M 2014

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

MULTI-ROBOT COOPERATION SYSTEMS FOR ASSEMBLY AUTOMOBILE INDUSTRY

JOANA SANTOS
DISSERTAÇÃO DE MESTRADO APRESENTADA
À FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO EM
ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES **2013/2014**

A Dissertação intitulada

"Multi-robot Cooperation Systems for Assembly Automobile Industry"

foi aprovada em provas realizadas em 24-07-2014

o júri

Presidente Professor Doutor Aníbal Castilho Coimbra de Matos
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor José Luís Sousa de Magalhães Lima
Professor Adjunto do Departamento de Engenharia Eletrotécnica da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Bragança

Professor Doutor António Paulo Gomes Mendes Moreira
Professor Associado do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Joana Santos
Autor - Joana Raquel Ramiro Santos

Faculdade de Engenharia da Universidade do Porto

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Multi-Robot Cooperation Systems for Assembly Automobile Industry

Joana Raquel Ramiro Santos

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Advisor: António Paulo Moreira

Co-Advisor: Germano Veiga

July 30, 2014

Resumo

Cada vez mais, surge uma necessidade crescente das empresas otimizarem o seu sistema de produção, e a automatização dos seus processos é muitas vezes uma das soluções encontradas.

A coordenação de múltiplos robôs móveis representa um desafio na comunidade Robótica, não existindo uma solução única, e podendo variar conforme a aplicação, nível de dificuldade exigido ou requisitos temporais. No entanto, é garantido que qualquer sistema multi-robô bem desenvolvido, permite obter maior eficiência e rapidez na concretização de um dado conjunto de tarefas, do que a utilização de apenas um robô. Nesse sentido, o desenvolvimento de um algoritmo que realize essa coordenação é exigido.

Esta dissertação pretende dar resposta a um problema de coordenação multi-robô, no âmbito do projeto STAMINA. Este projeto está inserido na Unidade de Robótica e Sistemas Inteligentes do INESC TEC Porto que é um centro de investigação dedicado à procura de soluções robóticas que contribuam quer para as necessidades da indústria, quer para o desenvolvimento de novas tecnologias. Para além do estudo e familiarização com as plataformas disponiveis para desenvolver e testar algoritmos de planeamento e coordenação multi-robô, este trabalho teve como objetivo desenvolver um algoritmo robusto capaz de coordenar uma frota de veiculos móveis autónomos.

Como plataforma de integração, foi usado o ROS(Robotic Operating System) devido às suas excelentes propriedades como ferramenta de desenvolvimento de código para sistemas robóticos. Além disso, devido à existência de uma comunidade ROS, é possibilitada a partilha e reutilização de métodos e funções.

A principal vantagem na sua utilização está relacionada com a sua capacidade para facilitar a integração de vários nós de processamento, no mesmo sistema. Assim, uma tarefa complexa pode ser facilmente decomposta em várias mais simples, de forma a que cada uma seja desenvolvida de forma independente num nó ROS, para mais tarde ser integrada no sistema completo. Nesta dissertação, o ROS foi usado para a integração do nó que contém o algoritmo de coordenação, com os nós de cada AGV e a respetiva comunicação com o simulador.

Inicialmente foram estudados os simuladores existentes no mercado, analisando as caracteristicas de cada um, e a sua adequação às necessidades deste projeto. Foram considerados 4 programas de simulação.

De seguida, foi realizado um estudo sobre alguns métodos de path planning e alguns trabalhos desenvolvidos na área da coordenação multi-robô. Tendo como base, o algoritmo de path planning A*, considerou-se interessante desenvolver uma extensão deste método, o Time enhanced A*. Uma das principais diferenças deste novo algoritmo é ter em conta, o tempo. Isto permite-lhe determinar o caminho ótimo para cada AGV, considerando os obstáculos do mapa, mas também o movimento dos outros AGVs.

De forma a uma melhor avaliação do algoritmo proposto, foi também implementado um dos métodos estudados na Revisão da Literatura, o Modified Banker's Algorithm. Este algoritmo apresenta uma abordagem distinta, quer na construção do mapa, quer na forma, como o controlo do caminho de cada AGV é avaliado.

Um dos factores considerados na comparação foi a capacidade de cada algoritmo evitar colisões e deadlocks. Estes são os problemas mais críticos que um método de coordenação deve ser capaz de resolver. Nesse sentido, são apresentadas as formas como cada algoritmo soluciona ou evita essas situações.

Finalmente são apresentados alguns resultados do Time enhanced A* para uma melhor compreensão e análise do seu comportamento perante algumas situações mais complexas.

Apesar de terem sido realizados alguns testes na simulação, existe ainda trabalho futuro, no sentido de tornar os algoritmos implementados mais robustos. Seria também interessante testá-los numa plataforma fisica de teste, de forma a analisar o seu comportamento, e assim poder melhorá-los.

Abstract

Increasingly, companies have the need to optimize their production system and the automation of the processes is, many times, one of the solutions.

The coordination of multiple robots, represents a challenge in the robotic community. A single solution does not exist, because it depends of the application, the level of difficulty and the time requirements. However any well developed multi-robot system, ensures greater efficiency and speed in the execution of a set of tasks, instead of the use of a single robot. In this sense, it's required an algorithm that executes the coordination of multiple automoted guided vehicles.

This dissertation proposes to solve a multi-robot coordination problem, under the project STAMINA. This project is inserted in the Robotics Unit and Intelligent Systems of the INESC TEC Porto. This is a research center dedicated to the development of the robotic solutions that contributes to the industry needs and to the development of new technologies.

In addition to the study of the available platforms to develop and test the coordination algorithm, this work had as objective to build a robust method to coordinate several AGVs.

ROS (Robotic Operating System) was used as an integration platform due to its excellent properties as tool for the code development to robotic systems. Furthermore, there is a ROS community that allows sharing and reusing several methods and functions.

The main advantage of its use it's the possibility of integrate many processing nodes, on the same system. Thus a complex task can be divided, in order to develop simple tasks in independent ROS nodes. Later, these tasks can be integrated on the complete system. On this dissertation, ROS was used to integrate the node which contains the coordination algorithm with the respective nodes of each AGV and finally communicates with the simulator.

Initially, considering the requirements of this project, were studied the available simulators on the market, and analyzed the features of each one . Four interesting simulation softwares are described.

A study about some path planning methods and some developed works in the area of multi-robot coordination were made. Based on the A* path planning algorithm, was considered interesting to develop an extension of this method, Time enhanced A* Algorithm. The main difference of this algorithm is that the initial map is a three-dimensional matrix that includes the time component. This additional feature allows it, to determine the optimal path for each AGV, considering the map obstacles, and the movements of the other AGVs.

To provide a better evaluation of the proposed algorithm, was also implemented a method studied on the Literature Review, the Modified Banker's Algorithm. This method presents a different approach, in terms of the map's construction and in the way how the path control is done.

The capacity to avoid collisions and deadlocks were some of the comparison criteria considered. These are the main critical problems that any coordination method should be able to solve. In this sense, are presented the methodologies used by each algorithm in order to avoids or solves these situations.

Finally, are shown some results of the Time enhanced A*, for a better understanding and analysis of its behaviour towards some conflict situations.

Despite being made some simulation tests, there are still some future work, in order to add more robustness to the implemented algorithms. It would also be interesting to test them in a physical test platform, evaluating its behaviour and improvement.

Acknowledgments

I would like to thank all the people who helped me throughout my dissertation. I am grateful to my supervisor Dr.António Moreira and Co-Supervisor Dr.Germano Veiga for the opportunity to work with them on this study area, and also for their provided support.

I must also acknowledge Dr.Pedro Costa for his suggestions during the development of my dissertation. His insight was an excellent contribution for the different stages of this work.

I would also like to thank the Electrical Engineering Department (DEEC) of FEUP by the people and resources provided during my last five years as student. It allowed me to develop my capabilities with the best professors and to learn good techniques.

A special ackowledgment goes to my parents and brother for their constant support and encouragement during my entire life. This thesis wouldn't be possible, without their contributions to my personal and academic success. I would also like to thanks my boyfriend for his affection and patience.

I would like to express my gratitude to the people that helped me on the review of the dissertation report.

Finally, but not least, I would like to thanks all my friends for their contribution to my leisure moments.

Joana Santos

“You have to learn the rules of the game.
And then you have to play better than anyone else.”

Albert Einstein

Contents

List of Figures

List of Tables

Abbreviations and Symbols

AGV	Automated Guided Vehicle
DV	Voronoi Diagram
RRT	Rapidly-exploring Random Trees
PRM	Probabilistic Roadmaps
CD	Coordination Diagram
ROS	Robotic Operating System

Chapter 1

Introduction

1.1 Objectives

This dissertation is motivated by the participation of INESCTEC Robotics and Intelligent System Unit on the European project, STAMINA. This project consists on a set of AGVs that execute tasks on the plant of PSA group (Peugeot-Citroen). The objective of this dissertation is to implement navigation algorithms in order to coordinates a fleet of automoted guided vehicles.

The following objectives were proposed for this dissertation:

1. Develop a realistic simulation platform of the system including:
    - factory environment, using plants and sensors data
    - robots
    - other dynamic elements of the factory.
2. Study multi-robot coordination algorithms.
3. Implementation of a coordination algorithm.
4. Test the algorithm using the simulation platform previously developed.

Finally, this dissertation is meant to solve a coordination problem, and develop a robust algorithm that satisfies the industrial requirements, avoiding collisions and deadlocks.

1.2 Motivation

Robotics has gained an important role during the years, not only for common people, but also for the industry. The use of multiple robots has many advantages to the production system and to cooperative tasks. It allows more flexibility, more efficiency and less dependency of a single robot. In this way, the coordination of multiple robots brings up a coordination problem which is an interesting challenge in Robotics and can be applied in many areas.

The study presented on this dissertation, will be tested in a practical situation, in the automotive industry. In order to improve their assembly line, a coordination algorithm was required. This work is a contribution not only to the robotic community concerned with multi-robot coordination, but it also has a personal and professional interest. The development of this work allowed me to get acquainted about many techniques, tools and good practices used in robotics. During this dissertation work, I learned about some powerful simulators and softwares used for programming robots.

1.3 Problem's Description

Nowadays, industry found in the automation, the solution to optimize their production system. However, its important that these solutions guarantees the same level of security than traditional/manual solution. The application of robotic system in an industrial environment has many requirements such as shared spaces, production's times, satisfying sequence of tasks, and the most important the security related to other automated systems and operators. This issue was also considered in this dissertation, since the final demonstrator of this project is an industry automotive.

It was developed a robust coordination algorithm, that gives more flexibility to each AGV, and avoids simultaneously the vehicles' collisions.

1.4 Methodology

To perform this dissertation will be used the following methodology:

- Search for the available simulators on the market that satisfy this project requirements. In order to justify the choice, 4 softwares were analyzed.
- Search for the available path planning and multi-robot coordination algorithms.
- Familiarization with the ROS environment. Implementation the A* algorithm a ROS node.
- Integration of the node A* on the V-Rep simulator.
- Development of the simulation environment. This environment should be as closest as the environment where AGVs will be tested.
- Development of the coordination algorithm, Time enhanced A*.
- Integrating algorithm on V-Rep.
- Testing and comparison with an algorithm studied in the literature review.
- Writing the final report

In order to develop a robust coordination method, it's important to know about works already developed in this area. Some available algorithms and their advantages or disadvantages were studied and its scenarios. Considering that the implemented algorithms need to be tested, some available simulators were also analyzed, considering their main features.

1.5 Dissertation Structure

In addition to the Introduction Chapter, this dissertation has five more chapters.

In chapter 2, it's described the State of the Art, including the main topics about path planning algorithms. Also some multi-robot coordination work will be analyzed.

In chapter 3, it's presented a comparison between some available simulators, and the main features of each one. A simulator was chosen to perform a realistic simulation, considering the main elements of the factory. In the development of this work were used some scenes, due to the tests that were made. So this chapter describes each scene and the differences between them.

In chapter 4 it described the general architecture implemented in this work. The communication between the algorithm and the V-Rep scene was made using ROS topics. So, this chapter reports the topics used and the respective dates that are changed between each node.

In chapter 5 is described the two algorithms that were implemented and compared. It was described each algorithm and the way how each one solves a collision and a deadlock problem.

In chapter 6 the two implemented algorithms are compared and reported the main results. In order to evaluate the behaviour of each method, some critical situations were tested.

Finally, in chapter 7 are presented the conclusions of the developed work as well as some future work that can be implemented.

Chapter 2

State of the Art

This chapter presents the literature review about main topics that will support all the work developed in this dissertation. In section 2.1 is presented a theorical study about trajectory planning algorithms, and multi-robot coordination. Regarding to the methods of path planning some available libraries were analysed in section 2.1.2.

2.1 Multi-Coordination Algorithms

In multi-robot systems, there are two main subjects, which are important to consider. The trajectory planning for each robot, and the coordination between robots. Both should be studied simultaneously, instead of separated problems. In this section, will be presented some existing algorithms for path planning and some methodologies to solve multi-robot coordination. For the coordination algorithm, it's required a first step, that consists in planning the optimal path, from a starting point to a final destination, see section 2.1.1.

2.1.1 Path Planning

The planning of the path is to describe the path between initial point to the final point, avoiding collisions with objects or other robots. This planning can be accomplished using classic methods (in mathematical or geometric terms) or probabilistic methods.

2.1.1.1 Probabilistic Methods

In larger dimension spaces, geometric methods are not feasible, whereas probabilistic methods can plan the path more efficiently. Will be presented the PRM(Probabilistic Roadmap) and RRT(Rapidly-Exploring Random Tree). These are the most cited methods on the bibliography.

The method **PRM** consists to select random samples of the space and check if it belongs or not to the free space. This method is divided in two phases, according to [4] and [5]:

- Learning Phase: the roadmap is constructed and stored in the graph. The nodes correspond to free collisions configurations and edges correspond to paths between nodes.

- Consultation Phase: all initial and final nodes, are connected.

The Learning Phase/Roadmap Construction, consists of the following steps:

1. Initialize the graph G(V, E). V is the set of nodes, randomly generated. Represents free configurations of the robot; E is the set of the free paths that connects two nodes.
2. It's randomly selected one configuration/node, q.
3. Check if the selected configuration belongs to the free space. If yes, it is added to the graph G.
4. Repeat the previous steps, for the N selected nodes.
5. To each node q, select k neighbors nodes.
6. Using the local method, connect nodes q and q'. q' is the neighbor of node q.
7. If connection exist and belongs to a free path, it is added to the graph G.

The Consultation Phase consists in the following steps:

1. Define the initial and final points on the graph.
2. Find the k neighbors of the initial and final points, in the graph G.
3. Construct the path, using a planning method, for example Dijkstra.

Another probabilistic method is the **RRT**. The objective is to map a path, between an initial point until a final point, by sampling points belonging to free space. According to [6], RRT has most properties of the *Probabilistic Roadmap*. The additional advantage is the possibility to be applied to the planning of nonholonomic systems, that considers the dynamic and cinematic restrictions of the robot. A nonholonomic system, is the system whose space dimension is greater than the possible system's degrees of control. The tree is generated incrementally, using randomly samples of the space. Two trees are used, one with the root on the initial point, and another tree with the root on the final point.

The method consists in, [4]:

1. Considering settings q_{init} and q_{goal}, initialize the tree with the root on T_{init} and T_{goal}, respectively.
2. Each tree grows according to the following steps:
    (a) q_{rand} is generated by the normal distribution. q_{rand} it's a random point of the space.
    (b) it calculates q_{near}, the nearest point of the q_{rand}.
    (c) add a branch in q_{near}, towards q_{rand}, with the predefined size for each branch. This step is executed only if there is no collision. Otherwise, it will be selected another point.

2.1.1.2 Classic Methods

In the classic methods, there are two strategies, according to [1]:

- Search Graph: the map is constructed with nodes and their respective connections. The path planning problem consists of a graph search problem.

- Potential Fields: it's described as a mathematical function associated to the free space. The gradient of this function is followed up to the final point.

According to [7], the method **Potential Fields** creates a gradient along the map, that will guide the robot to the destination point. The final point is represented as a set of attractive and repulsive forces (obstacles). The robot is treated as a point bound to a potential field, which is created by these forces. It is not necessary to build the map where robot will operate.

The **Search Graph**, as opposed to the last method, requires a earlier step: construction a discrete map. This map is a representation of the environment model. Differents path planning algorithms uses the discrete map in different forms. To **build the map**, and initialize the representation of free and ocuppied spaces, several methodologies are used. Here, will be presented the main approaches.

- **Roadmap:** consists of representing the physical space as a set of nodes and connections between these nodes. The nodes correspond to real localizations and the connections correspond to paths between these nodes. The roadmap can be constructed in different forms. In **Visibility Graph** all pairs of nodes that are visible to each other are connected by a segment. The nodes corresponds to the vertices of each obstacle. In this method, the number of nodes and segments increments with the number of the polygons of the obstacles. For this reason, this algorithm could be slow and inneficient, when compared with other tecniques, when considered an environment with many elements.
  Another method is **Voronoi Diagram** (DV). This method maximizes the distance between the robot and the obstacles of the map. The space is divided into regions and on each region only exists one obstacle. The diagram is a set of equidistant points, to two or more obstacles. The weakness of DV algorithm, is the limited range of the sensors localization. The objective of the diagram is to maximize the distance to the obstacles, therefore any short range sensor, is not capable of detecting its position.

