

# 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

© 2016 by Farid Alijani, All rights reserved.  
Printed in Sweden by Media-Tryck  
Lund 2016

# Abstract

Docking of mobile robots requires precise measurements along the robot and the platform coordinate systems. Besides, the pose estimation of the robot with the sensors in an indoor environment should be accurate enough for localization and navigation toward the docking platform. However, the sensors are entitled to errors if they are not perfectly made. The sensor data fusion is exploited to decrease the measurement errors yet increases the accuracy of the docking.

The first approach in the autonomous docking of the mobile robots 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 the marker mounted on the docking platform, identified as the target. The vision-feedback control system instantly evaluates the current position of the robot and the target to compute the velocity commands for the actuators and minimize the error. 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 which training is handled in the simulation environment with the reward distribution.

The obtained results of the approaches are evaluated based on the optimal docking behavior of 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 robots 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 freedom they gave me to go beyond the initial proposed topic to investigate the model-free Reinforcement Learning (RL) as extra part of this project. It would not have been possible to conduct this research without their help. I highly appreciate all the practical ideas about docking of the mobile robot from them.

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



# Contents

1. Introduction.....	9
1.1. Background.....	9
1.2. Motivation .....	9
1.3. Problem formulation.....	10
1.4. Platform .....	10
1.5. Software and Hardware Integration.....	11
1.6. Thesis Outline.....	13
2. Methodology .....	14
2.1. Autonomous Laser Scanner-based Docking.....	14
2.2. Computer Vision.....	17
2.3. Fiducial Markers.....	22
2.4. Autonomous Vision-based Docking.....	27
2.5. Reinforcement Learning (RL) .....	34
3. Result .....	41
3.1. Laser scanner sensor .....	41
3.2. Computer Vision.....	48
3.3. Vision-feedback Control Design .....	50
3.4. Reinforcement Learning .....	61
4. Discussion .....	66

5. Conclusion .....	70
6. Appendix.....	73
6.1. Camera Calibration.....	73
6.2. Source Code of Project.....	75
7. Bibliography .....	76



# 1. Introduction

This MSc thesis addresses the extracted information of the measured data from different sensors for docking the mobile robots. Sensor data fusion is exploited to accomplish the precise docking. The control method is designed for each sensor to instantly evaluate the current state of the robot and the target. The laser scanners, the computer vision and the 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 the industries employing mobile robots with the platforms to be lifted or moved temporarily. It is also crucial for the applications in which the mobile robots recharge its batteries autonomously. Moreover, the autonomous operations are highly preferable in robotics since it eliminates the human interactions. Such behavior demands the robots to accomplish the certain tasks precisely. For this purpose, the robot should be equipped with sensors to first localize itself in the unknown environment then achieve the further goal configurations. However, the sensor measurements are not flawless and can be inconsistent. Therefore, data fusion of the multiple sensors could perceive higher accuracy in the docking.

## 1.2. Motivation

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

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

### **1.3. 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. Multiple sensor data fusion is investigated to evaluate the smoothness and the pose estimation accuracy of the sensors with respect to the docking platform.

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

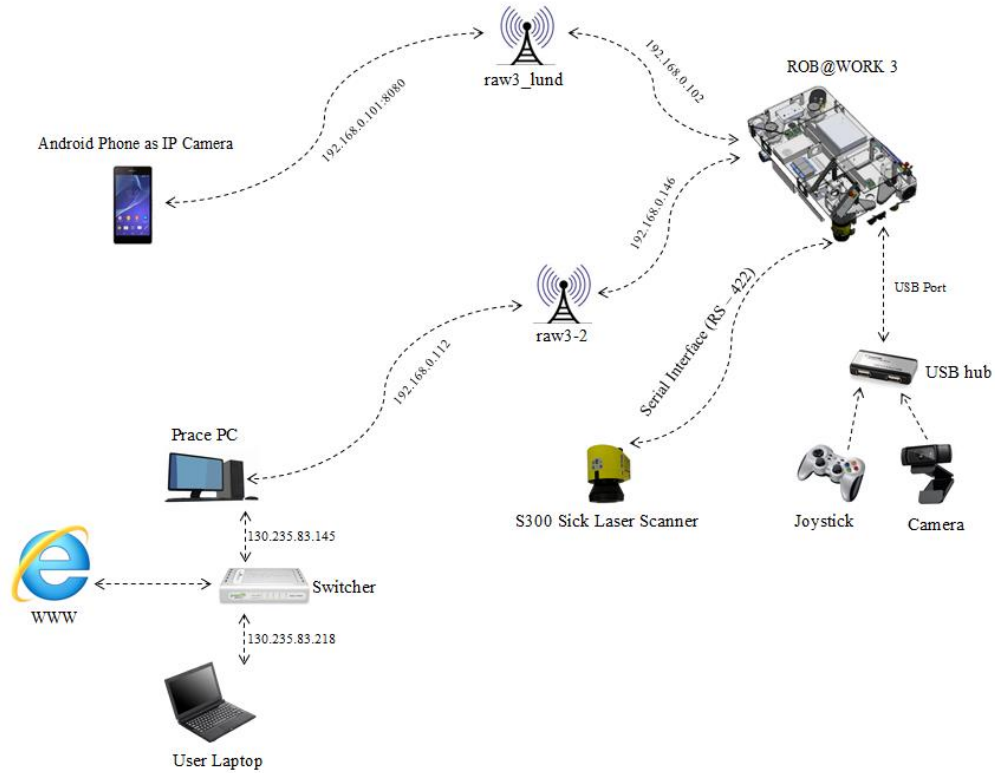
### **1.4. Platform**

The mobile robot used in this project is Rob@work 3, designed at the Fraunhofer Institute for Manufacturing Engineering and Automation (IPA, 2009) and developed for further research in Robotics Lab at Lund University as an omnidirectional platform which realizes arbitrary velocity and rotational commands. Omnidirectional wheels enable robot to move freely in constrained places and control the complete kinematic chain of platform in all direction. The platform contains physical components, including sensors, actuators and on-board computer. An optimal performance demands a desirable control of actuators attached to platform as well as other components and setups on the docking platform such as end-effectors and collinear joints.

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

### 1.5. Software and Hardware Integration

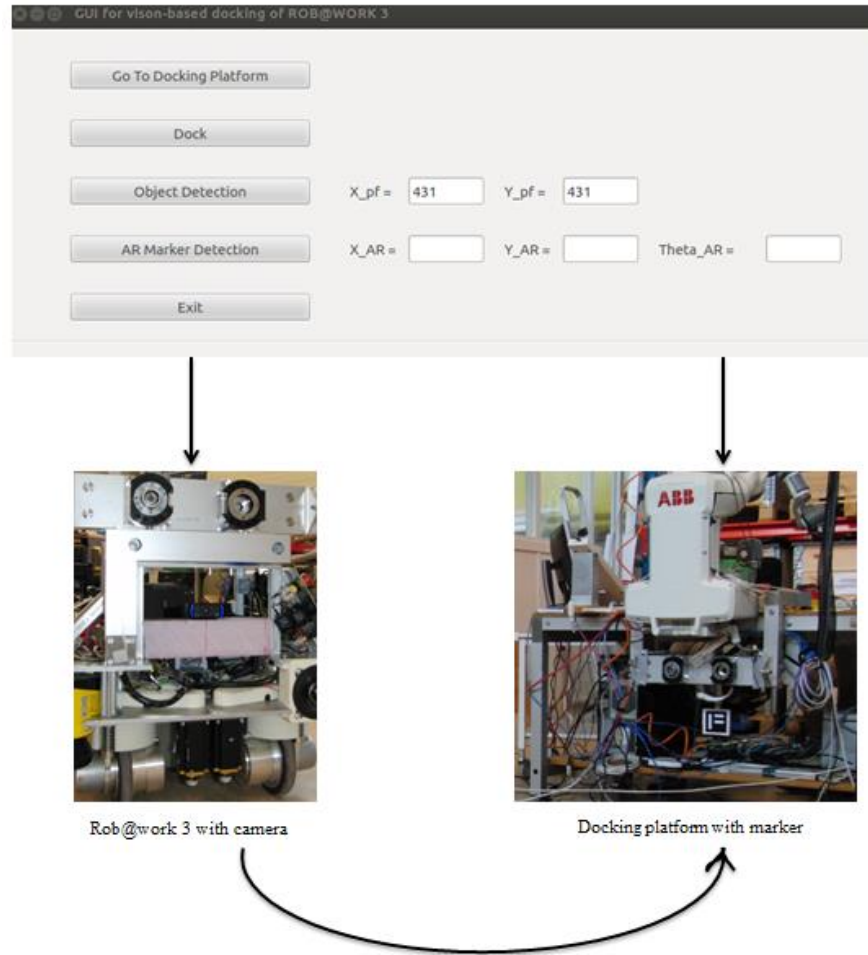
In this thesis, several hardware and software are employed to design the control system to achieve the precise docking. The omnidirectional platform, the onboard and the local PCs, the laser scanners, and the cameras are among the subsystems whereas the open source C++ interfaces such as the OpenCV (OpenCV, 2000) and the Qt-Creator (Qt, 1994) are the software involved in this project. The MATLAB software also employed to plot the obtained results of the different sensors. All the interfaces of the Rob@work 3 operate under the Robot Operating System (ROS) as the open source project platform.



**Figure 1** Sensor 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 with surrounding objects. The vision sensors are two cameras, e.g., the android phone 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 with the joystick.



**Figure 2** Hardware and software architecture for docking the Rob@work 3.

The first layer of the software is written in the C++ framework as the main program of the video streaming, the image processing, the controller/reinforcement learning, the localization and the navigation. The position of the robot is estimated in an indoor environment and rendered with the 3D vector graphical arrows in Rviz environment, whereas the vision sensors detect the marker. The visualization with different cameras, the obstacle detection with laser scanners and the vision-feedback control are among the further layer of the program to compute the robot commands and send them to the actuators.

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 Open Graphic Library (OpenGL) and the vision-oriented 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 of augmented reality marker attached to the docking platform.

Section 2.5 simulates a model free Q-Learning approach for docking. A virtual grid for Q-matrix is developed to find an optimal action selection policy while the robot obtains the highest possible rewards.

Chapter 3 presents the obtained results for docking mobile robots with laser scanners, vision-feedback control and Q-Learning.

Chapter 4 and 5 discusses the obtained results and compare different methods used in this project to find optimal solution for the docking problem.

Chapter 6 represents some detail information for the formulas used in the computer vision (2.2) and the Q-Learning (2.5) sections.

## 2. Methodology

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

### 2.1. Autonomous Laser Scanner-based Docking

Knowing the position of the robot in a known or 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. One of the possible choices is to use 2D laser rangefinder as it has become increasingly popular in mobile robotics (Nguyen, Viet; Martinelli, Agostino; Tomatis, Nicola; Siegwart, Roland;, 2005).

High accuracy of the laser scanners in non-contact measurements has increased their applications in the robotics to detect and avoid obstacles in walking, line following and position estimation in an indoor environment (Teixidó, Mercè; Pallejà, Tomàs; Font, Davinia; Tresanchez, Marcel; Moreno, Javier; Palacín, Jordi;, 2012).

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

environment, the control system which is established based on sensor measurement will avoid collision.

Besides, two simple shaped objects such as a cylindrical bottle ( $r = 7.5 \text{ cm}$ ) and a box ( $30 \times 23.5 \times 21 \text{ cm}^3$ ) have been used as markers and placed on the reference area for analyzing the raw data obtained from laser scanners. There are two different cases for using markers in this context. First, the robot is fully docked and the cylindrical marker is placed in front of the laser scanner sensor to get the approximate reference position. The second case, however, represents the robot moving towards the docking platform while the marker is still placed in the reference point. There is a minimum proximity range of 10 cm for obstacle detection on the sensor defined to protect the robot from collision.

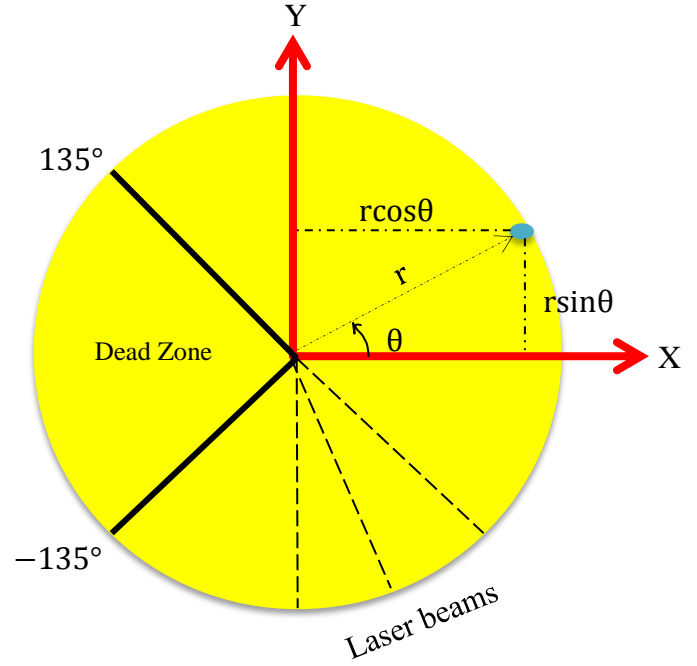


**Figure 3** Different markers investigated for docking with the laser scanner sensor, circular bottle (left), box (right).

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



a) Front view.



b) Geometrical representation.

**Figure 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 on the plane and the angle  $\theta$ . The 2D position can be derived as below:

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

The laser beam indicates an angle increment ( $\theta_{inc}$ ) with a constant value and is represented with an index ( $i$ ) within the computation range of  $[-135^\circ, 135^\circ]$  in which lowest index ( $i = 0$ ) resembles to angle ( $\theta_{inc} = -135^\circ$ ). The index, however, needs to be declared as a beam angle ( $\theta_b$ ) in the process for simplification. The following formulas convert the index to the actual beam angle of the laser scanner:

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

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



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

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

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

Docking of the Rob@work 3 with laser scanners demands a detectable marker attached to the docking platform, act as a reference point and identified as an obstacle. The laser sensors scan the environment to detect all possible objects in the front side within the angular range of  $[-135^\circ, 135^\circ]$ . Backside of the laser scanners, e.g., 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**

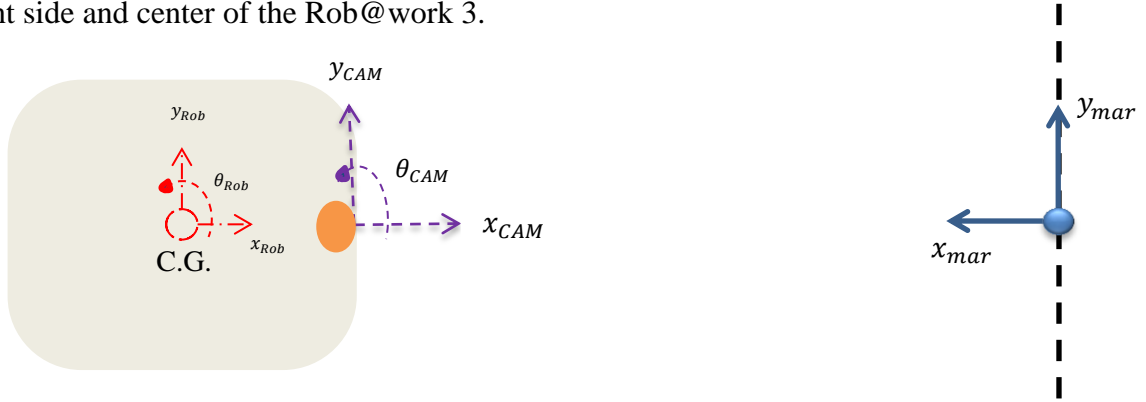
Similar to eyes, cameras are helpful and effective in robotics since vision allows noncontact measurement in various domains, including object recognition, localization and manipulation. Camera is practically a vision sensor mounted on robot to sense environment and plan appropriate actions in different conditions (Corke, 2013). The position of the camera is totally independent of which control configuration is employed as long as it visualizes the local environment, however, it is necessary to calibrate camera before the visualization process started and determine the geometric parameters of the camera (Connette, Christian P.; Parlitz, Christopher; Hagele, Martin; Verl, Alexander; 2009).

In the computer vision, the geometric camera parameters, e.g., intrinsic and extrinsic matrices are calculated for the calibration process. In this project, the image processing has two arguments for the ARTags detection, the calibration file and the marker size. The calibration process and the algorithms are completely presented in section 6.1.

### **2.2.1. Geometric Representation and Transformation**

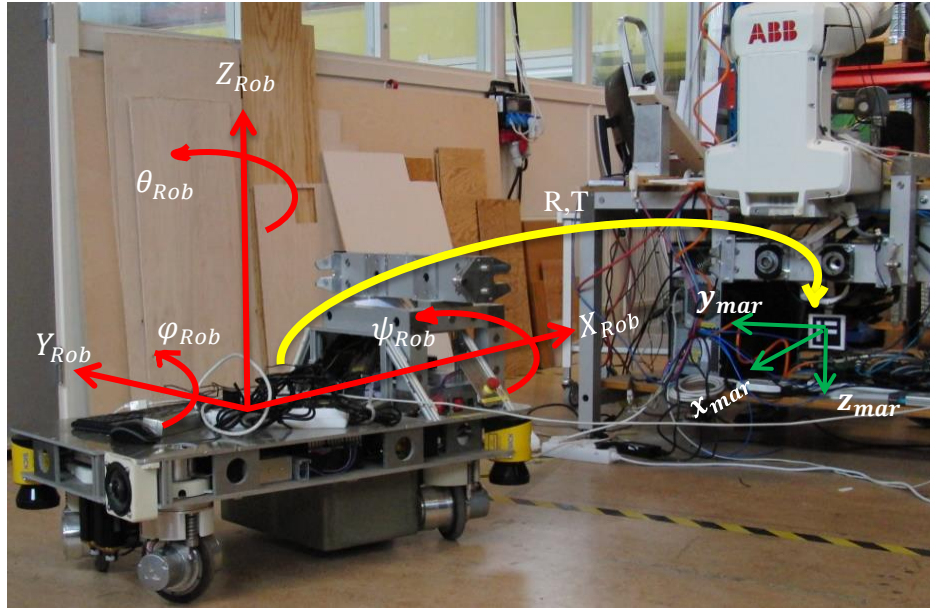
An analytical geometry also known as coordinate geometry is used to determine the camera correlation with captured objects in a fixed coordinate system in which position and orientation of the robot with its camera onboard are estimated accordingly (Forsyth, David A.; Ponce, Jean; 2012). The geometric modeling in this project is categorized to three different coordinates in real time, collaborating in localization and docking. Figure 5 illustrates the marker frame which is

attached to docking platform, the robot frame (Rob), and the camera frame (CAM) mounted to the front side and center of the Rob@work 3.



