Universidade do Minho
Escola de Engenharia

João Pedro Sousa Mendes da Cunha

Docking Solutions for Smart Autonomous Mobile Units

Dissertação de Mestrado

Ciclo de Estudos Integrados Conducentes ao Grau de Mestre em Engenharia Eletrónica Industrial e Computadores

Trabalho efetuado sob a orientação do

Doutor Sérgio Monteiro

Outubro 2017

Anexo III

DECLARAÇÃO

Nome: João Pedro Sousa Mendes da Cunha

Endereço eletrónico: joaopedrocunha20@gmail.com

Telefone: 917421429

Cartão do Cidadão: 14500836

Título da dissertação: Docking Solutions for Smart Autonomous Mobile Units

Orientador(es): Sérgio Paulo Carvalho Monteiro

Ano de conclusão: 2017

Designação do Mestrado: Mestrado Integrado em Engenharia Eletrónica Industrial e Computadores

Universidade do Minho, 06 / 12 / 2017

Assinatura:

Acknowledgements

I would first like to thank my dissertation supervisor, Doctor Sérgio Paulo Carvalho Monteiro, for all the guidance and help given. I would also like to express my sincere gratitude for all his patience, motivation and knowledge shared to support this master dissertation work.

I would also like to acknowledge Bosch, for providing the opportunity to allow me to contribute to an important and challenging project. With this I would also like to thank all the people involved in this work, especially the researcher Ricardo Gomes for all his input and availability.

In addition I would also like to thank my parents for all of their support and belief, crucial to keep me motivated until the end. The same goes to my brother, José Cunha, who always listened to my reflections, helping me to see the bigger picture.

To Bárbara Oliveira, a special thank. For putting up with me during the most difficult moments over this last years, for the encouragement she gave me, for all the companionship shared and for always being there for me. Thank you.

Lastly, I would like to thank all my peers, friends and family for all the support and continuous encouragement throughout my academic journey and throughout this final year.

Abstract

Docking solutions are key in the development of smart autonomous vehicles for industrial applications. Throughout the course of every autonomous vehicle workflow inside a factory, there will be moments where parking or picking cargo is needed. As these maneuvers require a precise positioning, without a correct approach, the maneuvers become ineffective jeopardizing all the automation of the process and as result, the maneuver success.

Collaborating with the team of the sub-project P24 "Autonomous Milk-Runs" result of a partnership between University of Minho and Bosch Car Multimedia Portugal S.A., this dissertation aims to conceive and develop docking maneuvers solutions for one of the prototype smart autonomous units, the stacker vehicles.

In Bosch's Braga Plant (BrgP) factory stacker vehicles are required to deliver packagings material and move finished goods within different areas on the warehouse. Stackers will then need to pick the products placed on pallets at the production lines end as the established workflow requires. This area consists in a long and narrow corridor where the stacker vehicles will need to maneuver correctly in order to pick the targeted pallets signalized by logistics.

To develop a docking solution, the study of mobile vehicles kinematics, the development of behavioral based dynamics and the implementation of a pallet detection algorithm was needed to match the factory workflow and requirements. The proposed solution allows the stacker vehicles to respect their workspace constraints, with docking capabilities under multiple circumstances.

Keywords:

Autonomous Navigation, Docking, Mobile Robots, Flexibility, Pallet Detection

Resumo

Soluções de acostagem são importantes no desenvolvimento de veículos autónomos para aplicações industriais. Durante o curso das tarefas de um veículo autónomo dentro de uma fábrica, irão existir momentos em que parqueamento e o levantamento de cargas é necessário. Como estas manobras requerem um posicionamento preciso, se a abordagem não for a mais correta, as mesmas tornam-se obsoletas, colocando em causa toda a automatização do processo e, como consequência, o seu sucesso.

Colaborando com a equipa do sub-projeto P24 "Autonomous Milk-Runs" resultante da parceria entre a Universidade do Minho e a Bosch Car Multimedia Portugal S.A., esta dissertação aponta à conceção e desenvolvimento de soluções para manobras de acoplamento para uma das unidades autónomas inteligentes em análise, os empilhadores.

Na fábrica da Bosch Braga Plant (BrgP), é requerido aos veículos empilhadores a entrega de bens finalizados dentro de diferentes áreas no armazém. Os empilhadores deverão então levantar os produtos colocados em paletes no final da linhas de produção, tal como o fluxo de trabalho estabelecido para estes indica. Esta área consiste em um longo e apertado corredor, onde os veículos empilhadores irão necessitar de manobrar corretamente de modo a levantar as paletes alvo sinalizadas pela logística.

Para a desenvolver uma solução de acoplamento, o estudo da cinemática de veículos móveis, o desenvolvimento de uma dinâmica baseada em comportamento e a implementação de um algoritmo de deteção de paletes foram necessários de modo a cumprir com os requisitos e fluxo de trabalho impostos pela empresa. A solução proposta permite aos veículos empilhadores respeitarem o seu espaço de trabalho, tendo capacidade de acoplamento para circunstâncias múltiplas.

Palavras-Chave:

Navegação Autónoma, Acostagem, Robôs Móveis, Flexibilidade, Deteção de Paletes

Table of Contents

List of Figures

List of Tables

List of Acronyms

AGV Automated Guided Vehicle

API Application Programming Interface

DoF Degrees-of-Freedom

LIDAR Light Detection And Ranging

ROI Region of Interest

ROS Robot Operating System

SAMU Smart Autonomous Mobile Unit

ToF Time-of-Flight

V-REP Virtual Robot Experimentation Platform

Chapter 1

Introduction

1.1 Context

The dissertation project "Docking Solutions for Smart Autonomous Mobile Units (SAMUs)" is part of the sub-project P24 - "Autonomous Milk-Runs" of the project "iFactory", resultant from the partnership between University of Minho and Bosch Car Multimedia Portugal S.A. .

Bosch's multimedia components production for automobiles requires distribution of different raw materials and picking up finished goods from production lines. At the moment, manual operated tugger and stacker vehicles are used to accomplish both processes. The purpose of "Autonomous Milk Runs" project is to develop autonomously driven vehicles (tugger and stacker) that will substitute the human operated ones.

According with the operation parameters for both vehicles, developing autonomous navigation solutions for industrial mobile robots, foresees the existence of moments where each vehicle must undertake precise and specific maneuvers for docking and parking. In case of tugger vehicles, stopping to load raw materials and in case of stacker vehicles, docking to pallets containing finished goods.

With this dissertation project it is intended to develop diverse autonomous docking solutions for one of the prototype SAMUs in P24 sub-project, the stackers. This will imply studying the vehicle kinematics, analyzing and choosing the sensors used in the docking maneuvers and finally developing the according control architectures within a simulation environment.

1.2 Motivation

Considering the growing knowledge in areas such as control, automation and robotics, the inclusion of autonomous and automated robots in industry processes is now considered an asset rather than a investment risk. AGVs (Automated Guided Vehicles) are the most common mobile robots used nowadays in industrial applications. These are preprogrammed vehicles that use markers, wires in the floor, vision, magnets or lasers for navigation. In figure 1.1 an AGV solution is shown. Despite being an effective solution for repetitive tasks inside a factory working process, it suffers when flexibility is required. Rapid change of working environment turn the AGV tasks almost impossible. Implementing an automatic solution also requires an high initial investment when compared with hiring human personnel. Besides the technology costs, the factory infrastructures and processes must also undergo tremendous changes. This disadvantages will not affect an autonomous solution, where adaptations to an ever-changing workspace and factory working processes are possible. With the dissertation project, "Docking Solutions for SAMUs" it is aimed to develop knowledge and experience in autonomous navigations for mobile industrial robots.

Figure 1.1: The Tow Type AGV following a predefined blue path (Mukeshhrs, 2009).

Globally the inclusion of SAMUs in a work space will benefit both workers and the company itself. Safety will be improved since the vehicles will move in controlled and predictable manners with safety sensors for obstacle detection. The same goes for reduced labor costs, damage to products/facility and improved material tracking.

As it was before mentioned, an autonomous navigation solution for mobile industrial robots must provide precise docking maneuvers, allowing the mobile robots to accomplish their tasks. In P24 sub-project those tasks consist in

handling raw materials and finished products, by a tugger and stacker vehicles respectively. The inclusion of SAMUs in a factory workspace will benefit from this dissertation project research and development results.

1.3 Goals

With this dissertation project it is intended to develop flexible, secure and performance improved docking solutions for stacker vehicles. In figure 1.2 a stacker vehicle is shown picking a pallet from a specific docking area (finished goods corridor). Flexibility will allow the vehicles to respect the current workspace layout without the need for significant changes. Security and safety will allow the vehicles to handle materials and products gently ensuring that no damage occurs for materials, infrastructures and operators. Stacker vehicles will also perform smooth maneuvers, easy to predict by human operators ensuring their safety. As it will be presented later, besides the docking maneuvers to pick pallets, stackers will navigate autonomously on several areas on the factory floor, in order to accomplish their defined workflow. This implies the development of a navigation solution that is outside the scope of this dissertation. The solution developed will only address the docking problem.

Figure 1.2: Stacker vehicle picking a pallet from a specific docking area.

To guarantee a proper working solution, some steps must be taken. First the factory layout must be studied in order to understand the workspace limitations for the stacker vehicles. Second the vehicle kinematics must be drawn allowing for a better understanding of it's behavior. Computer vision is a significant field of research to approach next.

Implementing this on the vehicle will allow detection and tracking of targeted pallets when the docking maneuvers occurs. Finally the docking maneuver must be established for several different conditions.

1.4 Dissertation Outline

This dissertation was organized in a series of chapters, which will contribute for a better insight and understanding of the research and development work behind the proposed solution.

- **Chapter 1: Introduction** - First chapter contextualizes and presents the motivation behind this dissertation project and the defined goals to achieve.
- **Chapter 2: State of the Art** - Second chapter explores the state of the art, where developed solutions related to this project are reviewed and analyzed.
- **Chapter 3: Theoretical Background** - Third chapter provides the theoretical basis behind the project, helping the reader to understand the elements applied in the project development.
- **Chapter 4: Requirements and System Overview** - Fourth chapter details the workspace constraints and the workflow requirements of the factory. According with those requirements, the resulting system architecture is described, as well as the required tools and hardware for this dissertation project.
- **Chapter 5: Analysis and Proposed Solution** - In the fifth chapter, a docking solution is proposed according to the analysis of system constraints and requirements.
- **Chapter 6: Development and Implementation** - The solution implementation is presented in the sixth chapter, according with all previously performed analysis.
- **Chapter 7: Results and Discussion** - Seventh chapter presents the results obtained from this project, as well as their review and discussion.
- **Chapter 8: Concluding Remarks and Further Research** - The main conclusions drawn from this project are presented in the last chapter. Finishing this eighth and last chapter, the possible next steps for the development of this project are also presented.

Chapter 2

State of the Art

Considering the goals before stated for this dissertation context, in this chapter the research results for existing projects are described. For this state of the art survey docking related work will be the main focus, from the control standpoint to pallet detection and tracking. This last proved to be the most relevant for this project development. Navigation is also approached in this chapter briefly considering the relevance for this project.

2.1 Docking Maneuvers

As Bostelman and Hong (2016) mentioned, docking maneuvers is a matter of research on several areas since 1950 when the inception of the automated vehicle occurred. The first forms of industrial mobile robots were based on electrical wires on the floor following navigation. This type of solution is not flexible since any adaptation for a new warehouse change becomes costly. Technology has come very far from those solutions and nowadays computers are a common navigation component for robots.

The first form of AGV navigation was based on following electrical wires in the floor, and according to Bostelman and Hong (2016) some vehicles still operate using this mode. The problem is that induction wire-guided vehicles do not have the flexibility to adapt transport routes to changes in the production process. Mellado et al. (1999) developed concepts to overcome these deficits. The project included a consortium that developed an on-board, AGV operating-system as a general purpose, real-time, knowledge-based control system. The control and sensor data processing were based on fuzzy logic control and real-time expert systems. Fuzzy logic cannot be expressed as ”true” or ”false”,

Table 2.1: Comparison of controllers (Villagra and Herrero-Pérez, 2012).

Controller	**St. Ang. FFT median (Hz)**	**Tracking Error (m)**		
Fuzzy (3cm)	0.4935	0.0068	0.0164	0.0085
Fuzzy (5cm)	0.4631	0.0081	0.0160	0.0108
Fuzzy (10cm)	0.1902	0.0127	0.0219	0.0004
Vect. pursuit (3cm)	0.1594	0.0025	0.0171	0.0050
Vect. pursuit (5cm)	0.0892	0.0036	0.0162	0.0097
Vect. pursuit (10cm)	0.0386	0.0036	0.0167	0.0007
Flatness	0.0137	0.0039	0.0164	0.0022

hence the term fuzzy given by the ”partially true” logic. Both fuzzy logic control and real-time expert systems enabled a vehicle to deal autonomously with the typical uncertainties of an industrial working environment. The approach provided good navigation, collision avoidance, and target docking capabilities, based on measurements by cheaper, less accurate sensors.

Villagra and Herrero-Pérez (2012) proposed a control architecture focused on robustness, performance and on ease of controller configuration for AGV path tracking. The work focus on the control law that allows AGVs to operate by tracking a predefined route with industrial grade of accuracy, repeatability and reliability. Fuzzy and vector pursuit control techniques were applied to a new nonlinear control strategy. The simulations (table 2.1) presented very interesting results with no need of physical parameter knowledge and providing at same time high efficiency for AGV path-tracking in industrial environments. The results presented have been obtained with noisy measurements from a pure kinematic model. In this table all the different control configuration used were compared to flatness based control in terms of tracking error (mean, max -imum and final values) and softness of control actions (median value of the steering angle FFT). As the table 2.1 shows fuzzy controllers are significantly less precise and have a more abrupt behavior than the other control approaches. The tracking error obtained with vector pursuit control largely fulfills the before mentioned constraints and it acts in a softer way than fuzzy control. Flatness-based controller provides a steering angle FFT median three times lower than the best fuzzy and vector pursuit implementations.

Lucas et al. (2005) proposed a soft-computing technique based on a “multi-objective, evolutionary algorithm using multiple fuzzy logic controllers” that aims to improve duration, accuracy, and stability features of precise docking tasks

for a forklift that interacted with objects.

The solution results show that orientation and position errors are reduced when comparing the performance between the evolutionary learned controller and the initial controller. Steering angle has also lower variation in the evolved controller, reducing control effort and avoiding abrupt velocity changes to achieve stable maneuvers. This results satisfied imposed constraints for docking tasks.

Defining the position in which the autonomous vehicle starts performing its docking motion is critical to develop an efficient solution. For the stacker vehicle, Herrero et al. (2013) developed an automatic configuration consisting in an unconstrained optimization method coupled with probabilistic techniques of initial vehicle waypoint for robotic forklifts. As the simulation results show in figure 2.1, this solution increases flexibility and quick adaptability for the vehicles compared with manually waypoint configuration by a human operator. Docking stations can be reached from different orientations, so there is a finite range of possible orientations to initiate docking maneuvers. A human operator configured the initial orientations using a trial and error approach, finding a tradeoff between proximity and repeatability. Distant waypoints usually generate robust but long trajectories, while close waypoints can generate different routes. This can compromise safety of load transfer operations. In figure 2.1 it was chosen a waypoint too close to the docking station. The trajectories generated seem robust and repeatable after many tests, but when perturbations are added then the manually selected waypoints are not suitable. A second experiment shows the robotic forklifts transporting two empty pallets to the storage system. First they navigate from initial location to the corresponding stations. A navigation conflict occurs when they reach the feeding storage area. This is resolved with a decentralized traffic approach. When they are near enough to the docking station, a waypoint is selected following a minimum angle difference criteria between the orientation of the vehicle and the orientation of the station. It is important to mention that this concept of navigation conflict is not an objective for this dissertation.

2.2 Sensing

In order to develop a proper docking solution there must be a combination between sensors and intelligent vehicle control.

Baglivo et al. (2011) used a combined double-sensor architecture, laser and camera, and a new algorithm named RLPF (Range and Look Pallet Finder) to solve the problem of identifying and localizing a pallet, where the *a priori* knowledge of position and angle are usually denoted by large uncertainty. RLPF algorithm calculates pallet pose by

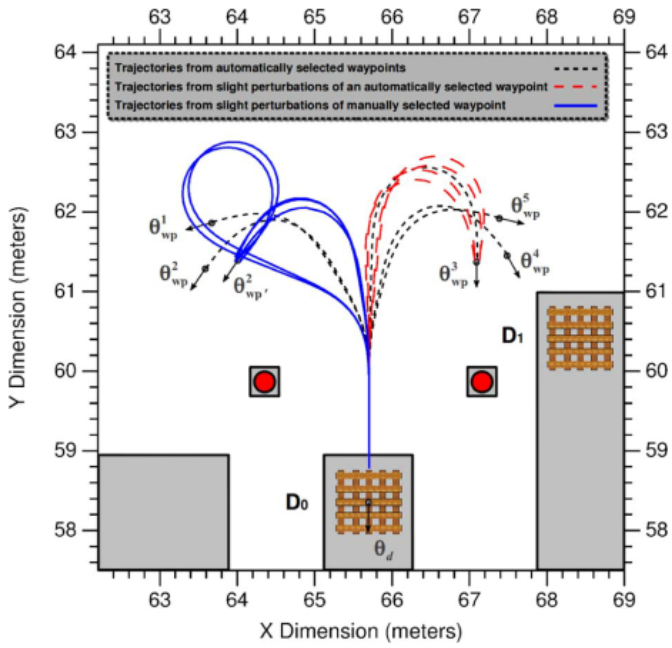

Figure 2.1: Waypoints automatically configured for five possible initial orientations and manually waypoint configuration by human operator (Herrero et al., 2013).

considering the following parameters: distance from the camera, face angle, supporting blocks width and color. The pallet pose computed from the initial position was taken as reference from which to calculate four steps of displacements and rotations. The errors were also computed. Their covariance, together with mean error, was assumed as RLPF uncertainty. In table 2.2 the results are shown in terms of the standard deviation for two pallet dimensions and two colors in normal size.

The results prove this solution can be implemented in industrial autonomous vehicles. The tests provided a miss detection rate of about 1% and zero wrong detection in about 300 trials. Still, Baglivo et al. (2011) has not developed a autonoumous docking maneuver but proposes a nonlinear control as a solution.

Nygards et al. (2000) provided a solution to dock a AGV or mobile robot to a pallet using a combination of a range camera, a video camera and a laser. In it's research, Nygards et al. (2000) mentions that errors in cross range position are difficult to reduce since the pallet orientation is unknown for the robot until it is very close to the pallet. For finding the pallet, first a ROI is de□ned based on prediction (figure 2.2), then the distance and orientation is calculated using linear regression. Finding the intervals where the laser-intensity peaks makes it possible to determine the middle block

Table 2.2: RLPF experimental uncertainty (Baglivo et al., 2011).

	X (mm)			Y (mm)			Z (mm)		
	s_x	e_{mean}	e_{max}	s_y	e_{mean}	e_{max}	s_θ	e_{mean}	e_{max}
Standard max format	3.9	-0.7	-10.4	2.2	1.2	6.7	0.3	0.006	-0.78
Normal wood	3.0	-2.0	-9.2	2.9	0.8	5.9	0.28	0.10	0.76
Normal blue	2.1	-0.9	4.9	3.0	1.4	7.4	0.27	0.12	0.67

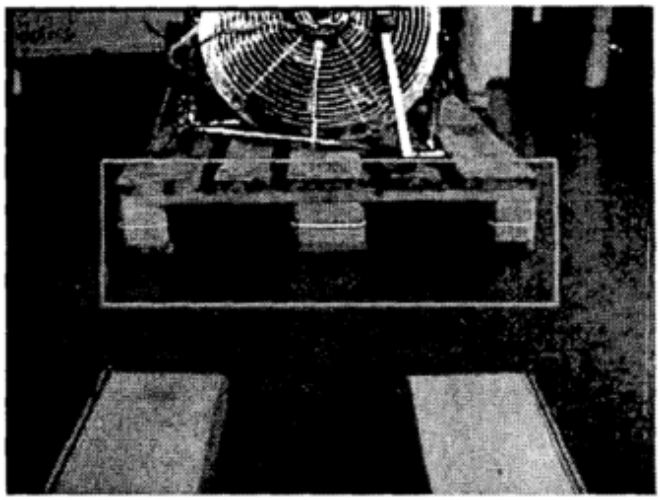

Figure 2.2: Using the prediction from previous images the laser line is searched for within a ROI (Nygards et al., 2000).

of the targeted pallet and then the midpoint, which is the center of the middle block front face. Despite of this system being feasible and self-monitoring as the author mentions, the integration between the control system on board of the robot and the sensor was weak, since the resolution of the range camera is strongly distance-dependent.

Providing a more efficient and robust solution, Jun et al. (2008) developed an intelligent monocular vision-based AGV system. Using Arabic numerals as docking landmarks, high-performance algorithms were designed to allow their detection.

In a different approach, Lecking et al. (2006) presents a solution to automate forklift trucks using two laser scanners, to locate and pick-up pallets. Contrary to camera based solutions, this approach provides luminance condition independence, what proves to be a great advantage. Whereas one approach uses pallets modified with adhesive reflectors which enables very flexible handling of pallets, the second approach detects pallets based on geometric

Figure 2.3: Euro pallet modified with reflectors (Lecking et al., 2006).

characteristics without any modifications to the environment. In these experiments the sensors employed were: the SICK 2D laser scanner S3000 to recognize pallets at ground plane and SICK 2D laser scanner LMS200 to recognize pallets at variable heights. As mentioned before, two solutions were developed. One using reflectors to help locate the pallets in the industrial environments (figure 2.3). The reflector marks emit energy beams that are computed by the laser scanner. This solution operated for 5 days (8 hours a day) without any failure.

Since it is not possible to place reflector marks in all fields of activity, another solution was developed. This one used geometrical information to recognize pallets. The Iterative Closest Point (ICP) algorithm (Besl and McKay, 1992) was used. This is an iterative algorithm that matches two scans in a point-to-point manner. In this task instead of two scans being registered for the ICP algorithm an actual 2D scan S of the environment is registered with a model of the pallet. Thereby the position and orientation of the pallet is discovered in an actual scan. The number of scan points is reduced in a first working phase, followed by the help of an expectation of the pallet position.

Results (table 2.3) show that the discrepancy between the actual and the calculated pallet position has an average of 5 mm in x-axis, 10 mm in y-axis and 0.5° of orientation error with a standard deviation of 5 mm, 12 mm and 0.3°.

Using a laser range-finding sensor, Lee et al. (2014) tried to find the goal position for docking and to segment the docking station legs. The segmentation result and the experimental docking in a Korean facility is shown in figure 2.4. This is not a generic solution since it was developed for AGVs transporting products for medium/small scale factory automation systems.

Table 2.3: Geometrical localization results. Actual coordinates (x_R,y_R,ϕ_R), first estimation of coordinates (x_E,y_E,ϕ_E) and calculated coordinates (x_C,y_C,ϕ_C) (Lecking et al., 2006).

x_R mm	y_R mm	ϕ_R °	x_E mm	y_E mm	ϕ_E °	x_C mm	y_C mm	ϕ_C °
2000	0	0	2017	142	7.1	2001	-18	0.7
2000	0	0	1993	147	-1.8	1997	16	-0.6
2000	0	0	1940	93	14.4	1999	30	-0.7
2000	0	0	1900	93	15.9	2001	30	-0.7
2000	0	0	2180	80	4.3	1992	6	0.1
2000	0	50	1875	80	41.0	1991	2	49.5
2000	0	50	2022	-173	43.4	1987	-5	49.0
2000	0	50	1795	-133	50.8	1998	-3	50.1
2000	0	50	1795	13	45.9	1988	3	49.4
2000	0	50	2115	67	49.6	2002	2	50.2

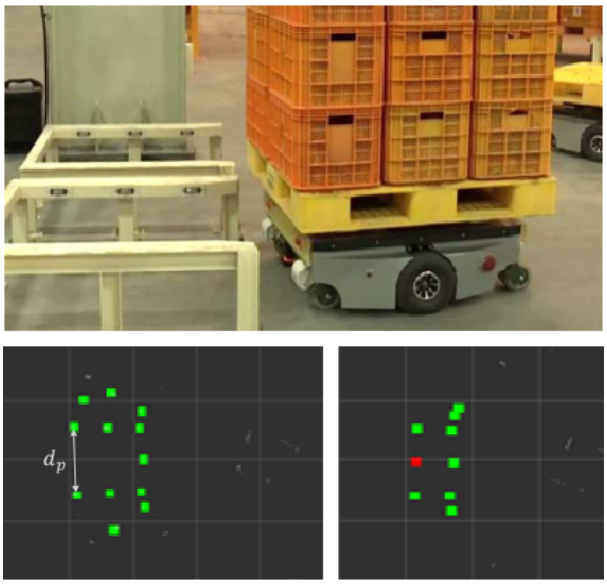

Figure 2.4: Above the experimental docking and bellow the result for docking station segmentation. (Lee et al., 2014)

Figure 2.5: Prototype forklift engaging a pallet during experimental tests (Seelinger and Yoder, 2006).

Presenting relevant experimental results, Vaz et al. (1997) showed a sensor-based docking strategy for a non-holonomic, mobile platform used to support material handling operations in industrial-like environments. Using a low-cost infrared sensor system to locate the mobile platform relative to the docking station, Vaz et al. (1997) was able to dock with an accuracy of approximately 0.5 cm in x and y, and 1.0° in θ.

As a more relevant case study solution, Seelinger and Yoder (2006) developed a prototype system for the vision-guided automatic pallet docking by a computer controlled forklift. A forklift prototype was created, which consisted in a three-wheeled electric lift truck, housing a computer system running Windows NT 4.0 and a 450 MHz Cleron processor. For image acquisition the prototype incorporates two Sony XC-75 cameras with wide-angle lenses control using two frame grabbers with 640x480 pixel resolution each. The solution is based entirely in the MCSM ('Mobile Camera-Space Manipulation') controller. This controller is an extension for the CSM method, which was restricted by the stationary position of the vision sensors relative to the mobile manipulator system. The method was developed originally for the visual guidance of a planetary exploration rover equipped with a robotic arm (Seelinger et al., 2002) and has been adapted and further developed to control the forklift prototype system. In this project, first a solution based on features (also called fiducials) placed on the pallets was created. Each feature was placed on the pallet side column and mid-point. Then another solution based on pallet natural features such as their geometry was also developed.

