Autonomous Vision-based Docking of a Mobile Robot with four Omnidirectional Wheels

Farid Alijani

LUNDS UNIVERSITET

Department of Automatic Control

MSc Thesis
ISRN LUTFD2/TFRT—6018—SE
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Media-Tryck Lund 2016

Abstract

Docking of mobile robots requires precise position measurements in relation 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 sensor measurements are entitled to disturbances and measurement uncertainties. In this thesis, sensor integration is exploited to decrease the measurement errors and increase the accuracy of the docking.

The first approach in the autonomous docking of a mobile robot considered in this thesis is to use the already built-in laser scanner sensor to examine the feasibility of the precise docking with several experiments employing different markers.

The second approach is a vision-based control method with a marker mounted on the docking platform, identified as the target. The vision-feedback control system evaluates the current position of the robot relative to the target in real time to compute the velocity commands for the actuators to reach the docking platform. The versatility of the markers is investigated in this method.

The final approach is a Reinforcement Learning (RL) framework to investigate and compare the optimality of the docking with the vision-based control method. In the RL approach training is handled in a simulation environment with the reward distribution.

The obtained results of the approaches are evaluated in experiments based on the desired docking behavior with respect to the trajectory and time. The control design with a real-time vision system demonstrates the capabilities of this approach to conduct accurate docking of the mobile robot starting in different configurations.

Acknowledgment

I would like to express my gratitude to my examiner Anders Robertsson and supervisors, Björn Olofsson and Martin Karlsson at LTH, for their precious support during my MSc thesis and the freedom they gave me to go beyond the initial proposed topic to investigate the model-free Reinforcement Learning (RL) as an 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 the 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.

Contents

1. Introduction

This MSc thesis addresses the measured data of different sensors for docking of the mobile robots. Sensor integration is exploited to check the feasibility of the precise docking in real-time. The control method is designed for each sensor to evaluate the current state of the robot and the target. Laser scanners, computer vision, and machine learning methods are investigated to obtain the desired performance of docking with respect to time and shortest path.

1.1. Background

Docking is a critical task for industries employing mobile robots with platforms to be lifted or moved temporarily. It is also crucial for applications in which the mobile robots recharge their batteries autonomously (Kim, Choi, Yoon, Lee, Ryu, Woo, & Kwak 2005, p. 287). Moreover, autonomous operations are highly preferable in robotics since human interactions and errors are eliminated. Such behavior demands the mobile robots to accomplish a certain task precisely (Hachour, 2009, p. 127). For this purpose, the robot should be equipped with sensors to first localize itself in the unknown environment then achieve the final goal configurations. However, the sensor measurements are not flawless and can be inconsistent. Therefore, sensor integration could result in higher accuracy in the docking.

1.2. Motivation

The sensor integration is 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). This is particularly 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 influences

the pose-estimation accuracy or the environment compatibility which may cause irreversible damages on the workplace. Control system is an applicable tool to obtain the desired skills to perform the docking tasks and satisfy all the constraints concurrently.

1.3. 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 (DOF) which gives the robot 2D transformation and 1D 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, Parlitz, Hagele & Verl, 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 mobile robot has an on-board computer operating Linux Ubuntu 14.04 LTS (Ubuntu 1991) in which the image analysis and control commands for actuators to move the platform are performed simultaneously.

1.4. Problem Formulation

The end-effectors and the mechanical collinear joints mounted on the mobile robot and the docking platform demand a precision of 2-3 mm for the position and 1-3 degrees for the orientation to accomplish the docking task. Sensor integration is investigated to evaluate the accuracy and robustness of the pose estimation for sensors with respect to the docking platform.

However, multiple sensors are not the only aspect of the high accuracy in the docking of the robot. The control method which takes the feedback from the sensor measurements is a bridge between the sensor data and the desired docking behavior. The autonomous docking of the mobile robots demands accurate sensor measurements cooperating with the control design. In

this thesis, the feasibility of the laser scanners and the vision sensors are investigated to obtain a docking policy for the docking trajectory and low average docking time.

1.5. Software and Hardware Integration

In this thesis, several hardware and software components are employed to design the control system to achieve the precise docking. The omnidirectional platform, the onboard and the local PCs, the laser scanners, and the cameras are among the physical subsystems whereas the open source C++ libraries such as the OpenCV and the Qt-Creator (Qt 1994) are the software involved in this project. The MATLAB software is employed to plot the obtained results from different sensors. All the interfaces of the Rob@work 3 operate under the Robot Operating System (ROS) as the open source project platform (qdzhaozx 2016).

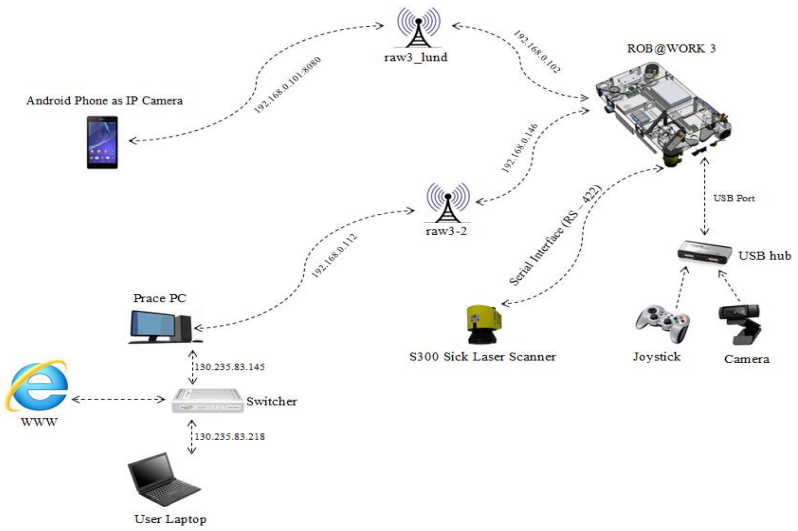


Figure 1 Hardware integration for docking the Rob@work 3.

Figure 1 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 possibly containing surrounding objects. The vision sensors, in this project, are the android phone camera and the USB camera 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 mobile robot can move freely in all directions, either autonomously by the developed controller or by the operator interaction commanded by the joystick.

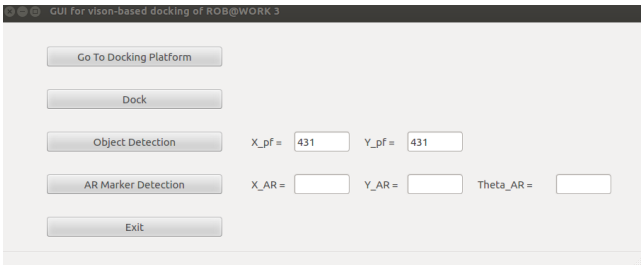


Figure 2 Developed GUI for docking the Rob@work 3.

The first layer of the software is written in the C++ framework as the main program to start visualization with markers to identify the docking platform. The image processing section of the program has two different computer-vision algorithms to detect the docking platform depending on which markers are employed. First, a simple ball is placed in the docking platform and it is tracked by the morphological opening and closing transformation (OpenCV, 2011b), whereas a more advanced marker is detected by an Aruco library in OpenCV on the docking platform (OpenCV, 2015) and the source code is borrowed from (Lier, 2013).

The position of the robot is estimated in an indoor environment and rendered with the 3D vector graphical arrows in the Rviz environment (Coleman, 2016), whereas the vision sensors detect the marker. The visualization with different cameras, the obstacle detection with laser scanners and the vision-feedback control are also implemented in the program to compute the robot commands and send them to the actuators.

Figure 2 depicts the Graphical User Interface (GUI) developed as the core of the platform in the Qt creator software (Qt, 1994) to switch between pose visualization in the 3D visualizer of the ROS framework, called Rviz, as a module in the Open Graphic Library (OpenGL) and the vision-based docking.

1.6. Thesis Outline

Section 2.1 investigates docking with laser scanners mounted diagonally on the mobile platform. 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 used in the project, 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 based on augmented reality marker attached to the docking platform.

Section 2.5 simulates a model free Q-Learning approach for docking. A virtual grid for the 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 for docking mobile robots with laser scanners, vison-based feedback control, and Q-Learning.

Chapter 4 discuss the obtained results and compare different methods used in this project to find the most suitable solution for the docking problem.

Chapter 5 gives a conclusion of the most desired approach for the autonomous docking of the mobile robot employed in this thesis.

Appendix A presents some detailed information for the formulas used in the computer vision (Section 2.2) and the Q-Learning (Section 2.5).

2. Methodology

In this chapter, multiple sensors are investigated to find out how to accomplish the docking. Designing a control system with sensor integration is prioritized since sensor-based feedback control yields more robust and accurate result.

2.1. Autonomous Laser Scanner-based Docking

Knowing the position of the robot in a known or an unknown environment is usually a critical aspect in mobile robotics since the position will be used for further goal investigations. Therefore, higher precision in position estimation makes the goal achievement more accurate as the heart of the navigation system. By means of motion sensors with which the mobile robot is equipped, approximate pose estimation is obtained over time relative to a starting point. However, large position errors make this approach sensitive 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, Martinelli, Tomatis & Siegwart 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, Pallejà, Font, Tresanchez, Moreno & Palacín, 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 cm) and a box (30 × 23.5 × 21 cm^3) have been used as markers and placed on the reference area for analyzing the raw data obtained from the laser scanners (see Figure 3). 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 reference position. The second case, however, represents the robot moving toward 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 embedded in the sensor defined to protect the robot from collision.

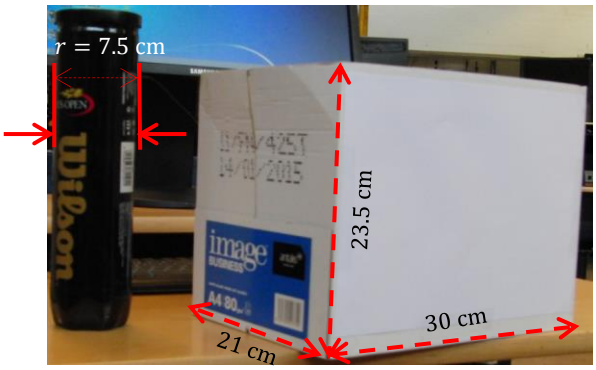
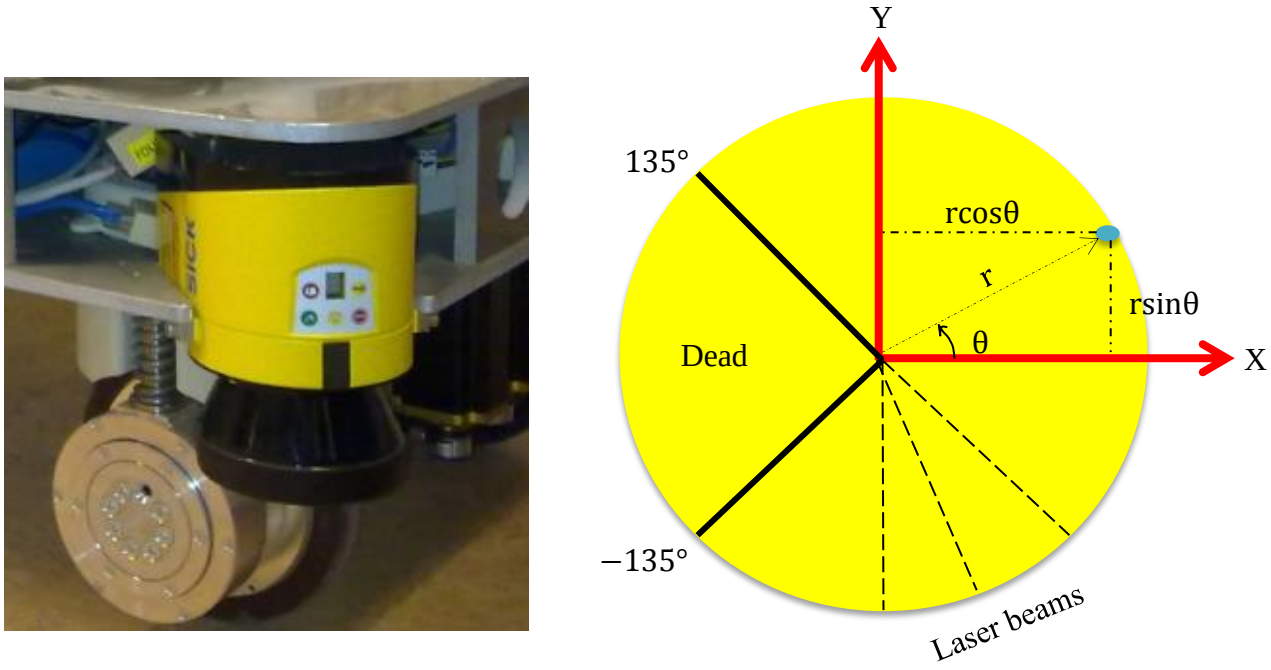


Figure 3 Different markers investigaed for docking with the laser scanner sensor, circular bottle (left), box (right).

The SICK S300, as illustrated in Figure 4 a), is a two-dimensional radial laser scanner operating at 12V and 1A (AG, 2016). It measures 535 distance points and corresponding 534 laser beams within the range of $[-135°, 135°]$, where $\theta = 0°$ points to the 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 the distance to the obstacles and is measured in meter.



a) Front view.

b) Geometrical representation.

Figure 4 S300 Sick laser scanner.

The first approach is to employ the two-dimensional radial laser scanners already mounted on the cross-side of the Rob@work 3 to investigate the precision of the measurements. Figure 4 b) depicts the geometrical representation of the S300 laser scanner to detect the obstacles. The detected obstacle is graphically drawn as the blue point with the radial distance (r) in the plane and the angle θ. The 2D position can be derived as below:

$$(x, y) = (r\cos\theta, r\sin\theta) \tag{1}$$

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 lowest index ($i = 0$) corresponds to angle ($\theta = -135°$). The index, however, needs to be declared as a beam angle (θ_b) in the process for simplification. The following formula converts the index to the actual beam angle of the laser scanner:

$$\theta_{inc} = \frac{\theta_{max} - \theta_{min}}{lb} \tag{2}$$

in which lb refers to the number of the laser beams, which is $lb = 534$.

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

Therefore, the angle increment is roughly 0.505 and the beam angle is approximated as:

$\theta_b = [-135, -134.494, -133.988 \ \ldots \ 133.988, 134.494, 135]$.

