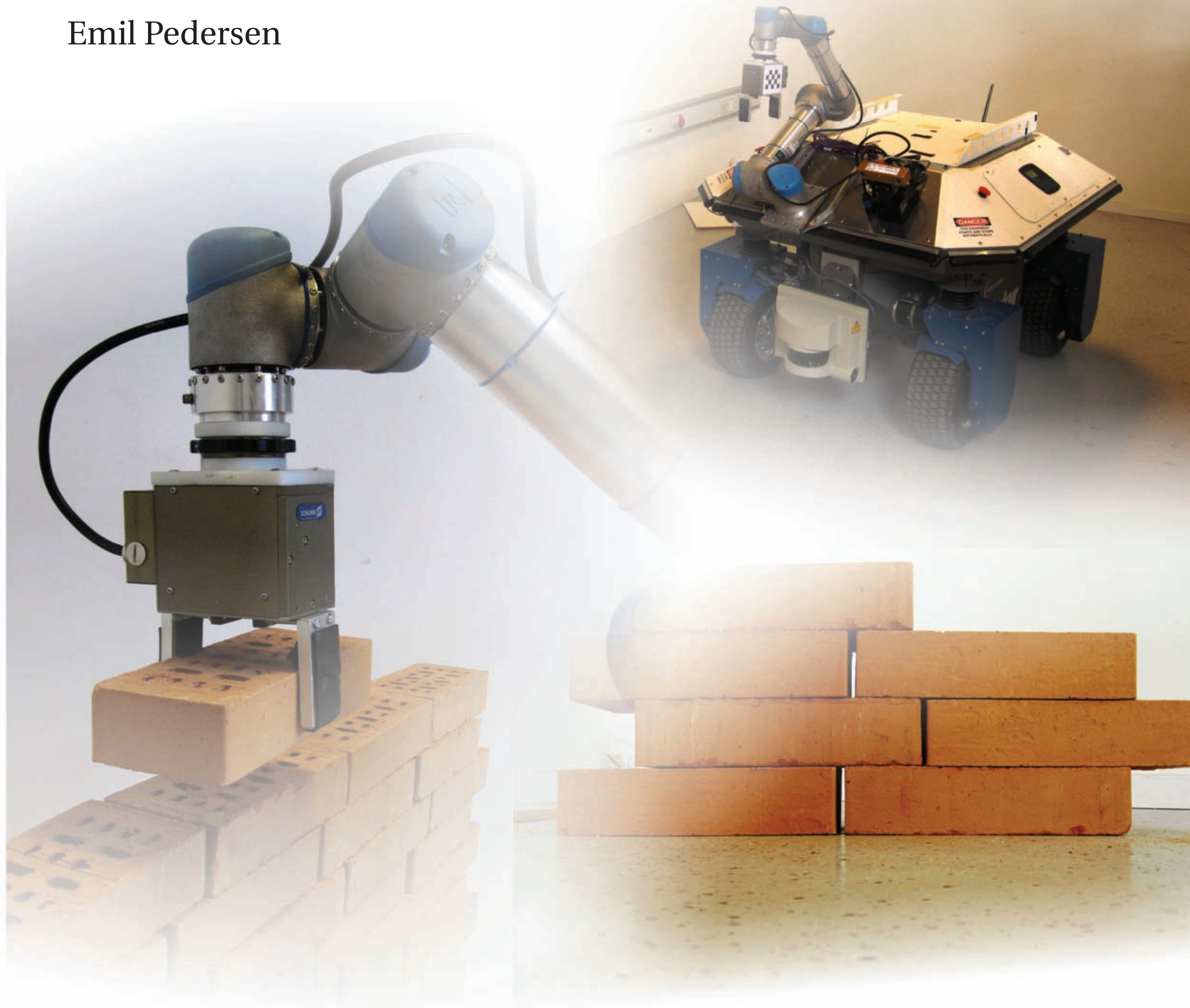
SYDDANSK UNIVERSITET
UNIVERSITY OF SOUTHERN DENMARK

