Project no.: **610917**

Project full title: **Sustainable and reliable robotics for part handling in manufacturing**

Project Acronym: **STAMINA**

Deliverable no.: **D2.5**

Title of the deliverable: **Multi-Robot Coordination and mapping**

Contractual Date of Delivery to the CEC:	**31.10.2016**
Actual Date of Delivery to the CEC:	**31.10.2016**
Organisation name of lead contractor for this deliverable:	**INESC Porto**
Author(s):	**Rafael Arrais, Germano Veiga, Pedro Gomes Costa, António Paulo Moreira**
Participants(s):	**P01, P06**
Work package contributing to the deliverable:	**WP2**
Nature:	**R**
Version:	**1.0**
Total number of pages:	**22**
Start date of project:	**01.10.2013**
Duration:	**42 months – 31.03.2017**

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 610917

Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Abstract:

This deliverable addresses the Multi-Robot Coordination and Mapping problems within the project STAMINA. A path planning method to coordinate a robotic fleet and a spatial inconsistencies detection system are presented.

Keyword list: Multi-Robot coordination, Routing, Automated Guided Vehicles, Scheduling, Advanced Logistics, Hierarchical Spatial Representation and Detection, World Model

Document History

Version	Date	Author (Unit)	Description
0.1	Sep 15, 2016	Rafael Arrais, Germano Veiga (INESC Porto)	Started document.
0.2	Sep 22, 2016	Rafael Arrais, Pedro Gomes da Costa, António Paulo Moreira (INESC Porto)	First draft with multi-robot coordination.
0.3	Sep 28 2016	Rafael Arrais, Germano Veiga (INESC Porto)	Second draft with dynamic mapping.
0.4	Sep 30, 2016	Rafael Arrais (INESC Porto)	Some adjustments to the previous draft.
0.5	Oct 7, 2016	Rafael Arrais, Germano Veiga (INESC Porto)	Minor improvements to the overall document.
0.6	Oct 10, 2016	Rafael Arrais, Germano Veiga (INESC Porto)	Draft for review.
0.7	Oct 21, 2016	Rafael Arrais, Germano Veiga (INESC Porto)	Updated draft after revision by the consortium.

Table of Contents

1 Executive Summary

The coordination of a multi-robot system and mapping in a dynamic logistic supermarket is an interesting scientific challenge. Due to the constant changes to the layout of the logistic supermarket, in order to cope with a variable production, there is an implicit need to increase the flexibility of the STAMINA robotic fleet.

A deliberate effort to enhance the capacity of the robotic fleet to adapt to dynamic environments has been conducted, in the sense of developing mechanisms to ensure coordination between a multi-robot system.

As a result, a coordinated multi-robot path planning methodology has been developed, to deal not only with the constant changes in the configuration of the logistic supermarket, but also with the matter of having a moving fleet of robots, which instant positions and future trajectories must be accounted for, in a dynamic way.

Additionally, a continuously monitoring of the consistency of the virtual representation of the logistic supermarket has been developed. The spatial inconsistency detection system, by using the sensing capacity of the robotic fleet traversing the logistic supermarket to assess the reliability of the virtual model, can increase the accuracy of the model, thus enabling the robots to perform in a more effective way.

2 Introduction

2.1 Scope and Objectives

This document aims to report the developments regarding Task 2.5: "Multi-robot coordination and mapping". The main objective associated with this task is to provide a multi-robot framework in which the robots can efficiently coordinate their moves and exchange information about the environment.

The developed mechanisms require an exchange of information not only to coordinate the robotic fleet movement, but also to build a global map of the environment, by fusing the data provided by each individual robot.

2.2 Document organization

Following this introductory chapter, the organization of the document is as follows:

- Multi-Robot Coordination (chapter 3) – This chapter presents the developed path planning methodology, a scheduling module, which enhances the planning capacity, and some accessory tools developed.
- Automatic Inconsistency Detection (chapter 4) – Chapter 4 focus on the spatial inconsistency detection, explaining the approach and the connection with the STAMINA architecture.

3 Multi-Robot Coordination

This chapter presents the developments for multi-robot coordination, and is divided into 5 sub-chapters. Firstly, a definition of the multi-robot coordination concept is presented, followed by an overview of some of the scientific advancements in the field. Sub-chapter 3 focuses on the Time Enhanced A* methodology, an online procedure to compute the path of a multi-robot system. Sub-chapter 4 presents the scheduling module developed to enhance the proposed path-planning methodology. Finally, some accessory tools, developed to aid the concepts presented previously, are addressed.

3.1 Definition

In the context of a multi-robot system in a confined area, such as a logistic supermarket, the application of a path planning algorithm must be considered, as to ensure collision free routes. In an industrial environment, specifically in a logistic supermarket, the problem of ensuring a collision free route can lead to a more complex challenge, due to the variability associated with the physical configuration of the environment.

According to the STAMINA architecture, the mission planner component is responsible for generating mission assignments for individual robots in the fleet. Missions are created using information on the status of the robots and the set of available kitting orders, and are defined as a goal assignment to a specific robot for completing a single trip around the kitting zone.

Considering the robotic fleet control system, the task planner component is responsible for the allocation process of a single robot to each mission, corresponding to a unique kitting order. If the kitting orders for a given planning period are known in advance, and the environment can be considered static, the problem of allocating missions to the robotic fleet can be solved through an offline procedure. However, in a real industrial scenario, several changes and unpredictable events can lead to alterations to the kitting orders list. If a robot of the fleet is stationary due to a malfunction, or if an obstacle is detected, the defined planning could be compromised.

In order to cope with practical industrial constrains, a multi-robot coordination module was developed to compute the path of each robot considering not only the position and movement of other robots of the fleet, but also the introduction of obstacles to the environment. The developed module allows the online computation of a conflict free route for each robot and for each kitting order, being easily adapted to new configurations of logistic supermarkets and also to a robotic fleet of variable size.

3.2 Industrial multi-robot coordination overview

When considering the task of allocating routes for a multi-robot system, the possible approaches fall into two categories: online and offline (Le-Ahn, 2015). The offline approach specifies that all available orders are scheduled at once. With an offline approach, if there is an unforeseen change, the generated scheduling needs to be reviewed and updated. The online approach dictates that the scheduling decisions are taken in a dynamic way. As such, short-termed decisions are made over time according to changes in the system state (Sabuncuoglu, 2000). This way of proceeding ensures that unpredictable changes in the scenario can be controlled efficiently.