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

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



**Figure 6** Rob@work 3 and ARTag coordinate systems.

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

along  $X_{\text{Rob}}$ -axis and  $Y_{\text{Rob}}$ -axis are neglected. Therefore, it has three degrees of freedom on 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 a primitive perpendicular case in which the robot is meant to approach the goal directly and the orientation is usually adjusted by the operator. Besides, vision feedback control is necessary to control orientation after extracting measurement of the sensor. Therefore, it facilitates for the robot to approach the docking platform from more than one configuration.

### 2.2.2. Pose Estimation

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, David A.; Ponce, Jean, 2012). The target is assumed to be at the origin of the marker to simply initialize the reference values ( $y_{\text{mar}} \cong 0$  m and  $\theta_{\text{mar}} \cong 0^\circ$ ). Translation vector and rotation matrix are needed to align the axes of the two frames.

In the robotics and the computer vision, the pose estimation plays an essential role for the navigation and the localization to accomplish the goal configuration without collisions and misalignment. The vision sensors, e.g., mounted cameras on the robot, and markers 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 broadcasted 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 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 camera ( $x_{\text{CAM}}, y_{\text{CAM}}, \theta_{\text{CAM}}$ ) are calculated in the fixed marker coordinate system since the robot moves on a plane. Figure 5 depicts the camera coordinate frame on the Rob@work 3 while fixed marker coordinate frame is attached to the docking platform. 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  $3 \times 1$  vector  $[x_{mar}, y_{mar}, z_{mar}]^T$  in which the  $z_{mar}$  is eliminated on the plane, therefore its 2D position  $(x_{mar}, y_{mar})$  is used to define the marker frame. 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, 2011).

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

$$R_{3 \times 3} = \cos\theta I_{3 \times 3} + (1 - \cos\theta)rr^T + \sin\theta \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix} \quad (5)$$

$R_{3 \times 3}$  matrix can be simplified as follow:

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

Therefore, orientation of camera along its entire axes, e.g., roll ( $\psi_{CAM}$ ), pitch ( $\phi_{CAM}$ ) and yaw ( $\theta_{CAM}$ ) is computed with respect to the fixed marker as below:

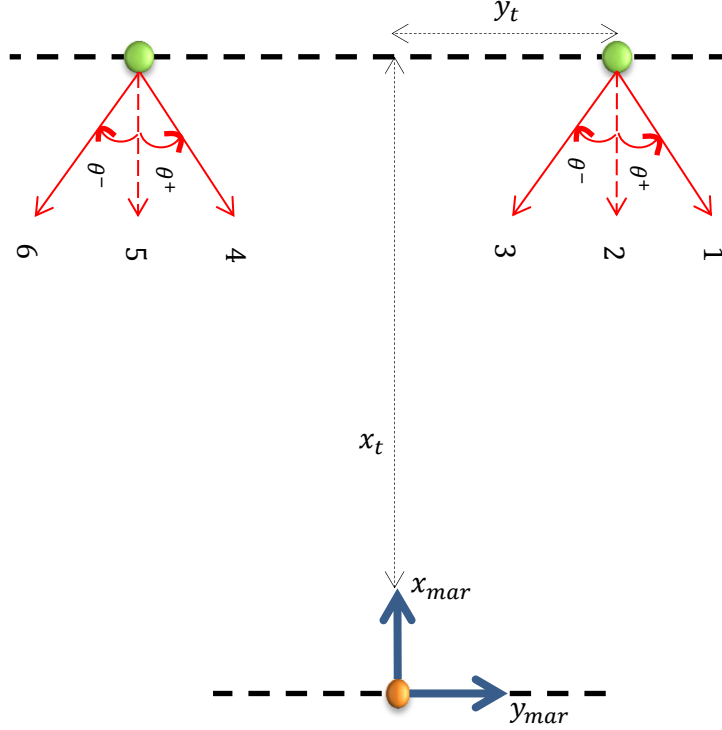
$$\psi_{CAM} = \tan^{-1}\left(\frac{r_{21}}{r_{11}}\right) \quad (7)$$

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

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

Roll and pitch are not applicable since the robot and the camera accordingly rotates neither along the  $x_{Rob}$ -axis nor  $y_{Rob}$ -axis. Therefore, yaw angle measured in the fixed marker coordinate system resembles to the orientation of the camera ( $\theta_{CAM}$ ). The configuration is such that positive

angle means counter clockwise rotation, whereas clockwise rotation corresponds to negative orientation. Further, if the robot is placed perpendicular to the platform, the orientation is approximately zero ( $\theta_{CAM} \cong 0^\circ$ ). Figure 7 represents six various configurations in which the Rob@work 3 is placed in different angles initially. Control signals along  $\theta_{Rob}$  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:

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

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.	$y_{\text{mar}}$	$\theta_{\text{mar}}$	Expected $\dot{y}_{\text{rob}}$	Expected $\dot{\theta}_{\text{rob}}$
1	+	$> 0$	$\rightarrow$	CW
2	+	$\approx 0$	$\rightarrow$	0
3	+	$< 0$	$\rightarrow$	CCW
4	-	$> 0$	$\leftarrow$	CW
5	-	$\approx 0$	$\leftarrow$	0
6	-	$< 0$	$\leftarrow$	CCW

After the pose estimation, the camera coordinates are published with respect to 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 will be subscribed in a callback function to begin the motion control process to move the robot toward the docking platform.

### 2.3. Fiducial Markers

Fiducial marker is a special 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 marker is required. It also provides accurate pattern with a digital word that contains a unique ID protected from false detection (Fiala, 2005).

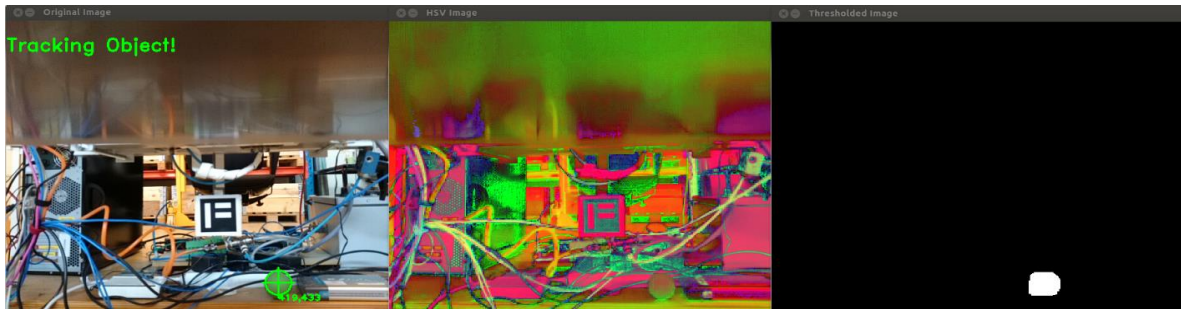
Fiducial marker system has some advantages to be considered as the target of docking the Rob@work 3 in this project. First, the identification is practically easy since such a black and white square sketch with embedded internal pattern facilitates the marker detection. Second, it speeds up the marker detection with accompanying computer vision detection algorithm. 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 pose estimation correctly. Finally, it provides accurate and low-cost solutions to develop robust detection algorithm (Lepetit, Vincent; Fua, Pascal;, 2005).

Fiducials are divided in two general categories of a point fiducial and a planar fiducial with which the vision-based docking of the Rob@work 3 is investigated. In the point fiducial, the camera coordinates are chosen has the 2D setup with only positions  $(x_{CAM}, y_{CAM})$ , whereas more advanced setup with the 3D coordinates to indicate positions and orientation  $(x_{CAM}, y_{CAM}, \theta_{CAM})$  is described at 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 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, Vincent; Fua, Pascal;, 2005).



a) Original image.

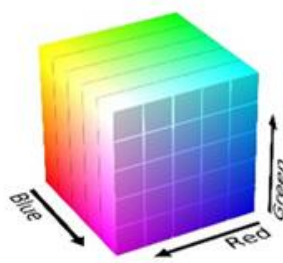
b) HSV image.

c) Threshold image.

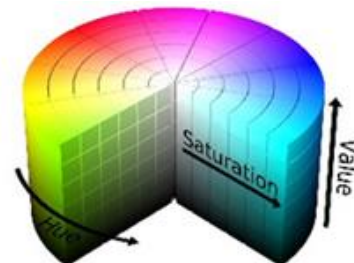
**Figure 8** Point fiducial detetction.

A video stream with the resolution of  $640 \times 480$  is an input matrix for the tracking algorithm to start the image processing to detect the point fiducial. Figure 8 a) depicts the detected fiducial with the OpenCV in the camera coordinate system in which a green ball attached to docking platform as the target. Figure 8 c) illustrates the threshold window and detected point fiducial after the morphological operation is performed to eliminate the noise.

Figure 8 b) shows the docking area in HSV color space with green ball. The input matrix has a RGB (Red, Green and blue) color space which is a convergence matrix each of which contains integer values ranges from 0 to 255. For color based segmentation, RGB format is represented 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. In the HSV color-space, Hue represents color, Saturation is amount of color that is mixed with white, and Value is the mixture with black (D'Silva, 2008).



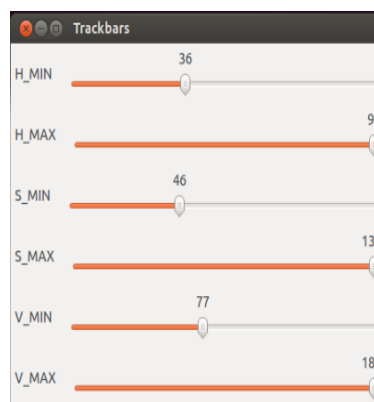
a) RGB (0-255).



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

**Figure 9** Image color-spaces (Alves, 2015).

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 OpenCV library (OpenCV, 2011). The track bar shown in Figure 10 adjusts HSV values to detect the green ball as a point fiducial in docking the Rob@work 3.



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




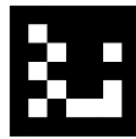



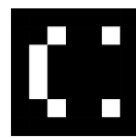


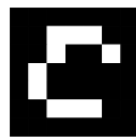



The lighting condition in this approach, however, is quite challenging since the object used as a marker may not be reflective or symmetrical enough to be tracked properly or the environment does not have a uniform illumination with strong contrast. Therefore, the camera needs to be equipped with either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

### 2.3.2. Planar Fiducial

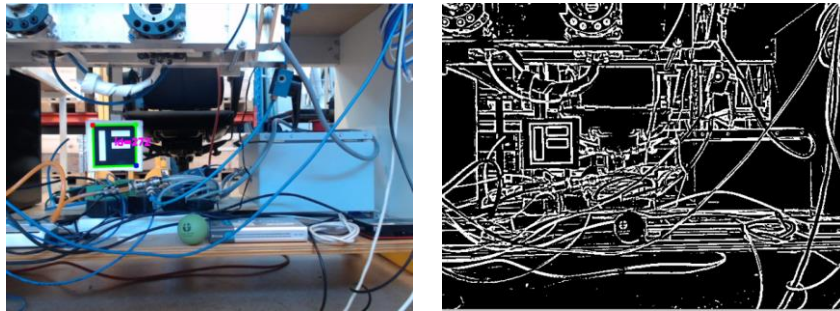
The augmented reality (AR) field has increasingly widened planar fiducial application in computer vision since it consists of unique patterns compatible with the Aruco marker detection algorithm (OpenCV, 2015). The square-bordered structure with opposite background color, e.g., black and white facilitates the recognition of the unique internal pattern with the definite shape and ID called as ARTag. 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 marker falsely, whereas the inter-marker confusion rate is the probability of obtaining wrong marker ID (Fiala, 2005). Table 2 represents a few standard AR markers compatible for Aruco marker detection algorithm in OpenCV, given specific IDs.

**Table 2** Standardized 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 docking area with high reliability in different lighting conditions and backgrounds. The different lighting conditions, however, do not effect on robust detection since ARTag benefits from edge linking and edge-based detection methods coupled with digital coding scheme of identification. The edge-based method is utilized to detect quadrilateral shapes directly from an image. Although line segment detection demands higher computations, occluded marker can still be detected even if some corners and sides are missing (Fiala, 2005).

The planar fiducial used in this project is chosen from the samples shown in Table 2 which is similar to the English letter “F” (ID # 272). It is generated based on a unique ID that is encoded in the marker image, edge size of the back box containing the marker and thickness of the white border surrounding the marker. Length of marker is 80 mm with 8 mm of paddle as a thickness of white border. The genrated marker is mounted to center of docking platform, e.g., 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 in the docking platform.

The pose estimation of the camera with respect to the fixed marker coordinate system is calculated with 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 the result chapter and control signals are computed accordingly.

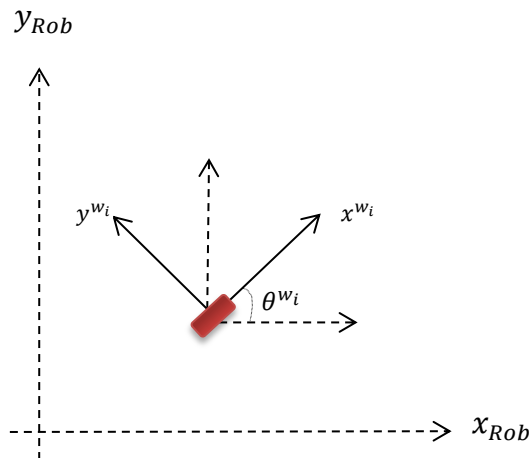
## 2.4. Autonomous Vision-based Docking

### 2.4.1. Kinematic Model

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

- 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 on a flat horizontal condition

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



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

Each wheel has also the local coordinate system  $(x^{wi}, y^{wi})$  which are parallel to the robot coordinates. Figure 12 illustrates a rotated wheel by  $\theta^{wi}$  angle. This angle is the orientation angle and each wheel has its own orientation. It is geometrically the angle between  $x_{Rob}$ -axis and  $x^{wi}$ -axis. The rotation of the wheels is computed with respect to the center of the gravity in which if the wheels rotate  $\theta^w$  degrees, the entire platform rotates with the same  $\theta^w$  degrees.

#### **2.4.2. Control Design**

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

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. This policy is economically not feasible and the vision system is only employed in the pose estimation to obtain the required motion commands to achieve the docking goal (Connette, Christian P.; Parlitz, Christopher; Hagele, Martin; Verl, Alexander;, 2009).

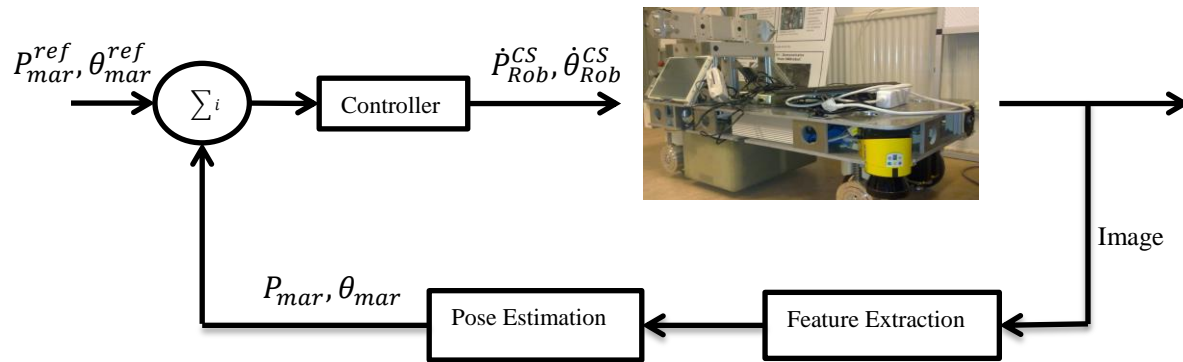
In contrast, the second control approach seems more practical since visual information on the feedback loop gets involved to increase the overall accuracy of the control system. In this approach, two images are virtually involved, one with the target which has an estimate position of the docking platform,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

In fact, this alternative approach constantly evaluates the current position of the vision sensors and the target to compute appropriate velocity commands to manipulate the robot to achieve docking task. The closed loop vision-based control, also known as Visual Servoing (VS) carried out by the visual information in separate images (Hutchinson, Seth; Gregory, Hager D.; Corke, Peter I., 1996). 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 platform of Rob@work 3 (Lepetit, Vincent; Fua, Pascal;, 2005).

In this project, fiducial markers represent the target, e.g., docking platform. The visual system provides position and orientation references for the control loop. Therefore, Position-Based Visual Servoing (PBVS) is employed to serve basically as the feedback to extract the features of the image in the control loop and compute the difference between the current positions of the mounted camera on the Rob@work 3 with the fiducial markers as the target. The PBVS approximately estimates the position and orientation of the camera coordinates with respect to the fixed marker as the target and sends appropriate the control signals to the robot (Wilson, William J.; Williams Hulls, Carol C.; Bell, Graham S., 1996).

The target plays a crucial role to design the vision-based control system since the current position of the robot is instantly evaluated to provide appropriate control signal to eventually minimize the error. Therefore, the more accurate the reference point, the more appropriate the control signals are. Therefore, the target is graphically sketched in as a circle with an origin of the 2D marker position, e.g.,  $x_{ref}^{mar}, y_{ref}^{mar}$  and a radius of threshold along  $x_{mar}$ -axis, e.g.,  $x_{thresh} = 1 \text{ mm}$ .



