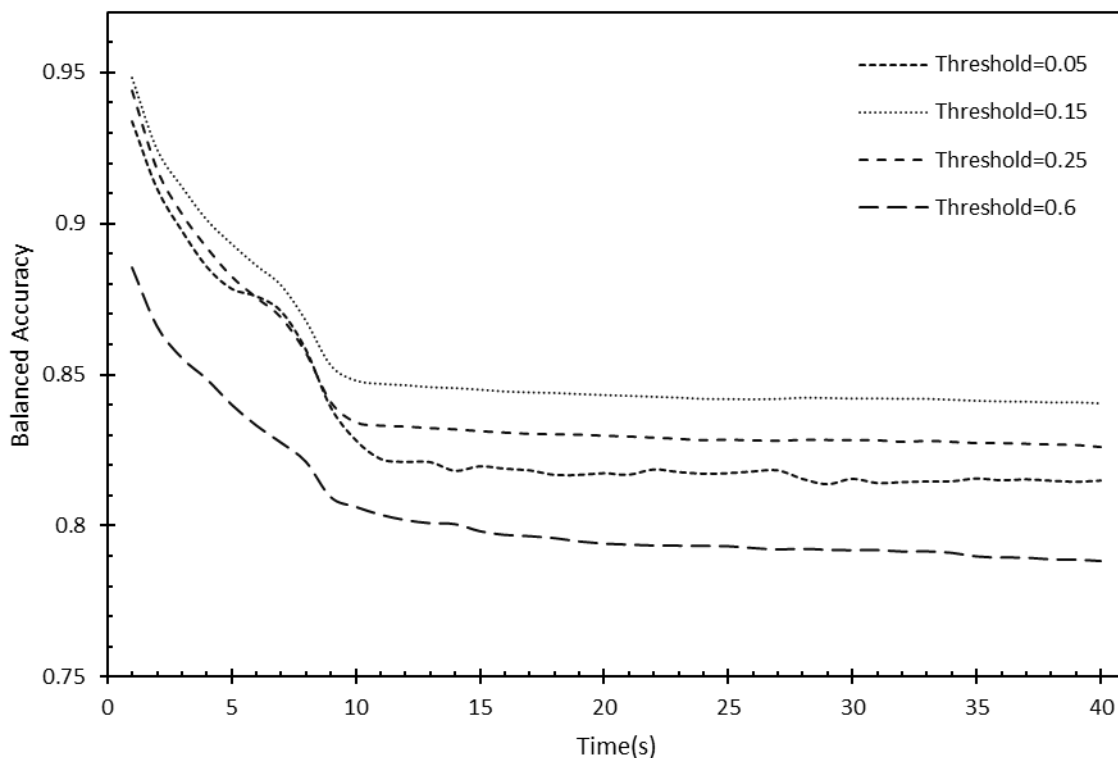


Figure 23: The balanced accuracy over time for different thresholds.

A similar conclusion can be drawn from figure 24 and figure 25 which visually show the predictions over time. Figure 24 shows the regression outputs as a heatmap over time. The regression outputs represent the belief by the model about occupancy at that cell. Moreover,

the results are converted to binary values using a threshold which can be seen in Table 1. Both figure 24 and figure 25 show that the predictions in  $t=10$ ,  $t=20$ , and  $t=40$  are visually very similar, indicating some steady state is reached.

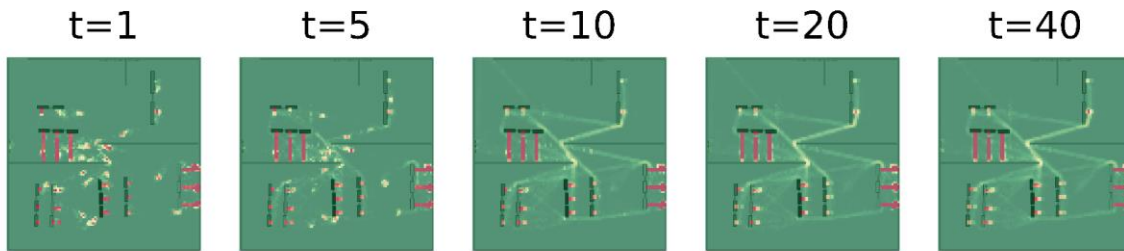


Figure 24: Predicted heatmaps for different time periods

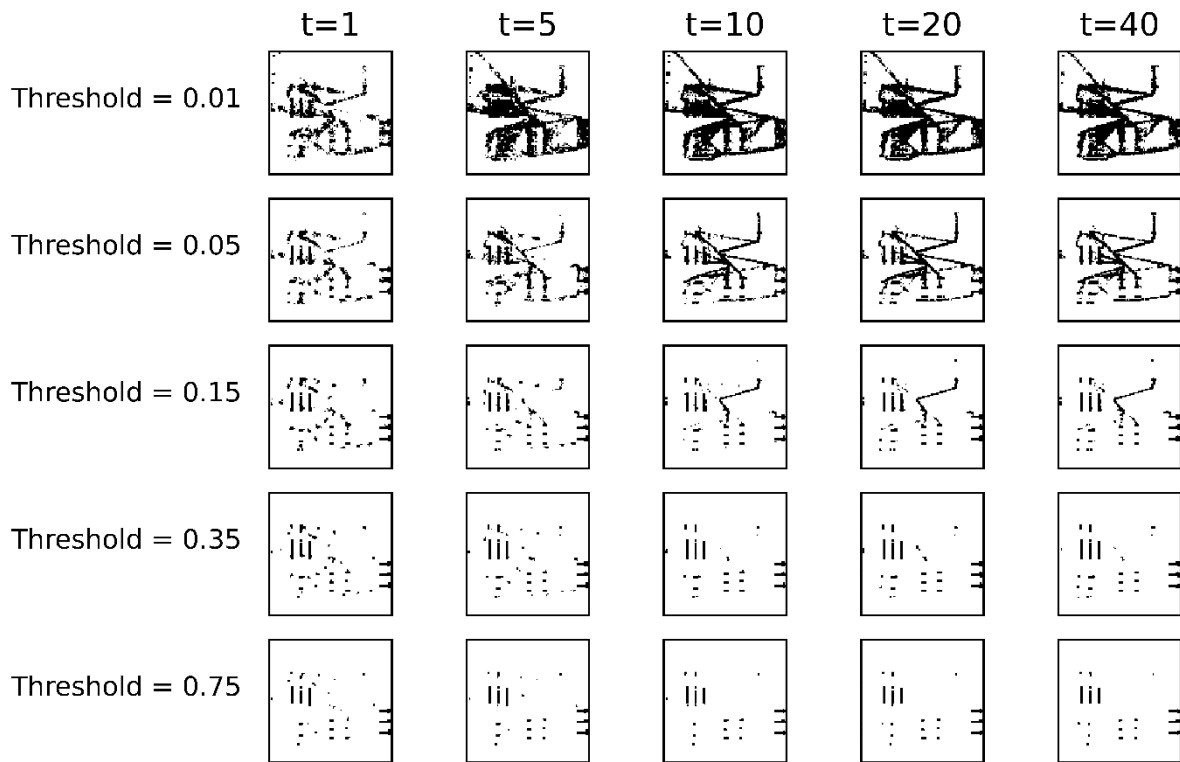


Figure 25: Predicted occupancy grids for different thresholds and time periods.

