LAPPEENRANTA UNIVERSITY OF TECHNOLOGY
LUT School of Energy Systems
LUT Mechanical Engineering

Farid Alijani

AUTONOMOUS VISION-BASED DOCKING OF A MOBILE ROBOT WITH FOUR OMNIDIRECTIONAL WHEELS

Examiners: Professor Aki Mikkola
Professor Anders Robertsson

ABSTRACT

Lappeenranta University of Technology
LUT School of Energy Systems
LUT Mechanical Engineering

Farid Alijani

Autonomous vision-based docking of a mobile robot with four omnidirectional wheels

Master's thesis

2017

82 pages, 37 figures, 6 table and 2 appendices

Examiners: Professor Aki Mikkola
Professor Anders Robertsson

Keywords: Mobile robot, docking, vision sensor, docking, control design, multiple sensor data fusion.

Docking of mobile robots requires precise position measurements relative to the docking platform to accomplish the task successfully. Besides, the pose estimation of the robot with the sensors in an indoor environment should be accurate enough for localization and navigation toward the docking platform. However, the sensors are entitled to errors due to the measurement uncertainty. In this thesis, sensor data fusion is exploited to decrease the measurement errors and increase the accuracy of the docking. Different approaches are employed in this thesis for the sensor fusion to investigate the precise docking of the mobile robot. Laser scanners, vision sensors and reinforcement learning are evaluated to find the optimal approach for docking of the mobile robot. The final approach is a reinforcement learning framework to investigate and compare the optimality of the docking with the vision-based control method in which training is handled in the simulation environment with the reward distribution.

ACKNOWLEDGMENT

I would like to express my gratitude to my examiner Anders Robertsson and supervisors, Björn Olofsson and Martin Karlsson at the department of Automatic Control at Lund University, for their precious support during my MSc thesis and freedom they gave me to go beyond the initial proposed topic to investigate the model-free RL as extra part of this project. It would not have been possible to conduct this research without their help. I highly appreciate all the practical ideas about docking of the mobile robot from them.

I would also gratefully thank great employees of the Robotics Lab at LTH who provided generous help with the Rob@work 3, software, and hardware connections and other physical setups during the course of this research.

Farid Alijani

Lappeenranta 26.02.2017

TABLE OF CONTENT

ABSTRACT

ACKNOWLEDGMENT

TABLE OF CONTENT

LIST OF SYMBOLS AND ABBREVIATIONS

1 INTRODUCTION 10

1.1 Background 10

1.2 Motivation 11

1.3 Objectives 11

1.4 Mobile Robot Platform 12

1.5 Software and Hardware Integration 12

1.6 Thesis Outline 15

2 METHODOLOGY 16

2.1 Autonomous Laser Scanner-based Docking 16

2.2 Computer Vision 19

2.2.1 Geometric Representation and Transformation 20

2.2.2 Pose Estimation 21

2.3 Fiducial Markers 25

2.3.1 Point Fiducial 26

2.3.2 Planar Fiducial 28

2.4 Autonomous Vision-based Docking 30

2.4.1 Kinematic Model 30

2.4.2 Control Design 32

2.5 Reinforcement learning 38
2.5.1 Background 38
2.5.2 Problem 40
2.5.3 Solution 40
3 ANALYSIS OF RESULTS 45
3.1 Autonomous laser scanner-based docking 45
3.2 Computer Vision 52
3.2.1 Camera and marker frame 52
3.2.2 Camera Calibration 53
3.3 Vision-feedback Control Design 55
3.4 Reinforcement Learning 66
4 DISCUSSION 71
5 CONCLUSION 75
LIST OF REFERENCES 78
APPENDIX
Appendix I: Camera Calibration
Appendix II: Project Source Code

LIST OF SYMBOLS AND ABBREVIATIONS

Δt	Time difference
α	Learning rate
β	Norm
γ	Discount factor
θ	Counter clockwise angle with respect to horizontal axis
θ^{w}	Rotation angle of robot wheel
θ^{w_i}	Orientation of each wheel
θ_b	Beam angle
θ_{CAM}	Camera orientation of yaw
θ_{inc}	Angle increment
θ_{mar}	Estimated orientation
θ_{mar}^{ref}	Marker reference orientation
θ_{max}	Maximum angle
θ_{min}	Minimum angle
θ_{Rob}	Robot orientation of yaw
θ_{thresh}	Docking threshold for orientation
$\dot{\theta}_{Rob}^{CS}$	Angular velocity as control signal
φ_{CAM}	Camera orientation of pitch
φ_{Rob}	Robot orientation of pitch
ψ_{CAM}	Camera orientation of roll
ψ_{Rob}	Robot orientation of roll

$C_o(u_o, v_o)$	Principal point
dt	Iteration time
e(t)	Error signal
i	Index
K_d	Derivative gain
K_i	Integral gain
K_p	Proportional gain
lb	Number of laser beams
P_{cur}	Current position of camera
P_{mar}	Estimated position
P_{mar}^{ref}	Marker reference position
P_{ref}	Reference position of camera
$\dot{P}_{Rob}^{CS}$	Linear velocity as control signal
$Q(s_t, a_t)$	Q-value for specific state and action
r	Radial distance to objects
$R(s_t, a_t)$	Reward for specific state and action
rob_{cmd}	Robot control signal
$R_{3\times3}$	Rotation matrix
R_{CAM}	Camera orientation matrix
$\vec{r}_{mar}$	Marker rotation vector
T_{CAM}	Camera translation
t_f^{ROS}	Final time
t_i^{ROS}	Initial time
t_{real}	Vector of real time

t_s^{ROS}	Recorded sampling time
$Vel_x^{App.}$	Driving velocity
Vel_x^{SM}	Finishing velocity
x^{w_i}	x-axis of local wheel coordinate frame
X_{CAM}	Camera pose estimation
X_{mar}	Marker pose estimation
x_{mar}^{ref}	x-axis of reference values of marker coordinate system
x_{thresh}	Radius of threshold along marker frame
x_{Rob}	x-axis of robot coordinate system
y^{w_i}	y-axis of local wheel coordinate frame
y_{CAM}	y-axis of camera coordinate system
y_{mar}	y-axis of marker coordinate system
y_{mar}^{ref}	y-axis of reference values of marker coordinate system
y_{Rob}	y-axis of robot coordinate system
y_{thresh}	Docking threshold for position
z_{mar}	z-axis of marker coordinate system
z_{Rob}	z-axis of robot coordinate system

CG	Center of gravity
fps	Frame per second
GUI	Graphical user interface
HSV	Hue, saturation and value color space
OpenCV	Open source computer vision
PBVS	Position-based visual system

RGB	Red, green and blue color space
RL	Reinforcement learning
ROS	Robot operating system
SM	Safety margin
TD	Temporal difference

1 INTRODUCTION

This MSc thesis addresses the extracted information of the measured data from different sensors for docking a mobile robot. Sensor data fusion is exploited to accomplish the precise docking. The control system is designed for sensors to instantly evaluate the current state of the mobile robot on the docking area and the target. The docking area includes the mobile robot and the platform at which another robot is mounted (see Figure 1).

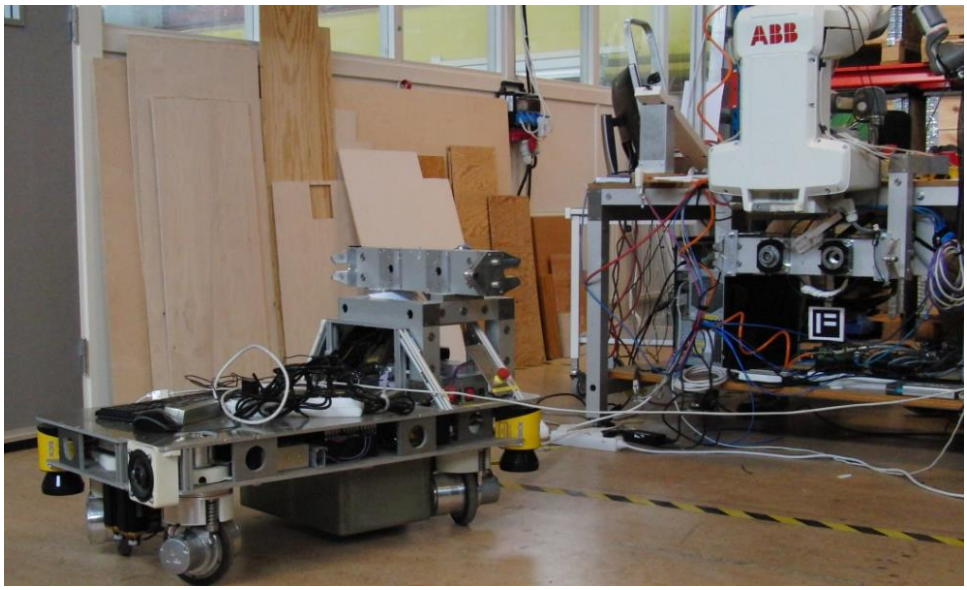


Figure 1. Mobile robot and docking platform.

1.1 Background

Docking is a crucial task for the industries employing mobile robots with the platforms to be lifted or moved temporarily. It is also critical for the applications in which the mobile robots recharge its batteries autonomously (Kim et al. 2005, p. 287). Moreover, the autonomous operation is highly preferable in robotics if it eliminate the human interactions. The autonomous behavior makes the mobile robot accomplish the certain tasks precisely (Hachour, 2009, p. 127). For this purpose, the mobile robot should be equipped with sensors to first localize itself in an unknown environment then achieve the further goal configurations.

However, the sensor measurements are not flawless and can be inconsistent. Therefore, data fusion of the multiple sensors could perceive higher accuracy in the docking.

1.2 Motivation

The sensor integration and data fusion are crucial to increase the versatility and the application domains of the mobile robots since it manipulates environment to achieve different tasks (Hutchinson, Hager & Corke 1996, p. 651). It is important in robotics if the robot is required to accomplish its tasks autonomously and the human control of the actions is eliminated.

Time is a crucial parameter in robotics and the robot is required to accomplish the docking task in the shortest possible time. However, fulfilling merely time-oriented requirements influence the pose estimation accuracy or the environment compatibility which may cause irreversible damages on the workplace. The control system is an applicable tool to obtain the desired skills to perform the docking tasks and satisfy all the constraints concurrently.

1.3 Objectives

The end-effectors and the mechanical collinear joints mounted on the mobile robot and the docking platform demand a precision of 1-3 mm for the position and 1-3 degrees for the orientation to accomplish the docking task. Multiple sensor data fusion is investigated to evaluate the smoothness and the pose estimation accuracy of the sensors with respect to the docking platform.

However, the multiple sensors are not the only aspect of the high accuracy in the docking of the Rob@work 3. The control method which has the supervisory feedback of the sensor measurements is a bridge between the sensor data and the desired docking behavior. The autonomous docking of the mobile robots demands the accurate sensor measurements cooperating with the control design. In this thesis, the feasibility of the sensor integration of the laser scanners and the vision sensors are investigated to obtain the optimal docking policy for the docking trajectory within the less average docking time.

1.4 Mobile Robot Platform

The mobile robot used in this project is Rob@work 3, designed at the Fraunhofer Institute for Manufacturing Engineering and Automation (Fraunhofer-Gesellschaft 2009) and developed for further research in the Robotics Lab at Lund University. It has three degrees of freedom which gives the robot two-dimension transformation and one-dimension rotation for planar motion. An omnidirectional platform enables the Rob@work 3 to employ arbitrary velocity and rotational commands. Besides, it can move freely in constrained places while controlling the kinematic chain of the platform along all directions (Connette et al. 2009, pp. 4124-4125).

The mobile robot platform contains physical components, including sensors, actuators and an on-board computer. A desired performance demands a desirable control for driving and steering of actuators as well as other components and setups on the docking platform such as end-effectors and collinear joints.

The on-board computer operates Linux Ubuntu 14.04 LTS and mounted on beneath of platform. It is responsible to bring up nodes, runs image processing and transforms commands to actuators to move robot towards the goal configuration. (Ubuntu 1991.)

1.5 Software and Hardware Integration

In this thesis, several hardware and software are employed to design the control system to achieve the precise docking. The omnidirectional platform, the onboard and the local computer, the laser scanners, and the cameras are among the subsystems whereas the open source C++ interfaces such as the open source computer vision (OpenCV) and the Qt-Creator (Qt 1994) are the software involved in this project. The MATLAB software also employed to plot the obtained results of the different sensors. All the interfaces operate under the robot operating system (ROS) as an open source project platform.

Figure 2 illustrates the designed software and hardware architecture for docking the Rob@work 3 in this project. The Rob@work 3 has two laser scanners that enable the robot to perceive its environment with surrounding objects. The vision sensors are the android phone and the USB camera which added to the front side of the robot to embed the live video of the

docking area for the image processing and the marker detection. The Rob@work 3 can move freely in all directions with the joystick.

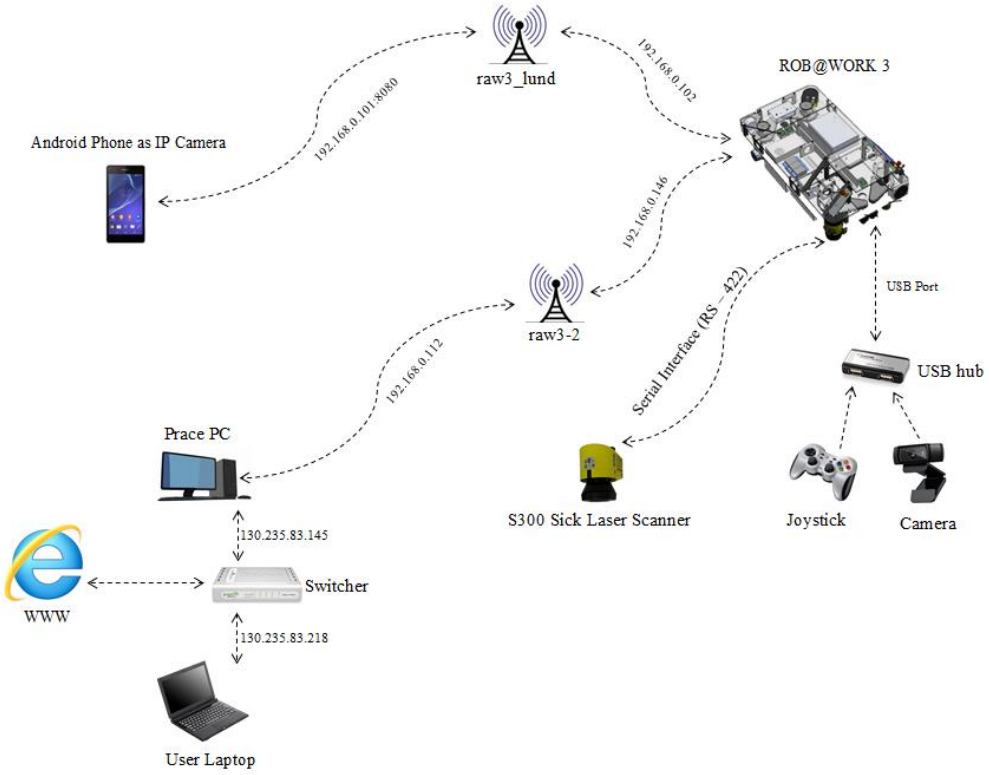


Figure 2. Sensor Integration for docking the Rob@work 3.

Figure 3 depicts the graphical user interface (GUI) developed as the core of the platform in the Qt creator software to switch between pose visualization in the three-dimensional visualizer of the ROS framework, called Rviz, as a module in the open graphic library and the vision-oriented docking.

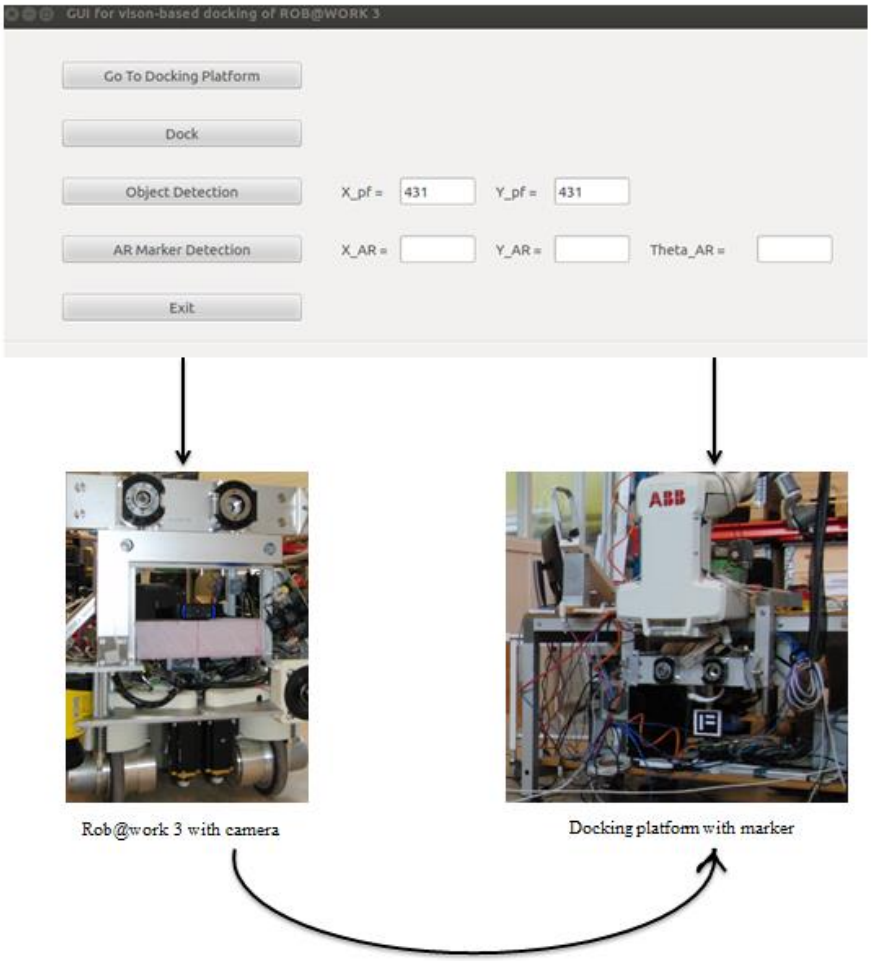


Figure 3. Hardware and software architecture for docking the rob@work 3.

The first layer of the software is written in the C++ framework as the main program of the video streaming, the image processing, the controller/reinforcement learning, the localization and the navigation. The position of the robot is estimated in an indoor environment and rendered with the three-dimensional vector graphical arrows in the Rviz, whereas the vision sensors detect the marker. The visualization with different cameras, the obstacle detection with laser scanners and the vision-feedback control are among the further layer of the program to compute the robot commands and send them to the actuators.

1.6 Thesis Outline

Section 2.1 investigates docking of the mobile robot with laser scanners mounted diagonally on its platform. Various markers with different shapes are investigated to check the feasibility of precise docking. Section 2.2 deals with computer vision to investigate the geometry of different coordinate systems employed in this thesis, compute camera parameters for calibration, and pose estimation. Section 2.3 describes different fiducial markers, including point and planar markers, used in vision-based docking and computer vision methods for marker detection in the OpenCV library. Section 2.4 develops a vision-based control design with a feedback control of augmented reality marker attached to the docking platform. Section 2.5 simulates a model free Q-learning approach for docking. A virtual grid for Q-matrix is developed to find an optimal action selection policy while the robot obtains the highest possible rewards.

Chapter 3 presents the obtained results of several conducted experiments for docking the mobile robot with radial laser scanners, vison-feedback control system and machine learning method. Chapter 4 and 5 discuss the obtained results and compare different employed methods in this project to find the most optimal solution for the docking problem.

2 METHODOLOGY

In this chapter, multiple sensors are employed to find out how to accomplish the docking of the mobile robot with high precision. Designing a control system with sensor integration is prioritized since sensor data fusion yields more accurate result.

2.1 Autonomous Laser Scanner-based Docking

Knowing the position of the robot in a known or an unknown environment is a critical aspect since it is used for further goal investigations and accurate pose estimation makes the goal achievement more robust. By means of the motion sensors with which the mobile robot is equipped, approximate pose estimation is obtained over time relative to a starting point. However, the large position errors make this approach unreliable and demand other sensors to roughly estimate the current position of the robot within an indoor environment. Two-dimensional laser scanners are widely used to obtain the pose estimation of the mobile robot in the environment. (Nguyen et al. 2007, p. 97.) Laser scanners has a capability of the more accurate measurement to detect and avoid obstacles in walking, line following and position estimation in the indoor environment (Teixidó et al. 2012, p. 16483).

The laser scanners in this project are two radial safety laser scanners S300 manufactured by SICK AG (S300 Standard 2016) mounted diagonally opposite on the robot to monitor dangerous areas indoors. The benefit of such design is to implement protective fields on the robot for all-round protection in all directions. Therefore, if the robot moves to hazardous states in the indoor environment, the control system which is established based on sensor measurement will avoid collision.

Besides, two simple shaped objects such as a cylindrical bottle ($r = 7.5\,\text{cm}$) and a box ($30 \times 23.5 \times 21\,\text{cm}^3$) have been used as markers and placed on the reference area for analyzing the raw data obtained from laser scanners. There are two different cases for using markers in this context. First, the robot is fully docked and the cylindrical marker is placed in front of the laser scanner sensor to get the approximate target position. The second case,

however, represents the robot moving towards the docking platform while the marker is still placed in the reference point. There is a minimum proximity range of 10 cm for obstacle detection on the sensor defined to protect the robot from collision.