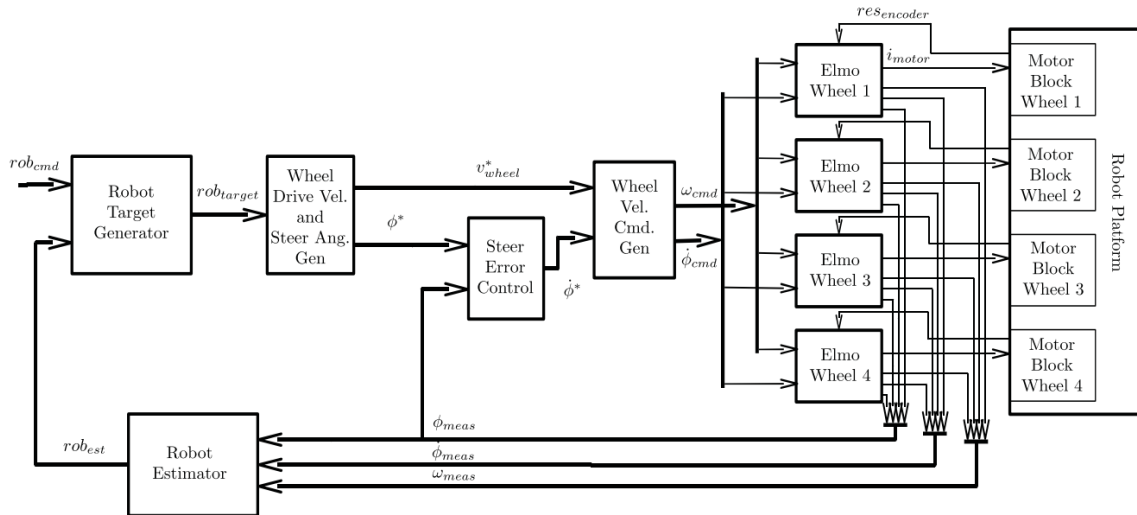
**Figure 13** Position-Based Visual Servoing (PBVS).

The most important part of the control system, however, is a plant. Theoretically, the plant is the combination of the process and actuator and expressed as a transfer function to declare the relation between the input and output signal (Ogata, 2010). 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 the embedded control system which sends the appropriate steering and driving velocity commands to the wheels within an interval of approximately 20 milliseconds depending on the command generator and computational load (Nilsson, 2010).

Figure 13 represents the vision-based closed loop control design employed to computer the control signals for the plant, e.g., the Rob@work 3 and its actuators. The desired 2D positions  $P_{mar}^{ref}(x_{mar}^{ref}, y_{mar}^{ref})$  and orientation ( $\theta_{mar}^{ref}$ ) are set as the reference signals when the robot is manually docked. The reference is constantly compared with estimated positions  $P_{mar}(x_{mar}, y_{mar})$  and orientation  $\theta_{mar}$  of the robot after the feature extraction of the markers to calculate the error signal as the input of the controller to obtain the robot commands correspondingly.

The robot commands are practically applied velocities for steering and driving motors of actuators to move the platform. Figure 14 depicts the under-carriage control system of the mobile robot. A  $rob_{cmd}$  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 in is simplified to a first order integrator to simplify the design in the vision based docking method.



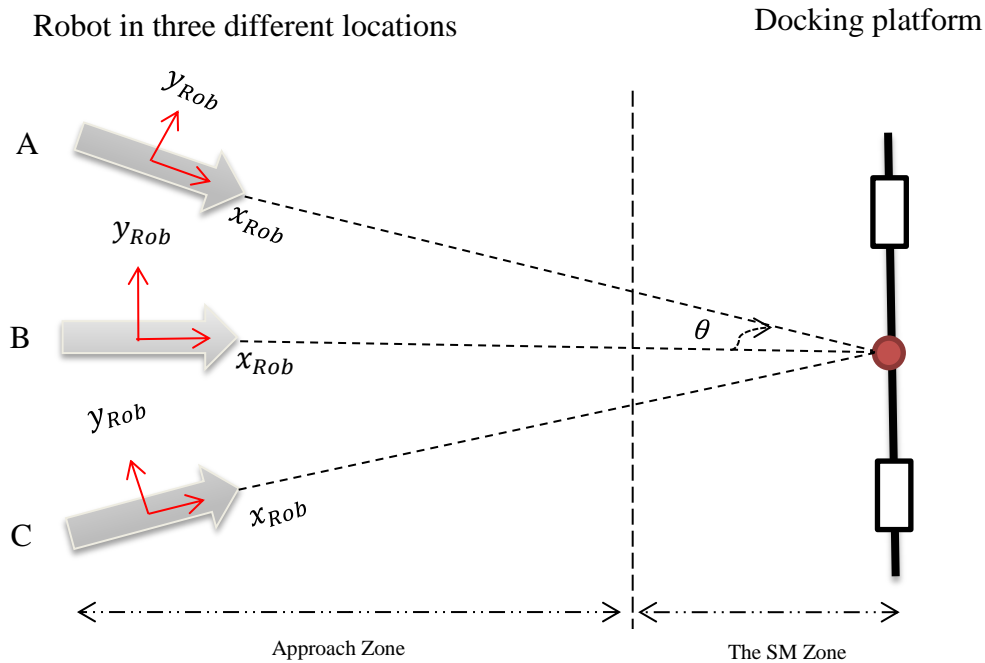
**Figure 14** The under-carriage control of the Rob@work 3 (Nilsson, 2010).

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 ( $\theta_{Rob}$ ) as stated below:

- Approach zone
- Safety Margin (SM) zone

It is assumed the Rob@work 3 always starts the process somewhere in the approach zone which is far from the docking platform along  $x_{mar}$ -axis (more than 18 cm from the platform). The SM, on the contrary, is closer (18 cm away from the platform) and only considered to adjust the orientation along  $\theta_{mar}$ -axis, the aside position along  $y_{mar}$ -axis, and the smoothness to accomplish higher accuracy before docking is completed.

In this project, various configurations are investigated to compute the appropriate steering and driving velocity commands for the actuators. According to Figure 15, adjusting a movement along  $y_{Rob}$ -axis and orientation along  $\theta_{Rob}$ -axis seems more crucial than movement along  $x_{Rob}$ -axis since robot can move toward the docking platform even if a constant driving velocity is applied, e.g.,  $Vel_x^{App.} = cte$ . Therefore, the control signal along  $x_{Rob}$ -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 docking area with Approach zone and The SM zone.

The control system is designed such that the robot always moves towards 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, e.g.,  $Vel_x^{SM} = 0$ . 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 ( $y_{thresh} = 2.5 \text{ mm}, \theta_{thresh} = 2^\circ$ ) are employed to terminate the controllers and switch to forward movement along the  $x_{Rob}$ -axis again to complete the docking task.

Time is critically crucial in docking 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 less average docking time. Gain tuning plays a key role to guarantee accurate behavior within less average docking time. The time is initially recorded on the embedded controller of the Rob@work 3 via ROS client library package with an appropriate time API (application programming interface) code in data acquisition process (Gomes, 2013).

The ROS is a built-in framework on the Ubuntu operating system to provide the deterministic scheduler for the given tasks which is not executed in the real time. In fact, the ROS framework has its own time configuration to deal with the control signals 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 ( $t_i$ ) final time ( $t_f$ ) and recorded samples of ROS time ( $t_s^{ROS}$ ). Each step is mathematically derived as the difference between two consecutive nodes. The vector of real time, on the contrary, is calculated as follow and utilized for further computations and control design:

$$\Delta t = t_f^{ROS} - t_i^{ROS} \quad (11)$$

$$t_{real} = 0 : \frac{\Delta t}{t_s^{ROS}} : t_d \quad (12)$$

The error is the difference of the target, e.g.,  $P_{ref}$  and the current position of the camera ( $P_{cur}$ ) in the fixed marker calculated below:

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

And control signal is derived accordingly:



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

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  $y_{\text{Rob}}$ -axis and orientation along the  $\theta_{\text{Rob}}$ -axis. The movement along  $x_{\text{Rob}}$ -axis, on the contrary, is given constant values depending on the zones the robot is placed.

```
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 =  $\frac{\text{error}_{\text{current}} - \text{error}_{\text{previous}}}{dt}$ ;

control signal =  $K_p * \text{error}_{\text{current}} + K_i * \text{error}_{\text{integrated}} + K_d * \text{error}_{\text{derivative}}$ ;

error_previous = error_current

sleep(dt)

}
```

**Algorithm 1** Linear feedback control pseudo code.

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

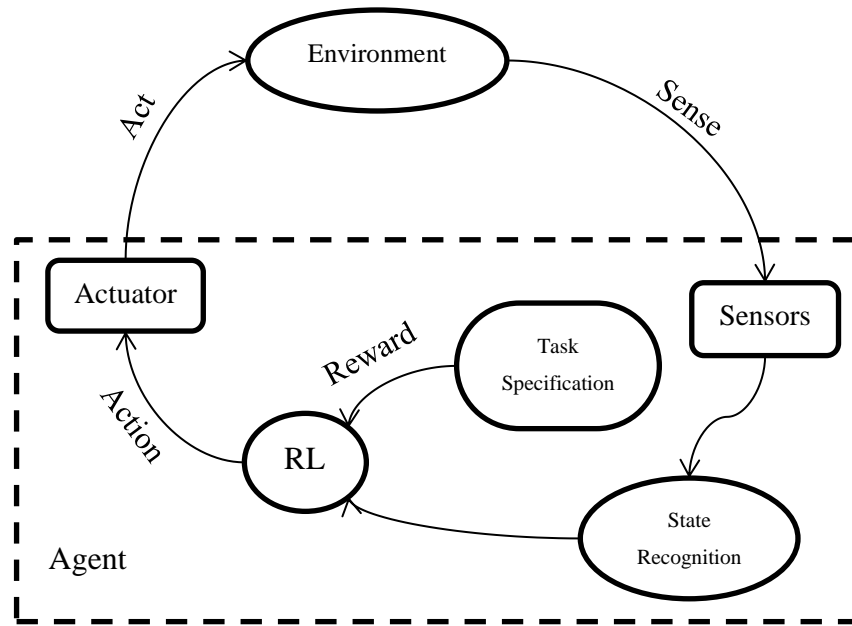
## **2.5. Reinforcement Learning (RL)**

### **2.5.1. Background**

Interacting with the environment is most likely the first and simplest step in the learning process. In the RL, despite supervised learning (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 learner is not told what to do or which action to take; instead it is the responsibility of the learner to explore the environment with several trials while taking different actions to eventually make the correct decision to accomplish the desired performance. This interaction is called goal-directed learning method and is among a focus of the RL approach (Sutton, Richard S.; Barto, Andrew G., 1998).

In fact, learning is the process of trying different actions based on the current state which ultimately put the agent in the target while obtaining the highest award. The reward, which is similar to the evaluative feedback in the control theory, is scalar and leads the robot toward desired behavior, eventually. Appropriate actions according to the current state of the robot in the docking area rewards the agent meaning that it reinforces helpful behavior that should be persistent, whereas wrong actions yield more negative rewards, e.g., punishments to inhibit the non-helpful behavior of the robot (Gaskett, 2002).

There are few nomenclatures 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 toward. It either learns or uses a provided knowledge to achieve its goal. A state physically represents the location of 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 the new one. It eventually effects the learning environment (Sutton, Richard S.; Barto, Andrew G., 1998).



**Figure 16** Reinforcement Learning setups (Sutton, Richard S.; Barto, Andrew G., 1998).

According to Figure 16, Robot acts on the environment with set of actions to eventually accomplish a given goal and practically manipulate the robot towards the target according to current state. The position and orientation will be changed while the robot moves from state to state accordingly. The actions are command velocities applied to robot platform with four omnidirectional wheels.

Among three fundamental approaches dealing with the RL problems, 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 estimate without waiting for the final outcome. The TD learning method has two different classes, including on-policy and off-policy control approaches (Sutton, Richard S.; Barto, Andrew G., 1998).

The off-policy control approach, known as Q-Learning, uses the prior knowledge to expedite the RL despite the value of the optimal policy learned independently from the actions (Jun, Li; Lilienthal, Achim; Martinez-Marin, Tomas; Duckett, Tom, 2006). On the contrary, the on-line policy learns to takes the slower path with the random policy since it wants to explore both optimal and non-optimal actions. Therefore, it is more careful about the environment. The target

in the Q-Learning method is the state identified as the docked robot and the actions are influences of the robot on the environment to obtain optimal policy of the system.

### **2.5.2. Problem**

Finding the optimal policy in docking of the mobile robots with respect to the trajectory and the average docking time is already investigated with the vision feedback control in the section 2.4. The model-free RL approach is investigated in this thesis, as an alternative, to compare the results with the vision-feedback control system to select the dominant approach.

### **2.5.3. Solution**

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

The Q-matrix acts as the brain and represents the memory of several experiences. Similar to the control design (section 2.4.2), controlling aside position along the  $y_{Rob}$ -axis and orientation along the  $\theta_{Rob}$ -axis are prioritized for the docking. Therefore, the Q-matrix mathematically contains orientation and position of the marker ( $\theta_{mar}, y_{mar}$ ) which are set as row and column of the matrix, respectively while movement along  $x_{Rob}$ -axis is not considered among actions ( $\dot{x}_{Rob} = cte$ ).

The successful docking is equivalent to the desired behavior which is evaluated with position and orientation accuracy to attach the end-effectors, smooth docking in which mechanical set ups do not collide or hit others 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 set up collision
- Obstacles detected by laser scanners

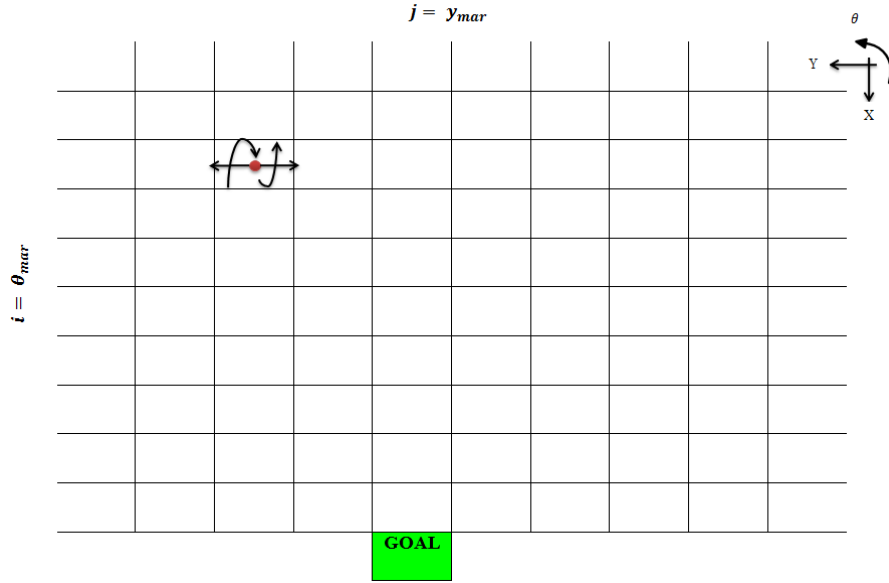
- Leaving the grid boundary

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 vision sensor, e.g., the camera. It can be lost if the 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 be failed.

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

Collision with the obstacle is strictly prohibited and stops the robot for the 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 collides with obstacles quite a lot of time since training takes place with the random actions, thus, as soon as one of the scanners detects the obstacle within its territory, learning terminates, 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, e.g.,  $\theta_{\text{mar}}$  and  $y_{\text{mar}}$  which are defined not only to protect the robot from mechanical collisions but also to make sure the marker is always visible in the image coordinate system. If the robot ends up at a place outside the imaginary grid, the robot gets punishment for the previous state the robot was before leaving the grid, moves to a new random position to start over while learning fails and episode counter resets.



**Figure 17** Developed grid for Q-Learning action selection policy.

Figure 17 depicts the virtual 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  $\theta_{mar}^{Ref}$  and  $y_{mar}^{Ref}$  are saved once the robot was manually docked. Red dot represents the agent, e.g., the Rob@work 3 placed at its current state.

The initial state is randomly chosen inside the grid to diversify states which are passed by robot. Actions, as mentioned earlier, are linear velocity along the  $y_{Rob}$ -axis and angular velocity along the  $\theta_{Rob}$ -axis. In another word, the robot moves left and right if the linear velocity, e.g.,  $\dot{y}_{Rob}$  is applied, whereas it rotates clockwise (CW) or counter-clockwise (CCW) if the angular velocity, e.g.,  $\dot{\theta}_{Rob}$  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 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 optimal policy even if the actions are totally exploratory (Russell, Stuart; Norvig, Peter;, 2010).

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 circumstances mentioned above. In the case of failure, the robot deserves the negative reward, e.g., punishment since it has learned 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 the new random position inside the docking area to start the learning process over. The transition rule of Q-Learning is represented below:

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

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

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

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

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

in which:

$0 \leq \gamma < 1$  : Discount factor  $\Rightarrow \gamma \cong 0$  : immediate reward &  $\gamma \cong 1$  : long-term high reward

$0 \leq \alpha < 1$  : Learning rate  $\Rightarrow \alpha \cong 0$  : No learning &  $\alpha \cong 1$  : prior values neglected

Generally, the learning is started with a large learning rate, e.g.,  $\alpha \cong 1$  to change Q-Values faster and lowered as the time progresses (Even-Dar, Eyal; Mansour, Yishay, 2003). An ideal environment does not deal with stochastic problems and it converges to optimal Q-function with  $\alpha = 1$ , whereas in practice, the convergence requires more prior information and learning factor closer to zero. An optimal constant learning rate of  $\alpha = 0.1$  is chosen for learning process (Sutton, Richard S.; Barto, Andrew G., 1998).