### 4.3.7 Contextual Node Features

Each node in the graph can incorporate additional (contextual) information that can be used to predict the future occupancy. In this section, it is evaluated if this additional contextual information increases performance and if so, which information is the most useful.

Table 5 reports the MSE for a full factorial experiment with the node features. Compared to the results without any additional context information, all additional node features see a small increase in performance. Furthermore, the combination of several contextual node features further increases the performance of the model. When including all node features the best performing model is obtained. It must be noted that the improvement in performance is modest.

*Table 5: Regression results with different contextual information.*

<b>Contextual Node Features</b>	<b>Mean Squared Error</b>
Workbench + Coffee + Wall	0.4095
Workbench + Wall	0.4132
Workbench + Coffee	0.4128
Wall + Coffee	0.4145
Workbench	0.4164
Coffee	0.4177
Wall	0.4149
None	0.4219

Table 6 shows the classification performance metrics for different thresholds, averaged over the first 10 time periods. From table 6, it can be concluded that the additional contextual information does slightly increase the balanced accuracy of the mode across all thresholds. Furthermore, the same conclusion can be drawn based on precision and recall, which also present a small improvement.

*Table 6: Classification results with different contextual information*

<b>Thresholds</b>	<b>0.00</b>	<b>0.05</b>	<b>0.10</b>	<b>0.15</b>	<b>0.25</b>	<b>0.6</b>
<b>Workbench + Coffee + Wall</b>						
Balanced accuracy	0.774	0.878	0.898	0.896	0.887	0.843
Precision	0.199	0.383	0.563	0.665	0.779	0.924
Recall	0.979	0.913	0.867	0.836	0.797	0.692
<b>Workbench + Wall</b>						
Balanced accuracy	0.709	0.869	0.891	0.890	0.880	0.837
Precision	0.155	0.347	0.531	0.647	0.776	0.927
Recall	0.987	0.923	0.862	0.827	0.783	0.679
<b>Wall</b>						
Balanced accuracy	0.715	0.858	0.882	0.881	0.869	0.827
Precision	0.160	0.325	0.501	0.620	0.766	0.926
Recall	0.986	0.920	0.853	0.814	0.763	0.660

None

---

Balanced accuracy	0.700	0.857	0.882	0.882	0.870	0.827
Precision	0.151	0.323	0.495	0.615	0.765	0.928
Recall	0.983	0.918	0.855	0.817	0.768	0.659

---

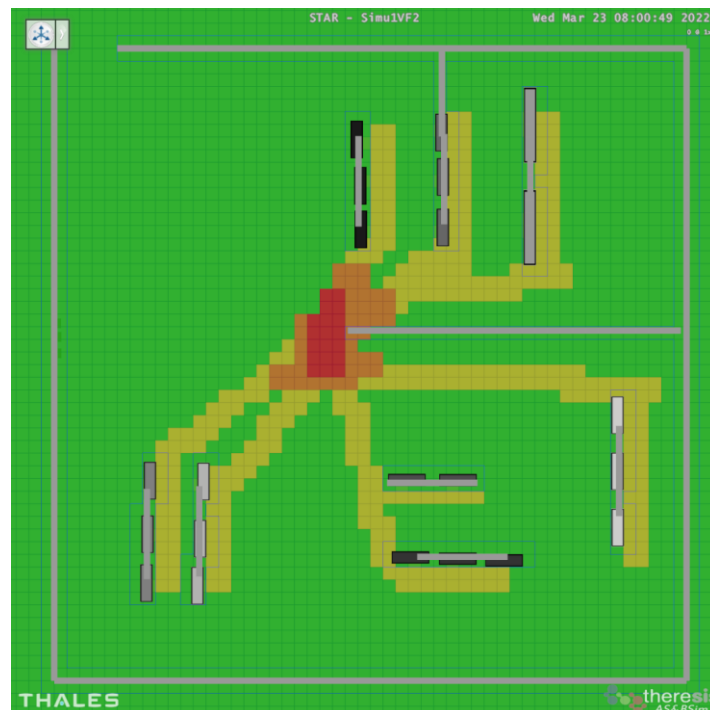
## 5 AMR fleet controller

### 5.1 Objectives of the fleet

The goal of the AMR Fleet is to perform logistic tasks. Each task consists in carrying some goods from one working station to another, with possibly a priority level that helps sorting all tasks. As explained previously, the commands sent to each AMR should take into account the human presence. To some extent, it relies on well-known technology such as pathfinding.