Docking of the Rob@work 3 with laser scanners demands a detectable marker attached to the docking platform, acting as a reference point and identified as an obstacle. The laser sensors scan the environment to detect all possible objects on the front side within the angular range of $[-135°, 135°]$. The backside of the laser scanners, known as the 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). Camera is practically a vision sensor mounted on a robot to sense the environment and plan appropriate actions in different conditions. 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 the camera before the visualization process is started and to determine the geometric parameters of the camera.

In computer vison, the geometric camera parameters, including the intrinsic and extrinsic matrices, are calculated for the calibration process (Forsyth & Ponce 2012, pp. 120-123). In this project, the image processing has two arguments for the marker detection, the calibration file and the marker size. The calibration process and the algorithms are summarized in Appendix A.

2.2.1. Geometric Representation and Transformation

An analytical geometry, also known as coordinate geometry, is used to determine the camera correlation with captured objects in a fixed coordinate system in which position and orientation of the robot with its onboard camera are estimated accordingly. The geometric modeling in this project is categorized into three coordinate systems in real time, collaborating in localization and docking. Figure 5 illustrates the marker frame which is attached to the docking platform, the

robot frame (Rob), and the camera frame (CAM) mounted to the front side and center of gravity (C.G.) of the Rob@work 3.

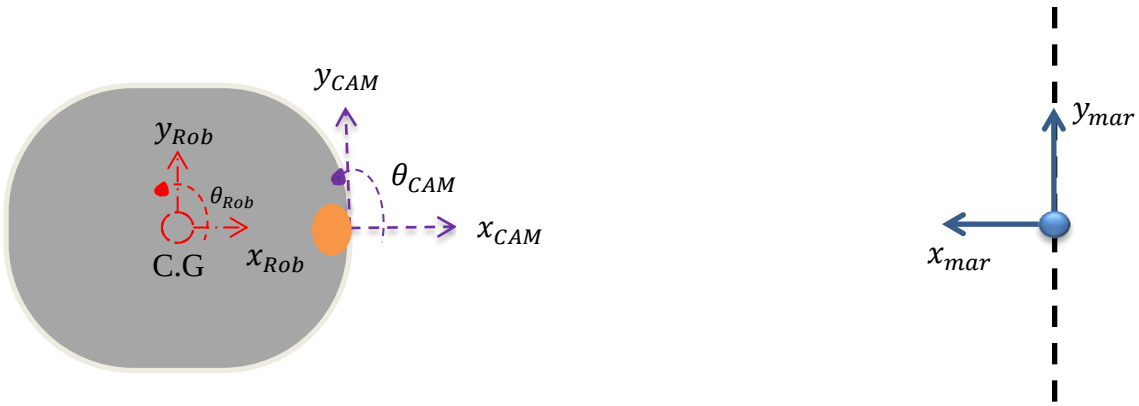


Figure 5 Top view graphical representation of mobile robot, camera, and marker coordinate systems.

According to Figure 5, y_{CAM} , y_{Rob} and y_{mar} are chosen such that they have the same direction to decrease the problem complexity in terms of the coordinate adaptation for docking the Rob@work 3. Such an assumption also simplifies the control signal computation after the pose estimation.

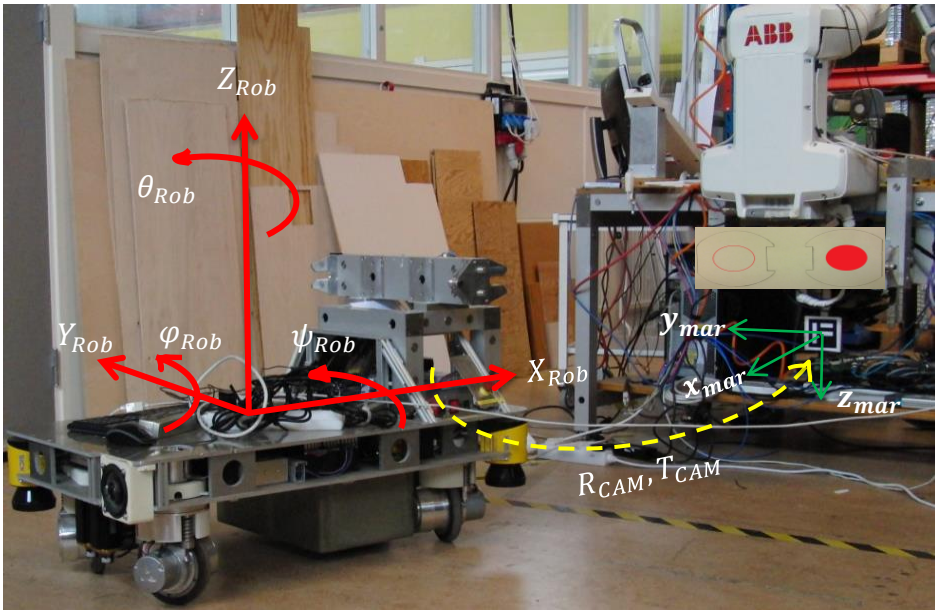


Figure 6 Rob@work 3 and ARTag coordinate systems.

The Rob@work 3, depicted in Figure 6, moves in a six-dimensional coordinate system with 3D position ($x_{Rob}, y_{Rob}, z_{Rob}$) and 3D orientation ($\psi_{Rob}, \varphi_{Rob}, \theta_{Rob}$) known as roll, pitch and yaw of the Rob@work 3. However, since it moves in a plane, movement along the z_{Rob}-axis and orientations along x_{Rob}-axis and y_{Rob}-axis are neglected. Therefore, it has three degrees of freedom in the plane.

The advantage of such coordinate system is that orientation would enable the robot to approach the docking platform from different angles rather than only from the 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 the orientation after extracting the measurements from the sensor. Therefore, it facilitates for the robot to approach the docking platform from more than one configuration.

2.2.2. Pose Estimation

As graphically represented in Figure 5, the camera and the marker frames are distinct. The extrinsic matrix indicates the correlation of the camera in the fixed marker coordinate system, which is fixed to the docking platform (Forsyth et al. 2012, p. 132). The target is assumed to be at the origin of the marker to simply initialize the reference values ($y_{mar} \cong 0$ m and $\theta_{mar} \cong 0°$). Translation vector and rotation matrix are needed to align the axes of the two frames.

In robotics and computer vision, the pose estimation plays an essential role for the navigation and the localization to accomplish the goal configuration without collision and misalignment. The vision sensor and marker are two main components in pose estimation context in which the target is identified by the camera. 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.

Translation and rotation of the camera are calculated with respect to the fixed coordinate system set on the marker attached to the docking platform. Therefore, the 2D position and the 1D orientation of the camera are calculated in the fixed marker coordinate system since the mobile robot moves in a plane. Both 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.

Marker translation is a 3×1 vector $[x_{mar}, y_{mar}, z_{mar}]^T$ in which z_{mar} is eliminated in the plane. Therefore its 2D position (x_{mar}, y_{mar}) is used to define the marker frame translation. On the contrary, a rotation vector of the marker $\vec{r}_{mar}{}^T = [r_x, r_y, r_z]^T$ is converted to the rotation matrix to determine the orientation of the camera with respect to the fixed marker. The rotation matrix is derived according to Rodriguez formula which has a ready-made function in OpenCV (OpenCV 2011a).

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

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

The $R_{3\times3}$ matrix can be simplified as follows:

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

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

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

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

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

Roll and pitch are not applicable since the robot, and the camera accordingly, rotates neither along the x_{Rob}-axis nor the y_{Rob}-axis. Therefore, the yaw angle measured in the fixed marker coordinate system resembles to the orientation of the camera (θ_{CAM}). The configuration is such that positive angle means counter-clockwise (CCW) rotation, whereas clockwise (CW) rotation corresponds to negative orientation. Further, if the robot is placed perpendicularly to the

platform, the orientation is approximately zero ($\theta_{CAM} \cong 0°$). Figure 7 represents six various configurations in which the Rob@work 3 is placed in different angles initially. Control signals along θ_{Rob} and y_{Rob} axes are computed according to the pose estimation with respect to the fixed marker.

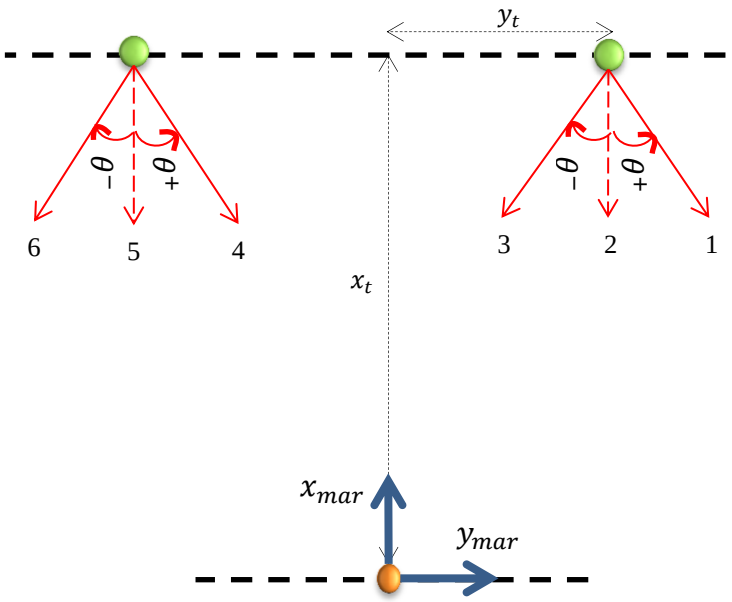


Figure 7 Six different initial robot configurations used for evaluation purpose.

According to Figure 7, the 2D position of the camera in the fixed coordinate system is calculated below (LaValle 2006, pp. 94-97):

$$\begin{cases} x_{mar} = x_t \cos\theta_{CAM} + y_t \sin\theta_{CAM} \\ y_{mar} = -x_t \sin\theta_{CAM} + y_t \cos\theta_{CAM} \end{cases} \tag{10}$$

Equation **10** illustrates the transformation between the camera mounted on the Rob@work 3 and the fixed marker mounted on the docking platform for all configurations. In this equation, x_t and y_t are vertical and horizontal distances respectively to one specific configuration in which the Rob@work 3 is placed perpendicularly toward the docking platform (configuration 2 or 5 in Figure 7).

The measurements for position and orientation are collected in Table 1 to identify the expected behavior of the robot, e.g., control signals in different configurations.

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

Specification / Config.	$\mathbf{y_{mar}}$	$\boldsymbol{\theta}_{\mathbf{mar}}$	Expected $\dot{\mathbf{y}}_{\mathbf{rob}}$	Expected $\dot{\boldsymbol{\theta}}_{\mathbf{rob}}$
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 were published with respect to the mobile robot with the ROS built-in package called transform (tf), 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 3D position of the camera is published with respect to the fixed marker, it was subscribed in a callback function to begin the motion control process to move the robot toward the docking platform.

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).

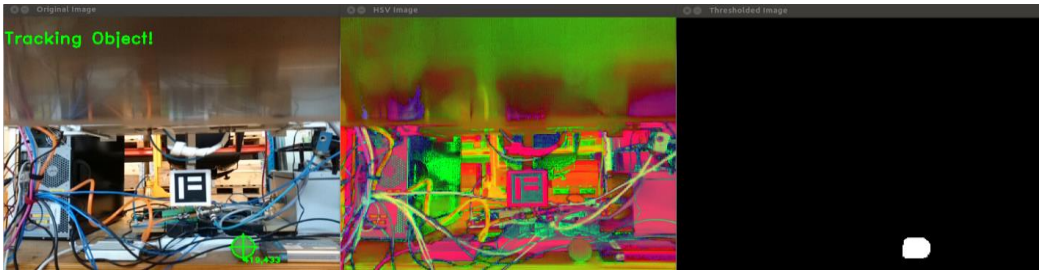
Fiducial marker systems have some advantages to be considered as the target of docking the Rob@work 3 in this project. First, the identification is practically easy since such a black and white square sketch with embedded internal pattern facilitates the marker detection. Second, it

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 2D positions (x_{CAM}, y_{CAM}), whereas the 3D coordinates are used to indicate positions and orientation ($x_{CAM}, y_{CAM}, \theta_{CAM}$) with the planar fiducial.

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 clearly defined 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) HSV image. c) Threshold image.

Figure 8 Point fiducial detetction, tracking the green ball in the docking platform.

A video stream with the resolution of 640×480 pixels is an input matrix for the tracking algorithm to start the image processing for detecting the point fiducial. Figure 8 a) depicts the detected fiducial with OpenCV in the camera coordinate system, in which a green ball attached to the docking platform to identify the target. Figure 8 c) illustrates the threshold window and the

detected point fiducial after the morphological transformation is performed to eliminate the noise (OpenCV 2011b).

Figure 8 b) shows the docking area in the HSV color space with a green ball. The input matrix has a RGB (Red, Green and Blue) color space which contains integer values ranging from 0 to 255. For color based segmentation, the RGB format is transformed by a cylindrical coordinate called HSV (Hue, Saturation, and Value) 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, saturation is the amount of color that is mixed with white, and value is the mixture with black (Forsyth et al. 2012, pp. 68-69).

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

Figure 9 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 OpenCV library (OpenCV 2011c). The developed track bar GUI adjusts the HSV values to detect the green ball as the point fiducial when docking the Rob@work 3.

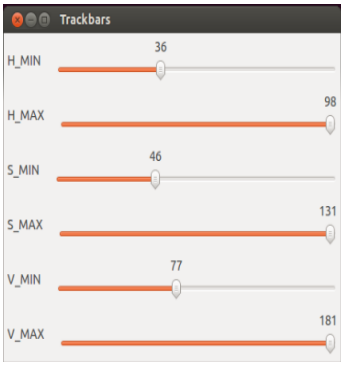


Figure 10 HSV track bar GUI for color adjustment to detect green ball as the point fiducial.

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 (AR) field has widened planar fiducial applications in computer vision since it consists of unique patterns compatible with the Aruco marker detection algorithm. The square-bordered structure with black and white background facilitates the recognition of the unique internal pattern with the definite shape and ID called ARTag (OpenCV 2015).

ARTag uses digital coding theory to obtain low false positive rate and inter-marker confusion rate with small required marker size. The low false positive rate is a probability of recognizing markers falsely, whereas the inter-marker confusion rate is the probability of obtaining wrong marker ID (Fiala 2005, pp. 590-591).