Regarding optimization of routes, several approaches are presented in the literature to address the route that a robot should follow to minimize the travelled distance and to avoid collisions and deadlocks. A deadlock is said to occur when a shared resource is simultaneously requested by two or more entities (robots). In this situation, the opposing parties are in a stalemate, and their execution will be frozen indefinitely.

3.3 Time Enhanced A*

The development of the multi-robot coordination module arises from the need of an online procedure to compute the path of the robots, suitable for multi-robot systems and that also avoids collisions, deadlocks and guarantees the efficient execution of a set of missions. As such, a multi-mission algorithm that incrementally builds the path of each robot considering the movements of other robots of the fleet and the position of eventual obstacles was developed.

The developed **Time Enhanced A*** (Santos, 2015) is based on the search algorithm A* (Hart, 1968), pronounced as "A star", that is noted for its performance and accuracy on plotting an efficiently traversable path between two locations in a mapped area. The A* search algorithm is widely used in several areas of artificial intelligence concerned with path finding problems, ranging from industrial applications to computer games.

The developed methodology enhances the traditional search method A* with a third dimension, time. As such, besides the map's two-dimensional coordinate system, denoting free and occupied cells, time is represented in a layered fashion. The developed Time Enhanced A* is an online method, that in each iteration computes the shortest path between two locations in space, over the temporal layers, considering the robots' positions, obstacles and changes introduced to the environment.

The introduction of a time component to the task of path planning on a multi-robot system is an important component in efficiently predicting the position of robots during execution. Through computations of each robot path by portraying time as a third dimension using a layered representation, as depicted in Figure 1, future possible collisions can be identified and repaired in the beginning of the computation of the path.

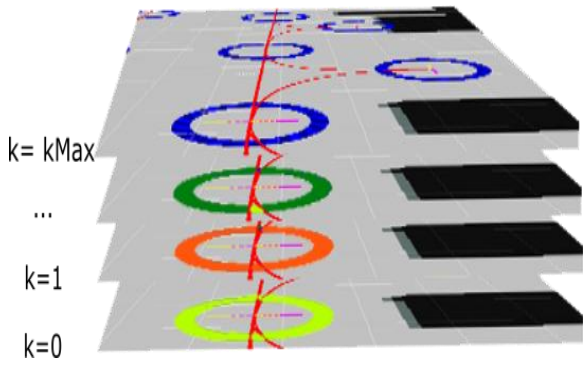


Figure 1 - TEA Layered representation of time.*

On calculating the path for a single robot, much like the A* algorithm, the next analysed neighbouring cell is evaluated using a cost function, given by the distance between the initial location and the final location. Due to the addition of time, the developed mechanism has to consider two additional aspects:

1. **The analysed neighbourhood cells include the cell containing the current position of the robot.** This allows a robot to consider the space occupied by other robots as occupied cells. In order to avoid potential deadlock situations, a given robot occupies only a single cell in space on each layer. In this situation, each robot only flags the current space as occupied, and not the past and future paths, thus allowing a more efficient path planning and, as a result, a decrease in the occurrences of deadlock situations.
2. **The analysed neighbourhood cells belong to the next temporal layer.** As the mechanism is iterative, the analysis of the path of each robot in a given time needs to be computed using the next layer, representing the immediate future. The number of layers is dependent on the number of iterations to achieve the objective location of the mission and on the dimension of the map.

The STAMINA skill structure defines a set of actions that are ordered with the objective of completing a kitting order. In the context of the path planning mechanism, each stage of the mission, corresponding to the execution of a robotic skill, can be seen as a sub-mission.