### 5.2 Pathfinding

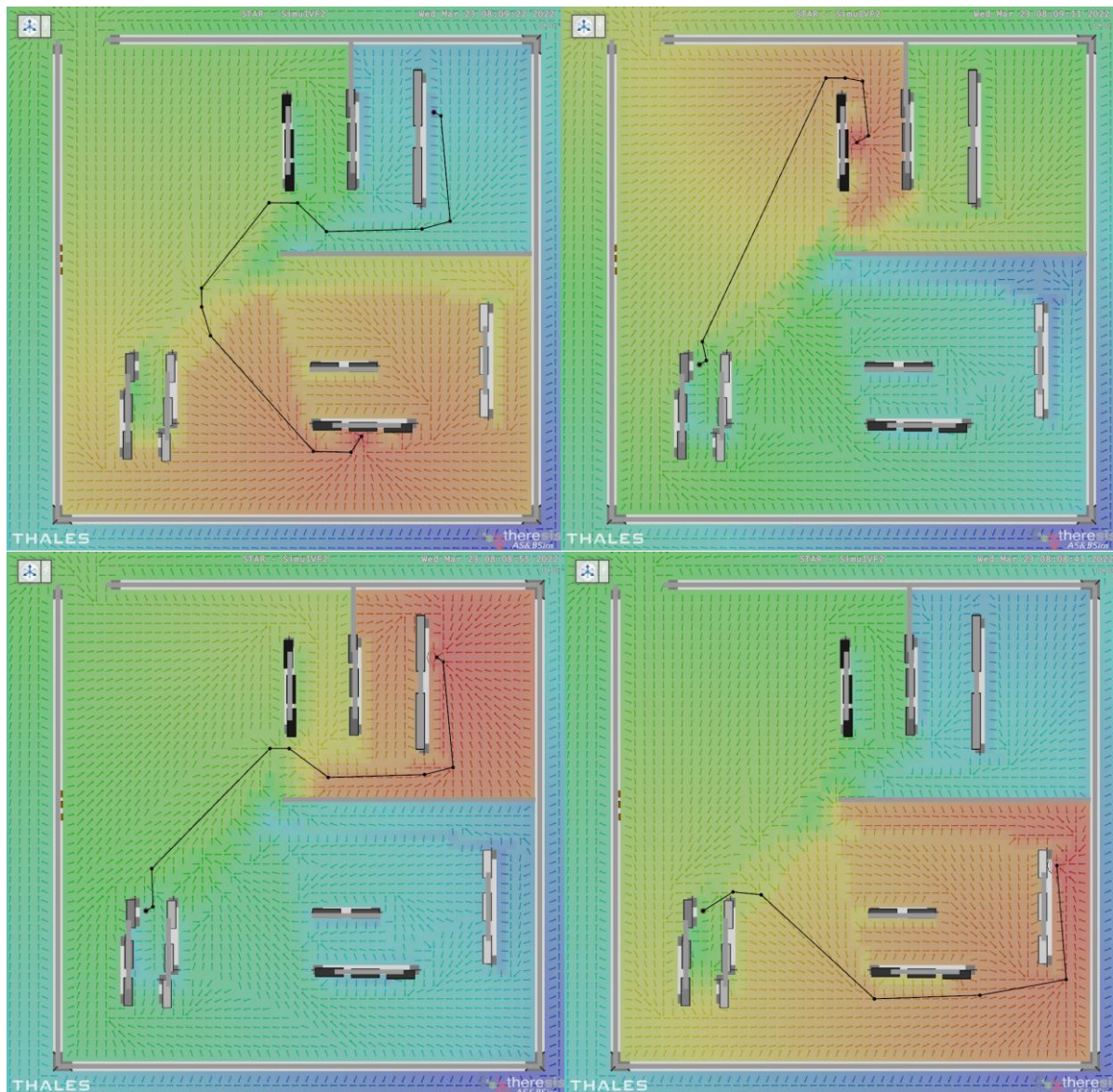
In particular, approaches such as the Field D\* [REF-10], that are in the node-based exploration category and take a grid as nodes graph, map very well to the Heatmap. Cost on the graph can be easily deduced from cell occupancy of the Heatmap. If we take the temporal average Heatmap as input, we can set different path costs (that tells how much the cell should be avoided) depending on how many people there are in this temporal average. Figure 26 illustrates such temporal average and gives an overview of which are the areas that need to be avoided. In this example, four thresholds are used to discretise the cell average occupancy, displayed in four different colours.



*Figure 26: Area to avoid (the hotter the colour, the more it needs to be avoided)*

Note again that the RL Optimiser will take more than the temporal average Heatmap as input, but instead all the instantaneous ones (or at least a subset) from  $t=0$  to  $t=\text{current time}$ , plus some forecasted (see section 4) ones ( $t=\text{currentTime}+1s$  to  $t=\text{currentTime}+Ns$ ).

Thanks to STAR developments, using Field D\* to such temporal average Heatmap leads to distance vector fields that lead to the destination. Integrating such distance fields then produces paths for AMRs that avoid crowded areas, as displayed in Figure 27.



*Figure 27: Distance fields to several destinations*

Another benefit of Field  $D^*$ , comes from the fact that it can be also used as input of the RL Fleet Optimiser, depending on the approach used, for instance, as part of state space (see section 5.3). Computed paths may be used as reference paths to follow, but the goal of the optimiser is wider: it needs to determine the destination of each AMRs (allocating each AMR to a logistical task). Moreover, this kind of paths are computed on a static heatmap assumption: relying too much on it may reduce reactivities and anticipation capabilities.

When using the Field  $D^*$  as input of the RL, the global overview given in Figure 1 is updated as follows, in Figure 28.

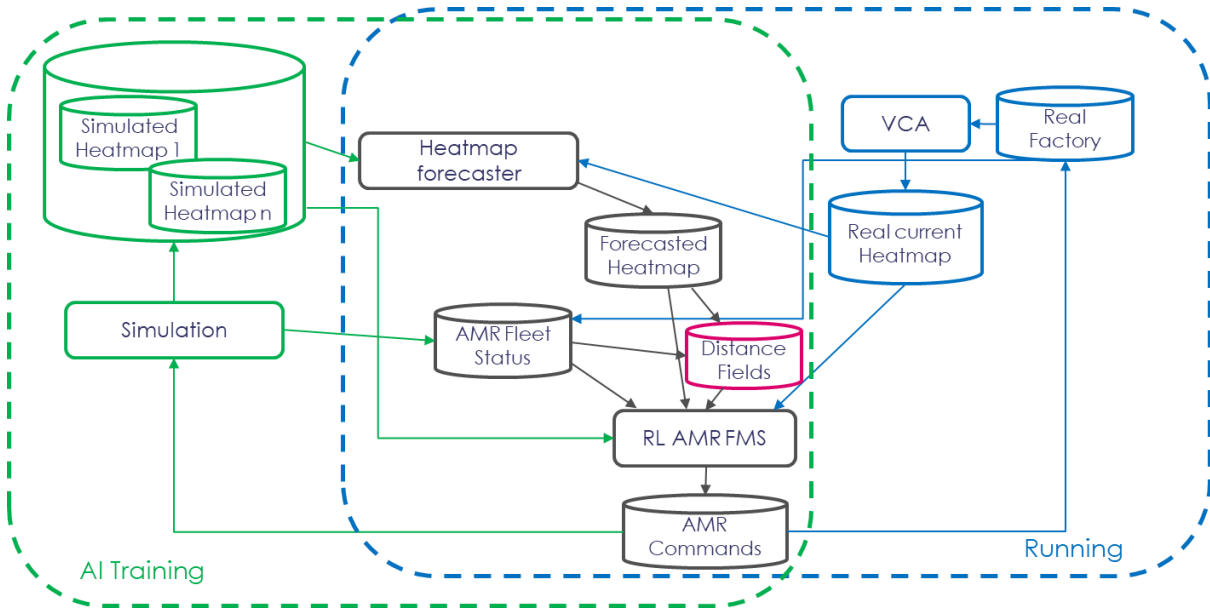


Figure 28: Adding Field Distance to RL input

Note that the “AMR Fleet Status” embeds the current positions of all AMRs, their remaining energetic autonomy, their characteristics such as max speed, but also the list of all logistic tasks to be performed.

### 5.3 RL Path Optimisation

The AMR Fleet optimisation relies on Reinforcement Learning (RL). It is trained by trial-and-error on the simulated environment with various factory layouts and outputs a decision-making policy that defines our AMRs’ behaviours. We call RL Agent this decision-making process, both during training and exploitation, as illustrated by Figure 29. To model the decision-making problem it has to solve, RL uses the Markov Decision Process (MDP) formalism [REF-28]. In particular, we focus on defining the state space  $S$ , action space  $A$  and reward function  $R$  of our agent.