Two types of tests were carried out, one with the features placed on the pallet and the other using only the pallet natural features. For the first case, 100 test runs were carried out with different starting position for the pallet. Images showing the protoype forklift performing a typical pallet engagement experiment with the features placed on the pallet is shown in figure 2.5. 98 of those 100 tests were successful, and in 20% of those, the forklift was unable to dock at first attempt. For the second case 11 in 13 trials were successful. These tests were carried out without obstructions or other pallets in the surrounding so the system can identify the targeted pallet. Another negative aspect of this project is the pallet placement for docking. Pallets are docked sideways, so the pockets where the forks engage increase in area allowing for a much easier docking maneuver.

Continuing the sensor and control fusion for pallet detection in this state of art survey, Varga and Nedevschi (2014) proposes a solution using two main sources of visual information, 2D intensity images and 3D stereo cameras and the selection of an ensemble classifier with 100 boosted decision trees classifier . In this project intensity images were used to provide information about the location of the pallet, where the position and orientation is obtained using the 3D stereo cameras. For pallet detection, first a gaussian kernel and a histogram equalization is performed to restrict the region to analyze in the center. Using the image gradient through a Sobel filter, the candidate is found for the classifier to determine if this candidate is indeed a pallet. This classifier was trained only on a manually labeled dataset with pallets as positive examples and other regions as negative examples.

To evaluate pallet detection, the detection rate and false positive rate of different classifiers on the acquired datasets are evaluated. Scoring is based on the intersection and the union of the two rectangles. A strong positive match corresponds to a high absolute overlap in the x direction, and a high relative overlap in the y direction. A weak positive match corresponds to a moderate/high absolute overlap in the x direction, and a moderate relative overlap in the y direction. strong false positive - corresponds to a low overall overlap, and a weak false positive corresponds to the rest. The results show a detection rate for weak positive of 94% (5007 out of 5333), for strong positives 845 (4476 out of 5333), and for false positives 1.5% (80 out of 53333). In figure 2.6 two successful detections are shown with the 3D point and orientation angle indicated for each pallet. More tests were performed to evaluate if the selected classifier was suitable for the detection task. The results are shown in table 2.4.

Another relevant project to be considered is ROBOLIFT,"a Vision Guided Autonomous Fork-Lift for Pallet Handling" from Garibott and Masciangelo (1996). This project, resultant from the partnership between Elsag Bailey Telerobot and Fiat OM Carrelli Elevatori SPA, consists in the creation of a robotic mobile forklift, that performs autonomous missions as well as allows a human operator to drive it. In this project, and considering the subject of this master dissertation,

Figure 2.6: Successful detections in difficult cases. 3D point and orientation angle is indicated for each pallet (Varga and Nedevschi, 2014).

Table 2.4: Classifier evaluation results. DT-Decision Trees (Varga and Nedevschi, 2014).

Classifier type	Detection rate [%]	False positive rate [%]	Execution time [ms]
Boosted 100 DTs	99.81	0.18	3140
Cascaded 1000 DTs	99.07	0.19	140
Boosted 100 DTs	98.69	0.74	180
Linear SVM	58.69	39.25	55000
Manual Decision Stumps	51.40	37.57	4

the most important feature is the technology behind pallet detection and docking. ROBOLIFT uses a camera rigidly connected directly on it's forks. Then using an implemented vision algorithm based on the identification of pallets central pockets it computes the 3D location of the pallet. A prediction verification paradigm was also implemented, which consists on projecting to the image the predicted geometry of the pallet model. This is followed by a controlled region growing process, which aims to propagate the grey level up to the border, with constraints on the expected size and shape. This process is illustrated in figure 2.7. To ease docking maneuvers this vehicles forks are able to shift left or right without needing the vehicle to move. To detect a correct loading operation, a microswitch is used, between the forks, to sense the contact of the pallet and the fork's support.

In Pagès et al. (2001) a similar system was implemented. This system allows the robotic vehicle to compute the 3D position and orientation of the pallet and generate vehicle trajectory to fork it. As some of the other presented projects this project is aimed for forklifts operating in industrial environments. In this project pallets are only picked from the floor. The proposed vision system uses an image segmentation method based on color and geometric characteristics of the pallet. For this segmentation to work properly, a controlled environment is assumed. That means the warehouse walls and floor should have adapted colors in order to ease pallet segmentation based on color. The result for this

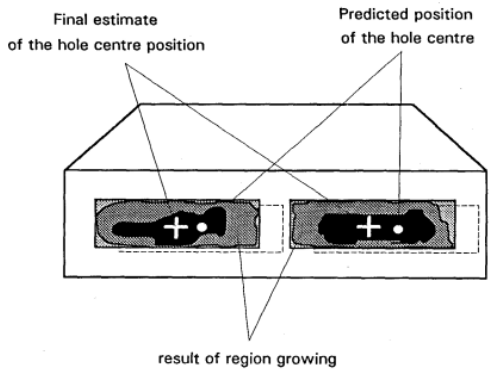

Figure 2.7: Model based vision and 3D recognition and positioning of the pallet (Garibott and Masciangelo, 1996).

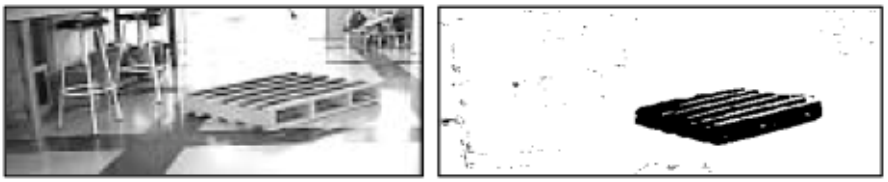

Figure 2.8: Model based vision and 3D recognition and positioning of the pallet (Pagès et al., 2001).

color segmentation can be seen is figure 2.8.

The system then proceeds to detect the pallet vertexes and then segments the sides of the pallet which are in contact with the floor. Hough transform was used to allow detecting such features. The transform used was based on the algorithm introduced by Pratt (2001). To obtain the 3D real position and orientation in the reference frame of the pallet, a method chosen was a transformation algorithm to get 3D coordinates of 2D points from a single image. This method demanded the use of a calibration algorithm to calculate the relationship between the 2D points and the corresponding 3D optical rays. For this, the Coplanar Tsai calibration method was used (Tsai, 1987). Then a trajectory is calculated so the forklift can dock the pallet. This is not a continuous feedback system, and so the trajectory is never adapted in case there are environment changes. The results show a discrepancy between real and calculated point has an average of 31.5mm along x-axis and 12.5 mm along the y-axis, with a standard deviation of 21.7 mm and 10.1 mm for cases where the loss of pallet geometry in segmentation is low. If the segmentation process erodes the pallet edges, the forking side identification decreases considerably. However, this occurs only when the pallet is far from the vehicle, so Pagès et al. (2001) also points out that the larger the distance between the vehicle and the pallet, the higher is the error between the predicted position and the real position of the pallet.

Karaman et al. (2010) presents a system able to accomplish docking maneuvers to pallets on arbitrary positions, whether is to engage pallets on the ground or on a platform. An estimation framework is presented based on individual 2D LIDAR scans. To detect the pallets one Hokuyo UTM laser range finder is used. This has 30 m range a 140° field-of-view. To detect the platform in which the pallets will be placed, a pair of UTM laser range finders mounted to the left and right sides of the carriage assembly are employed These laser scanners have also a 30 m range but an improved field-of-view of 270°. Some part of this vehicle tasks are remotely controlled. A user has to point what pallets the vehicle should pick and where the vehicle should place it.

Chapter 3

Theoretical Background

In this chapter the theoretical fundamentals behind the development of this dissertation project are presented. According with the solution developed for maneuvering the stacker vehicle during the docking maneuver, different topics will be approached, from control related aspects to sensing and vision principles.

3.1 Behavior Based Robotics

First introduced by Schöner et al. (1995), and providing vehicles an autonomous navigation behavior, attractors dynamics were part of the control solution developed in this dissertation project. The dynamics implemented in this project was based in Bicho (1999). Autonomous vehicles or robots, have no prior knowledge of the work environment, relying only on sensor information. Based on the information retrieved a decision making process will occur in order to adapt the vehicles navigation to the real time constraints. In order to accomplish this step, a control technique must be employed to ensure the best navigation route and smooth behavior.

Behavioral variables describe the particular behavior of a system. The value at a given time of a behavioral variable describe their present state. Choosing correct variables is the first step to develop an behavior based dynamics solution, where assigning values for those variables is the last step, or in other words behavior generation for the described system.

For the particular case studied in this dissertation project, choosing behavioral variables will have into account the matter of research, autonomous vehicles. As Schöner et al. (1995) refers, the dynamic approach for navigation

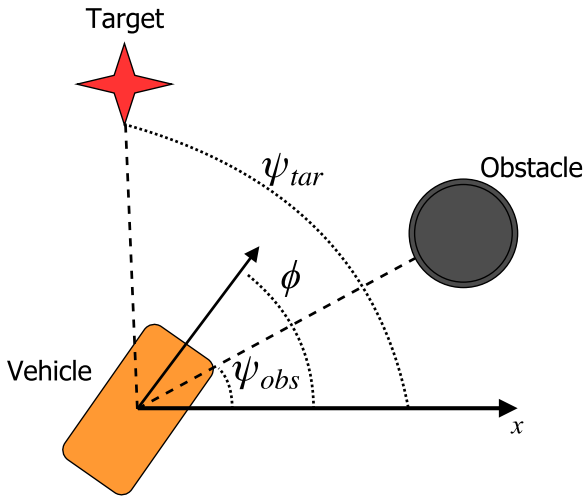

Figure 3.1: Constraints for the dynamics of ϕ are the directions at which obstacles and target lie from the current position of the vehicle, i.e. directions ψ_{obs} and ψ_{tar} (Adapted from Bicho (1999)).

uses the heading direction relative to the world axis (ϕ) as a suitable planning variable. By manipulating this variable it is possible to induce the intended navigational behavior. Another important behavioral variable besides the heading direction, is the traveling velocity of the vehicle (v). For the navigational purpose, a vehicle must shift to a given point, called target, and at the same time avoid obstacles. Considering this two cases it is possible to define two system constraints, being $\phi = \psi_{tar}$ the direction where the target is placed and $\phi = \psi_{obs}$ the direction where an obstacle lies from the vehicle's view point. In figure 3.1 this case scenario is illustrated.

As mentioned before, the second step for achieving a behavior based dynamics solution for a vehicle is generating values through out time, for the defined behavioral variables ϕ and v, in order to control the vehicle's direction. The values to be assigned are generated as solutions of a dynamical system described by the equations 3.1 and 3.2.

$$\dot{\phi} = f(\phi(t), parameters) \tag{3.1}$$

$$\dot{v} = g(v(t), parameters) \tag{3.2}$$

As Schöner et al. (1995) states, $\dot{\phi}$ is a dynamical system which describes the change rate of the heading direction as a function of current heading direction ϕ. The same applies to $\dot{v}$. This dynamical systems define a vector at each point in phase space, that determines the direction rate in which the system will move from each point. Together,

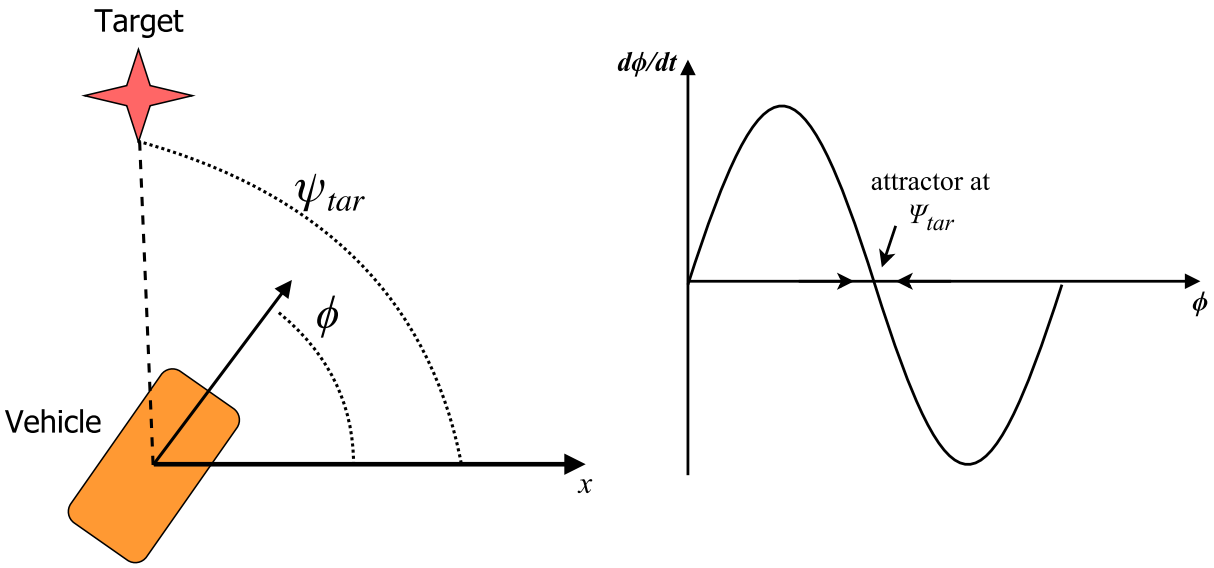

Figure 3.2: At right the placed target in the direction $\phi = \psi_{tar}$ and at left the respective plot representing the change rate $\frac{d\phi}{dt}$ (Adapted from Bicho (1999)).

this vectors will produce a vector field. For the navigational purpose the solution of dynamical systems to focus is called fixed points. A fixed point implies zero change rate of a variable, meaning the vector field at this point is null, $\dot{\phi}_{fixedpoint} = f(\phi_{fixedpoint}) = 0$. If the system converges in time to the fixed point it is called asymptotically stable. This points are called attractors. If the contrary is verified, in other words, if the system diverges in time to the fixed point from points nearby, the fixed point is a repeller. Both fixed points can be applied in the navigational purpose accordingly. An obstacle can be represented by a repulsive force and a target can be represented by an attractive force. These concept is illustrated in figures 3.2 and 3.3. Fixed points are the intersections of the function which describes the change rate of heading direction $\dot{\phi} = \frac{d\phi}{dt}$. The system will then be attracted to the desired value of the heading direction where a target will be placed, and will avoid the obstacles direction.

The behavior generation is then parted in obstacle avoidance and target orientation. Both behaviors will describe two different dynamics where one will create a repulsive force-let around the obstacles to avoid and the other will create an attractive force-let around the goal position represented by targets.

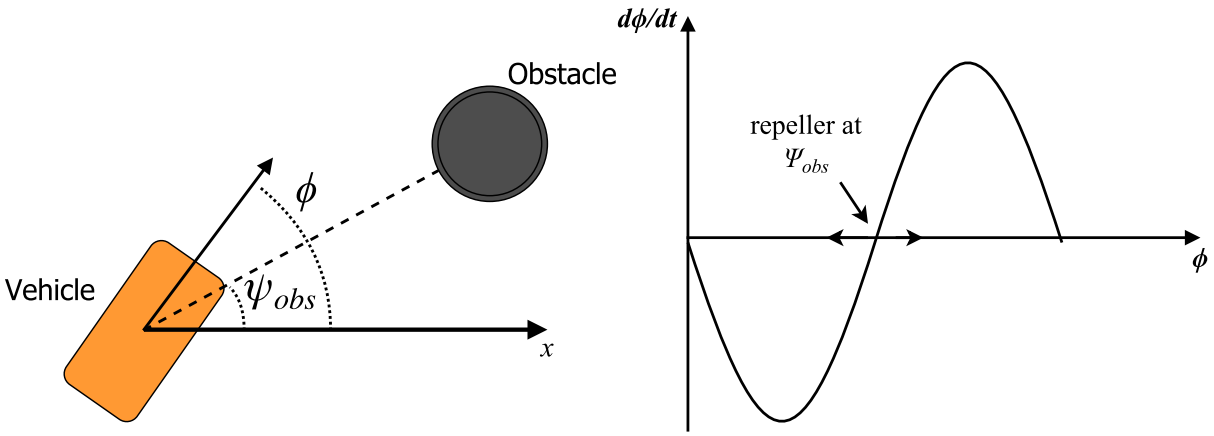

Figure 3.3: At right the placed obstacle in the direction $\phi = \psi_{obs}$ and at left the respective plot representing the change rate $\frac{d\phi}{dt}$ (Adapted from Bicho (1999)).

3.1.1 Target Orientation

For this dynamics, target absolute coordinates are assumed to be known externally, information that can also be retrieved by the vehicle's own sensory information. This coordinates are described by (X_{target},Y_{target}), and ($X_{vehicle}$,$Y_{vehicle}$), which describe the vehicle's own position estimation. The target direction, ψ_{tar}, according with the world axis is described by:

$$\psi_{tar} = tan^{-1}\left(\frac{Y_{target} - Y_{vehicle}}{X_{target} - X_{vehicle}}\right) \tag{3.3}$$

An attractor is placed in the direction $\phi = \psi_{tar}$. The range of values for the attractive effect of the fixed point must be the full circle (from 0 to 2πrad), so that the target acquisition dynamics happens regardless of the vehicle's orientation. With this considered, the mathematical which represents this attractive force-let is:

$$\frac{d\phi}{dt} = f_{tar}(\phi) = -\lambda_{tar} sin(\phi - \psi_{tar}) \tag{3.4}$$

In equation 3.4, λ_{tar} value manipulation will influence the strength of the attraction towards the target.

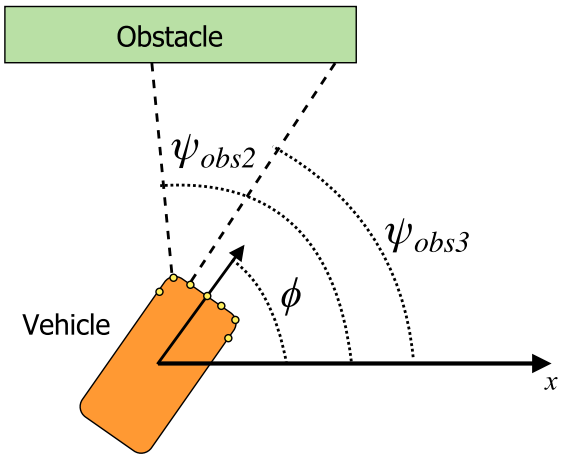

Figure 3.4: Two virtual obstacles are detected in the directions ψ_2 and ψ_3 (Adapted from Bicho (1999)).

3.1.2 Obstacle Avoidance

In this subsection the vehicle is assumed to have 7 infra-red sensors i mounted on the front (in Chapter 5 a more detailed explanation will be added, regarding sensor information on the stacker vehicle). Each sensor is looking in a direction $\psi_i = \phi + \theta_i$, as figure 3.4 illustrates. The strategy adopted consists in creating repulsive force-lets centered in the direction where a sensor i identifies a virtual obstacle ψ_i (equation 3.5).

$$f_{obs,i}(\phi) = \lambda_i(\phi - \psi_i)e^{\frac{-(\phi-\psi_i)^2}{2\sigma_i^2}}, \; i = 1, 2, ..., 7 \tag{3.5}$$

λ_i represents the strength of repulsion for a given direction ψ_i. This parameter is inversely proportional to the measured distance d_i, as equation 3.6 assures. Tunning parameters β_1 and β_2 control the maximum strength of repulsion for this direction and the rate of decay with increasing distance respectively.

$$\lambda_i = \beta_1 e^{-\frac{d_i}{\beta_2}} \tag{3.6}$$

The parameter σ_i determines the angular range of the repulsive effect that depends on the sensor sector $\delta\theta$ and also the distance d_i, because as Bicho (1999) states, the angle implied by half the vehicle at the sensed distance is added on each side of the sensor sector as a safety margin (equation 3.7). This is illustrated in figure 3.5.

$$\sigma_i = tan^{-1}\left[tan\left(\frac{\delta\theta}{2}\right) + \frac{B}{L + d_i}\right] \tag{3.7}$$

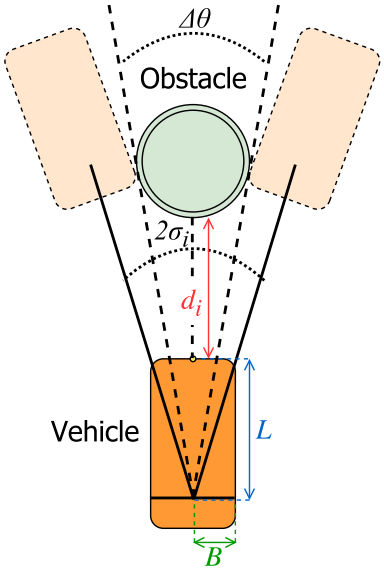

Figure 3.5: The range of the repulsive force-let is limited based on the sensor range and on the constraint of passing next to the virtual obstacle without contact (Adapted from Bicho (1999)).

All sensor contributions are summed, so the dynamics for obstacle avoidance results in:

$$\frac{d\phi}{dt} = F_{obs}(\phi) = \sum_{i=1}^{7} f_{obs,i}(\phi) \tag{3.8}$$

3.1.3 Merging Behaviors

By summing both behaviors before mentioned, target orientation and obstacle avoidance, is possible to merge them:

$$\frac{d\phi}{dt} = F_{obs}(\phi) + f_{tar}(\phi) \tag{3.9}$$

At this moment, the vehicle navigational behavior is affected by repellers placed on $\phi = \psi_{obs,i}$ and by attractors placed on $\phi = \psi_{tar}$ where the target lies. In figure 3.6 a situation where both dynamics affect the vehicle's behavior is illustrated. In this situation is shown the most challenging case scenario possible for obstacle avoidance, where the target orientation dynamics create an attractor in between the obstacles and the obstacle avoidance dynamics creates a repeller for the same orientation.

In this situations, stability of fixed points for the heading direction changes due to bifurcations of the vector field. From here, the heading direction can be stuck on a repeller. Therefore an external force-let must be added to the

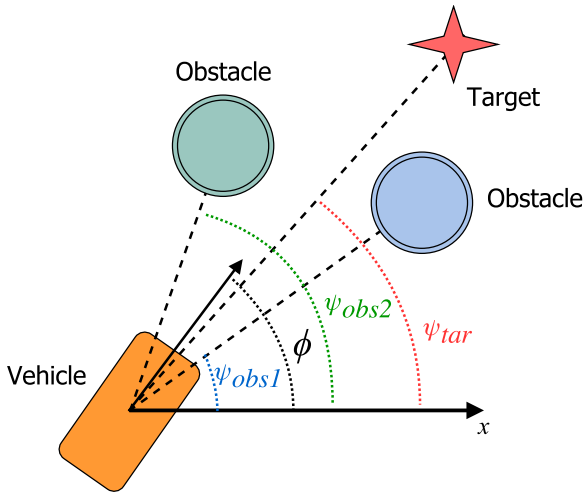

Figure 3.6: The distance between both obstacles is not sufficient for the vehicle to pass through, and the target lies in the direction pointing in between the two obstacles thus defying the obstacle avoidance behavior (Adapted from Bicho (1999)).

planning dynamics, so that the vehicle can escape repellers. This is done by adding a stochastic force with a preset Gaussian white noise,ξ_n, of unit variance, and Q the effective force variance. The noise does not affect the normal system navigation behavioral due to the asymptotically stable states generated for it. Bellow, the system full dynamics is shown, as well as the detailed stochastic force to be added:

$$f_{stoch} = \sqrt{Q}\xi_n \tag{3.10}$$

$$\frac{d\phi}{dt} = F_{obs}(\phi) + f_{tar}(\phi) + f_{stoch} \tag{3.11}$$

3.1.4 Traveling Velocity Control

As Bicho (1999) states, in order to keep the system stable, and considering the frequent changes to the system (attractors and repellers position), traveling velocity, v, must be controlled. The velocity which the fixed points shift is determined by the vehicle's relative velocity in respect to the world environment, thus turning critical the control of

traveling velocity.

$$\frac{dv}{dt} = g(v) = -c_{obs}(v - v_{obs}) - c_{tar}(v - v_{des}) \tag{3.12}$$

Using the dynamic system presented in equation 3.12 is possible to control v. The navigational speed in which the vehicle will travel with no obstacle around will be dictated by v_{des}. When obstacles emerge the navigational velocity is given by v_{obs}, that is described by equation 3.13:

$$v_{obs} = \frac{d_{mins}}{T_{obs}} \tag{3.13}$$

T_{obs} determines the time to contact between the vehicle and the detected obstacle, and d_{min} determines the minimum distance returned by sensors to read the obstacle to avoid.

Still in equation 3.12, the parameters c_{obs} and c_{tar} represent the contributions of obstacles or targets to the overall vehicle velocity. Appropriate tunning of the parameters will make obstacles prevail over targets. For this using a potential function of the obstacle dynamics, resulting from a integrate of obstacle force-lets is possible to conceive a systematic way to indicate if obstacles are present (equation 3.14).

$$U(\phi) = \sum_{i=1}^{7} \left[\lambda_i \sigma_i^2 e^{\left(\frac{-(\phi - \psi_i)^2}{2\sigma_i^2} \right)} - \frac{\lambda_i \sigma_i^2}{\sqrt{e}} \right] \tag{3.14}$$

Positive values indicate a repulsion zone of sufficient strength, and in the other hand, negative values indicate a weak or non-existence repulsion zone. Using the sinusoidal threshold function 3.15 which ranges from $-\frac{1}{2}$ to $\frac{1}{2}$ this potential values are transformed.

$$\alpha(\phi) = tan^{-1}\left(\frac{cU(\phi)}{\pi} \right) \tag{3.15}$$

Finally both parameters c_{obs} and c_{tar} can be written according with equation 3.16 and 3.17.

$$c_{obs} = c_{v,obs}\left(\frac{1}{2} + \alpha(\phi) \right) \tag{3.16}$$

$$c_{tar} = c_{v,tar}\left(\frac{1}{2} - \alpha(\phi) \right) \tag{3.17}$$

Parameters tunning is key to properly create a navigational behavior that moves towards a given target but first gives priority to the obstacle avoidance behavior. Bicho (1999) states the hierarchy in equation 3.18 must be respected.

$$\lambda_{tar} << c_{v,tar},\ \lambda_{obs} << c_{v,obs},\ \lambda_{tar} << \lambda_{obs} \tag{3.18}$$

3.2 Kinematics

A branch of classical mechanics, kinematics describes the motion of points, bodies and system of bodies without considering the acting masses or forces on them. In astrophysics is used to describe the motion of celestial bodies and systems. For mechanical engineering, or mechanics is used to describe the motion of systems composed of joined parts, such as an engine or the human body skeleton. The same goes for robotics, with the example of robotic arms or mobile robots. According to Craig (2004), kinematics science studies the position, the velocity, the acceleration, and all higher order derivatives of the position variables with respect to time or any other variable(s). Knowing this, it must be concluded that for this dissertation project, kinematic modeling is crucial, considering the need to control the vehicles navigational course and movement during the docking maneuver.