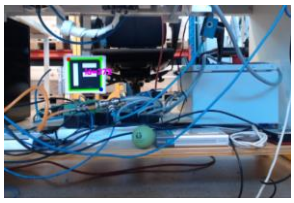
Table 2 represents a few standard AR markers compatible to detect the Aruco marker given specific IDs. The algorithm identifies the markers with the Aruco library in OpenCV.

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

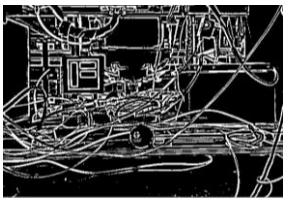
ID # 1	ID # 48	ID # 61	ID # 137
			
ID # 272	ID # 349	ID # 459	ID # 514
			
ID # 606	ID # 746	ID # 899	ID # 1023
			

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 encodes 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 the marker is 80 mm with 8 mm of paddle as a thickness of the white border. The generated marker is mounted at the center of the docking platform and between end-effectors. Figure 11 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 11 Detected Aruco marker on the docking platform.

The pose estimation of the camera with respect to the fixed marker coordinate system is calculated with the relation of equation 10. 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 Chapter 3 and control signals are computed in each sample.

2.4. Autonomous Vision-based Docking

2.4.1. Kinematic Model

For robot platforms such as the Rob@work 3, formulating an exact mathematical model with wheel torques as inputs to start the control process is challenging and mostly likely undesirable. Therefore, a simplified model with only the essential information is preferable. In this thesis, only the kinematic model of the system is deployed since the expected velocities and accelerations in the docking are low. Some assumptions are taken into account at simplifying the system under investigation (Nilsson 2010, p. 18):

- Wheels have solid rubber structure and are non-deformable
- The contact between the wheel and the ground is approximately at one point
- The load is equally distributed on all the wheels
- Robot moves in a flat horizontal 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 mobile platform 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 accuators. In this coordinate system, positive x-axis points along the forward direction, whereas positive y-axis indicates the wheels pointing to the left direction if it is seen from the above along the x-axis.

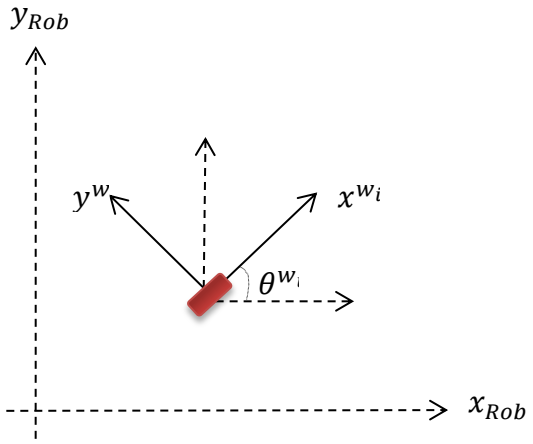


Figure 12 Top view of the rotated wheel *i* by θ^{w_i} angle in the robot coordinate system.

Each wheel also has the local coordinate system (x^{w_i}, y^{w_i}) which are parallel to the robot coordinates. Figure 12 illustrates a wheel rotated by θ^{w_i}. This angle is the orientation angle and each wheel has its own orientation. It 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 center of gravity in which the entire platform rotates with θ^{w} degrees if the wheels rotate θ^{w} degrees.

2.4.1. Control Design

A mobile robot equipped with vision sensors has the potential to yield accurate and non-invasive robot manipulation to achieve the goal since a non-contact evaluation of the environment is enabled by vision sensors while the robot moves toward the target (Lepetit et al. 2005, p. 4). The employed control approach theoretically computes precise control signals if the input measurements have enough accuracy.

For instance, developing an open-loop control technique for the visual manipulation, which is practically similar to the "look first, move next" fashion, would not have the desired outcome unless the high accuracy of the visual sensor and robot end-effectors are guaranteed in advance which makes this policy economically impractical and the vision system is only employed in the pose estimation to obtain the required motion commands to achieve the docking goal (Hutchinson et al. 1996, p. 651).

In contrast, there is another approach in which visual information in the feedback loop gets involved to increase the overall accuracy of the control design. 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 current position of the camera and the robot in real time.

In fact, this alternative approach evaluates the current position of the vision sensors and the target online to compute appropriate velocity commands to manipulate the robot to achieve the docking task. The closed loop vision-based control, also known as Visual Servoing (VS) is carried out by the visual information in separate images (Hutchinson et al. 1996, pp. 653-654). The VS practically involves a camera as the vision sensor and computer vision algorithms to detect the employed fiducial markers on the docking platform and control the position of Rob@work 3 (Hutchinson et al. 1996, pp. 653-654).

In this project, the docking platform is identified by the fiducial markers. The visual system provides position and orientation references for the control loop. Therefore, Position-Based Visual Servoing (PBVS) is employed to serve as the feedback. Features of the image are extracted in the control loop and the difference between the current positions of the mounted camera on the Rob@work 3 and the fiducial markers on the target is computed. The PBVS estimates the approximate position and orientation of the camera with respect to the fixed marker as the target and sends the control signals to the mobile 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, the more appropriate the control signals are. Therefore, the target is graphically sketched as a circle with an origin of the 2D marker position, identified as $(x_{ref}^{mar}, y_{ref}^{mar})$, and $x_{thresh} = 1$ mm which is a radius of threshold along the x_{mar}-axis.

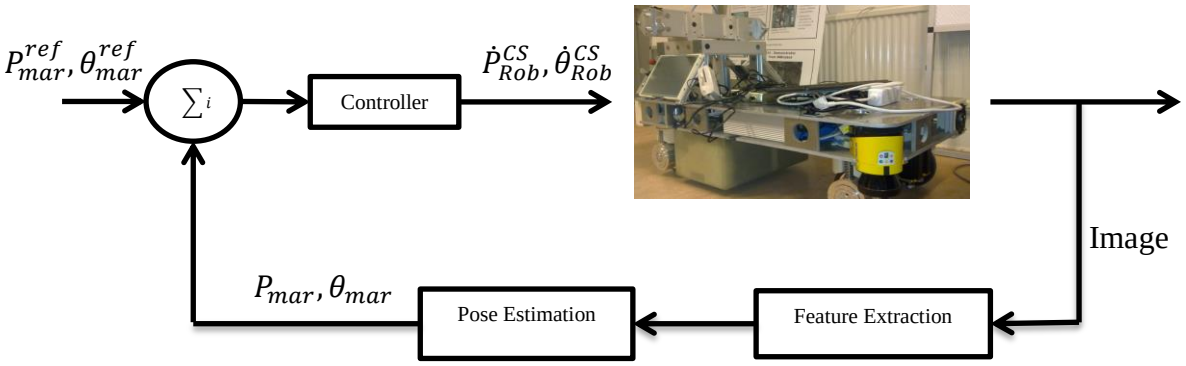


Figure 13 Position-Based Visual Servoing (PBVS).

The most important part of the control system, however, is the plant. Theoretically, the plant is the combination of the process and actuators and is expressed as a transfer function to declare the relation between the input and the 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 Rob@work 3 has an embedded control system which sends the appropriate steering and driving velocity commands to the wheels within a

sampling period of approximately 20 milliseconds, depending on the command generator and computational load (Nilsson 2010, p. 15).

Figure 13 represents the vision-based closed loop control design employed to computer the control signals for the Rob@work 3. The desired 2D positions ($P_{mar}^{ref}(\ x_{mar}^{ref}, y_{mar}^{ref})$) and the orientation ($\theta_{mar}^{ref}$) 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 $P_{mar}(x_{mar}, y_{mar})$ and orientation θ_{mar} of the robot after the feature extraction of the markers to calculate the error signal 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 14 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.

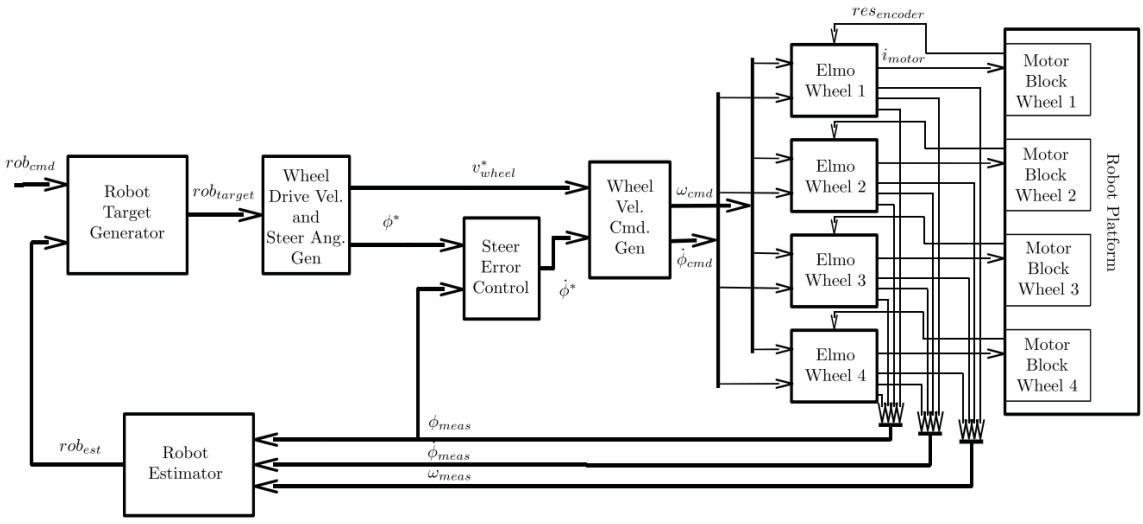


Figure 14 The under-carriage control of the Rob@work 3 (Nilsson 2010, p. 15).

In this project, the docking area is divided into two 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.

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 velocity commands for the actuators. According to Figure 15, adjusting a movement along the y_{Rob}-axis and orientation along the θ_{Rob}-axis is more crucial than movement along the x_{Rob}-axis, since the mobile 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.

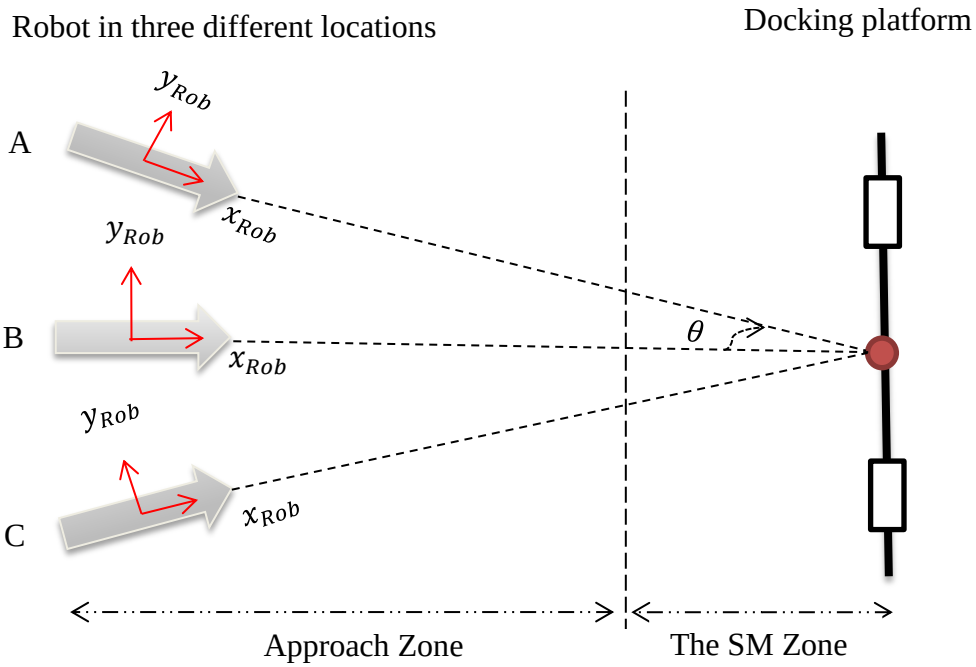


Figure 15 Top view schematic of the docking area with Approach zone and SM zone.

The control system is designed such that the robot always moves toward the docking platform while it adjusts the positions and the orientation ($x_{mar}, y_{mar}, \theta_{mar}$). Furthermore, the movement 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 coordinate system. Therefore, the thresholds values ($y_{thresh} = 2.5$ 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 critical in docking of the mobile robots and a certain time is defined as the desired docking time. The vision-based feedback control is designed such that the target is accomplished in desired average docking time which is roughly 35 seconds. Gain tuning is taken into account to achieve this goal. The time is initially recorded on the embedded controller of the Rob@work 3 via the ROS client library package, with an appropriate time application programming interface (API) code in the data acquisition process (Gomes 2013).

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 deal with the control signal calculations. 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 samples of the ROS time (t_s^{ROS}). Each step is mathematically derived as the difference between two consecutive samples. The vector of real times, on the contrary, is calculated as follows and utilized for further computations and control design:

$$\Delta t = t_f^{ROS} - t_i^{ROS} \tag{11}$$

$$t_{real} = 0 : \frac{\Delta t}{t_s^{ROS}} : \Delta t \tag{12}$$

The error is the difference between the target (P_{ref}) and the current position of the camera P_{cur} in the fixed marker coordinate system calculated below:

$$e(t) = P_{ref} - P_{cur} \tag{13}$$

The control signal is defined as (Ogata 2010, p. 84):

$$rob_{cmd} = u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \quad (14)$$

In (14), 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 C++ 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 PID feedback control pseudo code.

In Algorithm 1, the parameter dt is the sampling period of the designed controller, which is set to 10 milliseconds to make sure the controller is running in parallel with other nodes in ROS to avoid delays in the whole system. The designed controller is implemented without an anti-windup inside the subscriber to the published nodes of the fiducial marker since the sensor

measurements do not sense large changes in the set points. Therefore, the integral term does not accumulate significant error during the rise.

2.5. Reinforcement Learning (RL)

2.5.1. Background

Interacting with the environment is most likely the first and simplest step in the learning process. In the reinforcement learning (RL), despite supervised learning (Duda, Richard O.; Hart, Peter E.; Stork, David G.;, 2000), there is no physical teacher who gives lessons about how to obtain the desired behavior and the 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 by 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 a goal-directed learning method 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 reward the agent meaning that it reinforces helpful behavior that should be persistent, whereas wrong actions yield more negative rewards to inhibit the non-helpful behavior of the robot (Gaskett 2002, pp. 2-3).