- **Cell Decomposition:**This methodology is divided into exact cell decomposition and aproximate cell decomposition. In **decomposition into exact cells** each cell corresponds to a free or occupied space. No matter the robot's position in cell, it's only important the ability of robot to go from one free cell to another adjacent free cell. As in roadmap, with the environment complexity implies a sharp increase of cell number, and the algorithm become more computationally expensive. The **decomposition into aproximate cells** divides the map in free cells, occupied cells and partially occupied cells. This method is widely used

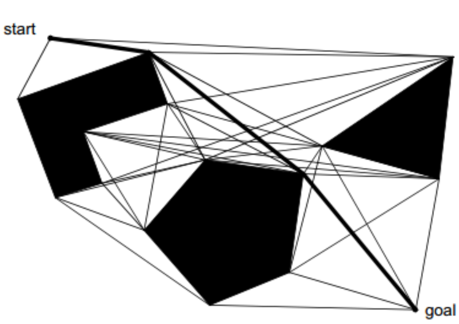


Figure 2.1: Visibility Graph ([1])

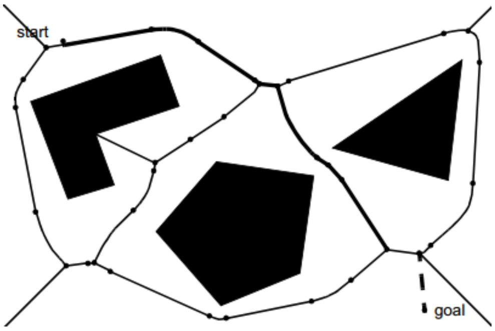


Figure 2.2: Voronoi Diagram ([1])

in robotics, because it leads to path planning algorithms with a low computional complexity. Depending of the environment, it is also possible to divide it into cells, where the size doesn't depend on the objects of the environment. In this case, if a cell is partially occupied it is divided into 4 rectangles recursively, until a minimum specified limit (generally K * size of robot).

The methods described are used to construct a graph map based in connections between nodes. Below will be presented some algorithms that are used to find a path between initial and final node. The more relevant algorithms are Search in Width, Search in Height, Dijkstra, A* and D*. All of them, begin the search on the initial node, and explore all of their neighbors nodes. For each analysed node, their unexplored neighbors are considered. During description of the path planning methods, will be considered: g(n) as the accumulated cost between initial node until node n, h(n) the expected cost between node n and final node (heuristic function). The total cost will be represented by f(n)= g(n)+h(n).

Search in Width, the nodes are always explored considering the distance to the initial node. All nodes connections, are analysed, before considering another node. The computation solution is fast, if the transition cost constant throughout the graph is considered. In this case, the search is optimal, returning the minimal cost path. However if the transition costs were not all equals, the optimal solution isn't guaranteed.

Search in Height, each node is explored to the deeper level of the graph. A disadvantage of this algorithm is the paths redundancy, because some nodes are explored more than one time. However, this method reduces the complexity of memory space. To each node, only the path between initial and final node, and their unexplored neighbors nodes are saved. After a node has been expanded, and your respective child nodes have been explored, this path can be deleted from the memory.

The **Dijkstra Algorithm** is similar to the Search in Width, but Dijkstra considers that cost of each iteration can take any positive value. The search of the nodes is similar to Search in Width, however the neighbors nodes are ordered in function of f(n), which is equal to g(n), because this method doesn't use any heuristic (h(n) = 0).

The **A* Algorithm** is the path planning algorithm most used in Robotics, due to the following properties:

- Optimum;
- Consistent: between linked nodes, the estimated cost to arrive to the destination point, is always less or equal than the cost to arrive to the neighbors node plus the cost from this node to the destination;
- Complete: has always solution when this exist.
- Admissible;

Will be presented the A* pseudocode in Algorithm 1. Before that, it's important to consider the definitions below:

- O: open list that has the non analysed nodes that can be selected.
- C: closed list which has the nodes already processed.
- Star(n): set of nodes neighbors to the node n.
- c(n1, n2): cost between node n1, and node n2.
- $\mathrm{n}_{\mathrm{init}}$: Initial node.
- $\mathrm{n}_{\mathrm{end}}$: Final node.

Algorithm 1 A*

```
$O \leftarrow n_{\mathrm{init}}$ {The open list is initialized with initial node.}
for $O \neq NULL$ do
    1. Select the node with the lowest value of function f(n), $\mathrm{n}_{\mathrm{best}}$.
    2. $C \leftarrow n_{\mathrm{best}}$ {Remove the best node of the open list}
    if $\mathrm{n}_{\mathrm{best}} == \mathrm{n}_{\mathrm{end}}$ then
        return Finish the algorithm
    end if
    for Star($\mathrm{n}_{\mathrm{best}}$) do
        if $(ni \notin O)$ and $(ni \notin C)$ then
            (If is neither in the open or closed lists)
            $O \leftarrow ni$
        else if $ni \in O$ then
            (If is in open list)
            Verify if this path, the cost is lower. If yes, then change the parent node x, to $\mathrm{n}_{\mathrm{best}}$
        else if $ni \in C$ then
            (If is in closed list)
            Verify if f(n) is best that when was processed. If yes, then change the parent node x to $\mathrm{n}_{\mathrm{best}}$
            $O \leftarrow ni$
        end if
    end for
    return Finish the algorithm
end for
```

This algorithm, uses a heuristic function to estimate the cost between a given node and a destination node. There are many heuristics that could be considered, so the heuristic function must be selected according to the problem. Generally in Robotics, the A* algorithm is used on map grid, and the most selected heuristic is the distance between a cell and the destination cell. This heuristic is named Euclidean distance, and gives the distance in straight line to the final point.

As seen before, the A* plans the path between two nodes, using the information of the map; moves the robot along the path, until it reaches the final point or until it finds a difference between the map and the real environment and if necessary constructs the map again, and recalculates the path. This solution may become inneficient, mainly in dynamic environments (environments that undergo many changes), and when exist little information about the map. Thus, arises D* Algorithm, as a generalization of the A*, and to be applied to this type of environments. The D* is an incremental algorithm, that holds the partial cost of each localization, refreshes the map, and reduces computational costs.

2.1.2 Path Planning Libraries

The evolution of robotics has led to the discover of many resources that allow sharing ideas between robotic community. Many times, our robotic problem has already been solved by someone and its solution is now available in an open source. This allows us to reuse code, and give more attention to other complex problems.

Although there are some path planning libraries that implements some of the algorithms viewed in the previous section.

Two softwares are presented below that allow integration of some path planning modules/libraries.

- **MoveIt**: According to [8], MoveIt is one of the most used softwares in Robotic, for robotic arm manipulation. It includes the most recent versions of manipulation, 3D perception, kinematics, control and navigation. Regarding to the path planning and the movement of the robot, this software gives an integration between some methods of path planning and some libraries, through a ROS service. In figure 2.3, is represented the high level architecture for the primary node move_group, that was provided by MoveIt. This node receives all components of the system (data of sensores, example) and gives a set of ROS services that can be used by the users.

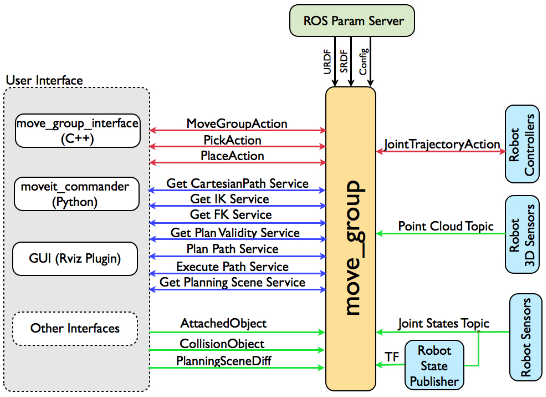


Figure 2.3: System Architecture for the primary node move_group ([2])

The interface to the several path planning algorithms is given by a ROS service, provided by move_group node. By definition, **MoveIt** uses the OMPL library (that will be presented below), to implement path planning algorithms.

- **KineoWorks**: is a software provided by Siemens that allows the trajectory's computation, avoiding collisions. This library can be used in any kinematic system, however is more suitable for robotic arms applications. This software controls the features for the safe control of movements: ([9]).
  - kinematic: the robot can be divided in a set of articulations, and some restrictions can be added according to the main features of the system.
  - Collision Detection: using fast methods that detect collisions throughout the trajectory.

Beside these softwares, it is also important to study the **OMPL** library. This library is provided by ROS, and has a set of path planning algorithms, without considering collision detection or visualization. For this project, is required an implementation of A* Algorithm and OMPL has two interesting variants that can be applied:

- **PRM***: Implements the PRM Algorithm (Probabilistic Roadmap), but uses the "star" strategy. This algorithm includes roadmap construction and path planning using probabilistic methods. According to the operation environment of the robot, this method computes automatically the number of neighbors to each node, considering the number of cells in which environment is subdivided. [10]

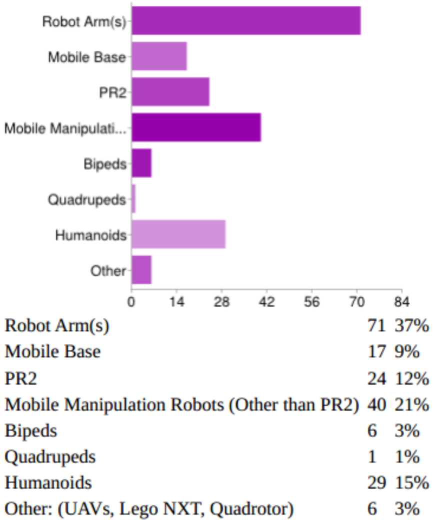
- **RRT***: Implements an incremental algorithm, RRT* (Rapidly-exploring Random Trees). This algorithm its an improvement of the RRT algorithm. A tree of feasible trajectories is built, considering that the root of the tree is the initial condition ([11]). In order to provide economic solutions in terms of memory requirements, RRT* removes the "redundant" edges, i.e edges doesn't belong to a shortest path from the root of the tree until to a vertex.

Both PRM* and RRT* Algorithm are based in sampling. In the two approximations, random sampling points are connected. The main difference between these algorithms is the way how graph joins these points. [12].

The first step in the PRM algorithm is to construct the roadmap that represents a set of free collisions paths. Returns the shortest path between the initial state to the final state. This method is probabilistically complete, such that the probability of failure, decreases exponentially to zero, with the increase of number of samples. In many applications, the environment has many elements or has elements that change frequently, so in these cases, the construction of the map it's a challenge. For these cases, appears the RRT, an incremental algorithm. In this type of algorithms isn't necessary to define initially the number of samples. When the algorithm consider that the set of paths constructed is enough, returns a solution.

Although these are interesting solutions, I want a library that implements the A* algorithm, because it leads with dynamic environments, systems with many restrictions, and can be implemented in many platforms. Furthermore, this algorithm guarantees an optimal solution.

According to the bibliography studied, it is possible to verify that these softwares are more appropriate to robotic arms applications. The figure 2.4, can be found on a report that considers the users's answers, to some questions about MoveIt, [2]. In this case, was asked what types of robots where the MoveIt used. As you can see, the bigger percentage corresponds to the robotic arms.



Robot Arm(s)	71	37%
Mobile Base	17	9%
PR2	24	12%
Mobile Manipulation Robots (Other than PR2)	40	21%
Bipeds	6	3%
Quadrupeds	1	1%
Humanoids	29	15%
Other: (UAVs, Lego NXT, Quadrotor)	6	3%

Figure 2.4: Survey of what kind of robots using MoveIt ([2])

Thus, was decided not to use any of these libraries, and build a ROS node to execute A* algorithm. This option allows to make some changes to traditional algorithm, such as collision detection or change the path according to the movement of the obstacle. Note that, the obstacle can be another robot.

2.1.3 Multi-Robot Methodologies

In the last years, more and more challenges have been proposed to the robotic community with the resolution of many complex problems. There are many situations where one robot operating alone, doesn't reach the performance required by industry. Thus, using multiple robots cooperating between them, allows better results, for many reasons:

- The distribution of tasks by multiple robots allows faster execution, and at the same time, the solution of minimum cost. Allows to subdivide a task in another tasks, and execute them, concurrently.
- The final solution is more robust, because there are redundancy of information, and so more assurance in the data obtained. An example consists in the map construction.
- Another advantage it's the robustness relative to faults. If a robot damages, the fault is communicated to the team or master, and the task is assigned to another vehicle.

There are many areas where multi-robot systems can be applied. Some of them, are [13]: management automatic systems, environments exploration, assembly lines, agriculture, intelligent environments...

Despite the advantages mentioned before, these systems have restrictions and performance requirements, such as, execution time of a set of tasks, security, communication and capacity of detecting inconsistent information.

So, the implementation of a multi-robot coordination algorithm, is a complex problem. Many authors, have studied this problem, and there are some examples of their implementation. It is possible to generalize some methodologies used in this kind of algorithms.

According to [13], there are 3 approaches:

- Centralized;
- Distributed;
- Economy-based;

A **Centralized approach** considers the multi-robot system as a system of only one robot with multiple degrees of freedom. It's selected a leader that will distribute the tasks by several robots. The main advantage of this methodology, is to provide a good solution in static environments (environments that don't change frequently) and applications with a reduced number of robots. In this conditions, this method has many probability to return optimal plannings. However this methodology also have some disadvantages. In case of occurring a fault on the leader, the operation of all system, is engaged. In this situation is necessary to choose another leader, carrying to time instants where the systems is inactive, without management system. Regarding to computational requirements, this methodology is exigent, because all important information, and data of all sensors are stored in the same processing central unit. In the figure 2.5 is presented an example of a structure of the centralized approach.

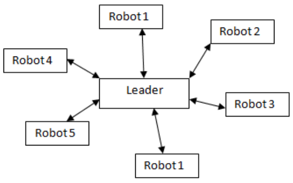


Figure 2.5: Centralized Architecture

The **Distributed approach** uses methods that allows to distribute the planning decisions for all team members. Each robot is independent, and its choices are based in the available information provided by sensors. The path of each vehicle is calculated with the local information. This method, don't have restrictions related to communication, because robots can only communicates with the nearest robots. Comparing with the centralized approach, has lower computational power, more robustness and less time to perform the tasks. In this case, no robot is dependent of a leader, so there is no single point of failure. In the figure 2.6 is presented an example of a structure of the distributed approach.

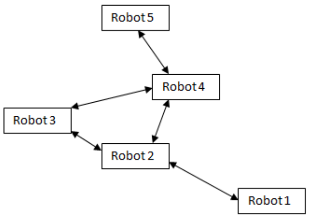


Figure 2.6: Distributed Architecture

The **Economy-based** approach is used to the control multi-agent systems. In a multi-agent system, each robot is a computational entity that makes its decisions based in the surround environment changes. According to [13] the terms multi-agent control and multi-robot control are distinct. In the first, the more frequent restrictions are related with resources, whereas in the second are considering communication restrictions. The control of each one is different, because robotic systems, must include intervals of errors by sensors faults and their interaction with the environment. It's also important to note that robotic systems are more susceptible of fault's occurrence.

Up to now were presented the main methodologies used to solve multi-robot problems. In the following sections are presented some solutions found in the bibliography. Although it is a complex problem, there are some articles with practical applications and different resolutions. All of them, considers that multi-robot coordination problem is divided in two phases: the tasks allocation and the coordination between robots. This dissertation only study the coordination problem, considering that the tasks have already been assigned. However in the next description of some algorithms, will be refered the way how the tasks are allocated, to understand better the algorithm that is been exposed.

The following papers will be presented:

1. **CoMutaR;** The objective of this work was the distribution of tasks by teams of mobile robots, ensuring the coordination between these teams, according to [14]. The description of this algorithm is presented in the sub-section 2.1.3.1.
2. **Trafcon;** This methodology considers a real AGV's system to be applied in automatic warehouse and the objective is to control the traffic using coordination diagrams and considering all of the real environment restrictions. This method is presented in the article [3]. In the sub-section 2.1.3.2 is described this methodology.
3. **Modified Banker's Algorithm;** This is a simple algorithm that solves the problem of deadlocks and collisions. The map used is a directional graph (which contains segments and nodes) and the coordination between the AGVs paths is based on the allocation of safe or unsafe states according to [15].This algorithm is described in sub-section 2.1.3.3.

2.1.3.1 CoMutaR

The methodology **CoMutaR(Coalition formation based on Multi-tasking robots)**, is used for the distribution of tasks by team's robots and their coordination. This approach is different of the others, because a robot can execute several tasks simultaneously, robot multi-task. Initially, it's defined a set of robots, R, and a set of tasks, T, where m is the number of robots and p the number of tasks:

$$R = r_1, r_2, ..., r_m \tag{2.1}$$

$$T = t^1, t^2, ..., t^p \tag{2.2}$$

Two problems are defined:

1. Task Allocation: find the function A:T $\mapsto$R, such that $A(t^k)$ is the team of robots that satisfy the task k.
2. Team Coordination: coordinate the actions, such that accomplish the task t^i.

Here will only be analysed the problem 2. It is assumed that the following data is known:

- A set of robots, R.
- A set of tasks, T.
- Task allocation, A.

Consider that $a_i,_j \longrightarrow$ is the action/task j assigned to robot i. It's possible to define a set of limited resources, $X^{i,k}$, relative to given resource. A restricted resource belongs to a robot (energy, position) or belongs to the environment (space's configuration):

$X^i = \{X^{i,1}, X^{i,2}, \ldots, \} \longrightarrow$ Resources of restricted participation, if i=0 belong to the environment, if i≥0, belong to robot (i=1, 2, ..., m).

For each resource $X^{i,k}$ is checked its availability, and is assigned a domain, $C^{i,k}$. For each action, $a_{i,j}$ is defined a function, such as:

$$\alpha_{i,j}^{l,k} : R \mapsto C^{l,k}$$

This restriction function is dependent of the time, and measures the amount of robot's shared resources, $X^{l,k}$, required by the action $a_{i,j}$.

The space $C^{l,k}$ has two operators that will define the restrictions. These restrictions affect the robot's decision. The operators are: composition operator $\oplus$ (the sum of restrictions due to a certain number of restriction function); and the comparation operator $\prec$, that takes the values $\{true, false\}$ (check if sum of restriction function exceeds the maximum capacity $\alpha_{max}^{l,k}$).

So, using this operators, it's possible to define,

$$\sum_{\mathbf{j=1}}^{\mathbf{n_i}} \varphi_{i,j}^{l,k} \cong \varphi_{i,1}^{l,k} \oplus \varphi_{i,2}^{l,k} \oplus \ldots \oplus \varphi_{i,n_i}^{l,k}$$

as the **restriction imposed by all actions** under execution in the robot R_l over the shared resource $X^{l,k}$. Likewise, must be added all robots' restrictions:

$$\sum_{\mathbf{i=1}}^{\mathbf{m}} \varphi_{i,j}^{l,k} \cong \varphi_{1,j}^{l,k} \oplus \varphi_{2,j}^{l,k} \oplus \ldots \oplus \varphi_{m,j}^{l,k}$$

This expression represents the **restriction imposed by all robots** over the shared resource $X^{l,k}$.