Understanding the relation between the actuators to control and resulting position for the vehicle is the goal for the kinematic study. As mentioned by Correll (2011) the process is straightforward for manipulator arms, where the position/angle of each joint can be used to calculate through trigonometry the position of it's end-effector, or the manipulator's device designed to interact with the environment. This is known as the Forward Kinematics process. If we invert this relationship, it is possible to know how the joints must be posed in order to have the end-effector in a certain goal position. This is the Inverse Kinematics process.

Correll (2011) continues mentioning that every robot assumes a position in the real world that can be described by its position (x, y and z) and orientation (pitch, yaw and roll) along the three major axes of the Cartesian space, also named task space or operational space. The Degrees-of-Freedom (DoF), that is the number of independent translations and rotations a robot can achieve in Cartesian space, of the system will also dictate the components of position and orientation to ignore and remain constant. The number of DoF will depend on the number of independent position variables that would have to be specified in order to locate all parts of a given mechanism.

In figure 3.7 a scheme is illustrated showing both kinematic processes mapping. Joint space variables are defined by q and task, or Cartesian space variables are defined by p. For a planar manipulator, the variables p and q can be

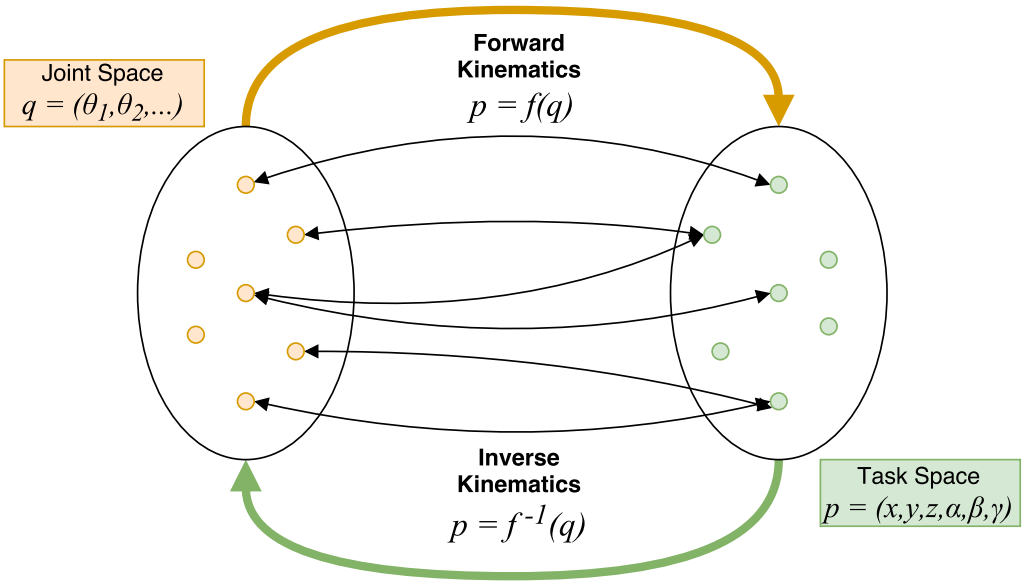

Figure 3.7: Mapping between different kinematic processes and descriptions.

described as in 3.19 and 3.20, where n will be equal to the number of DoF on the manipulator.

$$\tilde{p} = \begin{bmatrix} x_e \\ y_e \end{bmatrix} = \tilde{f}(\theta_1, \theta_2, ...\theta_n) = \begin{bmatrix} f_1(\theta_1, \theta_2, ...\theta_n) \\ f_2(\theta_1, \theta_2, ...\theta_n) \end{bmatrix} \tag{3.19}$$

$$\tilde{q} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} = \tilde{g}(x_e, y_e) = \begin{bmatrix} g_1(x_e, y_e) \\ g_2(x_e, y_e) \\ \vdots \\ g_n(x_e, y_e) \end{bmatrix} \tag{3.20}$$

In this project, kinematics is intended to be used for mobile robots. Unlike robotic manipulators, where the joint angles, which define the manipulator pose, are real-time values acquired by encoders, whereas in mobile robots the acquired data refers to wheel orientation. This data needs to be integrated over time.

As it will be later discussed, the vehicle studied in this dissertation is modeled by the tricycle model which is a non-holonomic system. In physics, a non-holonomic system is a system whose state depends on the path taken in order to achieve it. In other words, for non-holonomic mobile robots, not only traveled distance per wheel matters, but also the speed of each wheel as a function of time, as Correll (2011) refers. A robotic manipulator is a holonomic

Table 3.1: Wheel layouts and respective degrees-of-freedom (Correll, 2011).

Wheel Type	**Example**	**DoF**	**Description**
Standard	Car or tricycle	Two	Rotation around the wheel axle Rotation around its contact point with the ground
Caster wheel	Office chair	Three	Rotation around the wheel axle Rotation around its contact point with the ground Rotation around the caster axis
Swedish wheel	Standard wheel with actuated rollers around its circumference	Three	Rotation around the wheel axle Rotation around its contact point with the ground Rotation around the roller axles
Spherical wheel	Ball Bearing	Three	Rotation in any direction Rotation around its contact point with the ground

system, since each joint position corresponds to a unique pose. Going back through the same trajectory from the starting point will bring the robotic arm to the same position every time. The same goes for a train. Correll (2011) presents the car as an example of a a non-holonomic system. Going through a straight line and doing a right-turn leads to the same amount of wheel rotation as doing a right turn first and then going in a straight line. To get the vehicle to the starting position requires not only to rewind both wheels by the same amount, but also getting their relative speeds right.

In order to define the possible positions and orientations for a robotic system, the DoF concept is also important. Although being true, the definition presented before for robotic manipulators will have different variables to consider on mobile robots. The DoF will be defined by the types of wheels used and their orientation. In table 3.1 some common wheel layout types are presented.

Solving the forward kinematics problem, even for mobile robots, is often achieved through trigonometric relations and kinematic constraints of both vehicle and wheels. For the inverse kinematic problem, finding out how to choose the control parameters to reach a desired position is the main goal. As is forward kinematics, inverse kinematics is also part of the kinematic modeling or analysis, that will allow to understand the system's mechanics and behavior.

3.3 Computer Vision

In the same way as humans can visualize and process what they sense from the surrounding world, by using their eyes and brains, computer vision is a field that focus on mimicking those capabilities on to a computer system. According with Huang (1996) computer vision has two goals, that are intimately related: aiming for computational models of the human visual system, from the biological science point of view, and aiming to build autonomous systems that execute tasks as the human visual system does, and in some cases even surpassing it.

Computer vision is a difficult and challenging field of research. This is due to the complexity of the human visual system. Representing the vast amount of human knowledge in a computer is one of the two major difficulties identified. The other one is related to the hardware and software capabilities of a system to allow the task to accomplish being done in real time.

In robotics, vision includes the processing of images acquired from cameras. This images can have various forms that rely on the camera operation mode. It can operate at different electromagnetic spectrum wavelengths, that can go from visible light (same as the human eye) to other specific wavelengths (Infrared, X-ray, ...).

As Silberberg (2002) states, humans effortlessly navigate using vision, so is natural to consider vision for mobile robots. This being said it is plausible to conclude that vision technology is a key part of mobile robotics. The potential features are tremendous, considering that visual sensing is passive, has high resolution, and is long range in most cases. The same way vision appears to be powerful is also difficult to use in a robotic context. For humans, vision problems are generally underestimated since the human brains devotes most of it's use to solve them.

In computer vision systems there is a range of functions usually performed to meet applications requirements. Those functions can be: image acquisition, pre-processing, feature extraction/detection.

3.3.1 Image Acquisition

The first step in every computer vision system. As mentioned before, for mobile robotics vision sensing can be achieved using cameras. This are optical instruments that can record or captures images. A camera is considered to be a remote sensing device since it senses subjects with no need for physical contact. The camera basic design consists capturing light through a lens element, that can be fixed or zoomed, and then record an image on a light-sensitive sensor. The image sensor can be a charge coupled device (CCD) or a complementary metal–oxide–semiconductor (CMOS) sensor. A sensor is normally composed of few thousand to several million pixels in an array, with each one recording

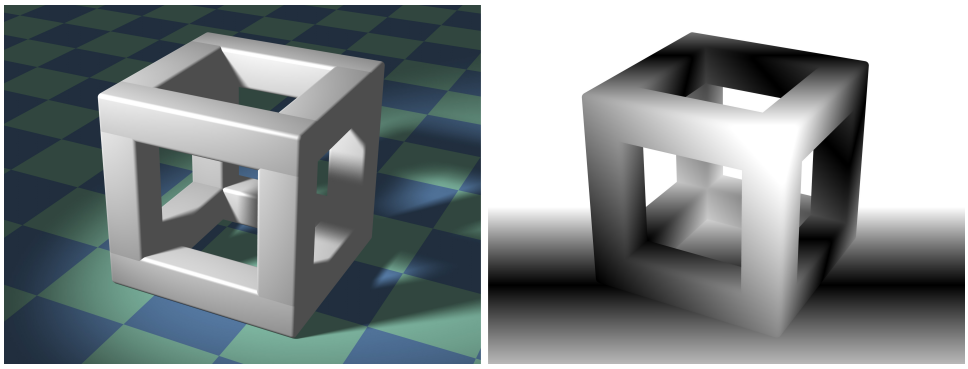

Figure 3.8: Depth map example. In the left the original image and on the right the resulting depth map, adapted from (Dominicos, 2011).

the intensity of light captured. The precision is typically 8 bits. A still camera captures a single static image, while a video camera records a series of static images at different frame rates.

As it will be mentioned in chapter 5 - System Overview, the cameras used in this dissertation project go from RGB color cameras to structured light based cameras and 3D vision sensors based on time of flight (TOF) principle. Both structured light and time of flight based cameras produce a range of different images that can go from depth maps to point clouds. Depth maps withhold data that relates distance of scene objects surface from a given point of view. A point cloud follows similar principals, where the result ultimately intends to represent the external surface of an object to track. In point clouds this result provides a set of data points in a given coordinate system. For a three-dimensional one, x, y, z define those points. In figure 3.8 a depth map is illustrated.

3.3.2 Image Pre-Processing

The image acquired will most of the times be processed before a feature extraction or detection method is applied, in a step called Pre-Processing. Here, the image is filtered by rejecting or transforming the data to the format expected as input. Re-sampling, noise reduction, contrast enhancement or scale space are examples of pre-processing operations.

Some image segmentation methods can also be applied in this step. This will simplify the acquired digital image into a more meaningful one to analyze. The resulting image is divided in multiple parts, easing the task for the feature extraction step. There are different ways to perform image segmentation, that can include color-based segmentation, thresholding, transformations (see figure 3.9) or even texture methods.

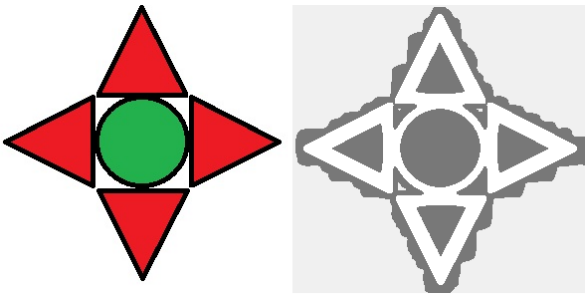

Figure 3.9: Output of HMRF-EM segmentation of color image into 3 clusters (Zeonzir, 2014).

3.3.3 Feature Extraction and Detection

Pre-processing operations focus on one or more input images, returning then the according result in the form of another image. Feature extraction will do the exact opposite. According with Corke (2011), feature extraction operates on an image and returns one or more image features. The features extracted can consist coordinates or even parameters of objects in the image, such as lines, edges and ridges, or even localized corners, blobs or points. This step is obligatory due to the resulting data rate reduction, providing more computational power for more important control purposes. For feature extraction there are some commonly used algorithms such as Canny (edge detection - figure 3.10), Sobel (edge detection) or SUSAN (edge and corner detection), FAST (corner and blob detection) among others.

Figure 3.10: Image before and after using Canny edge detection technique (McLoone, 2010).

The extracted features are selected based on application requirements. The application will always dictate how to organize the computer vision system. As an example, in industrial manufacturing, computer vision can be used to perform automatic inspection of defects in the production line or even for controlling processes.

Chapter 4

Requirements and System Overview

In this chapter the factory layout and the overall project requirements will be presented, as well as the the resulting system overview. There, the operating stacker vehicle, the software tools and the hardware tools used on this dissertation are listed and described. Finally, the simulation model used to develop this dissertation project is presented.

4.1 Factory Layout and Simulation Environment

The project will be centered in Bosch's intralogistics (end of production lines, warehouse area and expedition area), both located in the factory warehouse, as figure 4.1 shows. The stacker vehicle will move the finished goods ready to be collected, into the warehouse. Stackers operate in the finished goods flow, one of the two main flows employed by the company to minimize logistic problems. Since this dissertation addresses docking maneuvers for stacker vehicles, the finished goods flow will be described in more detail.

4.1.1 Stacker's Workflow

Storing the finished goods pallets at the warehouse and supplying production lines with finished goods packaging are the two tasks that compose finished goods flow. Finished goods ready to be moved on to the warehouse are palletized over carts in the blue area shown in figure 4.1. When a pallet is complete the packages are labeled with information regarding the content and destination, and receive an additional green label (identified as a Handling Unit tag - HU). The pallets with the green label strapped are ready to be transported. The stackers can move the pallet to a waiting

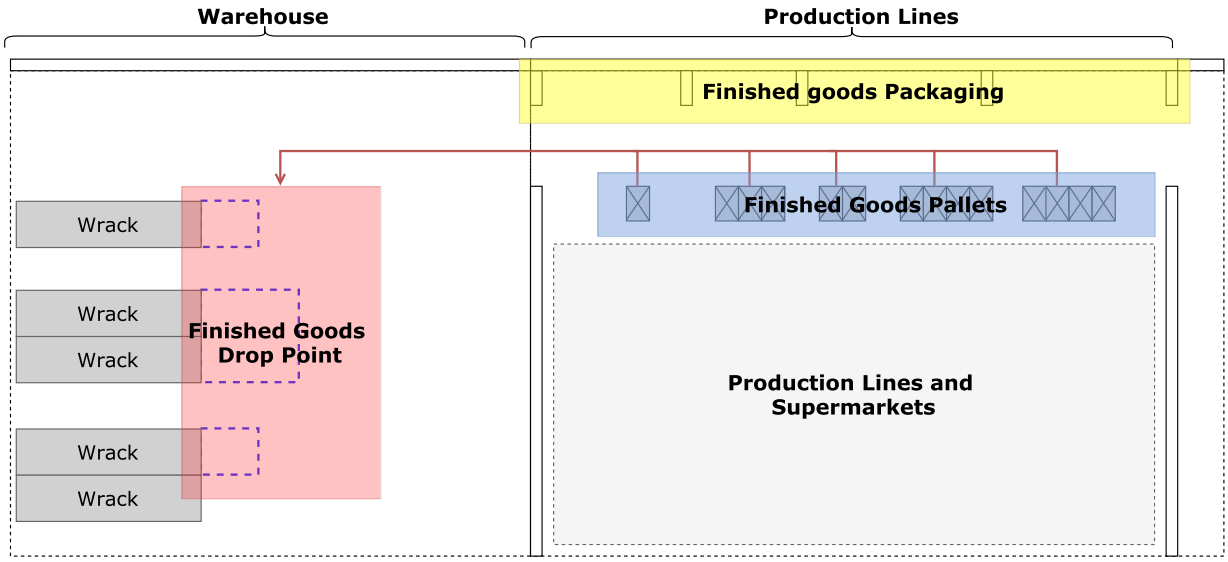

Figure 4.1: Warehouse and production lines plant of Bosch BrgP. Blue line shows the raw material flow and the black line the finished goods flow, the main focus in this dissertation.

area for posterior storage on the specified shelves (red area in figure 4.1) or to another waiting area near expedition in case the pallet is enabled for immediate shipping. During the palletizing process, packaging materials are consumed by the production lines. This packaging is stored on the corridor marked yellow in figure 4.1 and it is accessed by production line's operators to pick up container. The stackers are also responsible to deliver this packaging to this corridor when needed. In figure 4.2 a diagram shows the finished goods flow.

Navigating through the different spaces and corridors in the warehouse will require a pre-defined route to ease navigation towards a given target and to decrease time spent in navigation. The autonomous stacker will need to react

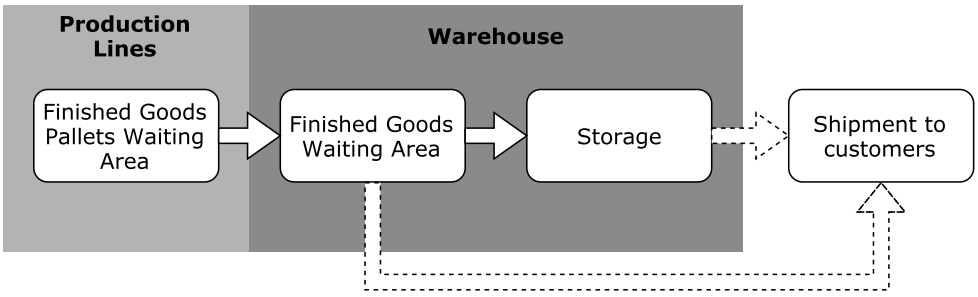

Figure 4.2: Diagram illustrating the finished goods flow. Dashed arrows represent the expedition flow, which is not under study.

to different obstacles during its course the same as a human operated vehicle. The trajectories approached during the docking maneuvers will be accomplished depending on the current sensory data retrieved. The stacker should also read the HU tag using a bar-code reader in order to enable the transport operation. As before mentioned, there are only two destinations for the finished goods pallets:

- The finished goods waiting area;
- The expedition area (located between the warehouse and shipping docks).

To pack the finished goods and assemble the pallets, it is necessary to do the supply of the one-way and returnable packages. The stackers are used to perform this transport for this transport and control of the buffers (with one-way packages) that are located along the production lines.

4.1.2 Stacker's Workspace

As stated before, stacker vehicles will navigate through different spaces in the factory. In this dissertation context, the most relevant workspace for the vehicles to operate will be the finished goods corridor, where the stacker picks pallets. Along the corridor pallets containing finished goods from the end of the production line are placed at one side, and in the opposing side, the finished goods packaging. Figure 4.3 shows the finished goods corridor. Dimension wise, this corridor, consists in a 122.40 m long and 2.98 m wide rectangle. An important aspect for the stacker's workspace is the physical security limiter placed in front of all finished goods pallets. Figure 4.4 illustrates the stacker workspace in the finished products corridor.

4.1.3 Material Handling

Finished goods are stacked over a pallet at the end of the production line. Some of those pallets met the standard measure of 1200 mm $\times$ 800 mm specified by the European Pallet Association (EPAL). This standard measures are the ones selected for this dissertation solution development. Pallets gathered at the end of the production line are filled with car components, making the pallets too heavy for manual carry (pallets containing finished goods are shown in figure 4.5). Therefore stackers are the specialized equipment used to accomplish such a task. These vehicles are equipped with an hydraulic system which provides a tremendous lifting power. This type of system uses oil pressure to enhance the input force.

Figure 4.3: Finished goods corridor.

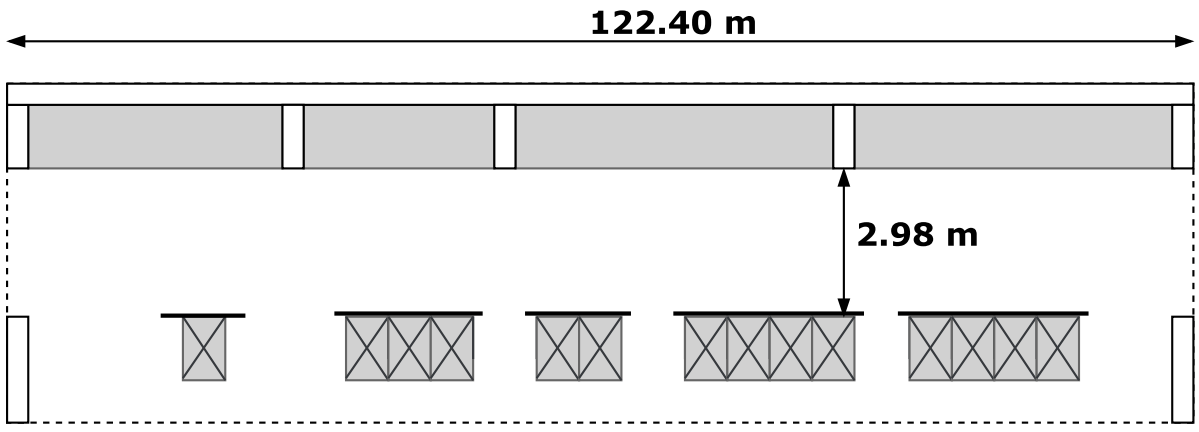

Figure 4.4: Workspace for stacker vehicles regarding the finished goods corridor. Above the finished goods packaging area and bellow the finshed goods pallets represented by crossed squares.

Figure 4.5: Different finished goods palletized.

Another important aspect concerning material handling in Bosch intralogistics, is the pallet placement for the different scenarios. In the end of the production line, has before mentioned, a safety limiter was introduced to avoid possible collisions between the circulating stacker vehicles with the finished goods pallets. This limiter will oblige the pallets to be at 0.30 m above the ground. This is the only exception, since pallets gathered from the finished goods packaging wrack will be placed directly on the ground.

4.2 System Overview

Considering the overall system requirements, as well as the factory workspace dimensions, the system architecture was designed according to blocks diagram illustrated in figure 4.6. Only the sensors needed for the docking maneuvers are mentioned in the context of this dissertation. In order to perform the docking maneuvers, the autonomous stacker vehicle must have sensors to allow detecting the target (pallet to pick) and reading obstacles to avoid. Also, a bar code scanner must be added, so the vehicle knows if the product picked is the correct to be picked, so the factory workflow requirements are met. The acquired data will then be processed by ROS, a middleware detailed in the next sections, and computed inside the vehicle's processing units. The system will then consist in a range of sensors to allow the stacker vehicle to properly navigate in it's workspace environment and a base vehicle that will need to be adapted to install all sensors and processing/computing modules. The autonomous stacker physical assembly is not part of this dissertation, but considering it's importance it is mandatory to approach. This task will be in charge of members in

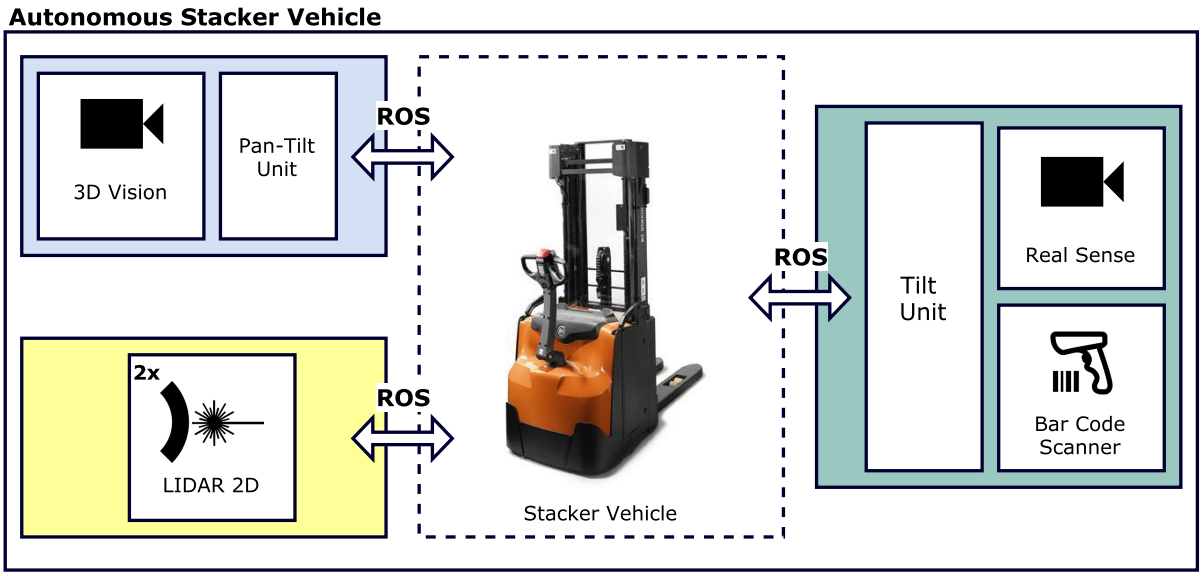

Figure 4.6: System overview.

the development team associated with the "iFactory" project.

4.2.1 Software Tools

In order to achieve the goals set for this dissertation project, the understanding and use of a range of tools and frameworks will be needed. Some will be used to understand kinematic behavior and dynamic through simulations, and also to develop controllers and simulation models for the final solution. Others will serve as a framework for the mobile vehicles acting as a middleware to ease robot and control communication process. Those tools are:

- Coppelia Robotics V-REP (Virtual Robot Experimentation Platform)
- ROS (Robot Operating System)
- MATLAB 2014b, The MathWorks Inc. (Integrated Development Environment)

V-REP - Simulation Platform

Coppelia Robotics V-REP is a recent physical simulator based on a distributed control architecture. V-REP will be the simulator used to create the simulation models and environments for this project: first, to understand vehicle kinematics, and then to develop a simulated docking solution the stacker vehicle.