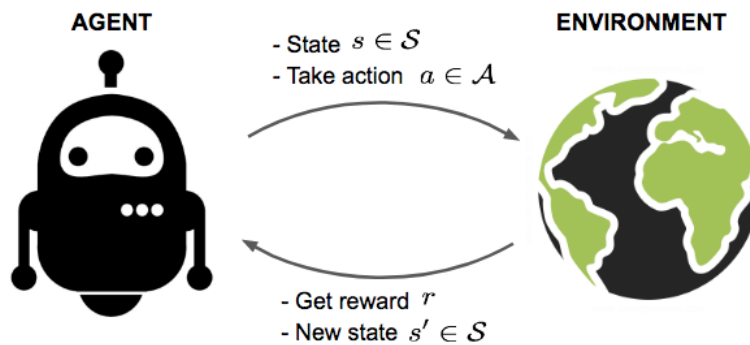


Figure 29: The Reinforcement Learning loop

As the action controls the different AMRs, it is modelled by joining the actions of each AMR. They perform logistic tasks: picking-up items from a working station and deposit them in other working stations. The training environment can handle the low-level behaviour, such as navigating precisely to working stations deposit points, avoiding pedestrians during the transit,

queueing before working station groups, etc. This will lead us to a relatively high-level discrete action space, e.g., choosing the next workstation group for each drone. On the other hand, we can use a more free “go to” action if we want the AMR to exhibit finer-grained behaviours such as patrolling between the working station groups as they wait for their next task.

A minimal state exploits the positioning and other properties (velocity, autonomy, etc) of the controlled AMRs, the current occupancy of the working stations and the current logistic tasks request for the fleet. We add-in information about the humans in the factory, thanks to the current and forecasted HeatMaps, that will affect how the AMRs navigate in the factory.

The primary objective of our reward function is to fulfil the logistics requests in due time and respect their eventual priority levels. We can add-in some secondary constraints and objectives such as minimising the AMR Fleet energy consumption.

As our state and actions are complex and highly dimensional, especially when we consider the multiple AMRs within the Fleet, we rely on Deep Reinforcement Learning (DRL) to train our agent. Initially, we identified two main directions for the modelling and solution.

- If we use a higher abstract level modelling, our optimisation problem exhibits similarities with the common Vehicle Routing Problem (VRP). We consider as a state representation the complete graph of the working station groups. Each node is a working station group with some features defining its current occupancy, capacity and expected availability time. Each edge holds the expected trip time for the AMRs, thanks to the HeatMap information and pathfinding capabilities. Thus, the RL agent computes the best set of routes between the different working stations for the AMR fleet. The nature of the logistic task request flows, the eventual priorities to manage, the constraints on the AMRs will probably make us deviate from the most classical VRP formulations that are handled with heuristic programming. RL in conjunction with Graph Attention Networks [REF-29] is a promising approach to solve a variety of routing problems without the need to redefine the approach as the modelling and objective changes. Possible shortcomings of this approach are a loss of real-world important information due to the abstract graph modelling, and that the VRP-like formulation does not fit our problem. For example, if the task requests involving very few routing possibilities for the AMRs.
- We can use a more complex and close-to-reality state. In this case, we can either mix the scalar information about AMRs, working stations, etc. with the image information of the HeatMaps, or try to represent all the state in the HeatMaps pixel representation. Thus, we train the Fleet controller agent with model-free DRL thanks to classical convolutional and fully connected Neural Network models. This is the most straightforward approach but we can expect scalability issues due to the number of AMRs, which leads to a high- and potentially variable-sized action space. To train a scalable and adaptive agent, we can leverage techniques from Multi-Agent Reinforcement Learning (MARL) such as a neural network model controlling each AMR but that is shared among all AMRs. The QMIX algorithm [REF-30] helps us better leverage global information sharing between all AMRs during the training and its extension REFIL [REF-31] helps dealing with variable number of AMRs. Possible shortcomings of this approach are an important and hard-to-evaluate need for hyper-parameter tuning, reward engineering, and training time before getting significant results. It is though ultimately more flexible than the graph-oriented approach.

Initially thoughts in the project where on improving the strategic part of the fleet management system (which robots to send where and when), see Figure 30.

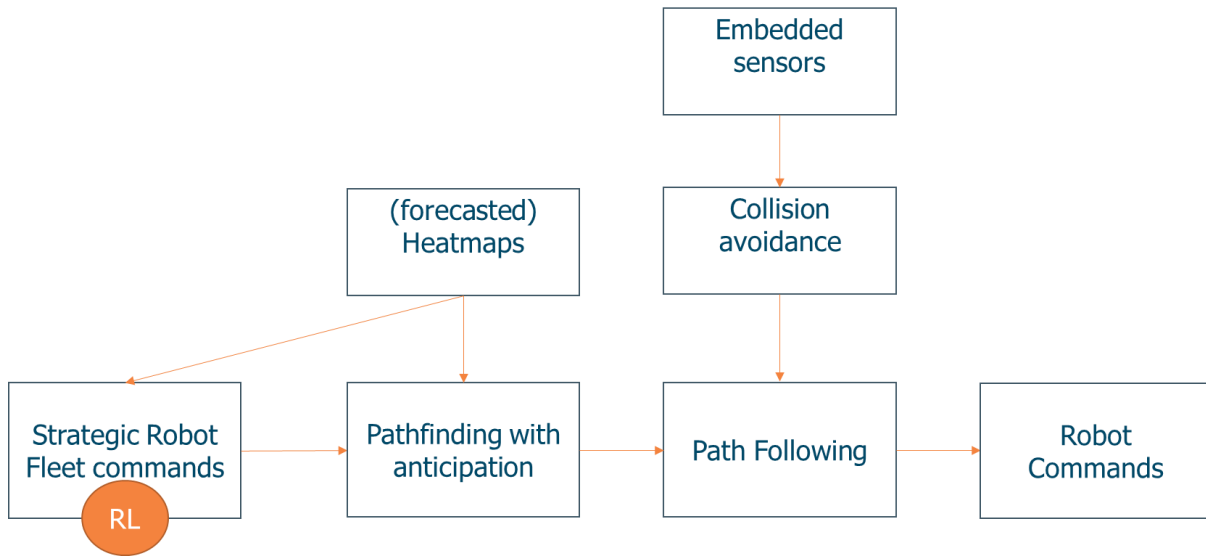


Figure 30: Using RL in the Strategic part

However, the strategic part seems too specific for each possible scenario. In other words, it is too much dependant on the number of robots, and on the tasks, they need to perform in general. By focusing instead on the path following, see below, it is easier to bring benefits to a Fleet Management System in short term, still being very generic which will lead to a faster adoption by end user.