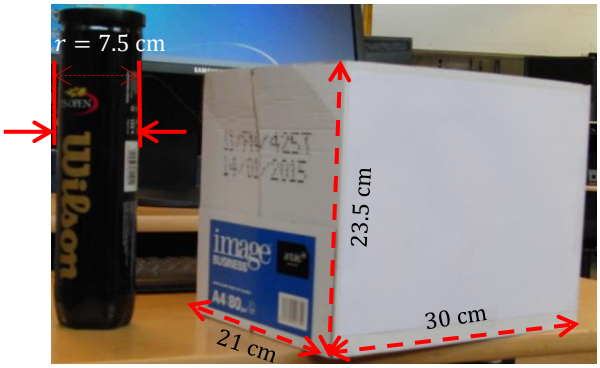
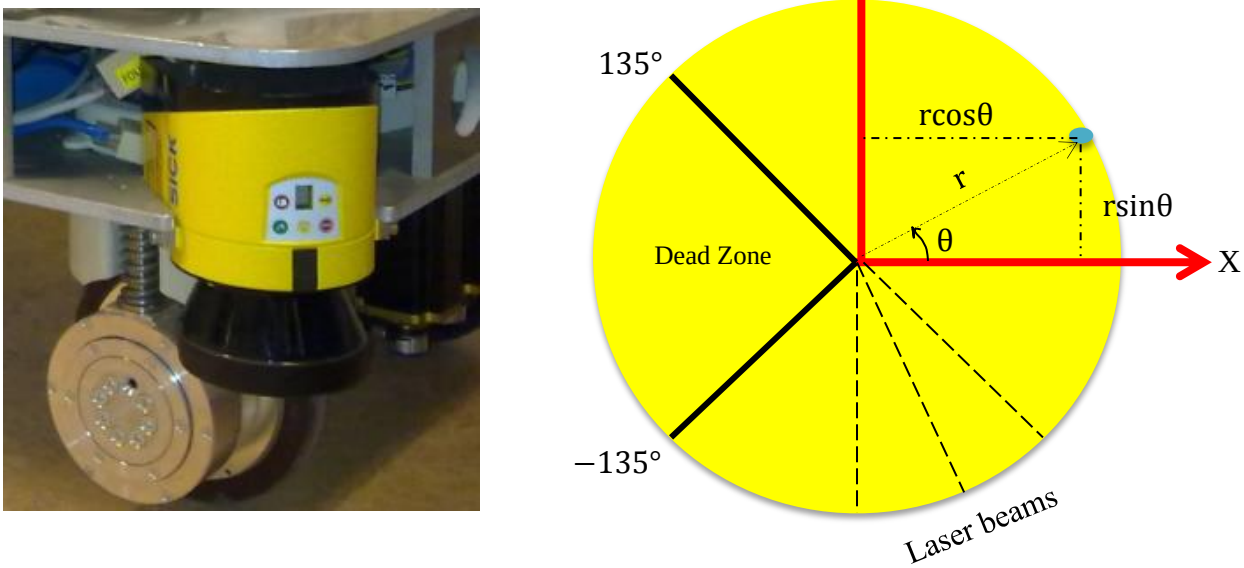


Figure 4. Different markers investigated for docking with the laser scanner sensor, circular bottle (left), box (right).

The SICK S300 is a two-dimensional radial laser scanner operating at 12V and 1A that measures 535 distance points and corresponding 534 laser beams within the range of $[-135°, 135°]$ where $\theta = 0°$ points to front side of the sensor and the middle laser beam. It is called radial laser scanner since the area to be monitored is scanned radially. The S300 cannot see through the objects during this process. The area behind the object is called dead zone since nothing is monitored. Besides of an angular range, the sensor has also a position range which is considered as distance to the obstacles and measured in meter.



a) Front view. b) Geometrical representation.

Figure 5. Sick S300 laser scanner.

The first approach is to employ the radial laser scanners already mounted on the cross-side of the Rob@work 3 to investigate the precision of the measurements. Figure 5 b) depicts the geometrical representation of the S300 laser scanner to detect the obstacles. The detected obstacle is graphically drawn as a blue point with specific distance and angle on the plane. The radial position of the objects can be obtained in the two-dimensional workspace (x, y) as follows:

$$(x, y) = (r\cos\theta, r\sin\theta) \quad (1)$$

in which r is radial distance to objects and θ is counter clockwise angle with respect to horizontal axis while laser scanner perceives the environment. The laser beam indicates an angle increment (θ_{inc}) with a constant value and is represented with an index (i) within the computation range of $[-135°, 135°]$ in which $i = 0$ resembles to $\theta_{inc} = -135°$. i, however,

needs to be declared as a beam angle (θ_b) in the process for simplification. The following formulas convert i to the actual $\theta_b$of the laser scanner:

$$\theta_{inc} = (\frac{\theta_{max} - \theta_{min}}{lb}) \quad (2)$$

In which lb refers to the number of laser beams and θ_{min} and θ_{max} refer to the minimum angle and maximum angle, repectively.

$$\theta_b(i) = \theta_{min} + (i \times \theta_{inc}) \quad (3)$$

Therefore, with θ_{inc} of roughly 0.505 and $lb = 534,\ \theta_b$ is approximated as:
$\theta_b = [-135, -134.494, -133.988 \ \ldots \ 133.988, 134.494, 135]$.

Docking of the Rob@work 3 with laser scanner demands a detectable marker attached to the docking platform, act as a reference point and identify as an obstacle. The laser sensor perceives the environment to detect all possible objects in the front side within the angular range of $[-135°, 135°]$. Backside of the laser scanners, is called dead-zone, is not the matter of interest since the robot is always heading toward the docking platform, which is located on the front side.

2.2 Computer Vision

Similar to eyes, cameras are helpful and effective in robotics since vision allows noncontact measurement in various domains, including object recognition, localization and manipulation (Corke 2013, pp. 221-222). In this project, the camera is practically employed as a vision sensor mounted on the mobile robot to sense the environment and retrieve the visual information of the docking platform. The position of the camera is totally independent of which control configuration is employed as long as it visualizes the local environment, however, it is necessary to calibrate camera before the visualization process started and determine the geometric parameters of the camera. In this project, the image processing has two arguments for the visualization process, the calibration file and the marker size.

2.2.1 Geometric Representation and Transformation

An analytical geometry also known as coordinate geometry is used to determine the camera correlation with the captured marker in a fixed coordinate system in which pose estimation of the robot with its onboard camera are determined accordingly. In computer vison, the geometric camera parameters, including intrinsic and extrinsic matrices, are calculated for the calibration process. (Forsyth & Ponce 2012, pp. 120-123.) The geometric modeling in this project is categorized to three different coordinates in real-time, collaborating in localization and docking. Figure 6 illustrates the marker coordinate frame (x_{mar}, y_{mar}) which is attached to docking platform, the robot coordinate frame $(x_{Rob}, y_{Rob}, \theta_{Rob})$, and the camera coordinate frame $(x_{CAM}, y_{CAM}, \theta_{CAM})$ mounted to the front side and center of gravity (CG) of the Rob@work 3.

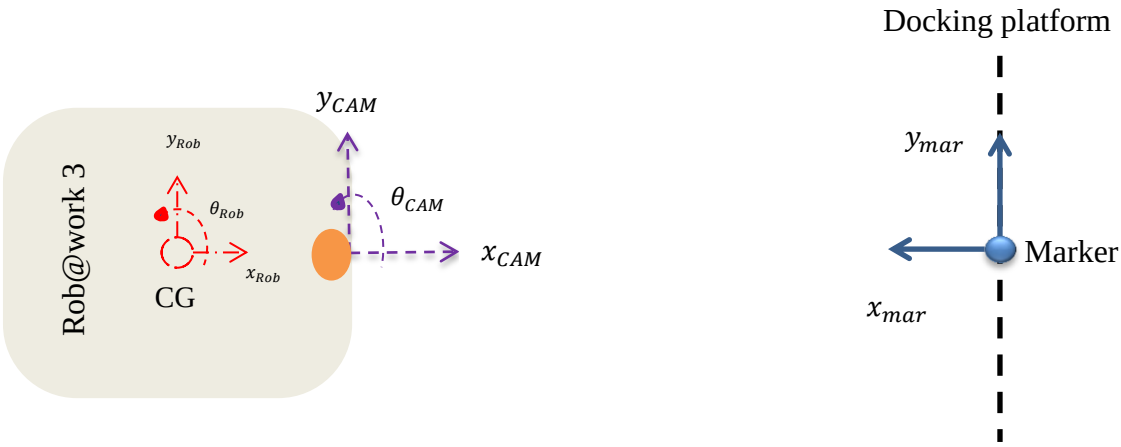


Figure 6. Top view graphical representation of robot, camera and marker frame.

According to Figure 6, y_{CAM} , y_{Rob} and y_{mar} are chosen such that they all have same direction to decreases the problem complexity in terms of the coordinate adaptation. Such an assumption also simplifies the control signal computation after the pose estimation.

The advantage of such coordinate system is that orientation would enable the robot to approach the docking platform from different angles rather than only a primitive perpendicular case in which the robot is meant to approach the goal directly and the orientation is usually adjusted by the operator. Besides, vision feedback control is necessary to control orientation

after extracting measurement of the sensor. Therefore, it facilitates for the robot to approach the docking platform from more than one configuration.

2.2.2 Pose Estimation

In this project, the camera pose estimation (X_{CAM}) with respect to the fixed marker on the docking platform is employed for the navigation and the localization to accomplish the docking without collision and misalignment. The vision sensor and marker are two main components in which the target is identified by the image features of the marker. Both of the components have coordinate frames in which their origins and rotations are calculated with respect to the Rob@work 3. The target is identified by a fiducial marker (Section 2.3) attached to the docking platform and used as a reference of the sensor measurements.

As graphically shown in Figure 6, the target is assumed to be at the origin of the marker to simply initialize the reference values of marker coordinate system ($y_{mar}^{ref} \cong 0$ and $\theta_{mar}^{ref} \cong 0°$). Both camera and marker frames follow the right hand rule for orientation conversion in three dimensions considering the fact that z_{CAM} points upward, whereas z_{mar} points downward. The direction of y_{CAM} and y_{mar} are considered the same to simplify the computation of magnitude and direction of control signal along y_{Rob}-axis. The extrinsic matrix indicates the correlation of the camera in the fixed marker frame which is fixed to the docking platform (Forsyth et al. 2012, p. 132). The local camera coordinate system needs to be expressed in the fixed marker frame on the docking platform.

Marker translation is a 3×1 vector $[x_{mar}, y_{mar}, z_{mar}]^T$ in which the z_{mar} is eliminated on the plane, therefore its position (x_{mar}, y_{mar}) is used to define the marker frame. On the contrary, a marker rotation vector ($\vec{r}_{mar}{}^T = [r_x, r_y, r_z]$) is converted to the rotation matrix ($R_{3\times3}$) to determine the orientation of the camera with respect to the fixed marker. $R_{3\times3}$ is derived according to Rodriguez formula which has a ready-made function in this library (OpenCV 2011a).

$$\beta = |\vec{r}_{mar}| = \sqrt{r_x^2 + r_y^2 + r_z^2} \tag{4}$$

$$R_{3\times3} = \cos\beta I_{3\times3} + (1-\cos\beta) r_{mar} {r_{mar}}^T + \sin\beta \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix} \quad (5)$$

in which β is the norm and $R_{3\times3}$ can be simplified as follows:

$$R_{3\times3} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}. \quad (6)$$

Therefore, camera orientation of roll (ψ_{CAM}), camera orientation of pitch (φ_{CAM}) and camera orientation of yaw (θ_{CAM}) is computed with respect to the fixed marker as below (LaValle 2006, pp. 97-100):

$$\psi_{CAM} = tan^{-1}\left(\frac{r_{21}}{r_{11}}\right) \quad (7)$$

$$\varphi_{CAM} = tan^{-1}\left(\frac{-r_{31}}{\sqrt{r_{32}^2 + r_{33}^2}}\right) \quad (8)$$

$$\theta_{CAM} = tan^{-1}\left(\frac{r_{32}}{r_{33}}\right). \quad (9)$$

In this project, ψ_{CAM} and φ_{CAM} are not applicable since the robot and the camera accordingly rotate neither along the x_{Rob}-axis nor y_{Rob}-axis. The configuration is such that positive angle means counter clockwise rotation, whereas clockwise rotation corresponds to negative orientation. Furthermore, if the robot is placed perpendicular to the platform, the orientation is approximately zero which corresponds to $\theta_{CAM} \cong 0°$.

The Rob@work 3, depicted in Figure 7, has a six-dimensional coordinate system with three-dimensional position ($x_{Rob}, y_{Rob}, z_{Rob}$) and three-dimensional robot orientation of roll (ψ_{Rob}), robot orientation of pitch (φ_{Rob}) and robot orientation of yaw (θ_{Rob}) of the Rob@work 3.

However, since it moves on a plane, movement along z_{Rob}-axis and orientations along x_{Rob}-axis and y_{Rob}-axis are neglected. Therefore, it has three degrees of freedom on the plane.

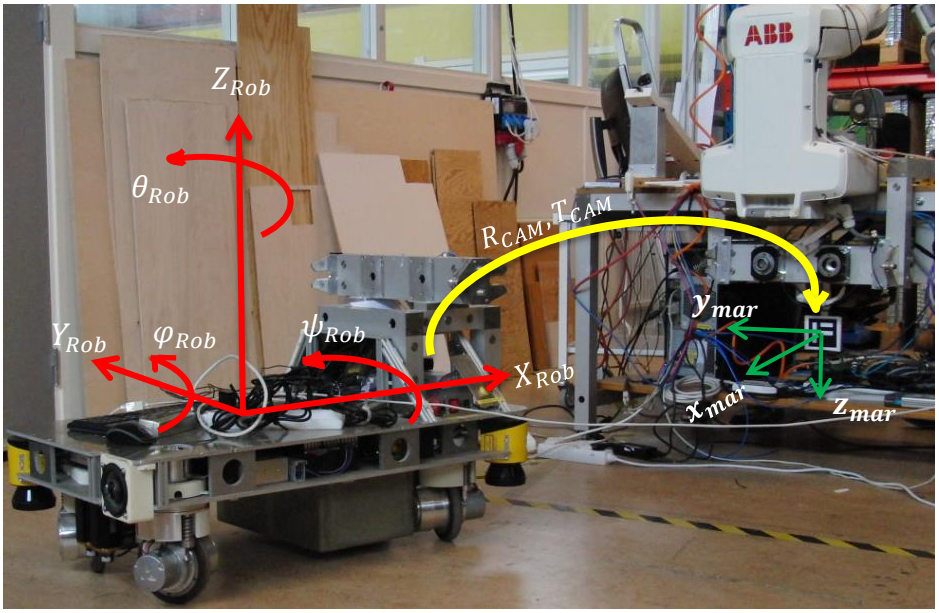


Figure 7. Rob@work 3 and docking platform coordinate systems.

The marker pose estimation (X_{mar}) is related to X_{CAM} as bellow:

$$X_{mar} = R_{CAM}X_{CAM} + T_{CAM} \tag{10}$$

in which $X_{mar} = [x_{mar}\ y_{mar}\ z_{mar}]^T$ and $X_{CAM} = [x_{CAM}\ y_{CAM}\ z_{CAM}]^T$. Camera translation ($T_{CAM}$) and rotation of the camera (R_{CAM}) are calculated with respect to the fixed coordinate system set on the marker attached to the docking platform as graphically sketched in Figure 7. In this project, the camera is practically mounted on the front and middle side of the Rob@work 3. Therefore, $T_{CAM}$is essentially considered at the origin.

$$R_{CAM} = \begin{pmatrix} \cos\theta_{CAM} & -\sin\theta_{CAM} & 0 \\ -\sin\theta_{CAM} & \cos\theta_{CAM} & 0 \\ r_{31} & r_{32} & 1 \end{pmatrix};\ T_{CAM} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{11}$$

Therefore, the two-dimensional position and the one-dimensional orientation of the camera are calculated in the fixed marker frame since the mobile robot moves in a plane. According to Figure 7 and Figure 8, the two-dimensional position of the camera in the fixed coordinate system is calculated below (LaValle 2006, pp. 94-97):

$$\begin{cases} x_{mar} = |x_{CAM}|\cos\theta_{CAM} - |y_{CAM}|\sin\theta_{CAM} \\ y_{mar} = -|x_{CAM}|\sin\theta_{CAM} + |y_{CAM}|\cos\theta_{CAM} \end{cases} \tag{12}$$

Several configurations corresponding to the different position and orientation of the robot with mounted sensors are collected in Table 1 to identify the expected behavior of the robot to design an accurate control system with appropriate control signals in different configurations.

Table 1. Expected control signals for different configurations of the robot.

Specification / Config.	y_{mar}	θ_{mar}	Expected $\dot{y}_{Rob}$	Expected $\dot{\theta}_{Rob}$
1	+	> 0	→	CW
2	+	≈ 0	→	0
3	+	< 0	→	CCW
4	-	> 0	←	CW
5	-	≈ 0	←	0
6	-	< 0	←	CCW

After the pose estimation, the camera coordinates are published with respect to robot with the ROS built-in package called transform, which keeps track of different frames over time to simply show the position of one frame with respect to others (Saito 2015). The marker coordinates are published in a node called *marker_pose*. Once the position of the camera is

published with respect to the fixed marker, it will be subscribed in a callback function to begin the motion control process to move the robot toward the docking platform.

Figure 8 represents six various initial configurations in which the Rob@work 3 is placed in different positions and orientations inside the docking area. The control system is designed such that the Rob@work 3 should accomplish the docking with the vision feedback from the camera and the control signals along θ_{Rob} and y_{Rob} axes are computed according to the pose estimation with respect to the fixed marker.

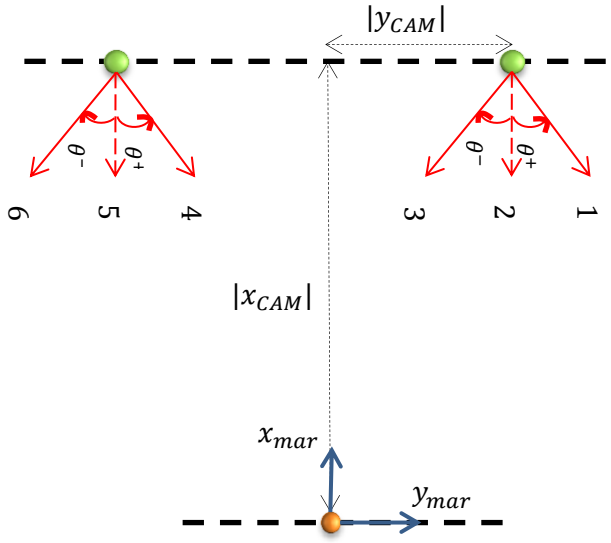


Figure 8. Six different initial robot configurations used to evaluate the controller.

2.3 Fiducial Markers

A fiducial marker is a standard design added to the environment as a target which appears in the image produced by the camera. Its application in robotics is quite useful where pose estimation between the vision sensor of the robot and the marker is required. It also provides an accurate pattern with a digital word that contains a unique ID protected from false detection (Fiala 2005, p. 596).

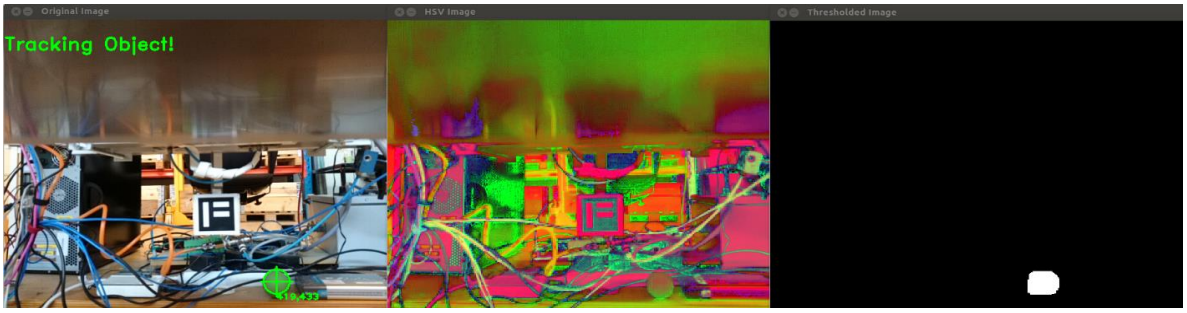
Fiducial marker systems have some advantages to be considered as the target of docking the Rob@work 3 in this project. First, the identification is practically easy since such a black and white square sketch with embedded internal pattern facilitates the marker detection. Second, it

speeds up the marker detection with accompanying computer vision detection algorithms. For detection, however, having a uniform lighting condition on the foreground and the background is essential since the environment should be visible enough to the vision sensor to obtain accurate pose estimation. Finally, it provides accurate and low-cost solutions to develop robust detection algorithms. (Lepetit & Fua 2005, pp. 1-6.)

Fiducials are divided into two general categories of point fiducials and planar fiducials, with which the vision-based docking of the Rob@work 3 is investigated. In the point fiducial, the camera coordinate system has the two-dimensional positions (x_{CAM}, y_{CAM}), whereas the planar fiducial is employed to indicate positions and orientation in the three dimensions $(x_{CAM}, y_{CAM}, \theta_{CAM})$.

2.3.1 Point Fiducial

The point fiducial is very common in computer vision for object recognition due to its simple shape and quick tracking algorithms (RoboRealm 2005). In particular, the point fiducial can have a distinctive geometric pattern made from reflective materials such as circular or spherical structures with centroid locations. Such circular shapes provide fast recognition and relative pose estimation only with their centers, known as point of correspondence, and facilitate the identification. (Lepetit et al. 2005, pp. 32-35.)