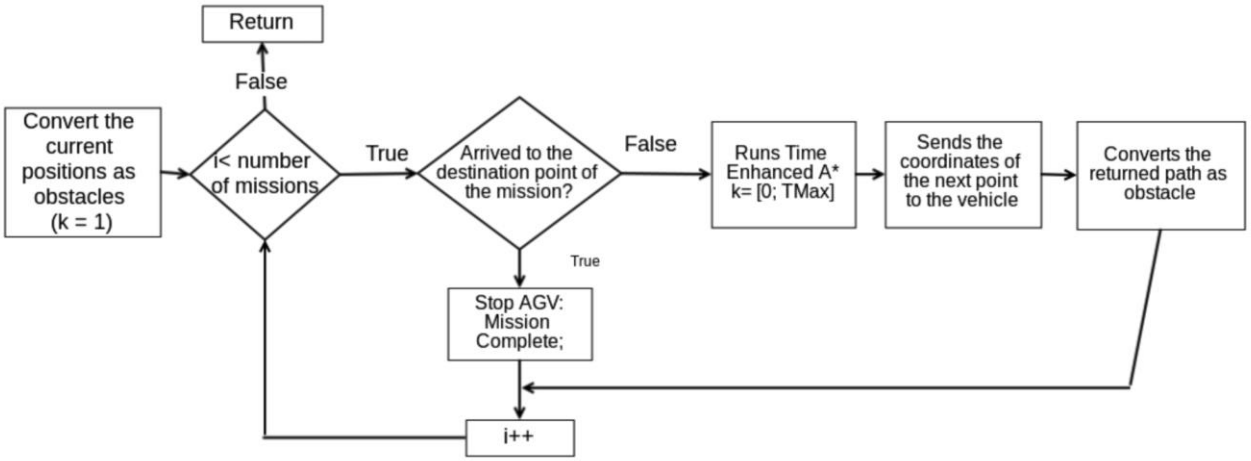


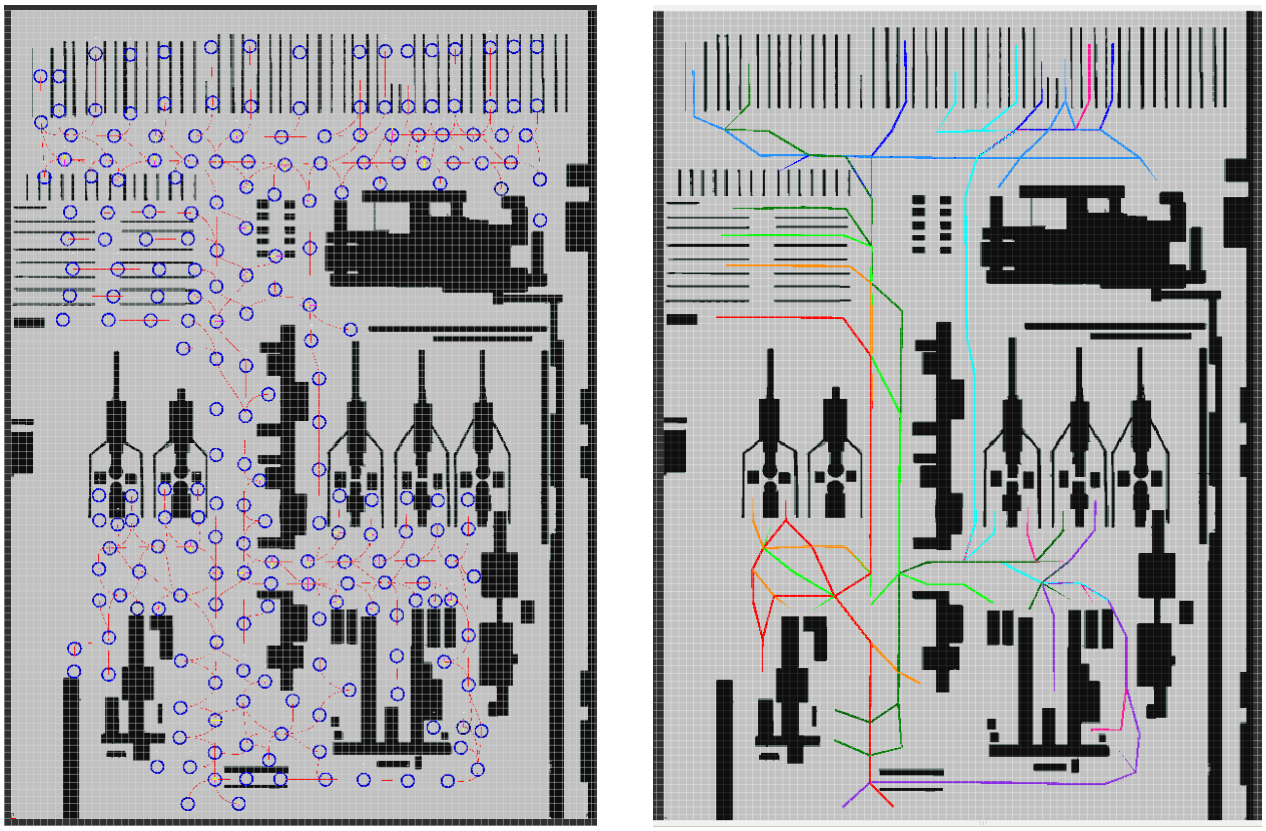
Figure 2 - Control diagram for each TEA algorithm iteration.*

Figure 2 depicts the system control diagram for the developed methodology. The first step in the process is the labelling of the current robot position as an obstacle. This step allows a robot to consider the position of other robots of the fleet as occupied cells.

The next step of the control loop consists of analysing the list of sub-missions, i.e., the list of robotic skills that are required to successfully satisfy a kitting order. Iteratively, for each sub-mission, a checking procedure occurs to assess if the robot has arrived in the destination location for the submission. This checking procedure is essential to guarantee that either the robot has completed its current sub-mission, or that the Time Enhanced A* is only performed when the robot needs to move in the logistic supermarket. Likely, there are usually sub-missions composing a mission that do not require the robot to move, for example, once the robot is stationary next to a container, the task of locating a component does not require the robotic platform to move.

After the checking procedure, if the robot needs to move, the Time Enhanced A* algorithm runs and outputs to the robot the path to reach the destination of the current sub-mission. At the same time, the cells composing the sub-mission's path are codified as occupied in time layers, as to avoid other robots of the fleet to interfere with the trajectory when the robot is expected to occupy a given position in space. Time is taken into account when codifying the path to occupied cells in the Time Enhanced A*, as a result, it is expected that robots can both use space that have been previously occupied on an ongoing sub-mission, or that will be occupied in the future. This novelty was built into the methodology to allow a better handling of potential deadlock situations.

Table 1 illustrates the missions' execution time for each robot with the Time Enhanced A* algorithm, using 10 robots and 30 sub-missions. For each robot, the time of advancement (T_{adv}) and the stopping time (T_{stop}) are reported. The developed methodology produces a collision-free path for each robot, and was compared with the results achieved by (Olmi, 2008), presented in Table 2. Figure 3 *(a)* presents a snapshot of the built layout while Figure 3 *(b)* presents the results of the routing algorithm for 10 AGVs.



(a) *(b)*

Figure 3 - Tested Environment: (a) Snapshot of the layout built in Rviz; (b) Results of the routing algorithm for 10 AGVs.

The average time ($T_{average}$) to complete the list of mission with the Time Enhanced A* is 116.7 seconds and 121.78 seconds in (Olmi, 2008). The last robot in the developed approach takes 135.8 seconds, while the compared mechanism finishes at 144.2 seconds.

Table 1 - TEA results for 10 robots.*

Robot	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
T_{adv} (s)	123.5	110.3	135.8	134.2	91.6	129.0	103.4	105.4	108.6	125.6
T_{stop} (s)	0	0	6.0	0	3.0	0	0	6.0	0	0
$T_{average}$ (s)	**116.7**									
T_{max} (s)	**135.8**									
T_{stop} (s)	**15**									

Table 2 - (Olmi, 2008) results for 10 robots.

Robot	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
T_{adv} (s)	144.2	118.6	115.2	127.0	73.4	151.0	123.0	107.0	121.6	136.8
T_{stop} (s)	26.8	0	1.6	0	17.0	0	0	1.4	0	0
$T_{average}$ (s)	**121.8**									
T_{max} (s)	**151**									
T_{stop} (s)	**46.8**									

3.4 Scheduling Module

In a multi-robot scenario, a routing system can be improved by using a search method for scheduling the order in which robots execute their missions. The robot execution order affects the calculation of routes, since the path positions over time, for a given robot, are obstacles for the remainder of the fleet. If the execution order changes, the paths and respective mission execution times for each robot are different. To cope with this, a Tabu search method was implemented to find the better robot configuration. A configuration is defined as the order in which robots should be processed by the Time Enhanced A* algorithm.