The Q-Learning pseudo-code algorithm has is represented as follow:

- Set parameters ( $\gamma, \alpha$ ) and environment reward (R).
- Initialize Q-matrix.
- For each episode:
  - Begin from a random initial state
  - While (the goal state has not reached)
    - {
    - Choose an action randomly.
    - Move to new state with a chosen action.
    - Consider a new sample estimate with 16.
    - Update Q-Values with prior knowledge of old Q-Values with 17.
    - Next state = Current state.
    - }
  - End While.
- End for.

**Algorithm 2** Psudeo-code algorithm for model-free Q-Learning.



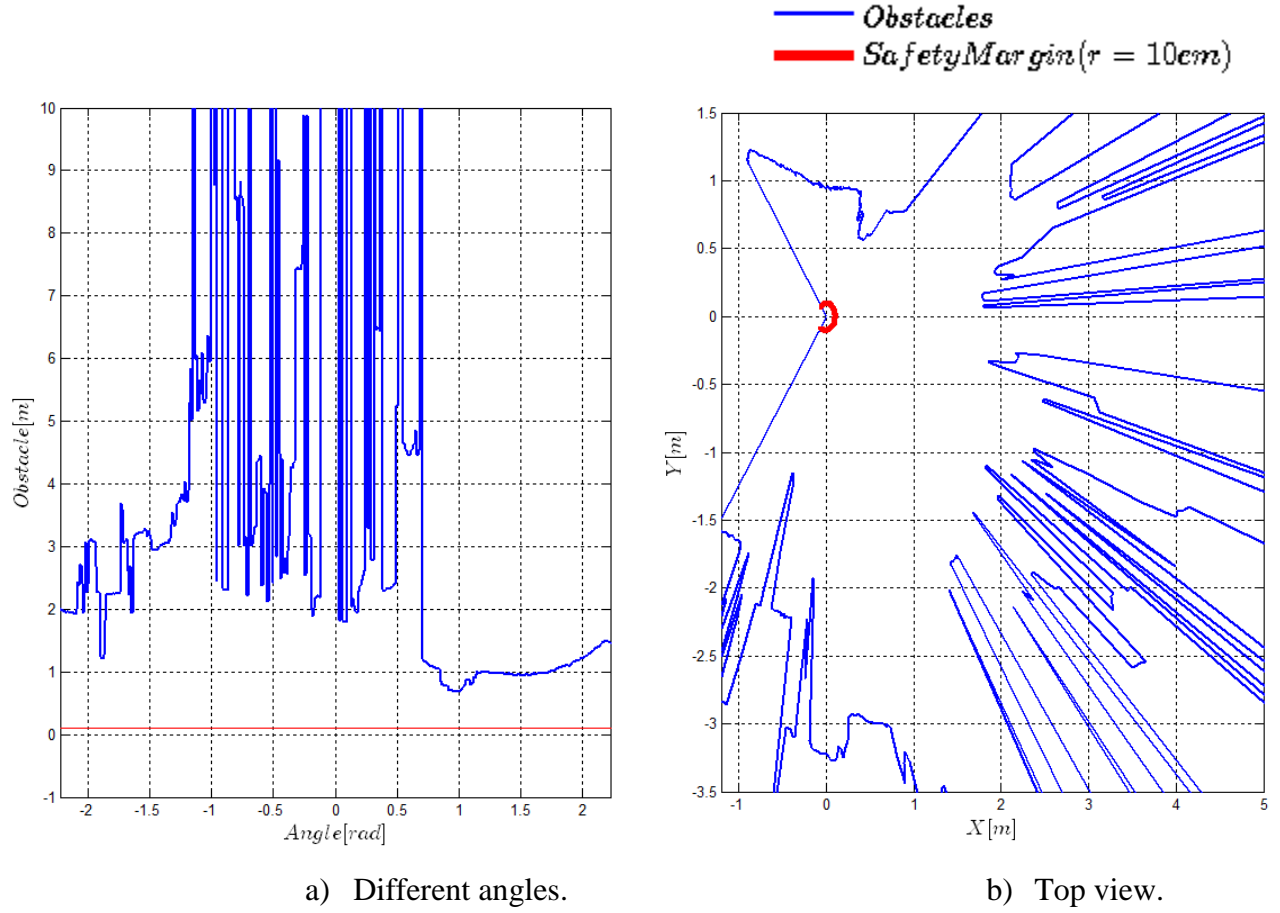
## 3. Analysis of Results

In this chapter, the obtained results from the autonomous docking of the Rob@work 3 with laser scanner and vision sensor are illustrated to compare the optimal behavior of each in detail. The obtained trajectory of 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 of docking the Rob@work 3 with the laser scanners are performed to evaluate the precision of the results. In each experiment, a plot of distance to the around objects with respect to the beam angle ( $\theta_b$ ) 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 maximum radial distance of 30 m (AG, 2016). However, figures are zoomed in to provide better visibility.

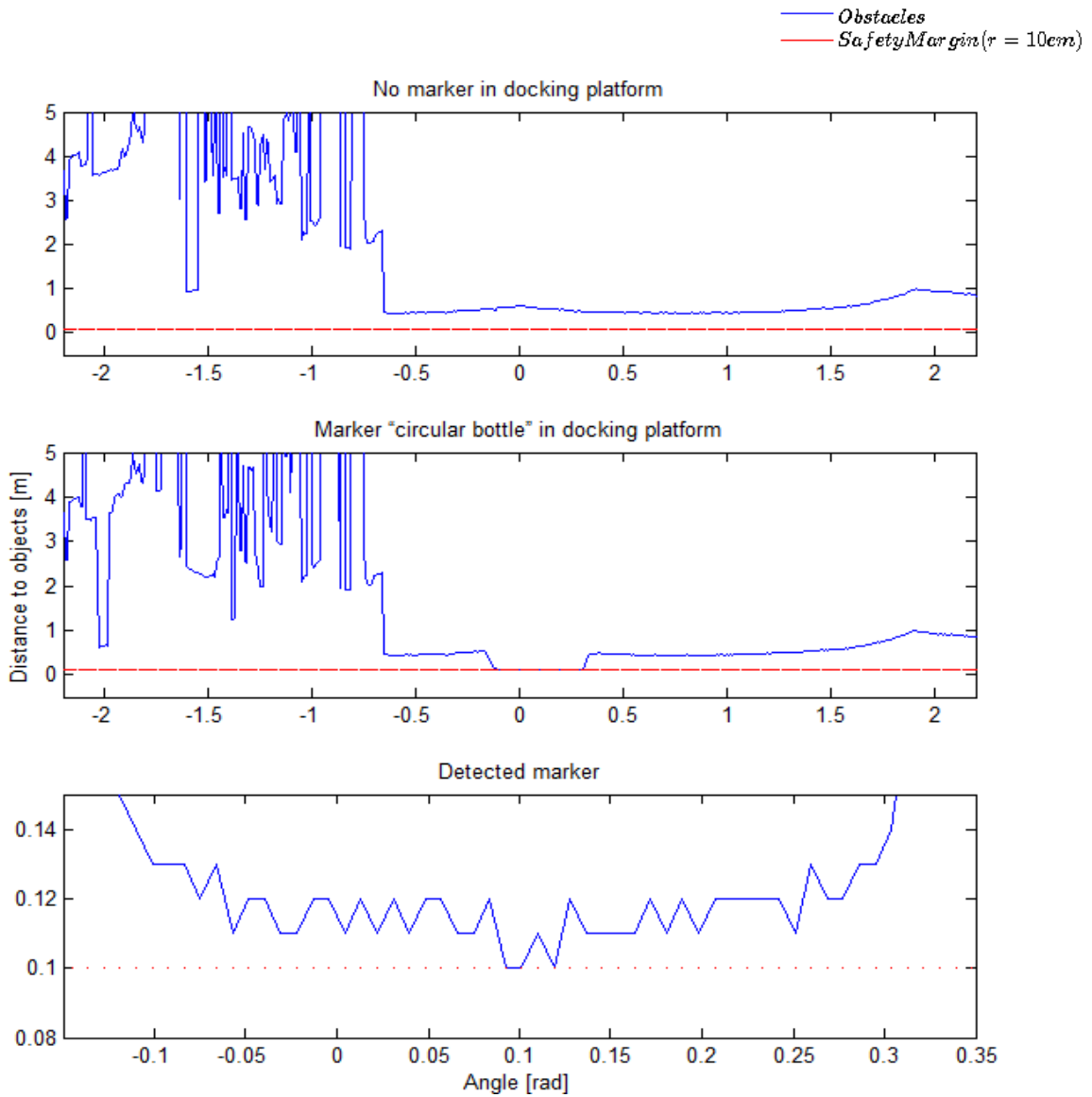


**Figure 18** Laser sensor analysis.

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  $[-135^\circ, 135^\circ]$  degrees. Angle is calculated from the index (i) and the angle increment ( $\theta_{inc}$ ) of the laser beams as derived in (2) and (3). The red color corresponds to safety margin distance defined for 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 if it is seen from top view, as depicted in Figure 18 b). According to Figure 18 a), any object with distance less or equal than 10 cm to the laser sensor is considered as the 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 sensor is attached to the cross sides of the robot.

In initial case in which 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 observed in Figure 18. Therefore, docking area is found clean from any physical obstacle within the distance of 10 cm while no marker is attached to target, e.g., docking platform.

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 a detection accuracy of laser scanner when no marker exists, whereas second scenario, Figure 19 b), a transparent cylindrical bottle placed on the docking platform used as a detectable marker.



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

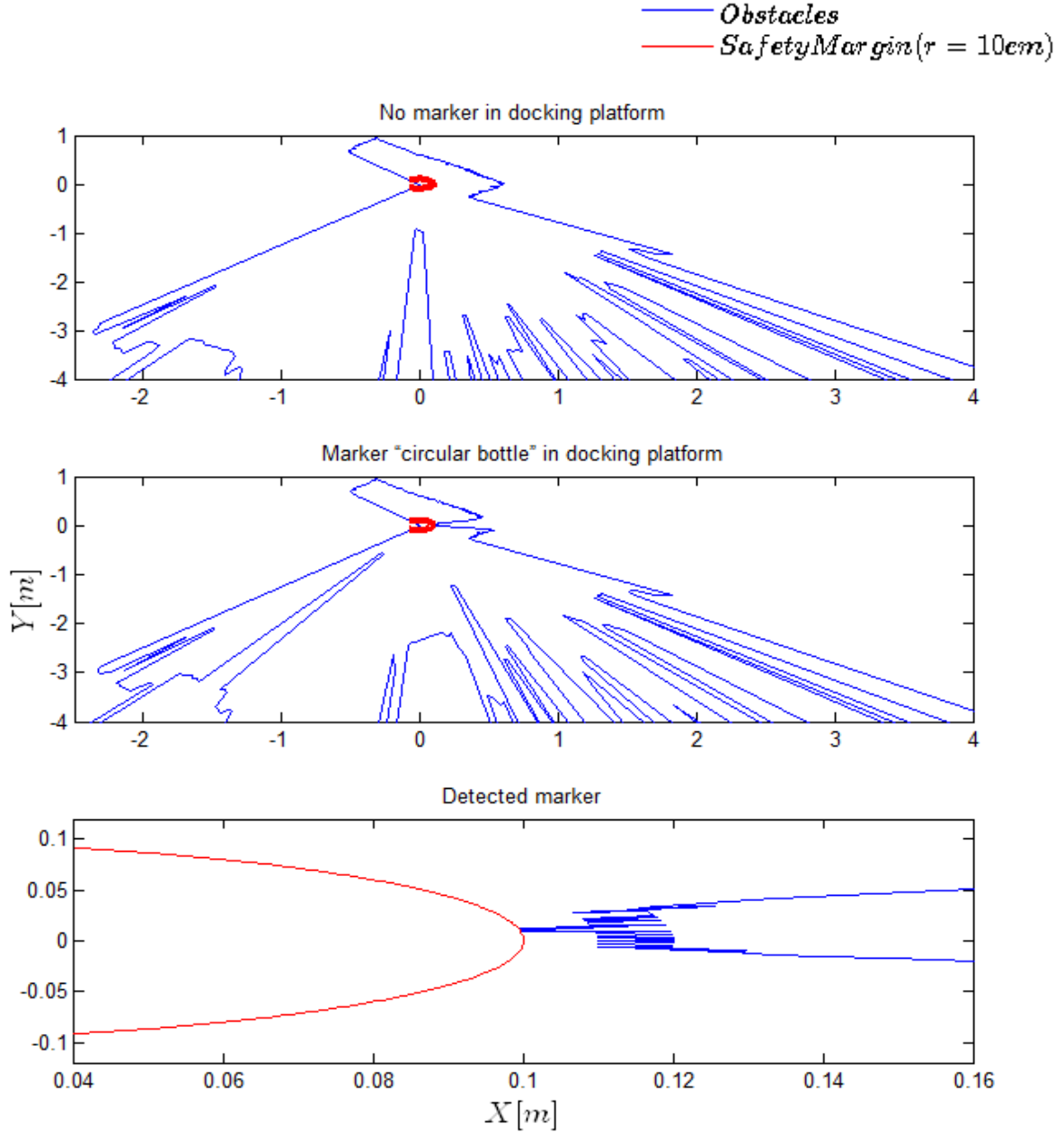
As mentioned earlier, the safety margin is defined as the distance to the obstacle which is 10 cm that laser scanner can detect. This experiment conducted with the docked robot to identify the reference, e.g., the marker. All the laser beams measure the distance to surrounding objects in order to identify the marker in the docking platform.

Since 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 (AG, 2016). However, there are few samples with same distance to the assigned obstacle in different angle 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 declares some inconsistencies between 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 whole experiment. The marker is only added to the docking platform. This is declared as another inconsistency in the docking of Rob@work 3 with the laser scanner.

The laser scanner data of the same experiments is also plotted from the top view to illustrate the perception 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.

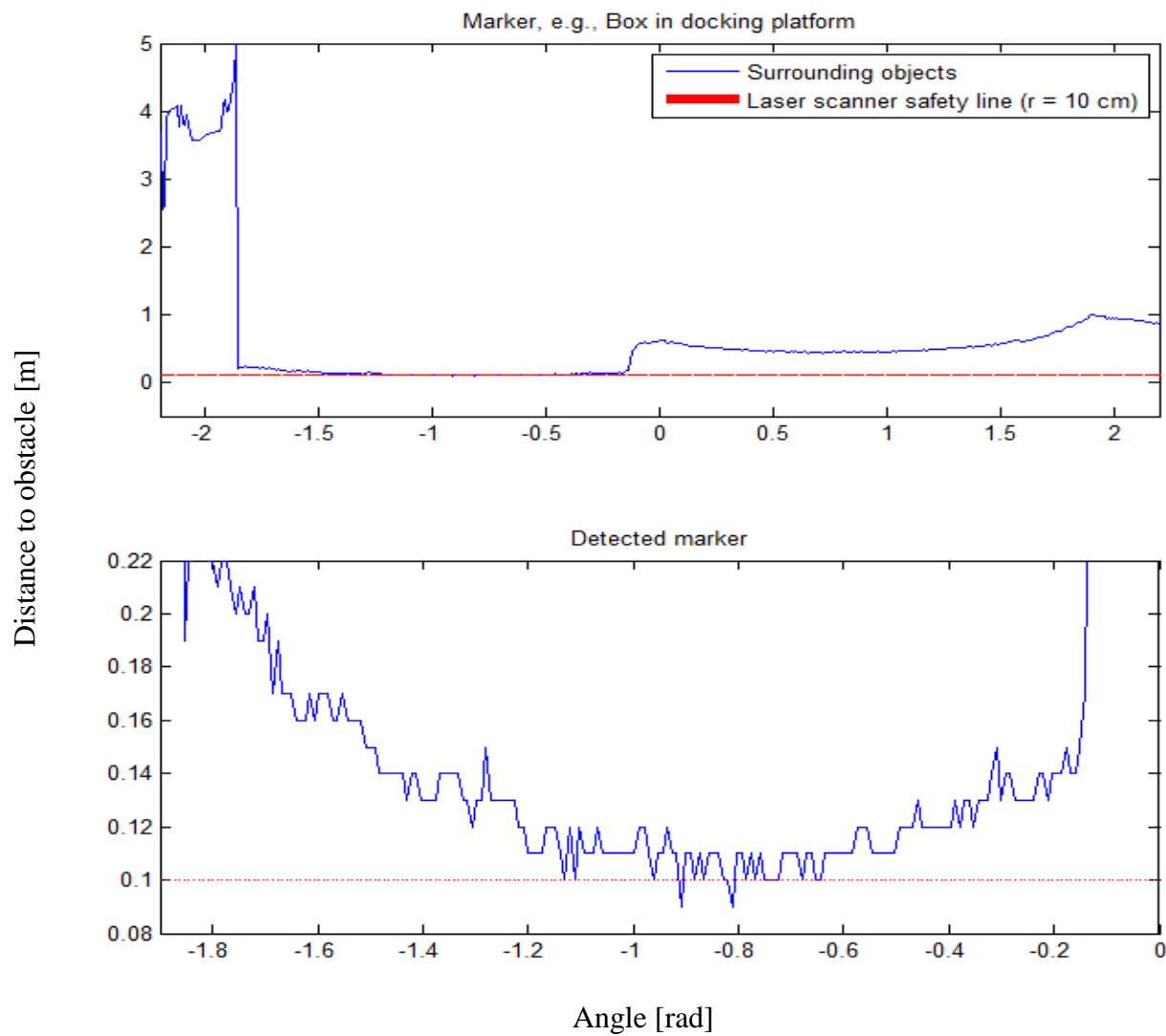


**Figure 20** Laser scanner analysis for detected marker while the Rob@work 3 is docked, top view.

The first trial is performed to make sure that there is no obstacle close to the area with the radial distance of 10 cm. Therefore, the marker is not placed in target area and no obstacle detected by the laser scanner within that distance. Therefore, the area close to the laser sensor is clean and no intersection is recorded accordingly. The second trial employs the marker placed to the radial distance less than or equal to 10 cm. In this experiment the safety curve is crossed and the marker is detected by the sensor successfully.

However, the detected marker has crossed the safety curve in different positions along the Y-axis when the robot is manually docked. It illustrates the inaccuracy of the sensor in the marker detection since few samples detected with the same distance equals to 10 cm. Therefore, if the Rob@work 3 performs the docking only with laser scanner, the inconsistencies of the marker detection do not let the mobile robot accomplish the docking task precisely.