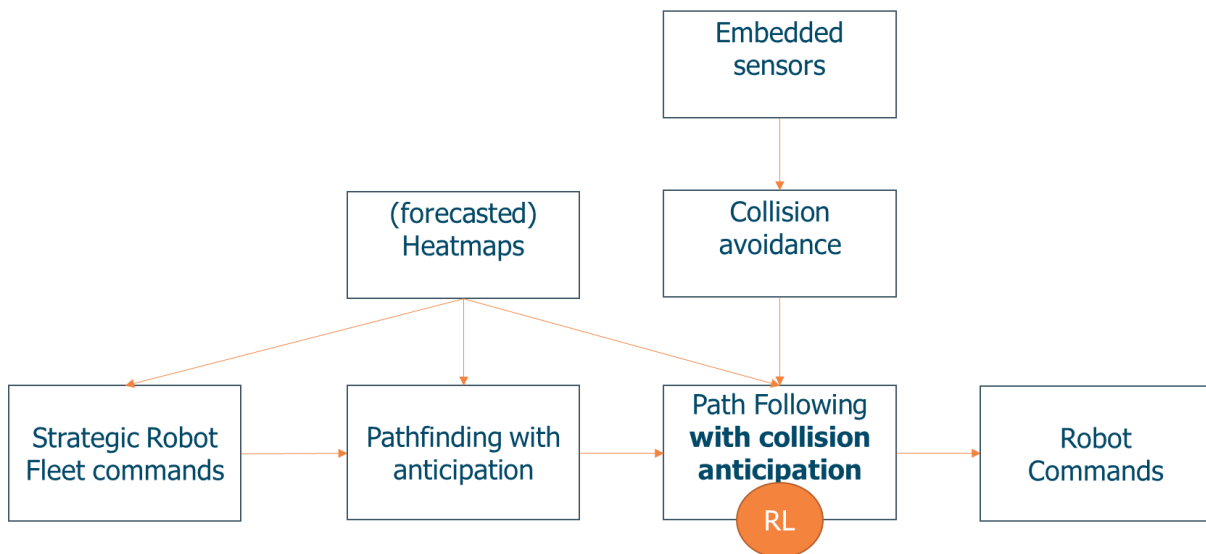


Figure 31: RL for Path Following

In particular, our approach leads to a better robots/human cohabitation since the collision avoidance is improved with better anticipation capabilities.

Existing works [REF-32] already use RL for path following among dynamic obstacles (pedestrians), by allowing robots not to strictly follow its guidance path, but still reaching its destination. This approach is illustrated in Figure 32.

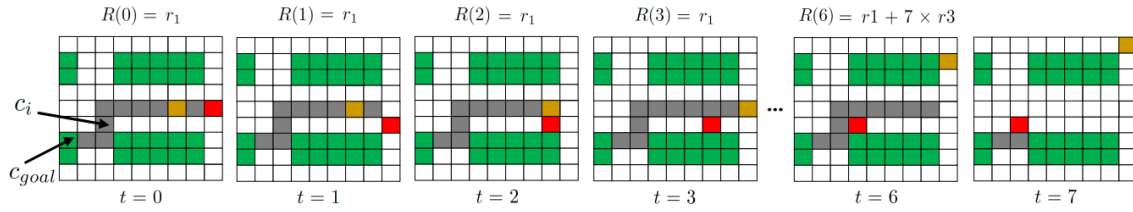


Figure 32: RL robot guidance

(The green, black, yellow and red cells represent the static obstacle, the global guidance, the dynamic obstacle and the robot, respectively)

Our first results following this approach were interesting and are illustrated in Figure 33.

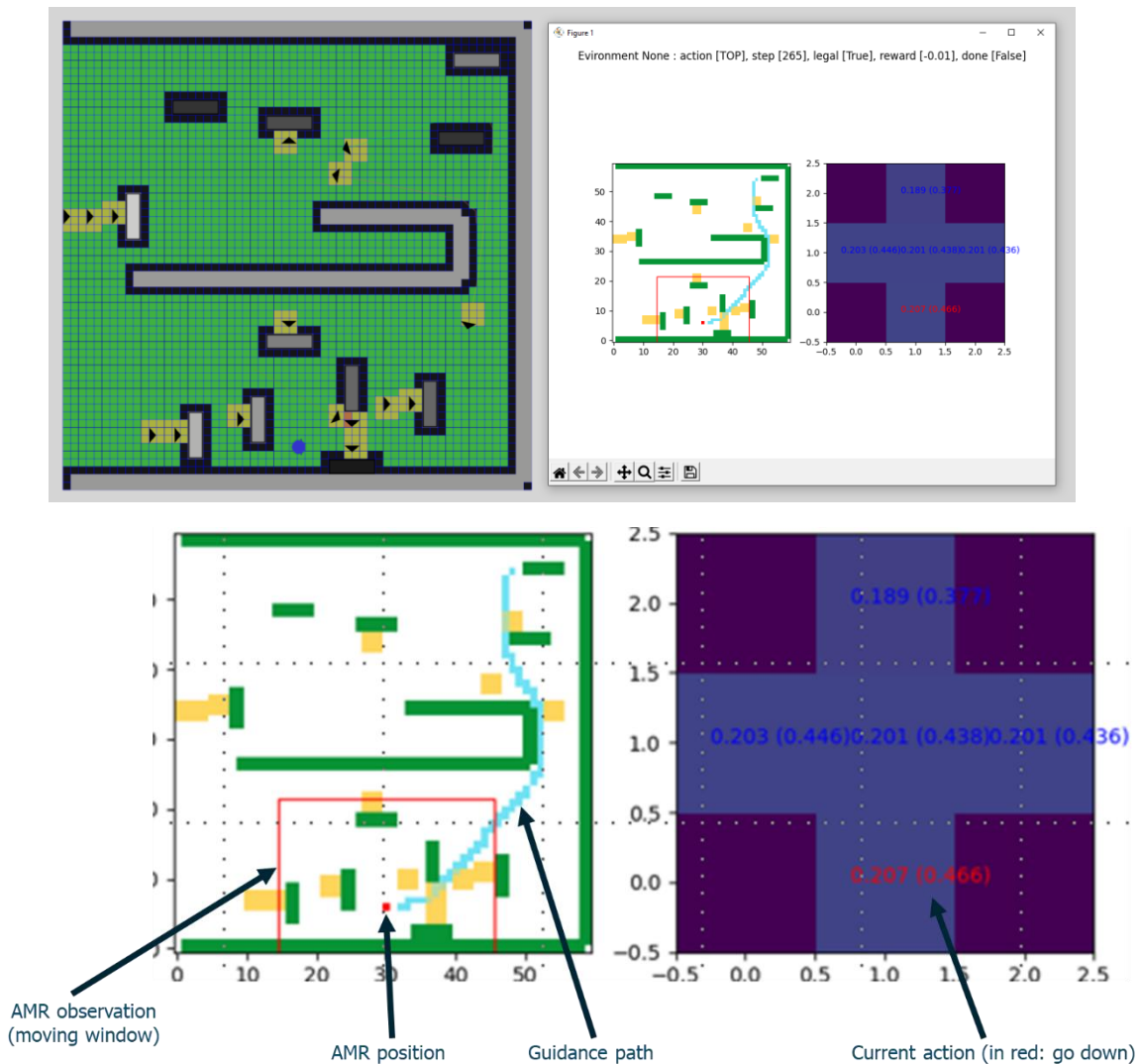


Figure 33: RL Guidance first results

The results highlighted two major ways for improvements:

- Using a distance map as reference instead of path;
- Increasing the number of possible actions from the initial 4+1 (left, right, up, down and stop) for the robot.