# Performing Masonry with a Mobile Manipulator

Oluf Skov Nielsen
Emil Pedersen



A thesis submitted in the fulfillment of the requirements for the degree of
Master of Science in Robot Systems Engineering

Supervisors: Professor Henrik Gordon Petersen
Assistant Professor Lars-Peter Ellekilde

The Maersk Mc-Kinney Moller Institute
University of Southern Denmark
June 2010

## Abstract

Masonry is a task that contains many repetitions of similar actions, and as wages increases, automatic approaches to brick laying is becoming feasible. This master thesis report describes the development of an autonomous brick laying mobile manipulator capable of building walls at a construction site. The robot consists of a mobile platform on which a lightweight manipulator is mounted. The robot should move around in a semi-structured environment, picking up bricks and placing them in a wall with an accuracy comparable to that of conventional masonry. The system assumes inaccurate movement of the platform, such that sensor readings are needed to avoid collisions and localize the robot. The system uses stereo vision to detect bricks in order to pick these up, and to accurately estimate the transformation of a reference frame, relative to which, the bricks are to be placed. The system relies on an accurate calibration of the manipulator with respect to the stereo vision system, for which an automatic procedure is developed. The developed prototype robot was capable of building a 1 m × 1 m wall in a quality close to the tolerances of the industry, meaning that the standard deviation of the position of a placed brick is around 3 mm. The report also contains detailed tests and evaluations of the uncertainties of the stereo camera calibration, manipulator position calibration, platform and manipulator accuracy and the quality of a constructed wall. These results and the methods for obtaining them are strong tools and references applicable for evaluating the performance of other robotic systems.

# Acknowledgements

Throughout the project we have received help, cheers and advice from a lot of people, and without their kindness and involvement, a lot of the results achieved in this project would not have been possible.

First of all, our supervisors, Henrik Gordon Petersen and Lars-Peter Ellekilde have both supplied excellent guidance, input and feedback and always been ready to help us resolve issues, small ones as well as big ones. Furthermore, they have shown faith in our project and granted permissions for the funding of several repair works of hardware and ordering of spare parts.

Next, Research Assistant Anders Bøgild has helped us resolving a lot of practicalities on a daily basis in the RoboLab and he has also been a great coworker in order to use, configure and install software on the Seekur platform. Also, he took care of the main correspondence with MobileRobots Inc., when that was needed for repairing the platform. Ph.D Student Jimmy Alison Jørgensen supplied extensive support in dealing with the Schunk PG70 gripper. Ph.D Student Anders Kjær-Nielsen gave useful inputs on the calibration and use of the BumbleBee2 stereo camera. Research Assistant Rune Søe-Knudsen has been helpful in calibrating the Universal Robots manipulator. Software Developer Frederik Kvistgaard Ellehøj of Universal Robots has supplied fast and valuable support on the controller of the manipulator, and has always responded quickly and wisely to our requests. Associate Professor Jørgen Maagaard has several times supplied guidance, especially regarding the mechanics of the robot. Electronics Technician Carsten Albertsen, also several times took time to thoroughly perform repairs of the electronics of the robot. Project Executive Niels Jul Jacobsen and Project Manager Bruno Hansen of RoboCluster who both have their daily work in the RoboLab, have throughout the project shown great interest to our work and been helpful when needed. Also the team working on “Little Helper”, Mads Hvilshøj and Simon Bøgh, Ph.D Students at Aalborg University deserve thanks for providing material and inspiration to the field of mobile manipulators. Zeb Dahl from the support team of MobileRobots Inc. did a very professional and dedicated job in guiding us with troubleshooting concerning the Seekur platform, without which it would not have been possible to reach the goals we did. Not the least, our fellow students Anders Glent Buch, Bent Møller, Jakob Munksgaard and Andreas Kryger Jensen have all given us cheerful moments as we have discussed both our respective projects and enjoyed the company of each other.

At last and not to forget we would like to give special thanks to our girlfriends Pia Maja Dyrendal Rosenlund and Trine Lindved Clausen, who have shown lots of patience with us when we were daydreaming about robot motion planning at the dinner table.