At the moment, several robot simulation platforms are available, such as Open HRP (Kanehiro et al., 2004), Gazebo (Koenig and Howard, 2004) or Webots (Michel, 2004). Despite offering important functionalities, some fail in offering portability of simulation models and controllers, and present a reduced range of programming techniques.

V-REP simulator is the result of the conciliation between all requirements into a versatile and scalable simulation framework. The framework presents multi-platform compatibility (Linux, Windows or Mac OS X) and allows to use a vast range of preset models, from different robotic arms and navigation robots, to human operators and real scenario obstacles. V-REP presents a friendly user interface and a vast community with forums and help tutorials, allowing for quick adaptation and mastering of the platform. According with Rohmer et al. (2013), by using a distributed control framework, the build of complex simulation scenarios is simplified by partitioning control entities, speeding-up simulation through CPU load distribution over several cores or machines, and finally allowing control for a simulation model through native code execution. Various programming techniques can be choosen simultaneously and even symbiotically (figure 4.7) using V-REP. Some of those techniques are: embedded scripts, add-ons, plug-ins, Remote API clients and ROS nodes.

A simulation scene in V-REP contains several scene objects or elemental objects organized in a hierarchy tree. V-REP supports a great range of scene objects, such as joints with one to three degrees of freedom (prismatic, revolute, screw-like, or spherical), basic shapes, proximity, vision and force sensors, cameras, embedded graphs, lights, dummies, mills and even paths. Calculation modules such as kinematics, dynamics, collision detection, mesh-mesh distance calculation, path/motion planning modules are used to allow operation in conjunction.

V-REP is a versatile simulation framework which offers a multitude of functionalities and programming techniques allowing for a reduced complexity of use. As Rohmer et al. (2013) mentions, V-REP is present in the academic field as well as in the industrial field, performing tasks ranging from system verification, algorithm optimization, simulation of complex assembly chains in factory automation applications, to even robot task planner and controller.

ROS - Vehicle's Middleware

ROS, according to Quigley et al. (2009), when compared with the traditional process management performed by an operating system, cannot be classified as one, but rather as a framework that provides a structured communications layer above the host operating systems. It is composed by a collection of tools, libraries and conventions that aim to simplify the task of creating complex robust robot behavior. ROS was developed to be peer-to-peer, tools-based,

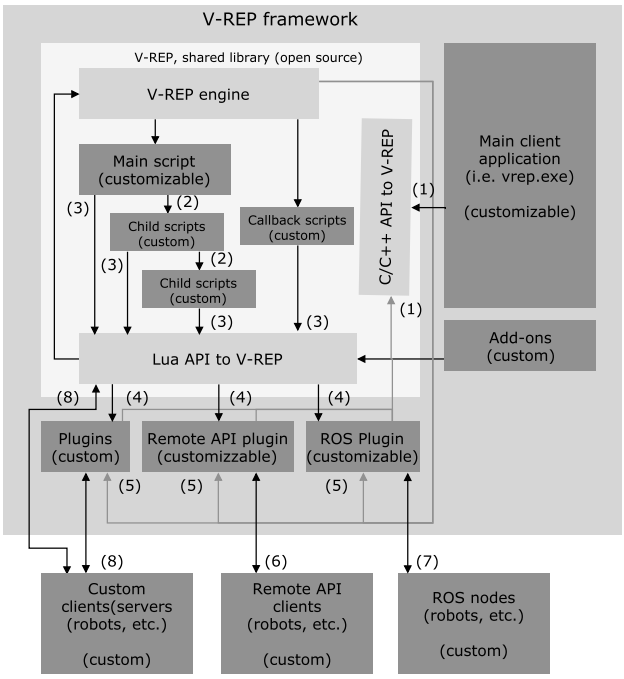

Figure 4.7: V-REP control architecture. Control entities in dark gray, (1) C/C++ API calls, (2) cascaded child script execution, (3) Lua API calls, (4) custom Lua API callbacks, (5) V-REP event callbacks, (6) remote API function calls, (7) ROS transit, (8) custom communication (socket, serial, pipes, etc.) (Rohmer et al., 2013).

multi-lingual, thin, free and open-source.

The fundamental concepts in ROS are nodes, messages, topics and services. Nodes are "software modules" that perform computation. They communicate with each other using messages, by publishing them into a given topic. According with the type of data, a node will subscribe with the appropriate topic. A topic can have multiple publishers and subscribers, and a node may publish and/or subscribe to multiple topics. In figure 4.8 is shown a graph with a simple example of a communication using a pipeline. Usually, ROS graphs are far more complex, with one-to-many or many-to-many connections in cycles. The topic-based publish-subscribe model is not appropriate for synchronous transactions. For this services are implemented. They are defined by it's name and a pair of messages, one for the request and the other for the response. Unlike topics, only one node can advertise a service of any particular name.

Quigley et al. (2009) mentions that ROS can be a powerful tool when facing some robotics development use case scenarios, such as debugging a single node, logging and playback, packaged subsystems, collaborative development, visualization and monitoring, and creating transformation trees.

Figure 4.8: ROS graph with simple communication using a pipeline (Quigley et al., 2009).

MATLAB - Integrated Development Environment

MATLAB (MATrix LABoratory) is a high-performance language for technical computing, visualization and integrated programming environment (IDE). Allowing for object oriented programming, sophisticated data-structures, built-in-editing and debugging tools, the MATLAB IDE proves to be a great asset for research an teaching purposes. For this dissertation purpose, MATLAB was used to develop the control and behavior for the vehicle, as well as the computer vision algorithm. By using this tool together with V-REP it was possible to better understand the stacker vehicle kinematic behavior and dynamics.

This tool was first written to provide easy access to matrix software developed by the LINPACK and EIPSACK projects (Moler, 2004). Besides being built around it's own scripting language, MATLAB uses an array as the basic data element, without dimensioning needs. This is key to easily develop applications for control engineering, algebra, numerical analysis or image processing. In addition, MATLAB offers graphic commands which provide immediate

visualization of results. Packages are gathered containing specific applications for signal processing, symbolic computation, control theory, simulation, optimization, image processing and other science and engineering fields. In the tool, this packages are referred to as toolbox.

As Houcque (2005) states, comparing to FORTRAN or C, MATLAB is more useful to solve technical problems, addressing matrix and array mathematics directly, combined with iterative analysis and design processes.

4.2.2 Hardware Tools

The tools used in this project can also be extended to the acting sensors implemented in both vehicles, that will be used to achieve the final proposed solution. Those sensors are:

- Intel Real Sense Camera R200
- SICK 3Vistor-T 3D Vision Sensor
- SICK S300 Laser Scanner

Intel RealSense R200

The Intel RealSense R200 is installed in the back of the vehicles and allows a long range, stereovision 3D imaging, with a compact size, easy to integrate in any given system. This camera is used for a variety of case scenarios, but for this dissertation purpose the main use is centered on pallet detection during the docking maneuver. The R200 camera then provides color, depth and infrared imaging. The depth video stream is generated using stereo vision technology assisted by an infrared laser projector and two infrared imaging sensors. Color data is provided by the full HD color imaging sensor.

The color camera on the R200 module provides texture information. This can be used to create a point cloud or a 3D model for reconstruction through an overlay over the depth image. The most important characteristics of this camera module are described in table 4.1. The R200 module has also two identical infrared cameras, placed in opposite sites, and labeled ”left” and ”right” accordingly. In table 4.2 the infrared cameras parameters are listed.

Table 4.1: Intel RealSense color camera properties (Intel Corporation, 2016).

Parameter	**R200 Color Camera**
Active Pixels	1920x1080
Sensor Aspect Ratio	16:9
Filter Type	IR Cut Filter
Focus	Fixed
Shutter Type	Rolling Shutter
Vertical Field of View	$43^{\circ} \pm 2^{\circ}$
Horizontal Field of View	$70^{\circ} \pm 2^{\circ}$
Diagonal Field of View	$77^{\circ} \pm 4^{\circ}$

SICK 3Vistor-T 3D Vision Sensor

Up next, it's the SICK 3Vistor-T 3D Vision sensor. This sensor will be the vehicle main vision source during navigation. In this dissertation the sensor is used to check and track the targeted pallet when initiating the docking maneuver. The vision sensor is installed on a pan-tilt unit on the top of the vehicles for a better all around view. The pan tilt unit allows a turning angle around 180°. The 3Vistor-T offers real-time depth information for each pixel based on ToF principle measurement. It provides real time 3D data up to 30 frames per second (fps). In table 4.3 other important parameters for this vision sensor are listed.

The SICK 3Vistor-T is based on the Time-of-Flight (TOF) principle and uses LIDAR (Light Detection and Ranging) technology. LIDAR is a surveying method that uses light in the form of a pulsed laser to measure distances. By illuminating a given target with a low-powered pulsed laser light, and measuring the reflected pulses with a sensor, it is possible to create digital 3D representations of that target, through the differences between laser return times and wavelengths. This method is also known as Lidar, LiDAR or even LADAR (Laser Detection And Ranging).

As Weitkamp (2006) mentions, LIDAR systems are based on a transmitter and a receiver. The transmitter, consisting in a laser, emits light pulses, that can be captured by the receiver. Some transmitters use beam expanders to reduce light divergence. At the receiver, a telescope is used to capture photons on the atmosphere. From this collected light, an optical analyzing system selects specific wavelengths or polarization states. The optical signal is then converted into an electrical one through a detector. Finally, a computer will gather data and store it accordingly.

Table 4.2: Intel RealSense infrared camera properties (Intel Corporation, 2016).

Parameter	**Infrared Cameras**
Active Pixels	640x480
Sensor Aspect Ratio	4:3
Filter Type	IR Band Pass
Focus	Fixed
Shutter Type	Global Shutter
Vertical Field of View	$46^{\circ} \pm 5^{\circ}$
Horizontal Field of View	$59^{\circ} \pm 5^{\circ}$
Diagonal Field of View	$70^{\circ} \pm 4.5^{\circ}$

Table 4.3: SICK 3Vistor-T 3D Vision Sensor properties (SICK, 2015).

Parameter	**3D Vision Sensor**
Active Pixels	176x144
Scan/Frame Rate	30 fps
Working Distance	0.5m to 7.2m
Example Field of View	7m x 5.3m
Detection Angle	69° x 56°

This technology presents high accuracy in ranging, wide-area view, low sensibility to different environmental conditions and low data-processing requirements. Still, this technology has high-cost and it's effectiveness depends much on the objects to detect. Depending on the surface of the objects, LIDAR systems detection capability can be affected. This system are usually blind to glass and can be affected by direct sun contact. The information listed related to this vision sensor was gathered from the product datasheet in SICK (2015).

SICK S300 Laser Scanner

Lastly, the SICK S300 Expert Laser Scanner also used for vehicle navigation in the factory. In this dissertation, the S300 Laser Scanner is used to detect obstacle during the docking maneuver. The same as the 3Vistor-T vision sensor,

the S300 Laser Scanners scan their surroundings and measures distances by using the ToF principle of measurement. The integrated rotating mirror creates a two-dimensional scan. It uses up to 16 configurable protective fields. In this project two of this sensors are used, installed on the bottom of the vehicles in the front corners, and considering the 270° scanning angle it's possible to have a complete monitoring around the vehicle. In table 4.4 other important characteristics of this sensor are listed. The information related to this vision sensor was gathered from the product datasheet in SICK (2016).

Table 4.4: SICK S300 Laser Scanner properties (SICK, 2016).

Parameter	**Laser Scanner**
Distance Measuring Range	30m
Scanning Angle	270°
Type of field set	Triple field sets
Number of field sets	16
Number of fields	48
Resolution	30mm, 40mm, 50mm, 70mm, 150mm, configurable
Angular Resolution	0.5°

4.2.3 Operating Stacker Vehicles

Considering the workspace specifications (detailed in the previous chapter), and other requirements, the selected stacker vehicle for this project was the BT Staxio SPE200DN. The SPE200DN is a 2.200×0.748 m stacker vehicle. The vehicle is front wheel driven, having one wheel at the front and two in the back for better handling. It offers a loading capacity up to 2 tons, with a loading center placed around 600 mm. The vehicle's lifting capability is 6.0 m, ideal to pick and drop loads at different heights. Another important asset of this model is the capability to transport two pallets at the same time. The SPE200DN can navigate at 10 km/h.

Considering it's advantages, this model presents itself as the optimal solution for horizontal transport and stacking, with a compact and reliable hardware, all ideal for intensive work and applications.

4.2.4 Vehicle's Assembly

To meet the workspace and workflow restrictions and specifications before mentioned, the acquired vehicle was modified. First, to accommodate the sensors in the ideal position and next to ease the maneuverability, some parts were removed. The vehicle turned out to be even more compact and flexible, now measuring in length 1.700 m, keeping the same width of 0.748 m and the same lifting capability of 6.0 m. During the assembly process, some new parts were also added, in order to include the acquisition, sensing and overall computing modules. The vehicle is also equipped with a GPS module and compass for location and heading direction purpose. Being a critical part on the assembly process, the position for the sensors will be discussed. As was said earlier, only the sensor modules relevant for this dissertation project context will be approached.

First, the Intel Real Sense camera. As figure 4.9 illustrates, the camera is installed on the back of the vehicle. As before mentioned the main purpose for this camera, in this dissertation context, will be to aid during docking maneuvers This will consist in retrieving the required features from the targeted pallet for the control application to use freely. The camera has an associated tilt unit that will provide an increased vertical field of view. Next, in front of the stacker vehicle is the 3Vistor-T 3D Vision Sensor. By placing this sensor in the top of the vehicle, it will provide visual guidance during navigation. A pan-tilt unit is coupled together with the vision sensor, allowing for a greater field of view horizontally and vertically. This is illustrated in figure 4.10. Lastly, also in front of the vehicle are the SICK S300 Laser Scanner. Installed on the lower part of the vehicle, these sensors are useful to detect obstacles aiding the navigation control. By installing them both in front an almost 360 degrees field of view is guaranteed since the back of the vehicle is guarded by the Intel Real Sense camera. The sensors are also placed at different heights to avoid incorrect data. In figure 4.11 the horizontal field of view provided by both 2D LIDAR sensors is illustrated.

4.3 Simulation Environment

Developing an autonomous navigation model for a vehicle requires the existence of a large number of tests and study around the proposed solution as well as the vehicle behavior. With this considered, developing a simulation environment is an essential step to take. Within this environment, testing and further development can be accomplished with safety regarding the factory or the vehicles in study. A simulation environment can even speed up the testing procedures for the solution developed or for future updates. In order to create such an environment, the factory layout and workflow

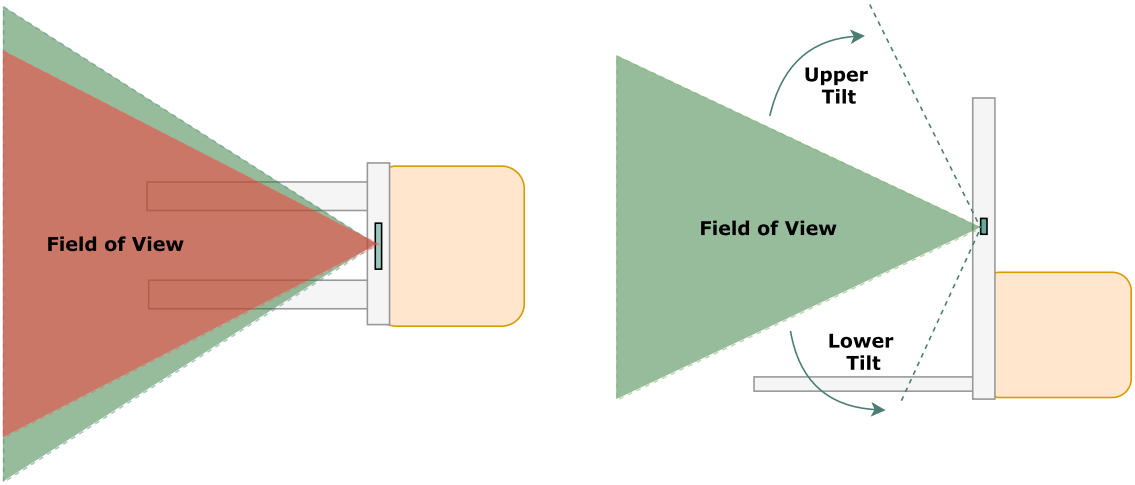

Figure 4.9: Intel RealSense field of view. In red the field of view for the 3D vision and in green for the color camera. Both are approximately meshed in the right picture where the field of view is increased due to the tilt unit. The rectangle in green illustrates the RealSense camera.

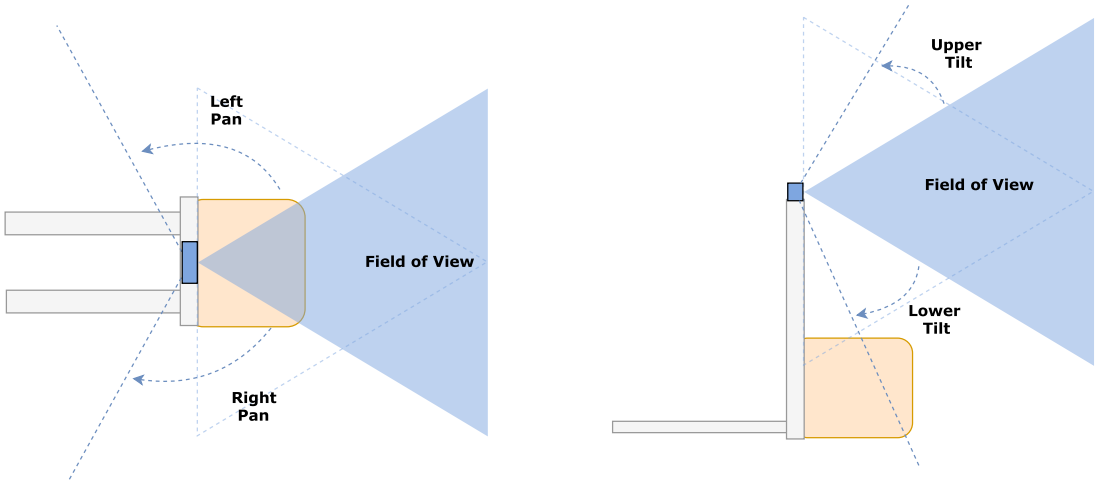

Figure 4.10: 3D Vision field of view. At right the vehicle seen from above and left the vehicle seen from the side. The blue rectangle illustrates the 3D Vision sensor and in front also in blue the resulting field of view with or without panning or tilting.

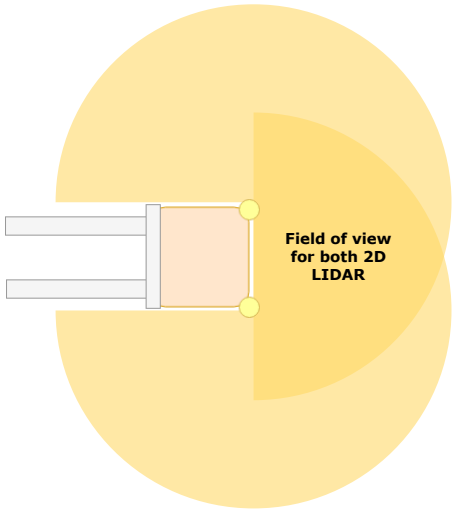

Figure 4.11: 2D LIDAR field of view. Both sensors are illustrated by the yellow circles and the resulting field of view is illustrated around them

were studied in detail, ensuring an environment similar to the one found in Bosch. A development team inside the project "iFactory" was in charge of creating the simulation environment. As mentioned before, V-REP was the software tool used as the simulation modeling tool. Regarding the stackers workflow and workspace, some of the warehouse areas were created inside the modeling program. The stacker vehicles chosen for this project were also accordingly created, as figure 4.12 shows. Stackers will navigate in the finished goods corridor or the end of the production line corridor (the main focus area in this dissertation context), carrying pallets to a drop point in the warehouse. Dimensions of these areas, as well as the pallets to pick and the drop off areas are important to ensure a similar workspace for the autonomous vehicles. In the simulator model, only the corridor length was decreased. Figures 4.13 and 4.14 illustrate a comparison between the real factory and the simulation environment modeled.

4.3.1 MATLAB Integration

Although V-REP allows for native code (LUA) to control a simulation model, MATLAB integrated programming environment was used to develop the control and computer vision algorithm in this dissertation. The iterative analysis offered combined with a matrix and array mathematics address proves to be useful for the amount of calculations needed and image processing steps in the implementation of this work.

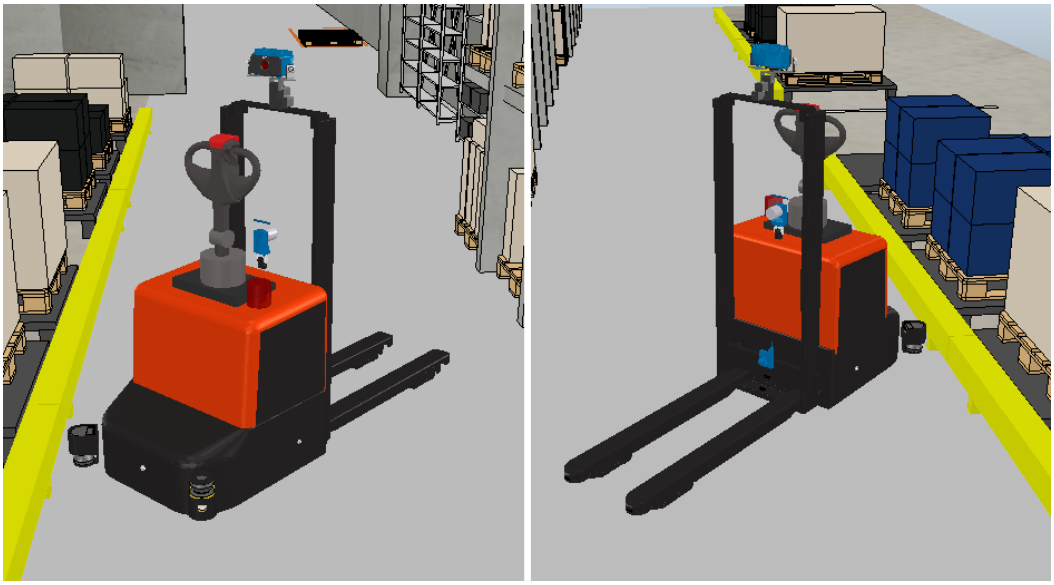

Figure 4.12: The finished goods corridor. In the right, photos of the warehouse and in the left, seen within the simulator.

Figure 4.13: Different points of view of the finished goods corridor. In the right, photos of the warehouse and in the left, seen within the simulator.

Figure 4.14: The finished goods drop point. In the right, photos of the warehouse and in the left, seen within the simulator.

MATLAB can be integrated in the simulation environment already created in V-REP through a remote API. This remote API uses two entities that interact via socket communication, the client side (MATLAB) and the server side (V-REP) implemented via a V-REP plugin loaded by default. To use the remote API functionality in MATLAB, three elements are needed: `remoteApiProto.m`, `remApi.m` and `remoteApi.dll`. Once the files are included, calling `vrep=remApi('remoteApi')` will build the object and load the library. V-REP allows different operation modes for controlling the simulation progress and execute function calls. Those operation modes are: Blocking function calls, Non-blocking function calls, Data Streaming and Synchronous operation. Due to the need of the client side to be synchronized with the simulation progress, synchronous mode was used. Using this operation mode requires the client to first preset the synchronous operation and to trigger the next simulation step. A non-blocking mode is only used for 'set-functions' where the return value is not important.

Chapter 5

Analysis and Proposed Solution

Analyzing the constraints and requirements for this dissertation will start by revisiting the theoretical background, the factory constraints, and the vehicle's components and dimensions. With those chapters covered the solution for docking maneuvers can be designed. The first step will be the kinematic analysis for the stacker vehicle, allowing for a better understanding of the vehicle's maneuverability. Next the docking maneuver should be analyzed and conceptualized considering the requirements and workspace limits before presented. Finally, pallet detection and tracking will be covered.

5.1 Kinematic Modeling

As mentioned before, the first step to develop the docking solution for the stacker vehicle involves studying and developing a kinematic model. This model will allow for a better understanding of the vehicles motion physics and behavior. The stacker vehicles (BT staxio SPE200DN) are mounted on three wheels. Two passive non-drive wheels, placed in parallel on the rear sides. One drive-steering wheel in the middle at the vehicle's front. With this considered, the vehicle structure is similar to the tricycle model, as it is illustrated in figure 5.1.

The rear passive wheels are mounted at the forks ends and the drive-steering wheel at the front of the vehicle. This wheel will be driven by a motor which changes orientation and velocity, thus acting as a steering device. The time varying steering angle of this wheel is described by $\theta_{steer}(t)$. The linear velocity of the drive-steering wheel is described by $v_{steer}(t)$. Both variables will compose the vehicle's two Degrees-of-Freedom. The linear velocity of this

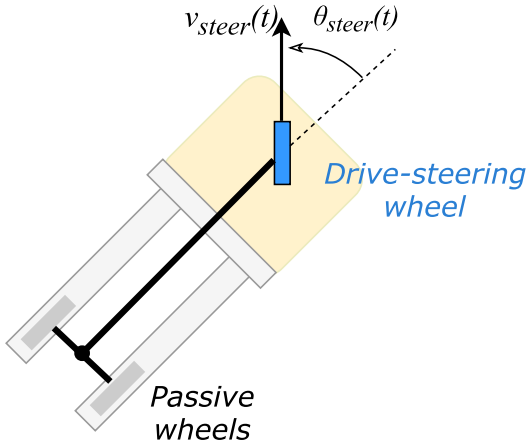

Figure 5.1: Tricycle kinematic model for the stacker vehicle.