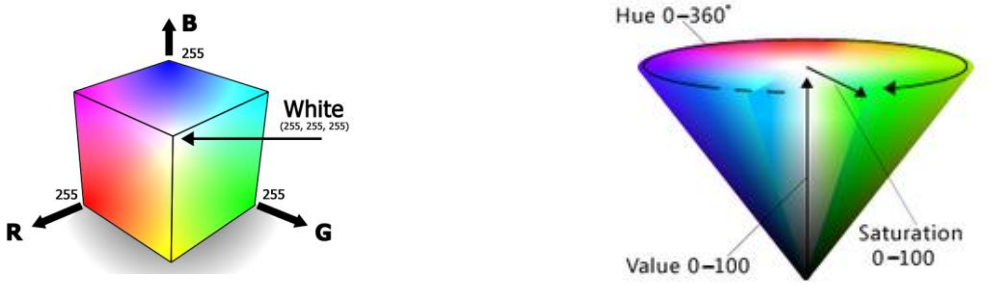
a) Original image. b) Colored image. c) Threshold image.

Figure 9. Point fiducial detection.

A video stream with the resolution of 640×480 pixels is an input matrix for the tracking algorithm to start the image processing to detect the point fiducial. Figure 9 a) depicts the

detected fiducial with OpenCV in the camera coordinate system in which a green ball attached to docking platform as the target. Figure 9 c) illustrates the threshold window and detected point fiducial after the morphological operation is performed to eliminate the noise (OpenCV 2011b).

Figure 9 b) shows the docking area in a hue, saturation, and value (HSV) color space with a green ball. The input matrix has a red, green and blue (RGB) color space which contains integer values ranging from 0 to 255. For color-based segmentation, the RGB format is transformed by the HSV cylindrical coordinate since it breaks down the color into simpler characteristics and separates the color components from intensity to add robustness to lighting changes or removing shadows. (Gonzalez & Woods 2008, pp. 289-295.) In the HSV color-space, hue represents color transform from red to green, saturation is the color transform from red to pink, and value is from black to white (Forsyth et al. 2012, pp. 68-69).



a) RGB (0-255). b) HSV (Value, Saturation 0-100%, Hue 0-360 degree).

Figure 10. Image color-spaces (Forsyth et al. 2012, p. 70).

Moreover, the conversion between HSV and RGB is practically handled by a track bar GUI developed to adjust different colors to detect point fiducial in the library (OpenCV 2011c). The track bar shown in Figure 11 adjusts HSV values to detect the green ball as a point fiducial in docking the Rob@work 3.

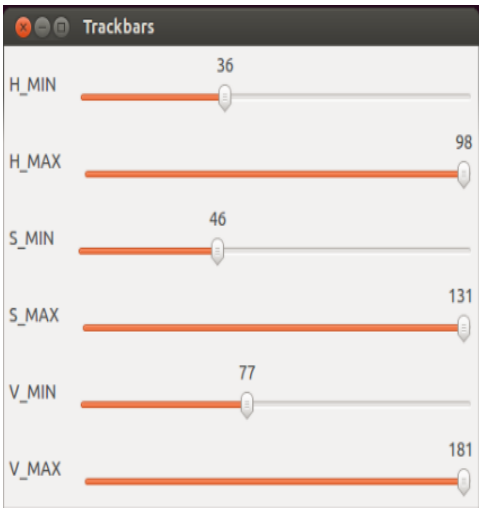


Figure 11. HSV track bar for color adjustment to detect green ball as the point fiducial.

The lighting condition in this approach, however, is quite challenging since the object used as a marker may not be reflective or symmetrical enough to be tracked properly or the environment does not have a uniform illumination with strong contrast. Therefore, the camera needs to be equipped with either a flash or an external symmetrical lighting source fixed in the right angle.

2.3.2 Planar Fiducial

The augmented reality field has increasingly widened planar fiducial application in computer vision since it consists of unique patterns compatible with the Aruco marker detection algorithm (OpenCV 2015). The square-bordered structure with black and white background color facilitates the recognition of the unique internal pattern with the definite shape and ID called an ARTag. In such systems, digital coding theory is employed to obtain decent patterns for further applications in the image processing. In the detection level, several parameters are required to achieve robust outlines in the target for different illuminations, such as low false positive factor and confusion rate to detect markers. The low false positive rate is a probability of detecting marker falsely, whereas the confusion rate is the probability of obtaining wrong marker ID. (Fiala 2005, pp. 590-591.) Table 2 represents a few standard ARTag compatible for Aruco marker detection algorithm in OpenCV, given specific IDs.

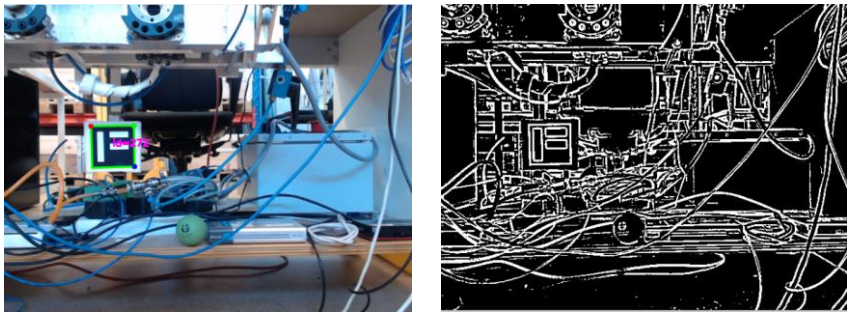
Table 2. Standralized Aruco markers with specific IDs [0 – 1023] (Welsh 2014).

ID # 1	ID # 48	ID # 61	ID # 137
			
ID # 272	ID # 349	ID # 459	ID # 514
			
ID # 606	ID # 746	ID # 899	ID # 1023
			

The fiducial design of ARTag with four corners is used as the fixed coordinate system mounted to the docking platform for pose estimation of the camera attached to the robot inside the docking area with high reliability under different lighting conditions and backgrounds. The different lighting conditions, however, do not effect on robust detection since ARTags benefit from edge linking and edge-based detection methods coupled with a digital coding scheme for identification. The edge-based method is utilized to detect quadrilateral shapes directly from an image. Although line segment detection demands higher computations, partially occluded markers can still be detected even if some corners and sides are missing. (Fiala 2005, pp. 593-594.)

The planar fiducial used in this project is chosen from the samples shown in Table 2. The chosen marker is similar to the English letter “F” (ID # 272). It is generated based on a unique ID that encods the marker image, the edge size of the back box containing the marker, and thickness of the white border surrounding the marker. The length of marker is 80 mm with 8 mm of paddle as a thickness of the white border. The genrated marker is mounted to the center of the docking platform and between end-effectors. Figure 12 shows the detected Aruco

marker in the OpenCV. Although the image background is occluded with several disturbances, the planar fiducial marker is fully trackable.



a) Original image. b) Threshold image.

Figure 12. Detected Aruco marker in the docking platform.

The pose estimation of the camera with respect to the fixed marker frame is calculated with (12). At this stage, the camera is already calibrated and its parameters are already derived. The detection process of fiducial and pose estimation approximately runs at 30 frames per second (fps) and can be applied in every frame. The visualization time of the USB and the IP camera is compared in the result chapter and control signals are computed accordingly.

2.4 Autonomous Vision-based Docking

In this section, the effect of the vision sensor on the precise docking is investigated to design an appropriate control system for the mobile robot.

2.4.1 Kinematic Model

For robot platforms such as the Rob@work 3, formulating an exact mathematical model to start the control process is sophisticated and mostly likely undesirable. Therefore, a simplified model with only essential information is preferred. The dynamic model of the robot is apparently crucial to extract the maneuvers that cause some complexities in the whole system. However, in this thesis only the kinematic model of the system is deployed to make a simpler approach since less maneuvers in docking of the mobile robots exist. Some assumptions are taken into account to simplify the system under investigation (Nilsson 2010, p. 18):

- Non-deformable wheels with solid rubber structure

- Single contact point between wheels and the ground
- Same amount of loads for all wheels
- Robot moves on a flat plane.

Although the movement of the robot toward the target is carried out in the indoor environment coordinates, the control algorithm is designed based on the pose estimation of the camera with respect to the fixed marker to compute the robot velocity commands to the under-carriage control. The Rob@work 3 has the coordinate system which is originated in its center of gravity and control signals are computed and applied to the actuators. In this coordinate system, positive x-axis points to the forward direction, whereas positive y-axis points to the left direction if it is seen from the top.

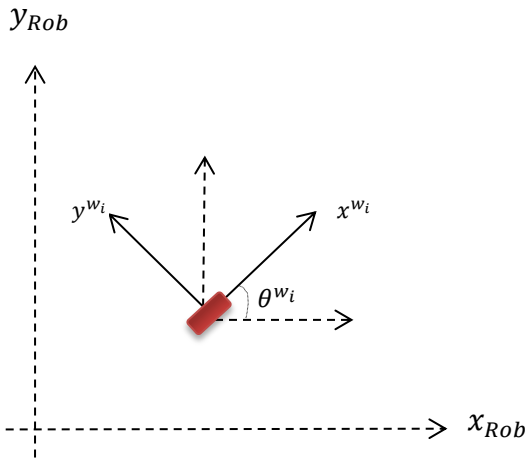


Figure 13. Top view of the rotated wheel *i* by θ^{w_i} angle in the robot coordinate system.

Each wheel also has the local coordinate system which is parallel to the robot coordinate system. Figure 13 illustrates a local wheel coordinate frame (x^{w_i}, y^{w_i}) rotated by θ^{w_i} which is the orientation of each wheel. θ^{w_i} is geometrically the angle between the x_{Rob}-axis and x^{w_i}-axis. The rotation of the wheels is computed with respect to the CG in which the entire platform rotates for rotation angle of robot wheel (θ^{w}) degrees if the wheels rotate for θ^{w} degrees.

2.4.2 Control Design

A mobile robot equipped with the vision sensors has the potential to obtain an accurate manipulation and achieve the goal because of a noncontact evaluation of the environment (Lepetit et al. 2005, p. 4). The employed control approach theoretically computes precise control signals if the input measurements have enough accuracy in the system.

The first control design approach is based on the open loop control system for visual manipulation. In this approach, the robot does exactly what the vision sensor perceives in the environment. It cannot guarantee the desired outcome with high accuracy unless very advanced vision sensors and tool changers are employed to achieve the certain task. This approach is, therefore, economically impractical to obtain the required motion commands and achieve the docking goal. (Hutchinson et al. 1996, p. 651.)

In contrast, the second control approach seems more practical since visual information on the feedback loop gets involved to increase the overall accuracy of the control system. In this approach, two images are virtually involved, one with the target which has an estimate position of the docking platform, whereas the second one acquires at the position of camera and robot in real-time.

In fact, this second approach constantly evaluates the current position of the vision sensor with respect to the target to compute appropriate control signals. The closed loop vision-based control, also known as visual servoing carried out by the visual information in separate images which practically involves the vision sensor and the computer vision algorithms to detect the employed fiducial markers on the docking platform and control the position of platform of Rob@work 3 (Hutchinson et al. 1996, pp. 653-654).

In this project, fiducial markers are employed to identify the docking platform. The visual system provides position and orientation references for the control system. Therefore, position-based visual servoing (PBVS) serves basically as the feedback of the control system to extract the features of the image and compute the difference between the current positions of the mounted camera at the Rob@work 3 with the fiducial markers as the target. The PBVS

approximately estimates the position and orientation of the camera coordinates with respect to the fixed marker as the target and sends appropriate the control signals to the robot (Wilson, Hulls & Bell 1996, p. 684).

The target plays a crucial role to design the vision-based control system since the current position of the robot is evaluated online to provide appropriate control signals to eventually eliminate the position and orientation errors. Therefore, the more accurate the reference point leads to more appropriate the control signals. Therefore, the target is graphically sketched as a circle with an origin of the marker reference position (P_{mar}^{ref}) and $x_{thresh} = 1$ mm which is a radius of threshold along marker frame.

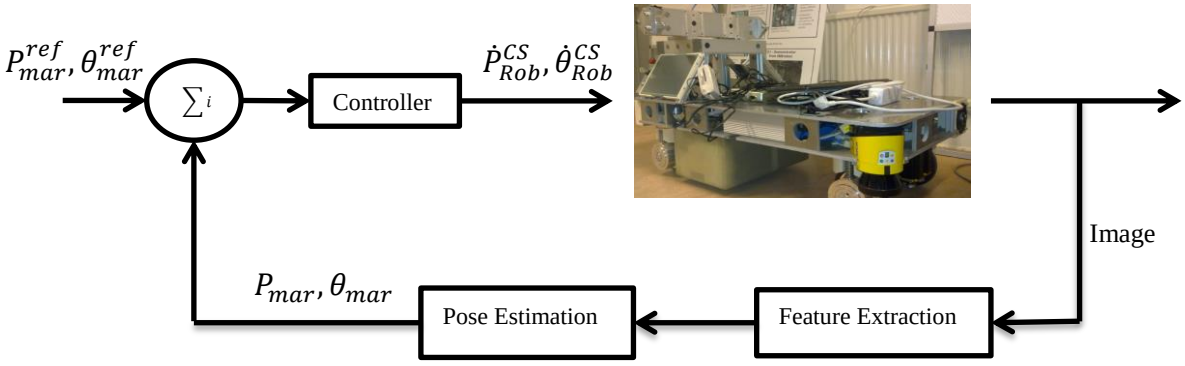


Figure 14. PBVS for docking the Rob@work 3.

The most important part of the control system, however, is the plant. Theoretically, the plant is the combination of the process and actuator and expressed as a transfer function to declare the relation between the input and output signal (Ogata 2010, pp. 1-3). In this thesis, the Rob@work 3 is the plant which contains the mobile platform as a process and four omnidirectional wheels as the actuators. The sampling period of the embedded control system on the Rob@work 3 is approximately 20 milliseconds to send the appropriate steering and driving speed commands to the actuators and move the platform (Nilsson 2010, p. 15).

Figure 14 represents the vision-based closed loop control design employed to computer the control signals for the Rob@work 3. P_{mar}^{ref} and the marker reference orientation (θ_{mar}^{ref}) are set as the reference signals. The reference values are determined when the Rob@work 3 is

manually docked. The reference is constantly compared with estimated positions (P_{mar}) and estimated orientation (θ_{mar}) of the robot after the feature extraction of the markers to calculate the error as the input to the controller to obtain the robot commands.

In practice, the robot commands are applied velocities for the streering and driving motors of the actuators to move the mobile robot platform. Figure 15 depicts the under-carriage control system of the mobile robot. An input rob_{cmd} is the designed feedback control system for the Rob@work 3 after the pose estimation of the camera coordinates in the marker frame attached to the docking platform as calculated in (14). It is not among the goal of this thesis to study the kinematics of the Rob@work 3 platform in detail. Therefore, the plant of the control system is simplified to a first-order integrator to simplify the design in the vision based docking method.

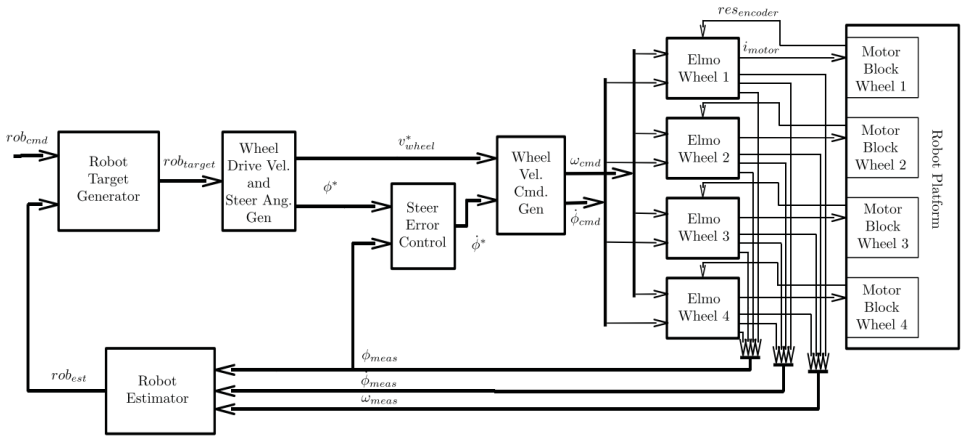


Figure 15. The under-carriage control of the Rob@work 3 (Nilsson 2010, p. 15).

In this project, the docking area is divided into three different zones in order to design a more precise controller for the aside position (y_{Rob}) and the orientation (θ_{Rob}) as stated below:

- Approach zone
- Safety Margin (SM) zone
- Target zone.

It is assumed that the Rob@work 3 always starts the docking process somewhere in the approach zone which is far from the docking platform along the x_{mar}-axis (approximately 18

cm). Higher velocities are also allowed in the approach zone. The SM, on the contrary, is closer (maximum 18 cm away from the docking platform) and only considers to finally adjust the orientation along the θ_{mar}-axis, the aside position along the y_{mar}-axis, and the smoothness to accomplish higher accuracy before docking is completed.

In this project, various configurations were investigated to compute the approprite steering and driving speed commands for the actuators. According to Figure 16, adjusting a movement along the y_{Rob}-axis and orientation along the θ_{Rob}-axis is more crucial than the movement along the x_{Rob}-axis, since robot can move toward the docking platform even if a constant driving velocity ($Vel_x^{App.}$) is applied. Therefore, the control signal along the x_{Rob}-axis is the constant velocity when the robot moves toward the docking platform even though values could differ from zone to zone.

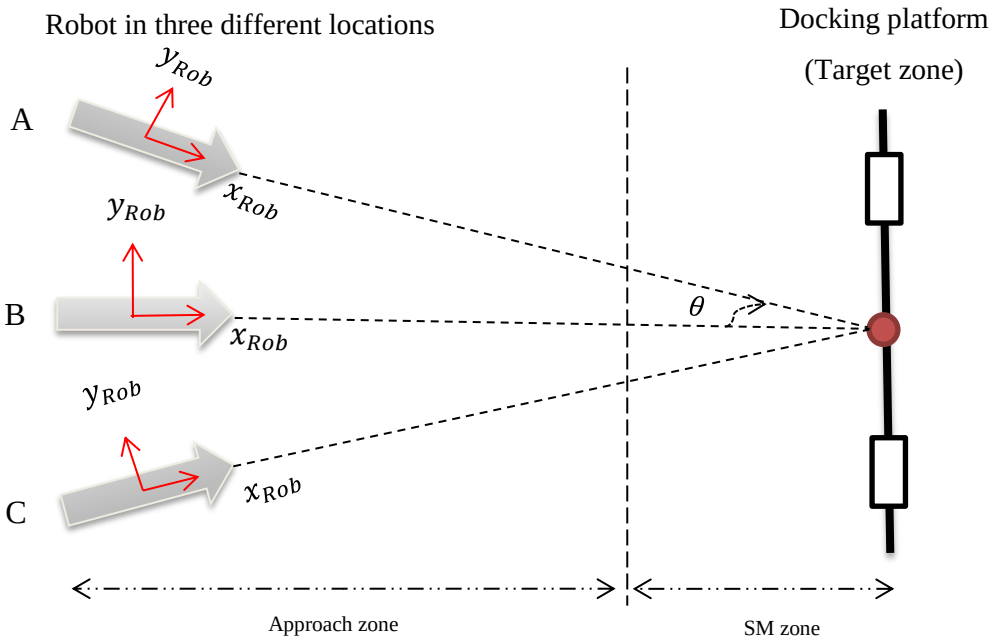


Figure 16. Docking area with different zones.

The control system is designed such that the Rob@work 3 always moves toward the docking platform to adjust X_{mar}. Furthermore, the movement with finishing velocity (Vel_x^{SM}) along the x_{Rob}-axis is prohibited as soon as the Rob@work 3 enters the SM zone, which

means $Vel_x^{SM} = 0$. This strategy is taken into account to make sure the position and the orientation of the camera are precisely adjusted compared to the fixed marker frame. Therefore, the docking threshold for position and docking threshold for orientation ($y_{thresh} = 2.5$ mm and $\theta_{thresh} = 2°$) are employed to terminate the controllers and switch to forward movement along the x_{Rob}-axis again to complete the docking task.

Time is critically crucial in docking and a certain time is defined as the desired docking time of mobile robots. The vision-based feedback control is designed such that the target is accomplished in shortest average docking time. ROS is a built-in framework running on the Ubuntu operating system to provide the deterministic scheduler for the given tasks which is not executed in real-time. In fact, the ROS framework has its own time configuration to compute the control signal. The time is initially recorded on the embedded controller of the Rob@work 3 via the ROS client library package during the data acquisition process (Gomes 2013).

In this thesis, the ROS time is converted to the real time to give a better sense of docking time. The conversion is done with an initial time (t_i^{ROS}), a final time (t_f^{ROS}), and recorded sampling time (t_s^{ROS}).

$$\Delta t = t_f^{ROS} - t_i^{ROS} \tag{13}$$

Each step is mathematically derived as the time difference of two consecutive samples (Δt). The vector of real time (t_{real}), on the contrary, is calculated as follows and utilized for further computations and control design:

$$t_{real} = 0 : \frac{\Delta t}{t_s^{ROS}} : \Delta t \tag{14}$$

The error signal ($e(t)$) is the difference between the reference position of camera (P_{ref}) and the current position of camera (P_{cur}) in the fixed marker frame calculated below:

$$e(t) = P_{ref} - P_{cur} \tag{15}$$

and robot control signal (rob_{cmd}) is defined as (Ogata 2010, p. 84):

$$rob_{cmd} = u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{16}$$

in which K_p , K_i , and K_d are proportional, integral and derivative gains, respectively. Pseudo code of the control design is provided in Algorithm 1, which is implemented in the program for the position along the y_{Rob}-axis and the orientation along the θ_{Rob}-axis. The movement along the x_{Rob}-axis, on the contrary, is given constant values depending on the zone the robot is placed at the moment.

```
while(true){
    error_previous = error_current = error_integrated = error_derivative = 0;
    error_current = set point − current pose;
    error_integrated = error_integrated + (error_current × dt);
    error_derivative = (error_current − error_previous) / dt;
    control signal = K_p * error_current + K_i * error_integrated + K_d * error_derrivative;
    error_previous = error_current
    sleep(dt)
}
```

Algorithm 1. Linear feedback control pseudo code.