It's possible to conclude that the sum of all restriction function, by all robots, exceeds the maximum capacity of a certain shared resource $X^{l,k}$, by the folowing expression:

$$\sum_{i=1}^{m} \sum_{j=1}^{n_i} \varphi_{i,j}^{l,k} \prec \varphi_{max}^{l,k}$$

In this methodology, described in [14], the first step is define the domain φ_{max}, and the operators $\oplus$ and $\prec$. As an example, consider a bus communication, that have a limited bandwidth. In this case, φ_{max} corresponds to the maximum bandwidth. The domain is a set of a real numbers and the restriction function $\varphi_{i,j}$ corresponds to the bandwidth required by the action $a_{i,j}$. The operator $\oplus$ is the algebraic sum and the operator $\prec$ is $\leq$. The result would be given by the sum of the needs, in terms of bandwidth, for each action, and check if it's less or equal to the maximum limit.

According to [14], this method allows to integrate on the same framework the simultaneous execution of tasks, considering both the requirements necessary to allocate tasks and the restrictions imposed by them, during their execution. The main advantage of the CoMutaR is the capacity of create a robust fault system. Furthermore, the absence of a master planner makes it suitable to multi-robot applications although it has not been tested with a larger number of robots.

On the other hand, if some failure occurs in the architecture communication, the correctly execution of the coordination may be compromised. Finally, wasn't mentioned any path planning method, so depending of it, the execution of an optimal path, does not been guaranted.

2.1.3.2 TRAFCON

The algorithm proposed in [3] is based in CD(coordination diagram), that allows to map a coordination problem as a planning problem. The use of Coordination diagrams has some disadvantages, such as high computational cost. Therefore, they shouldn't be used in real AGV systems. Furthermore, this technique is complete, so the coordination is initiated before the robots begin their travel, so if a vehicle needs to change their trajectory, for example, due to an unexpected obstacle, the computation must be repeated from the begining.

The experiment **TRAFCON** presents an extended version of the CD proposed to multi-robot systems. It is based in a coordination strategy coupled to a dynamic routing algorithm. This means that, this traffic manager controls the coordination motion for all AGVs and simultaneously changes the robots' paths dynamically.

In the work presented in [3], this methodology was implemented on a real AGV system. Triangulation method is used for robot's navigation, with reflectors distributed by the walls of the factory.

The roadmap is divided into segments and each segment is occupied by a single vehicle. For safety reasons, the AGV's can not move backwards. So, the method has some limitations, depending on the application. The control architecture can be summarized in figure 2.7.

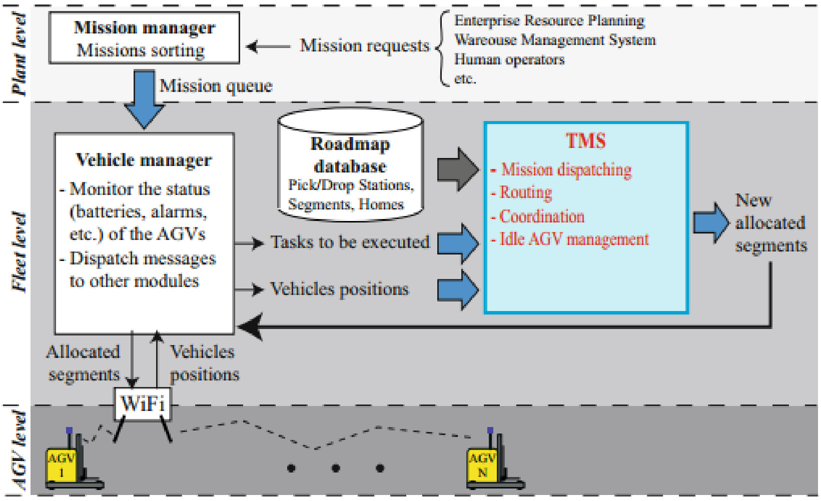


Figure 2.7: Control System for traffic manager proposed in [3].

The control architecture consists of the following systems:

- ERP System: Receives a set of tasks and returns a set assigned missions.
- The central unit is constituted by:
  - Vehicle Manager: Gives the AGVs position, and the segments that they occupy in the factory.
  - Traffic Manager System: Controls a set of AGVs, avoiding collisions. This system assignes to a given robot, during its path, using a management strategy.
- Low-level System: commands control of each AGV.

Consider the next parameters:

- $\mathfrak{R}$: Roadmap, comprising some segments τ.
- p_i: path defined as a sequence of adjacents segments, n_i.
- A_i: Vehicle i assigned to the path p_i.
- $A(x)$: Volume occupied by an AGV in the $\mathfrak{R}$.

Before the vehicle move from τ to τ', is necessary to ensure that collision is avoided. This is guarantee with the following expression:

$$A(\tau(\alpha)) \cap A(\tau'(\beta)) \neq 0 \tag{2.3}$$

$$(\alpha, \beta) \in [0, l_\tau] \times [0, l_{\tau'}] \tag{2.4}$$

A collision occurs, if this condition is true.

So, when a mission is assigned to an AGV, the path between two points is calculated (between $x_i^{init} e x_i^{goal}$). The CD is explored and the possible collisions are analysed, using the above expression. The next step, is to create a list for each vehicle, with a set of reserved and collidable segments. This list allows to manage the vehicles paths.

2.1.3.3 Modified Banker's Algorithm

In [15] is presented an improved version of the Banker's Algorithm. This methodology was tested on a plant with a real AGVs system, for packing and warehousing palettes. The innovation of this method compared with the Banker's Algorithm is the possibility to execute a path using some unsafe states, in order to a better usage of the vehicles. The avoidance of collisions is also guaranteed.

This method has a low computational complexity and can be applied to a plants in the real world. Similarly to other analysed methods, in this case, the map is also represented as a graph with segments and nodes. The vehicles can only travel by the defined segments.

This algorithm will be described, in detail, in the chapter 5.

2.2 Conclusions and Overview

The study presented here gives as overview of the main techniques used in robotics for path planning and multi-robot cooperation. All of the analysed algorithms have limitations and should be implemented in different applications (according to the map's dynamic, elements that interact with the AGVs and map's complexity). Another important point are the industry requirements. In many situations, robots cooperate with humans and other elements of the factory, and in these cases, the safety conditions should be ensured. So, the Modified Banker's Algorithm is a simple algorithm that could be applied to the industrial environment that ensures the client conditions and requirements.

Nowadays, most of multi-robot industrial systems, have a limited coordination algorithm, in terms of versatility. Meaning that, most applications consist in dividing the plant into segments and manage the permissions of a given robot to cross or not these segments. It would be interesting to create a multi-robot coordination algorithm that calculates the optimal path for each AGV, over the time, without segments. This optional solution, would allow more flexibility in the paths, collision avoidance and overtaking between robots. This new solution is considered in the chapter 5 and was compared with the Modified Banker's Algorithm.

In chapter 3 it is presented a comparative study of some available simulators on the market and the scenarios used in the simulation. Considering the requirements of the project, was chosen a simulator.

Chapter 3

Selected Simulator and Scenes

In this chapter some simulators available on the market will be studied, and the possibility of using them, in the development of the work. According to the main features of a generic factory plant, was selected a simulator that allows to develop a realistic simulation of the problem. In the sections 3.2 and 3.3 are presented the developed scenes and the main differences between them.

3.1 Simulators

3.1.1 Context

In robotic, the simulation plays an important role in situations where robots and humans share spaces, or when there is robots cooperation. On this dissertation, the simulation is very important, because allows us to test algorithms that will be used in various phases of the project, including coordination algorithms. The main objective of the simulation is to give us, a simulated environment with the relevant features of a generic factory plant. This simulation will be used to compare new algorithms, considering its robustness, efficiency and safety. In the next section will be presented the choice of the simulator. It will be compared 4 open-source robotic simulation software and the features of each one.

3.1.2 Comparison

Actually there are many simulators to help the robotic research. However, the features of this thesis allows me to select 4 simulation software (Gazebo, V-Rep, SimTwo, USAR-Sim). Each simulator was studied according to the requirements exposed below, [16].

Gazebo is a 3-D simulator that can be integrated in the ROS (Robotic Operating System). This simulator relies on two main libraries: OGRE(object-oriented graphics rendering engine) and ODE. This allows Gazebo to present 3-D dynamic environments with precision. It's possible to simulate a lot of environments combinations, because Gazebo offers the possibility to simulate robots with multiple shapes and joints. The simulation can be very realistic allowing to simulate all the objects with their physical characteristics (like mass, friction). However Gazebo doesn't allow changing the environment or objects during execution time.

Another interesting simulator is **USAR-Sim**, Unified System for Automation and Robot simulation. It also has a 3-D environment that supports Linux and Windows, like the Gazebo. This simulator is mainly used in situations where it is important to study human-robot interaction and multi-robot. USAR uses three main protocols for the control code which are Unreal Engine proprietary communication protocol, MOAST (Mobility Open Architecture and tools engine) and USAR-Sim Matlab Toolbox, [17] and [18]. However this simulator doesn't provide a clear tutorial with examples and details of implementation.

SimTwo is a versatile software that allows to simulate different types of robots and with different configurations. This simulator is mainly used for research and education. In this context, provides a simple interface to allow rapid test and environment design simulation, [19] and [7]. It also allows integration with ROS, but is necessary UDP connection, and configuration isn't simple. Another disadvantage is the lack of documentation.

VREP is a very interesting software with a set of features that makes it suitable for multi-robot application. It is based in a distributed control architecture which allows to break a complex system into several simple modules and to program each of them, independently. In terms of customization, V-Rep gives the possibility of controlling objects individually. Each object or model can be controlled in five different modes: an embedded script, a plugin, a ROS node, a remote API client or a custom solution. In order to provide a more realistic simulation, this software supports various degrees of precision, speed and other features which can be adjusted by the user. Another interesting feature is the possibility to configure customizable particles, which can be used to simulate water, air or jet engines. Finally, VREP offers the possibility to access the simulation from an external application or a remote hardware. Its properties are provided on a complete user-manual with main features, user interface, and comprehensive tutorials, [20].

As mentioned before, one of the objectives of this thesis is to test multi-coordination algorithms, and to develop a simulator that will allow the integration with other parts of the project (using STAMINA as a test-bed). **V-REP** satisfies all the requirements (see table 3.1).

Thus, this simulator was chosen to implement the environment simulation.

Below is presented the criteria for choosing the robotic simulator:

- Programming language: In this criteria will be considered all languages that are supported by the simulator. It must support a common language that can be integrated in many platforms.

- Multi-Thread Support: At this point is considered the capability to simulate more than one task, simultaneously. This is an important requirement for testing multi-coordination algorithm and others. This feature turns the simulation more efficient and robust.

- Sensors: It will be presented the main sensors included in the simulator.

- Collision Detection: This feature is present in almost all simulators. It should detect the possible collision of robots, walls and other objects.

- ROS integration: The communication between the program which includes the coordination algorithm and the simulator, is performed using ROS. Therefore, the simulation environment should allow ROS integration.

- Communication: In this criterion was considered what protocols are used to communicate between nodes.

- Format of importation files: This requirement allows to test compatibility with other platforms.

Considering [21], [22], [20], [19], [18], [23].

Programming Language	Multi-thread Support	Sensors	Collision Detection	ROS Integration	Communication	Importing Files
Gazebo						
C, C++, Java, Python	yes	Odometry, Range, Vision	yes	yes	TCP/IP	COLLADA
V-Rep						
C, C++, Python, Java, Lua, Matlab, Urbi	yes	Proximity, Vision, Odometry	yes	yes	TCP/UDP	OBJ, DXF, 3DS, STL, COLLADA, URDF
USAR yes						
C, C++, Java	yes, but use Java Virtual machine(JVM)	Sound, Vision, Odometry, Touch, Range	yes	yes	TCP/IP	FBX
SimTwo						
Pascal	No	Vision, Laser, Range, Luminosity	yes	yes, but isn't direct	UDP or Serial Port	3DS

Table 3.1: Comparative Table of simulators

In order to evaluate the main features of the V-Rep, and knowing the simulator, some tests using Lua, were performed. These tests consisted on some control of the robot's navigation. A simple algorithm of point navigation was developed. In addition to the presented features, this simulator is user-friendly, because presents an intuitive interface that facilitates the use of it. It's also important refer that the available tutorials provide the main acquirements for the development of the project.

In the follow sections will be presented the scenes used on this dissertation to perform some tests.

3.2 Initial Scene based on the PSA plant

The PSA's plant was used as test scenario in order to validate the work performed on this dissertation, under the STAMINA project.

The developed scene in the V-Rep includes the main corridors and walls of the plant. To perform the test of the multi-robot coordination algorithm, it is only required a map that includes the free and the occupied space.

This scene was used mainly to test the navigation of a single robot, and to test the ROS communication with the simulator. This scene is presented in the figure 3.1.

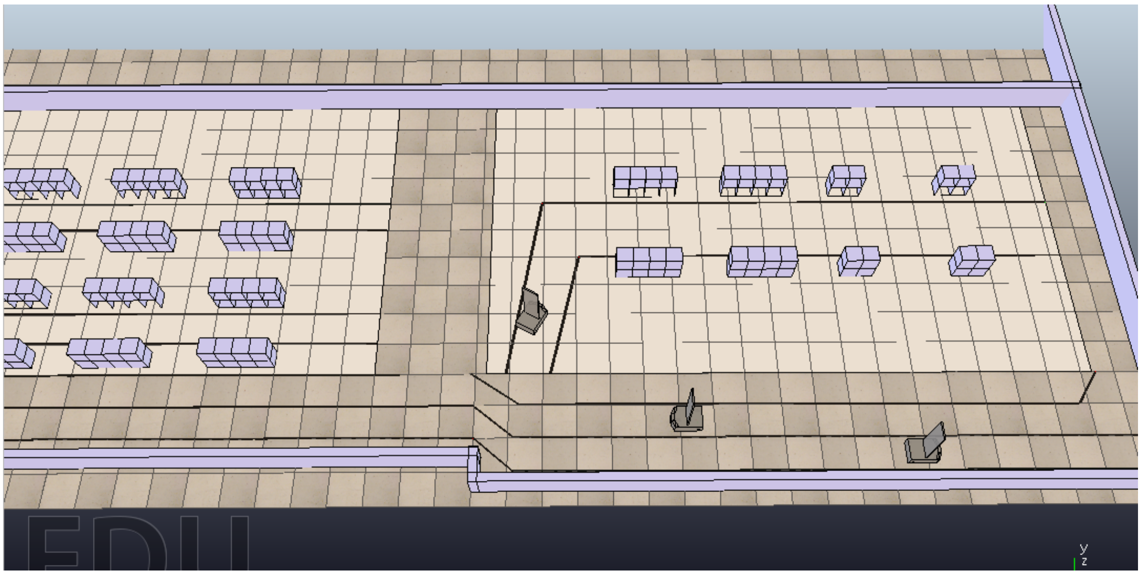


Figure 3.1: Initial V-Rep Scene

3.3 Simulation Scene

The scenario presented in the section 3.2 has few alternative paths, which doesn't allow to test some critical situations. According to this, was necessary to create a V-Rep scene with more corridors, in order to create more collisions' situations. The objective of this dissertation is to study the behaviour of the implemented algorithms, according to the collisions' avoidance, paths efficiency and execution times. Thus, it's important to develop a scenario that allows these situations.

It was decided to support the new simulation scene, in the PSA's plant. Some changes related to the initial scene, were performed. This developed V-Rep scene is represented in the figure 3.2.

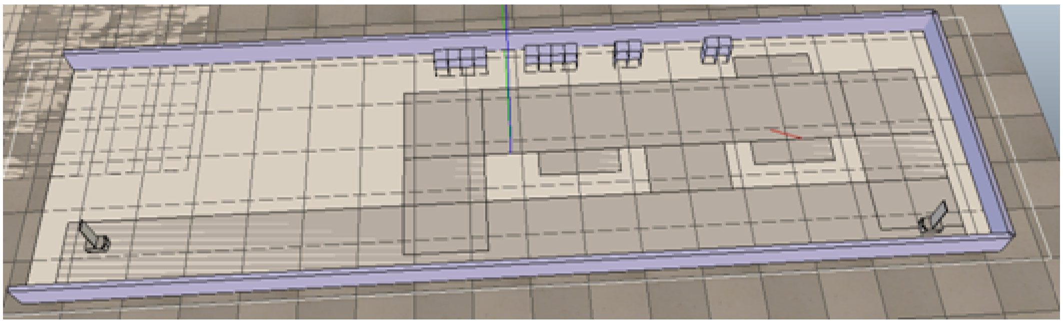


Figure 3.2: Simulation Scene developed in the V-Rep

The map used in the simulation scene is presented in the figure 3.3, including the coordinates of each corridor.

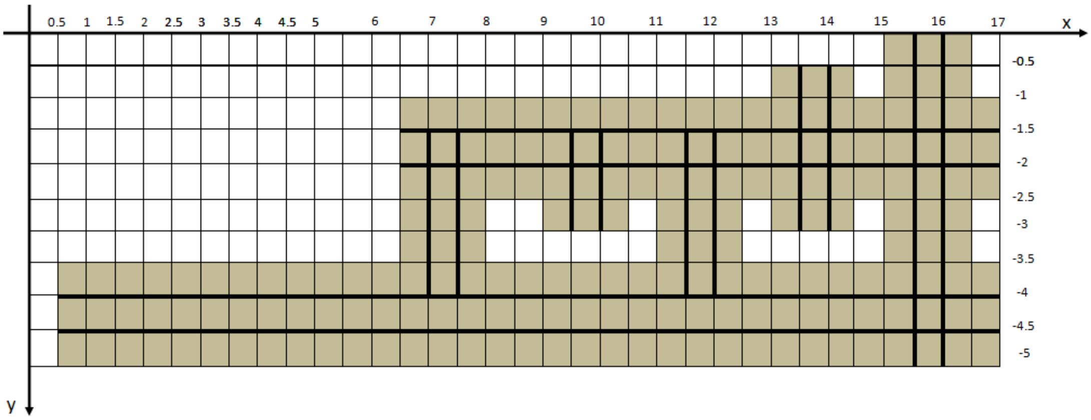


Figure 3.3: Map of the simulation scene

One of the implemented modification is related to the width of each corridor. Each corridor was extended allowing the simultaneous navigation of two AGVs. In this way, the coordination algorithm has the possibility to choose an alternative path.