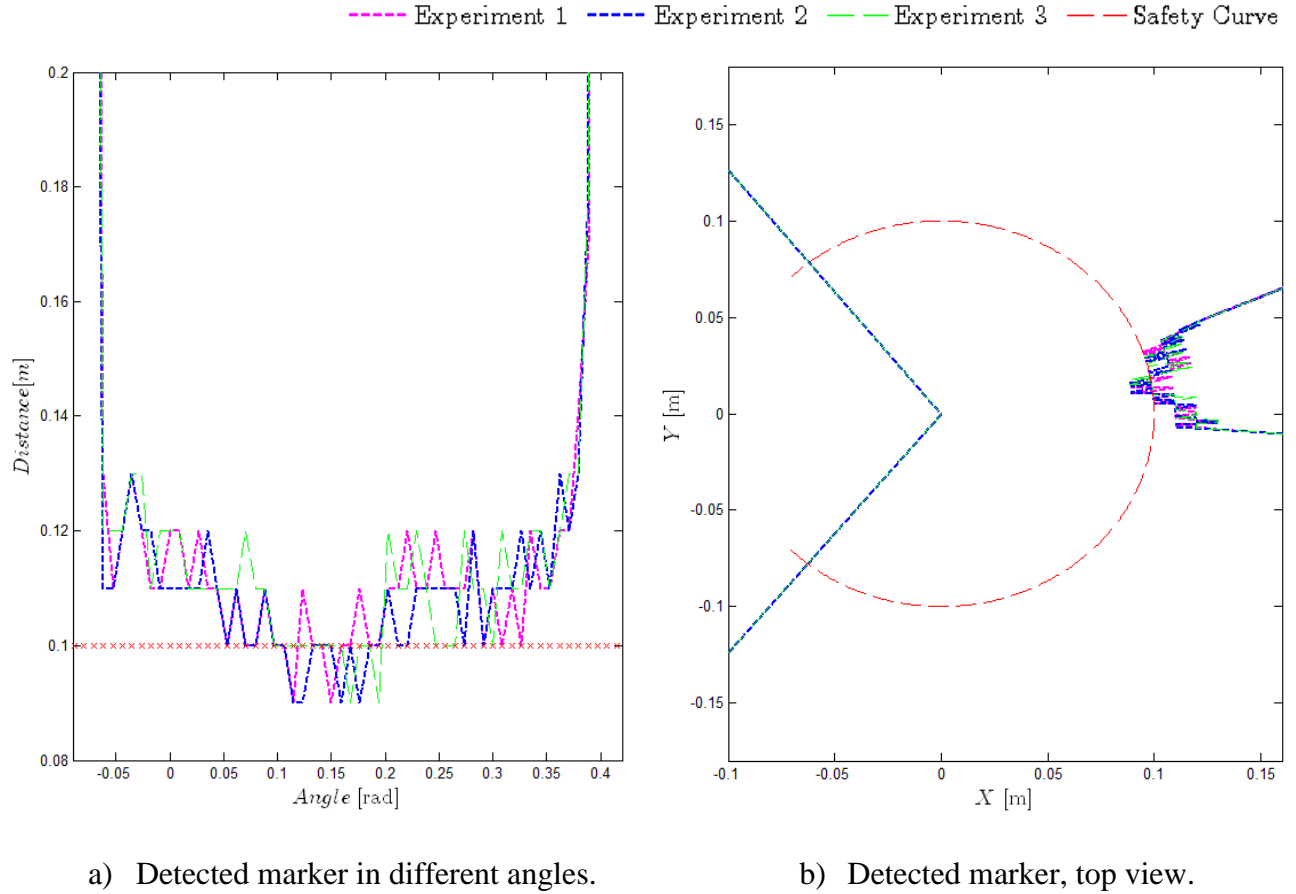
In the third experiment, a new marker is utilized to evaluate inaccuracy of laser scanner in marker detection. The new marker is a  $30 \times 23.5 \times 21 \text{ cm}^3$  box, located at the same place of the cylindrical marker. Figure 21 illustrates scanner analysis for this configuration.



**Figure 21** Scanner analysis with box in docking platform.

Since the length of the box is bigger than the radius of the cylinder, the detection area occupies more laser beams. The marker is detected with 1-2 cm accuracy even though Figure 21 illustrates plenty of intersections with the safety line. It practically implies that the marker detected in several angles and positions. The laser scanner is more inaccurate in this experiment with the box than the circular bottle and the Rob@work 3 cannot achieve the docking task with such 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 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.



**Figure 22** Three conducted experiments for same configuration.

According to Figure 22, there are few samples which are detected as the obstacles in each experiment in the first place. Other experiments, however, have other few samples which are detected as marker in different angles even though the position of robot neither marker changed

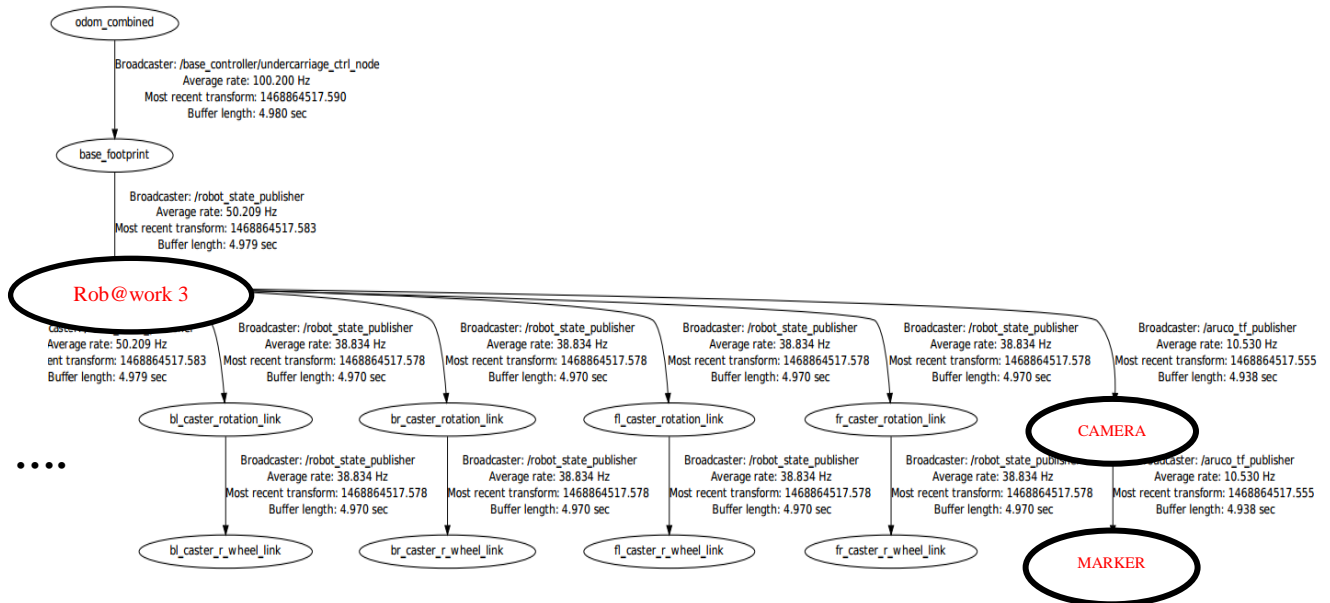
during the process. Therefore, laser scanner does not return reliable feedback each time from a stable marker placed in the docking platform.

## 3.2. Computer Vision

### 3.2.1. Camera and marker frame

The pose estimation demands different coordinate frames to work simultaneously and publish the position and the orientation of one frame with respect to others in real time. In order to get the privilege of the vision sensor in the docking, an extra frame for the camera should be added to the robot frame. The ROS transform package (tf) broadcasts a frame in real time with respect to its parent. After the coordinate submission, the 2D Rotation matrix is employed to map the camera frame to the fixed marker frame.

The physical setup of camera is such that it is mounted at the front side and approximately in the middle of the Rob@work 3 platform. Therefore, the target should be detected approximately in the center of the marker coordinate system. Figure 23 illustrates 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 to its platform.



**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 the OpenCV for the detection of the planar fiducial marker (OpenCV, 2015).

### 3.2.2. Camera Calibration

In this project, the OpenCV algorithm is employed to calibrate the mounted camera on the Rob@work 3. The purpose of the calibration is to find the parameters of the intrinsic and the extrinsic matrices (section 6.1). The provided input is a classical black-white chessboard with  $6 \times 9$  squares for the calibration. The algorithm basically captures 25 images with chessboard 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) Calibrating process.

c) Calibrated image.

**Figure 24** Camera calibration process.

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

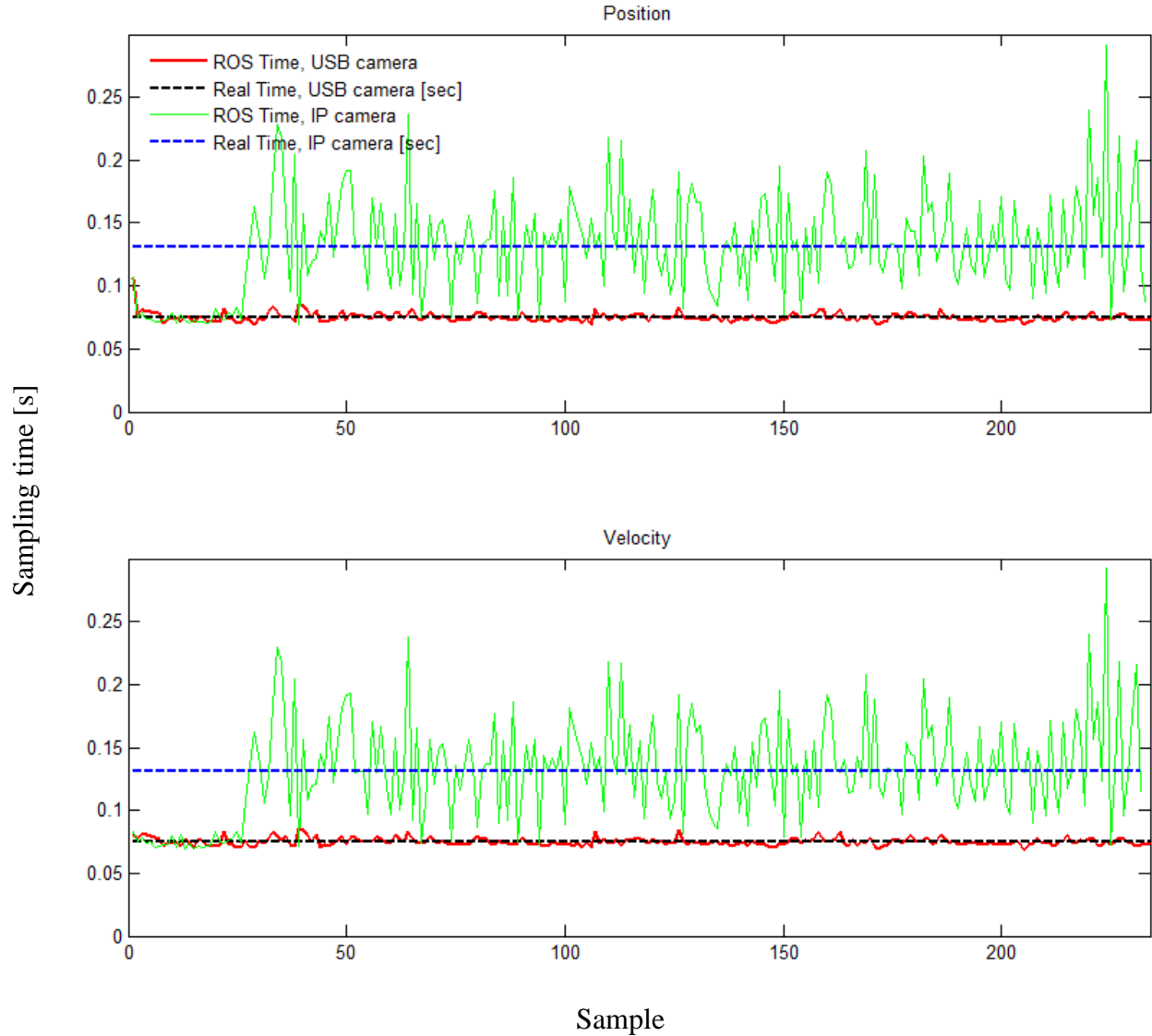
**Table 3** Configuration file for the camera calibration.

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 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 of docking the mobile robot are presented. The vision sensor employed to provide the feedback for the pose estimation of the camera with respect to the fixed marker on the docking platform. The intention of using the vision sensor is to increase the accuracy of the docking when the robot is placed in the different configurations. Figure 25 depicts the comparison between the recorded ROS time and the real time of the two different vision sensors employed, e.g., the USB and IP camera.

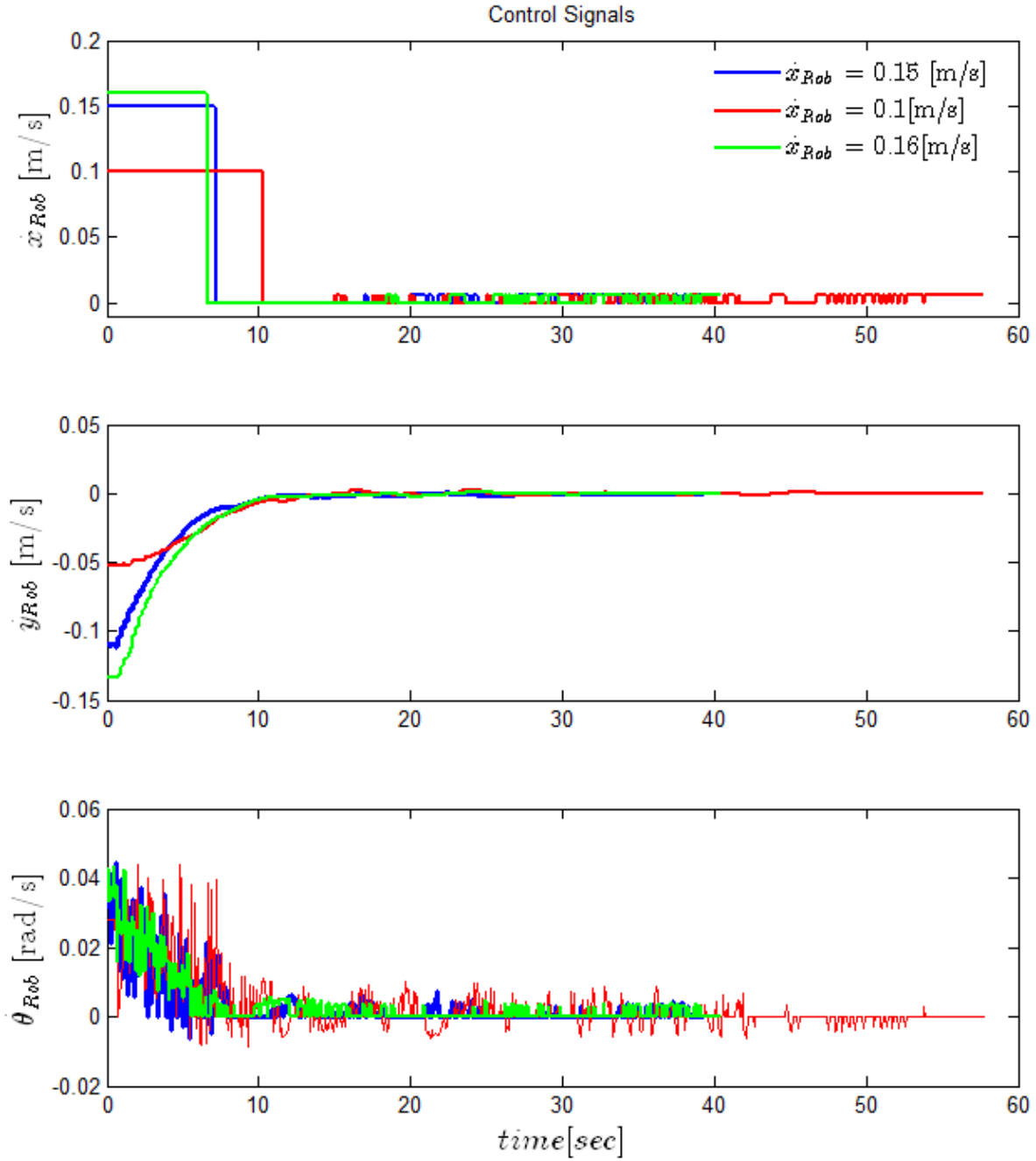


**Figure 25** Sampling time comparison of different vision sensors.

The ROS time, depicted in Figure 25, is the extracted data of the pose estimation matrix. 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 node receives no more messages.