A Tabu search method is an iterative meta-heuristic that finds a better solution with a better evaluation of a set of neighbor configurations. This method is characterized by the use of dynamic memory in the Tabu list which records all moves, avoiding local optimality. For this, several parameters must be defined:

- The structure of the candidate possible solutions (neighborhood) in each iteration;
- The objective function;
- The stopping rules.

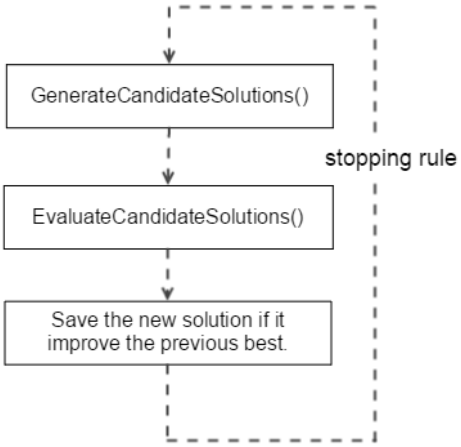


Figure 4 - Tabu Search Diagram.

Figure 4 represents the main blocks of the Tabu Search Method. Firstly, the candidate solutions list is generated according to the current robots' configuration. This list is then evaluated, calculating the cost function value for each neighbor solution. If a given solution is better than the previous best, it should be saved. The stopping condition in our approach is given by the number of iterations: more iterations represent a higher probability of achieving a better solution, with the downside of taking more time to compute.

In the developed method, the optimization goal is the minimization of three parameters:

1. **Time of the last vehicle**, denote as T_{max} in seconds;
2. **Average time of the missions' execution**, denoted as $T_{average}$, in seconds;
3. **Number of stoppages**, denoted as n_{stop}.

As such, the configuration that leads to a lowest cost function, minimizing the referred parameters, is chosen as the better solution. In the cost function, given by Equation (1), γ, ψ and τ are components that are used to personalize or take advantage of each parameter previously specified. In the developed approach, a solution is not required to be optimal, since a near-optimal configuration leads to an efficient Time Enhanced A* algorithm execution.

$$c = \gamma \times T_{max} + \psi \times T_{average} + \tau \times n_{stop} \quad (1)$$

In order to obtain the initial configuration of the Tabu search method, a heuristic function was defined. It consists in the path computation for each robot without considering the other robots as obstacles and ordering it by decreasing order of execution sub-mission times. The objective is then to process firstly the longer paths, minimizing the number of stoppages. The candidate solutions are generated changing two by two the robot execution order from the current solution.

Table 3 presents the results for the Time Enhanced A* algorithm using the robot configuration solution found by the Tabu search method. As a comparison, Table 1 presents the usage of Time Enhanced A* without the Tabu search method. The average time for completing all missions is lower, although the more significant improvement was the reduction in the waiting time. The total stoppage time without the search method was 15 seconds, while the total stoppage time with the Tabu search method was reduced to 6 seconds.

Table 3 - TEA result with the Tabu search configuration for 10 robots.*

Robot	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
T_{adv} (s)	125.4	110.3	122.6	133.7	91.3	129.0	103.4	105.4	108.6	124.2
T_{stop} (s)	0	0	0	0	3	0	0	3	0	0
$T_{average}$ (s)	**115.4**									
T_{max} (s)	**133.7**									
T_{stop} (s)	**6**									

3.5 Graph Editor and Splines Controller

The layout of the logistic supermarket was modelled on a graph structure as a set of vertices and edges. A graph editor tool was created using the ROS visualizer, RVIZ, and interactive markers, that allow the creation of the graph structure in an interactive way. Figure 5 is a screenshot of a portion of the roadmap used, which contains both Bézier curves and straight lines. Figure 3 *(a)* demonstrates the usage of the Graph Editor to create a graph structure in a large environment.

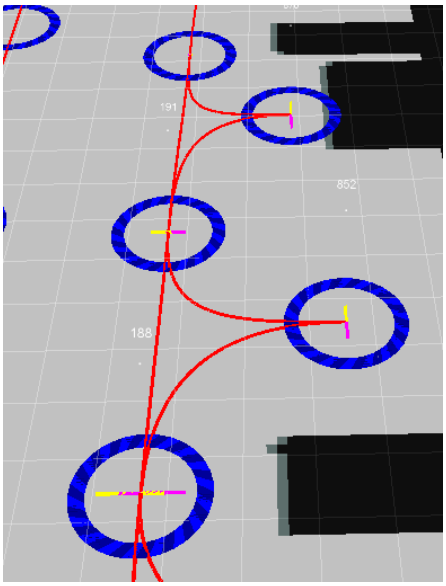


Figure 5 - Example of Cubic Bézier Curves.

A vertex is defined as a position in two-dimensional space, while an edge is the link between two vertices, represented by a cubic Bézier curve (Petrinec, 2005). The robotic fleet travels on these graph paths, from one vertex to another, through the predefined set of edges.

4 Dynamic Mapping

This chapter presents the developments conducted on the field of dynamic mapping. The first sub-chapter presents a definition on the concept of dynamic mapping, while the second presents some scientific principles associated with it. Finally, sub-chapter 3 presents the developed automatic inconsistency detection methodology.

4.1 Definition

With the advent of flexible robots, capable of sensing and interacting with large and complex environments, great challenges appear in terms of dynamically translating sensed information into structured information. Data captured by the sensors onboard of the robots can be used to continuously update three-dimensional structures that are a virtual representation of the environment, such as the logistic world model.

The logistic world model structure identifies all the physical objects of interest that are available in the logistic supermarket. All the parts and containers to be handled, directly or indirectly, by the robotic fleet are represented in the logistic world model and identified in terms of their geometrical dimensions and their internal structure (e.g. a rack is composed by several levels which might contain small boxes). The physical environment, mapped by the logistic world model, is in constant transformation, as parts are picked from boxes and placed inside kits. As boxes get empty, they are removed from containers and replaced by new ones. This dynamic is increased, as automotive parts evolve constantly in the life cycle of a vehicle, and parts can be added, removed or replaced, potentially also changing the configuration of the logistic supermarket, to optimize the access to the stored parts.