In Algorithm 1, the parameter dt is the iteration time of the designed controller which is set to one millisecond to make sure controller is running in parallel with other nodes in ROS to avoid delays in the whole system. Another fact about the designed controller is that it is

implemented inside the subscriber to the published nodes from the published position from the fiducial marker.

2.5 Reinforcement learning

In this section, a model-free Reinforcement Learning (RL) approach is investigated to obtain the optimal docking trajectory within the less docking time.

2.5.1 Background

Interacting with the environment is most likely the first and simplest step in the learning process. In the RL, despite supervised learning, there is no physical teacher who teaches how to obtain the desired behavior and learner is not told what to do or which action to take; instead it is the responsibility of the learner to explore the environment with several trials while taking different actions to eventually make the correct decision to accomplish the desired performance (Duda, Hart & Stork 2000, pp. 16-17). This interaction is called goal-directed learning method for a larger problem in an unknown environment and is among a focus of the RL approach (Sutton & Barto 1998, pp. 4-5).

In robotics, learning a certain task is the process of trying different actions based on the current state which ultimately puts the agent in the target state while obtaining the highest award. The reward, which is similar to feedback in the control theory, is scalar and eventually leads the robot toward the desired behavior. Appropriate actions according to the current state of the robot in the docking area are inspirational meaning that helpful behavior that should be persistent, whereas wrong actions yield more negative rewards to prevent the inappropriate behavior of the agent (Gaskett 2002, pp. 2-3).

There are few concepts in the RL framework. For instance, an intelligent agent or shortly agent consists of the sensors and the actuators to act on the environment. It either learns or uses a provided knowledge to achieve its goal. A state physically represents the location of the agent inside the learning environment. An action is simply a decision made by the agent according to the current state to move the robot to a new one. (Sutton et al. 1998, pp. 43-45.)

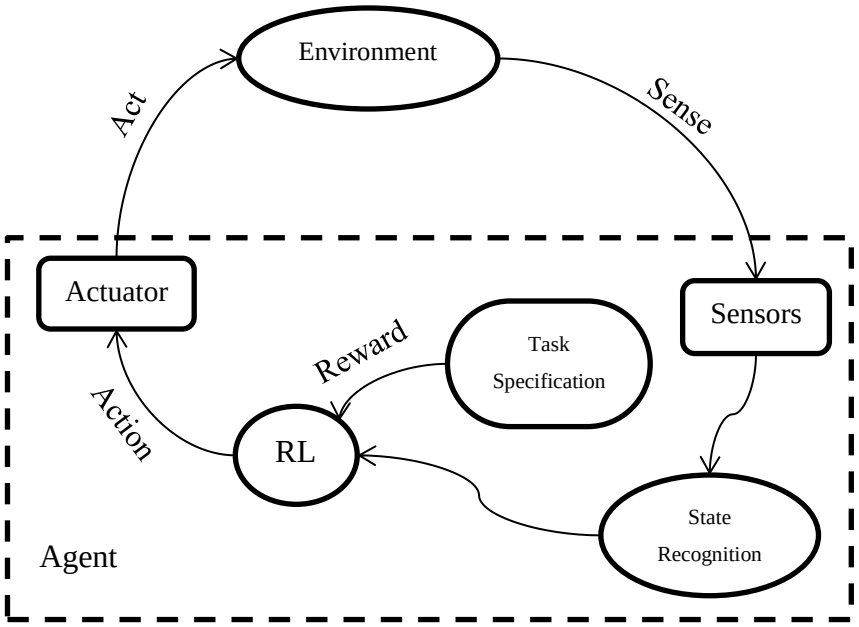


Figure 17. RL setup (Sutton et al. 1998, p. 44).

According to Figure 17, the robot acts on the environment with set of actions to eventually accomplish a given goal and practically manipulate the robot toward the target according to the current state. The position and orientation will be changed while the robot moves from state to state. The actions are commanded velocities applied to the mobile robot platform.

In this thesis, the temporal difference (TD) method, as one of the simplest RL approaches, is employed for training since it does not require a model of the environment to update the estimates based on previously learned ones regardless of the final outcome. The TD learning method has two different classes, including on-policy and off-policy control approaches (Sutton et al. 1998, pp. 129-131).

The off-policy control approach, known as Q-learning, uses the prior knowledge to expedite the RL despite the value of the optimal policy to obtain the largest value for the action (Jun et al. 2006, p. 2656). On the contrary, the on-line policy learns to takes the slower path with the random policy since it wants to explore both optimal and non-optimal actions. Therefore, it is more careful about the environment. The target in the Q-learning method is the state identified

as the docked robot and the actions are influences of the robot on the environment to obtain optimal policy of the system. (Poole & Mackworth 2010, p. 475.)

2.5.2 Problem

Finding the desired policy for docking of the mobile robot with respect to the trajectory and the average docking time was already investigated with the vision feedback control in the Section 2.4. The model-free RL approach is investigated in this thesis, as an alternative, to compare the results with the vision-feedback control system to select the dominant approach. The RL can lead to an optimal approach with respect to rewards given to the agent.

2.5.3 Solution

The learning system is made based on the off-policy method in which the Q-learning employs the prior knowledge of the agent in the docking area to obtain the optimal trajectory toward the target and dock the robot with the sufficient precision. The idea is to make a discrete grid presenting the matrix, known as Q-matrix, to update its components while transferring from state to state. In this design, the Rob@work 3 platform is considered as the agent with the vision and laser sensors. The pose estimation is done similar to the Section 2.2.2 to obtain the camera coordinates with respect to the fixed marker attached to the docking platform.

The Q-matrix acts as the brain and represents the memory of several experiences. Similar to the control design (Section 2.4.2), controlling aside position along the y_{Rob}-axis and orientation along the θ_{Rob}-axis are prioritized for the docking. Therefore, the Q-matrix mathematically contains orientation and position of the marker (θ_{mar}, y_{mar}) which are set as row and column of the matrix, respectively while movement along the x_{Rob}-axis is not considered among actions and constant velocity is applied this axis during the entire process.

The successful docking is equivalent to the desired behavior, which is evaluated with position and orientation accuracy to attach the end-effectors, smooth docking in which mechanical setups do not collide or hit obstacles and detectable marker throughout the whole process. On the contrary, the behavior is undesirable if it falls into one of the following situations:

- Marker is lost
- Mechanical setup collision
- Obstacles detected by the laser scanners
- Leaving the grid boundary.

The marker has a key role in the vision-based docking of the Rob@work 3 since the localization and the navigation are carried out with a vision sensor. It can be lost if sudden movements or rotations get involved or an object distracts the camera to detect the maker. As a consequence, the robot not only loses its location inside the docking area but navigation would also fail.

The robot is judged to be unsuccessfully docked if the end-effectors on the robot and the mechanical joints on the docking platform are detached. This behavior is dangerous and highly sensitive for the docking platform and the mobile robot. For safety reasons, a 2-cm margin along the x_{mar}-axis is defined to constantly evaluate the correct position of the robot before the final move toward the docking platform in the training phase.

Collision with obstacles is strictly prohibited and stops the robot for further translations or rotations. The laser scanners are responsible to detect obstacles on either front or rear sides of the mobile robot. It is very likely that the mobile robot occasionally collides with obstacles since training takes place with random actions. Thus, as soon as one of the scanners detects the obstacle within its territory, learning terminates, and the robot gets punishment for that specific state and it moves to a new random position to start over.

The developed grid has limited boundaries for both states of θ_{mar} and y_{mar}, which are defined not only to protect the robot from mechanical collisions but also to make sure that the marker is always visible in the image coordinate system. If the robot ends up at a place outside the defined grid, the robot gets punishment for the previous state the robot was in before leaving the grid, moves to a new random position to start over while learning fails and the episode counter is reset.

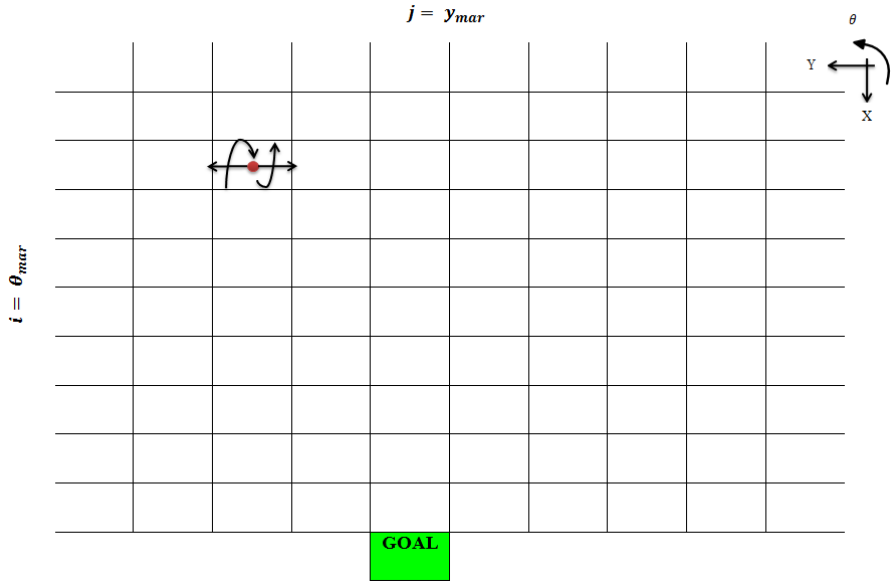


Figure 18. Developed grid for Q-learning action selection policy.

Figure 18 depicts the discrete grid developed for the Q-learning in the simulation environment. The docking area is hypothetically discretized with states. The goal state indicates the target in which θ_{mar}^{ref} and y_{mar}^{ref} are saved once the robot was manually docked before the training. Red dot represents the Rob@work 3 placed at its current state.

The initial state is randomly chosen inside the grid to diversify the states which are passed by the robot. Actions are linear velocity as control signal ($\dot{y}_{Rob}^{CS}$) along the y_{Rob}-axis and angular velocity as control signal ($\dot{\theta}_{Rob}^{CS}$) along the θ_{Rob}-axis. In another word, the robot moves left and right if the linear velocity is applied, whereas it rotates CW or CCW if the angular velocity is applied.

To begin the learning, the size of the reward and the Q-matrices are defined according to the goal states and boundaries. The Q-matrix is also initialized which practically means that the robot begins the learning while it does not know anything about the environment. It moves to different states and reaches the goal. Q-learning converges to an optimal policy even if the actions are totally exploratory. (Russell & Norvig 2010, pp. 844-845.)

Each exploration is called an episode in which the robot starts from a state and ends up in another state. The episode is finished if the robot succeeds to reach the goal state or fails under any of the circumstances mentioned above. In the case of failure, the robot deserves the negative reward since it has evaluated an undesired behavior which is not helpful to accomplish the docking task. If the episode is finished, the robot will be sent instantly back to a new random position inside the docking area to start the learning process over again. The transition rule of Q-learning is represented as (Sutton et al.1998, pp. 144-16):

$$Q(s_t, a_t) = R(s_t, a_t) + \gamma * \max[Q(s+1, \sum a_i)] \quad (17)$$

in which an specific component of the Q-matrix is updated and the reward for specific state and action ($R(s_t, a_t)$) the maximum value of the next state considering all possible actions are declared. The mathematical representation of (17) as a new sample estimate is expressed as (Sutton et al.1998, pp. 144-146):

$$samaple = R(s_t, a_t) + \gamma * \max[Q(s_{t+1}, a)] \quad (18)$$

in which γ refers to discount factor and $0 \leq \gamma < 1$. In this equation, $\gamma \cong 0$ is the immediate reward, whereas $\gamma \cong 1$ is the long-term high reward. The new estimate of the Q-value for specific state and action ($Q(s_t, a_t)$) in the Q-matrix is computed with an average function as (Sutton et al.1998, pp. 144-146):

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha(sample) \quad (19)$$

in which α refers to learning rate and $0 \leq \alpha < 1$. In this equation, $\alpha \cong 0$ corresponds to no learning, whereas $\alpha \cong 1$ indicates that prior knowledge is neglected.

Generally, the learning is started with $\alpha \cong 1$, to change Q-values faster and is lowered as the time progresses (Even-Dar & Mansour 2003, pp. 2-3). An ideal environment does not deal with stochastic problems and it converges to an optimal Q-function with $\alpha = 1$, whereas in

practice, the convergence requires more prior information and learning factor closer to zero. In this thesis, a constant α is chosen for the learning process ($\alpha = 0.1$). The Q-learning pseudo-code algorithm has is represented as follows:

```
Set parameters (γ, α) and environment reward (R).
Initialize Q-matrix.
For each episode:
        Begin from a random initial state
        While (the goal state has not reached)
                {
                        Choose an action randomly.
                        Move to new state with a chosen action.
                        Consider a new sample estimate with18.
                        Update Q-values with old Q-values with 19.
                        Next state = Current state.
                }
        End While.
End for.
```

Algorithm 2. Pseudo-code algorithm for model-free Q-learning (Sutton et al.1998, p. 145).

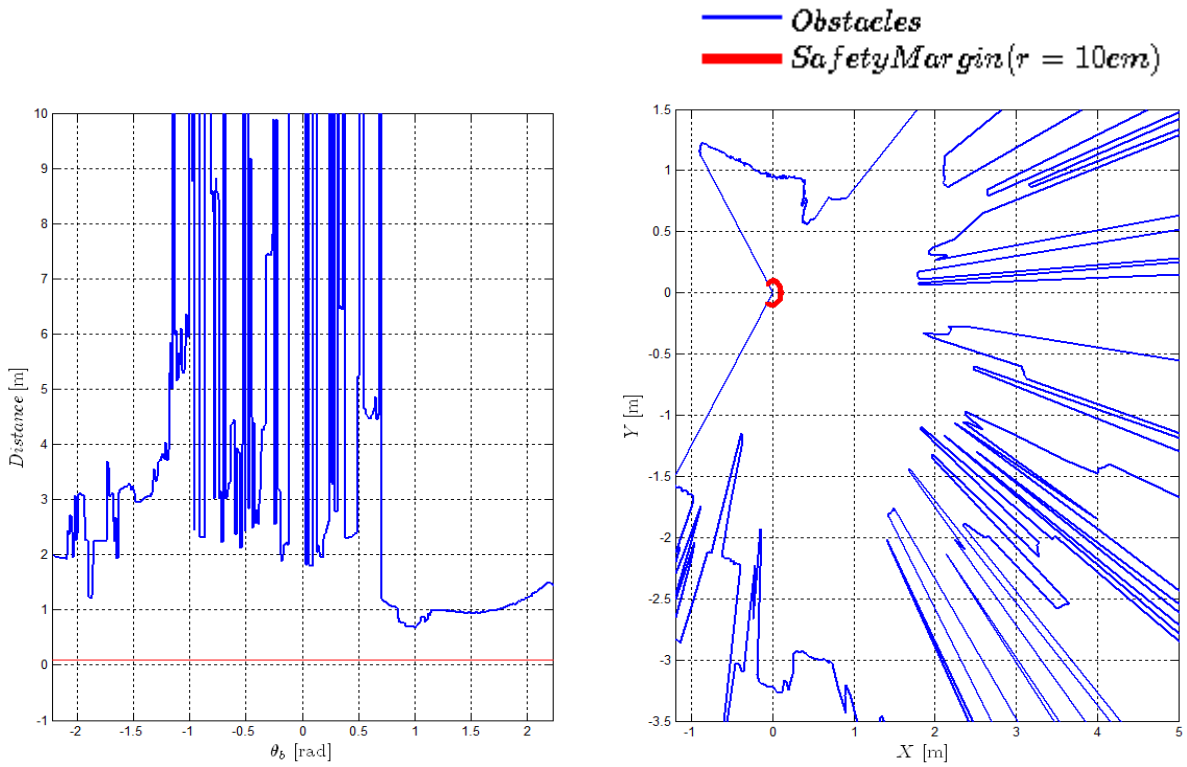
3 ANALYSIS OF RESULTS

In this chapter, the obtained experimental results from the autonomous docking of the Rob@work 3 with laser scanner and vision sensor are and evaluated. The obtained trajectory with the model-free Q-learning method is also depicted at the end of this chapter.

3.1 Autonomous laser scanner-based docking

In this section, several experiments for docking the Rob@work 3 using the laser scanners are presented to evaluate the precision of the laser scanner to dock the Rob@work 3. For each experiment, a plot of distance to the surrounding objects with respect to the beam angle (θ_b) and another plot of the top view of the front side laser scanner in the obstacle detection are presented.

An initial experiment is simply to check the functionality of the laser scanner inside the docking area, while no marker is attached to the docking platform. The reason to conduct such an experiment is to evaluate the accuracy of the laser scanner to determine the minimum and maximum ranges of the position with the datasheet of the manufacturer which indicates that the S300 Laser scanner detects surrounded objects up to a maximum radial distance of 30 m (S300 Standard 2016). However, figures are zoomed in to provide better visibility.



a) Different angles. b) Top view.

Figure 19. Laser sensor capability to detect objects in the environment.

Figure 19 a) illustrates the approximate distance to the surrounding objects recorded on the front laser scanner in different θ_b within the range of $[-2.35, 2.35]$ radians, which is equivalent to $[-135°, 135°]$ degrees. Angle is calculated from i and θ_{inc} of the laser beams as derived in (2) and (3). The red color corresponds to the safety margin distance defined for the laser scanner to detect the obstacles. The safety line is equivalent to the curve of the laser scanner which guarantees the safe movements without obstacle collision in any direction when it is seen from the top view, as depicted in Figure 19 b).

According to Figure 19 a), any object with distance less or equal than 10 cm to the laser sensor is considered as an obstacle and terminates the program. It can also be seen that areas closer to minimum and maximum i and corresponding angles are more entitled to obstacles rather than middle ones since the sensors are attached on the cross sides of the robot.

In the initial case where the reference marker does not exist and the robot is placed in the docking area, no intersection with the safety margin is recorded, the fact which also is observed in Figure 19. Therefore, the docking area is found clean from any physical obstacle within the distance of 10 cm while no marker is attached to the target.

The second experiment involves the Rob@work 3 when it is manually docked. Two different scenarios are compared and illustrated in Figure 20. The first scenario, Figure 20 a) investigates a detection accuracy of laser scanner when no marker exists, whereas second scenario, Figure 20 b), a transparent cylindrical bottle placed on the docking platform used as a detectable marker.

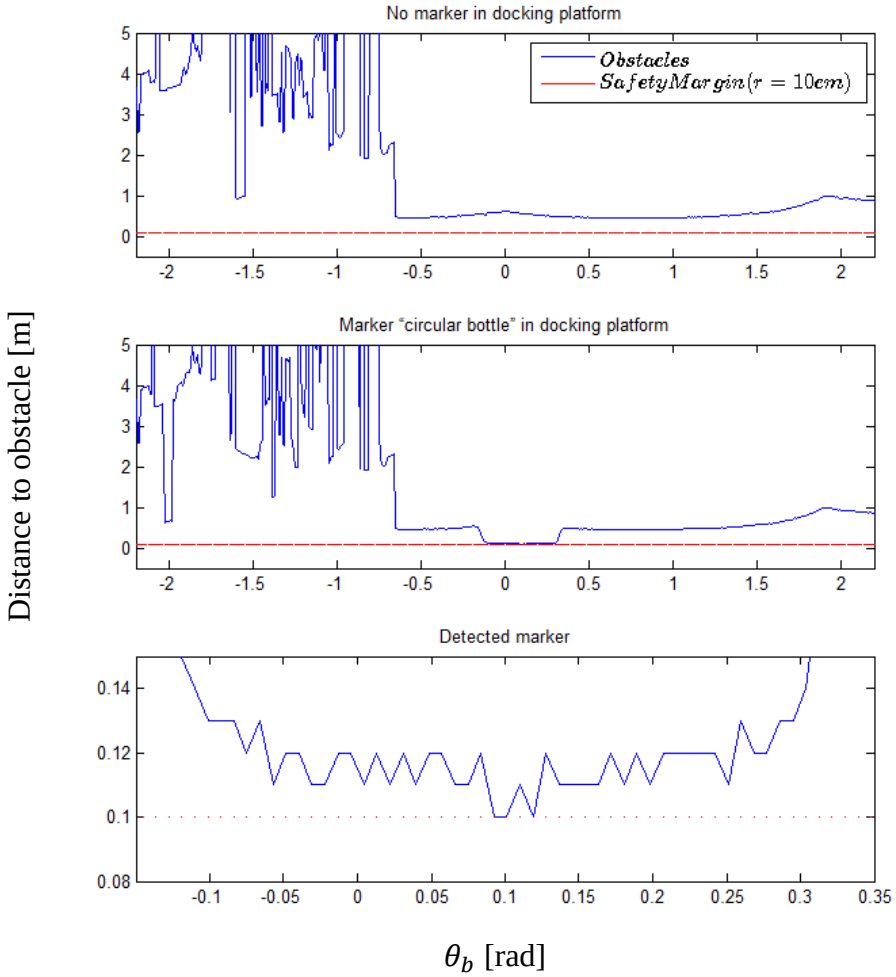


Figure 20. Laser scanner analysis for detected marker while the Rob@work 3 is docked.

As mentioned earlier, the safety margin is defined as a distance to the obstacle which is less than or equal to 10 cm. This experiment conducted with the docked robot to identify the marker. All the laser beams measure the distance to surrounding objects in order to identify the marker on the docking platform.