Also with this new scenario, it's possible to test overtaking situations, providing robust solutions. Finally, some corridors were added, making more connections between them.

Any coordination algorithm is evaluated by a set of parameters, one of them is their robustness. This means that, considering a set of missions, the algorithm must be capable to execute them, in the shortest possible time and avoiding collisions. If many restricted paths are imposed, the coordination algorithm does not use all of its capabilities. In this dissertation one of the objectives is to test the behaviour of the implemented algorithm using different situations. To perform a better evaluation of their results, it's important to have a scene that allows it.

3.3.1 Map Conversion

As can be seen in the image 3.3 the dimensions of the map are $(17 \times 5)cm$ and each cell have $0,5cm$.

The input of the implemented algorithms is a binary matrix with dimensions (32×10) that represents the map used on the simulation. Each cell of the matrix takes the values 0 or 1, indicating respectively if the cell is free or occupied.

The coordination algorithm determines on each iteration the coordinates of the point to which the vehicle should go. These coordinates correspond to the cell's positions in the matrix. Before sending the coordinates of the point to the simulation environment, a simple conversion needs to be done.

The coordinates of the point in the matrix (result of the algorithm) are twice of the corresponding coordinates in the simulation map. Thus, the point coordinates should be divided by two before being sent to the simulation.

3.4 Conclusions and Overview

In this chapter is presented an overview and comparison of some open-source simulation softwares, availables in the market. V-Rep seems to be a simulator with enormous capabilities that fulfills the requirements of the project. The following criteria were considered in the comparison: programming language, ROS integration and sensors.

The two presented scenes were used in different situations. The first scene was used during the first tests, in order to simulate the navigation of a single robot and to test ROS communication with the simulator. The second scene was used to test the multi-robot coordination algorithms, in order to provide some challenges in terms of navigation with multiple AGVs.

In chapter 4 it is described the general architecture of the system, considering the integration platform ROS.

Chapter 4

General Architecture

In this chapter it is described the general architecture of the final system. The multi-robot coordination algorithm was implemented using C++ language. The communication between simulation (in V-Rep) and the algorithm was performed using ROS. This tool is open-source and allows to integrate many platforms on the same system, from low-level tasks to navigation and coordination control. Furthermore, in real applications, ROS has an important role, as an integration framework. Although the work presented in this thesis, has only been tested in the simulation environment, the future work will be tested in a real system, with many robots. In this sense, the coordination algorithm developed on this dissertation, that has already been integrated with ROS, can be a simple node, that communicates with existing systems. Thus, the future integration with real AGVs system is more easier.

4.1 Brief Introduction to ROS (Robotic Operating System)

It was used the open-source platform ROS, for the integration of the different modules of the project (code of each robot, the coordination algorithm and simulation) . This is a framework dedicated to the development of robot code, and that provides a set of tools and libraries which simplifies the team work and the construction of a complex and robust robot, [24]. With this tool it's possible to distribute tasks between the team elements and in the final do the integration, in an easy way and use an architecture of cooperative development. The processing is performed in the nodes, that receive or/and send data.
So this dissertation was advantageous to use ROS, not only for the integration with the V-Rep simulator, but also, in order to facilitate the future integration in a real system.

4.2 Multi-Robot Coordination Algorithm

The methodology used to implement multi-robot coordination algorithm, was the centralized approach. As described in 2.1.3, this approach is based on a master that communicates with other systems, and provide them some tasks. In this case, was created a node with the coordination algorithm, that communicates with the nodes of each AGV. The nodes are executed concurrently, and data is published and subscribed with a defined frequency. The defined frequency and the respectively temporal analysis, are studied in the chapter 6.

In the figure 4.1, is shown the general architecture, comprising the node of the main program, and the node of each AGV.

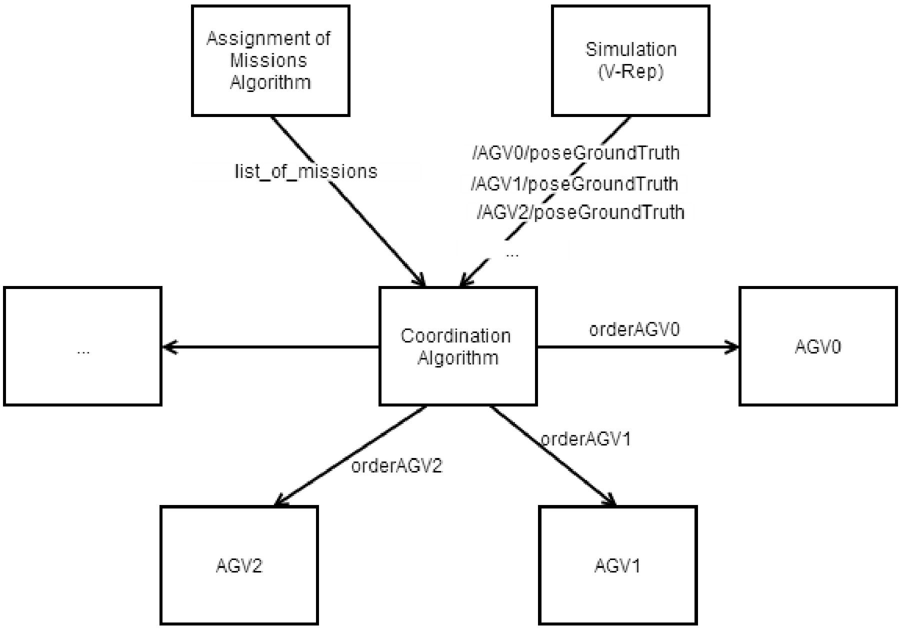


Figure 4.1: General Architecture of the multi-robot coordination system

There is a main program that:

- Receives the missions assigned to a given robot;
- Receives the positions of each robot;
- Runs the algorithm;
- Sends orders to each AGV's node;

The result of the *coordination algorithm*, in each iteration, is the path (set of points) of each AGV. It sends to the *AGV node*, the coordinates of a point to which AGV should navigate.

So, this node receives order to go, runs a point navigation algorithm and sends the commands to the simulator. The point navigation algorithm follows the sequence: (1) reads the actual position of the vehicle;(2)calculates the distance and the orientation until the final point;(3) Sends the velocity commands to the simulator and finally stops when the robot arrives to the destination point. The considered frequency to the coordination node was 20 Hz.

The figure 4.2 shows the main task of each node, and the relationship between them.

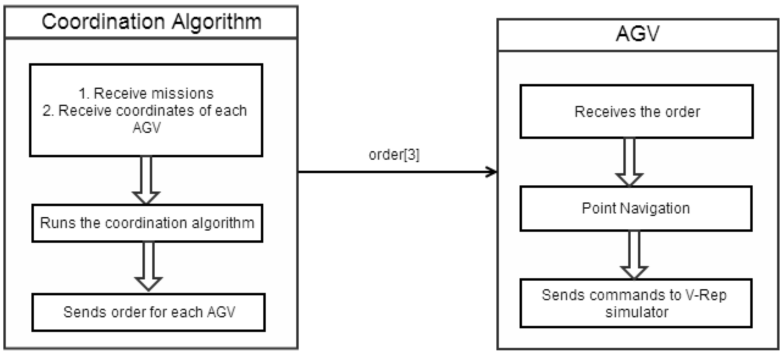


Figure 4.2: Structure of each node

4.3 Framework ROS

The *coordination algorithm* sends and receives data from the AGV using ROS topics. Topics are buses that allows exchange messages between the nodes, [8]. If a node is interested in some data, can subscribe this topic. The nodes that produce information, publish this data in a relevant topic. In this case, the topics that each node publishes and subscribes are described below. The figures 4.3 and 4.4 shows the topics changed between the nodes.

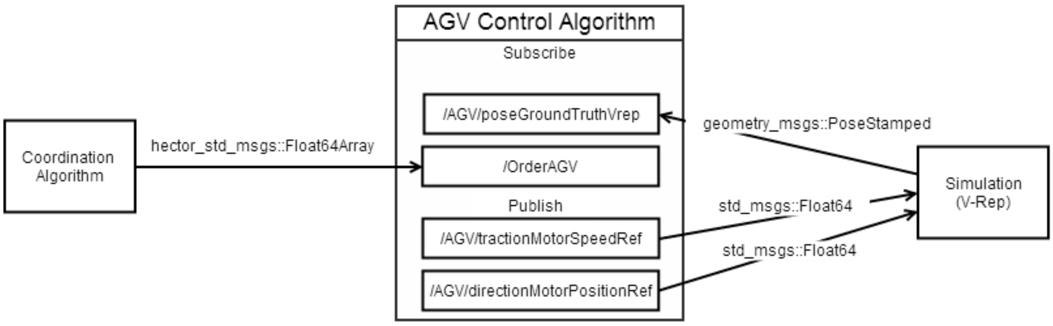


Figure 4.3: Topics subscribed and published by the AGV node

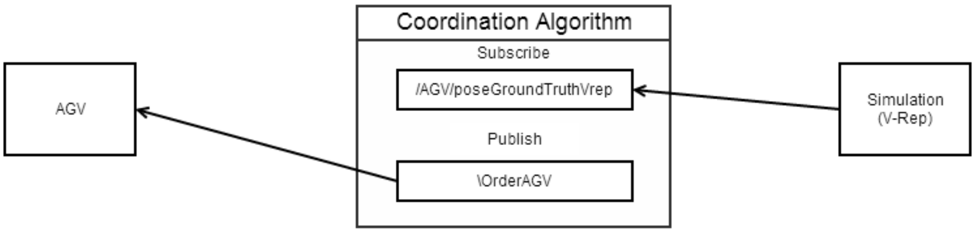


Figure 4.4: Topics subscripted and published by the node of the coordination algorithm

Description of each ROS topic:

- /AGV/poseGroundTruthVrep: This topic is subscribed by the node runnig the *AGV Control Algorithm*. Gives the coordinates (x, y) and the AGV's orientation.
  Type of data: geometry_msgs::poseStamped.
- /AGV/tractionMotorSpeedRef: This topic is published by the node runnig the *AGV Control Algorithm*. Sends to the simulator, the angular velocity of a given vehicle. The angular velocity is given by the linear velocity divided by the wheel radius.
  Type of data: std_msgs::Float64
- /AGV/directionMotorPositionRef: This topic is also published by the node runnig the *AGV Control Algorithm*, and sends to the simulator the orientation value that a vehicle should take.
  Type of data: std_msgs::Float64
- /OrderAGV: This topic is published by the *Coordination algorithm node*. It's an array of size three, and gives the coordinates (x, y) and a flag that controls the binary moving state.
  Type of data: hector_std_msgs::Float64Array

4.4 Conclusions and Overview

Considering the algorithms implemented in this thesis, this architecture was a good solution and fulfill the objectives. The processing time, as you can see in chapter 6 is relatively fast.

However there is some future work in this area, considering the powerful ROS capabilities. It would be interesting to improve the way how nodes of each AGV are created, mainly for applications with many robots. The code to each AGV is very similar, so ROS allows to create similar entities with a different identifier, instead of a repeating code.

In the next chapter will be described the two algorithms implemented in this dissertation.

Chapter 5

Multi-Robot Coordination algorithms

In this chapter the two tested algorithms are presented, along with their main features, advantages and behaviour in different scenarios.

In section 5.1, will be justified the chosen algorithms (Time enhanced A* and Modified Banker's Algorithm). The objective was to implement an algorithm that allows flexibility of the paths and in the same time, collision avoidance. It was decided to compare it with an algorithm proposed in the literature review (Modified Banker's Algorithm). Both have advantages and can be applied in different situations.

The time enhanced A* is explained in the section 5.2. It is described the algorithm and the main differences relative to the traditional A*.

The section 5.3 includes a brief description of the Modified Banker's Algorithm and some of the improvements relative to the Banker's Algorithm.

Considering that collisions detection and deadlocks are two fundamental challenges of any coordination algorithm, it is also presented here, the way how each algorithm solves this situation.

5.1 Selected Algorithms

As was seen in the chapter 2.1.3.3, the Modified Banker's Algorithm seems a good solution to be applied in industry environment, because it fulfills the client's requirements, in terms of security and efficiency. However, this algorithm has some limitations in terms of path flexibility. This method calculates the path assigned to each mission (and vehicle) in the beginning of the assignment. This means that, while the path is being executed, robots don't have the possibility to change it. The map is translated into segments and nodes. When a path is calculated, the segments and nodes that a given robot needs, are reserved. During the paths execution, the segments and nodes are progressively released and are made available to be reserved by other vehicles.

The *Time enhanced A** that is an extended version of the traditional A* was proposed. This new algorithm calculates to the over the time, the optimal path, considering the changes of the environment and the movements of the other vehicles. To account for those changes, the path is constantly being re-determined, making it an online method. For each instant of time, the map is analysed and the trajectory can be changed if found some obstacle.

Both methods were implemented in order to be compared.

5.2 Time enhanced A*

5.2.1 Description

As shown in section 2.1.1.2, A* is a path planning algorithm that returns the optimal path between an initial point to the destination point, using information of the map. This means that, if a robot, during the path's execution, finds a difference between the map and real environment, the path is recalculated. This method could have good results in static environments, but in dynamic environments it's inefficient. The traditional A* applied to multi-robot shares the same concept, but considers that the other vehicles are obstacles.

A new, Time enhanced, A* is proposed. The method consists of the path calculation, for each robot, to consider the positions and paths of other robots as moving obstacles. The algorithm is used on a map grid and the map is copied for all time instants, as shown in the image 5.1.

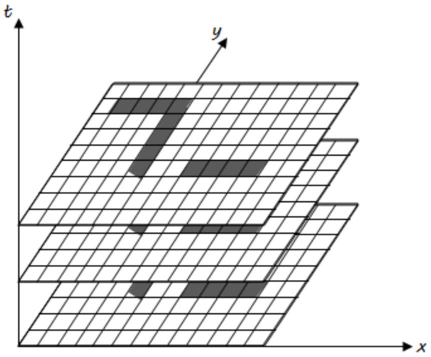


Figure 5.1: Example of a map grid, over the time. This is the input of the *Time enhanced A** algorithm

This input map is used to calculate the path. The main difference between this new strategy and the A* is that the map has three-dimensions, so each analyzed point has a coordinate x, y, and a time instant. This means that the neighbors of the analysed node, have to be searched in the next temporal layer. In figure 5.2 is shown the neighbors cells analyzed on this algorithm. Note that, the analysed neighbors cells include the cell with the actual position of the AGV.

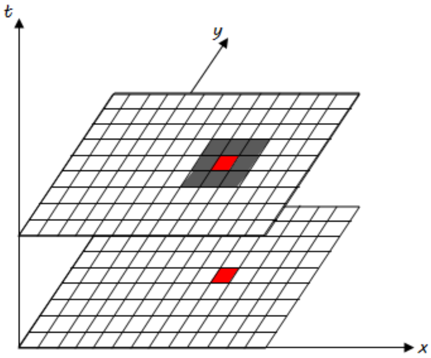


Figure 5.2: Representation of the analyzed neighbors cells

A relevant consequence is that, the AGV can stand in the same position (in the next temporal layer), if the movement is not safe. In figure 5.3, is shown a state where the vehicle stays in the same position, considering the next time instant.

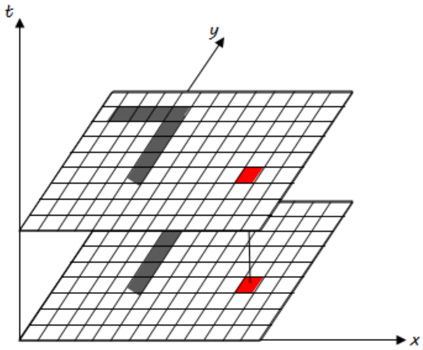


Figure 5.3: Example of a vehicle that stays in the same position

The description of the algorithm is presented in A.1.

The search order of the nodes are dependent of the cost function, which is calculated by the sum of the distance and the heuristic function. Here, the heuristic function used was the euclidian distance, that gives the distance in straight line to the final point.

The number of temporal layers depends of the number of iterations necessary to achieve the final point and the map dimension. How much larger the map is, more time layers are required. Also, if the vehicle finds many obstacles, can spend more time, to find the final point.

5.2.2 Implementation

The integration with the simulator, consists of a control loop that receives missions and runs de Time enhanced A*. This algorithm considers the other vehicles' path as obtacles. After the loop calculates the path for a given robot, it converts each position (in each instant of time), as obstacle cell to the next vehicles that will be analyzed.

The generic control loop is shown in figure 5.4. The missions are analysed, considering in a first-come first served approach. Each mission has a set of parameters that describes them: the status of the mission (complete or not complete); the vehicle to which it has been assigned to this mission; and the coordinates of the final point. It's important remember that the problem of multi-robot coordination does not include the assignment of the missions. It is assumed that there is another node that provides a list of the missions to be exectuted and a list of robots to be assigned.

Algorithm 2 shows in detail the integration of the Time enhanced A* considering a set of initial missions. Before that, it's important to note the following definitions:

- list_mission: Array of structs that contains the features of each mission such as assigned vehicle, status (completed or not-completed) and final point (fx, fy); Corresponds to the missions history.