The first idea of improvement about replacing the guidance path by a guidance map comes from the fact that robots often need to make great detours to avoid queues of workers in front of working stations. To some extent, this is partially handled by our path-planning solution which takes average Heatmap as “costs”. In other words, guidance path is computed to avoid areas that are crowded in average. However, unexpected pedestrians’ congestion. Sometimes, robots need to follow another path (passing left of working station instead of following the initial idea to pass right for instance). We need to allow detours instead of only allowing deviations. This is done thanks to the distance field that is computed by our path-planning solution and changing the state space of the RL agent and the reward to fit with this new representation.

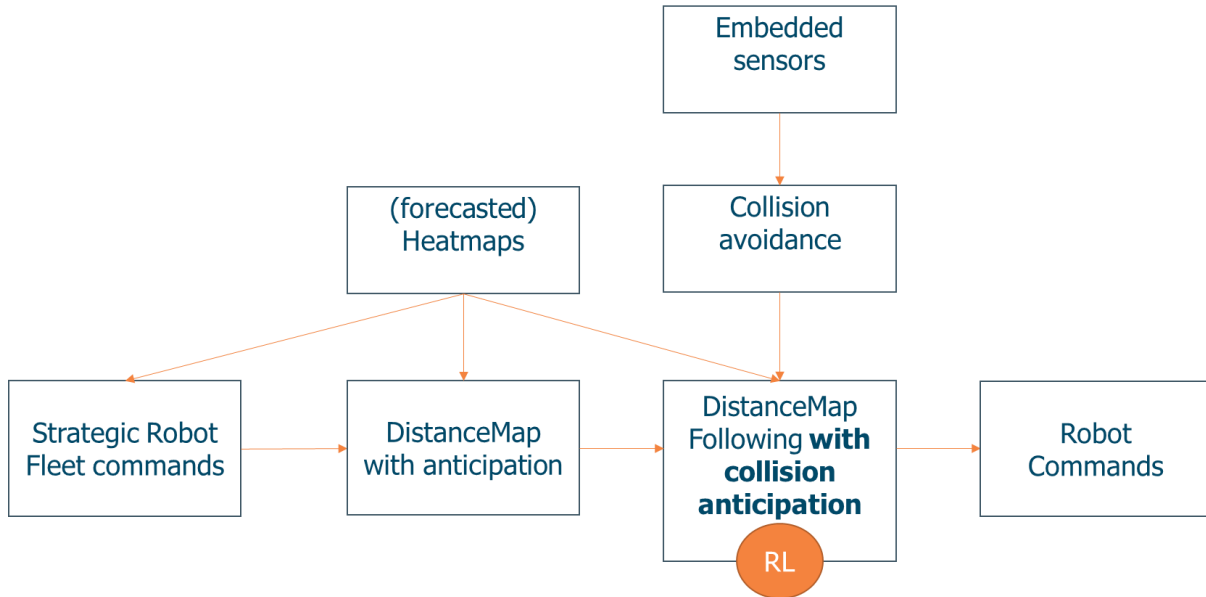
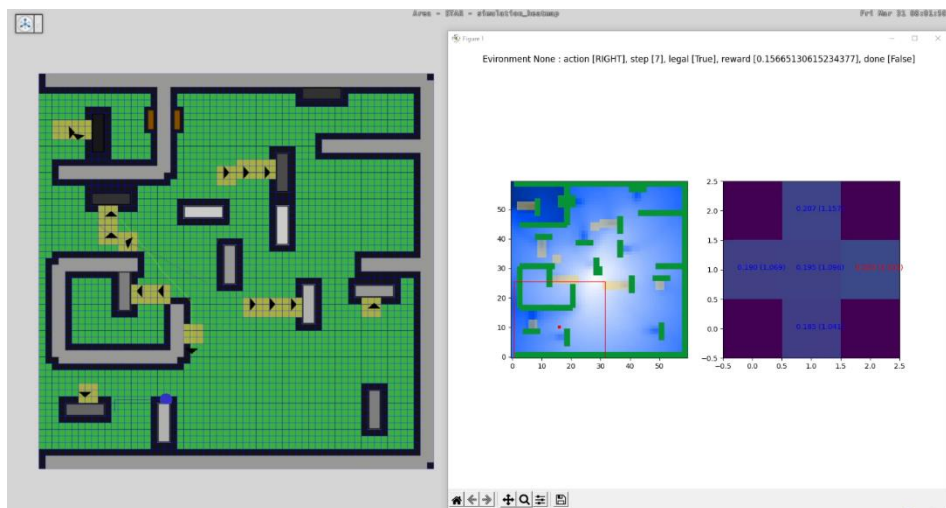


Figure 34: Replacing path guidance by distance map guidance



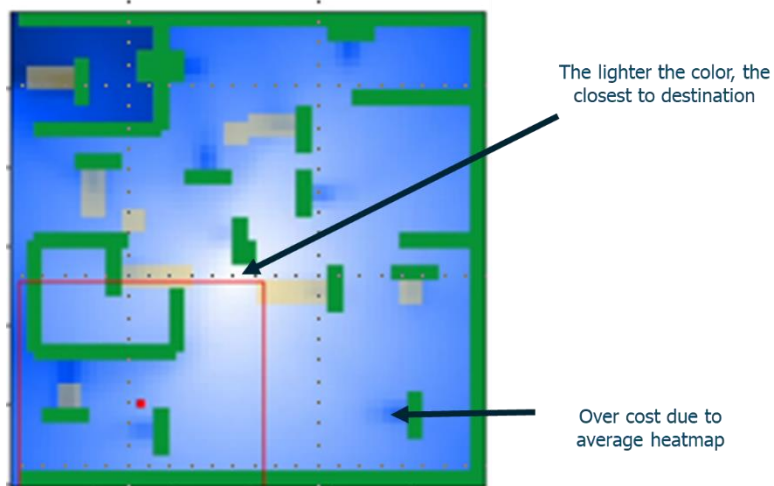


Figure 35: Use of Distance Map

Note that this approach allows to use the forecasted map easily by adding a new layer to the Convolutional Neural Network (CNN) used by the RL Agent.