Since the docking area is clean and the mobile robot is not surrounded by any physical objects, the target, noticed as the marker in the docking platform, can only have the minimum distance to the sensor as depicted in Figure 20.

Theoretically, the laser scanner records data with 1-3 cm of accuracy (S300 Standard 2016). However, there are a few samples with the same distance to the assigned marker in different angles that the laser beam detected. Those samples have crossed the safety line of 10 cm. Therefore, different beams lead to the safety distance which makes the robot confused about the real marker to terminate the docking process precisely.

Furthermore, Figure 20 indicates some inconsistencies between the recorded data in the angular range of [-2.2, -0.5] radians which corresponds to [-135, -28] degrees, even though the environment is not changed throughout the experiment. The marker is only added to the docking platform. This is declared as another inconsistency in the docking of Rob@work 3 based on the laser scanner.

The laser scanner data of the same experiments are also plotted from the top view to illustraite the percievement of the two-dimensional position of the surrounding objects. Figure 21 depicts the safety curve of the laser scanner and the radial distance to surrounding objects. The safety curve guarantees the movement and rotation of the robot in any direction as long as it is not crosseed, otherwise, the experiment is terminated.

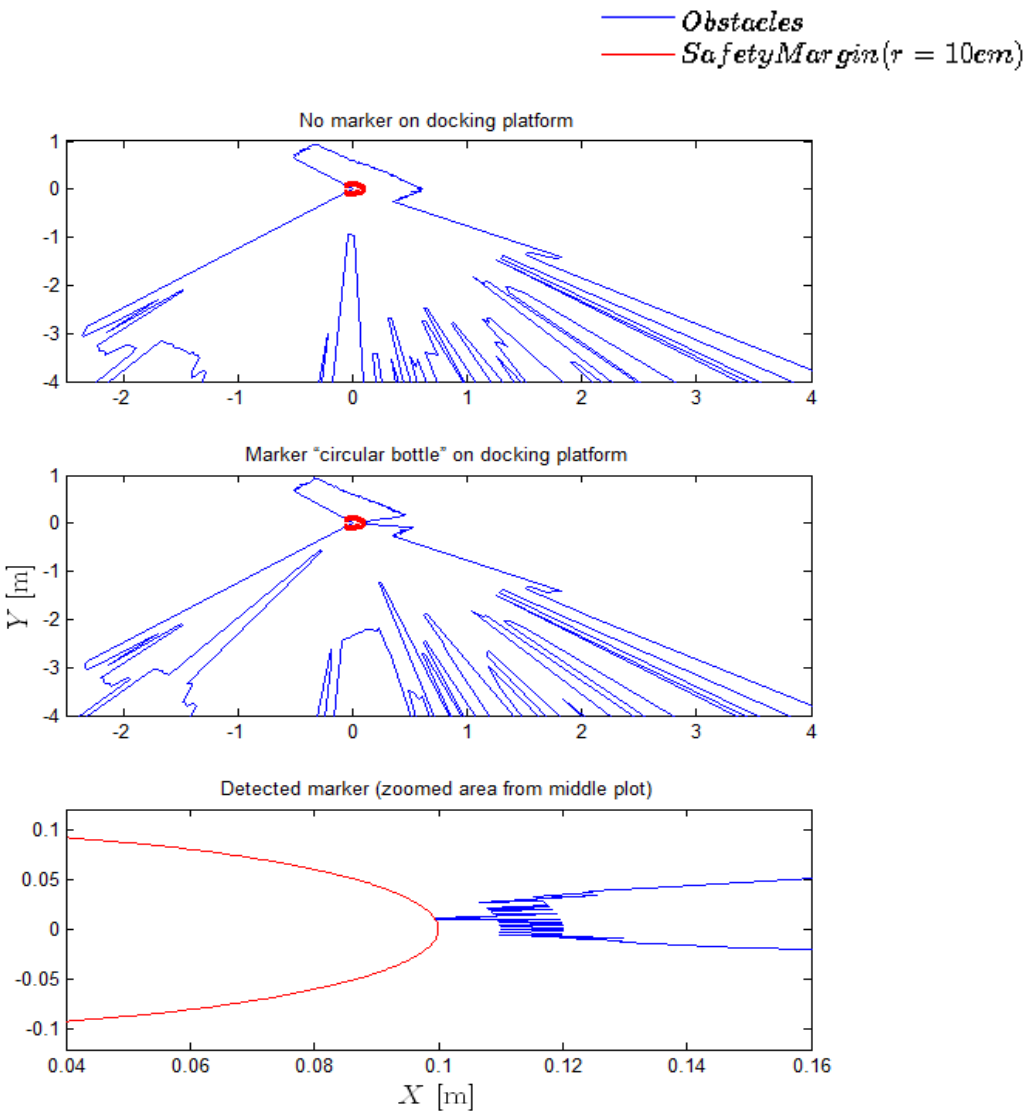


Figure 21. Laser scanner analysis for detected marker for the docked robot, top view.

The first trial was performed to make sure that there was no obstacle close to the area with the radial distance of 10 cm. Therefore, the marker was not placed in the target area and no obstacle was detected by the laser scanner within that distance. Therefore, the area closed to the laser sensor is clean and no intersection was recorded accordingly. The second trial employs the marker placed on the radial distance less than or equal to 10 cm. In this experiment the safety curve was crossed and the marker was detected by the sensor successfully. However, the detected marker has crossed the safety curve in different positions along the Y-axis when the robot was manually docked. It illustrates the inaccuracy of the

sensor in the marker detection since a few samples were detected with the same distance equal to 10 cm. Therefore, if the Rob@work 3 performs the docking only with the laser scanner, the inconsistencies of the marker detection do not let the mobile robot accomplish the docking task precisely.

In the third experiment, a new marker was utilized to evaluate the inaccuracy of the laser scanner in the marker detection. The new marker is a $30 \times 23.5 \times 21$ cm^3 box, located at the same place as the cylindrical marker. Figure 22 illustrates scanner analysis for this configuration.

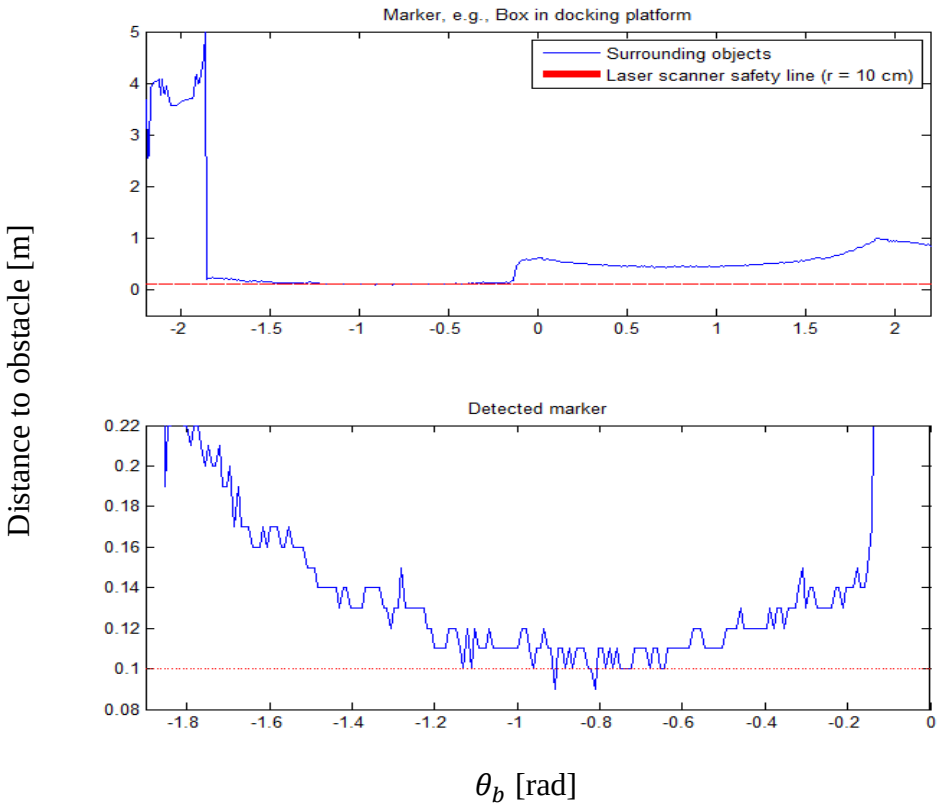
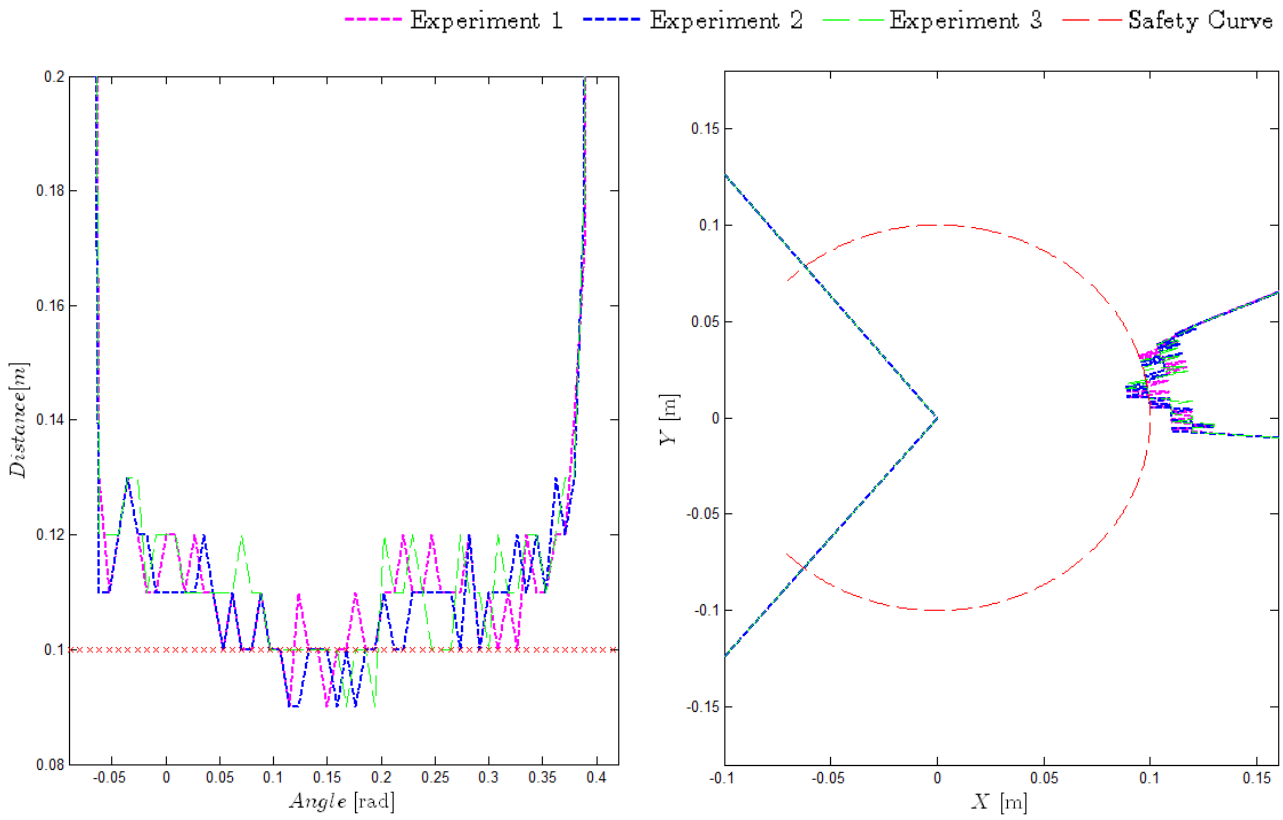


Figure 22. Scanner analysis with box in docking platform.

Since the length of the box is bigger than the radius of the cylinder, the detection area occupies more laser beams. The marker is detected with 1-2 cm accuracy even though Figure 22 illustrates plenty of intersections with the safety line. It practically implies that the marker was detected in several angles and positions. The laser scanner is more inaccurate in this

experiment with the box than with the circular bottle and the Rob@work 3 cannot achieve the docking task with such a marker.

In the last experiment, the laser scanner is required to detect the cylindrical marker considering the fact that neither the marker nor the robot were replaced or moved. Therefore, the sensor should detect the same distance to the marker in different experiments despite the accuracy of detection. The results of three experiments are illustrated in Figure 23.



a) Detected marker in different angles. b) Detected marker, top view.

Figure 23. Three conducted experiments for the same configuration.

According to Figure 23, there are a few samples which were detected as the obstacles in each experiment in the first place. Other experiments, however, have other samples which are detected as marker in different angles even though the position of the robot neither the marker

was changed during the process. Therefore, the laser scanners do not return reliable measurement for feedback each time from a stable marker placed on the docking platform.

3.2 Computer Vision

In this section, the results of the visualization and vision sensors are presented to analyze the obtained behavior of the control system.

3.2.1 Camera and marker frame

The pose estimation demands different coordinate frames to exist simultaneously and publishing the position and the orientation of one frame with respect to others in real time. In order to get the previledge of the vision sensor in the docking, an extra frame for the camera should be added to the Rob@work 3 frame. The ROS transform package broadcasts a frame in real-time with respect to its parent. After the coordinate insertion, $R_{3\times3}$ is employed to map the camera frame to the fixed marker frame.

The physical setup of the camera is such that it is mounted at the front side and approximately in the middle of the Rob@work 3 platfom. Therefore, the target should be detected approximately in the center of the marker frame. Figure 24 illustrates the different frames already created for different components of the Rob@work 3, in addition to the newly added frames of the camera and the marker.

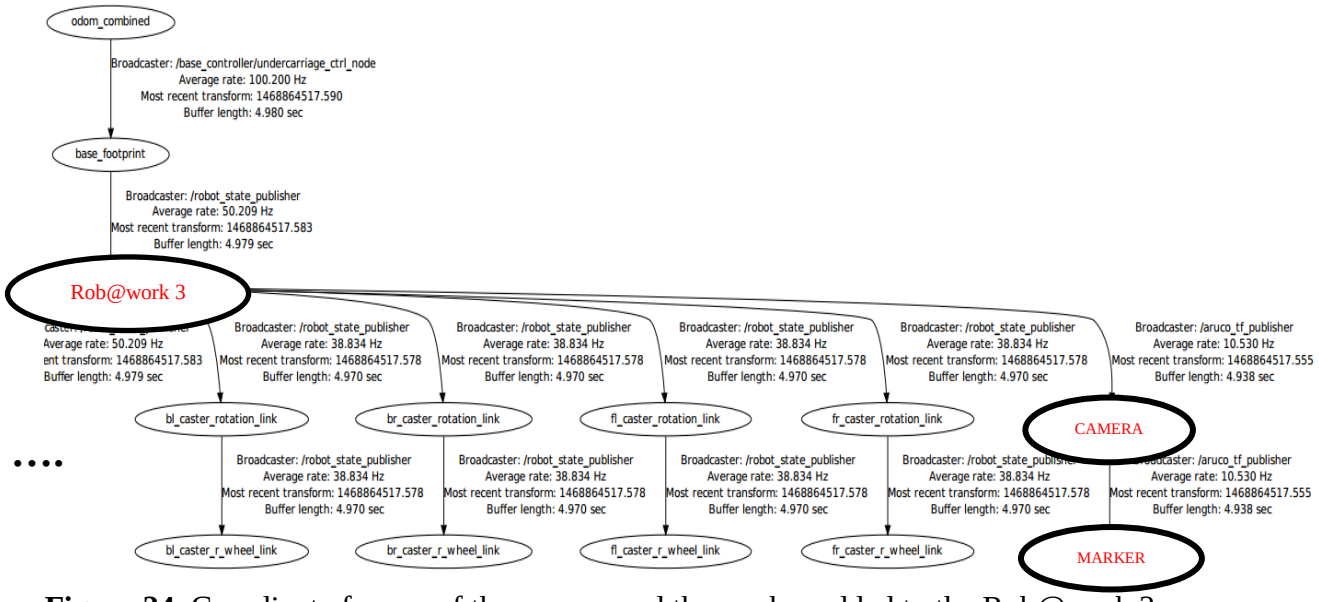


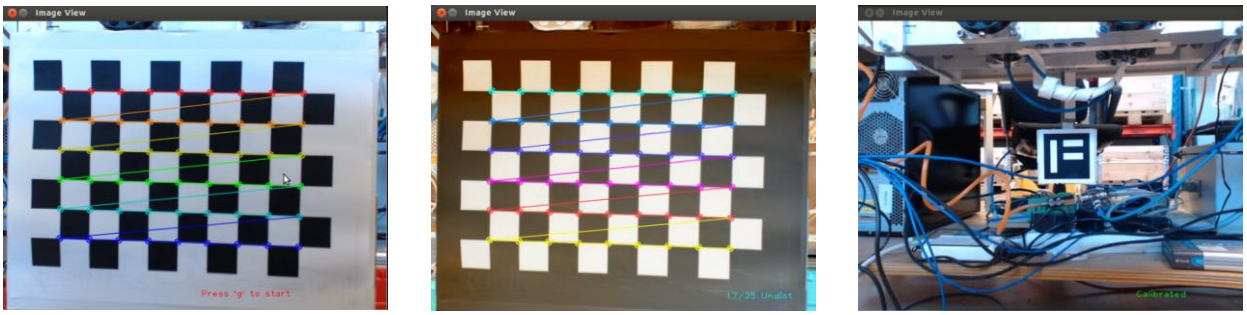
Figure 24. Coordinate frames of the camera and the marker added to the Rob@work 3.

According to Figure 24, the publisher for the added frames is the Aruco module based on the ArUco library in the OpenCV for the detection of the planar fiducial marker (OpenCV 2015).

3.2.2 Camera Calibration

In this project, the OpenCV algorithm is employed to calibrate the camera mounted on the Rob@work 3. The purpose of the calibration is to find the parameters of the intrinsic and the extrinsic matrices (Appendix A). The provided input is a classical black-white chessboard with 6×9 squares for the calibration. The algorithm basically captures 25 images with checkboard pattern to estimate correspondences between the two-dimensional points in the image and the three-dimensional points on the chessboard.

After the estimation, the radial lens distortion is corrected to calculate the intrinsic and the extrinsic parameters of the vision sensors. Figure 25 illustrates the calibration process of the camera with the chessboard pattern.



a) Detected squares. b) Calibration process. c) Calibrated image.

Figure 25. Camera calibration process.

The calibration program has a single argument which is a configuration file that consists of the characteristics for the chessboard pattern such as the height, the width and the square size. A summary of the given data is listed in Table 3.

Table 3. Configuration file for the camera calibration.

Input pattern type	Chessboard
Board Width [squares]	9
Board Height [squares]	6
Square size [mm]	25
Image source	Video Stream of IP camera or Android camera
Time delay between frames [ms]	500
fps	25
Tangential distortion	False
Principal point $C_0(u_0, v_0)$	Fixed at center → $C_0(0,0)$
Show undistorted image	True
Output file name	Camera_Model_Calibration_File.YAML

After the calibration process is done, a human friendly file which contains all the essential information of the calibrated camera is generated for the main image processing of the marker.

This file contains the image and board parameters such as the camera matrix, distortion coefficients, and the intrinsic and the extrinsic parameters for the further visualization process.

3.3 Vision-feedback Control Design

In this section, the results of vision-feedback control for docking of the mobile robot are presented. The vision sensor was employed to provide the feedback for the camera position with respect to the fixed marker on the docking platform. The idea of using the vision sensor is to increase the accuracy of the docking when the robot is placed in different initial configurations. It also is employed for the localization to find the relative position of the mobile robot with respect to the docking platform. Two different vision sensors were employed in this project. Figure 26 depicts the comparison between the recorded ROS time and the real time of the USB and IP cameras.

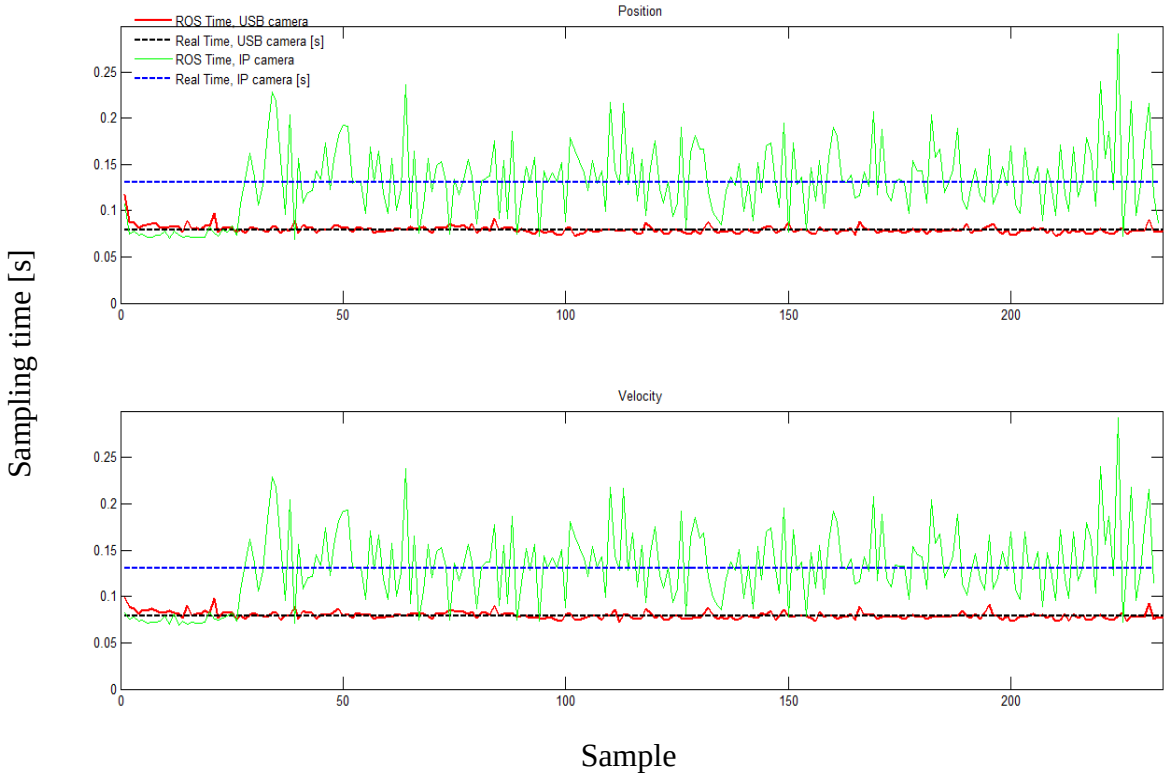


Figure 26. Sampling time comparison of different vision sensors.