- path[DX][DY][DT]: three-dimensional array with the map and the successive paths calculated;

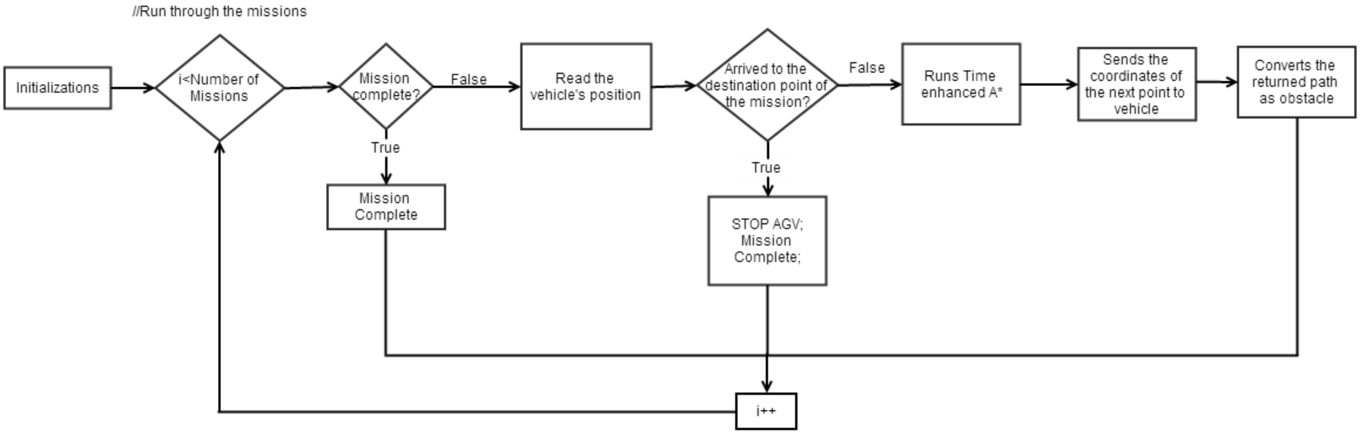

//Run through the missions
Initializations
i<Number of Missions
Mission complete?
False
True
Mission Complete
Read the vehicle's position
Arrived to the destination point of the mission?
False
True
STOP AGV; Mission Complete;
Runs Time enhanced A*
Sends the coordinates of the next point to vehicle
Converts the returned path as obstacle
i++

Figure 5.4: Diagram with the main sequence tasks of the control loop of the Time enhanced A*

Algorithm 2 Control Loop of the Time Enhanced A*

```
1:  {Verifies each mission}
2:  for i = 0 to number_of_missions do
3:    if mission is not concluded then
4:      vehicle = list_mission[i].vehicle {Check the assigned vehicle to mission i}
5:      {Reads the vehicle's position (x, y)}
6:      {For the vehicles that completed their mission and doesn't have any more mission, their position is converted to obstacle, for all time instants.}
7:      if first iteration then
8:        {Fill as obstacles the positions of all other vehicles}
9:      end if
10:     if $(abs(list\_mission.fx - x) \leq errorx) \&\& (abs(list\_mission.fy - y) \leq errory)$ then
11:       {Arrived to the destination point of the mission.}
12:       $orderAGV[vehicle] = STOP$ {stop order}
13:       $list\_mission[i].status = COMPLETED$ {Mission $i$ is concluded}
14:     else
15:       {The Time enhanced A* is executed, between the actual position to the final point of the mission}
16:       $path = AstarTime(x, y, list\_mission.fx, list\_mission.fy, path);$ {The returned path is assigned to the vehicle}
17:       {The coordinates of the second layer, are sent to the AGV node.}
18:       path= convertToObstacle(path) {The path is converted to obstacles for the next vehicles}
19:     end if
20:   end if
21:   {Mission Complete}
22: end for
23: {The matrix path is filled again with the initial map}
```

Before the start of the algorithm, the current position of the vehicle is verified and compared with the final point for that mission. This condition (line 11), considers an error relative to the final point. This error corresponds to the stopping distance.

5.2.2.1 Map Representation

The map construction is quite simple, because are only defined the free and occupied space. The simulation map is presented in the figure 5.5.

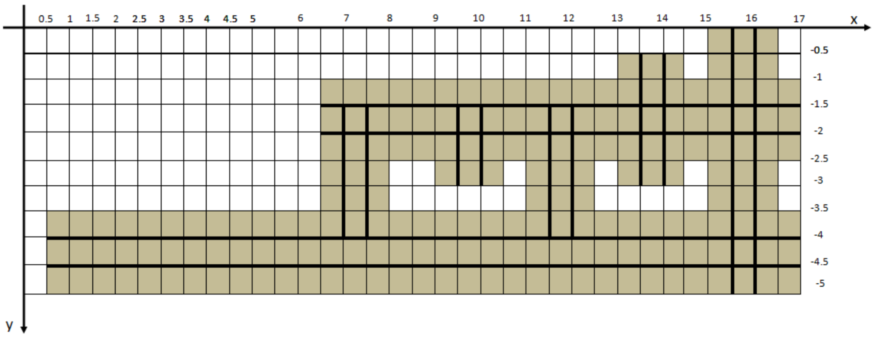


Figure 5.5: Map used to test the Time enhanced A* algorithm

Considering the size of each cell (50 cm) and the robot's size (60 x 40 cm), it was decided to extend the space that a robot occupied. In the control loop, the vehicles positions, are converted as obstacles, with a width of one cell per side. This situation was represented in the figure 5.6. Each AGV considers the other AGVs as extended obstacles.

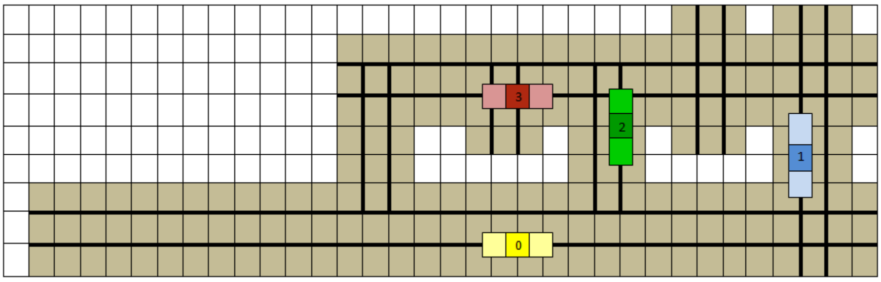


Figure 5.6: Example of extended obstacles for 3 vehicles

As can be seen in the figure 5.6, if the robot is in a horizontal corridor, the obstacle should be extended horizontally, and a similar way to the vertical corridor. In the case of the vehicle being in an intersection, reserves the cells according to their path's direction.

5.2.2.2 Trajectories Improvement

In order to improve the way how robots execute their paths, some modifications were made. However, to understand these changes, some explanations must be presented.

The result of *Time enhanced A** algorithm is a set of points (with 3 dimensions, x, y and time). Each temporal layer have the predicted robot's position, for that instant of time. For T=0 corresponds to the actual position and for $T=T_{max}$ corresponds the final point of the respective mission. Each iteration, return a three-dimensional matrix with dimensions $[DX][DY][T_{max}]$.

Therefore, the next point that should be sent to the AGV node, would be the position corresponding to the layer T=1. However, according to the map resolution, it was decided to send the point corresponding to the layer T=2. According to this, the vehicle trajectory is more constant, because it has fewer accelerations and disaccelerations.

Another improvement was in the sending of the points to the AGV node. Initially, a new point would only be sent if the AGV node confirmed the arrival of the last point. It's more efficient if the control cycle (with a given frequency) sends the sucessive commands with the position, without waiting for the indication of the AGV node. This way, the vehicle doesn't need to stop when reaching the intermediate point. The AGV node only takes in consideration the last point sent to it.

5.2.3 Collision Detection and Deadlocks

The system implemented on this dissertation it's a concurrently system, because it consists of some entities that execute code simultaneously and share resources. Some problems could occur on the control of shared resources: deadlocks and collisions. These are two challenges that any coordination problem should consider.

5.2.3.1 Deadlocks

A deadlock occurs when a set of shared resources, are simultaneously needed by two or more entities, none of which can/will grant priority/permission to others. It's called a deadlock because all of those entities' execution will be frozen forever. In this case, no process has the capacity to execute, or release the resource. There have been studied many approaches to solve the problem of deadlock, according to [25]:

- Ignore the problem: this solution makes sense in some systems, if the probability of occurs a deadlock was low and their consequences aren't serious .
- Detect and Recover: The algorithm has the capacity to detect a deadlock and recover the system, for example doing a system shutdown, roll-back of the processes, or among other solution adapted to the application.
- Avoidance: This is a dynamic solution that analyzes all the system, tests if the situation leads to a deadlock and executes accordingly.

Figure 5.7 shows a deadlock situation, applied to the map used.

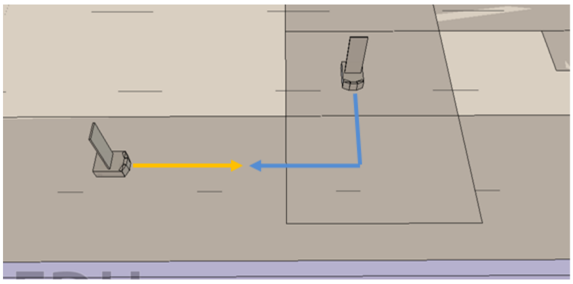


Figure 5.7: Example of a Deadlock

Although on this dissertation has not been implemented a resolution for the deadlock problem, some possible solutions will be presented. A possible methodology is to reserve the corridor until the next intersection. This way, these cells act as obstacles to the other vehicles, forcing them to stop or choose another path.

Another alternative may be an avoidance approach. It consists of a pre-processing function that checks the possibility of occurrence of a deadlock. If one is detected, there are some alternatives to solve it.

A test consisting of performing successive simulations, with permutations of N AGVs, N by N, P_N^N, can be made in order to predict deadlocks. In each simulation a robot's path is delayed while the other one keeps following it's original path. When a robot combination and their respective paths, which does not lead to a deadlock, is found, a solution is obtained.

5.2.3.2 Collision Detection

A problem that might arise while vehicles are moving are collisions. In order to avoid it, *Time Enhanced A** forces a robot to stop. It's considered unsafe movement, if the surrounding cells are obstacles (walls or paths of another AGVs). This is possible, due to the implemented changes relative to the traditional A*. The neighbors cells analyzed also includes the cell with the same coordinates of the actual robot cell (see figure 5.2). This allows that a vehicle keeps the position in the next instant of time.

Another implemented feature is related with the costs given to each movement. The cost to keep a given AGV in the cell, is lower than the cost of moving it in any direction. Depending of the costs difference, robots could be kept in the same position for more or less instants of time. The costs used on the simulation tests were:

1. COST1= 20 // Keep on the same position (This value was adjusted according to the performed tests);
2. COST2= 100 // Moves the robot in the orthogonal directions (size of each cell);
3. COST3= 141 // Moves the robot in diagonal direction (This value can be obtained using the Pythagorean theorem);

If the COST1 is zero, the AGV remains stopped. On the other hand, if $COST2 \leq COST1 \leq COST3$, instead of remaining on the same position, the AGV pointlessly moves to a neighboring cell and then returns to the cell from which it came.

5.3 Modified Banker's Algorithm

Modified Banker's Algorithm are based on the article [15]. This is an algorithm dedicated for the scheduling and routing in multi-robot systems, in order to avoid collisions and deadlocks. The Modified Banker's Algorithm presentes some improvements relative to the traditional Banker's Algorithm, according to [26]:

- If the segments belonging to a given path are available, the path is immediately assigned to the vehicle, without checking if the path is feasible. This means that, the only condition for a given path be assigned, is their segments are free, even that have already been reserved by another vehicle.
- Temporary allocation of unsafe states. A state is unsafe, if it was already reserved by another vehicle. Although a given segment has already been reserved by another path, this condition allows this segment to be able to be temporarily occupied by another vehicle, if some safety conditions were ensured. This will be explained in section 5.3.2.1.
- Split a mission's path in parts, in order to improve the exploitation of vehicles.

5.3.1 Description

5.3.1.1 Map Representation

The layout required to develop this algorithm, is a directed graph, which contains nodes (in the intersections and in the end of each path) and arcs (or segments). Here will be used the same notation of the article.

The graph is represented by G= (N, A), where $N = \{n_1, n_2, ..., n_n\}$ are a set of nodes, and $A = \{a_1, a_2, ..., a_m\}$ are a set of weight arcs. In the considered approach, the weight of each arc was the length of it.

It is assumed that the AGVs stays in the arcs and that only one vehicle can occupy a segment, at each instant of time.

Considering that this algorithm is only focused on the vehicle scheduling, a path planning algorithm to calculate the path for each vehicle, is also required. As in the Banker's Algorithm is not proposed any method to calculate the vehicles' path, it was decided to implement the traditional A*. However, the A* implemented needs of a grid map. So, it's required to adapt the directed graph used by Banker's Algorithm, to a grid map. However, this was a tiresome task that spents many time. This process should be improved, as future work, in order to reduce the time spent.

An example of the map used on the simulation is represented in the figure 5.8.

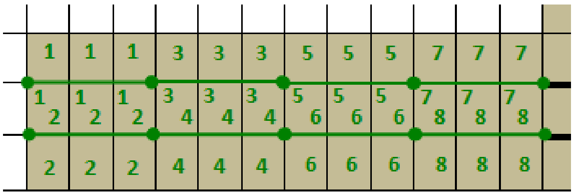


Figure 5.8: Portion of the map used to test the Banker's Algorithm

It's important to note that the numbers in the figure 5.8 represents segments' IDs. The width of each segment corresponds to three cells, due to the robot's size. The coordinates of all segments and nodes were read from a text file and the map was constructed.

5.3.1.2 Definition

The input of this algorithm is the same that was considered on the *Time enhanced A**: a set of missions and assigned vehicles. Each mission includes the initial and final points.

Before presenting the description of the Modified Banker's Algorithm, must be taken in consideration some definitions:

- Binary Array a: number of available resources. Indicates if each segment is available or occupied.

$$a = [s_1, s_2, ..., s_n]^T \tag{5.1}$$

- Binary Matrix H: resources required by each AGV. The dimensions of this matrix are $n \times m$. n refers to the resources/segments and m indicates the AGVs.

$$H = \begin{bmatrix} r_{1,1} & r_{2,1} & \cdots & r_{n,1} \\ r_{1,2} & r_{2,2} & \cdots & r_{n,2} \\ \vdots & \vdots & \ddots & \vdots \\ r_{1,m} & r_{2,m} & \cdots & r_{n,m} \end{bmatrix} \tag{5.2}$$

where $r_{n,m} = 1$ indicates that the resource n was reserved by the vehicle m.

- Binary Matrix N: indicates to each AGV, the segments that it needs to finish the mission. This matrix have the same dimensions of H.

The Algorithm 3 shows the pseudocode of the implemented algorithm. When a new mission arrives, the path is calculated using the A* method and the current position of the respective AGV. The trajectory planning is calculated offline and each path is determined only one time.

Algorithm 3 Modified Banker's Algorithm

```
Initializations
for i = 0 to NUMBER_AGVs do
  if AGVi has mission assigned then
    {Read the position of vehicle i}
    Refresh the array $a$;
    if NextSegment == SAFE then
      {A state is safe, if the segment aren't reserved by another robot. Check the values of $H[NextSegment][i]$, $a[NextSegment]$ and $n[NextNode]$, in order to avoid collisions. $n$ is described in 5.3.2.2.}
      AGVi GO;
      {The coordinates are sent to the AGV node. The node and the segment of the next position, are reserved.}
    else
      {NextSegment is unsafe, because was reserved by another robot. }
      if Following Segments of the vehicle's path == SAFE then
        AGVi GO;
      else
        AGVi STOP;
      end if
    end if
  end if
end for
```

In this algorithm, the result of each iteration is an order to each AGV node (consider the architecture described in 4). The order is a variable that controls the binary movement of a given AGV. In case of movement order, the coordinates of the next point are also sent. The next point is determined, checking the path that was assigned to the mission and the actual AGV's position.

5.3.2 Collision Detection and Deadlocks

5.3.2.1 Deadlock

The occurence of a deadlock situation was considered on the Modified Banker's Algorithm, described in [26]. As explained in the section 5.2.3.1 there are some approaches that can be implemented to solve this problem.

The solution proposed by the article was based on the avoidance approach, and consists in allocating temporarily unsafe states. As mentioned before, an unsafe state consists of a segment and respective node already reserved by another AGV path. This possibility allows a vehicle occupy briefly, the segment belonging to another path. However, some safety restrictions are imposed:

- There is a free pathway, between the atual segment of the vehicle (s_i) and the following segment of the already reserved segment (s_j);

- The segment s_j is a safe state. A safe state implies that the segment has not been reserved by any vehicle.

This is one of the implemented improvements by [26] relative to the Banker's Algorithm.

5.3.2.2 Collision Detection

In order to prevent the collisions in nodes, was created a binary array n with dimensions $NUM\_AGV \times 1$. When a vehicle arrives at the node security distance, this node is allocated on the array n as occupied node.

This way, in the next iterations, a vehicle is permitted to move, only if the respective position in the next node's path is zero.

5.4 Conclusions and Overview

The presented approaches have some advantages, and can be used in different situations. However, the more relevant difference between them is the path calculation.

In the *Time enhanced A* Algorithm*, the paths are determined on each iteration, allowing the vehicles to adapt their trajectories according to the environment changes and the other AGVs movements.

The *Modified Banker's Algorithm* has well-defined segments and nodes, and determines the path offline. The movements of each AGV are controlled considering only the calculated path.

In the chapter 6 it is described the behaviour of each algorithm in different scenarios and also focus the Time enhanced A* and shown some of their results.

Chapter 6

Results

This chapter compares the two tested algorithms, considering the main features of each one. The conclusions are presented and discussed the advantages and applications of each algorithm. In section 6.2 is shown some results of the *Time enhanced A** and its behaviour to some critical situations. It was also considered a temporal analysis, in order to evaluate their efficiency.

6.1 Comparison of the Algorithms

Any multi-robot coordination problem, considers two distinct problems: path planning and robot's coordination. The implemented algorithms handles these problems in different ways.

The *Time enhanced A* Algorithm* considers *path planning* and *coordination* simultaneously. The path of each vehicle is calculated according to the paths of the other vehicles. In this sense, the coordination between AGVs is ensured. *Modified Banker's Algorithm* is an algorithm dedicated for routing in multi-robot systems, and does not include a path planning method. In this case, two algorithms are necessary for path planning and coordination between the vehicles.

Another difference between these algorithms is in the map construction. Whereas the Time enhanced A* only indicates the free and occupied spaces, the Banker's Algorithm needs a database that includes the coordinates of the segments and nodes. The vehicles' path is restricted to the predefined segments, while the *Time enhanced A** gives the vehicle the flexibility to navigate in all free cells.