There is potential of interoperability of the logistic world model structure, as it is supposed to be an accurate virtual representation of the physical reality. As such, mechanisms that allow its continuous update using sensed data extracted from the robotic fleet are a necessity to cope with the dynamic and flexible nature of a logistic supermarket. The exchange of information between the robotic fleet and the world model structure is essential not only to coordinate their movement, but also to build a consistent and accurate global representation of the environment, by combining data from individual robots of the fleet.

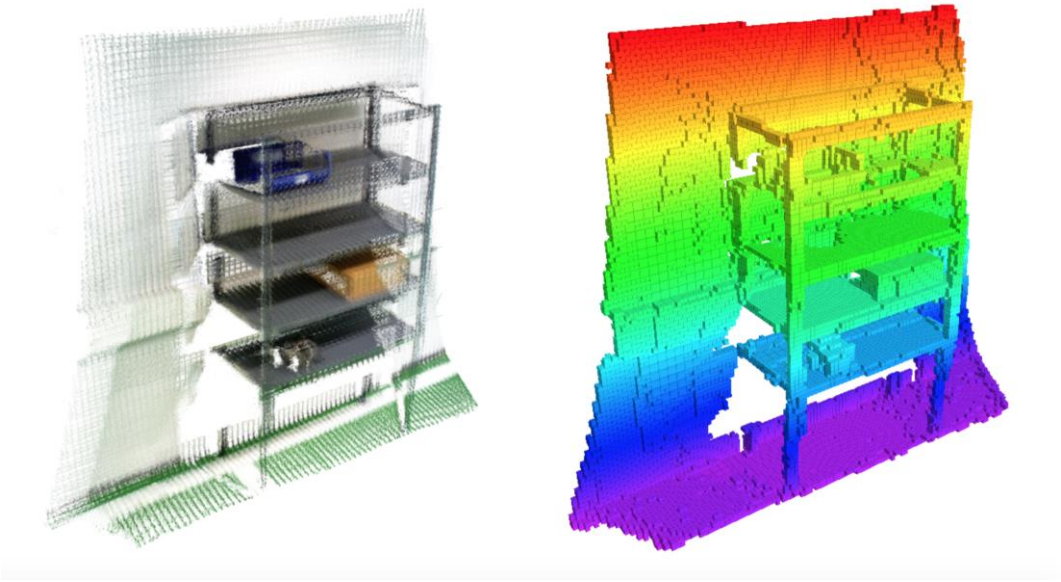
4.2 Dynamic Mapping Principles

As memory consumption is usually the main issue related with three-dimensional mapping system, it is essential that mapping is as efficient as possible (Hornung, 2013). Memory consumption increases with the amount of 3D point clouds measurements, without an upper bound, which is critical when mapping a large space, as a logistic supermarket.

Processing large 3D point cloud datasets is not memory efficient (Rusu, 2011). For this reason, spatial decomposition techniques, such as k-d trees and octrees are employed as access structures. While the first have a data-driven approach, enabling a faster data access by requiring less memory for the tree structure, the later, a tree-based hierarchical data structure for managing sparse 3D data based on the principle of recursive decomposition of space, supports a faster data update (Hornung, 2013).

The octomap framework is a tool for mapping three-dimensional environments based on octrees. Using the octomap framework, space is modelled by cells that can be explicitly defined as occupied, free or implicitly as unknown. As a result:

- **Occupied cells** correspond to those which contain range measurements from a sensor;
- **Free cells** are obtained from the ray cast from the sensors position to the measured point;
- **Unknown cells** represent unobserved volumes.



(a) *(b)*

Figure 6 - Hierarchical spatial representations: (a) Point Cloud; (b) Octree created using octomap.

Figure 6 *(a)* represents the three-dimensional point cloud information of a representative rack populated with a few small boxes and an alternator. Figure 6 *(b)* is the octree of the same rack. Although only the occupied cells are visible in the image, free cells (explicitly) and unknown cells (implicitly) are also modelled.

By combining, in a probabilistic manner, multiple measurements from a single sensor over time, the octomap framework is able to create and update a dynamic octree. The representation of a scene can be created regardless of the number of measurements collected from each scene. As a result, if the sensor travels at a high speed through a scene, very few measurements are collected, and this results in a scene representation which is less accurate because fewer data inputs where considered. Nonetheless, in these situations, the representations are still functional in the sense that it is still possible to use them for conducting dynamic mapping procedures.

4.3 Automatic Inconsistency Detection

The STAMINA robot is a mobile manipulator, used in a logistic supermarket for picking parts stored in racks and large boxes and placing them in a kitting box. To support this operation, the physical objects located in the supermarket are required to be represented in a virtual model (the logistic world model), which describes all their structural geometrical aspects and location in space.

By equipping the robotic fleet with sensing hardware such as depth cameras, each mobile manipulator has capabilities to, based on the sensed input, construct a virtual representation of the