The ROS time, depicted in Figure 26, is the extracted data of the recorded matrix from the pose estimation. Practically, the ROS time is the time when the Rob@work 3 nodes are brought up in the beginning of the process. The ROS time is started when the first message is received by the node and ends when the process terminates and the node receives no more messages.

After the real time computation, several experiments were conducted to compare different controllers to find the desired responses to accomplish the docking with high precision. As it was stated in Section 2.4.1, moving toward the docking platform along the x_{mar}-axis is implemented by applying constant velocities in the approach and the SM zones. Figure 27 depicts three experiments which graphically compare the effect of different constant velocities along the x_{Rob}-axis to achieve less average docking time and optimal trajectory. All three experiments are conducted using simple proportional controllers along the y_{Rob}-axis and the θ_{Rob}-axis.

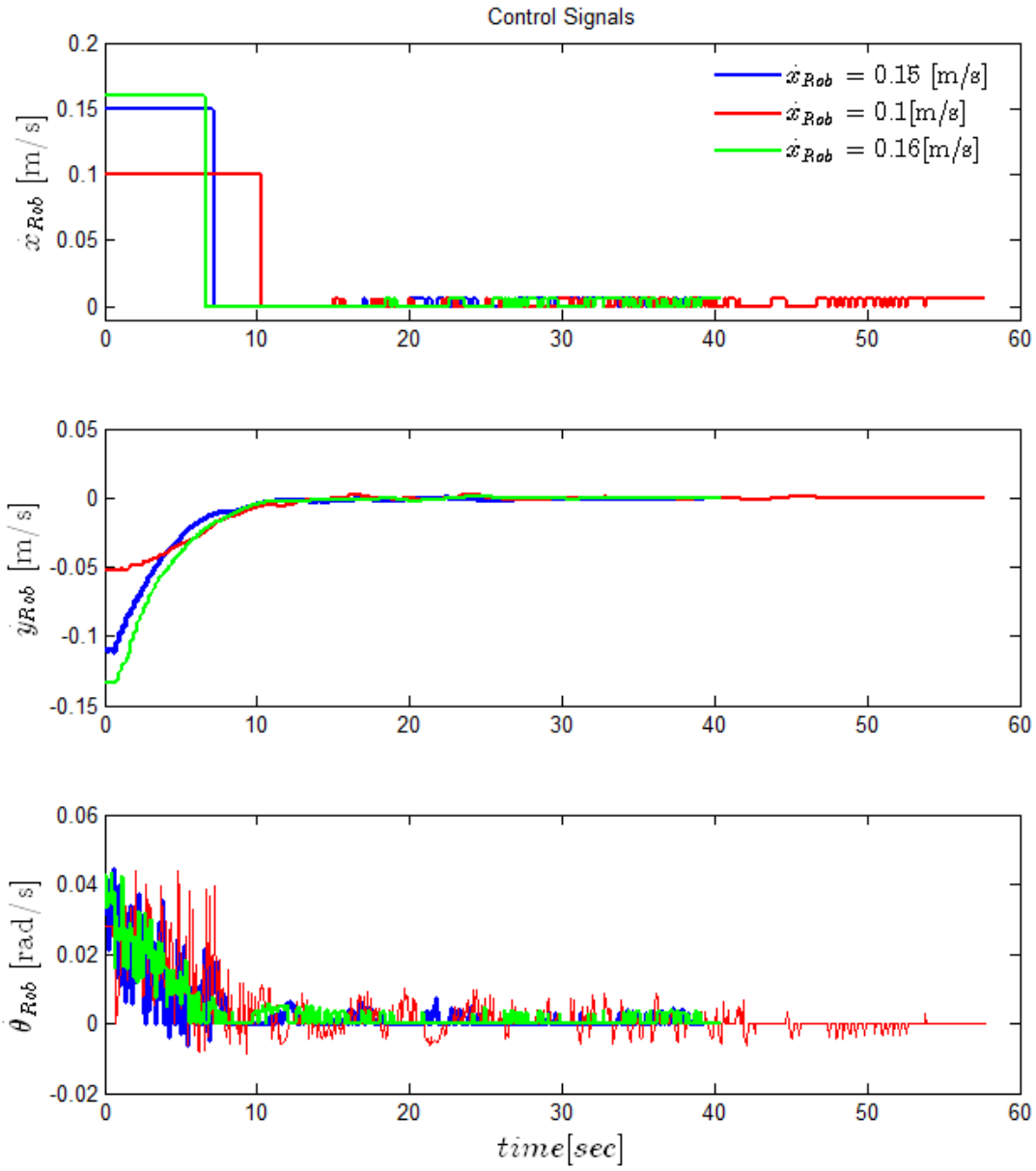


Figure 27. Control signals along axes of the Rob@work 3.

The corresponding position trajectories of the above experiments are plotted in Figure 28 which depicts the whole area of the docking tasks, the SM zone, and the target which is sketched as the circle with the origin of reference value of the marker ($x_o = 0.2$ m, $y_o = 0.0085$ m) and $x_{thresh} = 1$ mm.

According to Figure 27, it takes more time for the Rob@work 3 to approach the SM zone (roughly 10 seconds) if less forward velocity ($\dot{x}_{Rob}$) is applied. In this case, approach time is increased and the robot is slower in docking. However, Figure 28 declares less offset along the y_{mar}-axis which is approximately 2.5 cm with slower motion along the x_{mar}-axis and more optimized trajectory.

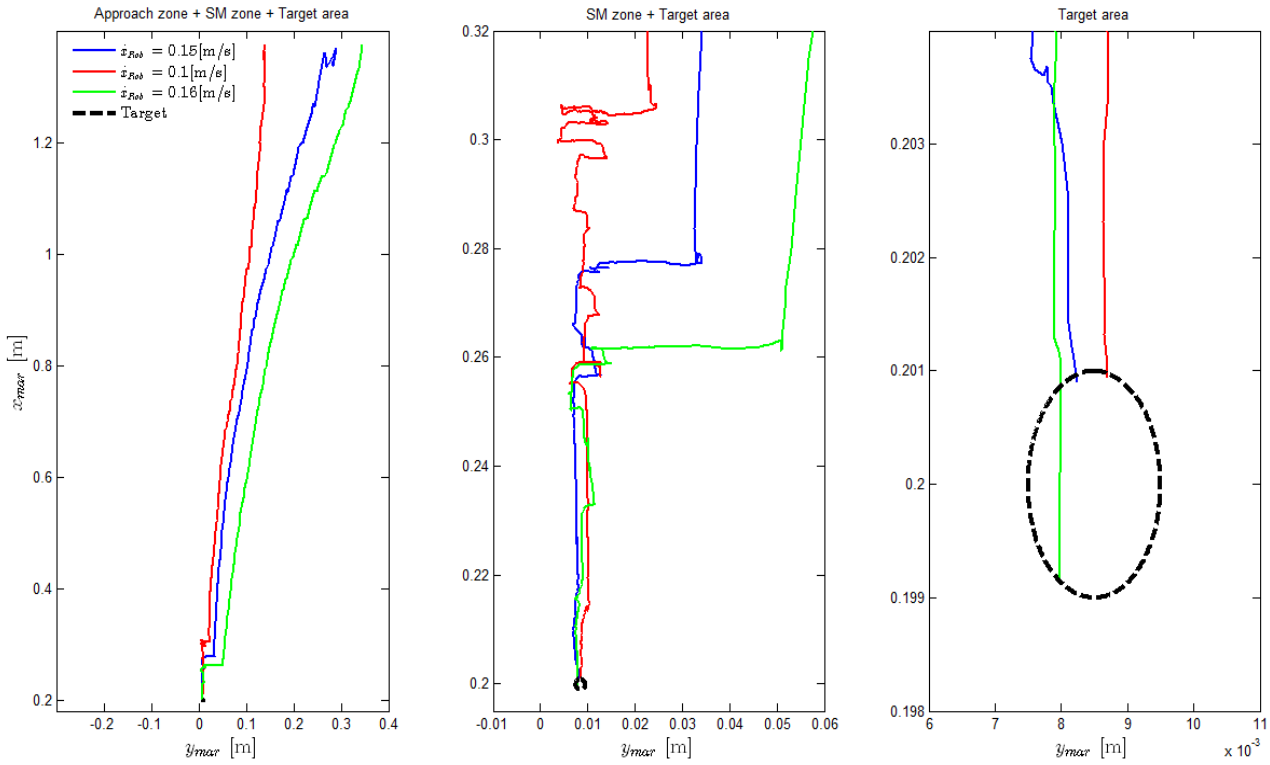


Figure 28. Docking trajectory.

Therefore, the forward velocity is chosen such that the approach time and the offset along the y_{mar}-axis is a compromise between the desired trajectory and less average docking time. Therefore, $\dot{x}_{Rob} = 0.15\,\frac{\mathrm{m}}{\mathrm{s}}$ is selected as the forward velocity for the further control design. This constant velocity, however, is only applied when the robot is inside the approach zone. In the final zone in which movement along the y_{mar}-axis and the θ_{mar}-axis is adjusted, very small constant velocity of $\dot{x}_{Rob} = 6\,\frac{\mathrm{mm}}{\mathrm{s}}$ is applied to finalize the docking.

After choosing the appropriate constant velocity for the approach and the SM zones along the x_{Rob}-axis, experiments with different gains were conducted to compare the effect of gain tuning to obtain the desired performance of the controller. Overall, six gains are tuned for two controllers along the y_{Rob}-axis and the θ_{Rob}-axis. However, the y_{Rob}-axis is prioritized since the difference in the orientation is typically less than the position. Therefore, for this experiment, gains of the controllers are listed in Table 4 to adjust the position and the orientation.

Table 4. Docking experiments with different PID gains for controller.

	Exp. / Gains	1	2	3	4
Position (y_{Rob}-axis)	K_P	0.86	0.66	0.86	0.51
	K_I	0	0	0	0.0005
	K_D	0.002	0.1	0.1	0.05
Orientation (θ_{Rob}-axis)	K_P	0.08	0.08	0.08	0.08
	K_I	0	0	0	0
	K_D	0	0	0	0

In the controller, the pose estimation of the camera with respect to the fixed marker leads to computation of the measurement errors and eventually control signals along the y_{mar}-axis and θ_{mar}-axis to adjust the position and the orientation of the mobile robot, respectively. Figure 29 presents the camera position for different gains recorded in the fixed marker frame. The x-axis of the plots is the real time converted from the ROS time using (13) and (14) even though the real experiments were conducted with the ROS time.

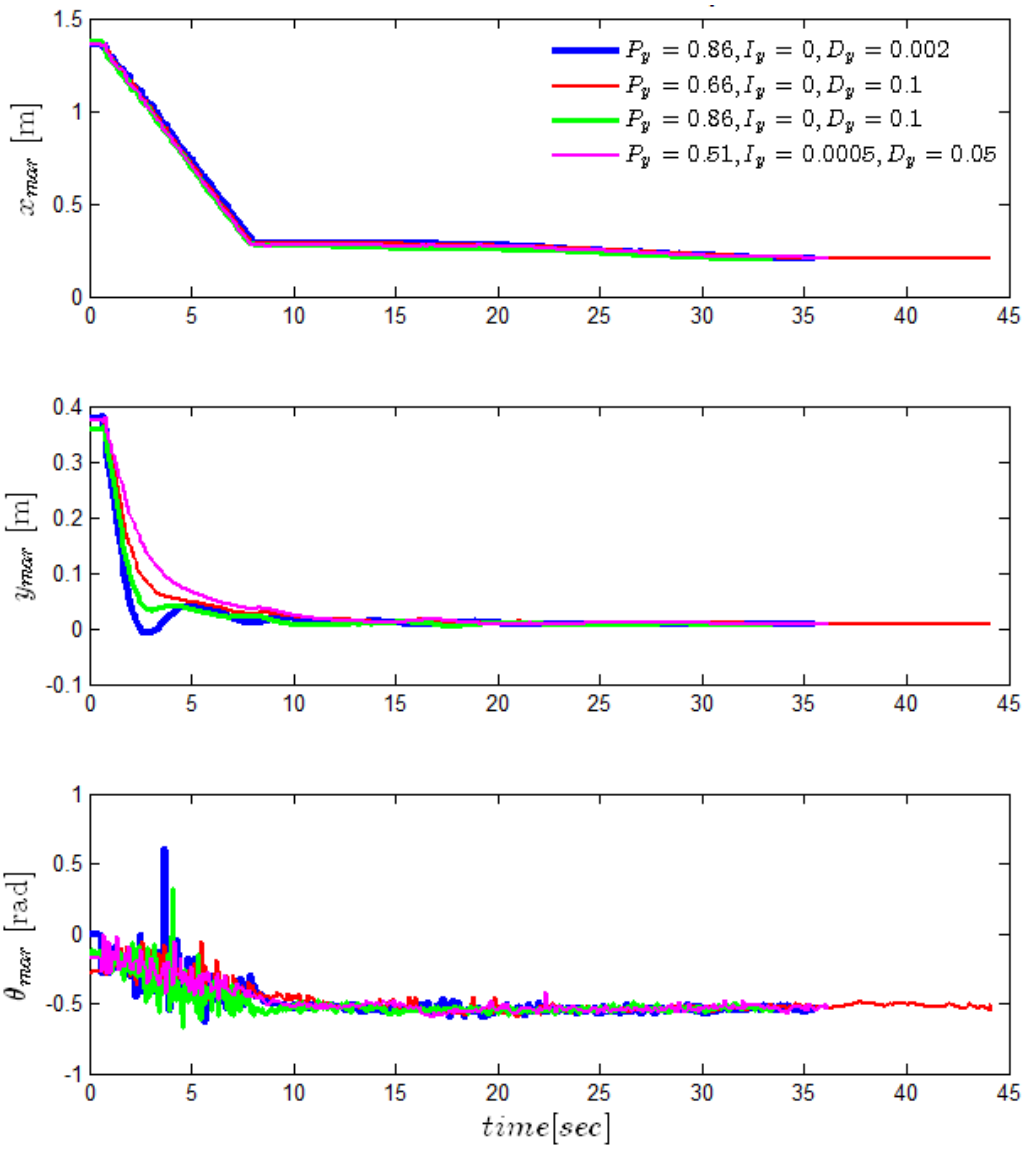


Figure 29. Camera pose estimation in the marker frame.

Theoretically, the slope of the position versus time plot indicates the velocity of the mobile robot. According to Figure 29, the position of the robot along the x_{mar}-axis is linear in all the experiments. The vision feedback control is not applied along the x_{mar}-axis and the velocity of the robot should thus be constant along the x_{Rob}-axis which can be observed in Figure 30 ($\dot{x}_{Rob} = 0.15\,\frac{m}{s}$).

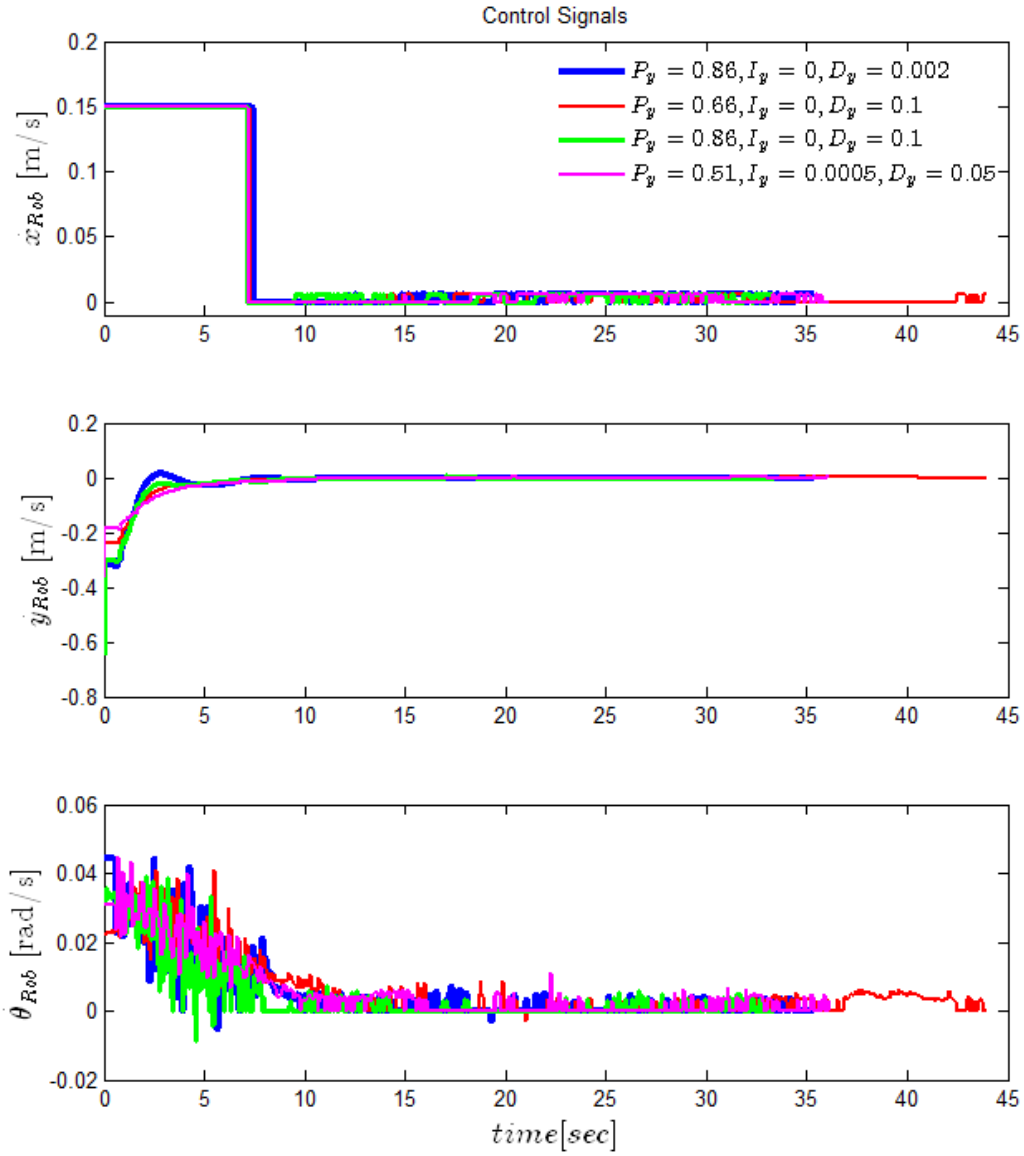


Figure 30. Control signals along the axes of the Rob@work 3.

On the contrary, the y_{mar}-axis and the θ_{mar}-axis show the visual feedback to compute the control signals along the y_{Rob}-axis and the θ_{Rob}-axis simultaneously as depicted Figure 30. In each case, the average docking time is evaluated with different PID gains. According to Figure 30, the slowest controller takes almost 44 seconds to accomplish the docking completely considering 7.5 seconds for transferring to the SM zone which is quite fast and reasonable according to limited space available in the lab for docking area.

The SM zone, however, is the main reason that docking takes longer time to be finished. It is made to increase the accuracy of the measurements along the y_{mar}-axis and the θ_{mar}-axis and the smoothness of the docking considering the fact that there is no forward movement along the x_{mar}-axis, meaning that $\dot{x}_{Rob}^{SM} = 0$. As soon as the measurement errors for the position along the y_{mar}-axis and the orientation along the θ_{mar}-axis are equal or less than the defined thresholds, the Rob@work 3 moves toward the docking platform with very slow speed until docking is completed. The successful docking is judged by a 1 mm threshold along the x_{mar}-axis and if the robot fulfills this requirement, the process is successfully completed and the program terminates.

The corresponding trajectories of the Rob@work 3 docking with different controllers are presented in Figure 31. The docking trajectory is practically the y_{mar}-x_{mar} plot which illuminates movement toward the docking platform.