Considering that the *Time enhanced A* Algorithm* includes time and movements of the other vehicles on their paths' calculation, overtaking is allowed. *Modified Banker's Algorithm* limits the vehicles' navigation to their respective paths, thus is not allowed vehicle overtaking.

Modified Banker's Algorithm although is a robust algorithm and could be applied in industrial environment is less efficient, in terms of the paths' selection and the consumption of resources. It would be advantageous to invest in other types of systems for the improvement of the industrial environment, such as *Time enhanced A* Algorithm*.

In the follow example, is presented a case that shows different results for each algorithm.

In figure 6.1 consider the positions of the vehicles 1 and 2 and their destination points. The final points are identified with a cross with the same color of each vehicle. Consider also that the mission of the vehicle 1 arrives first than the mission of the vehicle 2.

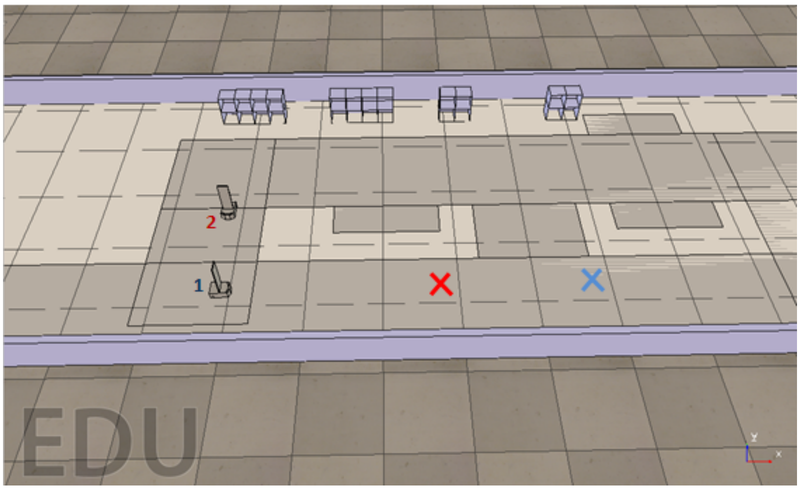


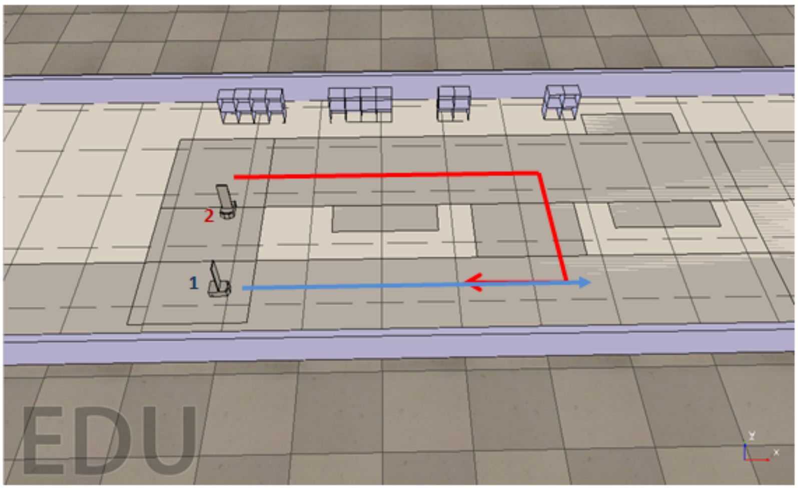
Figure 6.1: Comparative Example of the two algorithms. The cross represents the destination point for each AGV.

Modified Banker's Algorithm analyzes the destination point of the vehicle 1, and reserve all path cells. The second vehicle has to choose a long way, because no cell belonging to the shortest path is available. The calculated paths by this algorithm are represented in the figure 6.2a.

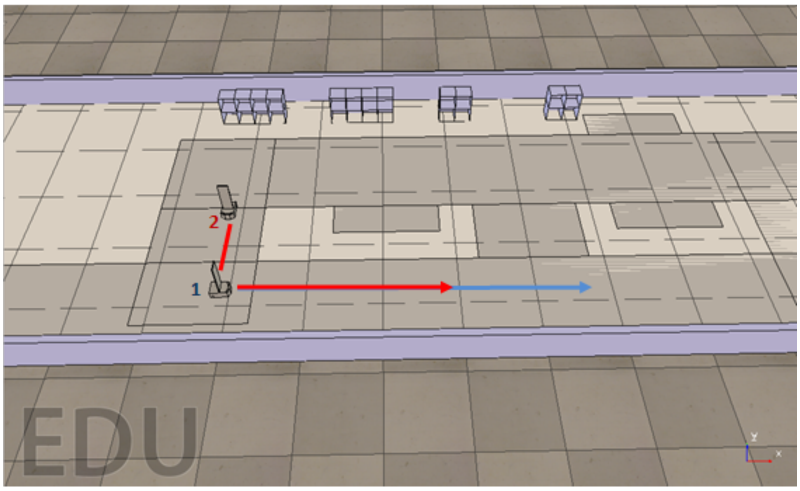
*Time enhanced A** first calculates the path of the vehicle 1 but when arrives the second mission, vehicle 2 calculates their path considering the movements of the vehicle 1, in each time instant. Unlike the Banker's Algorithm, the Time enhanced A* does not allocate the complete path, but only the cells occupied in each temporal layer. When the vehicle 1 releases the cells of the shortest path, vehicle 2 is allowed to go behind the vehicle 1. The result of this algorithm considers the path cost, and the cost of the longer path is bigger than the cost of the shortest path. The resultant path for the AGVs 1 and 2 are represented in the figure 6.2b.

Note that, the figure 6.2c shows the 3D-view of the calculated paths by the *Time enhanced A* Algorithm*. The vehicle 2 (shown in red) follows behind the vehicle 1, since the point (16, 8).

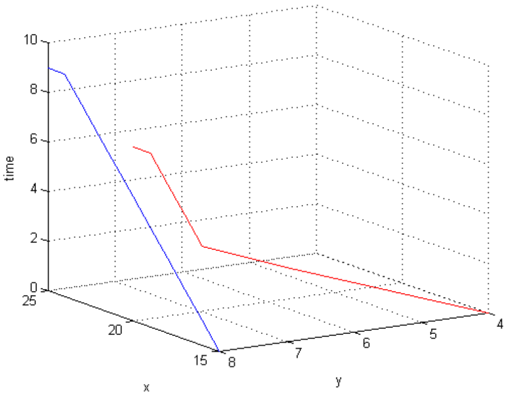
Considering that many of industrial applications requires systems that uses the minimal number of resources, the solution of the *Time enhanced A** seems to be a good result in terms of efficiency, because AGV2 choosed the shortest path spending less resources, such as battery.



(a) Result of the Modified Banker's Algorithm



(b) Result of the Time enhanced A* Algorithm



(c) Result of the A* Algorithm- 3D View

Figure 6.2: Comparison of the result trajectory between the Modified Banker's Algorithm and the Time enhanced A*

6.1.1 Conclusions and Overview

It's possible to conclude that the *Time enhanced A* Algorithm* is more flexible in the generation of paths, because the calculated paths can be adjusted according to their cost, obstacles and the other AGV movements.

As shown before, *Time Enhanced A** also allows saving resources, because the algorithm always chooses the low-cost path and in the last instance, sends a stop order to the AGV. Regarding this, it can be applied to many areas, including industrial environment, robot football, and other multi-robot systems.

Considering this, the section 6.2 presents some results of the *Time enhanced A** according to the main critical situations.

6.2 Time Enhanced A* in detail

6.2.1 Temporal Analysis

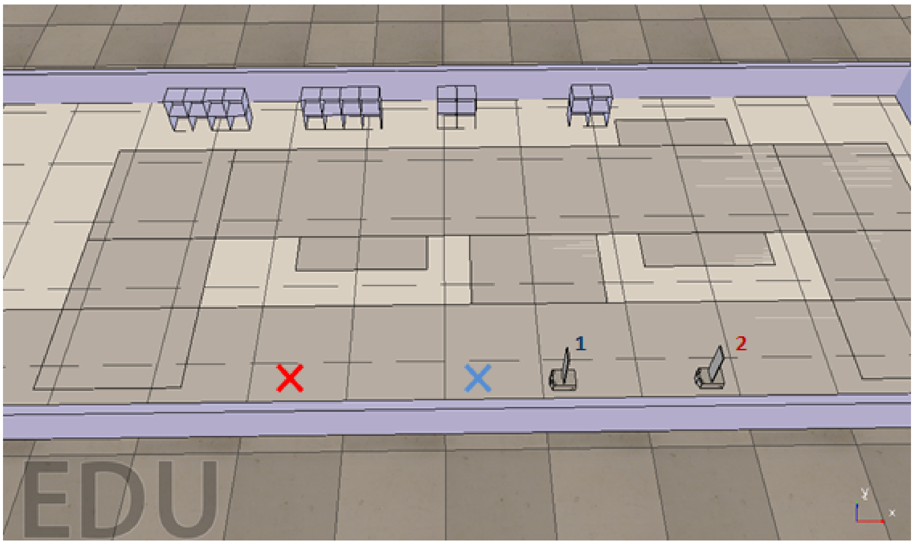
The efficiency of the multi-robot coordination algorithm is determined not only by the correctly execution of their missions, but also according to the processing time. In order to improve the performance of this algorithm, some parameters had to be adjusted:

- Cycle Time of the Coordination Control Algorithm $f = 15Hz$
- Cycle Time of the AGV Control Algorithm $f = 20Hz$
- Cycle Time of the Simulation $t = 100ms$
- Robot's Velocity $v_{linear} = 0,2ms^{-1}$

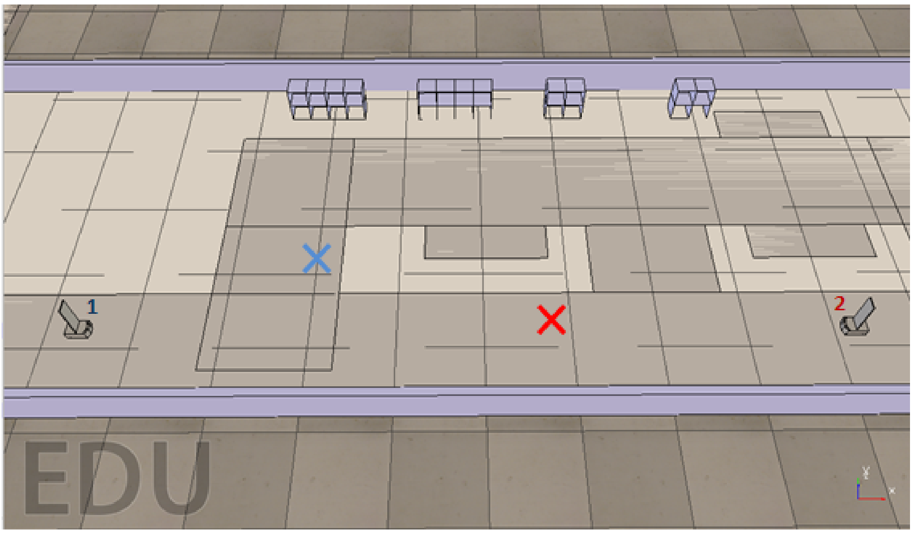
Remember that the Coordination Control Algorithm and the AGV Control Algorithm runs in parallel. In order to have a better evaluation of the algorithm efficiency, some tests were performed, considering three diferent situations:

- Test 1: Example of an overtaking. Two AGVs, 1 and 2, follow in the same direction with the same speed. Vehicle 2 overtakes the vehicle 1.
- Test2: Example of two paths without intersections.
- Test3: Example of the intersection of two AGVs, followed by the passage and overtake of vehicle 2 by the vehicle 1.

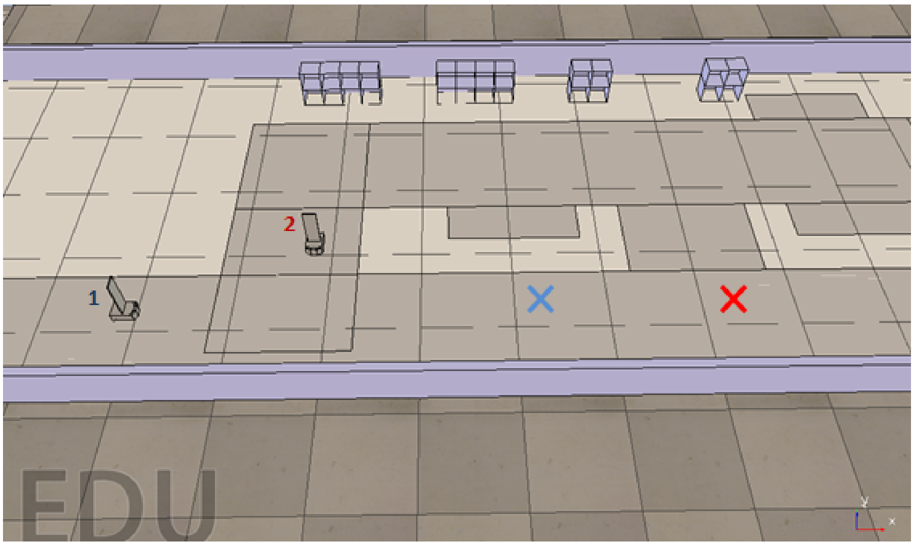
In the figures 6.3a, 6.3b, 6.3c are represented the initial and final points of each mission, for the three tests.



(a) Test1



(b) Test2



(c) Test3

Figure 6.3: Description of the performed tests to the Temporal Analysis

The temporal analysis was made considering two types of measures, see tables 6.1 and 6.2.

The processing time of the Time enhanced A* is evaluated on the table 6.1. In order to perform this analysis, the times spent by the algorithm, to calculate the path to each AGV on each iteration were written on a file. Then the average of the 100 samples, were taken. The values are represented in milliseconds.

	Test 1		**Test 2**		**Test 3**	
Experiments	AGV1	AGV2	AGV1	AGV2	AGV1	AGV2
1	1,66873	1,6501	1,6428	1,77901	1,71403	1,84881
2	1,5757	1,7476	1,70165	1,83552	1,72404	1,79396
3	1,6106	1,7681	1,60713	1,73009	1,70682	1,69733
Average	1,6106	1,748	1,6428	1,77901	1,71403	1,79396

Table 6.1: Temporal Analysis: Averages of the time spent by the Time enhanced A* to determines the paths for each vehicle. These values are represented in milliseconds.

As it's possible to see, the average of the time spent to calculate the path corresponding to the tests 1 and 2 are very similar. However, test 3 presents the largest duration. These results were expected because the test 3 includes the longest paths, and a bigger number of analysed cells.

Nonetheless, the changes between the tests are not very significant.

In the table 6.2 are indicated the execution time of the considered missions. For each test were made five experiences and were calculated the respective averages.

Experiments	**Test 1**	**Test 2**	**Test 3**
1	16,337	11,735	54,670
2	16,204	15,183	52,602
3	16,547	15,308	53,669
4	16,471	11,470	46,270
5	16,203	12,869	55,202
Average	16,352	13,313	53,669

Table 6.2: Temporal Analysis: Duration of the complete execution of each test, in seconds

Looking at the test's missions, it seems clear that test 3 has a largest duration since it requires the longest displacement and the avoidance of a crossing. In a similar way, test 2 has the smallest duration since it has the shortest path.

These values show that the processing time of this algorithm is satisfactory, however more tests must be performed, in order to have a better evaluation.

6.2.2 Vehicles Crossing

Vehicles crossing is an interesting situation that allows to take some conclusions relative to the behaviour of the algorithm. In example 1, 6.2.2.1 was simulated a crossing between two AGVs, considering that the obstacles do not allow the vehicles to choose an alternative path (one of the vehicles must stop). In the example 2, 6.2.2.2, three AGVs were used, in order to evaluate how the algorithm solves a more difficult situation.

6.2.2.1 Example 1

Consider the map presented in the figure 6.4.

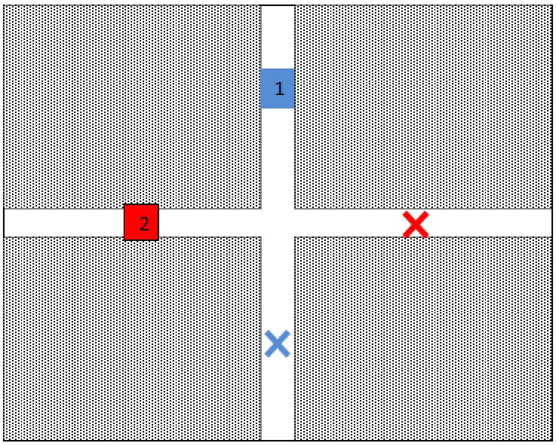
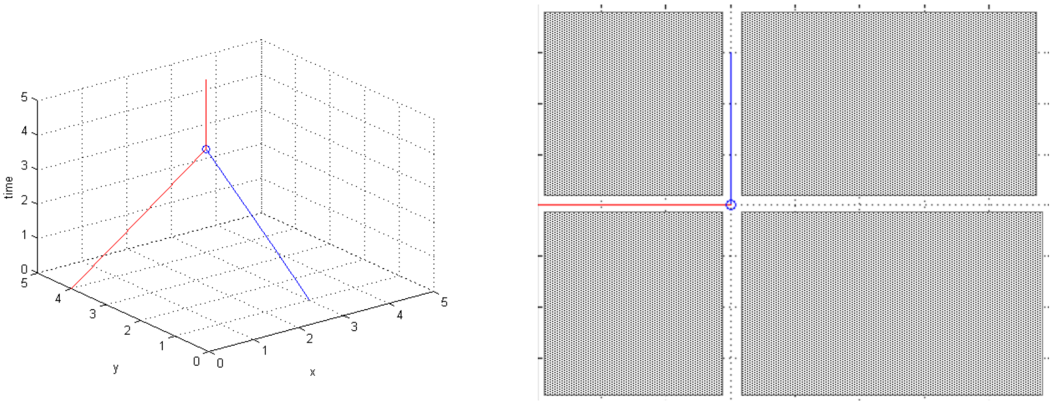


Figure 6.4: Crossing Example

The vehicles 1 and 2 are at the same distance of the collision point. Also note that the obstacles' position does not allow the AGVs to choose another alternative path.