As a second improvement, we choose to increase the number of possible actions to 36+1 (36 directions, each 10°, + stop). We decided to limit the number of actions to 36+1 because the Double Deep Q-Network (DDQN) algorithm do not scale well to higher actions number. However, it is still a great improvement from the initial 4+1 actions. Working with this new pseudo-continuous action space already brings new challenges. The main one comes from the fact that some actions may not lead to a change in the state space: moving in some directions may lead to stay in the same cell. In other words, the real position of the robot, which is now in a continuous space, is not exactly the perceived position, which is in a discrete space.

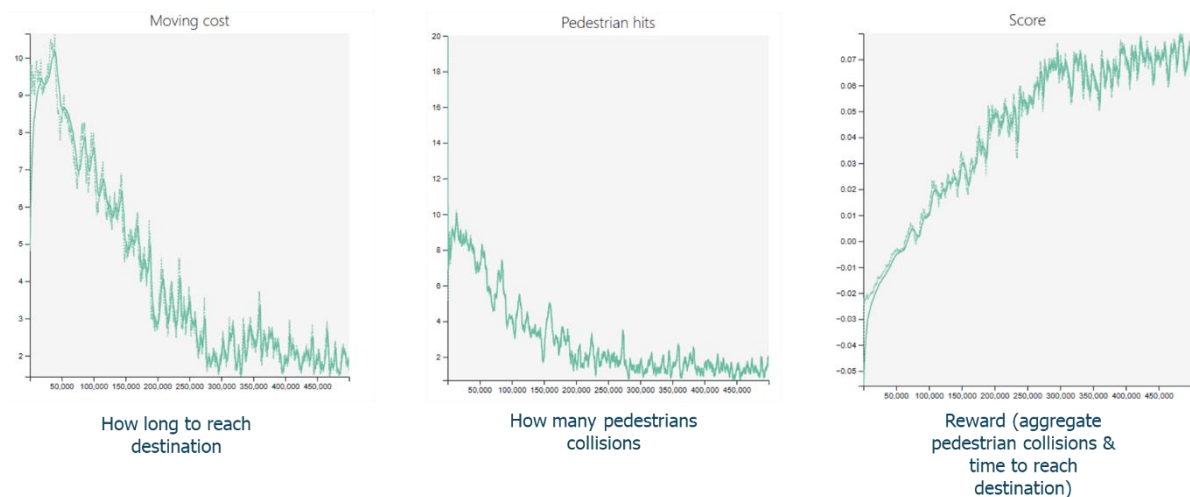


Figure 36: Learning curves

Thanks to a dashboard, we are able to monitor performance during learning as described in Figure 36.

## 6 Next Steps and Conclusion

Our approach, which is simulation based, allows us to test our solution in large virtual environments, with a great variety of situations.

An experimentation in a real environment is planned. Thanks to Human Digital Twin Framework (see Figure 37), our work in Task 5.4 will be plugged to other components, in particular the Task 5.3 Safety Zones Detection Modules, which brings the VCA populating the Heatmaps. At the same time, our AMRs Fleet Optimiser will also be linked to the real AMRs (a.k.a. AGVs). This will allow us to assess our solution in the DFKI Pilot of WP6.