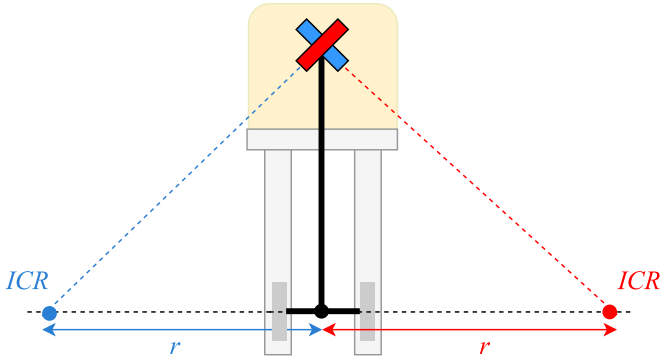

Figure 5.2: Different placements for the instantaneous center of rotation, ICR, that depend on the front wheel steering angle.

vehicle is related to the rotation speed of the steering wheel by:

$$v_{steer} = R_{wheel} \cdot \omega_{steer} \tag{5.1}$$

where ω_{steer} is the vehicle wheel's angular/rotation speed and R_{wheel} is the vehicle's wheel radius. Another aspect to consider in the tricycle model is the Instantaneous Center of Rotation, ICR. If the steering wheel is set to an angle θ_{steer}, the vehicle will rotate with angular velocity ω_{steer} around ICR. The ICR will lie on the line that passes through the perpendicular line of the rear passive wheels, as figure 5.2 shows, where r will dictate the distance between the rear axle midpoint and ICR.

Using trigonometry is possible to infer that $\beta = \frac{\pi}{2} - \theta_{steer}$ (figure 5.3). With this considered the distance

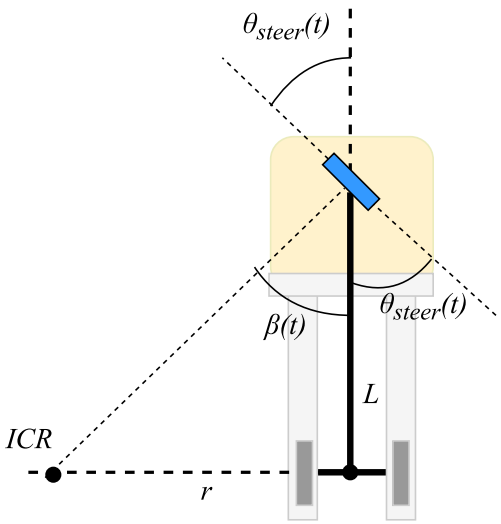

Figure 5.3: By using trigonometry, $\beta = \frac{\pi}{2} - \theta_{steer}$, since the two opposing angles comply with the vertical angle theorem.

between the rear axle and the center of rotation is determined by:

$$r(t) = L \cdot \tan\left(\frac{\pi}{2} - \theta_{steer}\right) \tag{5.2}$$

5.1.1 Forward Kinematics

In summary, to reach a targeted position and orientation, this three-wheeled vehicle uses a single controlled angle and speed wheel. Following the concepts previously presented for this vehicle kinematics, the stacker kinematic model can be illustrated as it is shown in figure 5.4.

At a given moment the position and orientation will be directly linked to the actual linear velocity and steering angle. As mentioned before in Chapter 3, Forward Kinematics are used to calculate the vehicle position. In this case, since the vehicle is non-holonomic speeds are obtained instead. The problem is described by p, retrieved by the

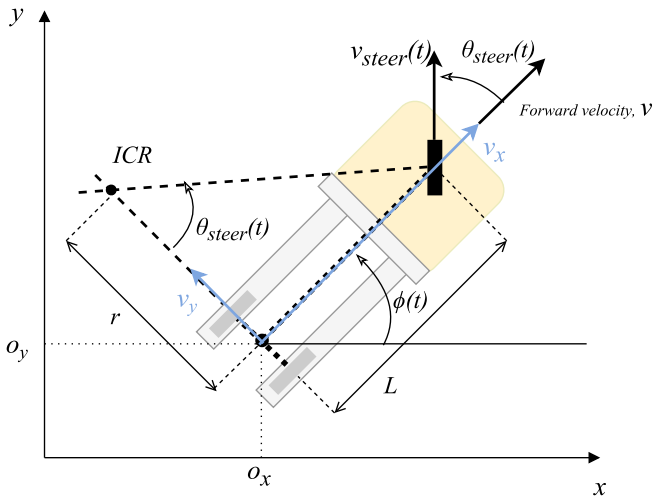

Figure 5.4: Tricycle kinematic model for the stacker vehicle.

vehicle's kinematic system of equations present in 5.3.

$$p = \begin{cases} \dot{o_x} = f_1(v_{steer}, \theta_{steer}) = ? \\ \dot{o_y} = f_2(v_{steer}, \theta_{steer}) = ? \\ \dot{\phi} = f_3(v_{steer}, \theta_{steer}) = ? \end{cases} \tag{5.3}$$

Based on the illustration in figure 5.4, some trigonometric assumptions can be made in relation to the vehicle's forward kinematics:

$$\cos(\phi(t)) = \frac{\dot{o_x}}{v} \Leftrightarrow \dot{o_x} = v \cdot \cos(\phi(t))$$

$$\sin(\phi(t)) = \frac{\dot{o_y}}{v} \Leftrightarrow \dot{o_y} = v \cdot \sin(\phi(t))$$

Continuing the trigonometric assumptions, the forward kinematics can be retrieved as shown in equations 5.4 and 5.5. This will allow to calculate the vehicle position based on actual linear velocity and steering angle.

$$\cos(\theta_{steer}) = \frac{v}{v_{steer}} \Leftrightarrow v = v_{steer} \cdot \cos(\theta_{steer})$$

$$\therefore \dot{o_x} = v_{steer} \cdot \cos(\theta_{steer}) \cdot \cos(\phi(t)) \tag{5.4}$$

$$\therefore \dot{o_y} = v_{steer} \cdot \cos(\theta_{steer}) \cdot \sin(\phi(t)) \tag{5.5}$$

The derivative of the turning radius (ϕ) is given by it's angular velocity $\dot{\phi} = \omega$, and considering that the vehicle's angular velocity is given by the linear velocity v and the distance to the center of rotation r, resulting in $\omega = v \cdot r$. Using trigonometry is possible to obtain the turning radius derivative (equation 5.6).

$$\tan(\theta_{steer}) = \frac{L}{r} \Leftrightarrow r = \frac{L}{\tan(\theta_{steer})}$$

$$\dot{\phi} = \frac{v}{L} \cdot \tan(\theta_{steer}) \Leftrightarrow \dot{\phi} = \frac{v_{steer} \cdot \cos(\theta_{steer})}{L} \cdot \tan(\theta_{steer})$$

$$\therefore \dot{\phi} = \frac{v_{steer}}{L} \cdot \sin(\theta_{steer}) \tag{5.6}$$

In summary, the forward kinematics for the world frame should be as following:

$$p = \begin{cases} \dot{o_x} = f_1(v_{steer}, \theta_{steer}) = v_{steer} \cdot \cos(\theta_{steer}) \cdot \cos(\phi(t)) \\ \dot{o_y} = f_2(v_{steer}, \theta_{steer}) = v_{steer} \cdot \cos(\theta_{steer}) \cdot \sin(\phi(t)) \\ \dot{\phi} = f_3(v_{steer}, \theta_{steer}) = \frac{v_{steer}}{L} \cdot \sin(\theta_{steer}) \end{cases} \tag{5.7}$$

or in the matrix form:

$$p = \begin{bmatrix} \dot{o_x} \\ \dot{o_y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} v_{steer} \cdot \cos(\theta_{steer}) \cdot \cos(\phi(t)) \\ v_{steer} \cdot \cos(\theta_{steer}) \cdot \sin(\phi(t)) \\ \frac{v_{steer}}{L} \cdot \sin(\theta_{steer}) \end{bmatrix} \tag{5.8}$$

5.1.2 Inverse Kinematics

Similarly as the forward kinematics, the problem for the inverse kinematics problem is described by q, given by the system of equations 5.9.

$$q = \begin{cases} v_{steer} = g_1(\dot{\phi}, v) = ? \\ \theta_{steer} = g_2(\dot{\phi}, v) = ? \end{cases} \tag{5.9}$$

There are two ways to solve the inverse kinematics problem. By deriving the trigonometric relationships from figure 5.4 a solution can be found. Another approach would be to perform the matrix inversion of the forward kinematic model. Due to it's simplicity, the first approach will be used.

In the tricycle model, as before mentioned, is known that:

$$v = \omega \cdot r \Leftrightarrow r = \frac{v}{\omega} \tag{5.10}$$

$$\tan(\theta_{steer}) = \frac{L}{r} \Leftrightarrow r = \frac{L}{\tan(\theta_{steer})} \tag{5.11}$$

By simply replacing the result for r from 5.10 in 5.11, and knowing that $\dot{\phi} = \omega$, the steering angle can be deducted:

$$\frac{v}{\omega} = \frac{L}{\tan(\theta_{steer})} \Leftrightarrow$$

$$\therefore \theta_{steer} = \tan^{-1}\left(\frac{\phi \dot{\cdot} L}{v}\right) \tag{5.12}$$

To calculate the linear velocity of the drive-steering wheel v_{steer}, first a trigonometric relation needs to be established:

$$\cos\theta_{steer} = \frac{v}{v_{steer}} \Leftrightarrow v_{steer} = \frac{v}{\cos\theta_{steer}} \tag{5.13}$$

By replacing θ_{steer} in equation 5.13 with the result obtained in 5.5, the result for the inverse kinematics of v_{steer} is found (see equation 5.14).

$$v_{steer} = \frac{v}{\cos\left(\tan^{-1}\left(\frac{\dot{\phi} \cdot L}{v}\right)\right)} \tag{5.14}$$

This result can be further simplified knowing that:

$$\cos(\tan^{-1}) = \frac{1}{\sqrt{1+x^2}} \tag{5.15}$$

Resulting in:

$$v_{steer} = v \cdot \sqrt{1 + \frac{\dot{\phi}^2 \cdot L^2}{v^2}} \Leftrightarrow$$

$$\therefore v_{steer} = \sqrt{v^2 + \dot{\phi}^2 \cdot L^2} \tag{5.16}$$

Once again, in summary, the resulting inverse kinematics for the tricycle vehicle are described by:

$$q = \begin{cases} v_{steer} = g_1(\dot{\phi}, v) = \sqrt{v^2 + \dot{\phi}^2 \cdot L^2} \\ \theta_{steer} = g_2(\dot{\phi}, v) = \tan^{-1}\left(\frac{\dot{\phi} \cdot L}{v}\right) \end{cases} \tag{5.17}$$

or in the matrix form:

$$q = \begin{bmatrix} v_{steer} \\ \theta_{steer} \end{bmatrix} = \begin{bmatrix} \sqrt{v^2 + \dot{\phi}^2 \cdot L^2} \\ \tan^{-1}\left(\frac{\dot{\phi} \cdot L}{v}\right) \end{bmatrix} \tag{5.18}$$

5.2 Docking Maneuver

As it was stated before, stacker vehicles follow the finished products workflow, where they can navigate across the factory warehouse into the finished products corridor or vice-versa. During the finished products crossing, the stacker vehicles obey to some navigation direction rules, where the vehicles should always be close to the limiter on its immediate right. By respecting this rule stacker will allow the navigation in both ways of the corridor as the illustration in figure 5.5 shows. In case an obstacle is found along the way the vehicle will need to work around it returning to

the correct place to continue following its navigation instructions. These key points are important for this dissertation purpose since the docking maneuver to perform will depend on the starting position of the stacker vehicle.

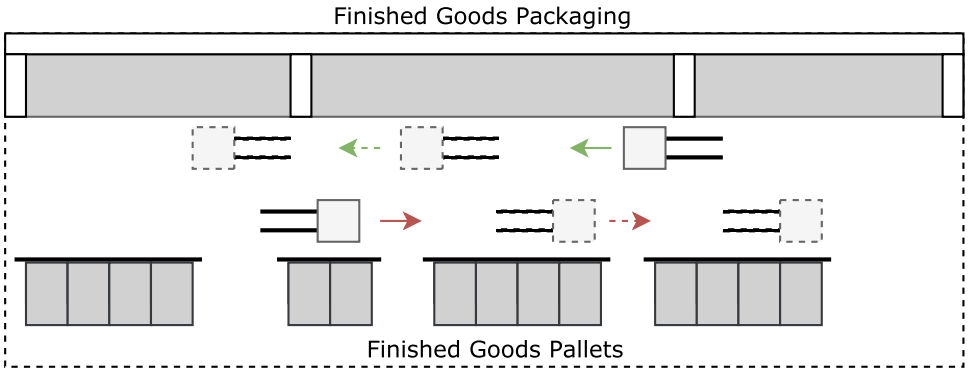

Figure 5.5: Finished products corridor with stackers moving on the according path.

As it was mentioned the vehicle can be either close to the finished goods pallets limiter, or on the opposing side, the finished goods packaging. Considering the confined workspace for the vehicle to maneuver, the docking maneuver will need to be somehow pre-programmed, allowing for a better suiting operation due to the workspace restrictions. This will imply a sequential maneuver, but one that is adapted to the limited workspace. The vehicle ”freedom” to maneuver only by its obstacle avoidance or target acquisition dynamics should be limited during the docking operation.

Navigating with the forks turned forwards would ease possible docking maneuvers encountered, but in order to ensure facility safety for both operators and infrastructures and also to speed up the navigational process the forks will be turned backwards. The vehicle will then need to find a way to turn the forks to the docking station or targeted pallet when a docking operation is requested. Studying the kinematic model for this stacker vehicle, where its model is similar to a tricycle, was a key in solving this problem. Through equation 5.2, which describes the position evolution of the instantaneous radius to the center of rotation relative to the steering angle, is known that if the stacker vehicle front wheel is turned to $\theta_{steer} = \frac{\pi}{2}$ the rotation center is going to be placed in between the rear axle. The vehicle will then be able to turn around it´s rear axle. This point is key in operating the stacker vehicle in the confined space that composes the finished products corridor.

A general concept for the docking operation can be outlined in 4 major steps (illustrated in figure 5.6) by the following order, where the stacker should know a priori where the docking maneuver will take place:

1. The stacker moves in a way its rear axle is aligned with the docking spot requested by logistics;

2. With a wheel steering angle, θ_{steer}, equal to $\frac{\pi}{2}$ the vehicle rotates until its is aligned with the targeted pallet;

3. The vehicle docks the pallet and picks it;

4. Finally, the docking station is left.

The docking solution could be solved simply by resorting to global information, regarding targeted pallets position provided by logistics, but as it is known, that information can have errors associated, and so local information provided by sensors must be used. And so, the before presented outline, for this 4 steps, had also in account the sensors built in the stacker vehicle. For the first step, using the 3Vistor-T 3D vision sensor in front of the vehicle, the stacker can detect the pallet to pick and retrieve its features easing the alignment operation for the stacker. At the same time the LIDAR 2D sensors allow the stacker to avoid possible obstacles during the maneuver. For the second step and third step listed before, the Intel RealSense comes to use. Once again the image acquired by the vision sensor will be processed in order to retrieve the pallet features, allowing the vehicle to fork its target. The same steps can be outlined for a stacker vehicle moving on the opposing direction as figure 5.7 illustrates.

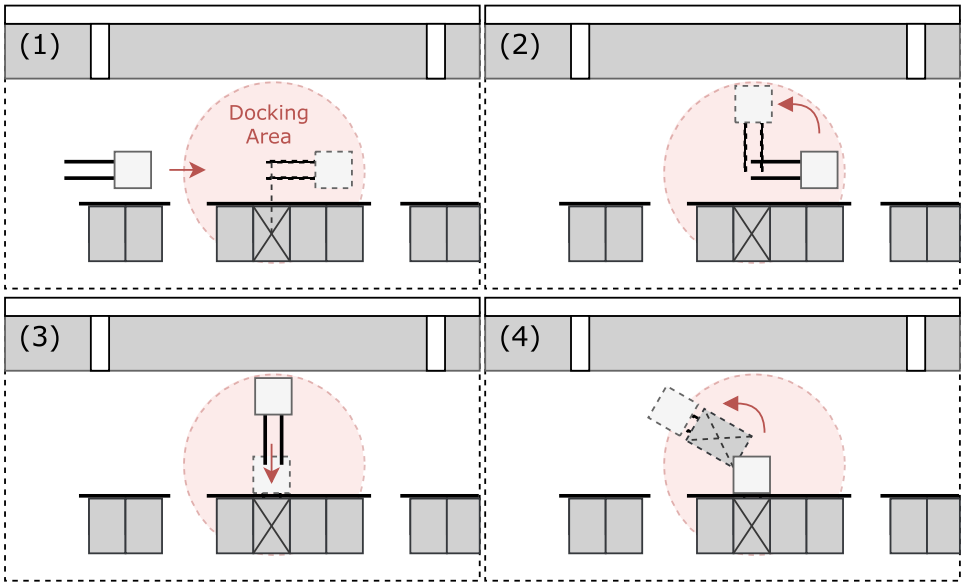

Figure 5.6: Docking maneuver flow, following the outlined 4 steps: (1) Aligning with pallet position; (2) Rotating with θ_{steer}, equal to $\frac{\pi}{2}$; (3) Forking the pallet; (4) Leaving the docking station. The crossed square represents the targeted pallet.

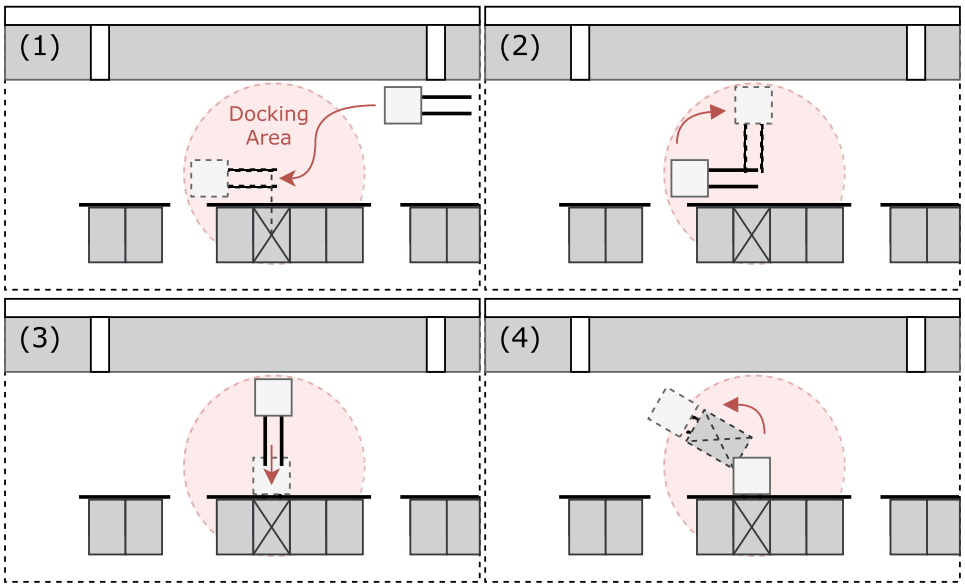

Figure 5.7: Docking maneuver flow in the opposing direction, following the outlined 4 steps: (1) Aligning with pallet position; (2) Rotating with θ_{steer}, equal to $\frac{\pi}{2}$; (3) Forking the pallet; (4) Leaving the docking station. The crossed square represents the targeted pallet.

5.3 Generating Motor Behavior

For the outlined 4 steps of the stacker docking maneuver concept, the first step can be highlighted due to its importance. A proper alignment between the rear axle of the stacker with the pallet midpoint will ease the overall maneuver by minimizing errors during the pallet forking in step 3. The behavior based dynamics presented in chapter 3, Theoretical Background, will be applied to solve this problem.

5.3.1 Target Acquisition

Global information from logistics will provide location data for the targeted pallet. With this data it is possible to place a target in the form of an attractor, right in front of that given pallet. Considering the expected workflow for the stacker vehicles, where navigation can occur from both sides of the corridor with different directions and orientations due to possible obstacles found, two targets will be used instead of just one, as figure (5.8) illustrates. Using one target would be optimal for a scenario where the stacker is parallel with the limiter, stopping at the goal position given by the target (represented by 1). But in a scenario (represented by 2) where the stacker starts the maneuver from a

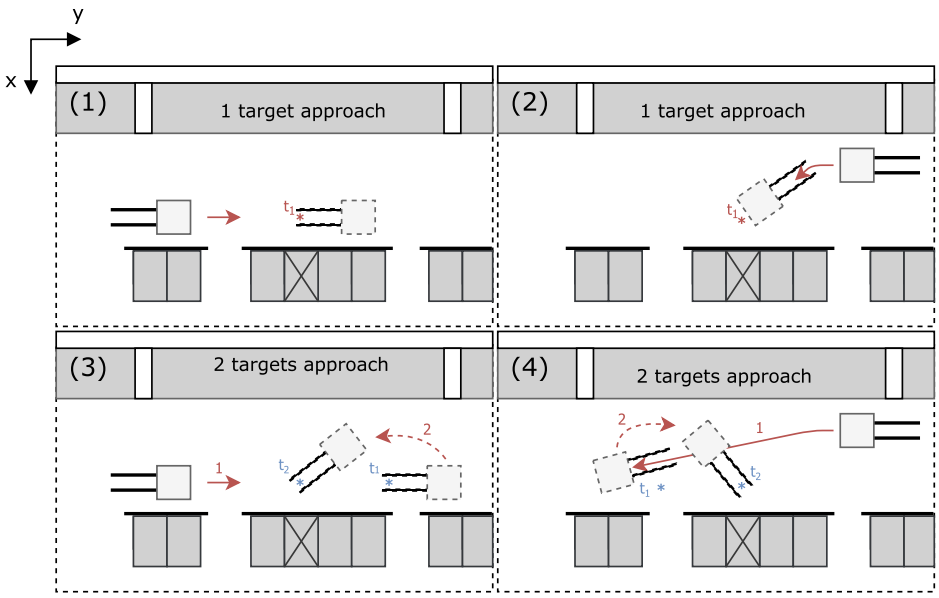

Figure 5.8: Outlined target placement for stacker behavior based dynamic, where (1) and (2) present an approach using 1 target. (3) and (4) present an approach using 2 targets.

different position, and considering the workspace limits, before presented, aligning the goal position with the stacker center of rotation between the rear axle would become almost impossible. Placing a target, with a positive or negative displacement in y-axis relative to the original target, depending on the starting position of the stacker, will ease the maneuver. First the stacker would navigate to the added target t_1 and then would navigate to the goal position in front of the targeted pallet t_2 (represented by 3 and 4 on figure (5.8)).

Using the 3Vistor-T acquired image the possible error associated to the pallet position provided by logistics can be minimized, assuring optimal target location for the maneuver to follow. The error for pallet position provided by logistics is expected to have a displacement in y-axis around 0.50 m.

5.3.2 Obstacle Avoidance

Analyzing the obstacle avoidance dynamics is obligatory in order to prevent the vehicle of possible collisions with the workspace limits. Random obstacles, such as workers, operators, other vehicles or even factory materials badly placed, are not considered since the workspace is expected to be clear before the stacker proceeds to the docking maneuver. In case one of those referred obstacles appear midway of the maneuver, the vehicle stops and waits for

the obstacle to step away or to be removed. With this, only the workspace limits are considered as obstacles for the obstacle avoidance dynamics.

Using the S300 Laser Scanners 2D LIDAR installed on the front of the vehicle as it is shown on chapter 5, it is possible to retrieve data regarding for the workspace limits around the stacker, feeding this information to the obstacle avoidance dynamics to implement. Considering their position, in front of the vehicle, and considering the center of rotation position for this stacker vehicle (in between the rear axle), a vectorial transformation must occur in order to translate the information given by the Laser Scanners, simplifying the complexity of the task in hand, that would require analyzing detected obstructions, using different coordinate systems.

Both sensors placed on left and right of the stacker will provide one or more points in relation to its Cartesian coordinate system., ${}^{\{S_{side}\}}P_{obs,i} = (x_i, y_i)$. The obstructions detected are then translated to the Cartesian coordinate system of the vehicle V (see illustration in figure 5.9). Given a point ${}^{\{S_{side}\}}P_{obs,i}$ representing an obstruction detected by the left Laser Scanner can be represented on the Cartesian coordinate system of the vehicle $\{V\}$ through equation 5.19, where ${}^{\{V\}}P_{obs}$ is the point P in relation to the Cartesian coordinate system of the vehicle and ${}^{\{V\}}T_{S_{left}}$ is the homogeneous transformation matrix between the Cartesian coordinate system of the vehicle and the Cartesian coordinate system of the left Laser Scanner. The same goes for the right one (equation 5.20).

$$ {}^{\{V\}}P_{obs} = {}^{\{V\}}T_{\{S_{left}\}} \cdot {}^{\{S_{left}\}}P_{obs} \tag{5.19} $$

$$ {}^{\{V\}}P_{obs} = {}^{\{V\}}T_{\{S_{right}\}} \cdot {}^{\{S_{right}\}}P_{obs} \tag{5.20} $$

Lastly, the total scan area is divided into sectors to allow calculating the angular range of the repulsive effect σ_i just as was presented in equation 3.7 in Chapter 3. In figure 5.10 an example is illustrated for a scan area divided into 10 sectors.

5.4 Pallet Detection and Tracking

In the docking maneuver process, picking pallets is the main purpose for the stacker vehicles, besides their transport within different stations. With this considered, a computer vision solution for pallet detection and tracking must be achieved, for both simulation and real image case scenarios. This will allow the vehicle to also rely on the local

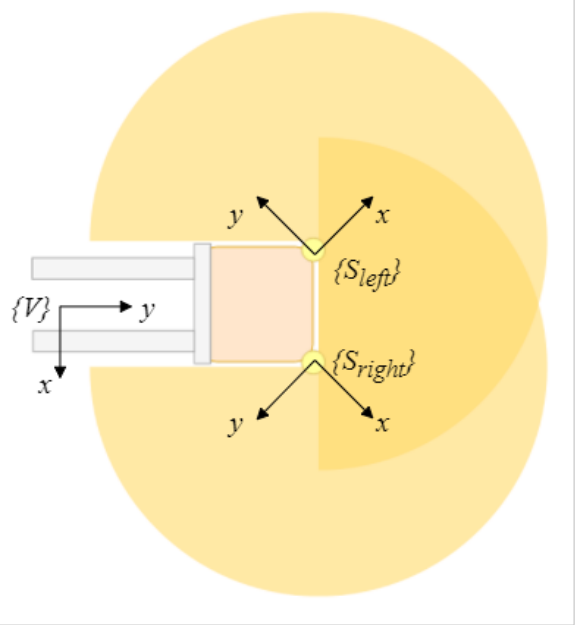

Figure 5.9: Cartesian coordinate representations of both vehicle and SICK S300 sensors.