The image 6.5 shows the paths calculated by the A*, independently for each AGV. As it's possible to see, occurs a collision in the point (3, 4). Each vehicle doesn't consider the movement of the other vehicle.



(a) 3D View

(b) 2D View

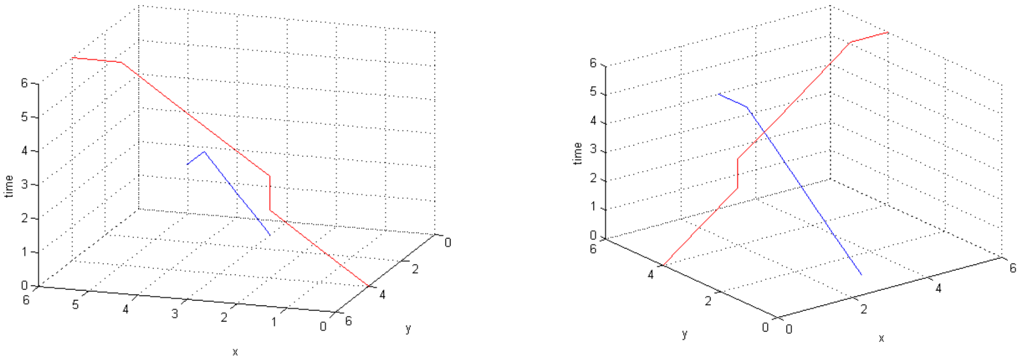
Figure 6.5: Example of a crossing, using the A*, independently for each AGV

It was assumed that after the collision, the AGV stops and stays in the same position. In this case, at the point of intersection, the two vehicles crash and stop (vertical line segment in the figure 6.5a).

This is due to the fact that in an industrial environment, if some undesirable situation occurs, the vehicles should stop in order to avoid a dangerous situation.

Using the *Time enhanced A* Algorithm*, the collision is avoided, as can be seen in figures 6.6 and 6.8.

It's important to note that the vehicle 2 (represented in red, in the figure 6.6) stops while vehicle 1 crosses the intersection point. This can be seen, by the vertical line segment in the red line.



(a)

(b)

Figure 6.6: Example of a crossing, using the Time enhanced A* - 3D View

The figure 6.6 allows to clearly see the collision avoidance. However, in order to a better visualization about the paths performed by each AGV, the figure 6.7 was generated. Each frame includes the position of AGV 1 and AGV 2.

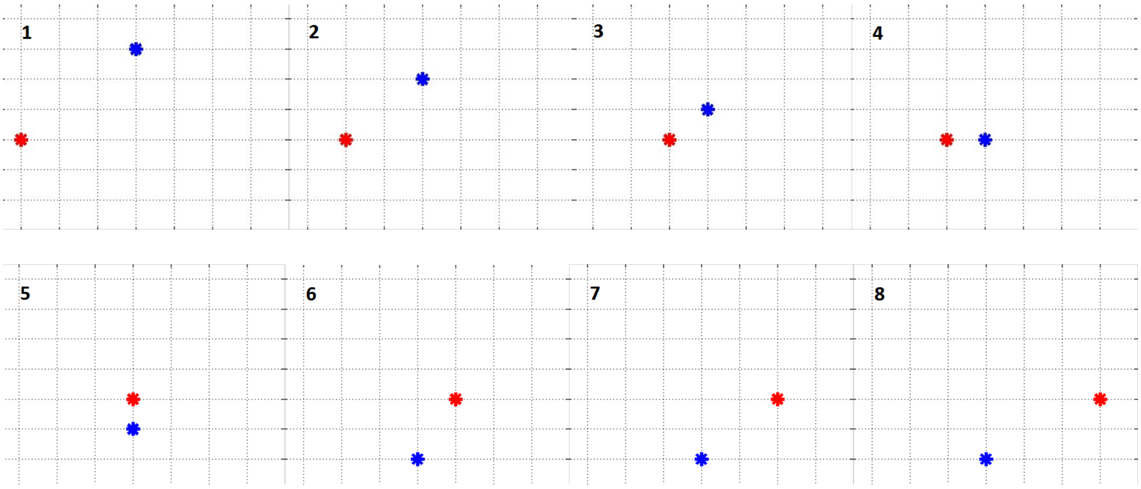


Figure 6.7: Example of a crossing, using the Time enhanced A* - Frames

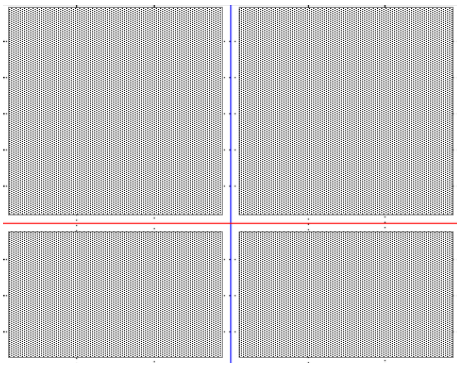


Figure 6.8: Example of a crossing, using the Time enhanced A* - 2D View

The 2D view, in the figure 6.8, allows to see in a clear way, the movement of each AGV, considering the position of the obstacles. In this case, in order to provide a difficult situation, none of the vehicles had the opportunity to choose another alternative path.

The possibility of a given AGV to stop in case of none neighbor cell available, is provided by the modifications added to the A*. The cost of a vehicle to stay in the same position, is lower than moving it to any direction. Also remember that the analyzed neighbors cells includes the cell with the same position of the considered point. So, the vehicle 2 keeps its position.

6.2.2.2 Example 2

Consider the map represented in the figure 6.9 and the initial and final points of each mission. In this example, the objective was to simulate a collision between three AGVs.

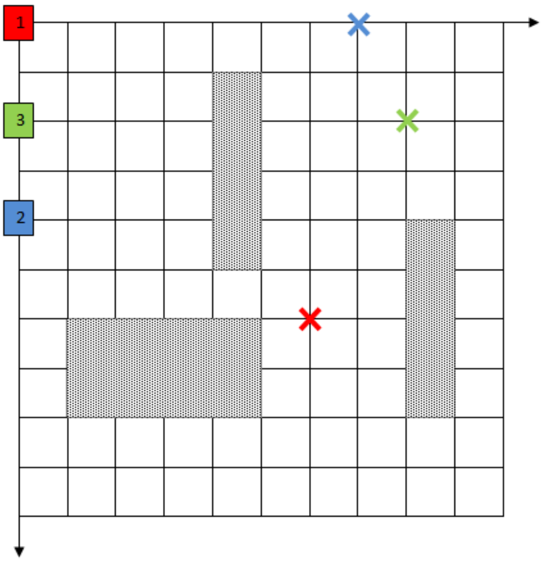
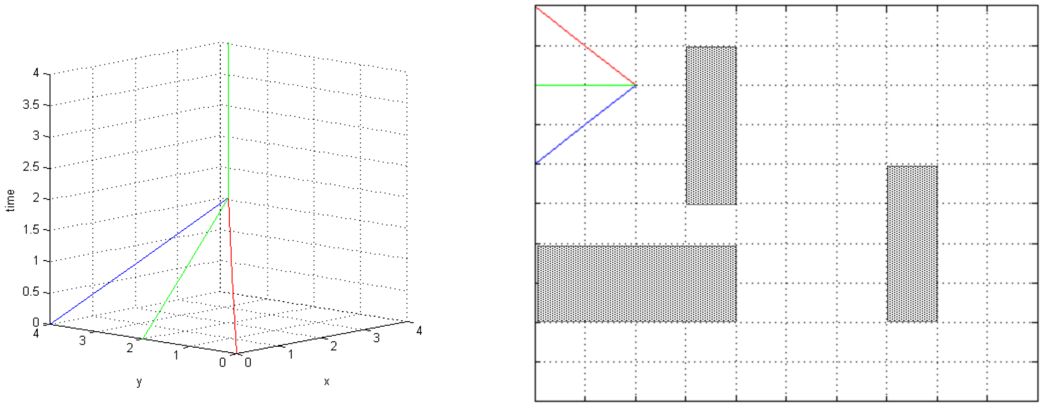


Figure 6.9: Map used in the example 2 of a crossing

As shown in the figure 6.10 the collision occurs at the point (2, 2), using the A* independently to each AGV. Thenceforth, was considered that the vehicles remained stopped. Since the A* doesn't consider the time movements of each vehicle, it cannot foresee their positions.

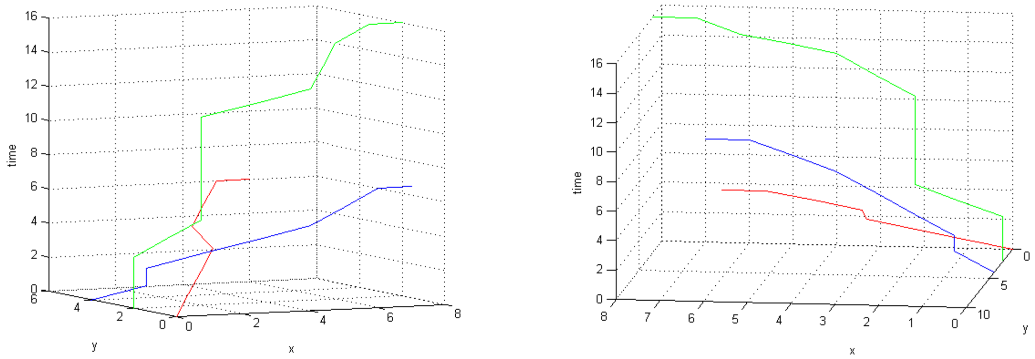


(a) 3D View

(b) 2D View

Figure 6.10: Example of a crossing, using the A* and 3 AGVs. The paths of each vehicle, were calculated in an independent way for each one.

Testing the same situation, using the *Time enhanced A**, verifies that the crossing it's avoided. The vehicle 2 (represented in blue) waits until the first vehicle (in red) crosses the critical point. Consequently the vehicle 3 also waits. In the images 6.11a and 6.11b it's possible to see, in different views, the avoidance of collisions. Note that when the second vehicle goes, the AGV 3 is still waiting for its turn. Only when the intersection point is free, the third vehicle has permission to go.



(a)

(b)

Figure 6.11: Example 2 of a crossing, using the Time enhanced A* - 3D View

In the figure 6.12 are represented a sequence of frames that show the vehicles' positions to each time instant.

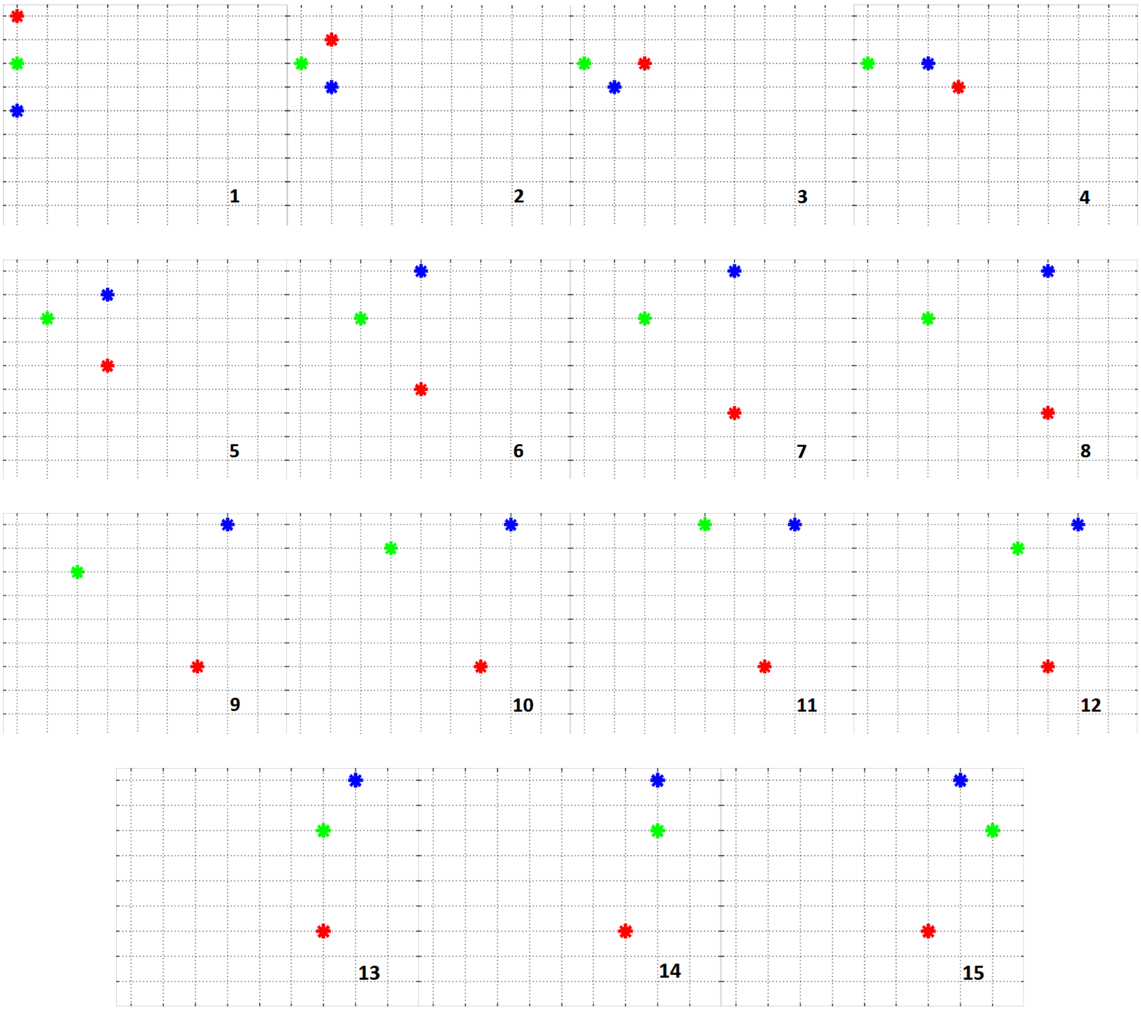


Figure 6.12: Example 2 of a crossing, using the Time enhanced A* - Frames

The figure 6.13 shows the 2-D view, and allows to see, in an easy way, the paths calculated by each AGV.

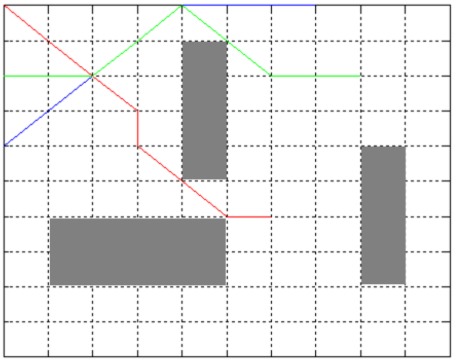


Figure 6.13: Example 2 of a crossing, using the Time enhanced A* - 2D View

The examples presented shows the powerful capabilities of the *Time enhanced A* Algorithm*.

The fact that, this algorithm considers time, it's an advantage because each AGV can predict the positions of the other vehicles, improving their path and avoiding colisions.

6.2.3 Overtaking

The study presented in this dissertation considers that the vehicles' velocity are equal. So, the overtaking here simulated considers the vehicle 1 stopped when the second overtakes it. The figure 6.14 shows the initial and final points of the missions assigned to each AGV.

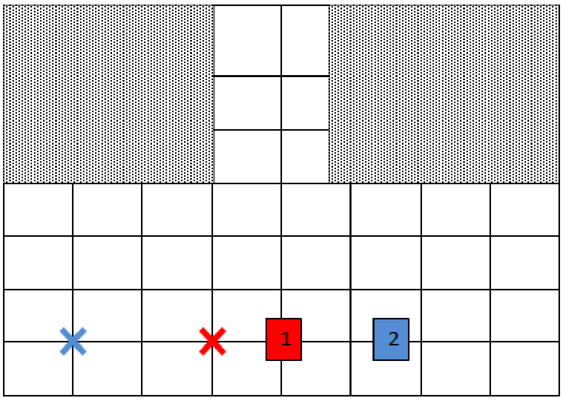


Figure 6.14: Overtaking: Initial and Final Points of the Missions

The figure 6.15 represents the 3-D view of the paths calculated using the traditional A*. In this case, when the first vehicle stops (in red), its position is extended over the time. This is due to the fact that the second vehicle, needs to know that there is a vehicle stopped in that position.

Using the A* Algorithm independently to each AGV, a collision occurs.

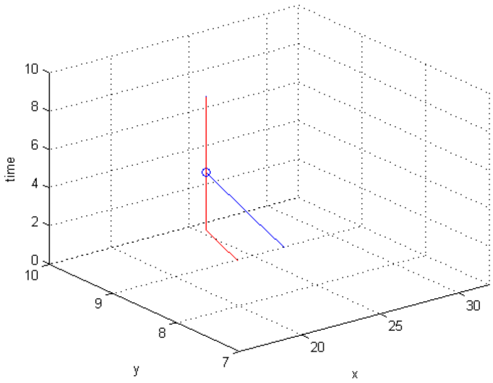
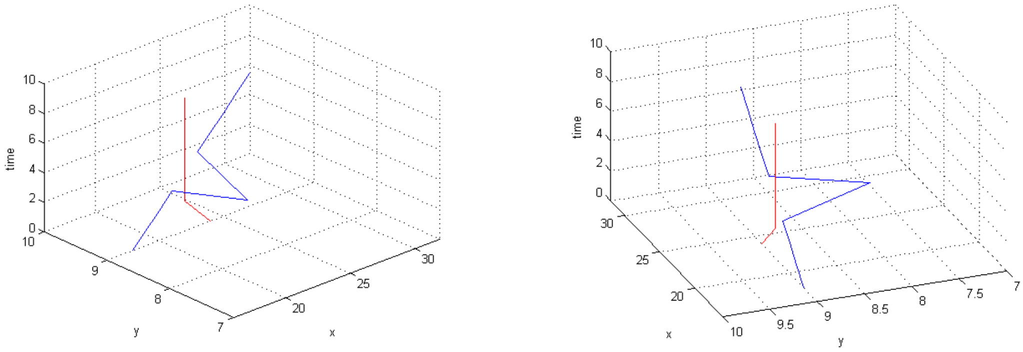


Figure 6.15: Example of an overtaking using the A*- 3D View

The figures 6.16a and 6.16b shows vehicle 2 avoiding the first vehicle. These images are the resultant paths of the *Time enhanced A* Algorithm*. As can be seen vehicle 2 deviates from the first vehicle, occupying the neighbors cells. If these cells are also occupied, vehicle 2 would stop, and causes a deadlock.



