

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: INTRASOFT International



Funded by the Horizon 2020
Framework Programme of the
European Union

DELIVERABLE

D5.7 – Reinforcement Learning Techniques for AMRs-Initial version

Dissemination level	PU -Public
Type of Document	Report
Contractual date of delivery	31/03/2022
Deliverable Leader	THALES SIX GTS FRANCE
Status - version, date	Final - v1.0, 31/03/2022
WP / Task responsible	WP5/T5.4
Keywords:	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 targets the description of the first version of our demonstrator of a Reinforcement Learning (RL) based Automatic Mobile Robots (AMRs) Fleet Management System (FMS). 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 sending AMRs, at a planning level. AMRs will still embed integrated collision avoidance mechanisms for last second avoidance. Our focus is on the planning of the fleet, not reactive avoidance, which is already covered by state of the art.

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 Technique to anticipate the evolution of human presence in the factory
- It allows demonstrate our solution in a large-scale scenario, which may not be available in the scope of the STAR project.
- It brings technical primitives such as pathfinding that can be used as a high-level action to the RL FMS.
- It allows to train a Reinforcement Learning agent to act as an FMS

This document is aligned with the current state of our work, which focused on the simulation of the environment, as a first step. The RL work has begun but it is too early to describe any results yet.

Deliverable Leader:	Bertrand Duqueroie (THALES)
Contributors:	Christos Emmanouilidis (University of Groningen) Alexandre Kazmierowski (THALES)
Reviewers:	Spyros Theodoropoulos (UNIFI)
Approved by:	Charalampos Ipektsidis, John Soldatos (INTRA)

Document History			
Version	Date	Contributor(s)	Description
0.1-0.2	18/03/2022	Bertrand Duqueroie	ToC and the first draft
0.3-0.5	22/03/2022	Bertrand Duqueroie Alexandre Kazmierowski	Additions to the draft deliverable
0.6	28/03/2022	Christos Emmanouilidis	Review of the document
0.7	28/03/2022	Bertrand Duqueroie	Minor corrections
0.8	30/03/2022	Bertrand Duqueroie Christos Emmanouilidis	Corrections following reviewers comments
0.9	30/03/2022	Bertrand Duqueroie	Ready for submission
1.0	30/03/2022	INTRA	Final QA'ed version.

Table of Contents

EXECUTIVE SUMMARY	2
TABLE OF CONTENTS.....	4
TABLE OF FIGURES.....	5
DEFINITIONS, ACRONYMS AND ABBREVIATIONS	6
1 INTRODUCTION.....	7
1.1 APPROACH	7
1.2 OVERVIEW	8
2 HEATMAP CONCEPT.....	9
2.1 HEATMAP DESCRIPTION	9
3 SIMULATION CAPABILITIES	12
3.1 GENERIC SIMULATION ENGINE	12
3.2 HEATMAP SIMULATED SAMPLES	14
3.3 GENERALIZATION AND REALISM	17
4 HEATMAP FORECASTER.....	19
4.1 HUMAN TRAJECTORY PREDICTION	19
4.2 PRIORITISATION OF METHODS	20
5 AMR FLEET CONTROLLER	22
5.1 OBJECTIVES OF THE FLEET.....	22
5.2 PATHFINDING.....	22
5.3 RL GLOBAL OPTIMIZATION.....	24
6 NEXT STEPS AND CONCLUSION	27
REFERENCES	28

Table of Figures

FIGURE 1: OVERVIEW OF THE APPROACH	8
FIGURE 2: TOY EXAMPLE	9
FIGURE 3: 2D GRID STRUCTURE	10
FIGURE 4: FROM GRID TO HEATMAP.....	10
FIGURE 5: HEATMAP RAW LOG.....	11
FIGURE 6: SE-STAR SIMULATION TOOL PRESENTATION	12
FIGURE 7: FACTORY SIMPLE EXAMPLE	14
FIGURE 8: SIMULATED WORKERS.....	15
FIGURE 9: GEOMETRIC LAYOUT	16
FIGURE 10: LAYOUT CONFIGURATION FILE	17
FIGURE 11: FROM WORKER SIMULATION TO AN AVERAGE HEATMAP	17
FIGURE 12: GENERALIZATION	18
FIGURE 13: SINGLE AND MULTI-MODEL PHYSICS-BASED PREDICTION	20
FIGURE 14: AREA TO AVOID (THE HOTTER THE COLOUR, THE MORE IT NEEDS TO BE AVOIDED).....	22
FIGURE 15: DISTANCE FIELDS TO SEVERAL DESTINATIONS	23
FIGURE 16 ADDING FIELD DISTANCE TO RL INPUT	24
FIGURE 17: THE REINFORCEMENT LEARNING LOOP.....	24

Definitions, Acronyms and Abbreviations

Acronym/ Abbreviation	Title
WP	Work Package
AMR	Automatic Mobile Robot
FMS	Fleet Management System
RL	Reinforcement Learning
ML	Machine Learning
AI	Artificial Intelligence
VCA	Video Content Analytics
RVO	Reciprocal Velocity Obstacle
ORCA	Optimal Reciprocal Collision Avoidance
SSO	Service Oriented Object

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 kind of robots in changing environment, without the need for a 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 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 the 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. Optimizing “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 the 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 in the robots. Moreover, the occlusion that embedded sensors face will not limit our global situation awareness. Having a clear picture of where the obstacles are (that could temporary be left on possible paths), and analyzing human presence is mandatory to optimize efficiently robot fleet behavior.