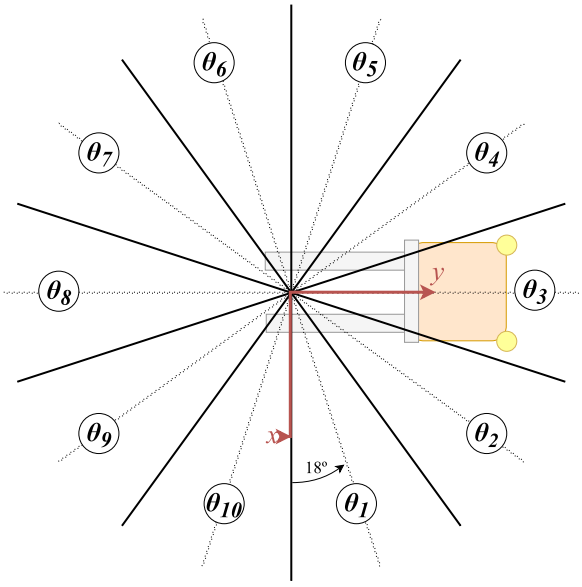

Figure 5.10: Resulting angles for a scan area divided in 10 sectors.

Figure 5.11: Pallet seen from the Intel RealSense inside V-REP simulator. At left the RGB image and on the right the depth image (RGB-D) resultant from the vision sensor.

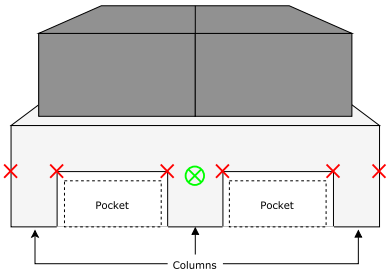

Figure 5.12: Pallet model with the resulting detected midpoint and pockets.

information gathered by vision sensors instead of relying only on global information for pallet location data.

Analyzing the pallet geometry and characteristics perceived by the vision sensor is key in developing a solution for pallet detection. In figure 5.11 the targeted pallet is shown from the stacker vehicle point of view, from both RGB color vision sensor and the resulting RGB-D image collected from the infrared projector on the Intel RealSense. A pallet will then mainly consist in three columns with the same width, separated by two spaces for the stacker to fork the pallet, that will be named pockets. This pattern is key to allow the stacker vehicle to detect a pallet, since the pallet pockets are the main features to extract.

The docking maneuvers are expected to start when the stacker vehicle is close to the picking location, with a distance around 2.0 m from the targeted pallet. This will allow the vehicle to exclude a great amount of false-positives (i.e. the different pallets on the corridor), simplifying the solution to develop. The resulting solution should allow the vehicle to understand where the pallet pockets are, and where the pallet midpoint (the center point of middle column) is. This result is illustrated in figure 5.12.

Chapter 6

Development and Implementation

After the analysis step is concluded, the solution can be developed. In this chapter all implementation that allowed this dissertation development is detailed. First, the implementation of the proposed behavior dynamics is presented. Next, the implementation for pallet detection/tracking is described. The merge between techniques are detailed in the last section resulting in the docking maneuvers solution.

6.1 Generating Motor Behavior

A key part on the docking maneuver will be the vehicle behavior dynamics when navigating to target positions in it's limited workspace. The implementation of this dynamics will be presented bellow.

6.1.1 Target Orientation

The development for target orientation obeys the dynamics described in Chapter 3. Using the dynamical system established by 3.4, is possible to assign an angular speed for the stacker vehicle front wheel, that will shift the heading direction ϕ of it towards the target. At this moment the traveling velocity to consider will be constant. Before assigning those behavioral variable values to the vehicle, both angular speed and traveling velocity must consider the kinematics established by 5.17. In figure 6.1 the behavior for target orientation can be seen.

As figure 6.1 shows, the heading direction of the stacker is always attracted to the target. The solution implemented followed on algorithm 1. In (1) an attractor is placed in the same orientation the first target (in red) is placed relative

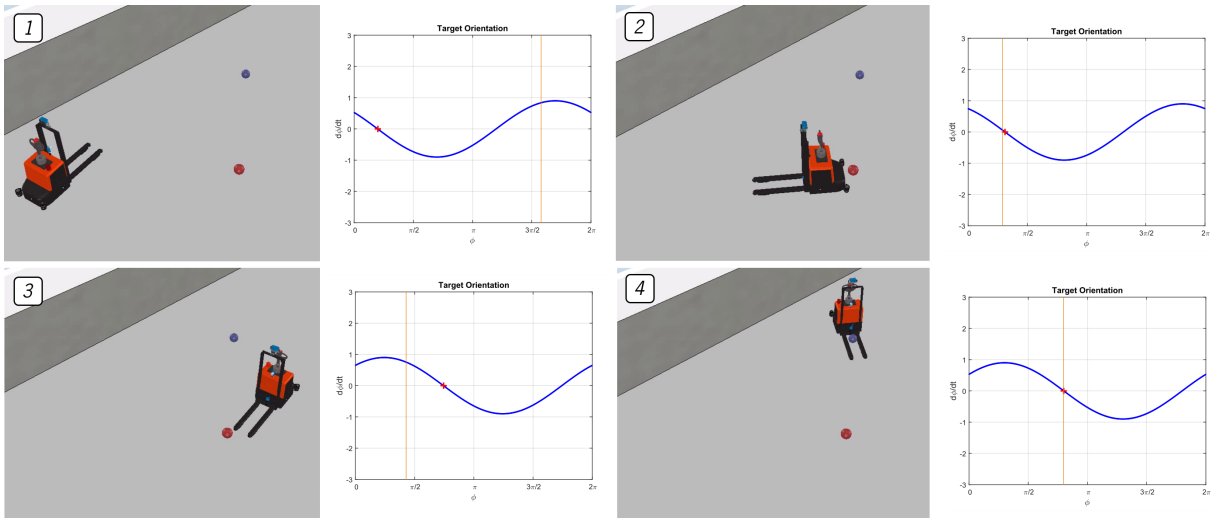

Figure 6.1: For each figure , on the right the stacker vehicle maneuvering to the targets (spheres in red and blue) inside V-REP simulator, on the left the resulting force for the target orientation dynamic implemented. The heading direction is represented by the vertical orange line, and the target orientation dynamics resulting force is represented in yellow.

to the vehicle. Then in (2) the vehicle turns in order to head for that target. As the goal position is matched, a second target is placed (in blue) at a different orientation, turning once again the vehicle in to that direction (3). As (4) illustrates, the vehicle matches the final goal position. For each step the vehicle's heading direction moves to the fixed point created by the target.

6.1.2 Obstacle Avoidance

Such as target orientation, the development for obstacle avoidance obeys the dynamics described in Chapter 3, where the angular speed for the stacker vehicle front wheel is given by $F_{obs}(\phi)$ described by equation 3.8, which represents the sum of all sensor contributions. As before, the traveling velocity will remain a constant value. The scan area was then divided in 41 sectors, with each sector covering 8.78°. This number of sectors offer a good balance between precision and processing time consumed. Considering the scanning angle of the Laser Scanners and the diminished relevance of obstacles on the back of the vehicle for the maneuvering established, only a part of those sectors are considered for the dynamics developed. Those are the 21 sectors in the front of the vehicle (frontal 180° that starts at the rear wheels axle). In figure 6.2 the resulting 21 sectors are shown inside the V-REP simulator.

Figure 6.3 shows the stacker vehicle using the implemented algorithm 2 to navigate in a scenario where two

Algorithm 1: Target Orientation dynamics

$n \leftarrow$ target number;

$dir \leftarrow$ variable to reverse heading direction, (1) forwards (-1) backwards;

$v \leftarrow$ travelling velocity;

$L \leftarrow$ stacker length;

$dx_{target_n} \leftarrow$ distance in x to target position;

$dy_{target_n} \leftarrow$ distance in y to target position;

$dist_{target} \leftarrow \sqrt{{dx_{target}}^2 + {dy_{target}}^2}$;

$\psi_{tar} \leftarrow \tan^{-1}(\frac{dy_{target}}{dx_{target}})$;

$f_{tar} \leftarrow -\lambda_{tar} \times \sin(\phi - \psi_{tar})$;

$v_{steer} \leftarrow dir \times \sqrt{v^2 + {f_{tar}}^2 \cdot L^2}$;

$\theta_{steer} \leftarrow \tan^{-1}\left(\frac{f_{tar} \cdot L}{v}\right)$;

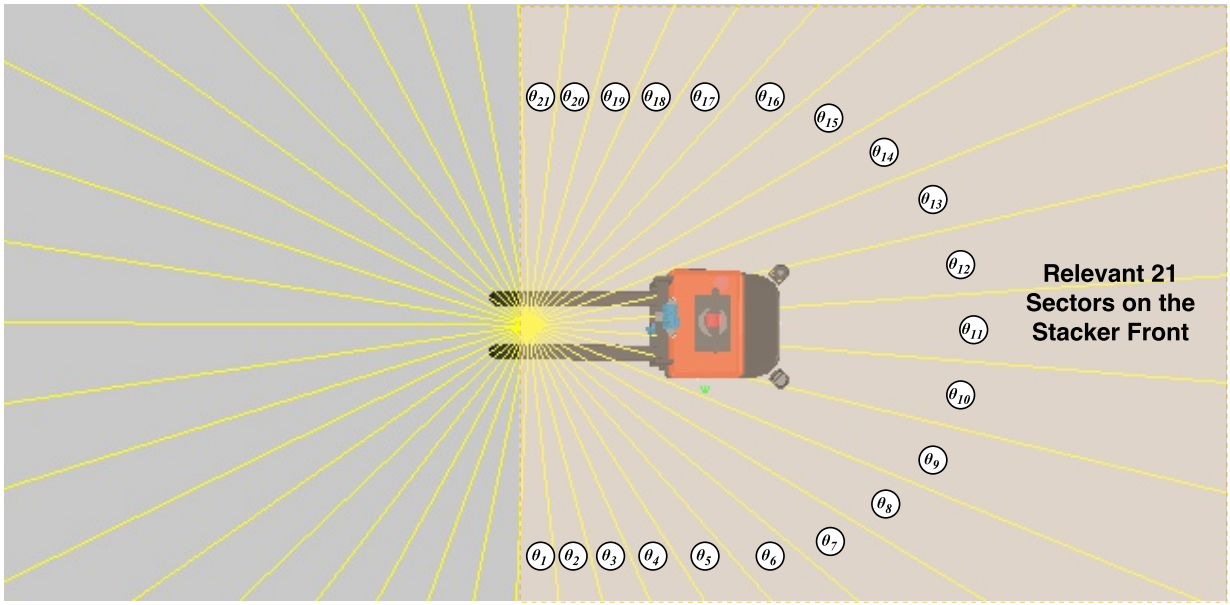

Figure 6.2: The resulting virtual sensors for a scan area divided in 41 parts. In the front of the vehicle are shown the 21 sectors to consider.

obstacles were placed around it, illustrating as well the resulting dynamic for the obstacle avoidance implementation. In this scenario, in (1) the stacker detects the wall at it's right and a repeller is immediately placed at this direction making the vehicle navigate to it's left. In (2) the repelling force dissipates and the vehicle navigates with a steering angle close to zero. In (3) an obstacle is found right in front of the vehicle. The resulting force will create a repeller at that orientation and immediately the vehicle's heading direction will be pushed out of that fixed point, as the figure illustrates, prompting the stacker to shift to its left. Finally, in (4) the stacker vehicle heading direction stabilizes after being shifted away from the obstacle, proceeding to the opening.

Algorithm 2: Obstacle Avoidance dynamics

$dist_{obs} \leftarrow$ array containing the distances read from the 21 established virtual sensors.;

$sdistance_{max} \leftarrow$ maximum distance for the established virtual sensors;

$v \leftarrow$ travelling velocity;

$L \leftarrow$ stacker length;

$B \leftarrow$ stacker width;

$F_{obs} \leftarrow 0$;

$i \leftarrow 1$;

while $i <= N_{sensors}$ **do**

  if $dist_{obs,i} <= sdistance_{max}$ **then**

    $\lambda_{obs} \leftarrow \beta_{1,i} e^{\frac{-dist_{obs,i}}{\beta_{2,i}}}$;

    $\sigma_{obs} \leftarrow tan^{-1}\left[tan\left(\frac{\delta\theta}{2}\right) + \frac{B}{L+dist_{obs,i}}\right]$;

    $f_{obs} \leftarrow \lambda_{obs}(\phi - \psi_i) e^{\frac{-(\phi-\psi_i)^2}{2\sigma_{obs}^2}}$;

    $F_{obs} \leftarrow F_{obs} + f_{obs}$;

    $i \leftarrow i + 1$;

  end

end

$v_{steer} \leftarrow \sqrt{v^2 + {F_{obs}}^2 \cdot L^2}$;

$\theta_{steer} \leftarrow \tan^{-1}\left(\frac{F_{obs} \cdot L}{v}\right)$;

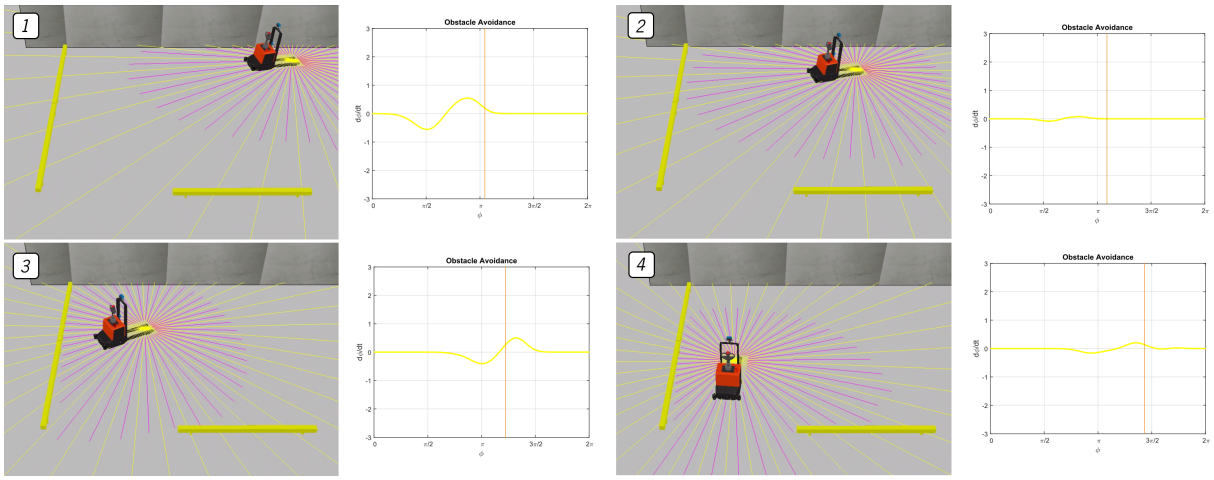

Figure 6.3: For each figure, on the right the stacker vehicle maneuvering on a restricted environment inside V-REP simulator, on the left the resulting force for the obstacle avoidance dynamic implemented. The heading direction is represented by the vertical orange line, and the obstacle avoidance dynamics resulting force is represented in yellow.

6.1.3 Merging Behaviors

The resulting dynamics of the merge between target orientation and obstacle avoidance will be composed of the sum of both attraction and repulsive forces, as equation 3.11 states. Traveling velocity, in it's turn, will be defined by $g(v)$ 3.12. The resulting dynamics is presented in algorithm 3. Using this implementation, figure 6.4 scenario shows the stacker vehicle moving to a given target surrounded by obstacles. As the figure shows, first, the vehicle is repelled from the wall at it's right the same way as it before happened in the obstacle avoidance dynamics (1). But as soon as the vehicle is, in someway, obstacle free it maneuvers to the defined target (in red). As it approaches the limiter in (2), the obstacle avoidance dynamics (in green) overlap the target orientation dynamics (in blue) and the vehicle first attracted to the target is now repelled from that direction as the sum of the forces (in purple) prioritizes obstacle avoidance, placing a repeller in the direction of the target and obstacle. The vehicle continues to move close to the limiter and as soon as an opening is found, the vehicle maneuvers around the obstacle as it can be seen in (3). Once again, since no obstacles are found at the vehicles front, the merged dynamics places an attractor in the target direction, so the vehicle can navigate towards it (4). At this moment the target orientation dynamics overlapped the obstacle avoidance one.

Algorithm 3: Merged Behavior dynamics

$dir \leftarrow$ variable to reverse heading direction, (1) forwards (-1) backwards;

$dist_{obs} \leftarrow$ array containing the distances read from the 21 established virtual sensors.;

$sdistance_{max} \leftarrow$ maximum distance for the established virtual sensors;

$v_{des} \leftarrow$ desired obstacle free navigation velocity;

$L \leftarrow$ stacker length;

$B \leftarrow$ stacker width;

$f_{tar} \leftarrow$ Attractive force resultant from target orientation dynamics;

$F_{obs} \leftarrow$ Repulsive force resultant from obstacle avoidance dynamics;

$U_{obs} \leftarrow 0$;

$i \leftarrow 1$;

while $i <= N_{sensors}$ **do**

  if $dist_{obs,i} <= sdistance_{max}$ **then**

    $\lambda_{obs} \leftarrow \beta_{1,i} e^{\frac{-dist_{obs,i}}{\beta_{2,i}}}$;

    $\sigma_{obs} \leftarrow tan^{-1}\left[tan\left(\frac{\delta\theta}{2}\right) + \frac{B}{L+dist_{obs,i}}\right]$;

    $u_{obs} \leftarrow \left[\lambda_{obs}\sigma_{obs}^2 e^{\left(\frac{-(\phi-\psi_i)^2}{2\sigma_{obs}^2}\right)} - \frac{\lambda_{obs}\sigma_{obs}^2}{\sqrt{e}}\right]$;

    $U_{obs} \leftarrow U_{obs} + u_{obs}$;

    $i \leftarrow i + 1$;

  end

end

$\alpha \leftarrow tan^{-1}\left(\frac{c \times U_{obs}}{pi}\right)$;

$v_{obs} \leftarrow \frac{d_{mins}}{T_{obs}}$;

$c_{obs} \leftarrow (0.5 + \alpha)$;

$c_{tar} \leftarrow (0.5 - \alpha)$;

$g \leftarrow -c_{obs} * (v - v_{obs}) - c_{tar} * (v - v_{des})$;

$F_t = F_{obs} + f_{tar}$;

$v_{steer} \leftarrow dir \times \sqrt{g^2 + {F_t}^2 \cdot L^2}$;

$\theta_{steer} \leftarrow \tan^{-1}\left(\frac{F_t \cdot L}{g}\right)$;

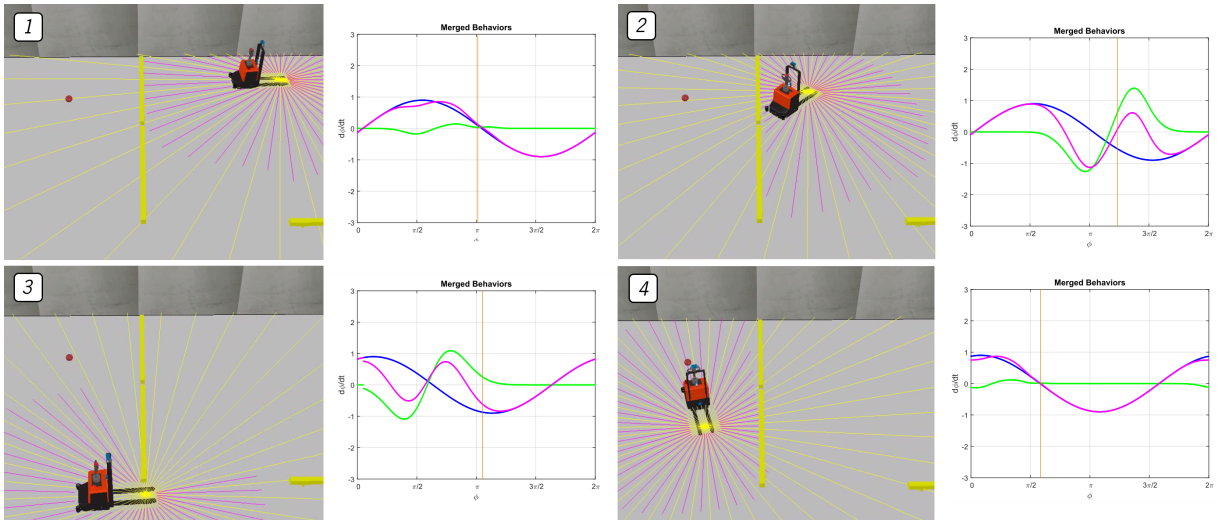

Figure 6.4: For each figure, on the right the stacker vehicle maneuvering on a restricted environment inside V-REP simulator and moving to a given target (in red), on the left the resulting force for the target orientation and obstacle avoidance dynamics implemented. The heading direction is represented by the vertical orange line. The obstacle avoidance dynamics resulting force is represented in green, the target orientation dynamics resulting force in blue and the merged behavior in purple.

6.2 Pallet Detection and Tracking

Considering the previously presented analysis for pallet detection, where pallet pockets, sides and midpoint are the main features to detect on the pallet, the methods described next were implemented.

6.2.1 General Concept

Image samples are sequentially processed through this implementation. Those samples are RGB-D images acquired from the vision sensors used. To start the image processing, a series of parameters must be set. Those parameters are related with pallet expected dimensions, the expected placement of the targeted pallet to create a ROI for image processing and finally the values that define the range of image to segment. All parameters were adjusted according with the vision sensor in use (Intel RealSense or the 3Vistor-T). The targeted pallet is expected to be placed in an estimated region. The ROI is then created, eliminating possible false positives and decreasing the amount of image to process. By segmenting this region, a binary image with enhanced contrast is created (as illustrated in figure 6.5). Next, the features are retrieved from the pallet using an iterative algorithm running on the segmented image that will be presented later. This first pallet features, retrieved from the sample image, are the pallet pocket corners that will allow to calculate the pallet midpoint. With these values the ROI can be re-established now that the targeted pallet location is known. In the end all features are returned. The vision solution developed for this dissertation project followed the flowchart sequence illustrated in figure 6.6.

6.2.2 Pallet Features Extraction

Extracting the pallet features includes, first detecting the pallet sides and pocket corners, and then calculating the pallet midpoint. At this stage, using the cropped segmented image, the corners can be found by cycling through the binary image pixels, and applying a pixel scan. A corner can be defined by a difference of contrast, in this case, since the binary image is used, from 1 to 0 or vice-versa. The found corners will then be matched against the expect threshold for the pallet pockets and columns dimensions.

The image is cycled from bottom to top and corners positions found are saved on a matrix pcorners containing their pixel position. This matrix will then be used to calculate the midpoint that should be in the middle of two of those pocket corners. The number of features to be found on the pallet must be 6, so as soon as 6 features are found, the

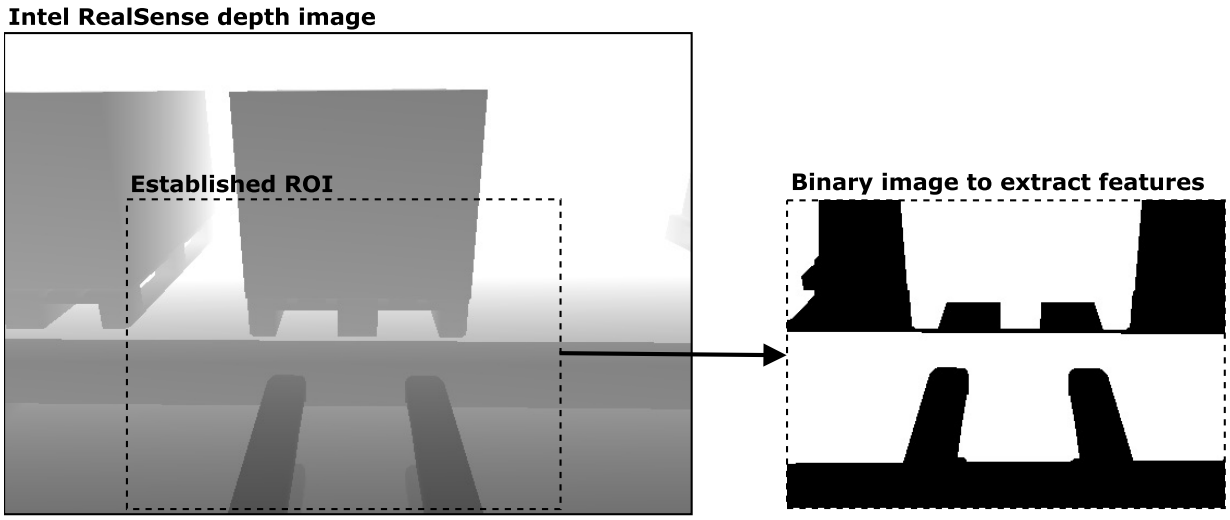

Figure 6.5: At left the Intel RealSense depth image and on the right the extracted binary image to process.

iterative algorithm is broken.