(a) (b)

Figure 6.16: Example of an overtaking, using the Time enhanced A* - 3D View

The position of each AGV, for each temporal layer, are represented in figure 6.17.

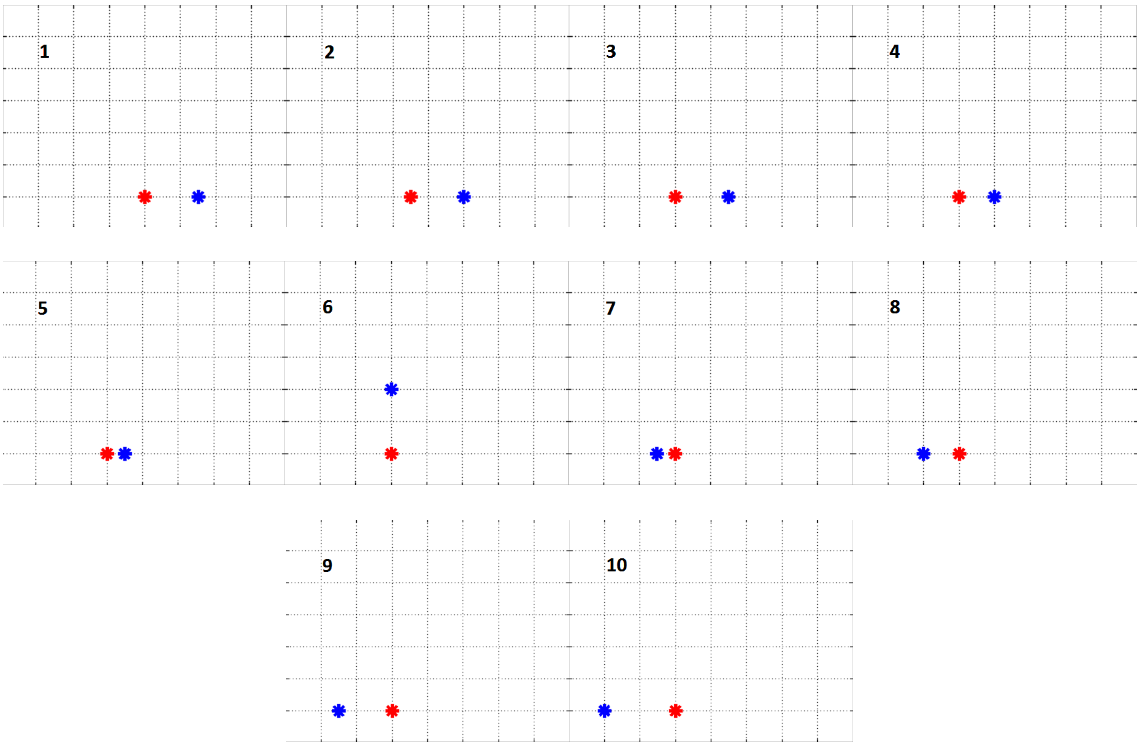


Figure 6.17: Example of an overtaking, using the Time enhanced A* - Frames

The 2-D View allows to see the paths of each AGV considering the obstacles positions. Vehicle 2 sees vehicle 1's position as an obstacle and diverts its path to the neighbouring cell.

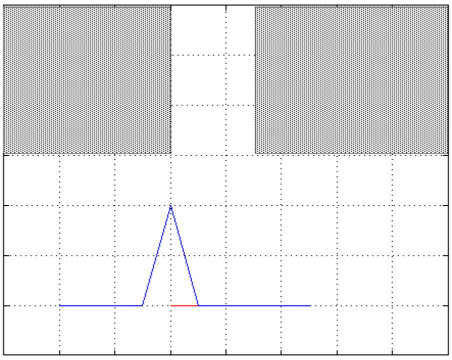


Figure 6.18: Example of an overtaking, using the Time enhanced A* - 2D View

An additional implementation that provides a more realistic simulation, is the cost assignment to each corridor. As can be seen in the figure 5.5 the map has two corridors in each segment. In the most cases, in a real industrial environment this is not acceptable. Industry needs to save resources, including usable space. Having two corridors, spends a lot of space. Thus, it seemed interesting to give a cost to each corridor, in order to establish priorities. The objective was to define some corridors as main corridors, and the others would only be used for special cases, such as overtaking or as an alternative path.

This implementation, was easy. In the algorithm when the costs corresponding to the heuristic function and the distance are considered, is also added the cost of the respective corridor.

6.3 Conclusions and Overview

The results presented here, show the enormous capabilities of the *Time enhanced A* Algorithm*, to be applied in any multi-robot systems. It's a robust algorithm, that provides good solutions in terms of paths efficiency, consumption of resources and processing time. In this sense, this new concept, is a good improvement of the path planning method, A*.

The tests presented here (crossing between two and three vehicles and an overtaking) are some of the main problems that a coordination algorithm must solve. As was seen, this algorithm can find good solutions. There are other tests that would be interesting to implement, mainly if considered others features and requirements of the project, such as different velocities between the AGVs.

In [27] are shown several videos that include some simulation tests.

In the next chapter are presented the conclusions of this dissertation and some future work.

Chapter 7

Conclusions

The initial objective of this dissertation was to solve a multi-robot coordination problem and develop a simulation scenario that comprises the main elements of a factory.

In this sense, a careful study of path planning methods was made, including their advantages and the techniques used to construct the map (roadmap, cell decomposition). It were also studied some developed works in the area of multi-robot coordination, and their main used approaches.

According to this, a coordination algorithm was chosen and implemented. It was decided to improve the path planning method A*, and compare it with a known coordination method, the Modified Banker's Algorithm.

The A* algorithm gives the optimal path between two points, however regarding to the coordination of multiple robots, some improvements can be made to this algorithm. This dissertaion contributes extending the A* algorithm with a time component, not only to provide a method that calculates the optimal path to each AGV, but also, to avoid collisions between them. Was proposed the *Time enhanced A**.

Furthermore, some available simulators on the market were studied, and a software that fulfilled the requirements of this project was choosen. V-Rep seems to be a good choice, because it includes many elements that creates realistic simulations, and gives us, in an easy way, the ability to program and incorporate the developed algorithms.

In order to make some tests two simulations scenes were created, based on the PSA plant. Each of them was used in different stages of this project. The first scene allowed to simulate the communication between V-Rep and the C++ programs using ROS, and the navigation of a single robot. The second scenario, was intend to create more collisions situations used to test the developed algorithms.

The description and a comparison between the *Time enhanced A* Algorithm* and the *Modified Banker's Algorithm* were made. These methods can be applied in different pratical solutions, however the *Time enhanced A** has the capacity to adapt a given path according to the environment changes and to the other vehicles paths.

The *Time enhanced A* Algorithm* provides a set of features that makes it a powerful algorithm. The fact that the time is being considered on this algorithm allows it to predict the paths of the other vehicles, for each instant of time. Beyond that, considering a vehicles' crossing situation, and that does not exist an alternative path, one of the vehicles must stop and remain on its position until the intersection point is free. This possibility gives to the algorithm the capacity to save resources (like energy), because it always chooses the path with the lowest cost, and in last instance, stops the AGV during a certain period of time.

Any feasible algorithm not only provides a correctly execution of an initial set of missions, but also completes them, during an acceptable time. According to this it was made a temporal analysis, that included some tests and the timing required by each of them.

Finally, some results using possible collisions' situations were presented, in order to validate the new implemented algorithm.

7.1 Fulfillment of the defined objectives

The objectives of this dissertation were fulfilled considering that a solution to the multi-robot coordination problem was developed, implemented and simulated.

A study about some powerful simulators was made in order to select one. V-Rep allowed to perform different realistic scenarios to this thesis.

Also, some works in the area of multi-robot coordination were studied. A new approach of the A* algorithm was implemented and tested. In order to evaluate its performance, was compared with a method studied on the literature review, the Modified Banker's Algorithm.

Considering the reasons presented above, it's possible to conclude that the main objectives were achieved although there are still some improvements that can be performed. So, in section 7.3 are discussed some future work that would be interesting to implement as complement of this dissertation.

7.2 Main Difficulties

In the development of this work, some difficulties came up and most of them, are related with the tools used.

ROS is a robotic platform that help programmers with the cooperative works. It's a tool with enormous capabilities as an integration platform. However, its learning time is relatively large considering the provided time to the development of this dissertation. Thus, some improvements could be made on the implemented ROS code.

Furthermore, the integration of the ROS on the simulator was a task which also consumed a substantial portion of attention on the beginning of this work. Initially, it was necessary to install a V-Rep plugin that supports the ROS functionality and then the tutorials were followed in order to receive and send commands to/from an entity in the simulation.

Regardless this, these difficulties allowed me to develop my programming skills and knowledges about some powerful robotics tools.

7.3 Future Work

There are some improvements and developments that would be interesting to perform, as an extension of this dissertation.

First of all more tests to the *Time enhanced A* Algorithm* should be performed using a higher number of robots increasing the probability of occurrence a collision.

Another interesting development would be to test the new extended version of the A* algorithm, on a real test platform. A system with multiple robots should be used. This test should allow to analyze the developed algorithm according to its robustness.

Finally, it would be interesting to give a new parameter (priority) to each mission. Then, each AGV should navigate respecting the priorities.

Appendix A

Time enhanced A*

In this appendix will be described the new A* algorithm, Time enhanced A*.

A.1 Pseudocode

Before presenting the algorithm some definitions should be clarified:

- O: open list, which has the non-analysed nodes and that could be selected;
- (ix, iy): initial point;
- (fx, fy): final point;
- DX, DY: dimensions x and y of the map;
- DT: number of temporal layers;
- map[DX][DY][DT]: three dimensional array with the map, over all time instants;
- heu[DX][DY][DT]: three dimensional array with the values of heuristic function, for each point. These values are calculated considering the estimated cost to go from the node's position to the final point;
- dis[DX][DY][DT]: three dimensional array with the values of the distance, for each point. These values represents the cost of going from the initial point to the analysed node;
- ptr[DX][DY][DT][0]: coordinate x of the parent node;
- ptr[DX][DY][DT][1]: coordinate y of the parent node;
- ptr[DX][DY][DT][2]: time instant of the parent node;
- COST1 /COST2 /COST3: cost of to stay in the same position, cost of following in the same direction and cost of to walk in the diagonal, respectively;

In 4 is presented the implemented algorithm.

Algorithm 4 Time Enhanced A*

```
Declare and Initialization $heu[DX][DY][DT], dis[DX][DY][DT]$
$ptr[DX][DY][DT][0], ptr[DX][DY][DT][1], ptr[DX][DY][DT][2]$
$O \leftarrow (xi, yi, 0)$ {The open list is initialized with the initial node, and is calculated the value of heuristic function.}
for $O \neq NULL$ do
    Select the node with the lowest value in O, and save it in $(mx, my, t_{\text{step}})$
    if (mx == fx) && (my == fy) then
        break;
    else if ( $t_{\text{step}}$ >= DT) then
        break; {If there are no enough temporal layers}
    else
        $map[mx][my][t_{\text{step}}] = FECHADO$ {Remove from the open list, the point (mx, my, $t_{\text{step}}$)}
        for i=-1 to 2 do
            for j=-1 to 2 do
                {Search in the neighbors nodes}
                if (my+i >= 0) && (mx+j >= 0) && (my+i<DY) && (mx+j<DX) then
                    {If the analyzed point is inside of the map}
                    if $map[mx+j][my+i][t_{\text{step}}+1] \neq PAREDE$ then
                        {If the analyzed point, in the next layer, isn't obstacle}
                        if $heu[mx+j][my+i][t_{\text{step}}+1] == 0$ then
                            {If the node has not been analyzed, calculate the heuristic value}
                            $(mx, my, t_{\text{step}}) \Rightarrow ptr$ {Save the analyzed node as parent node}
                            if (i==0) && (j==0) then
                                $dist[mx+j][my+i][t_{\text{step}}+1] = COST1 + dis[mx][my][t_{\text{step}}]$ {If the AGV stays in the same position}
                            else if (i==0) || (j==0) then
                                $dist[mx+j][my+i][t_{\text{step}}+1] = COST2 + dis[mx][my][t_{\text{step}}]$ {If the AGV follows in the same direction}
                            else
                                $dist[mx+j][my+i][t_{\text{step}}+1] = COST3 + dis[mx][my][t_{\text{step}}]$ {If the AGV follows in the diagonal direction}
                            end if
                            $O \leftarrow (mx+j, my+i, t_{\text{step}}+1)$ {Calculate the respective final cost, given by: $heu[mx+j][my+i][t_{\text{step}}+1] + dis[mx+j][my+i][t_{\text{step}}+1]$}
                        end if
                    else
                        if $dis[mx+j][my+i][t_{\text{step}}+1] > dis[mx][my][t_{\text{step}}] + COST$ then
                            {COST can take the following values: COST1, COST2 or COST3} {If the distance of the new path is lowest than previous path}
                            $O \leftarrow (mx+j, my+i, t_{\text{step}}+1)$ {Inserts the node in the open list, with the new heuristic value}
                        end if
                    end if
                end if
            end for
        end for
    end if
end for
{To fill the map with the path, it's necessary to go through the parent nodes. Starts on the final point and analyzes the parent nodes of each point respectively, until the initial point}
```

References

[1] Roland Siegwart, Illah R LinkNourbakhsh 1970-, and Davide LinkScaramuzza. *Introduction to Autonomous Mobile Robots*. 2nd ed edition, 2011.

[2] Moveit.ros.org. MoveIt! Survey Results. URL: `http://moveit.ros.org/data/surveys/MoveIt!-2013-Survey.pdf`.

[3] Cristian Secchi, Roberto Olmi, Cesare Fantuzzi, and Marco Casarini. TRAFCON- Traffic Control of AGVs in Automatic Warehouses. 94, 2014. URL: `http://link.springer.com/10.1007/978-3-319-02934-4`, `doi:10.1007/978-3-319-02934-4`.

[4] H Choset et. Al. Probabilistic Roadmap Path Planning.

[5] Lydia Kavraki, Petr Svestka, Jean-Claude Latombe, and Mark Overmars. Probabilistic Roadmaps for Path Planning in High Dimensional Configuration Spaces. *Robot. Autom. IEEE*, 1996.

[6] Steven M LaValle. Rapidly-Exploring Random Trees: A New Tool for Path Planning, 1998.

[7] Pedro Luís Cerqueira Gomes da Costa. *Planeamento Cooperativo de tarefas e trajectórias em múltiplos robôs* . PhD thesis, 2011.

[8] Ros.org. ROS: Topics. URL: `http://wiki.ros.org/Topics`.

[9] Siemens. KineoWorks, 2014. URL: `http://www.plm.automation.siemens.com/en_us/products/open/kineo/kineoworks/#lightview-close`.

[10] Ompl. PRMstar Class Reference. URL: `http://ompl.kavrakilab.org/classompl_1_1geometric_1_1PRMstar.html`.

[11] Ioan Sucan, Mark Moll, and Lydia Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, December 2012. URL: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6377468`, `doi:10.1109/MRA.2012.2205651`.

[12] Sertac Karaman and Emilio Frazzoli. Sampling-based Algorithms for Optimal Motion Planning. Technical report.

[13] M Bernardine Dias. *TraderBots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments*. PhD thesis, Robotics Institute, Carnegie Mellon University, 2004. URL: `http://www.ri.cmu.edu/publication_view.html?pub_id=4675`.

[14] Pedro M.Shiroma and Mario F M Campos. CoMutaR: A framework for multi-robot coordination and task allocation, 2009.

[15] Luka Kalinovcic, Tamara Petrovic, Stjepan Bogdan, and Vedran Bobanac. Modified Banker's algorithm for scheduling in multi-AGV systems. *2011 IEEE International Conference on Automation Science and Engineering*, pages 351–356, August 2011. URL: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6042433`, `doi:10.1109/CASE.2011.6042433`.

[16] Aaron Staranowicz and Gian Luca Mariottini. A survey and comparison of commercial and open-source robotic simulator software. *Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments - PETRA '11*, page 1, 2011. URL: `http://dl.acm.org/citation.cfm?doid=2141622.2141689`, `doi:10.1145/2141622.2141689`.

[17] Marcelo Roberto Petry. *A Vision-based Approach Towards Robust Localization for Intelligent Wheelchairs*. PhD thesis, Faculdade de Engenharia da Universidade do Porto, 2013.

[18] Saeid Mokaram, Khairulmizam Samsudin, Abdul Rahman Ramli, and Hamideh Kerdegari. Mobile Robots Communication and Control Framework for USARSim.

[19] Paulo José Cerqueira Gomes da Costa. SimTwo. URL: `http://paginas.fe.up.pt/~paco/wiki/index.php?n=Main.SimTwo`.

[20] Coppeliarobotics. V-Rep. URL: `http://www.coppeliarobotics.com/features.html`.

[21] Aaron Staranowicz and Gian Luca Mariottini. A survey and Comparison of comercial and open-source robotic simulator software. Technical report.

[22] Marcelo Roberto Petry. *A Vision-based Approach Towards Robust Localization for Intelligent Wheelchairs*. PhD thesis.

[23] Pedro Luís Cerqueira Gomes da Costa. *Planeamento Cooperativo de Tarefas e Trajectórias em Múltiplos Robôs*. PhD thesis, Faculdade de Engenharia da Universidade do Porto, 2011.

[24] Ros.org. About ROS. URL: `http://www.ros.org/about-ros/`.

[25] Mario Sousa. Sistemas Concorrentes e Distribuidos. Technical report, FEUP.

[26] Vedran Bobanac and Stjepan Bogdan. Routing and Scheduling in Multi-AGV Systems Based on Dynamic Banker Algorithm . In *16th Mediterranean Conference on Control and Automation*, 2008.

[27] Joana Santos. WebSite Joana Santos, 2014. URL: `http://paginas.fe.up.pt/~ee09133/tese/index.php`.