After the real time computation of the given tasks, several experiments are conducted to compare different controllers to find the optimal responses to accomplish the docking with high precision. As it was stated in 2.4.1, moving along the  $x_{\text{mar}}$ -axis, e.g., toward docking platform is implemented by applying the constant velocities in the approach and the SM zones. Figure 26

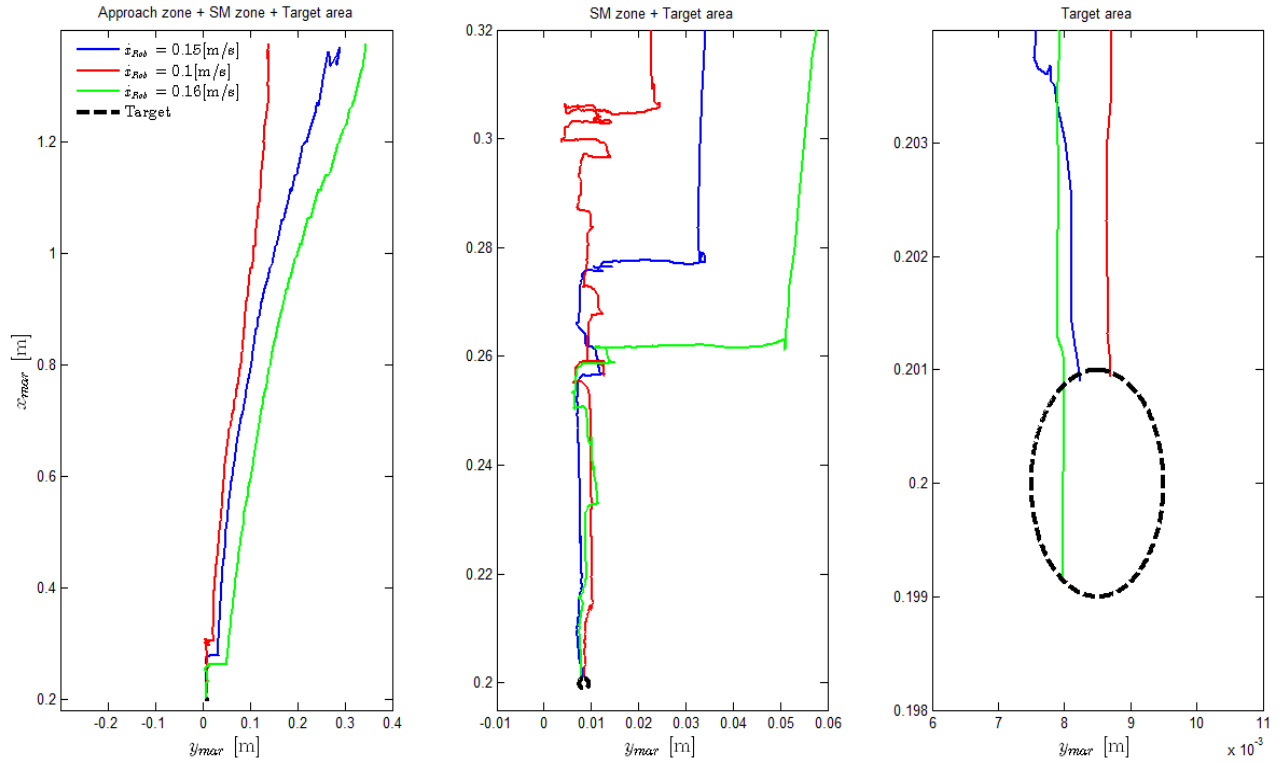
depicts three experiments which graphically compare the effect of the different constant velocities along the  $x_{Rob}$ -axis to achieve less average docking time and optimal trajectory. All three experiments are conducted using a simple P-controller for velocities along the  $y_{Rob}$ -axis and the  $\theta_{Rob}$ -axis.



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

The corresponding trajectory of the above experiments is plotted in Figure 27 which depicts the whole area of the docking tasks, the SM zone, and the target which is sketched as the circle with the origin of reference value of the marker ( $x_o = 0.2 \text{ m}, y_o = 0.0085 \text{ m}$ ) and the radius of the threshold along the  $x_{\text{mar}}$ -axis ( $x_{\text{thresh}} = 1 \text{ mm}$ ).

According to Figure 26, it takes more time for the Rob@work 3 to approach the SM zone (roughly 10 seconds) if less forward velocity ( $\dot{x}_{\text{Rob}}$ ) is applied. In this case, approach time is increased and the robot is slower in docking. However, Figure 27 declares less offset along the  $y_{\text{mar}}$ -axis which is approximately 2.5 cm with slower motion along the  $x_{\text{mar}}$ -axis and more optimized trajectory.



**Figure 27** Docking trajectory.

Therefore, the forward velocity is chosen such that the approach time and the offset along the  $y_{\text{mar}}$ -axis compromise between the optimal trajectory and less average docking time. Therefore,  $\dot{x}_{\text{Rob}} = 0.15 \frac{\text{m}}{\text{sec}}$  is selected as the forward velocity for the further control design. This velocity command, however, is just applied when the robot is inside the approach zone. In final zone in

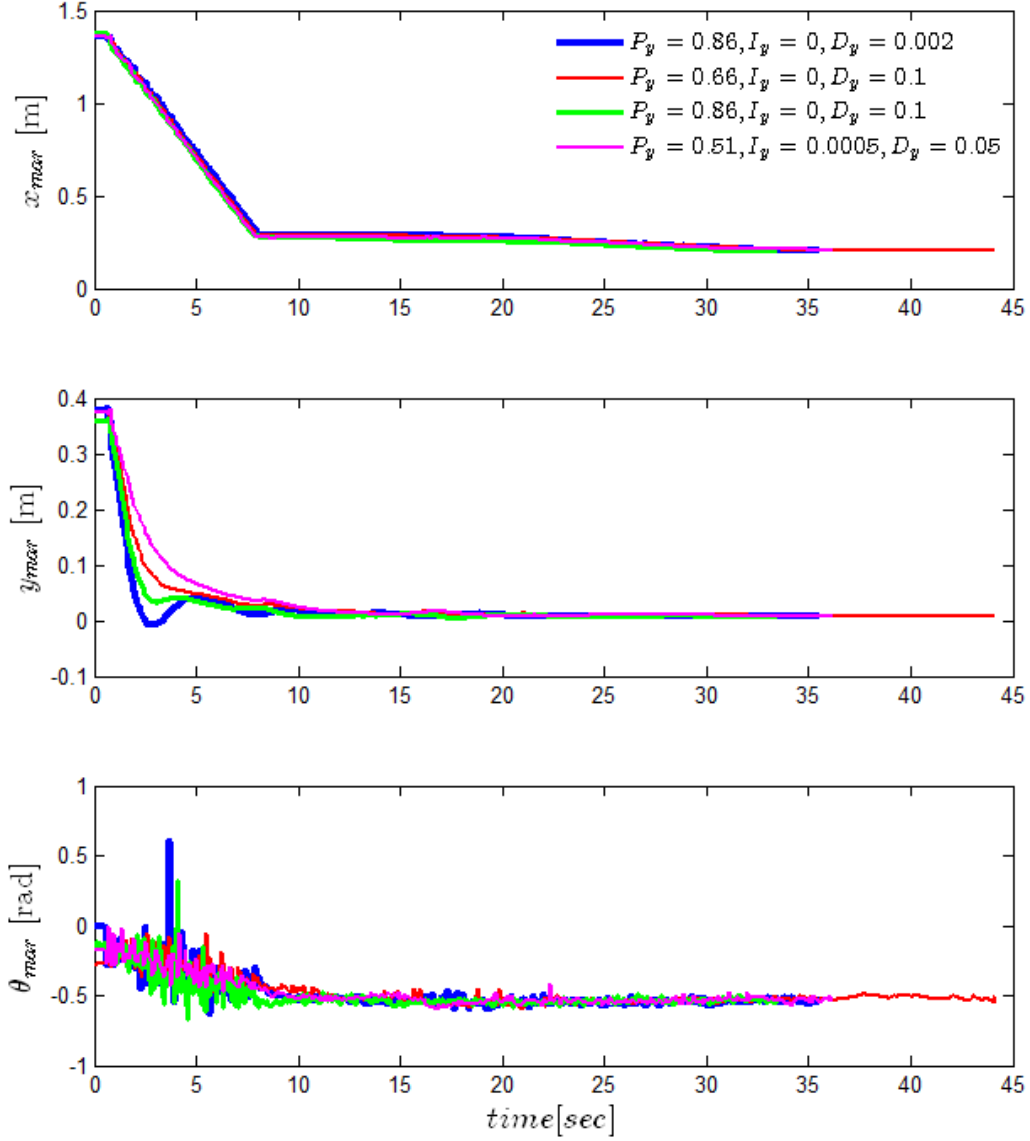
which movement along the  $y_{\text{mar}}$ -axis and the  $\theta_{\text{mar}}$ -axis adjusted, very small constant velocity, e.g.,  $\dot{x}_{\text{Rob}} = 6 \frac{\text{mm}}{\text{sec}}$  is applied to move the robot toward the docking platform.

After choosing the appropriate constant velocity for the approach and the SM zones along the  $x_{\text{Rob}}$ -axis, another experiment with different gains conducted to compare the effect of gain tuning to obtain optimal behavior in the control design. Overall, six gains are tuned for two controllers along the  $y_{\text{Rob}}$ -axis and the  $\theta_{\text{Rob}}$ -axis. However, the  $y_{\text{Rob}}$ -axis is prioritized since the difference in the orientation is far less than the position. Therefore, for this experiment, gains of the controllers are listed in Table 4 to adjust the position and the orientation.

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

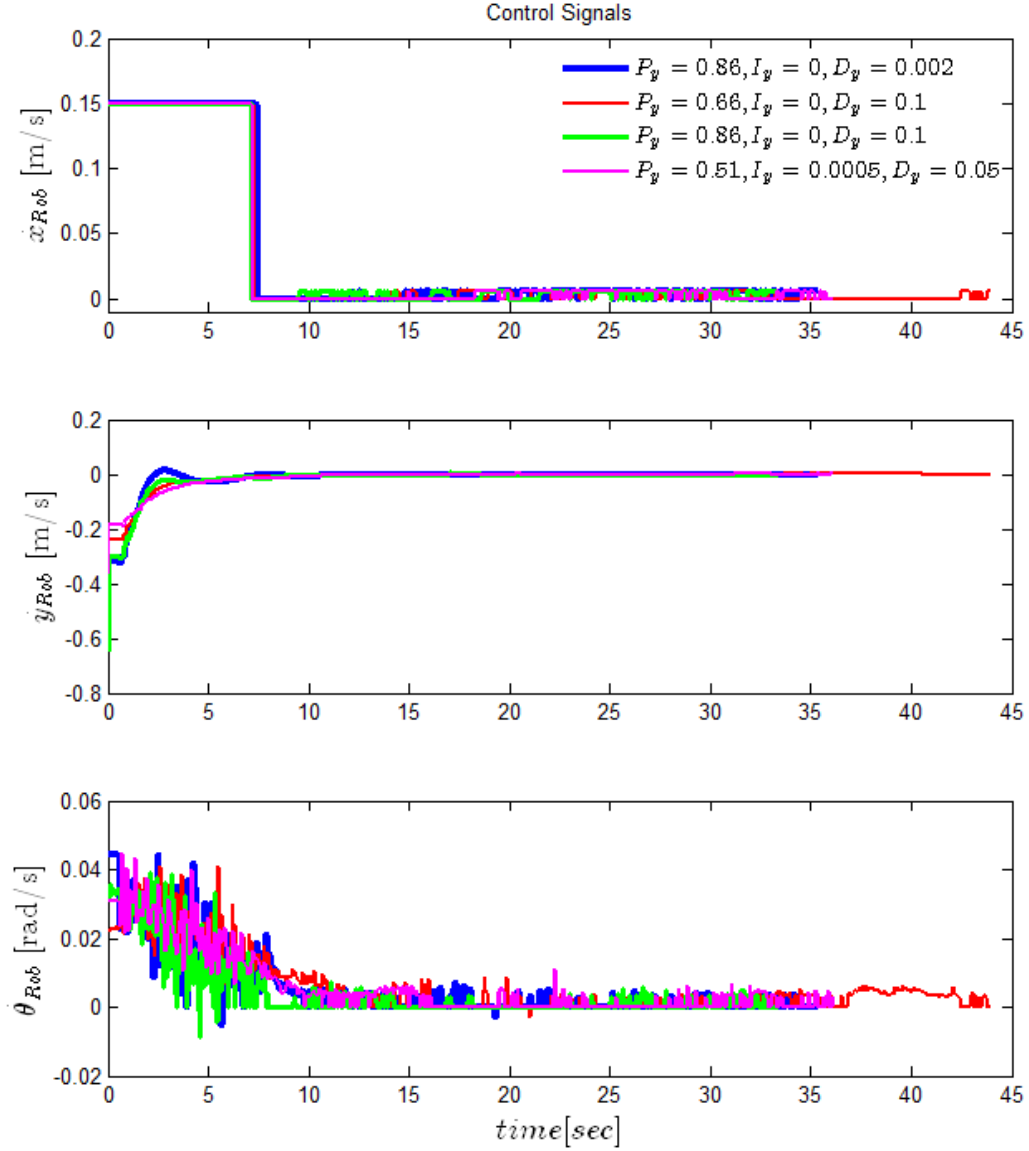
	Exp.	1	2	3	4
	Gains				
Position ( $y_{\text{rob}}$ -axis)	$K_P$	0.86	0.66	0.86	0.51
	$K_I$	0	0	0	0.0005
	$K_D$	0.002	0.1	0.1	0.05
Orientation ( $\theta_{\text{rob}}$ -axis)	$K_P$	0.08	0.08	0.08	0.08
	$K_I$	0	0	0	0
	$K_D$	0	0	0	0

In the control design, the pose estimation of the camera with respect to the fixed marker leads to computation of the measurement errors and eventually control signals along the  $y_{\text{mar}}$ -axis and  $\theta_{\text{mar}}$ -axis to adjust the position and the orientation of the robot respectively. Figure 28 represents camera pose estimation with 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 equations 11 and 12 even though real experiments conducted with the ROS time.



**Figure 28** Camera pose estimation in marker coordinate system.

Theoretically, the slope of the position versus time plot reveals pertinent information about the velocity of the mobile robots. According to Figure 28, the position of the robot along the  $x_{mar}$ -axis is linear in all the experiments. This indicates that the velocity of the robots should be constant along the  $x_{Rob}$ -axis which can be observed in Figure 29, e.g.,  $\dot{x}_{Rob} = 0.15 \frac{m}{sec}$ . Therefore, the vision feedback control is not developed along the  $x_{mar}$ -axis.



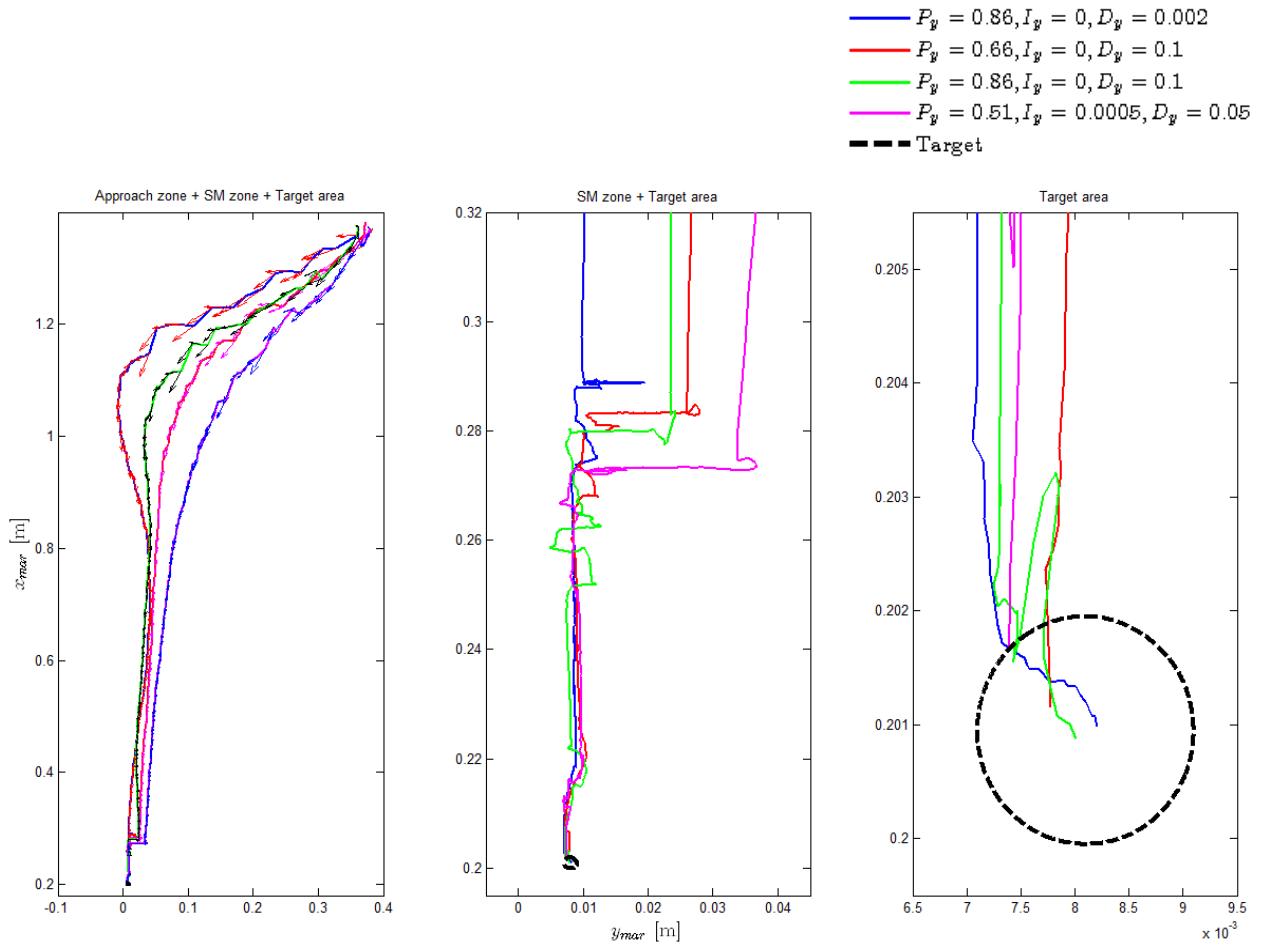
**Figure 29** Control signals along the axes of the Rob@work 3.

On the contrary, the  $y_{mar}$ -axis and the  $\theta_{mar}$ -axis have the visual feedback to compute the control signals along the  $y_{Rob}$ -axis and the  $\theta_{Rob}$ -axis simultaneously as depicted Figure 29. In each case, the optimality of the average docking time is evaluated with different PID gains. According to Figure 29, the slowest controller takes almost 44 sec. to accomplish the docking completely considering the 7.5-sec. for transferring to the SM zone which is quite fast and reasonable according to limited space provided in the lab for docking area.