The anticipation of human presence within the factory is a challenge on its own. Machine Learning is key here to extrapolate the current situation in a 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 optimized 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, optimized commands must be very fast to compute, in order 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 solution has been made famous thanks to impressive results where AI beats professional human players in different games such as Chess, Go, or StarCraft2. Here 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

As explained in the introduction, the goal of our work 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 the real cameras of a real factory. This makes the link with the previous STAR Task 5.3 and described in D5.5 Visual Scene Analysis for Safety Zones Detection.

Therefore, the next section (Section 2) will focus on describing this Heatmap data structure.

Section 3 focuses on the simulation capabilities available in our Task and how these are used to generate many different situations, which can be used both as a learning data set for ML or as an environment for RL.

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

Finally in section 5, a first glance of the AMR Fleet controller work is given. However, at this stage of the project, the focus is made on the simulation, which leads to fewer details on the RL part that will be improved later.

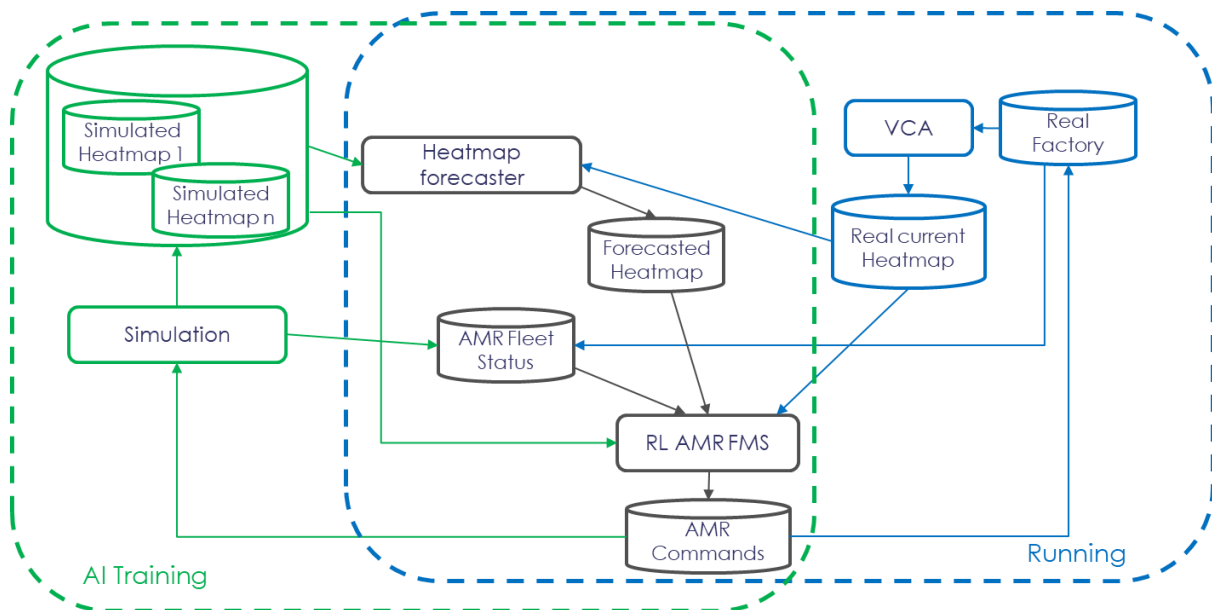


Figure 1: Overview of the approach

2 HeatMap Concept

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

Thanks to a Heatmap forecasting component (ML based, trained with simulated data set), the RL can have better anticipation capabilities on where the human 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:

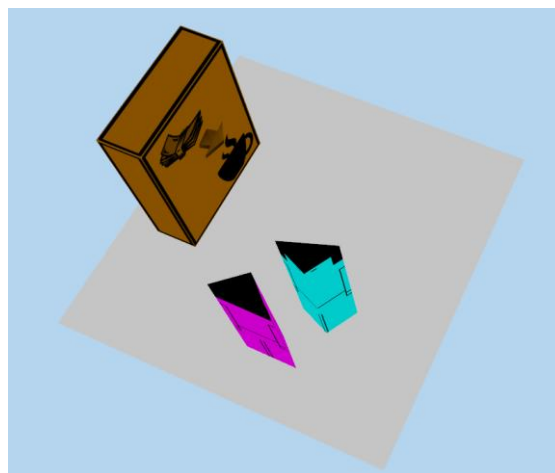


Figure 2: Toy example

In the example illustrated above, we have two humans and one obstacle, in a small square environment.

A Heatmap can be created to describe this environment, thank to camera analytics. Note that the positions of the cameras are not relevant here. We suppose they can convert their output (localization of human and object in their field of view) to a global coordinates system. If several cameras have a field of view that overlap, then their results should be merged when producing the Heatmap (this process is out of the scope of this Heatmap description). If there are some blind spots due to a lack of camera or 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 type of localization systems.