There are few concepts in the RL framework. For instance, an intelligent agent (IA) 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).

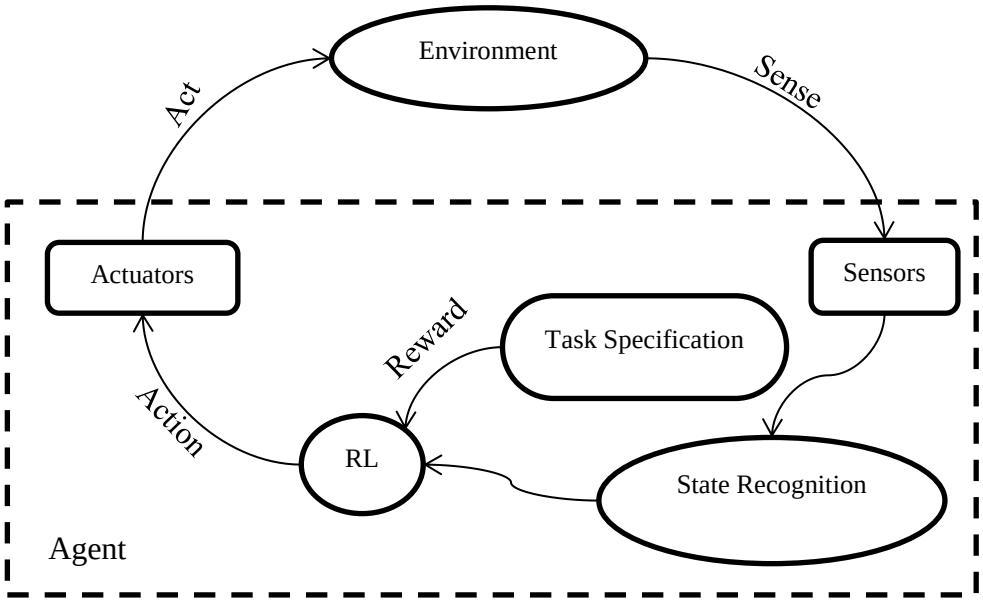


Figure 16 Reinforcement Learning setups (Sutton et al. 1998, p. 44).

According to Figure 16, the robot acts on the environment with a 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 with four omnidirectional wheels.

Among the three fundamental approaches dealing with the RL problems, the Temporal Difference (TD) method is employed in this thesis since it does not need a model of the environment and updates the estimates based on other learned estimates without waiting for 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 using the values of the optimal policy learned independently from the actions (Jun, Lilienthal, Martinez-Marin & Duckett 2006, p. 2656). On the contrary, the on-line policy takes the slower path to the optimal policy since it wants to explore both optimal and non-optimal actions. Therefore, it is more careful about the environment. The target in this thesis, in the Q-Learning method, is the state identified as the docked robot and the actions are influences of the

robot on the environment while it moves to obtain the 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 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 representing the matrix, known as the 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 similar to 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 2.4.1), 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 for 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, to achieve a smooth docking in which the 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 an 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.

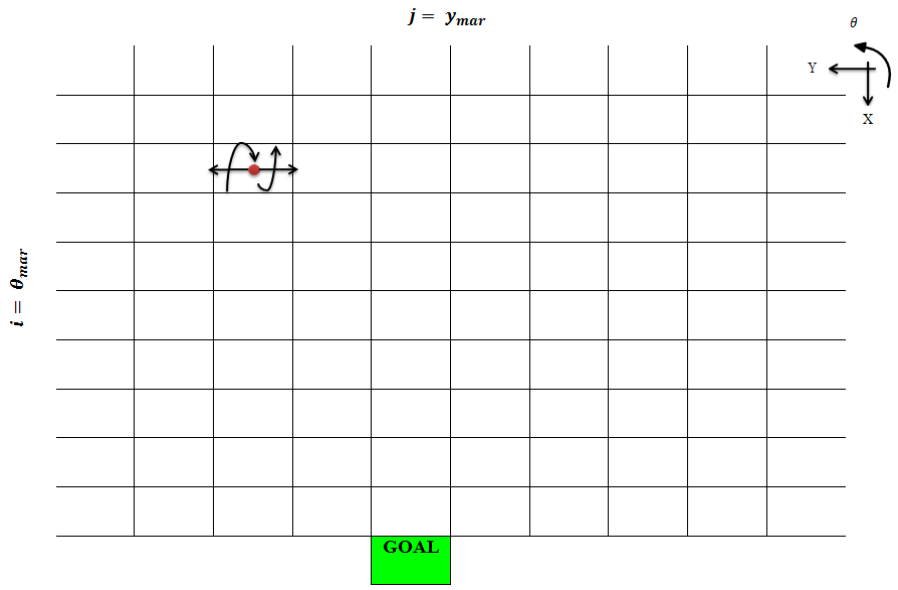


Figure 17 Discrete grid for the Q-Learning action selection policy.

Figure 17 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 its indexes for θ_{mar}^{Ref} and y_{mar}^{Ref} are saved once the robot was manually docked before the training. The 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, as mentioned earlier, are linear velocity along the y_{Rob}-axis and angular velocity along the θ_{Rob}-axis. In other words, the robot moves left and right if the linear velocity ($\dot{y}_{Rob}$) is applied, whereas it rotates CW or CCW if the angular velocity, ($\dot{\theta}_{Rob}$) 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:

$$Q(s_t, a_t) = R(s_t, a_t) + \gamma * \max[Q(s+1, \sum a_i)] \quad (15)$$

Equation (15) states that to update the specific component of the Q-matrix, the reward of that specific state and the maximum value of the next state considering all possible actions should be declared. The mathematical representation of (15) as a new sample estimate is expressed as:

$$\text{samaple} = R(s_t, a_t) + \gamma * \max[Q(s_{t+1}, a)] \quad (16)$$

The new estimate of the Q-value in the Q-matrix is computed with an average function as:

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha(\text{sample}) \quad (17)$$

in which $0 \leq \gamma < 1$: Discount factor $\Longrightarrow$ $\gamma \cong 0$: immediate reward & $\gamma \cong 1$: long-term high reward and $0 \leq \alpha < 1$: Learning rate $\Longrightarrow$ $\alpha \cong 0$: No learning & $\alpha \cong 1$: prior values neglected.

Generally, the learning is started with a large learning rate 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. A constant learning rate of $\alpha = 0.1$ is chosen for the learning process.

The Q-learning pseudo-code algorithm is represented as follows:

- Set parameters (γ, α) and environment reward (R).
- Initialize the Q-matrix.
- For each episode:
    - Begin from a random initial state
    - While (the goal state has not been reached)

    {

        Choose an action randomly.

        Move to a new state with the chosen action.

        Consider a new sample estimate (16).

        Update Q-values with prior knowledge of old Q-values with (17).

        Next state = Current state.

    }

    - End While.
- End for.

Algorithm 2 Pseudeo-code of model-free Q-Learning (Sutton et al.1998, p. 145).

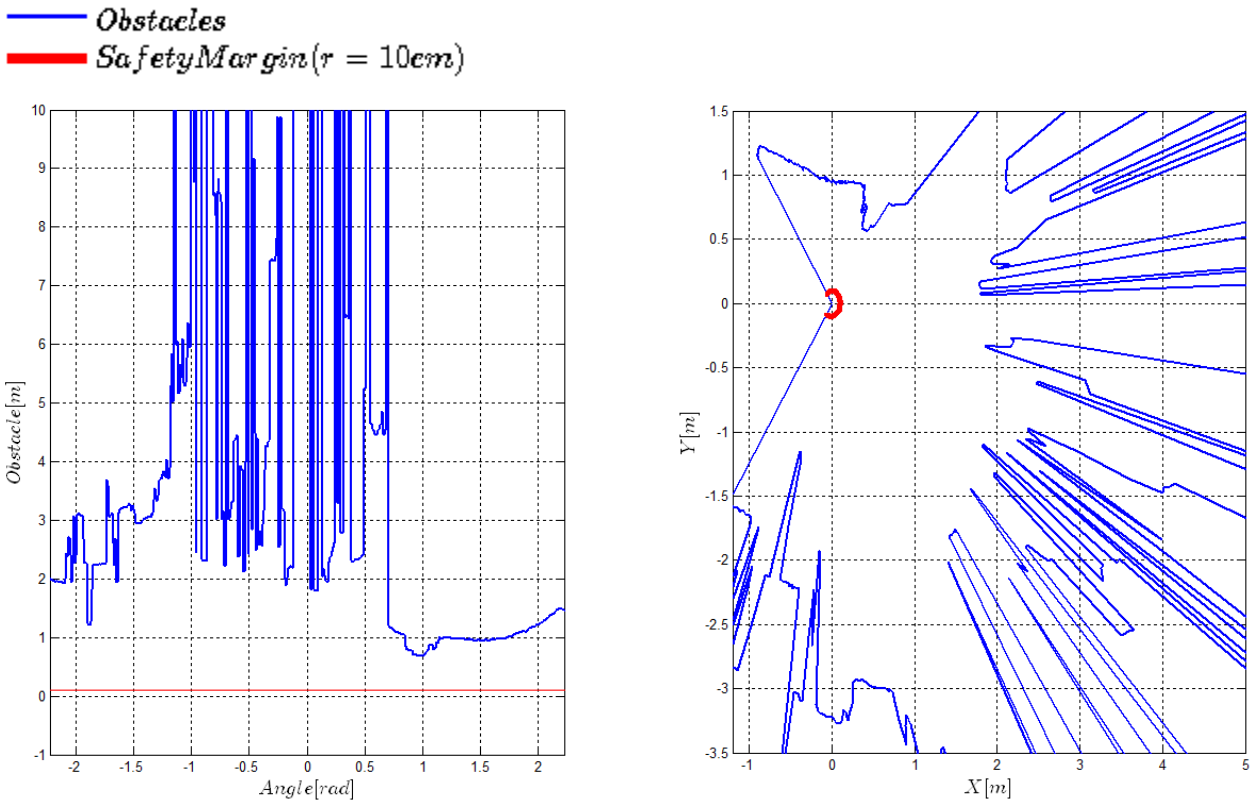
3. Results

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

3.1. Laser Scanner Sensor

In this section, several experiments for docking the Rob@work 3 using the laser scanners are presented to evaluate the precision of the results. For each experiment, a plot of distance to the 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 case 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 18 Laser sensor analysis.

Figure 18 a) illustrates the approximate distance to the surrounding objects recorded on the front laser scanner in different beam angles within the range of $[-2.35, 2.35]$ radians, which is equivalent to $[-135°, 135°]$ degrees. The angle is calculated from the index (i) and the angle increment (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 18 b).

According to Figure 18 a), any object with distance less than or equal to 10 cm from the laser sensor is considered as an obstacle and terminates the program. It can also be seen that areas closer to minimum and maximum indexes 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 18. 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 19. The first scenario, Figure 19 a) investigates the detection accuracy of the laser scanner when no marker exists, whereas in the second scenario, Figure 19 b), a transparent cylindrical bottle was placed on the docking platform and used as a detectable marker.

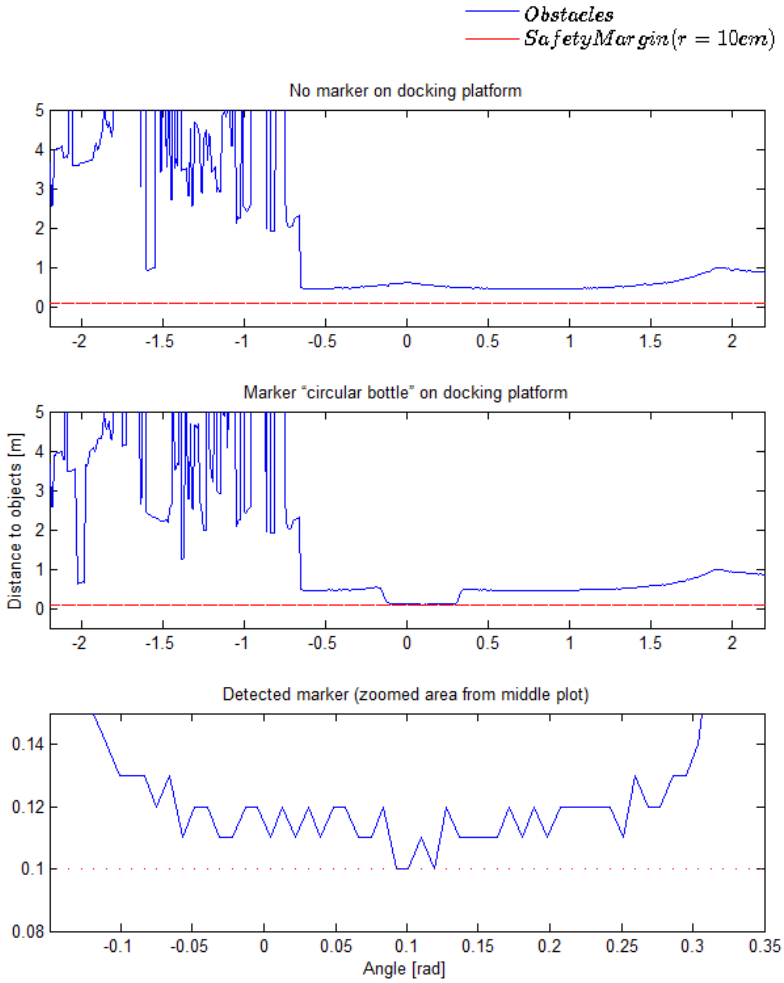


Figure 19 Laser scanner analysis for the detected marker while the Rob@work 3 is docked.

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 was 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 19.

Theoretically, the laser scanner records data with 1-2 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 19 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 illustrate the percievement of the 2D position of the surrounding objects. Figure 20 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 crossed, otherwise, the experiment is terminated.