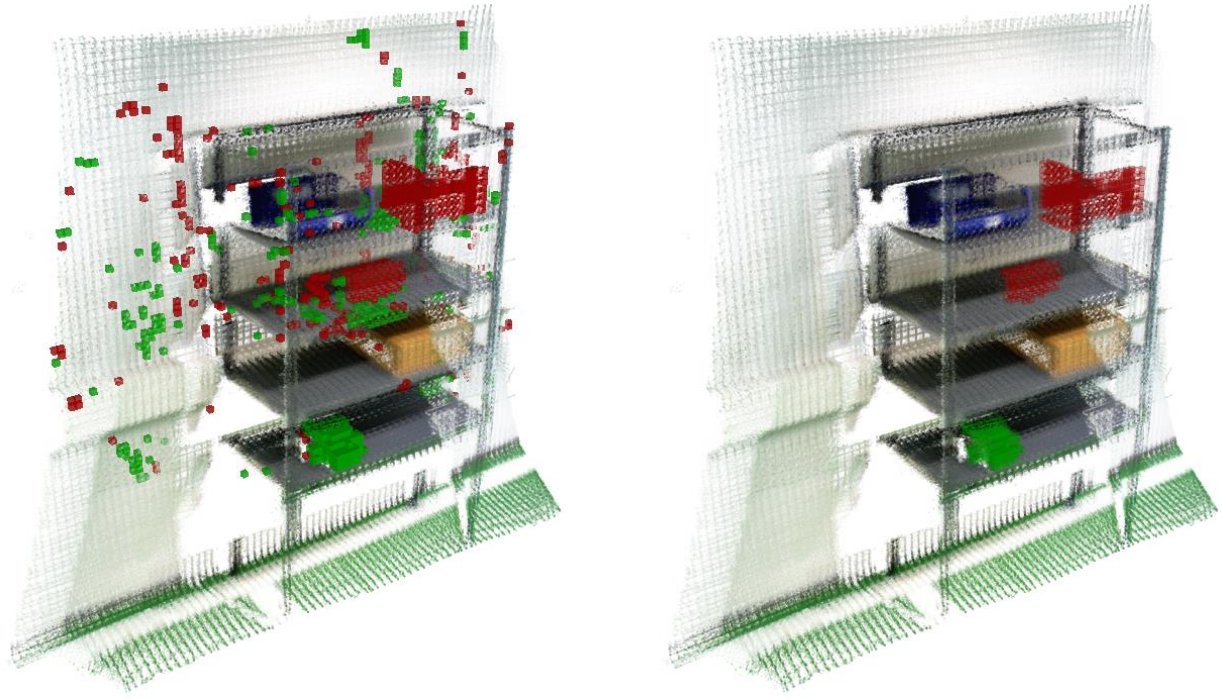
environment. Due to the dynamics of the logistic world model, besides the inputs from the technicians responsible for maintaining the logistic supermarket, an automatic inconsistency detection mechanism was developed to make a continuous and automatic assessment to the environment and report inconsistent or incoherent situations. Those potential inconsistencies are to be processes by the logistic planner and reported to the technician, who can then acknowledge the reported situation and choose to initiate a process to address it.

The developed mechanism is able to detect spatial changes between the virtual representation (the logistic world model) and the observed reality, i.e., the information sensed by the depth camera system. Spatial change detection is defined as the search for inconstancies between two scenarios. When comparing two scenarios, the model and the target, two types of inconsistency might occur:

- **Exceeding Inconsistencies**: If the target scenario contains an object which was not present in the model scenario, there is an inconsistency, labelled exceeding.
- **Missing Inconsistencies**: If an object which was present on the model scenario is absent from the target scenario, the inconsistency is said to be of the missing type.

In this application, two modes of operation were developed. In the first mode, **unsupervised spatial change detection**, the model and the target scene are octree representations of the scene at two different times, that could be created by the robot while traversing the logistic supermarket.

The model representation is used as a base for detecting spatial inconsistencies on the target octree representation. For each cell in the model octree, an occupation ratio is computed for all the target octree cells. Inconsistencies of type exceeding are detected when the model cell is defined as free and the occupation ratio is too high, i.e., above the occupation ratio threshold for type exceeding. Conversely, inconsistencies of type missing are identified when the model cell is occupied and the occupation ratio is too small.



(a) *(b)*

Figure 7 - Spatial change detection comparing an octrees of model and target scenes: (a) detected spatial inconsistencies of type missing (green) and exceeding (red); (b) spatial inconsistencies filtered and clustered.

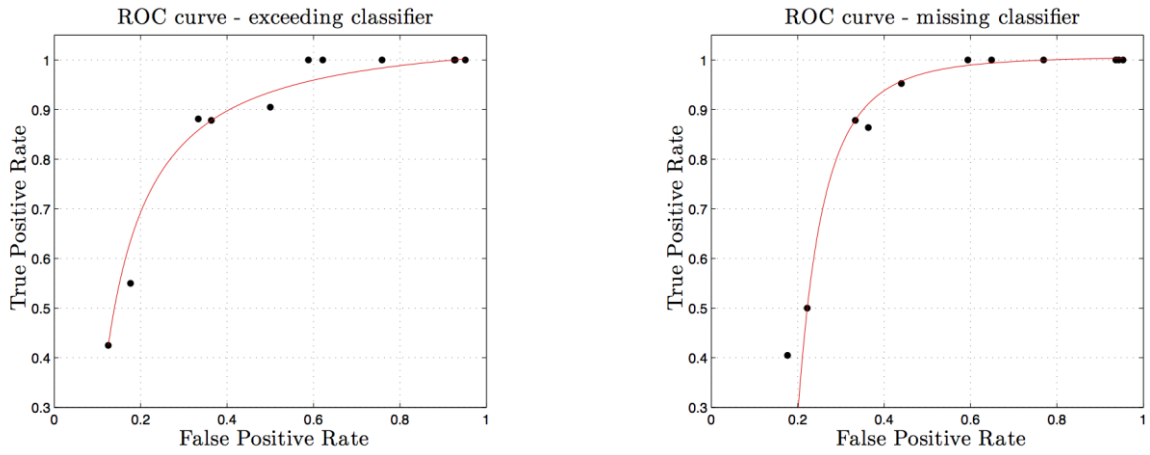
Figure 7 represents the comparison between model and target scenes, while Algorithm 1 shows the procedure designed to determine spatial inconsistencies of type exceeding and missing.

Input: $\boldsymbol{\alpha}$, the model octree
Input: $\boldsymbol{ß}$, the target octree
Input: $\boldsymbol{v}$, the search volume
Input: T_e, occupation ratio threshold for exceeding inconsistencies
Input: T_m, occupation ratio threshold for missing inconsistencies
Output: $\mathbf{I_e}$, list of α cells with inconsistencies of type exceeding
Output: $\mathbf{I_m}$, list of α cells with inconsistencies of type missing

1: $I_e \leftarrow \{\}$
2: $I_m \leftarrow \{\}$
3: **for** i = 1 ← length(α), ∀ α_i inside **v** **do**
4: Initialize the number of cells inside **v**, $N \leftarrow 0$
5: Initialize the number of occupied cells, $O \leftarrow 0$
6: **for** j = 1 ← length($\boldsymbol{ß}$), ∀ $\boldsymbol{ß}_j$ inside α_i **do**
7: N ← N+ 1
8: **if** occupied($\boldsymbol{ß}_j$) **then**
9: O ← O+1
10: **end if**
11: **end for**
12: Define the occupation ratio for cell α_i, r ← $\frac{O}{N}$
13: **if** occupied(α_i) **and** $r \leq T_m$ **then**
14: Mark α_i with type missing inconsistency, $\mathbf{I}_m \leftarrow \{ \mathbf{I}_m , \alpha_i \}$
15: **else if** occupied(α_i) and $r \geq T_e$ **then**
16: Mark α_i with type exceeding inconsistency, $\mathbf{I}_e \leftarrow \{ \mathbf{I}_e , \alpha_i \}$
17: **end if**
18: **end for**

Algorithm 1 - Octree to octree spatial change detection.