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

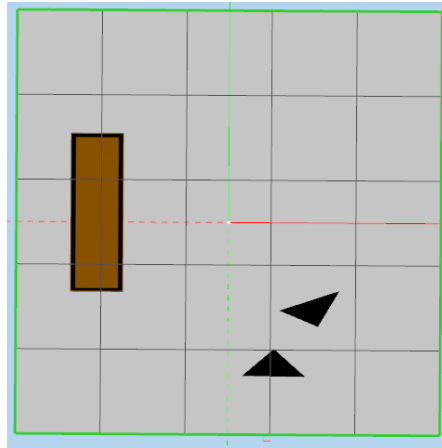


Figure 3: 2D Grid structure

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

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

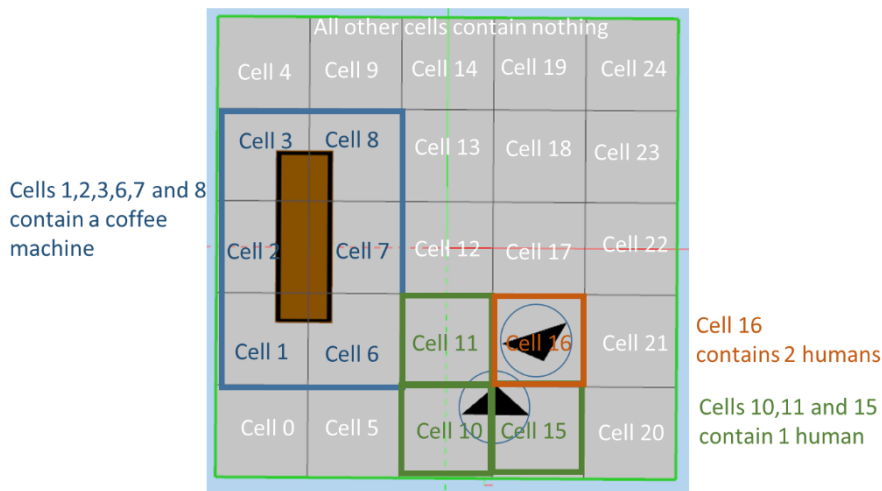


Figure 4: From grid to Heatmap

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.

So, 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 in line 1 (thus, there is an offset of one between row number and the cell index).

“coffee1” describes the fact that the obstacle on the left have been identified as a coffee machine by video analytics. Most of the obstacles do not need to be identified correctly. However, it can be useful to know which object can be a destination of human or AMR and where those are located. This means that identifying working stations can help.

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, Onhys, or Bentley Legion. 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 optimized 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 innovating technologies in the Synthetic Environment field and is developed by Thales. SE-Star (the engine) 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 optimizing their own motivations. In particular, a factory with workers movement can now be simulated, thanks to STAR task 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

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-01]), 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-07], 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 making the link with the heatmap concept. 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-03],[REF-04], with more or less realistic models. The one used in SE-Star is an approach derivate from RVO/ORCA [REF-05], which allows a good compromise between realism and computation load. Last second Collision-avoidance mechanisms of SE-Star could also be used as part of the controller for the AMR. However, those AMR already embeds such mechanisms, with more dedicated solution for robots.

Equipment

Besides virtual actors, all the equipment and devices are modelled independently from them. For that, the logic and behaviour of all 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 our 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 includes the fact that they may require dependencies to be used: “using” a working station of type 1 may require to have used a working station of type 2 before. 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.

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 more scripted one, derived from behaviours tree [REF-06],[REF-07] is used. It allows to define business processes, merged with behaviors that are more natural, and still let emerge a large variety of situation thanks to some random parameters.

Logging

Depending on the use case, different output 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.

3.2 HeatMap Simulated Samples

Several factory-like environments have been simulated in the project. A simple one, in particular, is presented here in order to provide a more explicit example. This is obviously a nonrealistic factory.

This sample factory is a square of 100meters 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.

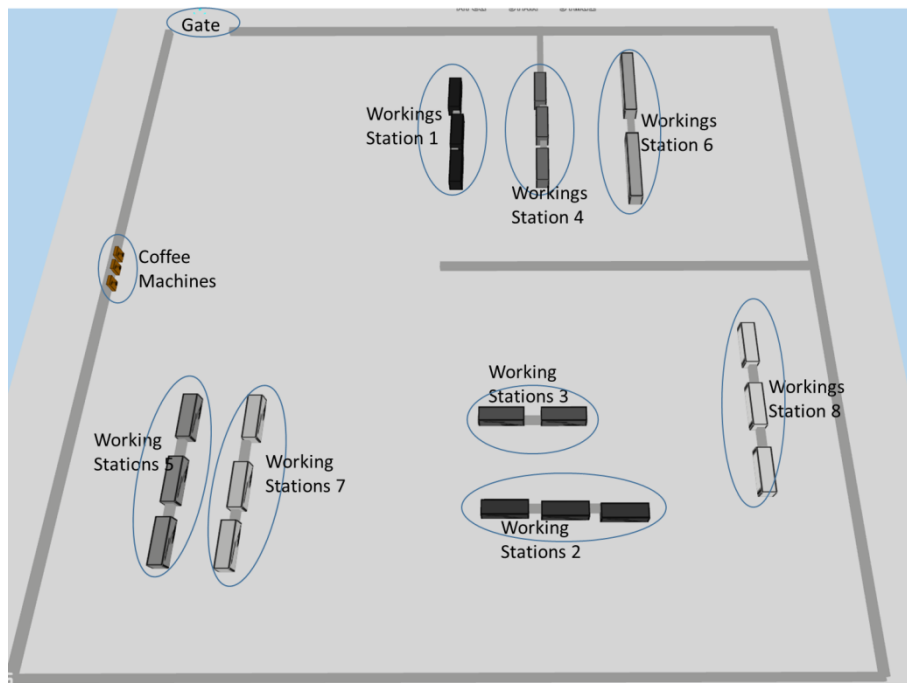


Figure 7: Factory simple example

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

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



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 and take a coffee first (probability of 0.1).

Working at working station of type 1, 2, 4, 5, 7 and 8 last in average 30 seconds. Working at the stations of type 3 and 6 last in average 10 seconds. But of course, those timings are just 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.