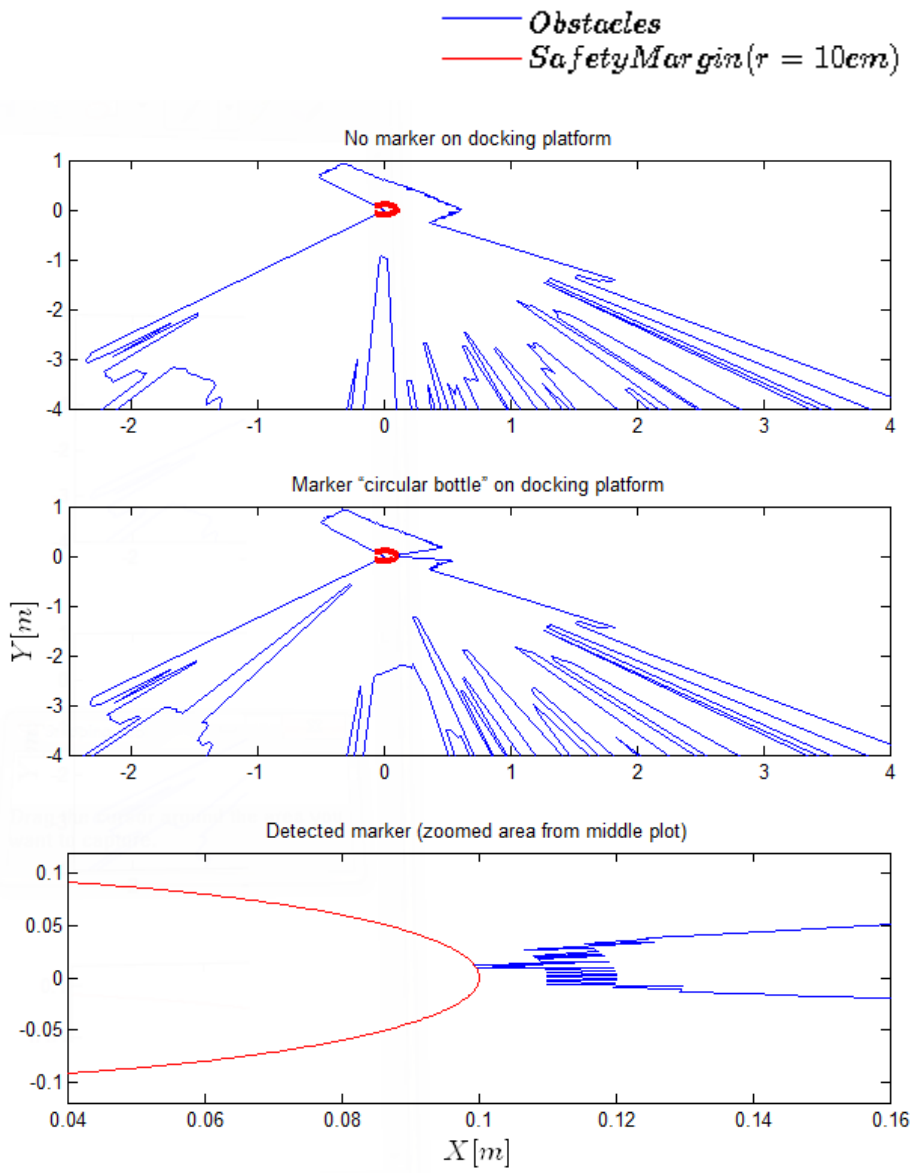


Figure 20 Laser scanner analysis for detected marker while the Rob@work 3 is docked, top view.

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 successfully detected by the sensor.

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 21 illustrates scanner analysis for this configuration.

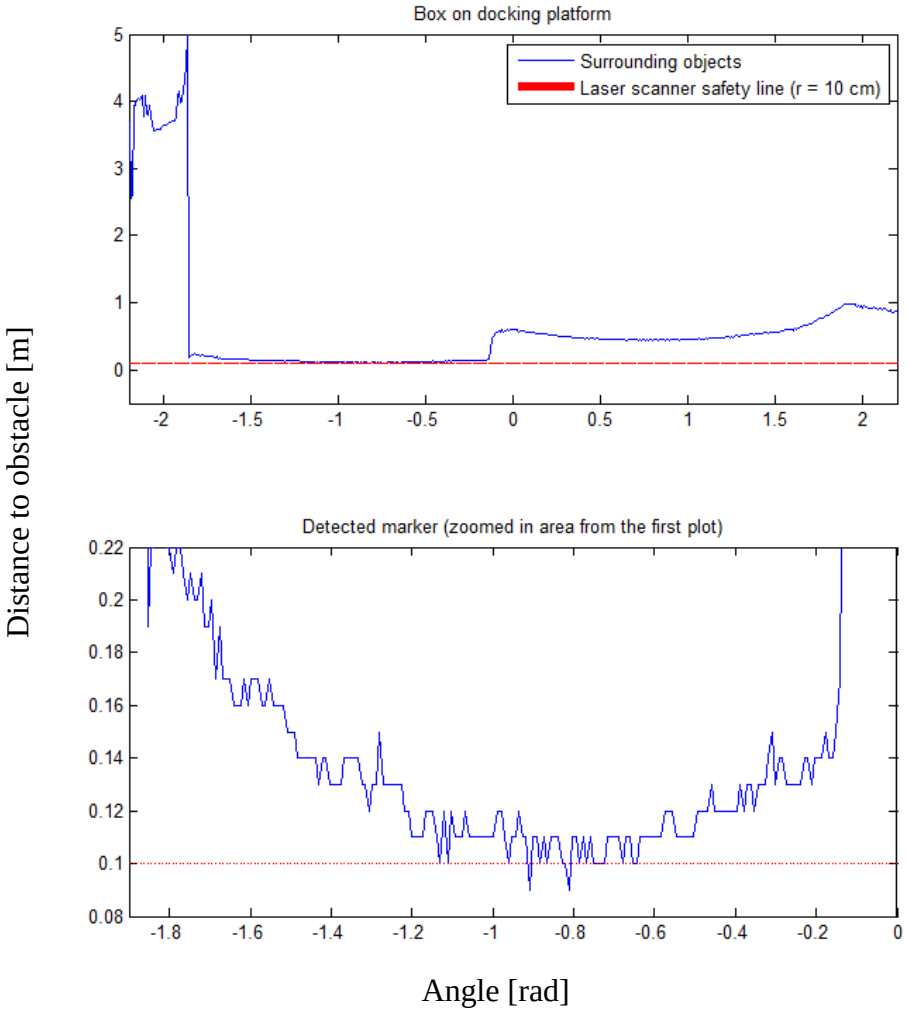
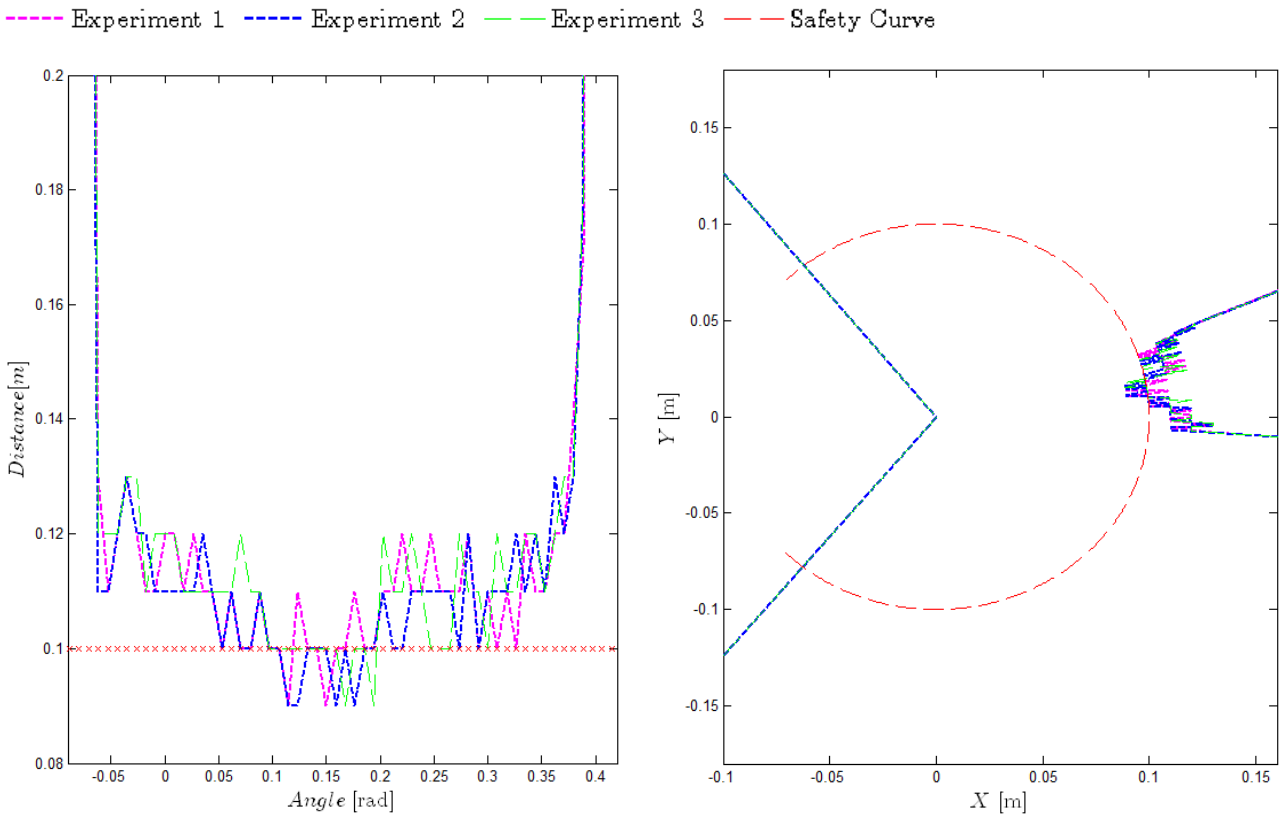


Figure 21 Scanner analysis with box marker on the docking platform.

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 21 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 22.



a) Detected marker in different angles.

b) Detected marker, top view.

Figure 22 Three conducted experiments for the same configuration.

According to Figure 22, 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

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 benefit 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 (tf) broadcasts a frame in real-time with respect to its parent. After the coordinate insertion, the 2D rotation matrix 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 coordinate system. Figure 23 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.

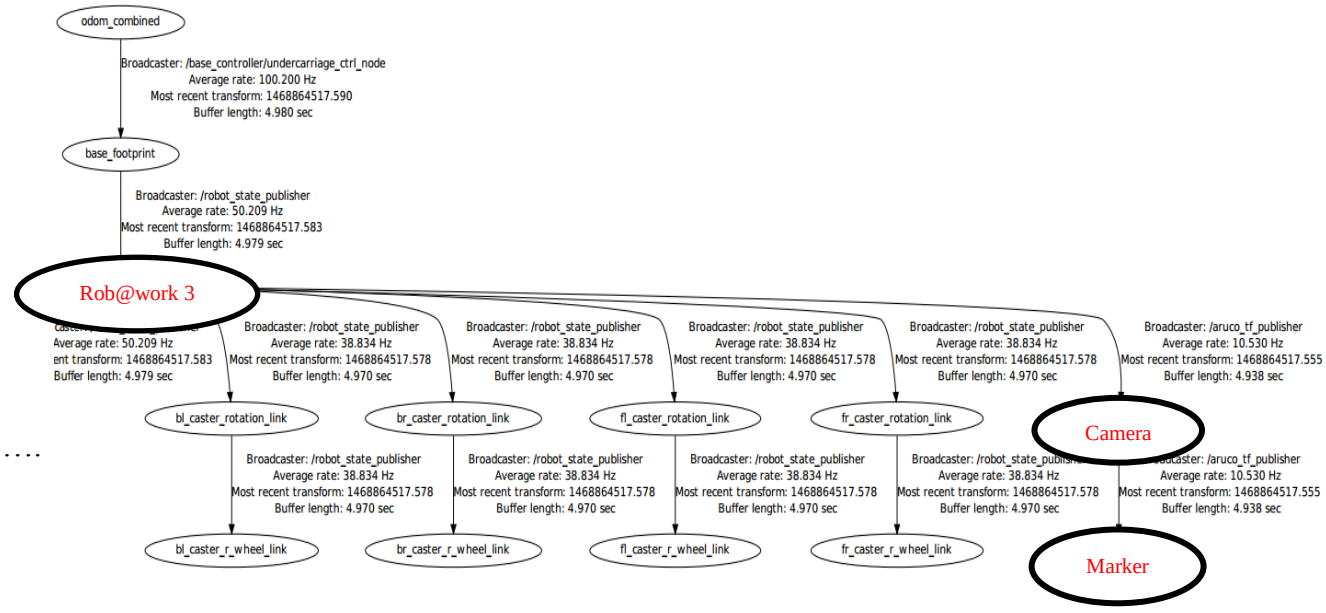


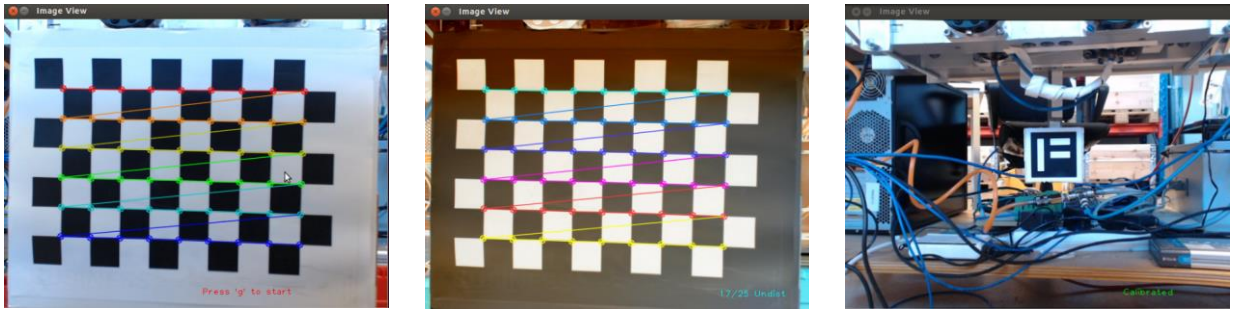
Figure 23 Coordinate frames of the camera and the marker added to the Rob@work 3.

According to Figure 23, the publisher for the added frames is the Aruco module based on the ArUco library in 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 2D points in the image and the 3D 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 24 illustrates the calibration process of the camera with the chessboard pattern.



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

Figure 24 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
Frames per second	25
Tangential distortion	False
Principal point $C_0(u_0, v_0)$	Fixed principal point 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 YAML 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 25 depicts the comparison between the recorded ROS time and the real time of the USB and IP cameras.