6.3 Docking Maneuver

The implemented docking maneuver was based on the 4 steps previously conceptualized, but in order to combine all implementation before presented, extra steps were included to assure a successful docking maneuver from any stacker starting point. First, the vehicle must always steer to a position parallel to the corridor length. For this step the vehicle inner compass is used to assure the correct orientation is matched. Then, it is possible to advance to the next step that will involve looking for the targeted pallet and when found, adjust the target previously calculated according with the pallet real positioning. This step will use the depth image provided by the 3Vistor-T 3D vision sensor, that will tilt -50° and pan $\pm 90°$ depending of the stacker initial position in relation to the targeted pallet. After the pallet being found and the target position is recalculated, the vehicle can start maneuvering for the targets, using the implementation for behavioral based dynamics previously presented. As soon as the goal position is hit, the stacker rotates with a angular steering velocity of $\pm\frac{\pi}{2}$, that once again will be directly linked to the stacker starting position. When a preset threshold for the orientation of the stacker is met, the Intel RealSense comes into play. In a real scenario, the rotation previously performed should happen with the camera operating at the same time, but due to excessive and somehow unnecessary computation the camera and the pallet features algorithm will only actuate when the vehicle finishes part

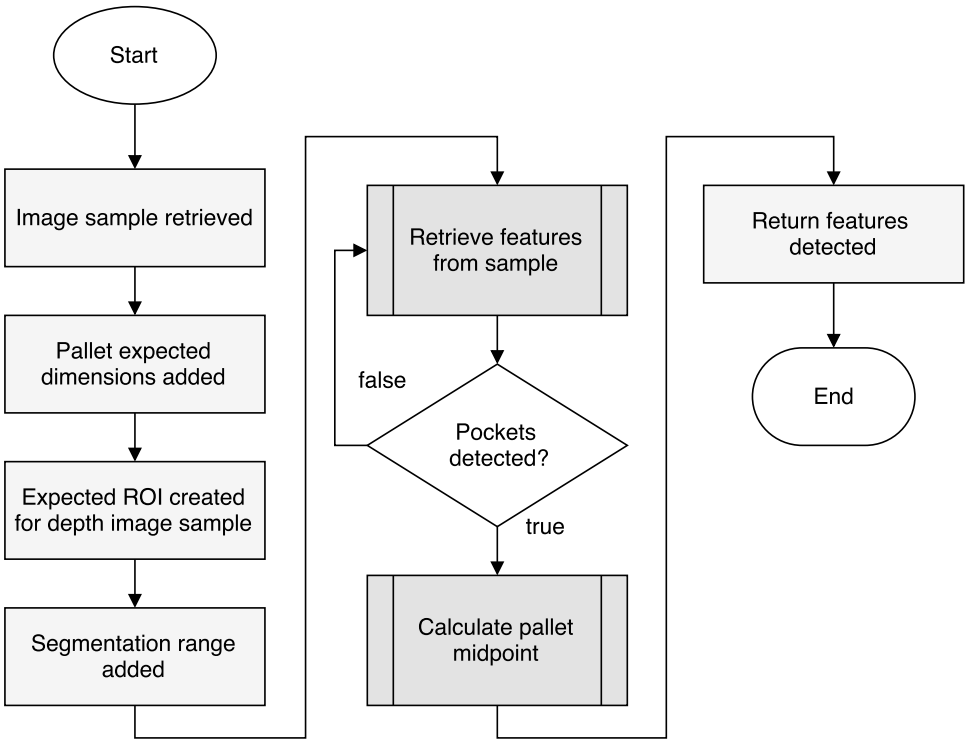

Figure 6.6: Pallet detection implementation general flowchart for each image sample retrieved.

of the required rotation. With the processed depth image provided by the RealSense camera, the stacker is able to adjust possible errors in it's orientation and positioning towards the pallet midpoint. With the errors minimized the vehicle is now able to end the maneuver and fork the targeted pallet. Figure 6.7 illustrates the flow implemented for the docking maneuvering. Due to the extreme importance of the implementation steps 2, 3 and 5 further details related to those steps will be presented.

6.3.1 Target Position Tunning

Knowing the pallet position on the finished products pallets line will provide enough information to calculate the two targets approach positions (see illustration in figure 5.8 in the Analysis chapter). The first target to position will be the one right in front of the pallet midpoint, named $target_2$. This is possible by using the trigonometric deduction equations 6.1 and 6.2 for x-axis and y-axis positioning respectively. The target will be placed at 0.7 m away from the pallet front side. This distance will allow the vehicle to be as close as possible to the targeted pallet and at the same time respect the corridor limits and have enough space for possible corrections when performing the rest of the

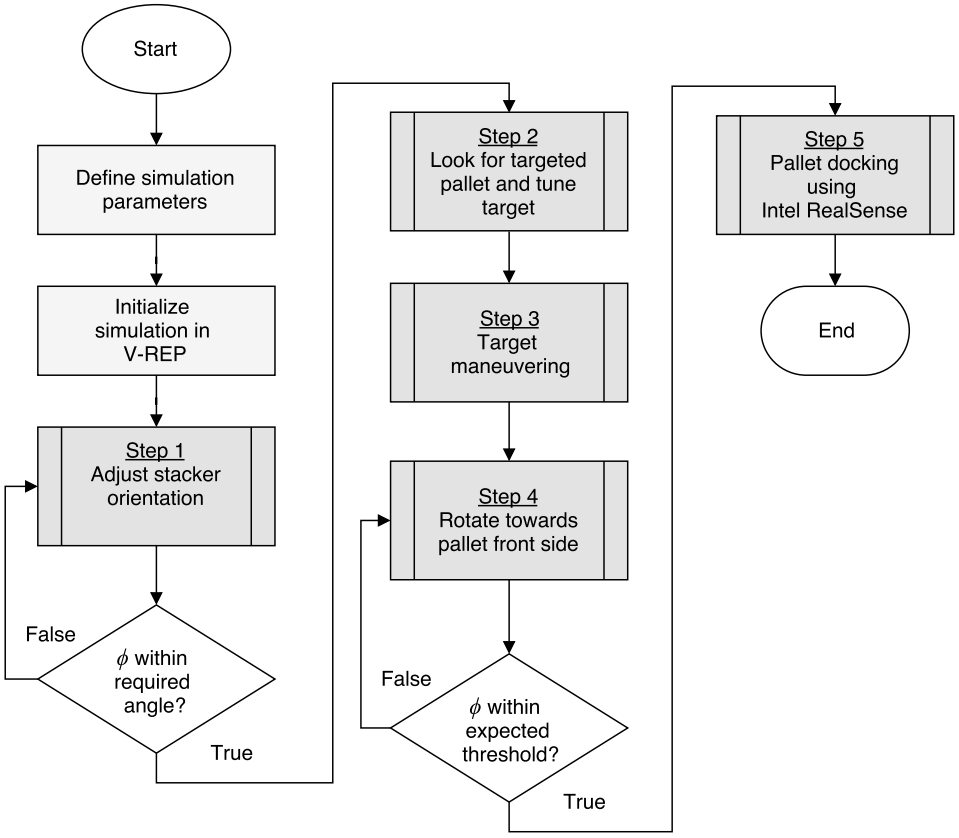

Figure 6.7: Docking maneuver implementation general flowchart.

docking maneuver. The next target to position will be the first target for the stacker to maneuver to, named $target_1$. This target will have the same x-axis position, but will have a ± 1.5 m displacement over the y-axis, that will depend again of the stacker starting position in relation to the targeted pallet.

$$posx_{target_2} = posx_{pallet} + \cos\left(\frac{\pi}{2} - \phi_{pallet}\right) \times 1.3 \tag{6.1}$$

$$posy_{target_2} = posy_{pallet} + \sin\left(\frac{\pi}{2} - \phi_{pallet}\right) \times 1.3 \tag{6.2}$$

As mentioned before the provided pallet position might have an error up to 0.50 m, what will imply corrections before the stacker vehicle maneuvers to the targets positions previously calculated. This error correction will be performed using the 3Vistor-T 3D vision sensor and the pallet features algorithm. The stacker advances at a constant traveling velocity at the same time it looks for the targeted pallet. When a match is found, and knowing the expected position for the targeted pallet the stacker evaluates if it is within the area for docking or not. If it is, the target position is corrected according with the position captured. If it is not, the vehicle discards the pallet found, and proceeds to find the targeted pallet.

6.3.2 Maneuvering to Targets

As soon as the targets are tunned, the stacker is ready to start maneuvering towards them. Using the target orientation and obstacle avoidance dynamics implemented, both targets will aid the stacker vehicle to align with the pallet midpoint. First the vehicle will move forwards to the first target and as soon as the distance between vehicle and target is lower than 0.63 m, the vehicle inverts it's heading direction and proceeds to the second target placed in front of the pallet midpoint previously measured. The vehicle is assumed to meet the target once the distance is lower than 0.05 m. Both threshold distance values were chosen due to multiple experiments performed. The first value is much higher because there is no need for the vehicle to actually equal the position of that target. For the docking maneuver conceptualized, the forks only need to have enough space to maneuver when moving towards the second target.

6.3.3 Pallet Docking

Using the computed data from the processed images gathered from Intel RealSense, it was possible to apply a simple control algorithm to enable the stacker vehicle to properly align with the pallet. The control algorithm is based on the

estimated error between the desired and the actual orientation for the stacker vehicle in relation to the pallet midpoint (equation 6.3). This error is then applied, together with a gain, to the control of both linear and angular velocity for the front steering wheel of the stacker vehicle (equations 6.4 and 6.5).

$$e_{ang} = \sigma - |\theta| \tag{6.3}$$

$$v_{steer} = K_1 \cdot e \tag{6.4}$$

$$\theta_{steer} = -K_2 \cdot e \tag{6.5}$$

The values for K_1 and K_2 will vary depending on the docking maneuver to perform and were determined based on experimentation. With proper tunning, the vehicle will prioritize linear speed in detriment of angular speed or vice versa, depending on the error displacement measured. σ is the reference to obtain, in other words, is the desired angle between the stacker vehicle and the pallet. This last step is broken in two parts. If the stacker is is still out of the defined threshold distance from the pallet midpoint, the vehicle navigates with a constant linear velocity of -0.25 m/s and uses the result from equation 6.5 to infer the angular speed to apply to the front wheel. When the vehicle reaches the expected threshold, equation 6.4 result will provide the linear speed to actuate, and the vehicle will steer $\pm\frac{\pi}{2}$ depending on the error estimated. When the absolute value for the accumulated error of orientation is lower than 1.0°, the vehicle should be properly aligned and ready to fork the pallet. This final step will provide the vehicle a way to fine tune the orientation and positioning before docking the pallet.

Chapter 7

Results and Discussion

This chapter describes and discusses all relevant results related to the docking maneuver implementation presented in the last chapter. The results are broken in two parts: experiments related with the pallet detection approach and experiments for the docking maneuver. The experiments performed for pallet detection were made on two different platforms, inside V-REP simulator and outside the simulator, using one of the cameras for which the algorithm was developed. In the second part, for the docking maneuvers, different starting poses as well as error induction were considered to evaluate the solution robustness.

7.1 Pallet Detection

In this section results related to pallet detection and features retrieval are presented. As mentioned before the experiments undertaken occurred in two very different platforms, one inside a simulation model, the other inside a testing facility using the 3Vistor-T 3D vision sensor.

7.1.1 Pallet Detection inside V-REP simulator

Inside the simulation model created in V-REP, the algorithm was tested for both cameras, the Intel RealSense and the 3Vistor-T, mounted on the stacker vehicle. To test the pallet detection algorithm solution for the Intel RealSense, first the stacker vehicle is placed in front of a targeted finished products pallet. Then two experiments were performed. First moving the stacker vehicle to different positions to evaluate the algorithm distance range limits. And then by

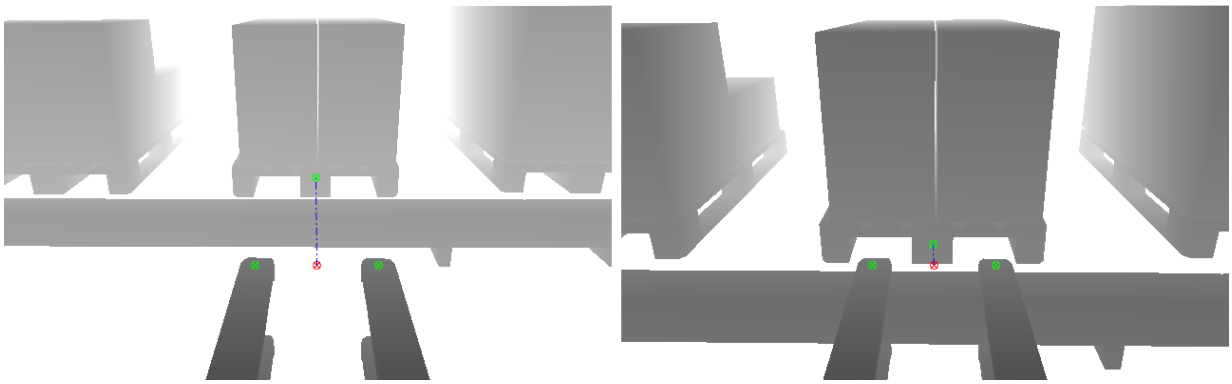

Figure 7.1: Detection results for a pallet at 1.88 m (right) and at 1.30 m away from the stacker Intel RealSense.

Table 7.1: Pallet detection results inside V-REP simulator for Intel RealSense.

Distance RS to Pallet (m)	**Pallet Orientation (º)**	**Detection Status**
1.96 – 1.88	0.0	Failed
1.87 – 1.30	0.0	Successful
1.29 – 1.00	0.0	Failed
1.63	0.0 – 4.7	Successful
1.63	-8.0 – 0.0	Successful

rotating the pallet to detect, to test the algorithm orientation range limits.

The first test results showed that the algorithm worked for a pallet within a distance of 1.87 m and 1.30 m away from the stacker's Intel RealSense. In this range the algorithm was able to retrieve pallet features such as it's pockets, columns and midpoint. In figure 7.1 detection results are shown for two distances at the edges of the detection range.

In the second experiment, the vehicle was placed inside the detection zone previously tested (at 1.63 m away from the targeted pallet front side). The test results showed that the algorithm can detect the pallet from an orientation displacement of -8.0° to 4.7°. The algorithm retrieved correctly the pallet features. In figure 7.2 the detection results are shown for both orientation displacements limits. The results for the Intel RealSense tests are summed up in table 7.1.

Following the methodology implemented before for the Intel RealSense tests, for the 3Vistor-T algorithm tests the

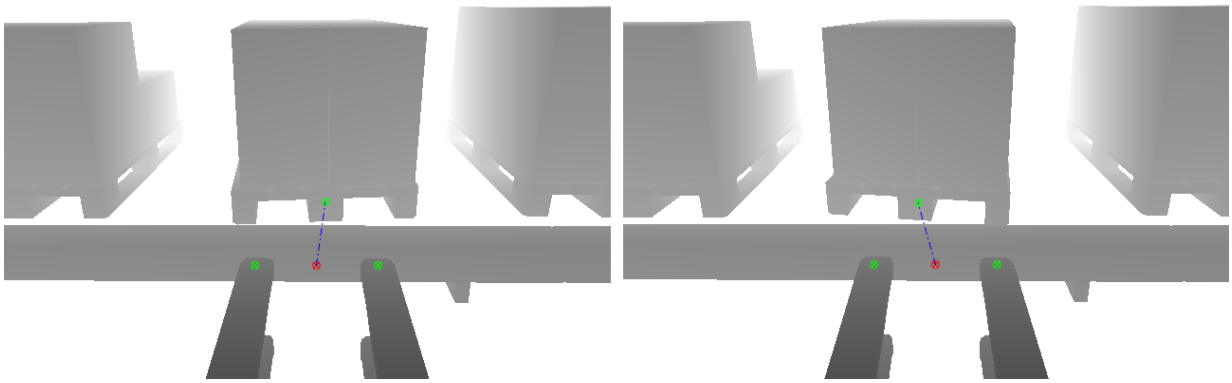

Figure 7.2: Detection results for a pallet with 4.7° (left) and -8.0° (right) orientation displacement.

Table 7.2: Pallet detection results inside V-REP simulator for 3Vistor-T.

Distance Vistor to Pallet (m)	**Pallet Orientation (º)**	**Detection Status**
2.18 – 0.73	0.0	Successful
2.18	0.0 – 14.8	Successful
2.18	-12.3 – 0.0	Successful
0.73	0.0 – 4.7	Successful
0.73	-2.3 – 0.0	Successful

same position and orientation displacement method was used to evaluate the solution robustness. In this case, a new scenario need to be defined, since the vehicle will use this camera panned at $\pm 90°$. Once again, the vehicle is moved to different distances to evaultate the algorithm distance detection range, and then the targeted pallet is rotated to test the algorithm orientation range limits. The results for both experiments are shown in table 7.2. As it is shown, in the first experiment the solution could detect the pallet from all possible tested ranges (the corridor width), going from 2.18 m to 0.73 m away from the targeted pallet. In the second experiment, for a distance of 2.18 m, the vehicle could identify the pallet if the orientation displacement is inside the interval of -12.3° to 14.8°. This value diminished to -2.3° to 4.7° once the stacker is at the minimum distance of 0.73 m. For both tests, the conditions (distance, orientation) shown are the ones that guaranteed a failure or successful experiment. In between those values a "transition area" exists where the results cannot be assumed to be positive or negative.

Figure 7.3: Testing bench created to simulate a real scenario for pallet detection.

Figure 7.4: Standard Euro Pallet used.

7.1.2 Pallet Detection using 3Vistor-T 3D Vision Sensor

For this experiment, a testing bench was developed to simulate the real scenario when the 3Vistor-T camera is used. The camera was mounted on top of a tripod at 1.64 m of the ground, the same as the camera would have on the stacker vehicle. The camera was tilted to around 50°, the same as it is in the docking solution developed. The testing tripod as well as the 3Vistor-T 3D vision sensor, used for this test scenario, is shown in figure 7.3. The pallet used for this test respects the standard EPAL measurements (1200x800 mm), and it is equal to the ones found on Bosch BrgP, pallets for which the solution developed was for. The pallet used is shown on figure 7.4.

The videos were captured using ROS middleware to establish the connection with the camera as well as to record

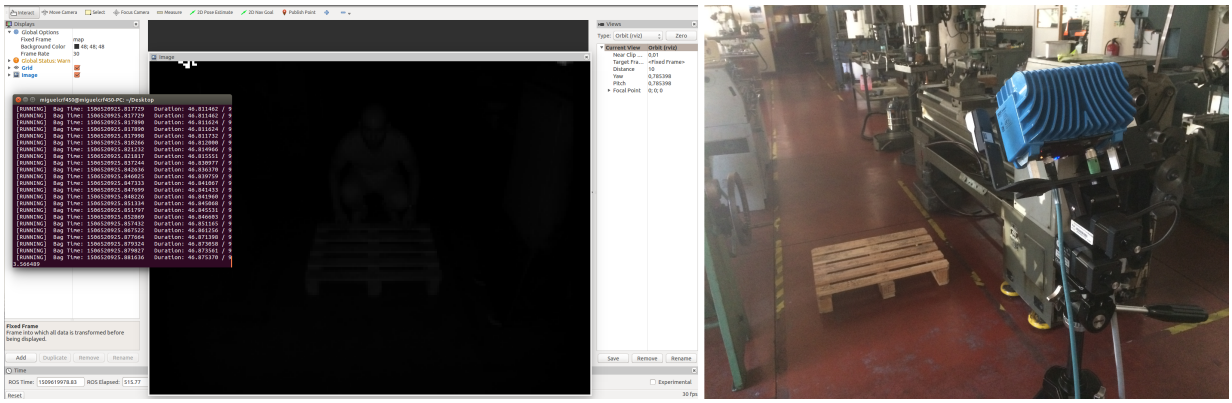

Figure 7.5: Experiment progress. At left the RViz environment, and at right the testing bench.

the depth image videos that the camera provides. The 3Vistor-T creates a series of topics for the different tools it contains, in which only one was chosen to stream the required data on this test. As the experiment progresses, the subscribed topic data stream is stored using a ROS bag. Bags are created by a tool like `rosbag`, which subscribe to one or more ROS topics, and store the serialized message data in a file as it is received. These bag files can also be played back in ROS to the same topics they were recorded from, or even remapped to new topics. Then using RViz, another ROS package that creates a 3D visualization environment, all imaging data required to later process through the MATLAB implemented algorithm was captured. The resulting videos were then processed in MATLAB to evaluate the algorithm robustness. In figure 7.5 the testing bench is shown as well as the data captured in a bag file streaming in RViz.

Two different experiments were performed. One where the pallet started at a predetermined position and was moved to a final position to test the capability to track the detected pallet. The second experiment consisted in moving the pallet to different orientations to once again evaluate the tracking and detection capability of the algorithm. The first experiment results are presented in figure 7.6. The figure shows the experiment progress through the depth imaging captured using the 3D vision sensor. The markers in red display the pallet columns and the green marker the calculated midpoint. The pallet was moved from a starting position of 2.00 m away of the tripod to 1.00 m away. The algorithm detected and isolated the pallet from the staring to position to it's end with success, since the tracked midpoint is placed approximately on the expected position. The same goes for the detected pallet pockets. From the starting point till the end the experiment took 30 seconds. The second experiment results are shown in figure 7.7. The experiment took 42 seconds. The pallet was rotated to an orientation displacement going from approximately -10° to

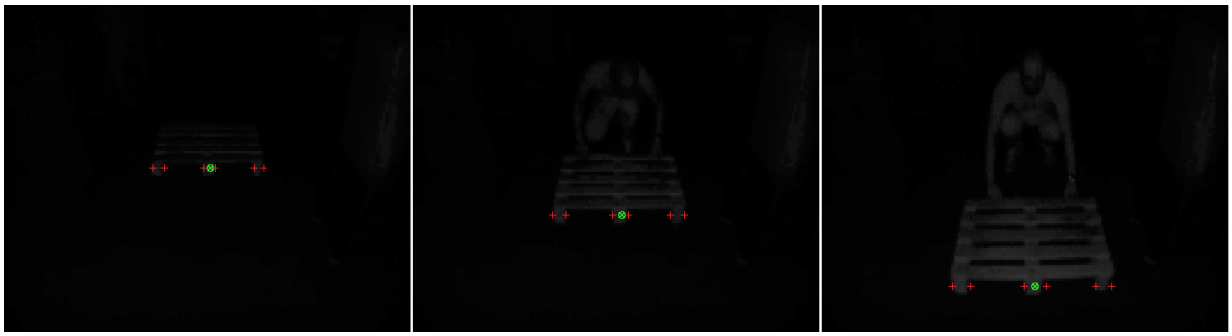

Figure 7.6: Testing results for pallet detection and tracking.

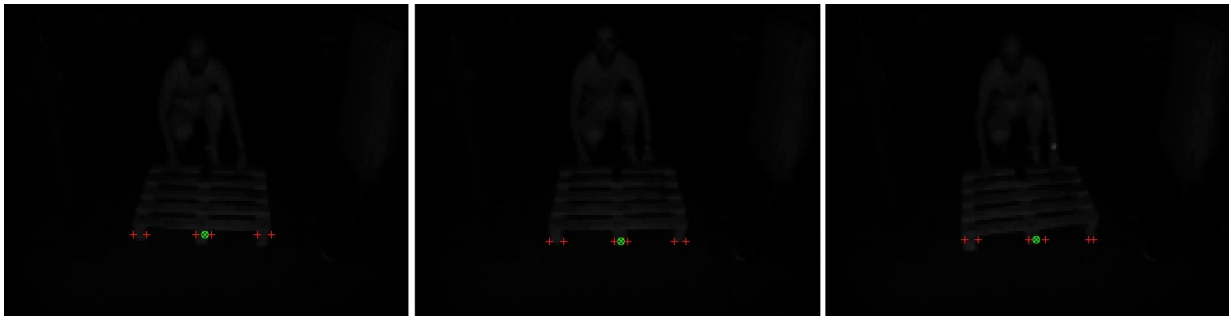

Figure 7.7: Testing results for pallet detection and tracking for different orientations.

10°. The algorithm tracked the pallet from a displacement of -6.2° to 7.8° approximately. The pallet was placed at 1.50 m away from the tripod.

7.2 Docking Maneuver

In this last section, the docking maneuver solution is tested, inside the simulation model created in V-REP, and all experimentations performed will be presented and discussed. The section is divided in different parts: docking maneuvering for predicted starting poses, for extreme starting poses and finally for logistics global information errors. Poses is understood as position and orientation.

7.2.1 Predicted Initial Poses

The stacker vehicle will usually navigate in the finished products corridor close to the limiter on it's right or at the opposing side, going also in the opposing direction. The orientation should be 0° and -180° respectively. In the

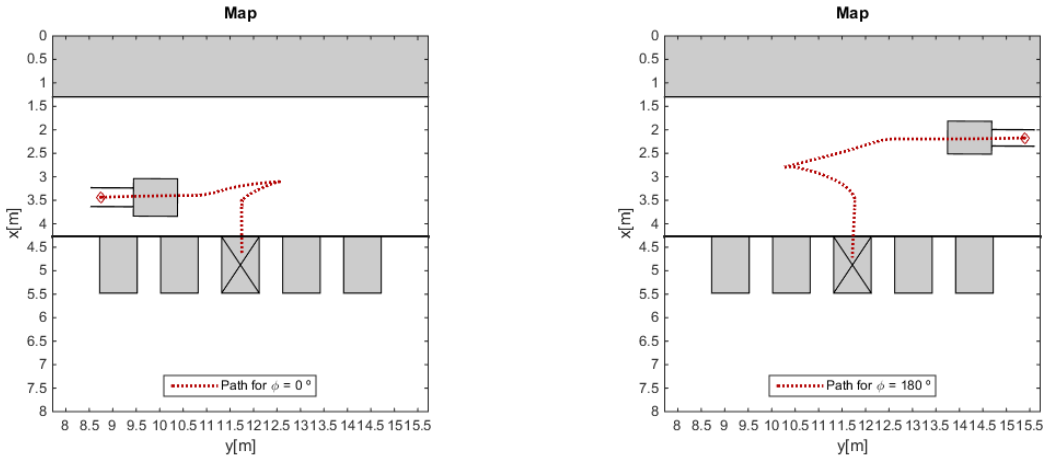

Figure 7.8: Docking maneuvers results for predicted starting poses. Left for the first starting pose, close to the limiter, and at the right for the second starting pose (opposing the limiter). The targeted pallet is crossed.

experiments presented in this part the stacker vehicle will start the docking maneuver in those two predicted poses, and the error considered for the logistics pallet position estimation is considered to be null. For this case scenarios the docking maneuver should be foolproof. The experiments results are shown in figure 7.8, for the first starting pose (close to the limiter) and for the second starting pose (opposing the limiter). In both maneuvers the stacker vehicle successfully docked the targeted pallet (marked with an 'X').