The controllers in the SM zone, however, are the main reason that docking takes longer time to be finished. This zone is made to increase the accuracy measurement along the  $y_{\text{mar}}$ -axis and the  $\theta_{\text{mar}}$ -axis and smoothness of the docking while there is no forward movement along  $x_{\text{mar}}$ -axis, e.g.,  $\dot{x}_{\text{Rob}}^{\text{SM}} = 0$ . As soon as the measurement errors for the position along the  $y_{\text{mar}}$ -axis and the orientation along the  $\theta_{\text{mar}}$ -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 1 mm threshold along  $x_{\text{mar}}$ -axis and if the robot fulfills such requirement, the process is successfully completed and the program terminates.

The corresponding trajectories of the Rob@work 3 docking with different controllers are presented in Figure 30. The docking trajectory is practically the  $y_{\text{mar}}$ - $x_{\text{mar}}$  plot which illuminates movement toward the docking platform.



**Figure 30** Docking trajectories for different vision-feedback controllers.

Figure 30 depicts the docking trajectories of the different designed controller for the entire area, the SM zone, and the target area. The SM zone clarifies the obtained offsets along  $y_{\text{mar}}$ -axis for different controllers in the conducted experiment, whereas the target area represents the circle with the reference origin of the marker recorded once when the Rob@work 3 was manually docked ( $x_{\text{mar}}^{\text{Ref}}, y_{\text{mar}}^{\text{Ref}}, \theta_{\text{mar}}^{\text{Ref}}$ ) and a radius of threshold along  $x_{\text{mar}}$ -axis ( $x_{\text{thresh}} = 1 \text{ mm}$ ). The orientations of different controllers are also addressed with arrows on each trajectory while moving in the  $y_{\text{mar}}$ - $x_{\text{mar}}$  plot.

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

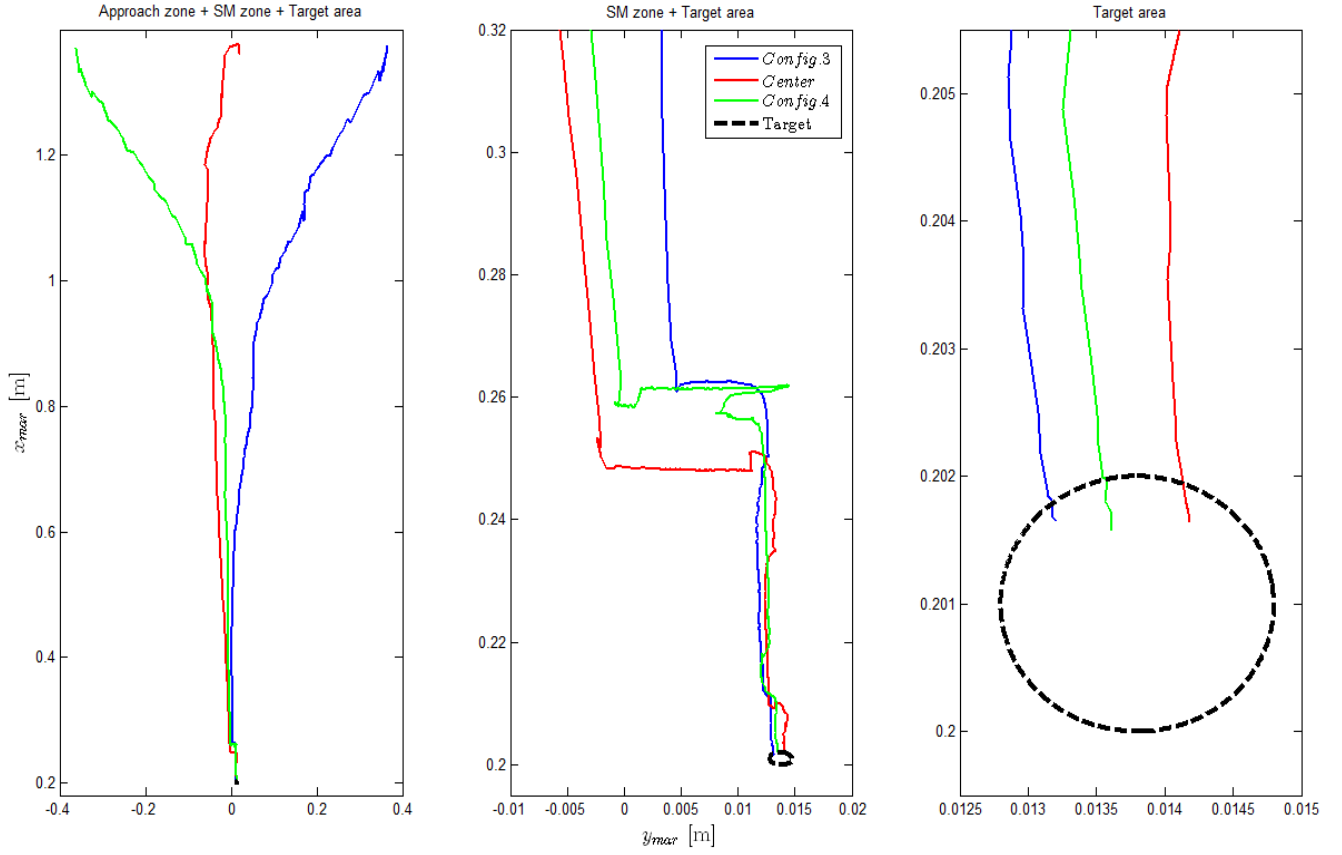
**Table 5** Comparison of the different controllers in the vision-based docking of the Rob@work 3.

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

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

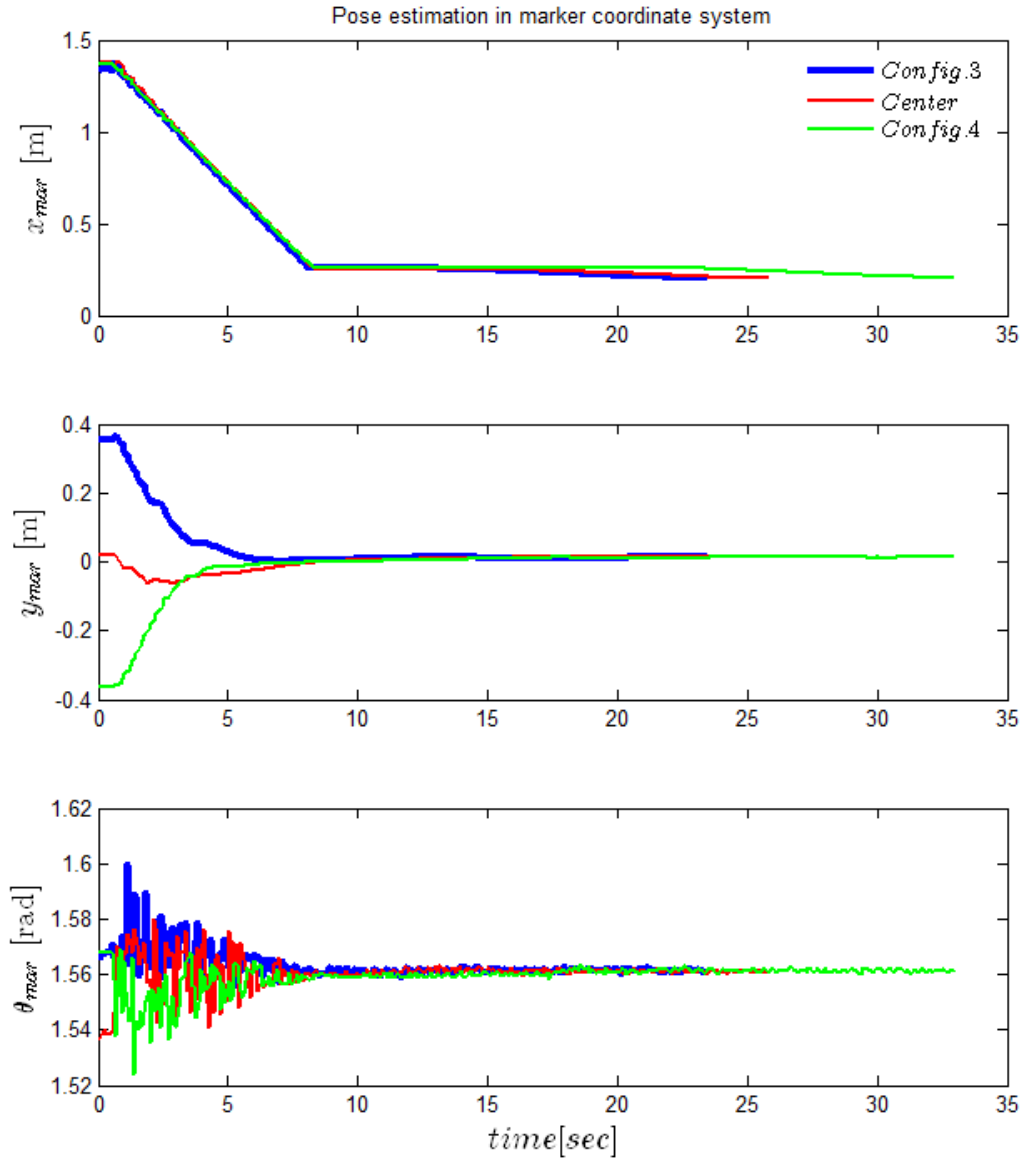
To test the designed controller of the conducted experiment in Table 5, the robot is placed in the three different configurations in the docking area to evaluate the efficiency. In this experiment,

the extreme orientations are selected for the robot, e.g., 3 and 4 in Figure 7 to test the chosen controller.



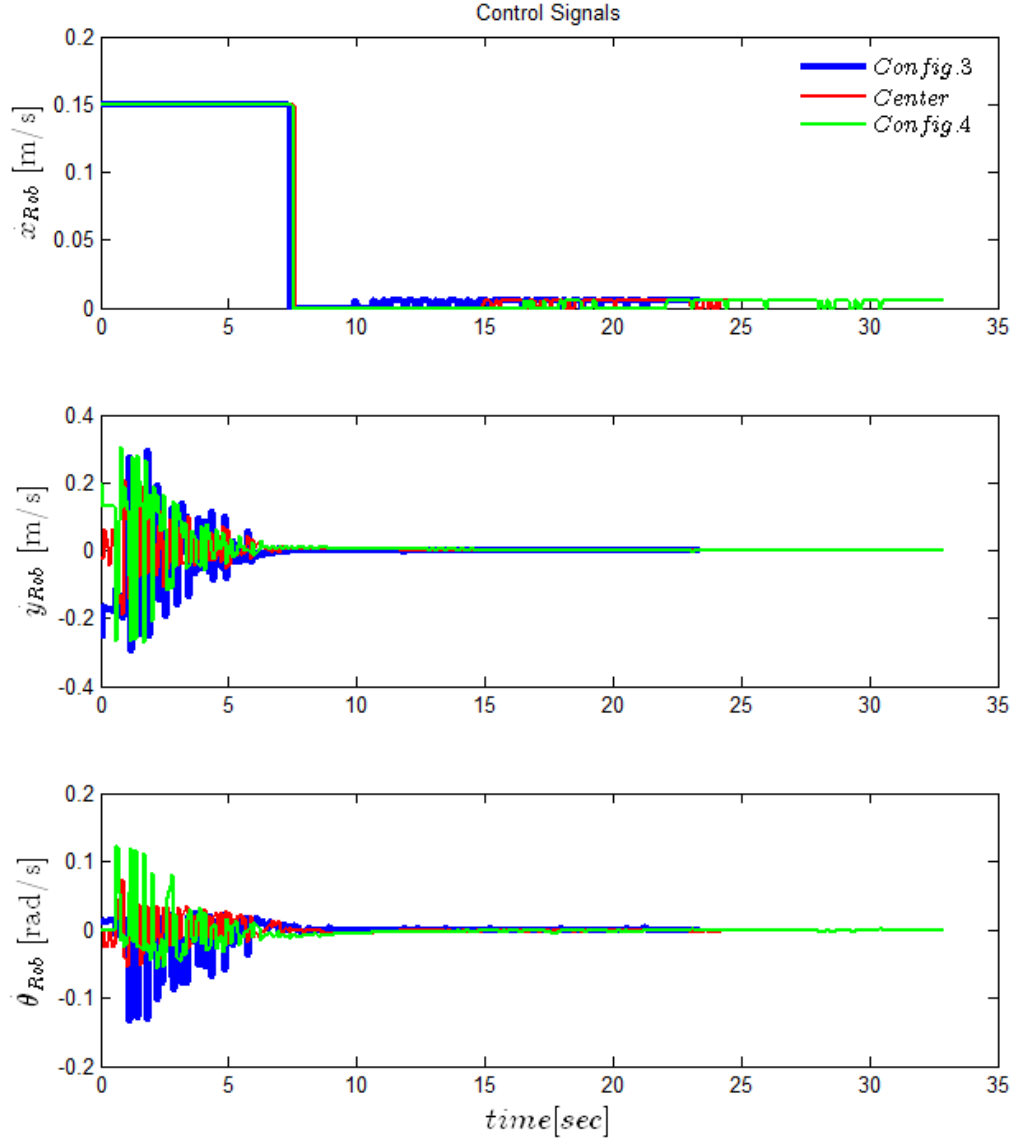
**Figure 31** Obtained trajectory of the designed controller for three different configurations.

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



**Figure 32** Camera pose estimation in marker coordinate system.

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



**Figure 33** Control signals for the Rob@work 3 along the axes of the Rob@work 3.

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

### 3.4. Reinforcement Learning

As stated in 2.5.3, the idea of using the model-free RL technique is to develop the optimal action selection 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 an obtained shortest path toward the goal.

The time step is practically the number of the rows or the columns in the Q-matrix.

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

**Table 6** Reward distribution for the Q-Learning.

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

The rows and columns of grid are the recorded orientation and position of the camera, attached to front side of robot ( $\theta_{CAM}, y_{CAM}$ ) respectively which are real numbers, whereas the indexes of the Q-matrix should be integers. Therefore, following function is proposed 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 grid segment and the sample rate (SR), e.g., number of rows or columns which is calculated as follow:

$$\begin{cases} \theta_{\text{seg}} = \frac{\theta_{\text{up}} - \theta_{\text{dwn}}}{\theta_{\text{SR}}} \\ y_{\text{seg}} = \frac{y_{\text{up}} - y_{\text{dwn}}}{y_{\text{SR}}} \end{cases} \quad (18)$$

The pseudo-code of the developed algorithm is presented below:

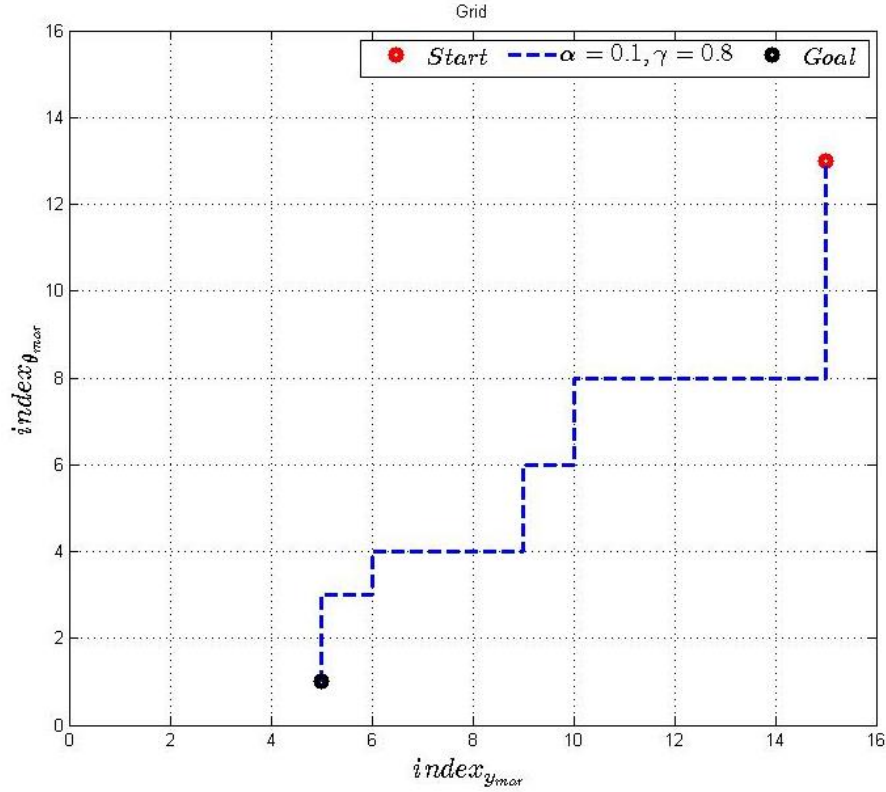
```
void i_j_Generator(double y, double theta)
{
    int sample_rate_y = col - 1;
    int sample_rate_theta = row - 1;
    double div_theta = (theta_up - theta_dwn)/sample_rate_theta;
    double div_y = (y_up - y_dwn)/sample_rate_y;

    // i selection
    for (int k = 0; k <= (row - 1); k++)
    {
        if (abs(theta) >= (theta_dwn - ((.5 + k) * div_theta)) &&
            abs(theta) <= (theta_dwn + ((.5 + k) * div_theta)))
        {
            i = k;
            break;
        }
        else if (abs(theta) < (theta_dwn - ((.5 + 0) * div_theta)) ||
            abs(theta) > (theta_dwn + ((.5 + (row - 1)) *
div_theta)))
        {
            ROS_INFO (" Outside theta - boundary => NEGATIVE REWARD!!");
        }
    }

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

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

The Q-Learning configuration demands the whole components, including target, to be represented with the corresponding integer indexes. Therefore, the goal should be identified state which can be recorded as the robot is docked manually ( $i_{\text{Goal}} = 1, j_{\text{Goal}} = 4$ ). Figure 34 illustrates the trained path towards the goal with the integer indexes for the orientation and the position ( $\theta_{\text{CAM}}, y_{\text{CAM}}$ ) 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.

Figure 34 depicts learning process of 1050 iterations to update the Q-matrix and obtain optimal action policy, e.g., 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 optimal path toward, considering the fact that 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 declared as the mobile robot is docked manually. Furthermore, the goal represents the



convergence of the Q-Learning to optimal Q-function with the learning rate ( $\alpha = 0.1$ ) and the discount factor ( $\gamma = 0.8$ ) while the single entry is updated on each step.

Although Figure 34 is not graphically compared with another path to be judged as the optimal policy in docking, the Q-matrix behind this grid provides evaluation with the neighbor components in which the current position of the robot with the recognized indexes are constantly evaluated with their neighbor values to obtain the shortest path toward the goal. In this approach, initial position could be chosen randomly 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 final state, e.g., goal state to finish the docking in real experiment even though horizontal movement is not restricted in other states of the grid. This is due to mechanical constraint of the platform for aside movements, e.g.,  $y_{\text{mar}}$  to secure the smooth docking with end-effectors. Therefore, the setup demands forward movements towards the goal when 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 the mobile robots demands accurate sensor measurement from the docking platform. In this thesis, the feasibility of the sensor integration, i.e., the laser scanner and the vision sensors, investigated to obtain the optimal docking trajectory within the less average docking time.

According to the datasheet of the S300 Sick Safety scanners, the resolution of 1-3 cm is expected in detecting the obstacles (AG, 2016). According to the experiments performed with different shaped markers and 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 fusion of the laser scanner sensors to achieve the docking task with high precision. Therefore, vision sensor is added to increase the accuracy of the measurements with the designed feedback control.

In the vision approach, the measurements are integrated with pose estimation of the camera with respect to the fixed marker to design the vision-feedback control system to maximize the precision. The target is represented with the marker which is capable of publishing the orientation, i.e.,  $\theta_{\text{mar}}^{\text{Ref}}$  besides of the position, i.e.,  $x_{\text{mar}}^{\text{Ref}}$ ,  $y_{\text{mar}}^{\text{Ref}}$ .

The ARTags with the unique marker ID are prioritized to the simple point fiducials since they provide the orientation if the Rob@work 3 is placed in the different configurations compared to the 2D positions estimated 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 noises caused by wires and other surrounding objects with the less illumination. The raw orientation measurement and the vision control design along the  $\theta_{\text{mar}}$ -axis ensure the precise docking even if the mobile robot is not practically perpendicular to the target. ARTags depict higher transparency for detection from further distances in the environments with which challenging illuminations exist in the workspace.

In the control design, The PBVS is developed as the vision feedback 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 platform with the ROS transformation package (tf) which lets the user track the multiple coordinate systems in the 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 ( $x_{\text{CAM}}, y_{\text{CAM}}, \theta_{\text{CAM}}$ ).

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

Although the ROS is quite fast for the online task scheduling, it does not always provide most deterministic timing of the certain tasks. According to Figure 25, the real time for both velocity and position sampling time seems constant even though the ROS time illustrates jitter on both plots. Therefore, the ROS time does not seem practical for tasks with high frequency PID or motion control.

Moreover, sampling time for the USB camera illustrates 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 does not have enough bandwidth for

transmitting the MPEG-4 video stream with resolution of  $640 \times 480$  and 30 fps. This indicates the reason for shaky behavior of the mobile robot when the IP camera is employed as the vision sensor in the docking of the Rob@work 3.

The Rob@work 3 is considered as the plant under investigation in the control system depicted in Figure 13 to design an appropriate controller for docking task. However, due to the sophisticated computations of the under-carriage control, the plant is linearized to the first order integrator to simplify the computation. The control signals are the velocity commands applied to the Rob@work 3 to move toward the docking platform. The commands are practically the linear velocities along the  $x_{\text{Rob}}$ -axis, the  $y_{\text{Rob}}$ -axis, and the angular velocity along the  $\theta_{\text{Rob}}$ -axis.

The movement along  $x_{\text{rob}}$ -axis does not require controller since the Rob@work 3 can move toward the docking platform with the constant velocity. However, the  $y_{\text{Rob}}$ -axis and  $\theta_{\text{Rob}}$ -axis require the vision-feedback controllers to achieve the target accurately. According to Figure 26, the constant velocities along  $x_{\text{Rob}}$ -axis on the different zones have direct impact on the obtained trajectory within the less average docking time. Although, the slower movement of the Rob@work 3 along the  $x_{\text{Rob}}$ -axis in the approach zone may decrease the offset along  $y_{\text{mar}}$ -axis to get as close as the reference in the SM zone, it would considerably increase the docking time.

Perhaps the biggest challenge to obtain the optimal action selection policy in the model-free approach of the RL framework is the maximum convergence on the real experiments in which constraints diminish the desired outcome. For this reason, the simulation is alternatively employed for the learning system in which the number of iterations is crucially important to achieve the maximum convergence and the helpful behavior with the less failure.

On the one hand, the larger number of iterations may increase the chance of better convergence but due to the limited power resources, the operating time and the mechanical setups of the Rob@work 3 large iterations may not be feasible in real experiments. That makes the iterations above hundreds infeasible in the real experiment 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 while is rather developed for the ideal yet simple goal achievement experiment.

Another challenge to utilize the Q-Learning is the pose estimation inside the virtual grid which is represented with indexes  $(i,j)$  after the proposed formula used for conversion. Smaller grid is not accurate enough to determine exact position of the robot compared to the vision-feedback control approach. It can be improved with the larger grid since it increases the accuracy of the acquired camera position on the robot in 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 adjustments with a very precise motion along its collinear joint axis to accomplish the docking. For this purpose, multiple sensors are employed to evaluate the docking time and the obtained trajectory. Since the position accuracy is prioritized in this project, the already built-in laser scanner sensors are not sufficient to accomplish the task. Therefore, the vision sensor added to the Rob@work 3 to increase the accuracy with the designed vision-feedback control.

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

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

The computer vision algorithm in the OpenCV is used to calibrate the vision sensors to determine camera parameters and remove distortion from the image. The file containing the parameters and the marker size are employed to run the main visualization process. In the process, the position and the orientation of the camera are estimated with respect to the docking

platform as soon as the marker is found. The camera coordinate frame is published with respect to the platform of the robot with the ROS transformation package (Saito, 2015). However, the camera coordinates must be known with respect to the docking platform. Therefore, the transformation matrix is used to obtain the pose estimation of the camera with respect to the fixed marker.

In the vision-feedback control design, the planar fiducial, known as the ARTag, is prioritized over the point fiducials since the orientation can be also 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 further control design.

As an alternative to achieve the docking, the model-free Q-Learning approach investigated to compare the optimal docking behavior with the vision-feedback control system. The idea is to obtain the optimal action selection policy to observe the prior knowledge to achieve optimal trajectory within less time steps. The robot is rewarded in the goal state depending on quality of the docking.

The size of the developed grid in the Q-Learning method is critically important and has direct impact on the result accuracy even though it takes more computation time. Training the real robot demands the simplified grid and smaller Q-matrix, whereas the simulation restricts neither the grid size nor iterations.

The Q-Learning is developed in the simulation environment for some practical reasons. First, the mechanical constraints of the Rob@work 3 and the advanced model of the collinear joints with the tool changers on docking platform are not required in the simulation as the simplified model is employed. Second, training in the larger grid with higher iterations is feasible in the simulation environment. Next, the convergence of the Q-Learning is most likely to be obtained for the goal state. Finally, the limited power supply of the batteries on the Rob@work 3 is not sufficient to conduct the training experiments on the real platform.

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

feedback control system is more accurate and reliable in docking of the Rob@work 3 considering the criteria of the optimal docking with respect to time and trajectory.



## 6. Appendix

### 6.1. Camera Calibration

The pose estimation of the camera with respect to the fixed marker coordinate system is practically described by physical parameters such as the focal length of the lens, size of the pixel, the position of the principal point and the position and the orientation of the camera (Forsyth, David A.; Ponce, Jean;, 2012). 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 minimizes the deviations between the captured and the original 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 belong to the same shapes or lines on a single pattern and the results are represented below (Forsyth, David A.; Ponce, Jean;, 2012):

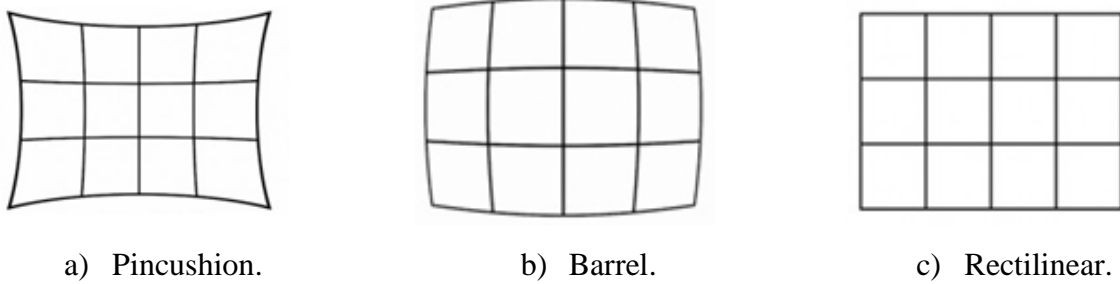
$$K_{3 \times 3} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix}; M_{3 \times 4} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (19)$$

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 of the camera with respect to the fixed marker coordinate system as illustrated in Figure 6.

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

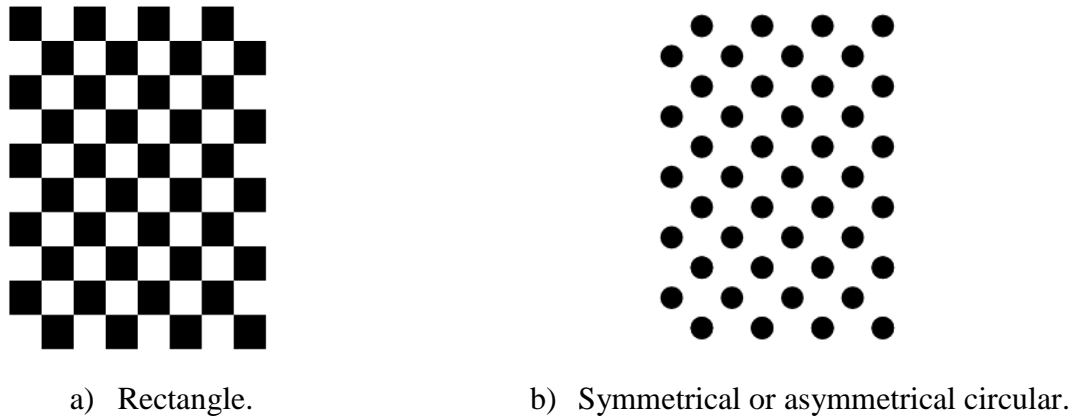
The lens distortion is categorized as the radial distortion and the tangential distortion.

Geometrically, the radial distortion is depicted as the 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, David A.; Ponce, Jean;, 2012). The calibrated image is expected to have a similar pattern to one of the patterns depicted in Figure 35.



**Figure 35** Different distortions in calibration process.

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



**Figure 36** Common patterns in camera calibration process.

The source code of the computer vision algorithm to calibrate the vision sensors under investigation with the particular pattern is borrowed from (OpenCV, 2011) and performed to

calibrate the android and the USB camera. The OpenCV algorithm takes the input of the camera, e.g., the video stream and detects the chessboard pattern. The algorithm is briefly explained below:

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

**Algorithm 4** Camera Calibration procedure with OpenCV.

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

## 6.2. Source Code of Project

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

## 7. Bibliography

AG, S., 2016. S300 Standard. [Online]

Available at: [https://www.sick.com/media/pdf/3/53/853/dataSheet\\_S30B-3011BA\\_1056427\\_en.pdf](https://www.sick.com/media/pdf/3/53/853/dataSheet_S30B-3011BA_1056427_en.pdf)

[Accessed 29 August 2016].

Alves, M., 2015. About Perception and Hue Histograms in HSV Space. [Online]

Available at: <http://www.slideshare.net/michelalves/about-perception-and-hue-histograms-in-hsv-space>

[Accessed 31 August 2016].

Connette, Christian P.; Parlitz, Christopher; Hagele, Martin; Verl, Alexander;, 2009. Singularity avoidance for over-actuated, pseudo-omnidirectional, wheeled mobile robots. Kobe, IEEE International Conference on Robotics and Automation.

Corke, P., 2013. Robotics, Vision and Control Fundamental Algorithms in MATLAB. Berlin: Springer.

D'Silva, P., 2008. A Little About Color: HSV vs. RGB. [Online]

Available at: [https://www.kirupa.com/design/little\\_about\\_color\\_hsv\\_rgb.htm](https://www.kirupa.com/design/little_about_color_hsv_rgb.htm)

[Accessed 31 August 2016].

Duda, Richard O.; Hart, Peter E.; Stork, David G., 2000. Pattern Classification. 2nd ed. New York: Wiley.

Even-Dar, Eyal; Mansour, Yishay;, 2003. Learning Rates for Q-learning. Journal of Machine Learning Research, Volume 5, pp. 1-25.

- Fiala, M., 2005. ARTag, a fiducial marker system using digital techniques. San Diego, IEEE Computer Society Conference on Computer Vision and Pattern Recognition.
- Forsyth, David A.; Ponce, Jean;, 2012. Computer Vision: A Modern Approach. 2nd ed. Upper Saddle River, New Jersey: Prentice Hall.
- Gaskett, G., 2002. Q-Learning for Robot Control, Canberra: The Australian National University.
- Gomes, D., 2013. roscpp/Overview/Time. [Online]  
Available at: <http://wiki.ros.org/roscpp/Overview/Time>  
[Accessed 1 September 2016].
- Hutchinson, Seth; Gregory, Hager D.; Corke, Peter I., 1996. A Tutorial on Visual Servo Control. IEEE Transactions on Robotics and Automation, 12(5), pp. 651 - 670.
- IPA, F., 2009. The rob@work 3. [Online]  
Available at: [www.rob-at-work.de/en](http://www.rob-at-work.de/en)  
[Accessed 29 August 2016].
- Jun, Li; Lilienthal, Achim; Martinez-Marin, Tomas; Duckett, Tom;, 2006. Q-RAN: A Constructive Reinforcement Learning Approach for Robot Behavior Learning. Beijing, IEEE International Conference on Intelligent Robots and Systems.
- LaValle, S. M., 2006. Geometric Representations and Transformations. In: Planning Algorithms. Cambridge: Cambridge University Press, pp. 81-92.
- Lepetit, Vincent; Fua, Pascal;, 2005. Monocular Model-Based 3D Tracking of Rigid Object: A Survey. Foundations and Trends® in Computer Graphics and Vision, 1(1), pp. 1-89.
- Nguyen, Viet; Martinelli, Agostino; Tomatis, Nicola; Siegwart, Roland;, 2005. A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics. Alberta, IEEE/RSJ International Conference on Intelligent Robots and Systems.
- Nilsson, S., 2010. Real-time Trajectory Generation and Control of a Semi-Omnidirectional Mobile Robot, Lund: Department of Automatic Control, Lund University.

Ogata, K., 2010. Modern Control Engineering. 5th ed. Upper Saddle River: Prentice Hall.

OpenCV, 2000. Open Source Computer Vision Library (OpenCV). [Online]

Available at: <http://opencv.org/>

[Accessed 29 August 2016].

OpenCV, 2011. Camera Calibration and 3D Reconstruction. [Online]

Available at:

[http://docs.opencv.org/2.4/modules/calib3d/doc/camera\\_calibration\\_and\\_3d\\_reconstruction.html#rodrigues](http://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html#rodrigues)

[Accessed 30 August 2016].

OpenCV, 2011. Camera calibration With OpenCV. [Online]

Available at:

[http://docs.opencv.org/2.4/doc/tutorials/calib3d/camera\\_calibration/camera\\_calibration.html#camera-calibration-with-opencv](http://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html#camera-calibration-with-opencv)

[Accessed 7 September 2016].

OpenCV, 2011. Miscellaneous Image Transformations: cvtColor. [Online]

Available at:

[http://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous\\_transformations.html#cvtColor](http://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html#cvtColor)

[Accessed 31 August 2016].

OpenCV, 2015. Detection of ArUco Markers. [Online]

Available at: [http://docs.opencv.org/3.1.0/d5/dae/tutorial\\_aruco\\_detection.html](http://docs.opencv.org/3.1.0/d5/dae/tutorial_aruco_detection.html)

[Accessed 31 August 2016].

Qt, 1994. Qt | Cross-platform software development for embedded & desktop. [Online]

Available at: <https://www.qt.io>

[Accessed 29 August 2016].

RoboRealm, 2005. Color Object Tracking. [Online]

Available at: [http://www.roborealm.com/tutorial/color\\_object\\_tracking\\_2/slide010.php](http://www.roborealm.com/tutorial/color_object_tracking_2/slide010.php)

[Accessed 31 August 2016].

Russell, Stuart; Norvig, Peter;, 2010. Artificial Intelligence: A Modern Approach. 3rd ed. Upper Saddle River: Pearson.

Saito, I., 2015. tf - ROS Wiki. [Online]

Available at: <http://wiki.ros.org/tf>

[Accessed 30 August 2016].

Sutton, Richard S.; Barto, Andrew G., 1998. Reinforcement Learning: An Introduction. 1st ed. Cambridge: MIT Press.

Teixidó, Mercè; Pallejà, Tomàs; Font, Davinia; Tresanchez, Marcel; Moreno, Javier; Palacín, Jordi;, 2012. Two-Dimensional Radial Laser Scanning For Circular Marker Detection And External Mobile Robot Tracking. Sensors, 12(12), pp. 16482-16497.

Ubuntu, 1991. Ubuntu: The leading OS for PC, tablet, phone and cloud. [Online]

Available at: <http://www.ubuntu.com/>

[Accessed 29 August 2016].

Welsh, J., 2014. Aruco Marker Generator. [Online]

Available at: <http://terpconnect.umd.edu/~jwelsh12/enes100/markergen.html>

[Accessed 31 August 2016].

Wilson, William J.; Williams Hulls, Carol C.; Bell, Graham S., 1996. Relative End-Effector Control Using Cartesian Position Based Visual Servoing. IEEE Transactions on Robotics and Automation, 12(5), pp. 684-696.