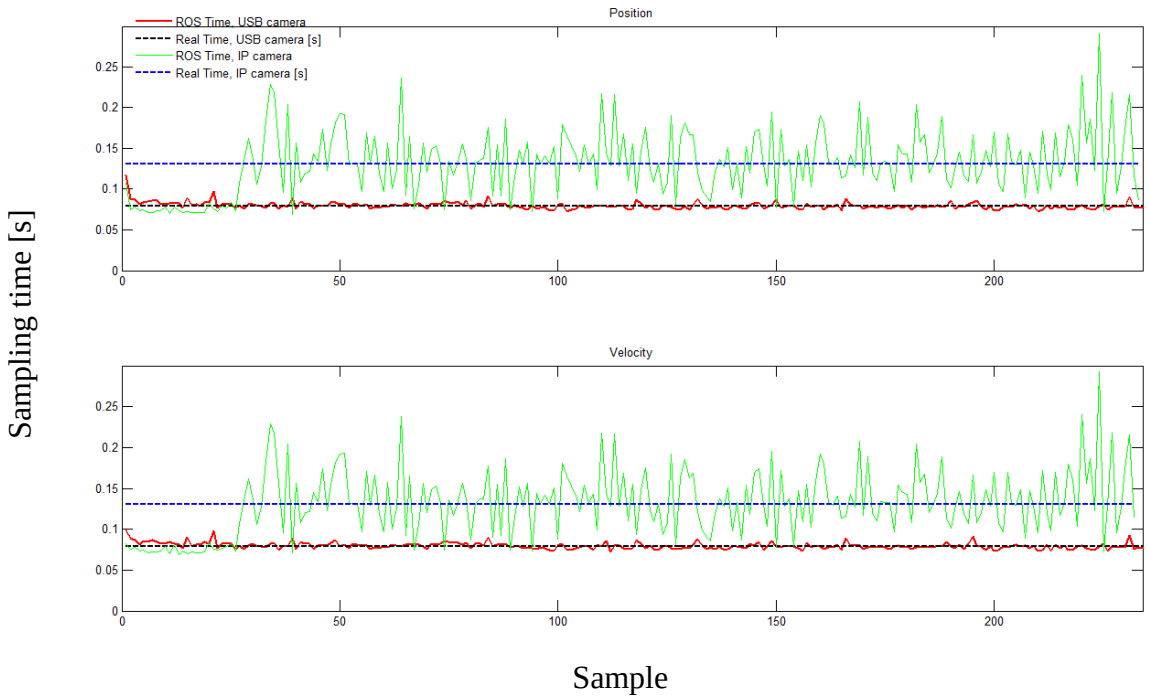


Figure 25 Sampling time comparison of different vision sensors.

The ROS time, depicted in Figure 25, 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 26 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.

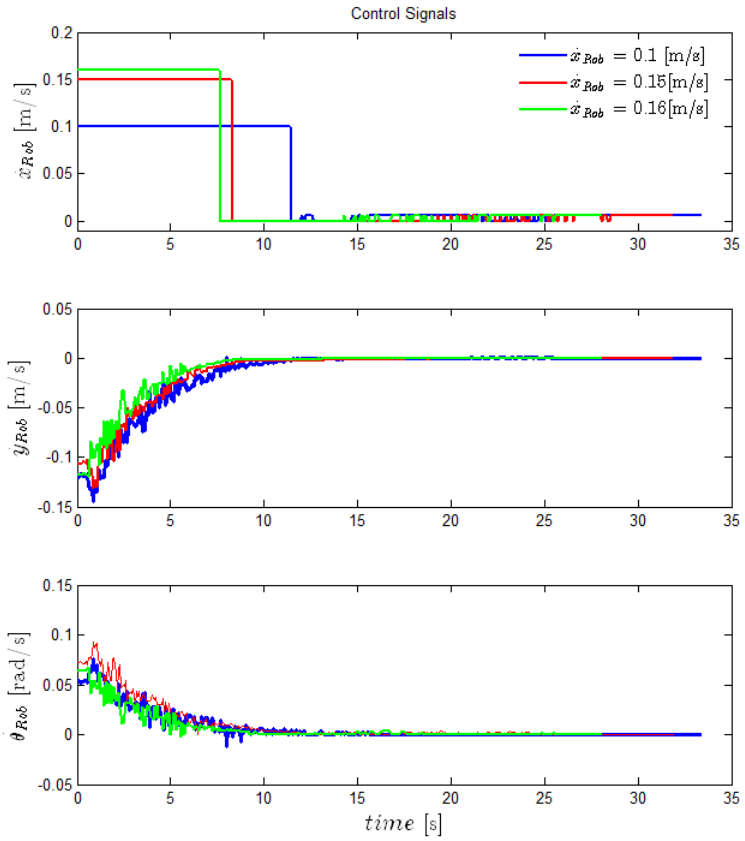


Figure 26 Control signals along the axes of the Rob@work 3.

The corresponding position trajectories of the above experiments are plotted in Figure 27, which contains the whole docking area, the SM zone, and the target which is sketched as the circle with the origin at the reference value of the marker ($x_o = -0.2025$ m, $y_o = -0.019$ m) and the radius of the threshold along the x_{mar}-axis ($x_{thresh} = 1$ mm).

According to Figure 26, it takes more time for the Rob@work 3 to approach the SM zone (roughly 11 seconds) if lower forward velocity ($\dot{x}_{Rob}$) is applied. In this case, the approach time is increased and the docking would be slower. However, Figure 27 shows less offset with the reference along the y_{mar}-axis which is approximately 1 cm with slower motion along the x_{mar}-axis.

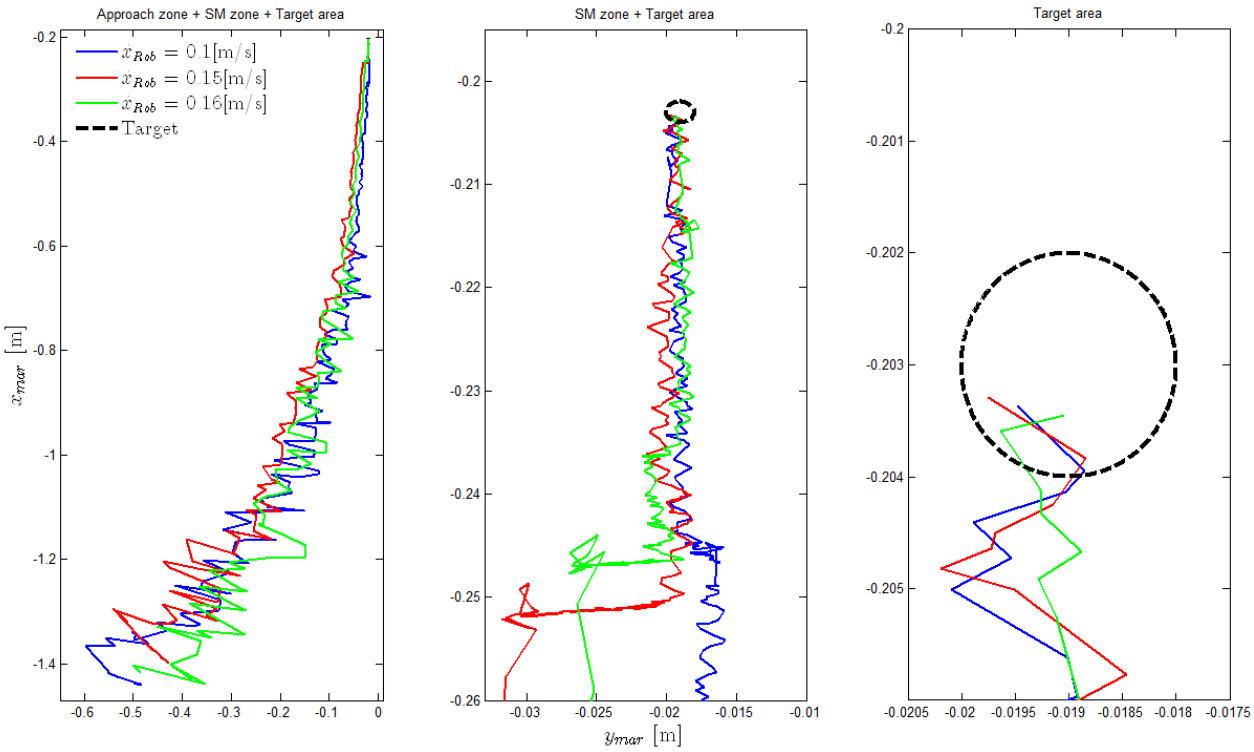


Figure 27 Docking trajectories.

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 the controller.

	Exp. / Gains	1	2	3	4
Position (y_{rob}-axis)	K_P	0.266	0.28	0.29	0.3
	K_I	0.0003	0	0.00026	0.0004
	K_D	0.014	0.006	0.015	0.02
Orientation (θ_{rob}-axis)	K_P	0.25	0.25	0.25	0.25
	K_I	0	0	0	0
	K_D	0	0	0	0

In the control design, camera pose estimation with respect to the fixed marker provides the measurement errors and computes the control signals along the y_{mar}-axis and θ_{mar}-axis to accomplish docking. Four PID controllers with different gains were implemented to evaluate desired docking behavior of the Rob@work 3 with respect to time and trajectory.

Figure 28 presents the camera pose estimation for different gains recorded in the fixed marker coordinate system. The x-axis of the plots is the real time converted from the ROS time using (11) and (12) even though the real experiments were conducted with the ROS time.

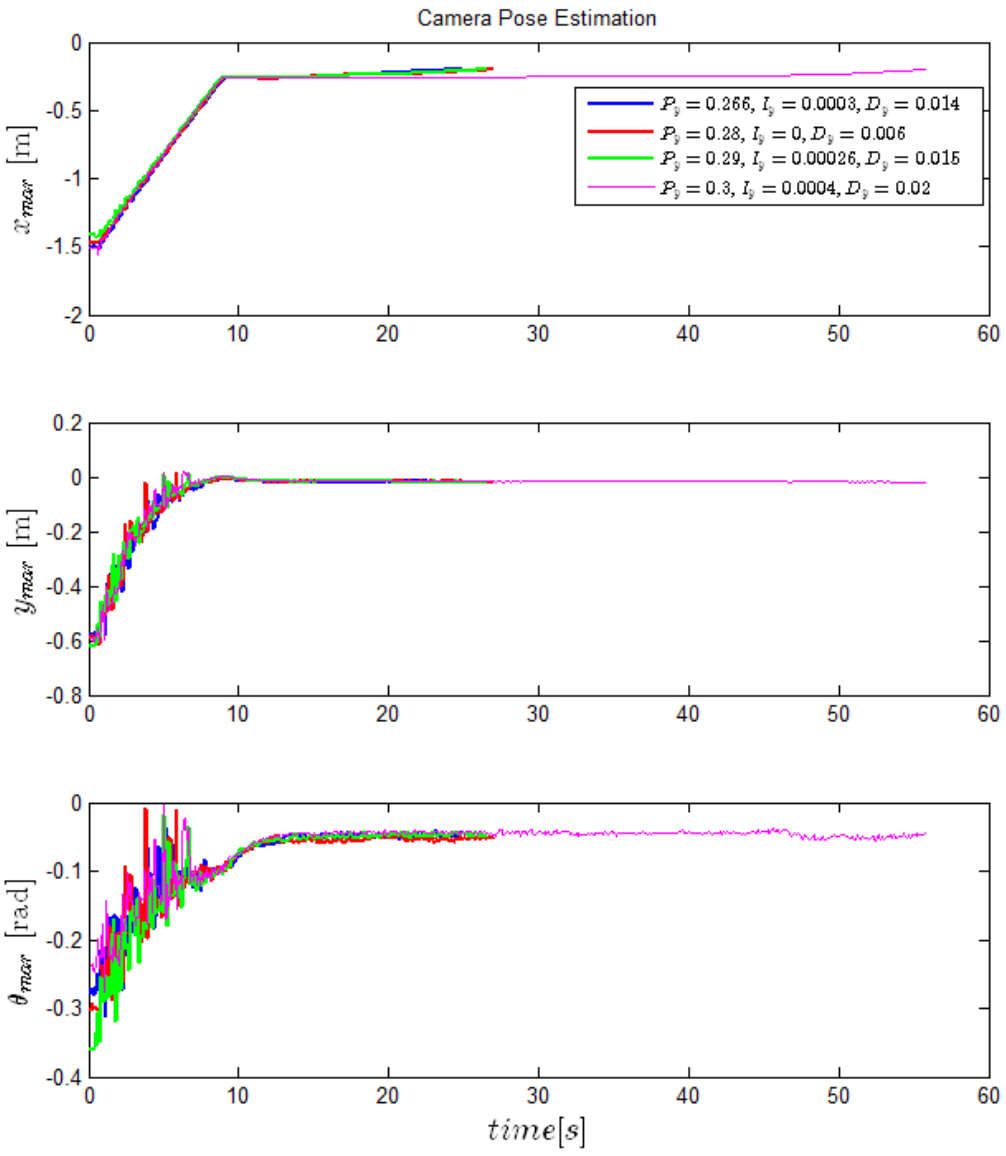


Figure 28 Camera pose estimation in the marker coordinate system.

Theoretically, the slope of the position versus time plot indicates the velocity of the mobile robot. According to Figure 28, 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 29 ($\dot{x}_{Rob} = 0.15\,\frac{m}{s}$).