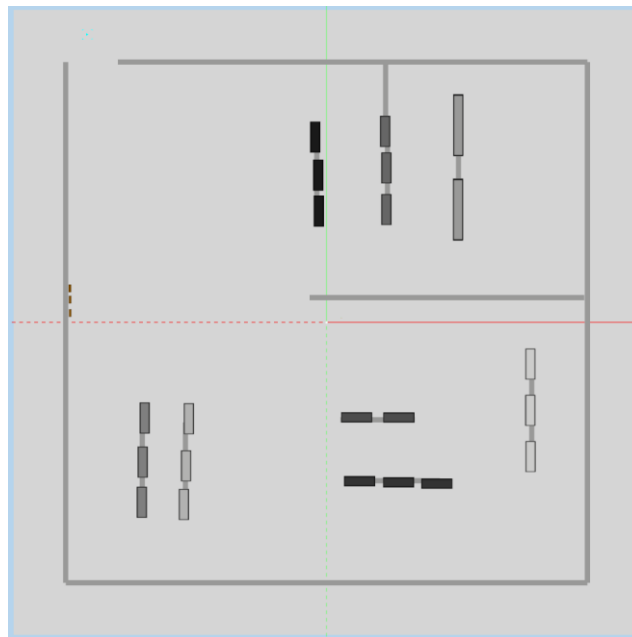


Figure 9: Geometric layout

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="(-49.16,4.38,0.00)" yprDegree="(0.00,0.00,0.00)"><!--coffee machine: size = (0.5,1.5,2.0)meters -->
<Object model="Coffee Machine" name="coffee2" pos="(-49.20,6.53,0.00)" yprDegree="(0.00,0.00,0.00)">
<Object model="Coffee Machine" name="coffee3" pos="(-49.15,1.79,-0.00)" yprDegree="(0.00,0.00,0.00)">
<Object model="WorkingStation1" name="WS1A" pos="(-2.17,35.59,0.00)" 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)" yprDegree="(0.00,0.00,0.00)">
<Object model="WorkingStation1" name="WS1C" pos="(-1.50,21.37,0.00)" yprDegree="(0.00,0.00,0.00)">
<Object model="WorkingStation2" name="WS2A" pos="(6.34,-30.50,0.00)" yprDegree="(90.00,0.00,0.00)">
<Object model="WorkingStation2" name="WS2B" pos="(13.81,-30.71,-0.00)" yprDegree="(90.00,0.00,0.00)">
<Object model="WorkingStation2" name="WS2C" pos="(21.14,-30.97,-0.00)" yprDegree="(90.00,0.00,0.00)">
<Object model="WorkingStation3" name="WS3A" pos="(5.77,-18.24,0.00)" yprDegree="(90.00,0.00,0.00)">
<Object model="WorkingStation3" name="WS3B" pos="(13.86,-18.27,0.00)" yprDegree="(90.00,0.00,0.00)">
<Object model="WorkingStation4" name="WS4A" pos="(11.46,21.68,0.00)" yprDegree="(0.00,0.00,0.00)">
<Object model="WorkingStation4" name="WS4B" pos="(11.47,29.67,0.00)" yprDegree="(0.00,0.00,0.00)">
<Object model="WorkingStation4" name="WS4C" pos="(11.23,36.69,0.00)" yprDegree="(0.00,0.00,0.00)">
<Object model="WorkingStation5" name="WS5A" pos="(-34.86,-18.35,0.00)" yprDegree="(0.00,0.00,0.00)">
<Object model="WorkingStation5" name="WS5B" pos="(-35.24,-26.82,-0.00)" yprDegree="(0.00,0.00,0.00)">
<Object model="WorkingStation5" name="WS5C" pos="(-35.41,-34.54,0.00)" yprDegree="(0.00,0.00,0.00)">
<Object model="WorkingStation6" name="WS6A" pos="(25.20,21.67,0.00)" scale="(1.00,2.00,1.00)" 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="(25.25,27.88,0.00)" scale="(1.00,2.00,1.00)" yprDegree="(0.00,0.00,0.00)">
<Object model="WorkingStation7" name="WS7A" pos="(-26.40,-18.50,0.00)" yprDegree="(0.00,0.00,0.00)"><!--middle wall, just to break the flow of people -->
<Object model="WorkingStation7" name="WS7B" pos="(-26.92,-27.54,0.00)" yprDegree="(0.00,0.00,0.00)">
<Object model="WorkingStation7" name="WS7C" pos="(-27.35,-34.96,0.00)" yprDegree="(0.00,0.00,0.00)">
<Object model="WorkingStation8" name="WS8A" pos="(39.06,-16.89,-0.00)" yprDegree="(0.00,0.00,0.00)"><!--working station (with the lightest color) -->
<Object model="WorkingStation8" name="WS8B" pos="(39.11,-25.77,-0.00)" yprDegree="(0.00,0.00,0.00)">
<Object model="Wall" pos="(-1.87,28.99,0.20)" 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="(11.67,29.33,0.20)" 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)" 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 basically the purpose of the Heatmap to give almost the same information. This means that neither the Heatmap forecaster component nor the RL Optimizer will take this "Objects.xml" as input. It is given just as background information.

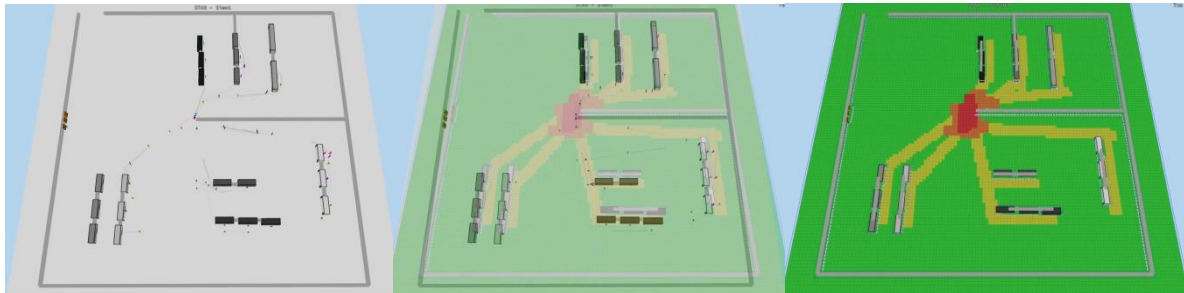


Figure 11: From worker simulation to an average Heatmap

Both Heatmap forecaster component and the RL Optimizer 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 shows 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, in average, there is the more chance to find workers.