To evaluate the exceeding and missing inconsistency detection mechanisms, five scenes were collected with a 3D camera mounted on a robotic manipulator. The manipulator was used to scan the shelves by moving the camera to different viewpoints. The 3D data is then used to create an octree of the complete scene using multiple viewpoints as input. Each scene is composed by a shelf with objects in different positions. All possible combinations of model versus target were used, resulting in 20 model target scene pars. Ground truth was created by marking the objects that are exceeding or missing for any given model / target scene pair.



(a) *(b)*

Figure 8 - ROC curves for the unsupervised spatial inconsistencies detection: (a) exceeding type inconsistencies; (b) missing type inconsistencies.

Figure 8 shows the ROC curves for each type of inconsistency, analyzing the impact of the two user defined parameters: the occupation ratio threshold and the volume threshold. The presented ROC curves make an analysis of the trade-off between the successful detection of inconsistencies and the false alarm rate for the set of parameters chosen.

In the second mode of operation, **supervised spatial change detection**, the model scene is given by a list of cuboids, which are referred to as logistic volumes, that specify portions of the space that must be free or occupied, by definition. The list of logistic volumes and their respective dimensions and locations in space are directly retrieved from the logistic world model. A flag variable is used to state which nodes of the logistic world model are to be considered logistic volumes, i.e., which nodes must be checked for inconsistencies.

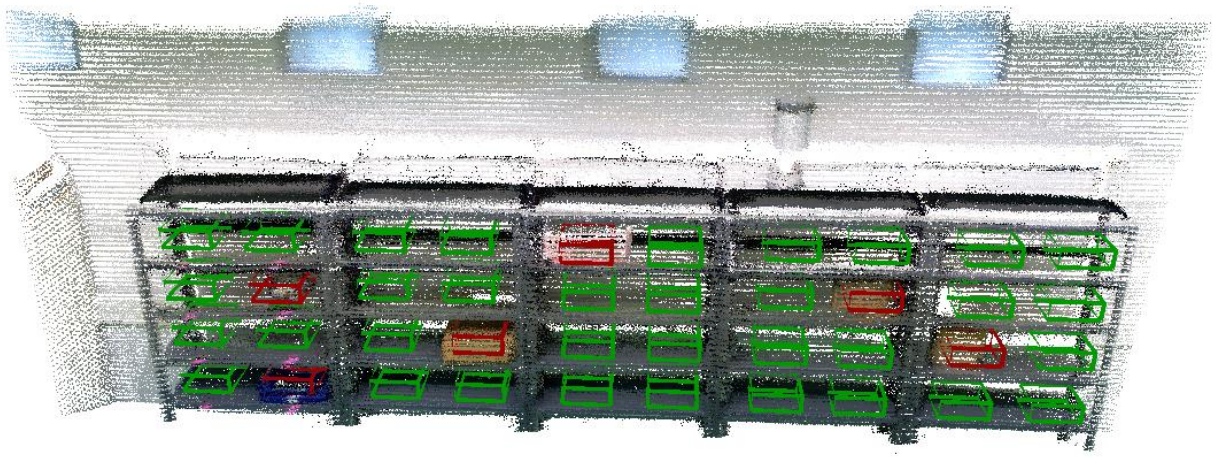


Figure 9 - Logistic volumes marked as cube wireframes (red: exceeding inconsistency; green: missing inconsistency).

Figure 9 represents the logistic volumes marked as wireframes. A red wireframe represents that the logistic volume is occupied, while a green wireframe represents that the logistic volume is free. The logistic world model is used to assess if a given logistic volume is free or occupied: if a rack cell has a children structure, i.e., if it has a small box associated with it, the logistic volume is flagged as occupied; if a rack cell has no children structure associated with it, and therefore no box, it is considered free.

During runtime operation, while the STAMINA robot is traversing the logistic supermarket, a target octree is constructed with information sensed from the environment, using the onboard camera system. This information is then processed by the spatial change detection mechanism and the resultant inconsistencies are reported back do the logistic planner. Algorithm 2 presents the method for detecting spatial changes from an octree and logistic volumes.

Input: **V**, list of logistic volumes
Input: ***ß***, the target octree
Input: T_e, occupation ratio threshold for exceeding inconsistencies
Input: T_m, occupation ratio threshold for missing inconsistencies
Output: $\mathbf{V_e}$, list of inconsistent logistic volumes of type exceeding
Output: $\mathbf{V_m}$, list of inconsistent logistic volumes of type missing

$V_e \leftarrow \{\}$
$V_m \leftarrow \{\}$
for i = 1 ← length(V) **do**
  Initialize the number of cells inside V_i, $N \leftarrow 0$
  Initialize the number of occupied cells, $O \leftarrow 0$
  for j = 1 ← length($ß_j$), ∀ $ß_j$ inside $\mathbf{V}_i$ **do**
    N ← N+ 1
    if occupied($ß_j$) **then**
      O ← O+1
    end if
    Define the occupation ratio for cell $ß_j$, r ← $\frac{O}{N}$
    if defined_as_occupied($\mathbf{V}_i$) **and** $r \leq T_m$ **then**
      Mark $ß_j$ with type missing inconsistency, $\mathbf{V}_m \leftarrow \{ \mathbf{V}_m , ß_j \}$
    else if defined_as_occupied($\mathbf{V}_i$) and $r \geq T_e$ **then**
      Mark $ß_j$ with type exceeding inconsistency, $\mathbf{V}_e \leftarrow \{ \mathbf{V}_e , ß_j \}$
    end if
  end for
end for

Algorithm 2 - Octree to logistic volume spatial change detection.