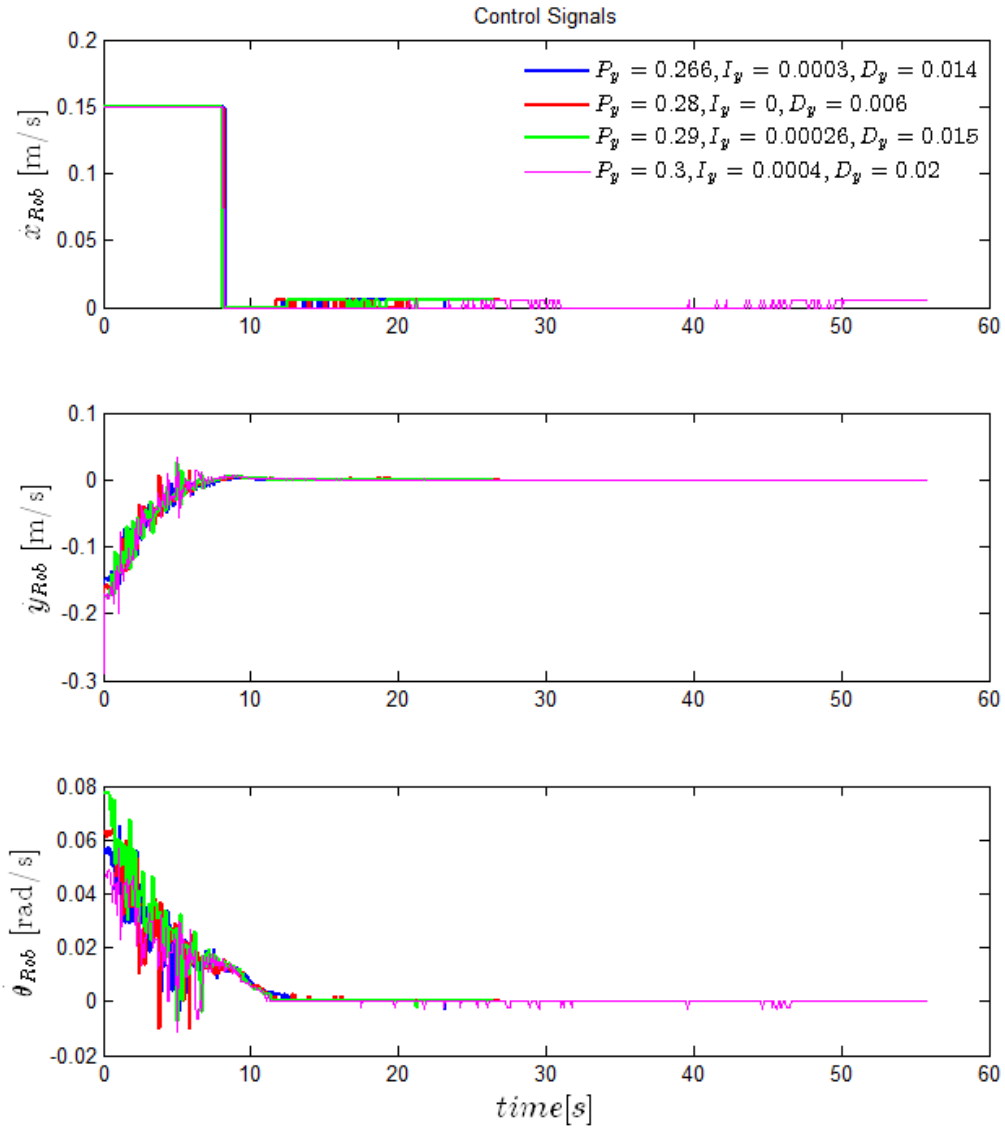


Figure 29 Control signals along the axes of the Rob@work 3.

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 in Figure 29. In each case, the average docking time is evaluated with different PID gains. According to Figure 29, the slowest controller takes approximately 55 seconds to accomplish the docking with 8 seconds for transferring to the SM zone, which is quite fast and reasonable according to limited space available in the lab for the 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.

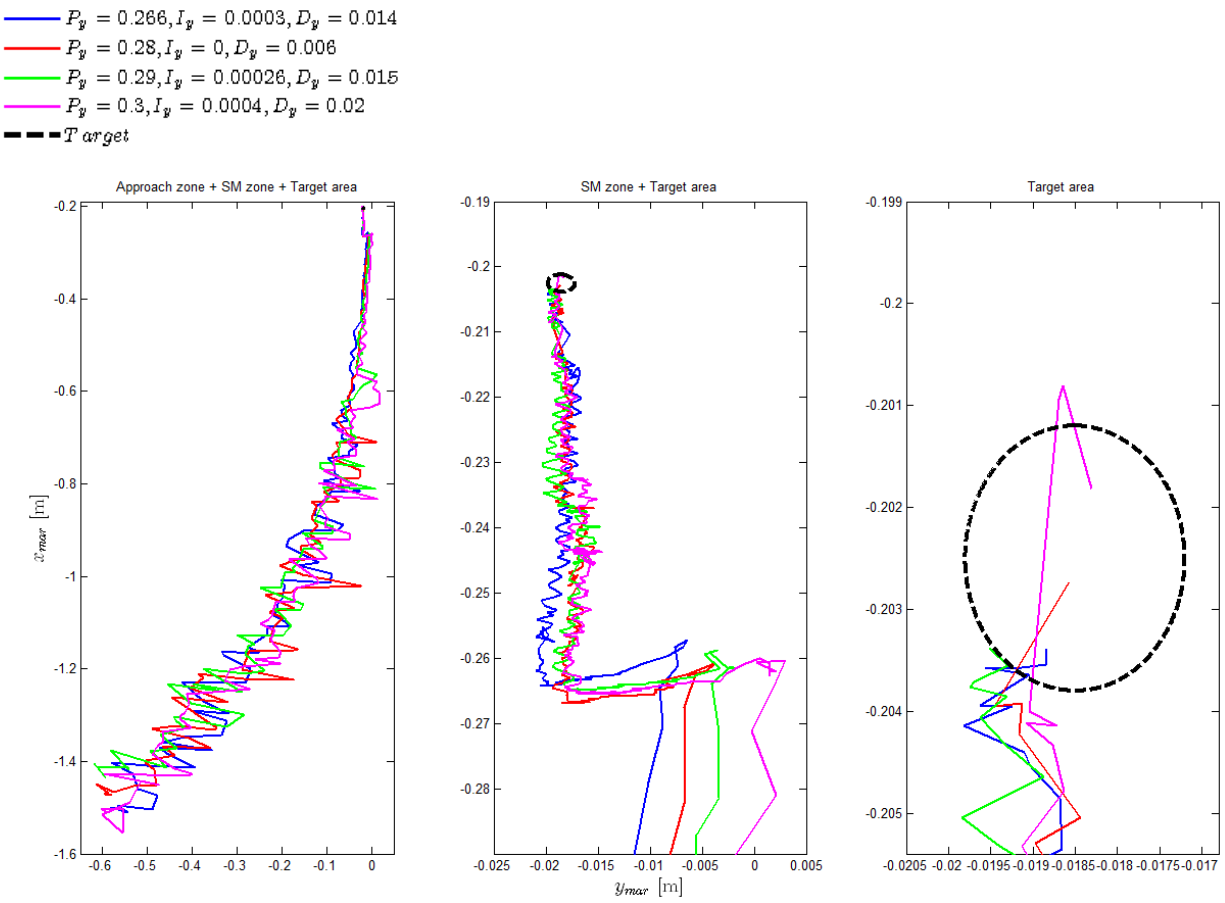


Figure 30 Docking trajectories for different vision-feedback controllers.

The corresponding trajectories of the Rob@work 3 docking with different controllers are presented in Figure 30. The docking trajectory is practically the y_{mar}-x_{mar} plot which illustrates

movement toward the docking platform. The plot contains the signal after the transformation matrix is applied to the raw measurements from the camera.

Figure 30 depicts the docking trajectories of the different designed controllers 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 experiments, 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 a radius of threshold along the x_{mar}-axis ($x_{thresh} =$ 1 mm).

The desired 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 evaluated controller gains.

Table 5 Gain tuning inof vision-feedback controlfor docking of the Rob@work 3.

Feature / Exp.	Docking Time [sec]	Offset (y_{mar}-axis) [m]	Successful Docking
1 (Blue)	24.88	0.01411	Yes
2 (Green)	26.61	0.01605	Yes
3 (Red)	27.01	0.01473	Yes
4 (Pink)	55.79	0.0201	No

According to Table 5, the first and second experiments are quite fast and have more desirable response to fulfill the criterion of successful docking. The first experiment has the shortest average docking time and the offset along the y_{mar}-axis is smaller than the second one. Overall, gains from the first experiment which is illustrated with blue color in Figure 30 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 different initial configurations inside 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.

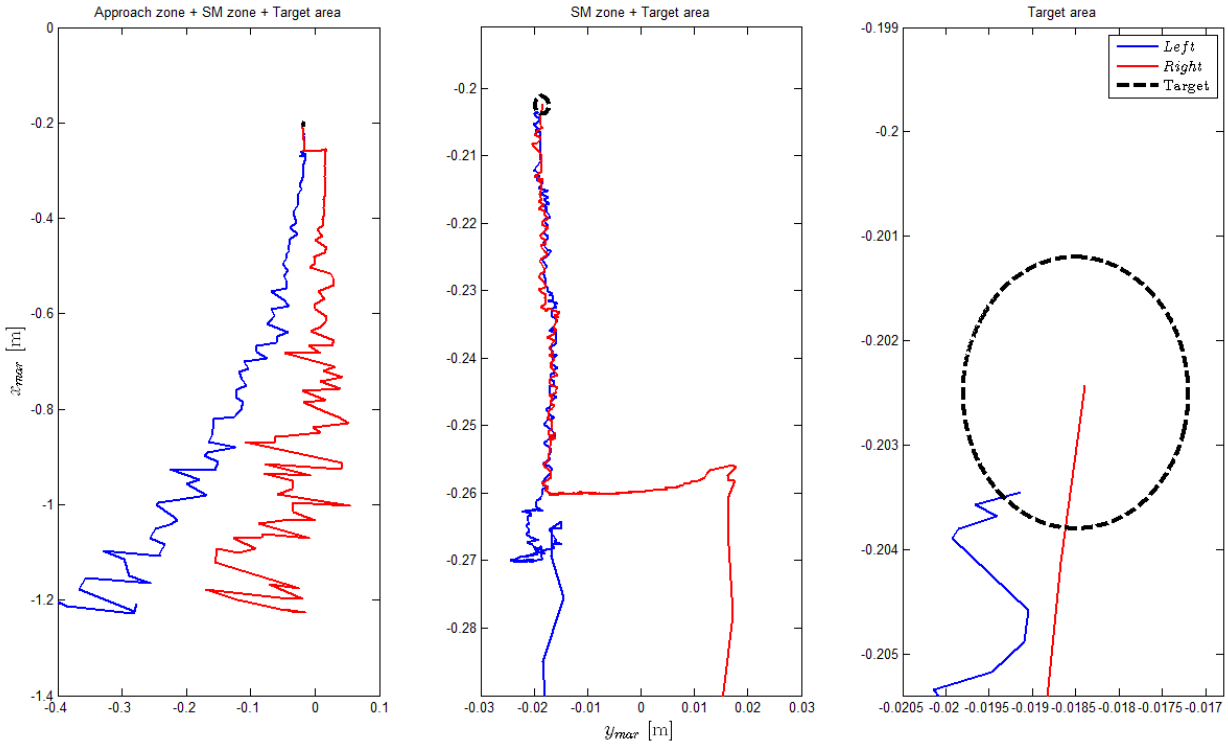


Figure 31 Obtained trajectory of the designed controller for two different initial configurations.

The camera position and orientation in the fixed marker coordinate system $(x_{mar}, y_{mar}, \theta_{mar})$ is depicted in Figure 32.

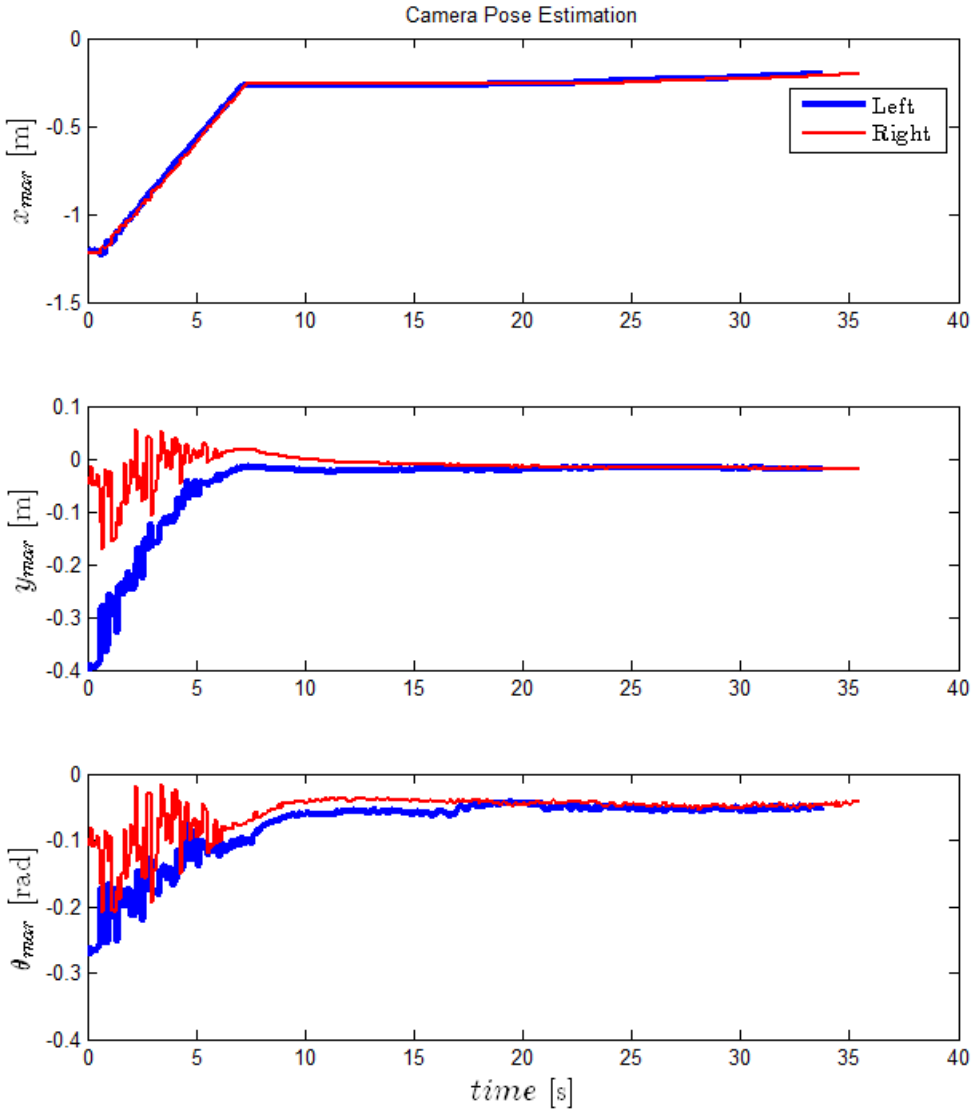


Figure 32 Camera pose estimation in the marker coodinate system.

The control signals for the robot coordinates ($x_{Rob}, y_{Rob}, \theta_{Rob}$) are illustrated in Figure 33.