3.3 Generalization and Realism

Another benefit of the SE-Star simulator is its capabilities to set a new environment for a low effort. In particular, thanks to new development in the STAR project, we can change the factory layout easily, or change the proportion of workers population easily. 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 a similar situation in training than the one it will face during its exploitation, when deployed in a real environment.

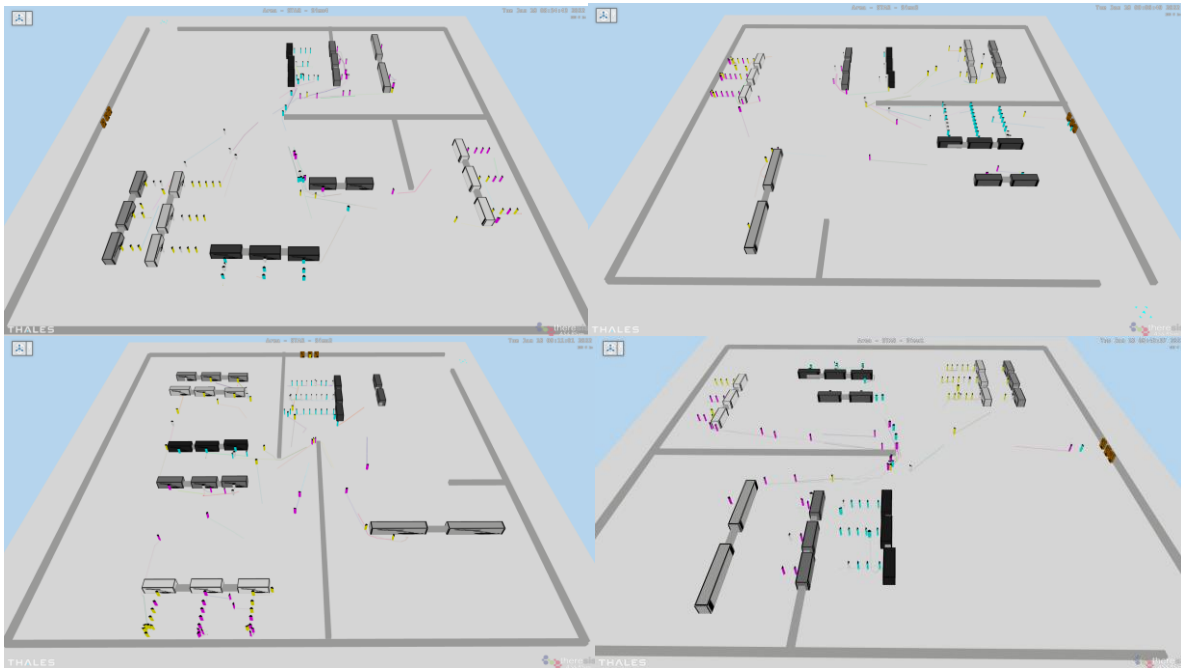


Figure 12: Generalization

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. 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 advanced capabilities in human behaviours modelling of SE-Star, the simulated data produced is 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 a lot 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 makes it difficult to perceive the difference between simulated data and real one. For instance, the position of human workers are not analysed as (x,y) precise data, but instead through the cells of the Heatmap. Meaning that the Heatmap creates a discretization of the environment, depending on its resolution, that smooth 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 on precision. However, the final goal of the AMRs Fleet Optimizer is to avoid, during planning, the areas where it is possible that workers are present. In this context, taking margin is nice to have, meaning less precision is not an issue, by rounding up cell occupancy: as soon as a worker (defined by a point with a radius) overlaps with a cell, this cell is tagged as containing the worker. Same apply to objects in the factory (cf. Figure 4). Note that later on, AMRs Optimizer will also use a minimal distance to obstacles (to cell containing an obstacle to be precise) when performing pathfinding.

4 HeatMap Forecaster

4.1 Human Trajectory Prediction

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 is immaterial for the forecasting and it could 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 applicable to this problem [REF-8]. 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-9]. This can extend to growing projected trajectories and employing the Minimum Description Length (MDL) principle for making a selection among the candidate trajectories [REF-10]. Although the idea has been applied for outdoors tracking, the principle also holds for indoors cases [REF-11].

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-11]. 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-12]. 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-13].

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-13].

Next we present the candidate approaches that have been provisionally selected to explore for the Heatmap-based human movement prediction process at STAR.