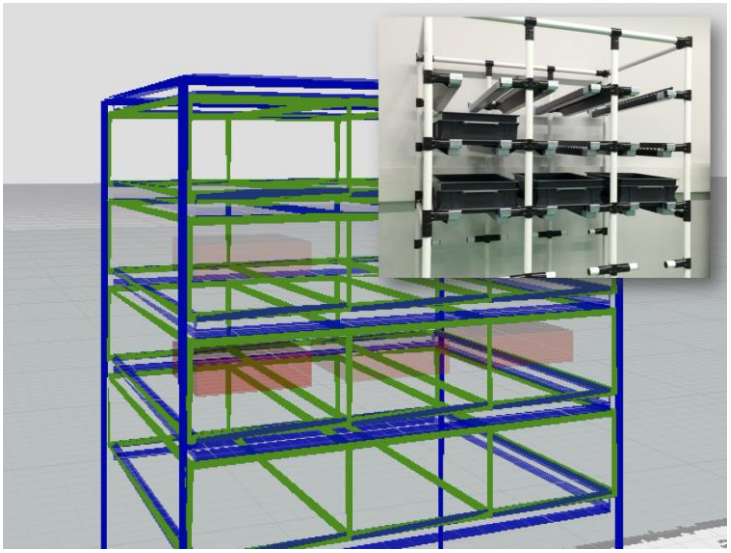


Figure 10 - Spatial change detection comparing an octree with the logistic volumes.

Figure 10 represents the result of the detection of inconsistent logistic volumes. The detection of inconsistent volumes is signalizing by filling the wireframes with the opposite color, i.e., if a volume is marked as free, and its occupation ratio is high, then the volume is filled with the red color, signalizing an exceeding inconsistency. Alternatively, if a logistic volume is flagged as occupied, and its occupation ratio is low, then the corresponding volume is painted with the green color, representing a missing inconsistency. In this example, all rack cells are flagged as free, but in fact there are four boxes located in the rack. As a result, four inconsistencies are detected, all of type exceeding.

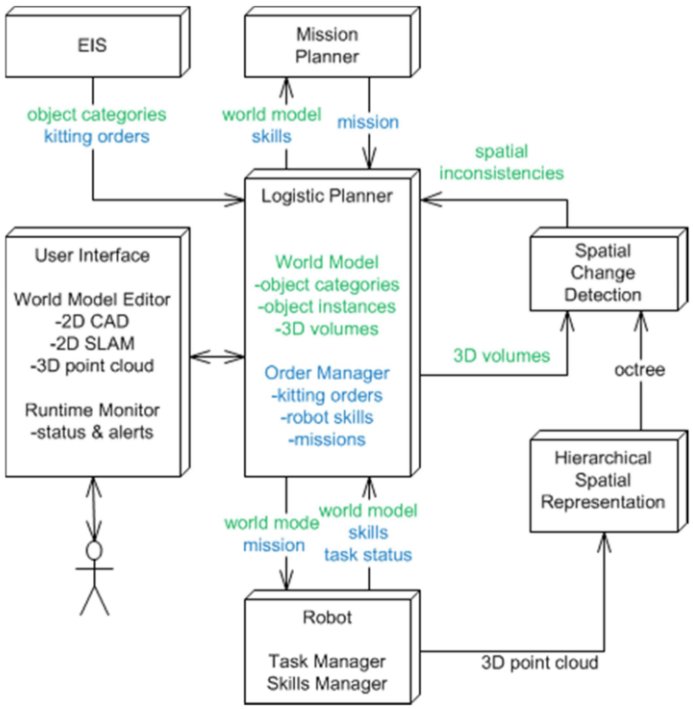


Figure 11 - STAMINA high level system architecture and major information flow.

The automatic inconsistency detection is organized as depicted in Figure 11. The modules developed for dynamic mapping run on the robot level: the camera hardware installed on the robot provide 3D point clouds to the hierarchical spatial representation module, which converts this information to the octree format, using the octomap framework; the spatial change detection module receives a copy of the logistic world model, which contains the logistic volumes, and the octree data from the hierarchical spatial representation component, and computes the spatial inconsistencies, which are transmitted as ROS messages to the logistic planner.

The continuous monitoring and assessment of the consistency of the logistic world model representation, when compared to the real time observations by the robotic fleet, contributes to the increase of the accuracy of the information stored, and to the detection of faults and errors in the virtual model.

Bibliography

(Le-Ahn 2015) Le-Ahn, T. (2005) *Intelligent Control of Vehicle-Based Internal Transport Systems*. ERIM Ph.D. Series Research in Management 51. Erasmus University Rotterdam.

(Sabuncuoglu 2000) Sabuncuoglu I.; Bayiz, M. (2000). *Analysis of reactive scheduling problems in a job shop environment*. EJOR, Amsterdam, v.126, p.567- 586, Nov.

(Santos 2015) Santos, Joana, et al. "Time enhanced A∗: Towards the development of a new approach for Multi-Robot Coordination." Industrial Technology (ICIT), 2015 IEEE International Conference on. IEEE, 2015.

(Hart, 1968) Hart, Peter E., Nils J. Nilsson, and Bertram Raphael. "A formal basis for the heuristic determination of minimum cost paths." IEEE transactions on Systems Science and Cybernetics 4.2 (1968): 100-107.

(Olmi, 2008) Olmi, Roberto, Cristian Secchi, and Cesare Fantuzzi. "Coordination of multiple AGVs in an industrial application." Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on. IEEE, 2008.

(Petrinec, 2005) K. Petrinec, Z. Kovai *The Application of Spline Functions and Be´zier Curves to AGV Path Planning*. Proceedings of the IEEE In- ternational Symposium on Industrial Electronics (ISIE), Dubrovnik, Croatia (2005)

(Hornung, 2013) A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, W. Burgard, Oc- tomap: An e cient probabilistic 3d mapping framework based on octrees, Autonomous Robots 34 (3) (2013) 189–206.

(Rusu, 2011) R. B. Rusu, S. Cousins, 3d is here: Point cloud library (pcl), in: Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE, 2011, pp. 1–4.

Abbreviations

TEA*	Time Enhanced A*
ROC	Receiver Operating Characteristic