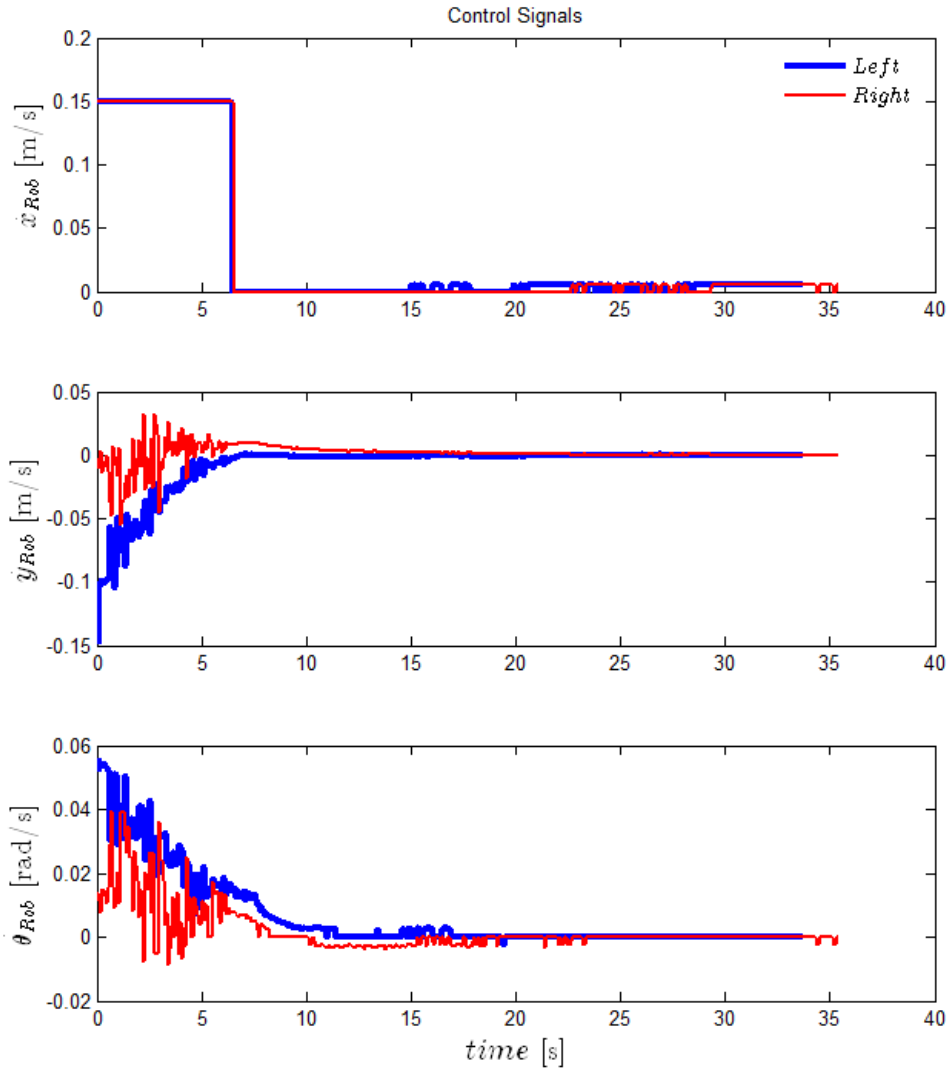


Figure 33 Cntrol signals for the Rob@work 3 along the axes of the robot.

In this experiment, the left configuration exhibits less offset along the y_{mar}-axis than the right one (1.5 cm) and the docking is faster ($t_{docking} = 34$ s).

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 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:

$$\begin{cases} \theta_{\mathrm{seg}} = \dfrac{\theta_{\mathrm{up}} - \theta_{\mathrm{dwn}}}{\theta_{\mathrm{SR}}} \\ \\ y_{\mathrm{seg}} = \dfrac{y_{\mathrm{up}} - y_{\mathrm{dwn}}}{y_{\mathrm{SR}}} \end{cases} \tag{18}$$

The pseudo-code of the developed strategy 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 into virtual grid.

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 34 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 coordinate system. Each index resembles to a specific state inside the grid which is also recognized with the real value in the docking area.

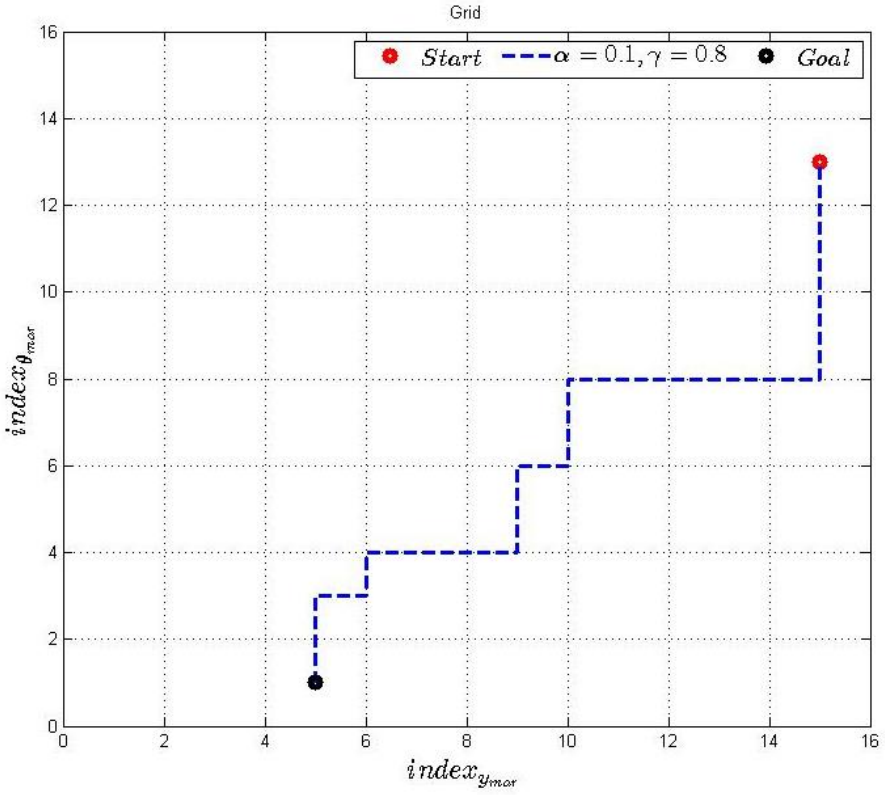


Figure 34 Optimal trajectory obtained after training with Q-learning in simulation.

Figure 34 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 34 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 the learning rate ($\alpha = 0.1$) and the discount factor ($\gamma = 0.8$) while the single entry is updated in each step.

Although Figure 34 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 34.

4. Discussion

The autonomous docking of mobile robots demands accurate sensor measurements. In this thesis, the sensor integration of the laser scanner and vision sensors were investigated to obtain the desired behavior with respect to the short docking time and trajectory.

The S300 Sick laser scanner has 270° angular range and the radial distance range of 30 m to perceive the surrounding objects. It has a resolution of 1-3 cm to perceive environment and detect obstacles (S300 Standard 2016).The conducted experiments on this paper, however, have shown docking a mobile robot demands a high precision from the sensor measurements and laser scanner measurements have several uncertainties to detect the target appropriately in the data acquisition process. 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 pose estimation with respect to docking platform.

In the vision-based approach, the raw measurements for pose estimation of the Rob@work 3 are approximated to the docking platform by applying the transformation matrix, then employed to design the vision-feedback control system to increase the precision. The target is represented by the ARTag marker, which is capable of publishing the orientation (θ_{mar}^{Ref}) besides the position (x_{mar}^{Ref} and y_{mar}^{Ref}).

The ARTags with the unique marker ID are prioritized over the simple point fiducials since they provide the orientation of the Rob@work 3, in contrast to the 2D pose estimation with the point

fiducials. Besides, Figure 8 and Figure 11 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. The orientation and the vision control design along the θ_{mar}-axis ensure the precise docking in different initial configurations even if the mobile robot is not perpendicular to the target. ARTags depict higher transparency to detect objects 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 for the camera pose estimation with respect to the fixed marker coordinate system. The camera and marker frames are added to the Rob@work 3 with the ROS transformation package (*tf*), which lets the user track multiple coordinate systems in real-time (Saito 2015). The control signal computations demand accurate measurements of the ARTag for the pose estimation of the camera with respect to the docking platform ($x_{mar}, y_{mar}, \theta_{mar}$).

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-app 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 for 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 to compute the control signals if the USB camera is employed as the vision sensor, whereas it takes approximately twice the time (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 for certain tasks. According to Figure 25, the average sampling time for both velocity and position samples are computed to get rid of jitters and inconsistent measurements of the ROS time in control design. Therefore, 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 in the control system depicted in Figure 13 to design an appropriate controller for the docking task. However, due to the sophisticated computations of the under-carriage control, a first order integrator is considered as the plant to simplify the computation. The control signals are the velocity commands applied to the Rob@work 3 to move toward the docking platform. The control commands are the linear velocities along the x_{Rob}-axis, the y_{Rob}-axis, and the angular velocity 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 26, 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 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 (i, j) 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 were employed to evaluate the accuracy of docking with respect to time and obtained trajectory. Docking platform is identified by a detectable marker for each sensor.

The laser scanners depicted several inconsistencies while detecting the marker throughout the experiments. Among the experiments, for instance, neither the cylindrical bottle nor the box detected with high accuracy by the sensor when the robot was docked manually within 10cm of safety margin line. Therefore, since high position accuracy is the main concern of this project and the built-in laser scanner sensors are mainly for the safety purposes, a vision sensor is added to the Rob@work 3 to increase the accuracy and robustness with the designed vision-feedback control.

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. Initially, camera position is estimated in the camera coordinate system. In the process, marker coordinate system is fixed to the docking platform. Therefore, the transformation is applied to the camera frame to estimate the camera position in the marker frame which will be published with the ROS transformation package (Saito 2015).

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 coordinate system 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 time for computation. The local Prace computer (see Figure 2) has sufficient memory to deal with large iterations in simulation. Training the Rob@work 3 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 reinforcement learning method reveals that the vision-feedback control system is more accurate and robust for docking of the Rob@work 3 considering desired 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.

6. Bibliography

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

Connette, C. P., Parlitz, C., Hagele, M. & Verl, A. 2009, Singularity avoidance for over-actuated, pseudo-omnidirectional, wheeled mobile robots. In: 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. In: Journal of Machine Learning Research, 5:1. Pp. 1-25.

Fiala, M. 2005, ARTag, a fiducial marker system using digital techniques. In: 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]. Doctor of Philosophy 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. In: 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. In: 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. In: 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. In: Foundations and Trends® in Computer Graphics and Vision, 1:1. Pp. 1-89.

Lier, F. 2013. ros_aruco [web document]. Published 2013, updated 16.6.2016 [Referred 23.10.2016]. Available: https://github.com/warp1337/ros_aruco.

Nguyen, V., Martinelli, A., Tomatis, N. & Siegwart, R. 2007. A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics. In: 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.

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.

Appendix A

The pose estimation of the camera with respect to the fixed marker coordinate system is dependent on physical parameters of the camera such as the focal length of the lens, the size of the pixel, the position of the principal point and the position and the orientation 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, for the further pose estimation. The result of the camera calibration is called the distortion parameters. 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):

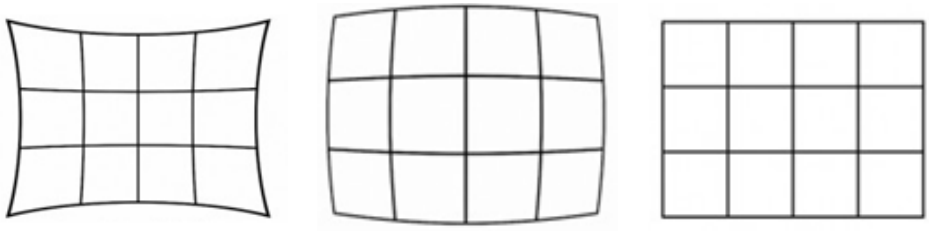
$$K_{3\times3} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix};\ M_{3\times4} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{19}$$

In (19), f represents the distance to the pinhole camera and the unit is meter (Forsyth, David A.; Ponce, Jean;, 2012). The extrinsic parameters are used in the pose estimation with the camera with respect to the fixed marker coordinate system as graphically sketched in Figure 6 for R and T matrices.

$$R = \begin{pmatrix} r_{11}\ r_{12}\ r_{13} \\ r_{21}\ r_{22}\ r_{23} \\ r_{31}\ r_{32}\ r_{33} \end{pmatrix};\ T = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} \tag{20}$$

The lens distortion is categorized into the radial distortion and tangential distortion. Geometrically, the radial distortion is depicted as curved lines, which are more visible on the

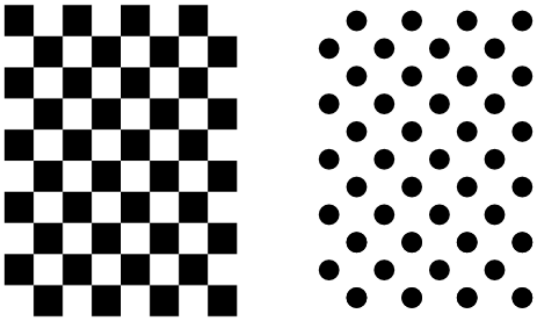
edges of the image frame since it changes the distance between the image center and an arbitrary point on the frame while has no effect on the direction of the vector between two points (Forsyth et al. 2012, p. 139). The final image has a pattern similar to one of the patterns depicted in Figure 35. An appropriate image for further image analysis is similar to Figure 36 c).



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

Figure 35 Different distortions in the calibration process.

To calibrate the camera and derive the internal parameters, standard patterns are employed as shown in Figure 36. They can vary from the classical black-white chessboard to the symmetrical or the asymmetrical circular pattern.



a) Rectangle. b) Symmetrical circle.

Figure 36 Common patterns in the 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 the camera and then calibrate.
- Find pattern in the 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 an exclusive file for each camera, which can be loaded for further image processing. The file is the human friendly YAML file containing the essential camera parameters.

Appendix B

The source codes for the visualization, the GUI and the RL can publicly be accessed at https://github.com/mrgransky/Autonomous-Vision-Based-Docking-Rob-work-3 .