4.2 Prioritisation of Methods

Among the aforementioned methods, a shortlist of candidate approach is as follows:

Physics-based:

This will 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:

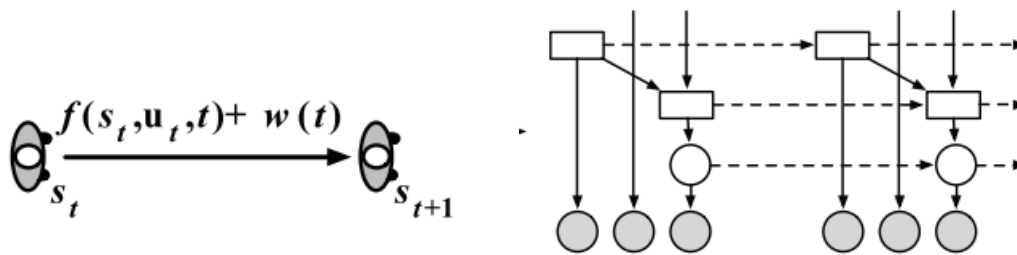


Figure 13: single and multi-model physics-based prediction

However, in this case the problem cannot be posed in control terms therefore

Pattern-based methods

These can follow sequential and non-sequential patterns. Due to the obvious relevance of time and spatial proximity, the sequential patterns will be 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 $s_{t-n:t}$ current time instance, :

$$s_t + 1 = f(s_{t-n:t})$$

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.

Selected methods

Different formulations for physics and pattern-based methods will be employed, including simple filter-based methods to account for the influence of the time and spatial proximal grid cell values, on the future heatmap – based values. These proximities will also be replicated and mapped into the structure of a Convolutional Neural Network – based approach for human movement prediction, with appropriate alignment and stacking of the inputs, according to the proximity assumption, but with enough free parameters to be learned through algorithmic training of the filter and CNN-based approaches.

5 AMR fleet controller

5.1 Objectives of the fleet

Basically, 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 help sorting all tasks. As explained previously, the commands sent to each AMR should take into account the human presence. To some extends, it relies on well-known technology such as pathfinding.

5.2 Pathfinding

In particular approaches such as the Field D* [REF-07], 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 cost (that tells how much the cell should be avoided) depending on how many people there is in this temporal average. Figure 14 illustrates such temporal average, and gives an overview of where the areas than needs to be avoided are. In this example, four thresholds are used to discretize the cell average occupancy, displayed in four different colours.

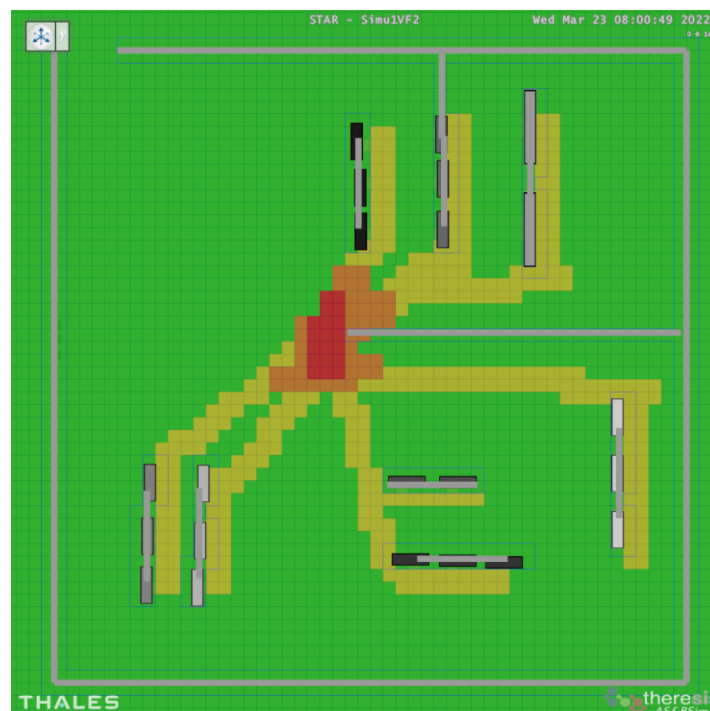


Figure 14: Area to avoid (the hotter the colour, the more it needs to be avoided)

Note again that the RL Optimizer will take more than the temporal average Heatmap as input, but instead all the instantaneous one (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 leads to destination. Integrating such distance fields then produces paths for AMRs that avoid crowded areas, as displayed in Figure 15.