7.2.2 Different Initial Poses

When approaching the docking area, the stacker vehicle might not be navigating in the expected pose. The vehicle might have encountered an obstacle on it's way and had to maneuver around it, changing the starting position as well as the orientation for the docking maneuver. With this considered, the solution must be tested to assure the maneuver can surpass this added complication. Different tests were conducted to assure the maneuver can successfully dock the targeted pallet, despite the starting pose. The stacker was placed in different locations with different starting orientations. The vehicle was placed on the left side of the targeted pallet, with two different starting positions and orientations, for the two first experiments. Then the vehicle was moved to the right side, and once again two experiments were performed for different poses. The results are for the two first experiments are shown in figure 7.9. With a displacement error in orientation relative to the pallet of around 1.2°, the stacker successfully docked the targeted

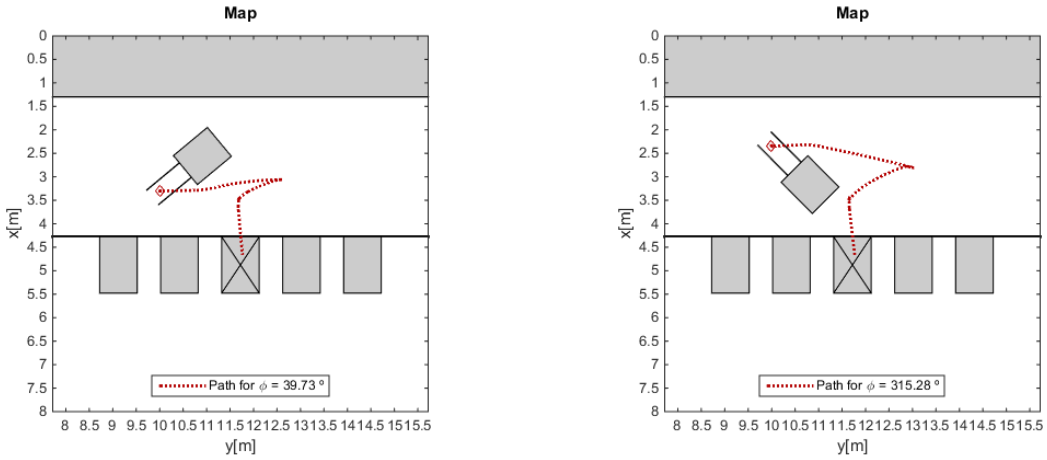

Figure 7.9: Docking maneuvers results for different poses on the left of the pallet. Left for the first starting pose, and at the right for the second starting pose. The targeted pallet is crossed.

pallet. The starting pose for the vehicle was x_0 = 3.30 m, y_0 = 10.00 m, $\phi_0 = 39.73°$ for the first experiment, and x_0 = 2.33 m, y_0 = 9.99 m and ϕ_0 = 39.73° for the second one. For the second two experiments, the results are shown in figure 7.10. Once again the stacker vehicle successfully docked the targeted pallet. The starting pose for the vehicle was x_0 = 3.40 m, y_0 = 13.23 m, ϕ_0 = 129.70° for the first experiment, and x_0 = 2.44 m, y_0 = 13.29 m and ϕ_0 = 214.73° for the second one.

7.2.3 Target Tunning

As it was mentioned before, the expected pallet location provided by logistics could have an error up to $\pm$50 cm in y-axis, or even for x-axis, but considering the workspace disposition, operators must guarantee a correct positioning close to the reference (physical limiter) so the maneuver is valid, mitigating possible x-axis error. In this part tests were conducted to evaluate the solution capability to correct the error, tune the targets and proceed with remaining steps of docking maneuver. Different errors will be introduced ranging from $\pm$20 cm up to $\pm$60 cm (illustrated in figure 7.11).

Two different stacker initial poses will be tested, close to the limiter and at the opposing side. The experiments results are shown in figures 7.12, 7.13 and 7.14, that show the stacker traveling path for different errors, $\pm$20 cm, $\pm$40 cm and $\pm$60 cm respectively. The different results show similar paths despite of the error introduced, since at the second step of the docking maneuver, the vehicle corrects the targeted pallet position and proceeds with a similar

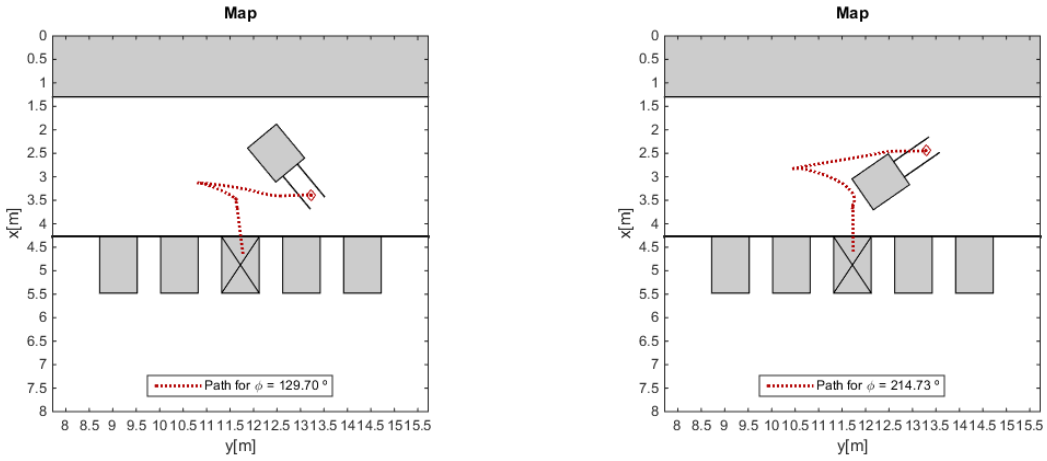

Figure 7.10: Docking maneuvers results for different poses on the right of the pallet. Left for the first starting pose, and at the right for the second starting pose. The targeted pallet is crossed.

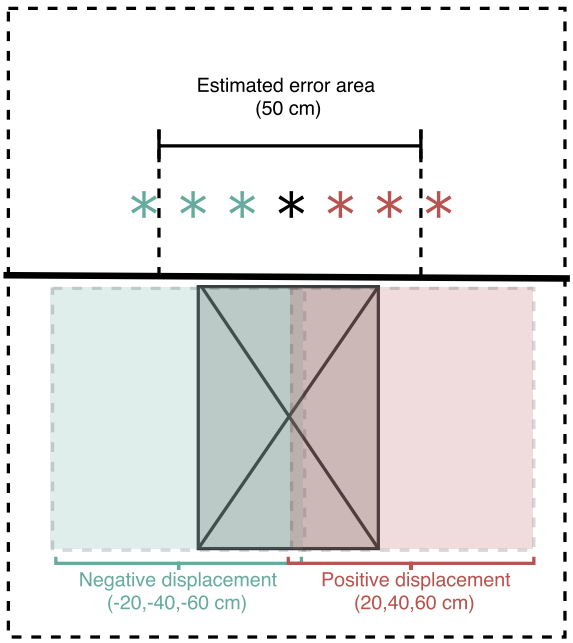

Figure 7.11: The possible different error displacements for the targeted pallet. In green the negative displacements, in red the positive ones. The green and red markers show the pallet expected midpoint target for different displacements ranging from $\pm$20 cm to $\pm$60 cm.

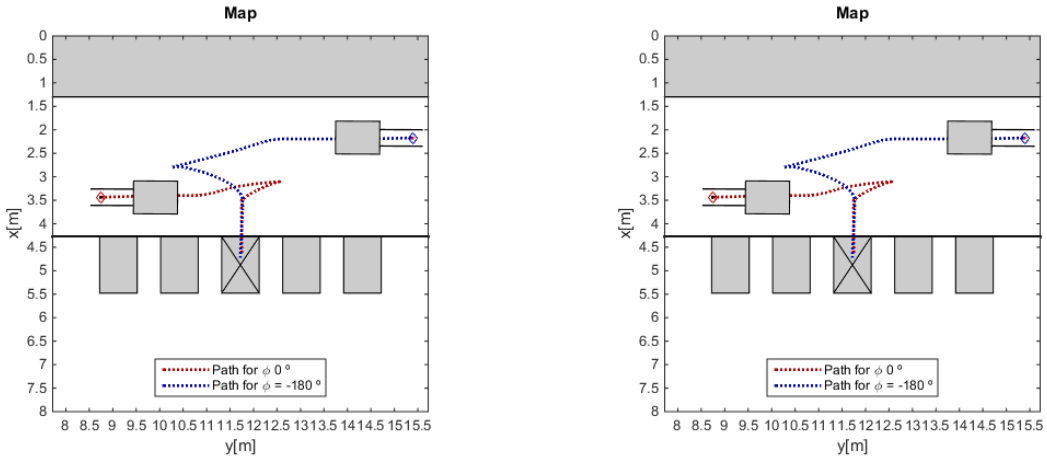

Figure 7.12: Docking maneuvers for target tunning implementation. Left for an error of 20 cm y-axis wise, and at the right for an error of -20 cm y-axis wise. The targeted pallet is crossed.

maneuver, as it was expected. With the experiments performed, the expected error associated with the pallet position provided by logistics can be corrected since positive results were obtained for errors higher than then 50 cm. In the end the solution proved to support errors up to $\pm$66 cm in y-axis wise. When this value is surpassed, the stacker usually docks to the pallet next to the one that is targeted.

7.3 Results Summary

The results shown in this chapter prove that the concept docking maneuver can be implemented. The stacker vehicle surpassed all obstacles and difficulties introduced to evaluate the solution robustness, and successfully docked the targeted pallets. Besides the before presented experiments and tests, more were undertaken, again for different vehicle poses, and the stacker could successfully dock in 88.9% of those experiments. With some tunning of established control variables values, those 6 failed experiments out of 54 undertaken can be surpassed. Tunning those control variables can be avoided, if the algorithm for pallet detection is enhanced. It was noticed in the experimental results that when the stacker is placed in front of the targeted pallet and the displacement position of the rear axle is too far away from the expected position, the adjustments performed using the Intel RealSense might fail. The algorithm should be enhanced allowing a better orientation retrieval from the pallet, for position adjustment. This would allow more flexibility for the stacker and to rely even more on it's vision sensors feedback. The algorithm also lacked performance when the pallet

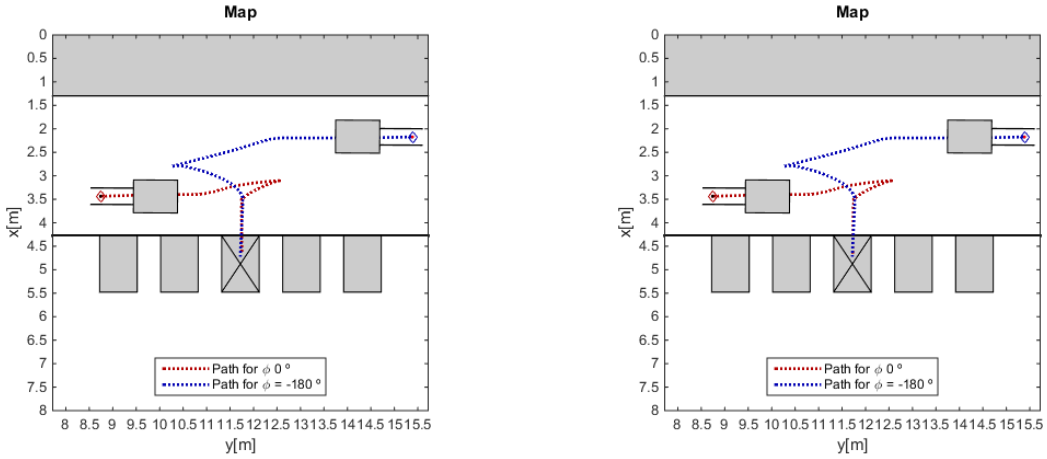

Figure 7.13: Docking maneuvers for target tunning implementation. Left for an error of 40 cm y-axis wise, and at the right for an error of -40 cm y-axis wise. The targeted pallet is crossed.

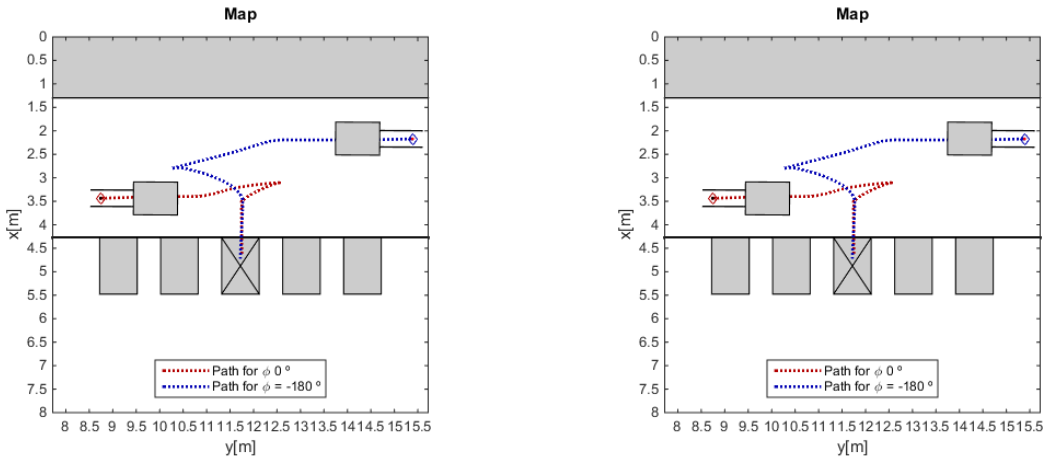

Figure 7.14: Docking maneuvers for target tunning implementation. Left for an error of 60 cm y-axis wise, and at the right for an error of -60 cm y-axis wise. The targeted pallet is crossed.

is too displaced. Still, the algorithm for pallet detection had a good response allowing the proposed solution to succeed with a very significant rate. The maneuvers to be performed follow a series of predefined steps, what can aid their predictability, ensuring a safer workspace for the operators on the factory. The maneuvers respect their workspace boundaries and are flexible enough to allow the vehicle to start them in different poses. With this being said, the established goals for this dissertation project were accomplished.

Chapter 8

Conclusions and Further Research

This last chapter presents the concluding remarks for this dissertation implementation, as well as how the future work should be targeted, to enhance the solution in hands.

8.1 Concluding Remarks

Thanks to Bosch BrgP, this dissertation work was developed for an industrial professional application, with a main focus centered around a real scenario, despite being implemented mainly inside simulation models. In the end, this dissertation work proposed a solution based on different processing steps to solve the docking problem for stacker vehicles. The solution was based on camera processing algorithms for pallet detection and a behavioral based dynamic control for the vehicle maneuvering. The implementation for the proposed solution also admits that the stacker knows it exact location and orientation through the GPS and compass modules to include in the future prototypes.

The solution developed, as the results show, presents goods results for the outlined goals, by dealing with success with the expected docking problems to encounter, like different starting poses or associated errors to the signalized targeted pallet. The implemented dynamics, based on the work of Bicho (1999), were successfully implemented as tested, and was critical to allow the solution to maneuver for different starting poses. The algorithm for pallet detection was also successfully implemented, and the results proved to be sufficient to allow the maneuver to be completed, although some inherent limitations exist as pointed out before.

The orientation and real position retrieved from the pallet is not precise, what could ease the entire maneuver and

reduce possible errors. An enhanced computer vision algorithm could allow the stacker vehicles to rely only on local information, minimizing the possible errors accumulated during the maneuver, when the vehicle relies on information of it's own GPS and compass. In the simulator those errors do not exist but in a real scenario they would, what could somehow compromise the solution.

8.2 Future Work

According to the outlined conclusions presented before, the future work in this docking maneuver implementation should be centered mainly on the computer vision algorithm. Enhancing the algorithm could start by implementing a method for dynamical segmentation, that would allow the vehicle to use the surrounding environment to understand it's orientation and location. The same could be applied to the pallet detection algorithm. Increasing the solution intelligence (p.e. implementing neural networks for patter recognition) or using a new 2D LIDAR to capture the pallet are two different ways to overcomes the inherent limitations. Both proposed solutions would allow the vehicle to create a better region of interest, limiting the computation to a smaller region, and also would allow to dynamically segment the pallet to better retrieve it's features.

With added intelligence, the solution should recognize pallets at different locations and with different orientations in a more effective way allowing the solution to easily isolate the pallet to target. The other solution would require to assemble a 2D LIDAR to some moving part (like the vehicles forks) allowing the vehicle to recognize the pallets features and once again isolate the targeted area in a faster and more effective way. This solution would also require a method for sensor fusion.

Increasing the solution intelligence for pallet pattern recognition would be a better, most suiting and least intrusive way to enhance the developed solution. This solution would undertake possible development costs and would have a steeper learning curve. But as soon as the learning period is surpassed and optimized, the solution would recognize pallets easily in very different and extreme poses, something that might not happen using the 2D LIDAR sensor. Adding a 2D LIDAR sensor would also require hardware costs, since the vehicle is not prepared to mount a new sensor near it's forks.

Bibliography

L. Baglivo, N. Biasi, F. Biral, N. Bellomo, E. Bertolazzi, M. Da Lio, and M. De Cecco. "Autonomous pallet localization and picking for industrial forklifts: a robust range and look method". *Measurement Science and Technology*, 22(8):085502, 2011. doi: 10.1088/0957-0233/22/8/085502.

P. Besl and N. McKay. "A Method for Registration of 3-D Shapes". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. doi: 10.1109/34.121791.

E. Bicho. *"Dynamic Approach to Behavior-Based Robotics: design, specification, analysis, simulation and implementation"*. Shaker, 1999. ISBN 3-8265-7462-1.

R. Bostelman and T. Hong. "Review of Research for Docking Automatic Guided Vehicles and Mobile Robots". Technical report, National Institute of Standards and Technology, New York, 2016.

P. Corke. *"Robotics, Vision and Control - Fundamental Algorithms in MATLAB"*. Springer, 2011. ISBN 978-3-642-20144-8.

N. Correll. "Introduction to Robotics 3: Forward and Inverse Kinematics", 2011. URL http://correll.cs.colorado.edu/?p=896. Last visited 2017-07-21.

J.J. Craig. *"Introduction to Robotics: Mechanics and Control"*, volume 1. Pearson Prentice Hall Upper Saddle River, 2004. ISBN 0201182408.

Dominicos. "Depth Map of an open cubic frame structure made in the proportions of a cube and floor, with the vertices connected by sold bars each with a square cross section.", 2011. URL https://commons.wikimedia.org/wiki/File:Cubic_Structure_and_Floor_Depth_Map_with_Front_and_Back_Delimitation.jpg. Last visited 2017-09-07, Licensing CC BY-SA 3.0.

G. Garibott and S. Masciangelo. "ROBOLIFT: a vision guided autonomous fork-lift for pallet handling". In *Intelligent Robots and Systems '96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, volume 2, pages 656–663, Osaka, Japan, 1996. IEEE. doi: 10.1109/IROS.1996.571028.

D. Herrero, J. Villagra, and H. Martinez. "Self-configuration of waypoints for docking maneuvers of flexible automated guided vehicles". *IEEE Transactions on Automation Science and Engineering*, 10(2):470–475, 2013. doi: 10.1109/TASE.2013.2240386.

D. Houcque. "Introduction to MATLAB for Engineering Students". Technical Report August, Northwestern University, 2005.

T. Huang. Computer Vision: Evolution And Promise. *19th CERN School of Computing*, pages 21–25, 1996. doi: 10.5170/CERN-1996-008.21.

Intel Corporation. "Intel ® RealSense™ Camera R200 Embedded Infrared Assisted Stereovision 3D Imaging System with Color Camera", 2016.

Y. Jun, L. Peihuang, Q. Xiaoming, and W. Xing. "An Intelligent Real-Time Monocular Vision-Based AGV System for Accurate Lane Detecting". In *Proceedings - ISECS International Colloquium on Computing, Communication, Control, and Management, CCCM 2008*, volume 2, pages 28–33, Guangzhou, China, 2008. ISBN 9780769532905. doi: 10.1109/CCCM.2008.87.

F. Kanehiro, H. Hirukawa, and S. Kajita. "OpenHRP: Open Architecture Humanoid Robotics Platform". *The International Journal of Robotics Research*, 23(2):155–165, 2004. doi: 10.1177/0278364904041324.

S. Karaman, M.R. Walter, and E. Frazzoli. "Closed-loop Pallet Engagement in an Unstructured Environment". In *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5119–5126, Taipei, Taiwan, 2010. doi: 10.1109/IROS.2010.5652377.

N. Koenig and A. Howard. "Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator". In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154, Sendai, Japan, 2004. IEEE. doi: 10.1109/IROS.2004.1389727.

D. Lecking, O. Wulf, and B. Wagner. "Variable pallet pick-up for automatic guided vehicles in industrial environments". In *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, pages 1169–1174, Prague, Czech Republic, 2006. IEEE. doi: 10.1109/ETFA.2006.355424.

J. Lee, Y. Shim, and G. Kim. "Autonomous navigation in an agriculture product processing center". In *2014 11th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 619–622, Kuala Lumpur, Malaysia, 2014. doi: 10.1109/URAI.2014.7057506.

J. Lucas, H. Martinez-Barbera, and F. Jimenez. "Multi-objective evolutionary fuzzy modeling for the docking maneuver of an automated guided vehicle". In *IEEE International Conference on Systems Man and Cybernetics Conference Proceedings*, volume 3, pages 2757–2762, Waikoloa, HI, USA, 2005. doi: 10.1109/ICSMC.2005.1571567.

J. McLoone. "Edge detection applied to a photograph", 2010. URL https://commons.wikimedia.org/wiki/File:%C3%84%C3%A4retuvastuse_n%C3%A4ide.png. Last visited 2017-09-07, Licensing CC BY-SA 3.0.

M. Mellado, E. Vendrell, A. Crespo, P. López, J. Garbajosa, C. Lomba, K. Schilling, H. Stützle, and R. Mayerhofer. "Application of a real time expert system platform for flexible autonomous transport in industrial production". *Computers in Industry*, 38(2):187–200, 1999. doi: 10.1016/S0166-3615(98)00118-3.

O. Michel. "Webots TM : Professional Mobile Robot Simulation". *International Journal of Advanced Robotic Systems*, 1(1):39–42, 2004. doi: 10.1.1.86.1278.

C. Moler. "The Origins of MATLAB - MATLAB and Simulink", 2004. URL https://www.mathworks.com/company/newsletters/articles/the-origins-of-matlab.html. Last visited 2017-07-18.

Mukeshhrs. "The Tow Type AGV - IntelliCart: Registered Trade Mark Of Hi-Tech Robotic Systemz Ltd., India", 2009. URL https://commons.wikimedia.org/wiki/File:IntelliCart1.jpg. Last visited 2017-07-21.

J. Nygards, T. Hogstrom, and A. Wernersson. "Docking to pallets with feedback from a sheet-of-light range camera". In *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)*, volume 3, pages 1853–1859, Takamatsu, Japan, 2000. ISBN 0780363485. doi: 10.1109/IROS.2000.895241.

J. Pagès, X. Armangué, J. Salvi, J. Freixenet, and J. Martí. "A Computer Vision System for Autonomous Forklift Vehicles in Industrial Environments". In *Proc. of the 9th Mediterranean Conference on Control and Automation MEDS*, pages 1–6, Dubrovnik, Croatia, 2001.

W.K. Pratt. *"Digital Image Processing - PIKS Inside"*, volume 5. John Wiley & Sons, Inc., 2001. ISBN 0-471-22132-5.

M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Mg. "ROS: an open-source Robot Operating System". In *IEEE International Conference on Robotics and Automation (ICRA) - workshop on open source software*, volume 3, page 5, Kobe, Japan, 2009.

E. Rohmer, S. Singh, and M. Freese. "V-REP : a Versatile and Scalable Robot Simulation Framework". In *Proceedings. 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1321–1326, Tokyo, Japan, 2013. IEEE. doi: 10.1109/IROS.2013.6696520.

G. Schöner, M. Dose, and C. Engels. "Dynamics of behavior: Theory and applications for autonomous robot architectures". *Robotics and autonomous systems*, 16(2-4):213–245, 1995. doi: 10.1016/0921-8890(95)00049-6.

M. Seelinger and J.D. Yoder. "Automatic visual guidance of a forklift engaging a pallet". *Robotics and Autonomous Systems*, 54(12):1026–1038, 2006. doi: 10.1016/j.robot.2005.10.009.

M. Seelinger, J.D. Yoder, E.T. Baumgartner, and S.B. Skaar. "High-precision visual control of mobile manipulators". *IEEE Transactions on Robotics and Automation*, 18(6):957–965, 2002. doi: 10.1109/TRA.2002.805647.

SICK. "3vistor-T CX/AG Datasheet and Quickstart", 2015.

SICK. "S30B-3011GB Product Information and Datasheet", 2016.

M. S. Silberberg. *"Computational Principles of Mobile Robotics"*. Cambridge University Press, 2002. ISBN 9780195142464.

R. Szeliski. *"Computer Vision : Algorithms and Applications"*, volume 5. Springer, 2010. ISBN 1848829345.

R. Tsai. "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses". *IEEE Journal on Robotics and Automation*, 3(4):323–344, 1987. doi: 10.1109/JRA.1987.1087109.

R. Varga and S. Nedevschi. "Vision-based Autonomous Load Handling for Automated Guided Vehicles". In *2014 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 239–244, Cluj Napoca, Romania, 2014. doi: 10.1109/ICCP.2014.6937003.

P. Vaz, R. Ferreira, V. Grossmann, and M.I. Ribeiro. "Docking of a Mobile Platform Based on Infrared Sensors". In *IEEE International Symposium on Industrial Electronics*, volume 2, pages 735—740, Guimaraes, Portugal, 1997. IEEE. doi: 10.1109/ISIE.1997.649089.

J. Villagra and D. Herrero-Pérez. "A comparison of control techniques for robust docking maneuvers of an AGV". *IEEE Transactions on Control Systems Technology*, 20(4):1116–1123, 2012. doi: 10.1109/TCST.2011.2159794.

C Weitkamp. *"Lidar: range-resolved optical remote sensing of the atmosphere"*. Springer Science & Business, 2006. ISBN 978-0-387-25101-1.

Zeonzir. "Output of HMRF-EM segmentation of color image into 3 clusters", 2014. URL https://commons.wikimedia.org/wiki/File:Sample_segmentation_HMRF-EM.png. Last visited 2017-09-07, Licensing CC BY-SA 3.0.