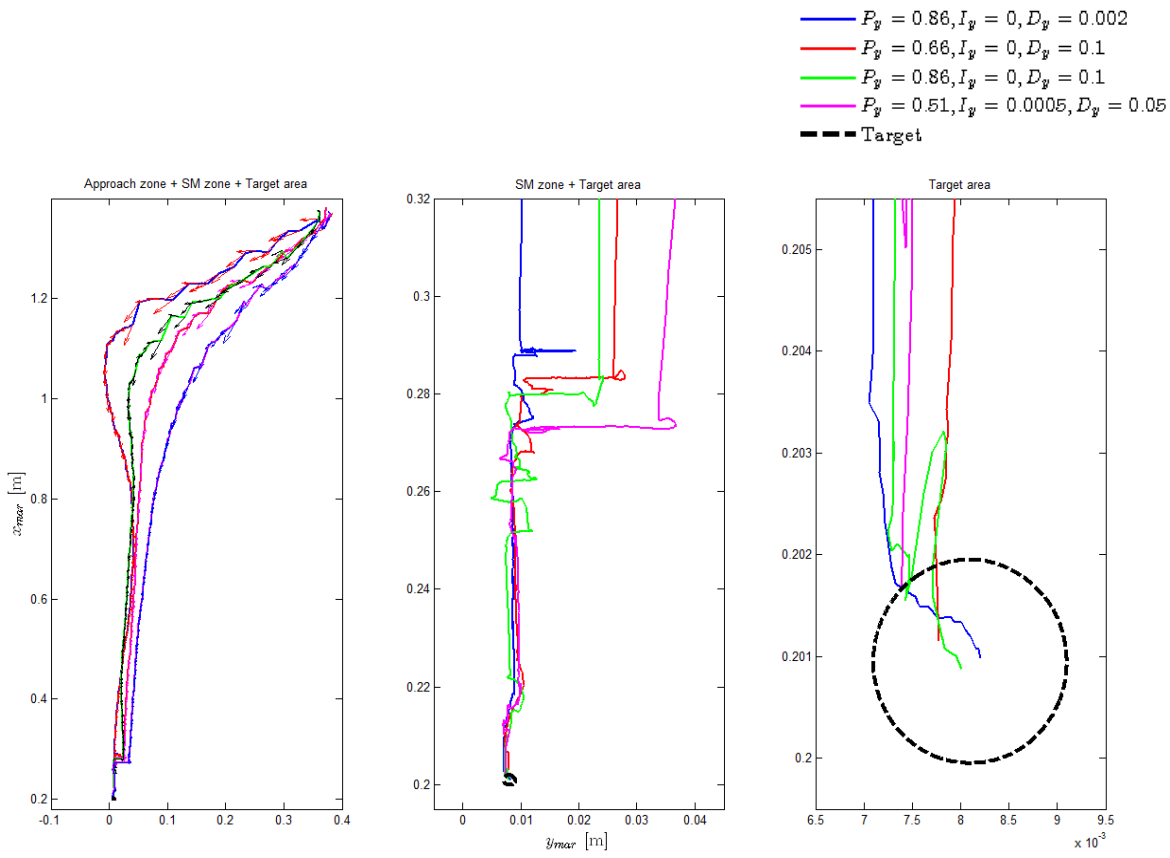


Figure 31. Docking trajectories for different vision-feedback controllers.

Figure 31 depicts the docking trajectories of the different designed controller for the entire area, the SM zone, and the target area. The SM zone clarifies the obtained offsets along the y_{mar}-axis for different controllers in the conducted experiment, whereas the target area represents the circle with the reference origin of the marker recorded once when the Rob@work 3 was manually docked $(x_{mar}^{ref}, y_{mar}^{ref}, \theta_{mar}^{ref})$ and $x_{thresh} = 1$ mm. The orientations of different controllers are also addressed with arrows on each trajectory while moving in the y_{mar}-x_{mar} plot.

The average docking time and the trajectory of the Rob@work 3 can now be evaluated with conducted experiments with the different controllers. Table 5 summarizes the features of the four implemented controllers.

Table 5. Comparison of different controllers in the vision-based docking of the Rob@work 3.

Feature / Exp.	Docking Time [sec]	Offset (y_{mar}-axis) [m]	Successful Docking	Maximum Overshoot [m]
1 (Blue)	35.52	0.01957	Yes	0.0164
2 (Green)	33.46	0.02422	Yes	NaN
3 (Red)	44.09	0.02802	Yes	NaN
4 (Pink)	36.23	0.03684	No	NaN

According to Table 5, the first and second experiments are quite fast and have more desirable and optimal response to fulfill the criteria of the successful docking. The second experiment has the shortest average docking time even though the offset along the y_{mar}-axis is larger than the first one. Besides, the first experiment shows the overshoot while approaching the target even though it is quite small. Overall, gains from the second (green) experiment are taken into account for further applications.

To test the designed controller of the conducted experiments shown in Table 5, the robot is placed in the three different initial configurations in the docking area to evaluate the efficiency

of the designed controller. In this experiment, the Rob@work 3 is placed in different sides to evaluate the designed controller. In this experiment, the extreme orientations (number 3 and number 4 in Figure 8) are selected for the mobile to test the controller.

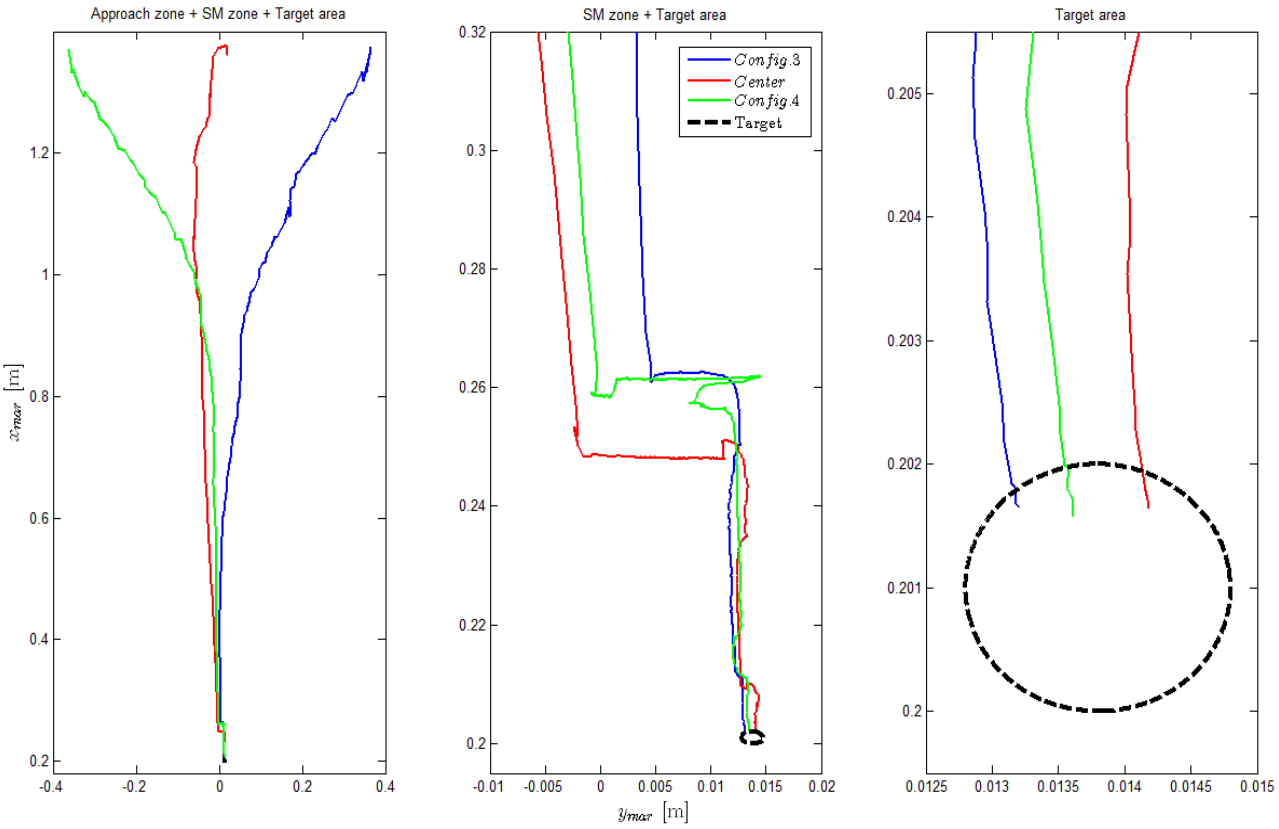


Figure 32. Obtained trajectory of the designed controller for three different configurations.

The camera position and orientation on the fixed marker coordinate system $(x_{mar}, y_{mar}, \theta_{mar})$ is depicted in Figure 33.

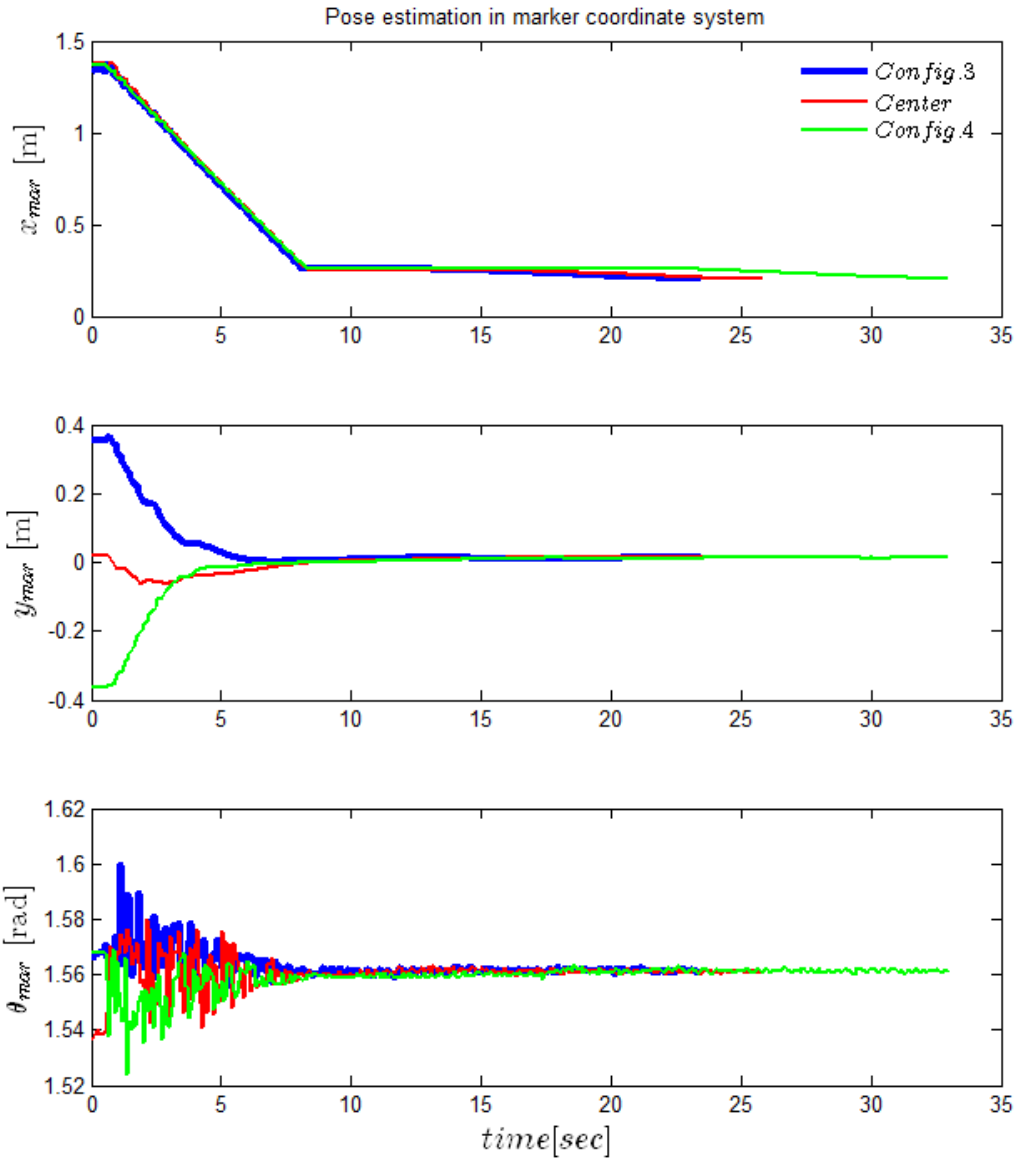


Figure 33. Camera pose estimation in marker frame.

The control signals for the robot coordinates ($x_{Rob}, y_{Rob}, \theta_{Rob}$) are illustrated in Figure 34.

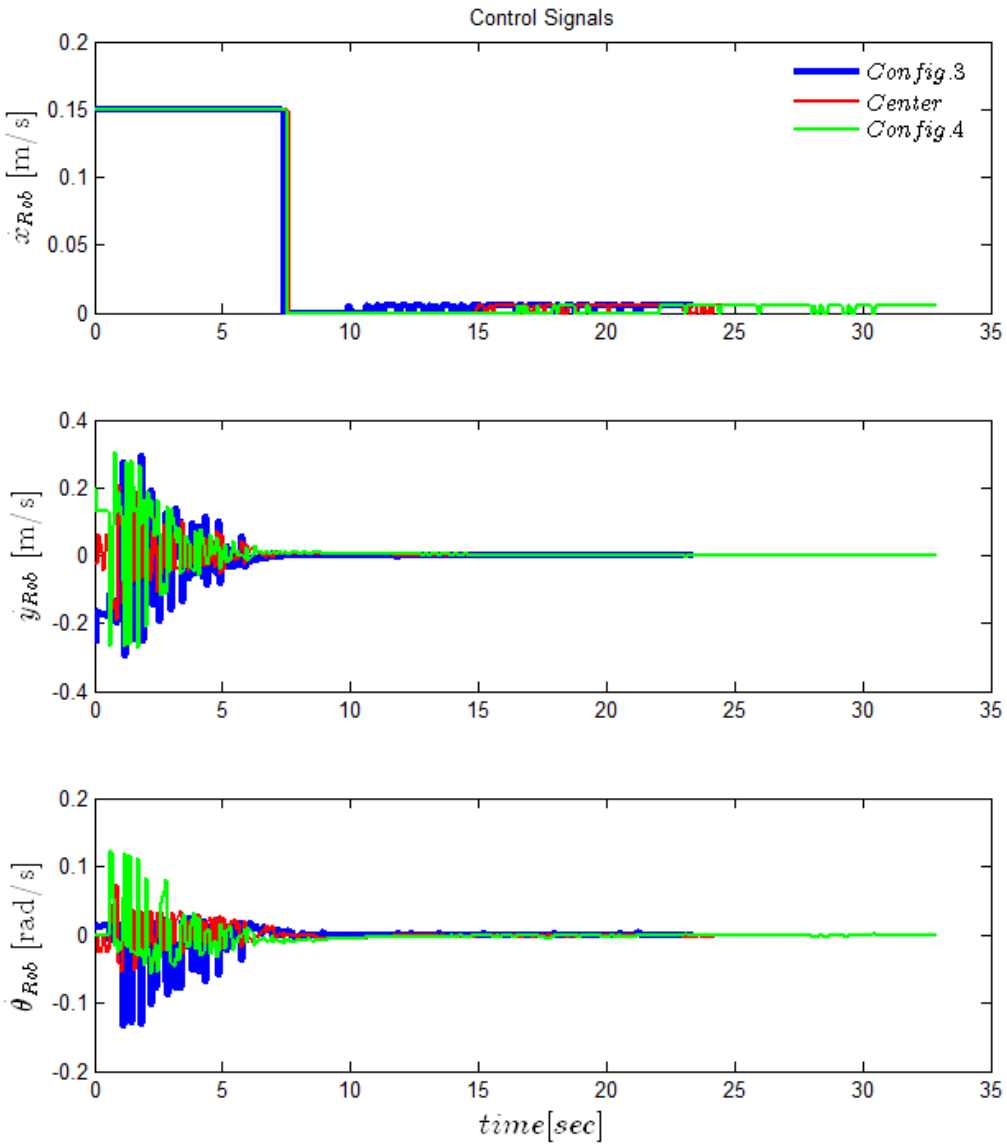


Figure 34. Control signals for the Rob@work 3 along the axes of the Rob@work 3.

In this experiment, the configuration 3 which resembles to the extreme orientation on the left side of the docking area has the quickest response for the docking time ($t_{docking} = 23$ s), while the offset along the y_{mar}-axis is about 2 cm.

3.4 Reinforcement Learning

As stated in Section 2.5.3, the idea of using the model-free RL technique is to develop the optimal action policy and compare the results with vision-feedback control. The optimal

policy of the Q-learning is evaluated by the number of time steps and the shortest path toward the goal.

The time step is practically the number of rows or columns in the Q-matrix. Mathematically, the rows and the columns are multiplied to give the maximum number of the steps. Accordingly, the robot is rewarded if it achieves its goal in less than or equal to the maximum steps, whereas punished if it exceeds the defined time steps. The reward distribution to obtain the optimal policy for docking the Rob@work 3 is summarized in Table 6.

Table 6. Reward distribution for the Q-learning.

Distribution		Reward [Unit-less]
Goal achieved	> Time step	-10
	< Time step	+50
Obstacle detected (by Laser Scanner)		-60
Marker lost (by Camera)		-60
Robot outside grid		-60

The rows and columns of the grid are the recorded orientation and position of the camera, attached to the front side of the robot (θ_{CAM}, y_{CAM}) respectively which are real numbers, whereas the indexes of the Q-matrix should be integers. Therefore, the following function is used as the segment part of the code to discretize the values in the real time for the simulation. In this function, the orientation and the position are converted based on the grid segment and the sample rate (SR) as the number of rows or columns calculated below:

$$\begin{cases} \theta_{seg} = \frac{\theta_{up} - \theta_{dwn}}{\theta_{SR}} \\ y_{seg} = \frac{y_{up} - y_{dwn}}{y_{SR}} \end{cases} \quad (20)$$

The pseudo-code of the developed algorithm is presented in Algorithm 3.

```
void i_j_Generator(double y, double theta)
{
         int sample_rate_y = col - 1;
         int sample_rate_theta = row - 1;
         double div_theta = (theta_up - theta_dwn)/sample_rate_theta;
         double div_y = (y_up - y_dwn)/sample_rate_y;

         // i selection
         for (int k = 0; k <= (row - 1); k++)
        {
if (abs(theta) >= (theta_dwn - ((.5 + k) * div_theta)) &&
                  abs(theta) <= (theta_dwn + ((.5 + k) * div_theta)))
{
       i = k;
       break;

} else if (abs(theta) < (theta_dwn - ((.5 + 0) * div_theta)) ||
                  abs(theta) > (theta_dwn + ((.5 + (row - 1)) * div_theta)))
{
         ROS_INFO (" Outside theta - boundary => NEGATIVE REWARD!!");
}
         }

         // j selection
        for (int l = 0; l <= (col - 1); l++)
        {
 if (y >= (y_dwn - ((.5 + l) * div_y)) &&
                                              y <= (y_dwn + ((.5 + l) * div_y)))
 {
         j = l;
         break;
 } else if (y < (y_dwn - ((.5 + 0) * div_y)) ||
                                        y > (y_dwn + ((.5 + (col - 1)) * div_y)))
{
         ROS_INFO (" Outside Y - boundary => NEGATIVE REWARD!!");
}
         }
}
```

Algorithm 3. The pose conversion to the rows and columns of the Q-matrix.

The Q-learning configuration demands all components, including the target, to be represented with the corresponding integer indexes. Therefore, the goal should be identified as a state which can be recorded when the robot is docked manually ($i_{Goal} = 1, j_{Goal} = 4$) before the training is started. Figure 35 illustrates the trained path toward the goal with the integer indexes for the orientation and the position (θ_{CAM}, y_{CAM}) in the fixed marker frame. Each index resembles to a specific state inside the grid which is also recognized with the real value in the docking area.

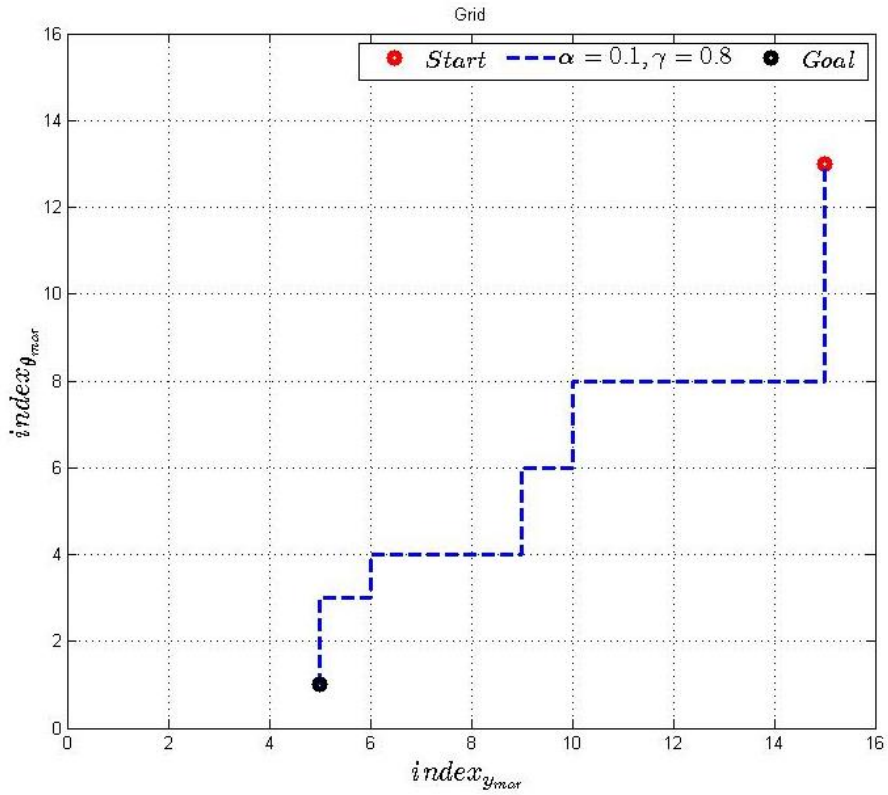


Figure 35. Optimal trajectory obtained after training with Q-learning.