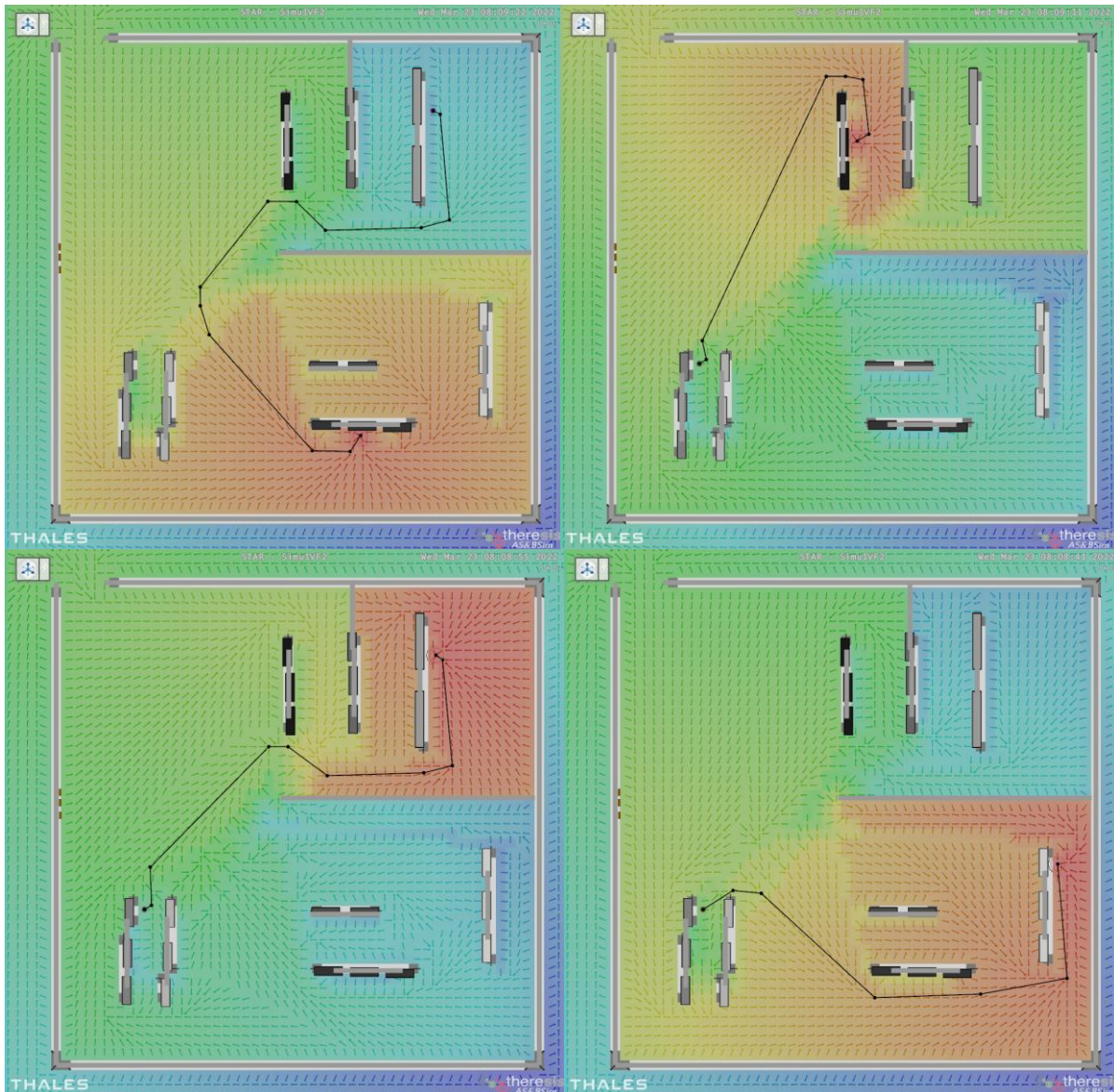


Figure 15: Distance fields to several destinations

Another benefit of Field D*, comes from the fact that it can be also used as input of the RL Fleet Optimizer, depending on the approach used, for instance, as part of state space (see section 5.3). Computed paths may be used as reference path 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 path are computed on a static heatmap assumption: relying too much on it may reduce reactivities and anticipation capabilities. Note again that at this stage of the project, we are still investigating several modelling approaches for the RL part.

When using the Field D* as input of the RL, the global overview given in Figure 1 is updated as following, in Figure 16.

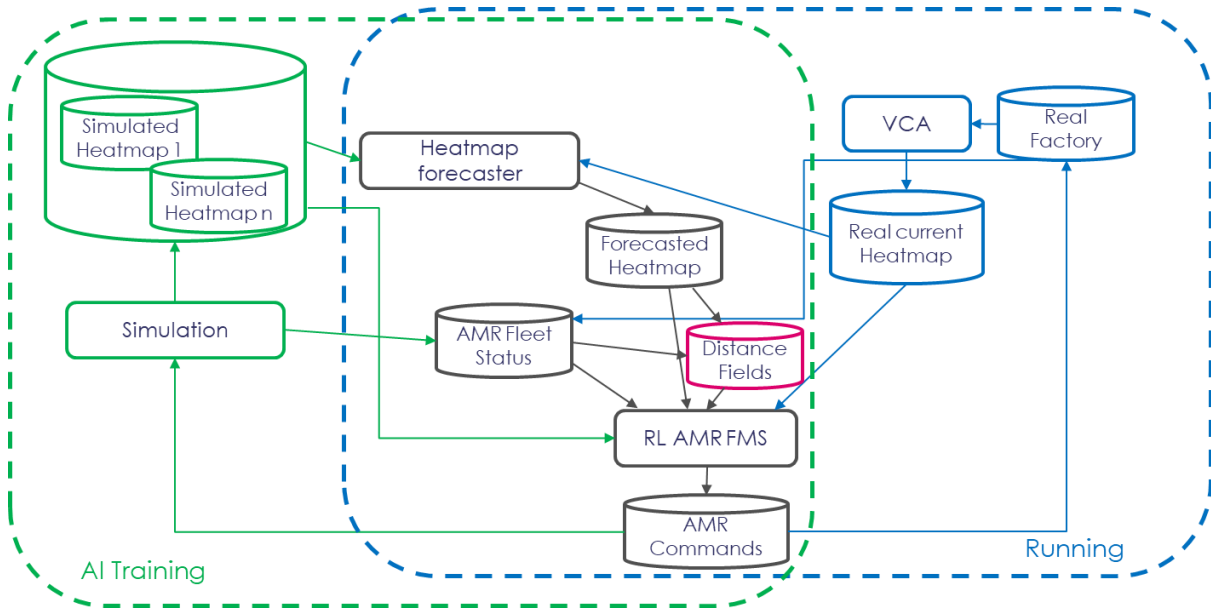


Figure 16 Adding Field Distance to RL input

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 all of logistic tasks to be performed.

5.3 RL Global Optimization

The AMR Fleet optimization 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 17. To model the decision-making problem it is to solve, RL uses the Markov Decision Process (MDP) formalism [REF-16]. In particular, we focus on defining the state space S , action space A and reward function R of our agent.