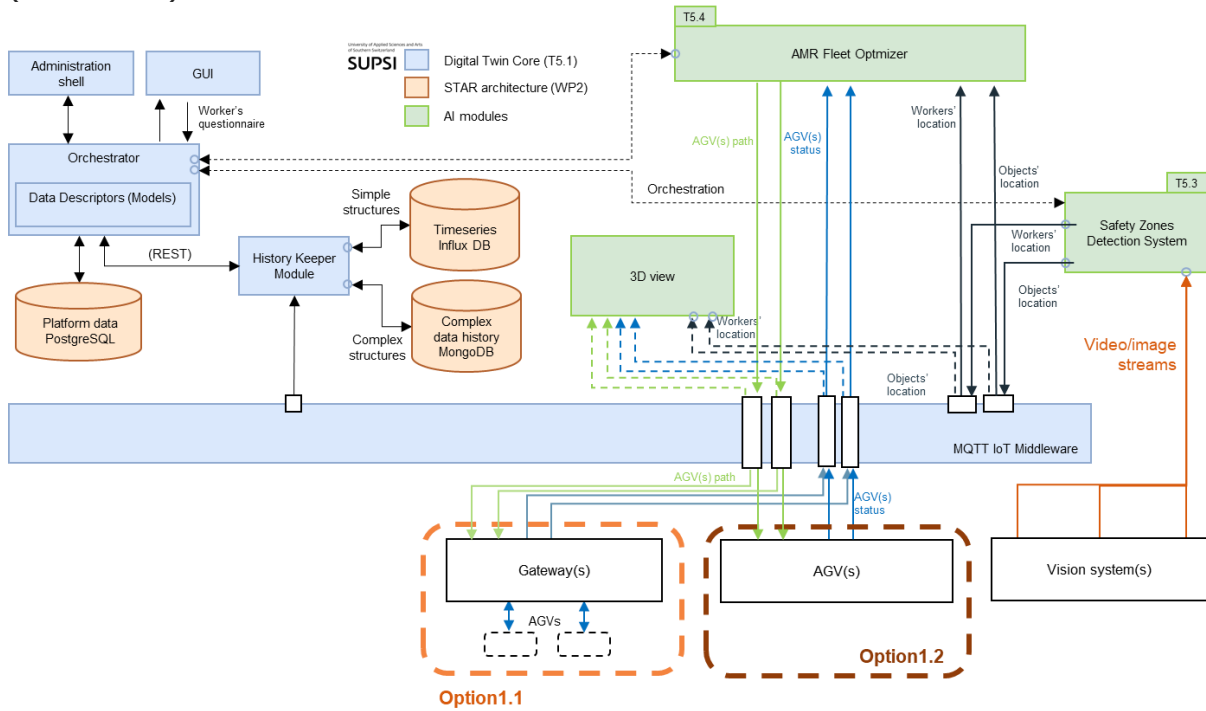


Figure 37: Human Digital Twin Framework

## References

Reference	Name of document
[REF-01]	<a href="https://www.oasys-software.com/products/massmotion/">https://www.oasys-software.com/products/massmotion/</a>
[REF-02]	<a href="https://www.onhys.com/">https://www.onhys.com/</a>
[REF-03]	<a href="https://www.bentley.com/software/legion/">https://www.bentley.com/software/legion/</a>
[REF-04]	Meyer Jean-Arcady, "Artificial Life and the Animat Approach to Artificial Intelligence", in Boden Margaret Ann, <i>Artificial Intelligence, A volume in Handbook of Perception and Cognition</i> , Academic Press, 1996
[REF-05]	Yang, Liang & Qi, Juntong & Xiao, Jizhong & Yong, Xia. "A literature review of UAV 3D path planning". Proceedings of the World Congress on Intelligent Control and Automation (WCICA). 2015.
[REF-06]	Dirk Helbing, Lubos Buzna, Anders Johansson, Torsten Werner, "Self-Organized Pedestrian Crowd Dynamics: Experiments, Simulations, and Design Solutions" <i>Transportation Science</i> 39. 2005
[REF-07]	Stephen. J. Guy, Jatin Chhugani, Changkyu Kim, Nadathur Satish, Ming Lin, Dinesh Manocha, Pradeep Dubey, "ClearPath: highly parallel collision avoidance for multi-agent simulation", SCA '09: The ACM SIGGRAPH / Eurographics Symposium on Computer Animation New Orleans Louisiana, 2009
[REF-08]	D. Zhang, Z. Xie, P. Li, J. Yu and X. Chen, "Real-time navigation in dynamic human environments using optimal reciprocal collision avoidance," 2015 IEEE International Conference on Mechatronics and Automation (ICMA), 2015
[REF-09]	M. Nicolau, D. Perez-Lieba, M. O'Neill and A. Brabazon, "Evolutionary Behavior Tree Approaches for Navigating Platform Games," in IEEE Transactions on Computational Intelligence and AI in Games, vol. 9, no. 3, 2017
[REF-10]	D. Ferguson and A. Stentz. "Field D*: An Interpolation-Based Path Planner and Replanner" Proceedings of the International Symposium on Robotics Research, 2005
[REF-11]	Rudenko, A., Palmieri, L., Herman, M., Kitani, K. M., Gavrila, D. M., & Arras, K. O. (2020). Human motion trajectory prediction: A survey. <i>The International Journal of Robotics Research</i> , 39(8), 895–935.
[REF-12]	F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The Graph Neural Network Model," <i>IEEE Trans Neural Netw</i> , vol. 20, no. 1, pp. 61–80, 2009, doi: 10.1109/TNN.2008.2005605.
[REF-13]	Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," <i>IEEE Trans Neural Netw Learn Syst</i> , vol. 32, no. 1, pp. 4–24, Jan. 2021, doi: 10.1109/TNNLS.2020.2978386.
[REF-14]	Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Graph Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting," <i>CoRR</i> , vol. abs/1707.01926, 2017, [Online]. Available: <a href="http://arxiv.org/abs/1707.01926">http://arxiv.org/abs/1707.01926</a>
[REF-15]	A. Mohamed, K. Qian, M. Elhoseiny, and C. Claudel, "Social-STGCNN: A Social Spatio-Temporal Graph Convolutional Neural Network for Human Trajectory Prediction," 2020.

[REF-16]	A. Leigh, J. Pineau, N. Olmedo, and H. Zhang, 'Person tracking and following with 2D laser scanners', Proc. - IEEE Int. Conf. Robot. Autom., vol. 2015-June, no. June, pp. 726–733, 2015, doi: 10.1109/ICRA.2015.7139259.
[REF-17]	Y. Ban, X. Alameda-Pineda, F. Badeig, S. Ba, and R. Horaud, 'Tracking a varying number of people with a visually-controlled robotic head', IEEE Int. Conf. Intell. Robot. Syst., vol. 2017-September, pp. 4144–4151, 2017, doi: 10.1109/IROS.2017.8206274.
[REF-18]	C. Medina Sánchez, M. Zella, J. Capitán, and P. J. Marrón, 'From Perception to Navigation in Environments with Persons: An Indoor Evaluation of the State of the Art', Sensors, vol. 22, no. 3, pp. 1–33, 2022, doi: 10.3390/s22031191
[REF-19]	S. Breuers, L. Beyer, U. Rafi, and B. Leibel, 'Detection- Tracking for Efficient Person Analysis: The DetTA Pipeline', IEEE Int. Conf. Intell. Robot. Syst., pp. 48–53, 2018, doi: 10.1109/IROS.2018.8594335.
[REF-20]	S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool, 'You'll never walk alone: Modeling social behavior for multi-target tracking', Proc. IEEE Int. Conf. Comput. Vis., no. Iccv, pp. 261–268, 2009, doi: 10.1109/ICCV.2009.5459260.
[REF-21]	A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, 'Social LSTM: Human trajectory prediction in crowded spaces', Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2016-December, pp. 961–971, 2016, doi: 10.1109/CVPR.2016.110.
[REF-22]	M. Goldhammer, K. Doll, U. Brunsmann, A. Gensler, and B. Sick, 'Pedestrian's trajectory forecast in public traffic with artificial neural networks', Proc. - Int. Conf. Pattern Recognit., pp. 4110–4115, 2014, doi: 10.1109/ICPR.2014.704.
[REF-23]	C. Emmanouilidis, E. Rica, and B. Duqueroie, 'Anticipating human presence for safer worker – robot shared workspaces', Proceedings of the 2023 IFAC World Congress, Yokohama, Japan.
[REF-24]	V. Karasev, A. Ayvaci, B. Heisele, and S. Soatto, 'Intent-aware long-term prediction of pedestrian motion', Proc. - IEEE Int. Conf. Robot. Autom., vol. 2016-June, pp. 2543–2549, 2016, doi: 10.1109/ICRA.2016.7487409.
[REF-25]	J. Bai et al., "A3T-GCN: Attention Temporal Graph Convolutional Network for Traffic Forecasting," ISPRS Int J Geoinf, vol. 10, no. 7, 2021, doi: 10.3390/ijgi10070485.
[REF-26]	L. Zhao et al., "T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction," IEEE Transactions on Intelligent Transportation Systems, vol. 21, no. 9, pp. 3848–3858, 2020, doi: 10.1109/TITS.2019.2935152.
[REF-27]	B. Rozemberczki et al., "PyTorch Geometric Temporal: Spatiotemporal Signal Processing with Neural Machine Learning Models," in Proceedings of the 30th ACM International Conference on Information & Knowledge Management, in CIKM '21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 4564–4573. doi: 10.1145/3459637.3482014.
[REF-28]	Puterman, Martin L. "Markov Decision Processes: Discrete Stochastic Dynamic Programming." (1994).
[REF-29]	Kool, Wouter, Herke van Hoof, and Max Welling. "Attention, Learn to Solve Routing Problems!." International Conference on Learning Representations. 2018.

[REF-30]	Rashid, Tabish, et al. "QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning." ICML. 2018.
[REF-31]	Iqbal, Shariq, et al. "Randomized Entity-wise Factorization for Multi-Agent Reinforcement Learning." International Conference on Machine Learning. PMLR, 2021.
[REF-32]	B. Wang, Z. Liu, Q. Li, A. Prorok "Mobile Robot Path Planning in Dynamic Environments Through Globally Guided Reinforcement Learning" <a href="https://arxiv.org/abs/2005.05420">https://arxiv.org/abs/2005.05420</a>