Figure 35 depicts the results of the learning process of 1050 iterations to update the Q-matrix and obtain an optimal action policy for the velocity commands and eventually convergence to the highest reward according to Table 6. Figure 35 also illustrates the result of the learning system to find the optimal path toward the goal, considering the fact that the initial position is randomly chosen inside the grid, whereas the goal is always placed at the same state. The trajectory is plotted after extracting the Q-matrix to compare each component to find the

shortest path toward the docking platform, which is defined when the mobile robot initially was docked manually. Furthermore, the goal represents the convergence of the Q-learning to an optimal Q-function with $\alpha = 0.1$ and $\gamma = 0.8$ while the single entry is updated in each step.

Although Figure 35 is not graphically compared with another path to be judged as the optimal policy in docking, the components of the Q-matrix are constantly evaluated with the neighbors to obtain the shortest path toward the goal. In this approach, the initial position could be randomly chosen inside the grid with indexes to simplify the simulation.

The real experiment and simulation are slightly different considering the fact that the robot is mechanically restricted to move horizontally on the goal state to finish the docking in the real experiments even though horizontal movement is not restricted in other states of the grid. This is because of the mechanical constraints of the platform for aside movements along the y_{mar}-axis to secure the smooth docking with end-effectors. Therefore, the setup demands forward movements toward the goal when the robot is in the goal state. This restriction is added to the simulation for the goal state to prevent any inconsistency in which no horizontal movements are observed in Figure 35.

4 DISCUSSION

The autonomous docking of mobile robots demands accurate sensor measurement of the docking platform. In this thesis, the feasibility of the sensor integration for the laser scanner and the vision sensors was investigated to obtain a docking trajectory with the low average docking time.

According to the datasheet of the S300 Sick Safety scanners, a resolution of 1-3 cm is expected in detecting the obstacles (S300 Standard 2016). According to the experiments performed with different shaped markers and the S300 laser sensor, the desired docking of the Rob@work 3 is not obtained. The S300 laser scanner is quite useful for safety purposes rather than tasks which demand high precision.

The experimental results show various detected obstacles in different angles which indicate the inconsistent data of the laser scanner sensors to achieve the docking task with high precision. Therefore, a vision sensor was added to increase the accuracy of the position estimates.

In the vision-based approach, the measurements were integrated with pose estimation of the camera with respect to the fixed marker to design the vision-feedback control system to increase the precision. The target is represented by the marker, which is capable of publishing θ_{mar}^{ref}besides of x_{mar}^{ref} and y_{mar}^{ref}.

The ARTags with the unique marker ID are prioritized to the simple point fiducials since they provide the orientation of the Rob@work 3, in contrast to the two-dimensional positions estimated with the point fiducials. Besides, Figure 9 and Figure 12 illustrate the detected markers on the docking platform in which the ARTags showed better detectability even if the docking area is full of noise caused by wires and other surrounding objects with the less illumination. The raw orientation measurement and the vision control design along the θ_{mar}-

axis ensure the precise docking even if the mobile robot is not initially perpendicular to the target. ARTags depict higher transparency for detection from further distances in the environments with which challenging illuminations exist in the workspace.

In the control design, The PBVS is developed as the vision feedback based on $X_{CAM}$with respect to the fixed marker frame. The camera and marker frames are added to the Rob@work 3 with the ROS transformation package which lets the user track multiple coordinate systems in real-time (Saito 2015). The control signal computations demand accurate measurements of the ARTag for x_{CAM}, y_{CAM} and θ_{CAM}.

The Android and the USB cameras mounted on the front side of Rob@work 3 were employed to compare the visualization time to detect the ARTag. The Android phone is used as the IP camera to embed live video as part of the main program for the pose estimation. The stream from the IP camera displays the docking area with the marker attached to the docking platform as a fixed coordinate system. Since the video stream is via an IP address on the shared network between the Rob@work 3, the Android phone and the local PC, the data transformation delay is larger than the USB camera and yields jitters resulting in shaky movement of the Rob@work 3. The task takes approximately 70-80 ms to detect the ARTag and compute the control signals if the USB camera is employed as the vision sensor, whereas it is approximately double (130-140 ms) if the IP camera is utilized.

Although ROS is quite fast for the online task scheduling, it does not always provide the most deterministic timing of certain tasks. According to Figure 26, the average sampling time for both velocity and position samples seems constant even though the ROS time illustrates jitter on both plots. Therefore, the ROS time does not seem practical for tasks with high frequency PID or motion control.

Moreover, the sampling time for the USB camera exhibits fewer oscillations than the IP camera. The network between the Rob@work 3 and the Android phone is a typical 2.4 GHz wireless band with the maximum speed of 54 Mbps which may not have enough bandwidth for transmitting the MPEG-4 video stream with a resolution of 640×480 pixels and 30 fps.

This indicates the reason for the shaky behavior of the mobile robot when the IP camera is employed as the vision sensor in the docking of the Rob@work 3.

The Rob@work 3 is considered as the plant under investigation in the control system depicted in Figure 14 to design an appropriate controller for the docking task. However, due to the sophisticated computations of the under-carriage control, the plant is considered linear as a first order integrator to simplify the computation. The control signals are the velocity commands applied to the Rob@work 3 to move toward the docking platform. The practical commands are the linear velocities along the x_{Rob}-axis, the y_{Rob}-axis, and $\dot{\theta}_{Rob}^{CS}$ along the θ_{Rob}-axis.

The movement along the x_{Rob}-axis does not require a controller since the Rob@work 3 can move toward the docking platform with constant velocity. However, the y_{Rob}-axis and θ_{Rob}-axis require the vision-feedback controllers to achieve the target accurately. According to Figure 27, the constant velocities along the x_{Rob}-axis on the different zones have direct impact on the obtained trajectory and the average docking time. Although, the slower movement of the Rob@work 3 along the x_{Rob}-axis in the approach zone may decrease the offset along the y_{mar}-axis to get as close as possible to the reference in the SM zone, it would considerably increase the docking time.

Perhaps the biggest challenge to obtain the optimal action policy in the model-free approach of the RL framework is the convergence on the real experiments in which practical constraints of diminish the desired outcome. Among the constraints, the limited power supply of the mobile robot and the complex dynamics of the docking platform make the learning process quite challenging in real experiments. For this reason, the simulation is alternatively employed for the learning system in which the number of iterations is crucially important to achieve the convergence and the behavior with less failure.

On the one hand, the larger number of iterations may increase the chance of better convergence but because of the limited power resources, the operating time, and the

mechanical setups of the Rob@work 3, several iterations may not be feasible in real experiments. Iterations above hundreds are infeasible in real experiments during the training. The simulation environment, on the other hand, does not fully contain the mechanical constraints, the tool changers and the end-effectors on the robot and the docking platform and is rather developed for the ideal yet simple goal achievement experiment.

Another challenge to utilize the Q-learning is the pose estimation inside the discretized grid which is represented with indexes after the formula used for conversion. A smaller grid is not accurate enough to determine the exact position of the robot compared to the vision-feedback control approach. It can be improved with a larger grid since it increases the accuracy of the acquired camera position on the robot in the docking area even though it demands higher computation time and power in the learning process.

5 CONCLUSION

The mechanical setups of the tool changers mounted on the docking platform and the Rob@work 3 demand smooth, safe, and flexible motion control with a very precise alignment along its collinear joint axis to accomplish the docking. For this purpose, multiple sensors are employed to evaluate the docking time and the obtained trajectory. Since the position accuracy is necessary in this project, the already built-in laser scanner sensors are not sufficient to accomplish the task. Therefore, a vision sensor is added to the Rob@work 3 to increase the accuracy and robustness with the designed vision-feedback control.

Docking the mobile robot was experimentally investigated with multiple sensors for docking the Rob@work 3. In each case, the docking platform is identified by different markers as the target. The laser scanners depicted several inconsistencies while detecting the marker. Among the experiments, neither cylindrical bottle nor the box showed high accuracy to be observed by the laser scanners when the robot was docked manually.

The S300 Sick laser scanner has 270° angular range and the radial distance range of 30 m to perceive the surrounding objects. Theoretically, the laser beams on the sensor has the accuracy of 1-3 cm when two consecutive samples are considered. However, the conducted experiments with different markers illustrate inconsistencies in the marker detection if the safety line of 10 cm is considered. The laser sensors are used for the safety functionality and the vision sensor is added to increase the accuracy of the Rob@work 3 to obtain the docking.

The computer-vision algorithm in OpenCV is used to calibrate the vision sensors to determine camera parameters and remove distortion from the image. The file containing the parameters and the marker size are employed to run the main visualization process. In the process, the position and the orientation of the camera are estimated with respect to the docking platform as soon as the marker is found. () is published with respect to the platform of the robot with the ROS transformation package (Saito 2015). However, the camera coordinates must be

known with respect to the docking platform. Therefore, a transformation matrix is used to obtain the pose estimation of the camera with respect to the fixed marker.

In the vision-feedback control design, the planar fiducial, known as the ARTag, is prioritized over the point fiducials since the orientation can also be measured with such markers. The position and the orientation of the camera are published according to the marker frame and the measurements are used for the feedback control.

As an alternative to achieve the docking, the model-free Q-learning approach was investigated to compare the optimal docking behavior with the vision-feedback control system. The idea is to obtain the optimal action policy by accumulating prior knowledge to achieve optimal trajectory within lowest possible time steps. The robot is rewarded in the goal state depending on achievement of the docking.

The size of the developed grid in the Q-learning method is critically important and has direct impact on the result accuracy even though it takes more computation time. The local Prace computer (see Figure 3) has sufficient memory to deal with large iterations in simulation. Training the real robot demands simplified grid and smaller Q-matrix, whereas simulation restricts neither the grid size nor iterations.

The Q-learning is developed in the simulation environment for some practical reasons. First, the approach is model-free, meaning that mechanical constraints of the Rob@work 3 and the complex model of the collinear joints with the tool changers on the docking platform are not essential to be considered in simulation. Second, training in the larger grid with more iteration is feasible in the simulation environment rather than real experiments. Next, convergence of the Q-learning is most likely to be obtained for the goal state. Finally, the limited power supply of the batteries on the Rob@work 3 is not sufficient to conduct the training experiments on the real platform.

The comparison between the results of the autonomous laser scanner-based docking, the autonomous vision-based docking and the machine learning method reveals that the vision-

feedback control system is more accurate and reliable for docking of the Rob@work 3 considering the criteria of the docking with respect to time and trajectory. The PBVS approach for the vision-feedback control has shown a great performance for docking the Rob@work 3 in this thesis.

LIST OF REFERENCES

Connette, C. P., Parlitz, C., Hagele, M. & Verl, A. 2009. Singularity avoidance for over-actuated, pseudo-omnidirectional, wheeled mobile robots. IEEE International Conference on Robotics and Automation, Proceedings of the International Conference, held at Kobe, Japan 12-17 May, 2009. IEEE. Pp. 4124-4130.

Corke, P. 2013. Robotics, Vision and Control Fundamental Algorithms in MATLAB®. Germany: Springer. 570 p.

Duda, R. O. Hart, P. E. & Stork, D. G. 2000. Pattern Classification. United States: John Wiley & Sons, Inc.. 320 p.

Even-Dar, E. & Mansour, Y. 2003. Learning Rates for Q-learning. Journal of Machine Learning Research, 5:1. Pp. 1-25.

Fiala, M. 2005. ARTag, a fiducial marker system using digital techniques. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Proceedings of the International Conference, held at San Diego, United States of America 20-25 June, 2005. IEEE. Pp. 590-596.

Forsyth, D. A. & Ponce, J. 2012. Computer Vision: A Modern Approach. United States: Pearson Prentice Hall. 813 p.

Fraunhofer-Gesellschaft. 2009. rob@work [web document]. Published 2009, updated 20.10.2016. [Referred 24 October 2016]. Available: http://www.care-o-bot.de/en/rob-work.html

Gaskett, G. 2002. Q-learning for Robot Control [web document]. Canberra: October 2002 [Referred 27.10.2016]. Doctoral thesis. The Australian National University. 200 p. + indexes 2

p. Available in PDF-file: https://openresearch-repository.anu.edu.au/bitstream/1885/47080/6/02whole.pdf

Gomes, D. 2013. roscpp/Overview/Time [web document]. Published 2013, updated 21.12.2013. [Referred 1.9.2016]. Available: http://wiki.ros.org/roscpp/Overview/Time

Gonzalez, R. C. & Woods, R. E. 2008. Digital Image Processing. United States: Pearson Prentice Hall. 793 p.

Hachour, O. 2009. The proposed path finding strategy in static unknown environments. In: International Journal of Systems Applications, Engineering & Development, 3:4. Pp. 127-138.

Hutchinson, S., Hager, G. D. & Corke, P. I. 1996. A Tutorial on Visual Servo Control. IEEE Transactions on Robotics and Automation, 12:5. Pp. 651-670.

Jun, L., Lilienthal, A., Martinez-Marin, T. & Duckett, T. 2006. Q-RAN: A Constructive Reinforcement Learning Approach for Robot Behavior Learning. IEEE International Conference on Intelligent Robots and Systems, Proceedings of the International Conference, held at Beijing, China 9-15 October, 2006. IEEE. Pp. 2656-2662.

Kim, K. H., Choi, H. D., Yoon, S., Lee, K. W., S., Ryu, H. S., Woo, C. K. & Kwak, Y.K. 2005. Development of Docking System for Mobile Robots Using Cheap Infrared Sensors. 1st International Conference on Sensing Technology, Proceedings of the International Conference, held at Palmerston North, New Zealand 21-23 November, 2005. IEEE. Pp. 287-291.

LaValle, S. M. 2006. Planning Algorithms. United States: Cambridge University Press. 842 p.

Lepetit, V. & Fua, P. 2005. Monocular Mode-Based 3D Tracking of Rigid Objects: A survey. Foundations and Trends® in Computer Graphics and Vision, 1:1. Pp. 1-89.

Nguyen, V., Martinelli, A., Tomatis, N. & Siegwart, R. 2007. A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics. Autonomous Robots, 23:2. Pp. 97-111.

Nilsson, S. 2010. Real-time Trajectory Generation and Control of a Semi-Omnidirectional Mobile Robot [web document]. Lund: May 2010. [Referred 20.8.2016]. Master of Science thesis. Lund University, Department of Automatic Control. 68 p. Available in PDF-file: http://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=8847433&fileOId=8859294

Ogata, K. 2010. Modern Control Engineering. United States: Pearson Prentice Hall. 894 p.

OpenCV. 2011a. Camera Calibration and 3D reconstruction. [web document]. Published 2011, updated 30.8.2016. [Referred 30.8.2016]. Available: http://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html#rodrigues

OpenCV. 2011b. More Morphology Transformations. [web document]. Published 2011, updated 24.10.2016. [Referred 24.10.2016]. Available: http://docs.opencv.org/2.4/doc/tutorials/imgproc/opening_closing_hats/opening_closing_hats.html

OpenCV. 2011c. Miscellaneous Image Transformations. [web document]. Published 2011, updated 31.8.2016. [Referred 31.8.2016]. Available: http://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html#cvtcolor

OpenCV. 2011d. Camera calibration With OpenCV. [web document]. Published 2011, updated 7.9.2016. [Referred 7.9.2016]. Available: http://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html#camera-calibration-with-opencv

OpenCV. 2015. Detection of ArUco Markers. [web document]. Published 2015, updated 31.8.2016. [Referred 31.8.2016]. Available: http://docs.opencv.org/3.1.0/d5/dae/tutorial_aruco_detection.html

Poole, D. L. & Mackworth, A. K. 2010. Artificial Intelligence Foundations of Computational Agents. United States: Cambridge University Press. 662 p.

Qt. 1994. Cross-platform software development for embedded & desktop. [web document]. Published 1994, updated 29.8.2016 [Referred 29.8.2016]. Available: https://www.qt.io

RoboRealm. 2005. Color Object Tracking. [web document]. Published 2005, updated 31.8.2016. [Referred 31.8.2016]. Available: http://www.roborealm.com/tutorial/color_object_tracking_2/slide010.php

Russell, S. & Norvig, P. 2010. Artificial Intelligence: A Modern Approach. United States: Pearson Prentice Hall. 1132 p.

S300 Standard. 2016. [www- data sheet]. Germany: Sick AG, 2016. [Referred: 29 August 2016]. Available: https://www.sick.com/media/pdf/3/53/853/dataSheet_S30B-3011BA_1056427_en.pdf

Saito, I. 2015. tf - ROS Wiki. [web document]. Published 2015, updated 19.7.2015. [Referred 30.8.2016]. Available: http://wiki.ros.org/tf

Sutton, R. S. & Barto, A. G. 1998. Reinforcement Learning: An Introduction. United States: MIT Press. 320 p.

Teixidó, M., Pallejà, T., Font, D., Tresanchez, M., Moreno, J. & Palacín, J. 2012. Two-Dimensional Radial Laser Scanning For Circular Marker Detection And External Mobile Robot Tracking. In: Sensors, 12:12. Pp. 16482-16497.

Ubuntu. 1991. Ubuntu: The leading OS for PC, tablet, phone and cloud. [web document]. Published 1991, updated 29.8.2016. [Referred 29.8.2016]. Available: http://www.ubuntu.com

Welsh, J. 2014. ArUco Marker Generator. [web document]. Published 2014, updated 31.8.2016. [Referred 31.8.2016]. Available: http://terpconnect.umd.edu/~jwelsh12/enes100/markergen.html

Wilson, W. J., C. C. W. Hulls & Bell, G. S. 1996. Relative End-Effector Control Using Cartesian Position Based Visual Servoing. In: IEEE Transactions on Robotics and Automation, 12:5. Pp. 684-696.

Camera Calibration

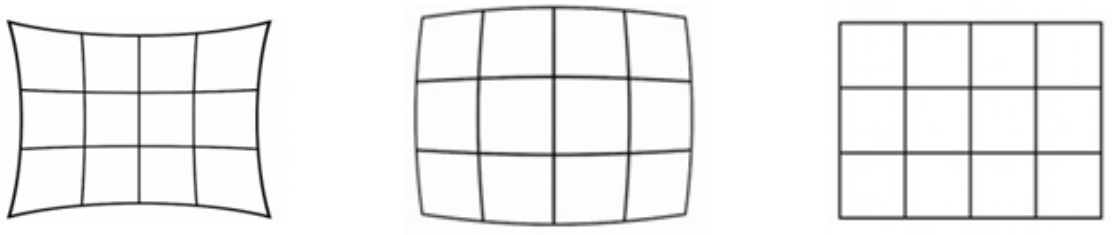
The pose estimation of the camera with respect to the fixed marker frame mounted at the docking platform depends on physical parameters of the camera (Forsyth et al. 2012, p. 129). The vision sensors in this project are not perfectly equipped with advanced high-tech lenses and entitled to significant distortion after being used for several times. Therefore, the calibration reduces the deviations between the captured and the actual image to remove the distortion of the lens.

In the image processing, the camera calibration is employed to correct the geometric lens distortions and estimate the internal parameters of the camera, known as intrinsic and extrinsic parameters. The calibrated camera has distortion parameters used for the further pose-estimation process. Internal parameters are calculated by detecting feature points belonging to the same shapes or lines on a single pattern and the results are represented below (Forsyth et al. 2012, pp. 129-131):

$$K_{3\times3} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix};\ M_{3\times4} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{21}$$

in which f represents the distance to the pinhole camera and the unit is meter. The lens distortion is categorized into the radial and tangential distortion. Geometrically, the radial distortion is depicted as curved lines, which are more visible on the edges of the image frame since it changes the distance between the image center and an arbitrary point on the frame while has no influence on the direction of the vector between two points (Forsyth et al. 2012, p. 139).

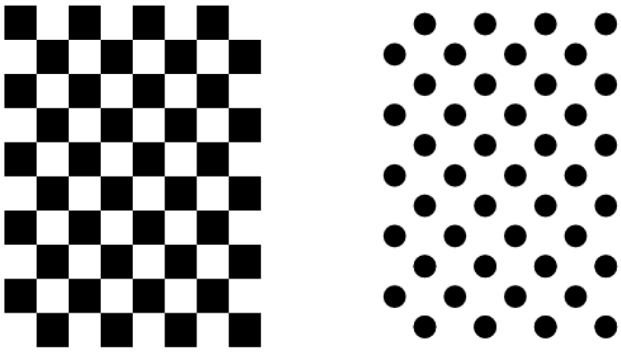
The final image has a pattern similar to one of the patterns depicted in Figure 36. An appropriate image for further image analysis is similar to Figure 36c); otherwise the calibration process is repeated.



a) Pincushion. b) Barrel. c) Rectilinear.

Figure 36. Different distortions in calibration process.

To calibrate the camera and derive the internal parameters, the standard patterns are employed as it is shown in Figure 37. They can vary from the classical black-white chessboard to the symmetrical or the asymmetrical circular pattern.



a) Rectangle. b) Symmetrical circle.

Figure 37. Common patterns in camera calibration process.

The source code of the computer vision algorithm to calibrate the vision sensors under investigation with the particular pattern is borrowed from (OpenCV 2011d). The calibration was performed to calibrate the android and the USB cameras. The OpenCV algorithm takes the input of the camera from the video stream, and detects the chessboard pattern.

The algorithm is briefly explained in Algorithm 4.

- Read the settings.
  - Import the XML file.
  - Check the validity of the file with a post processing function.
- Get the next input from camera then calibrate.
- Find pattern in current input, marked with a Boolean variable.
  - If chessboard, detect the corners of the squares.
  - If circle, detect circle.
  - Draw found points on the input image.
- Visualize the result for the user and show control commands for the application.
- Visualize undistorted image.

Algorithm 4. Camera calibration procedure with OpenCV.

After the calibration is done, the results are saved in the exclusive file for each camera which can be loaded for further image processing. The file is the human friendly file containing the essential camera parameters.

Project Source Code

The source code of the project, including the visualization, control design, the GUI and the RL can be accessed at https://github.com/mrgransky/Autonomous-Vision-Based-Docking-Rob-work-3 .