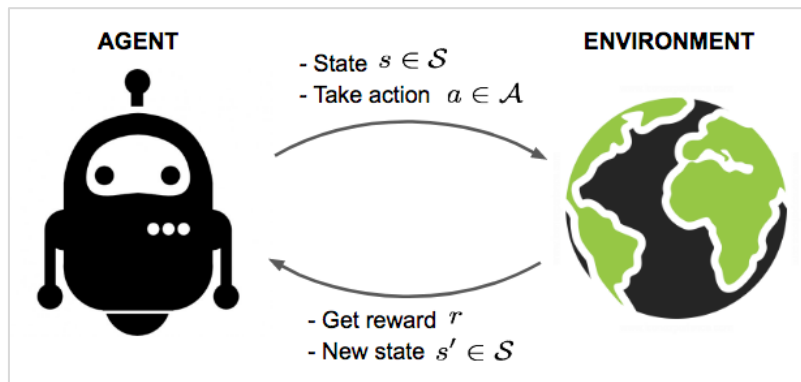


Figure 17: The Reinforcement Learning loop

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 minimizing the AMR Fleet energy consumption.

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

- If we use a higher abstract level modelling, our optimization 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-17] is a promising approach to solve a variety of routing problems without the need to redefine the approach as the modelling and objective change. 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 involve 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-18] helps us better leverage global information sharing between all AMRs during the training and its extension REFIL [REF-19] helps dealing with a 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.

6 Next Steps and Conclusion

As explained in this document, the focus of our work has been more on the simulation environment than on the RL Optimizer. This simulation work has allowed working in parallel on the Heatmap AI forecaster. Moreover, it will allow us to test our solution in large virtual environments, with a great variety of situations.

An experimentation in a real environment is still planned. Thanks to the Human Digital Twin Framework (see Figure 17), 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 Optimizer 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.

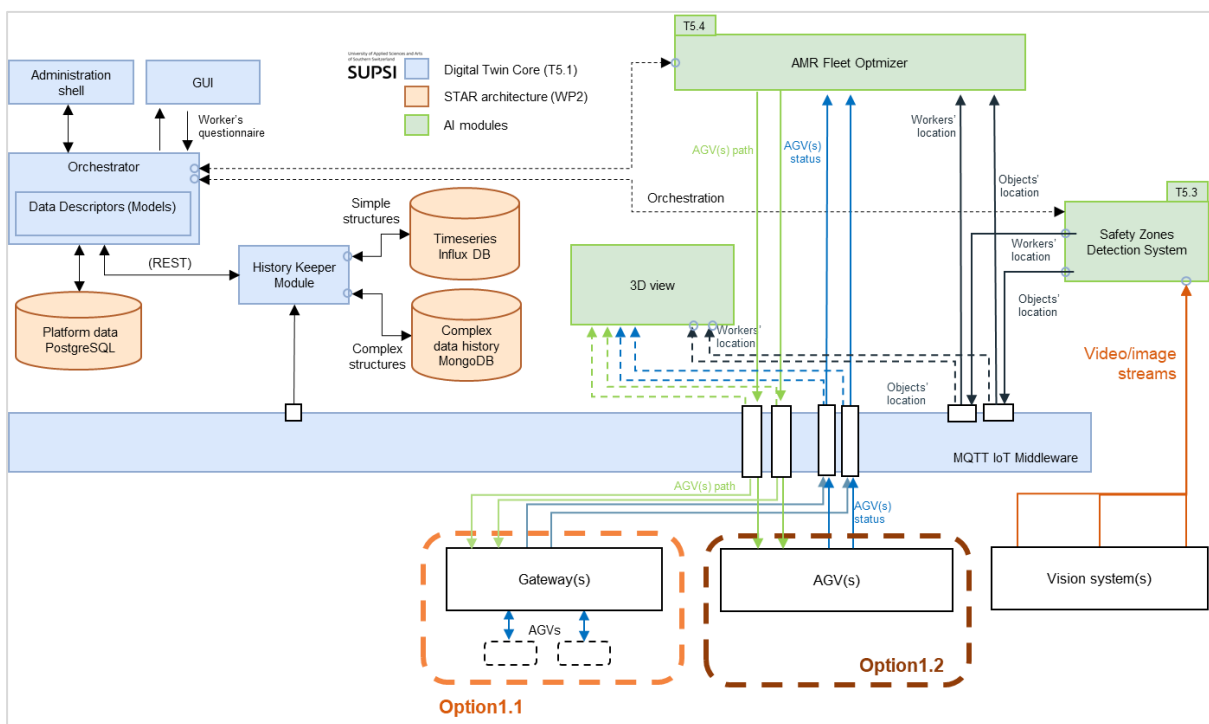


Figure 17 Human Digital Twin Framework

References

Reference	Name of document
[REF-01]	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-02]	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-03]	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-04]	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-05]	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-06]	M. Nicolau, D. Perez-Liebana, 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-07]	D. Ferguson and A. Stentz. "Field D*: An Interpolation-Based Path Planner and Replanner" Proceedings of the International Symposium on Robotics Research, 2005
[REF-08]	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-09]	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-10]	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', <i>Sensors</i> , vol. 22, no. 3, pp. 1–33, 2022, doi: 10.3390/s22031191
[REF-11]	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-12]	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-13]	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-14]	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-15]	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-16]	Puterman, Martin L. "Markov Decision Processes: Discrete Stochastic Dynamic Programming." (1994).
[REF-17]	Kool, Wouter, Herke van Hoof, and Max Welling. "Attention, Learn to Solve Routing Problems!." International Conference on Learning Representations. 2018.
[REF-18]	Rashid, Tabish, et al. "QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning." ICML. 2018.
[REF-19]	Iqbal, Shariq, et al. "Randomized Entity-wise Factorization for Multi-Agent Reinforcement Learning." International Conference on Machine Learning. PMLR, 2021.