# Contents

# Chapter 1

# Introduction

An omnipresent topic in modern societies is how to maintain and increase people's living standard. The problem in many of the industrialized countries is that the working fraction of the population becomes smaller and smaller as the life expectancy increases.[60] This means that the working force will have to work faster and faster in order to create the state revenues that are necessary to maintain the quality of the welfare system in terms of educating children and taking care of elderly and disabled people. It is therefore necessary that the workable group of the public is capable of producing even more value in each working hour. This can only be achieved if the companies are able to make products that can be sold with a large profit. Furthermore, high productivity standards are required in order for Danish products to be competitive on the global market, because of the difference in payroll costs among countries.

One solution to these problems is to make people work more and retire at a higher age, this is however only possible if the people are not worn down, and still it would require more expenses for the companies on salaries and thus a smaller profit. Therefore, the most feasible way is to improve the value created in each man-hour. This can be achieved by lowering the production cost or by creating new products with a high value.

During the last 50 years many tasks have been automated with the help of machines, computers and also robots. This means that many production facilities and warehouses can be run almost automatically with little need for human labor. Also, an improved quality of products is often achieved by the use of precise machines. Many tasks have however not yet been automated, and in the robotics field this is because robots until the latest years have been stationary mounted in work cells performing well defined repetitive tasks, as for example welding or palletizing.

## 1.1 Mobile robot applications

With the introduction of mobile platforms for the manipulators, the possible applications of robots are increased drastically. Because of that, it is now also possible to buy mobile manipulators off the shelf from companies such as Neobotix[4], MobileRobots Inc.[26] and IRobot[31]. Some fields of applications given attention in the robotic society and in the industry in the recent years are the following:

- **Service sector** - An emerging market for robots is the service sector, as it is one of the largest expenses to many countries, and introducing robots in households, elderly care or on hospitals would yield great opportunities of saving money. Typical tasks are room cleaning, emptying of dishwashers[20, 49]. Other fields include robots assisting the elderly in performing tasks by themselves [44].

- **Hazardous or inaccessible environments** - Mobile manipulators may find their use in environments inaccessible to humans, for example in applications such as neutralization of bombs and mines, distant space missions, underwater engineering, war zones exploration and surveillance. Mobile manipulators can be constructed in a way which make them effective for these tasks, however, the autonomy of such robots is often very low, many are just tele-operated by humans as the tasks often are complex or even unknown. Example robots are the well-reputed IRobot PackBot[31] and Nomad[67].

- **Task cycling and transportation** - Many of the tasks in the industry are automated by robotic solutions. However, in some cases, the expenses combined with the purchase and the installation of the robot may be greater than the actual gain of automating the task. In particular, this is the case for tasks which should be carried out only from time to time and does not take long to complete. In such situations, a stationary robot will be idle in the main part of the time. Today, these tasks are carried out manually by human personal who, in the meantime between task cycles, will take care of other things, thus increasing their efficiency. A static robotic solution to such tasks will therefore not be cost competitive. The project "Little Helper" [5] is an example of a mobile manipulator suited for cycling between several tasks. The tasks can also be transportation of goods, where a manipulator placed on the robot increases usability as the robot can load and unload many different types of objects at arbitrary places. A challenge of this type is held at ICRA2010[1] [61].

Many additional projects could be mentioned in the above fields, as a wide variety of approaches to solving these problems are pursued especially in the research institutes of universities. When considering autonomous mobile manipulators, especially in the home-care sector, the projects often ends with a huge focus on the extremely difficult task of object recognition before focus can be turned onto what to do with the recognized objects. Examples of that is recognition of objects for emptying a dishwasher, cleaning up or picking food in a refrigerator. The same problem often appears for robots acting in hazardous or inaccessible environments, where the solution often is to tele-operate the robots.

Task cycling and transportation tasks are tasks that are often performed in production environments. These environments can often be characterized as semi-structured, as although the environment may change over time, some information of the scene can be provided to the robot, such that the robot for example knows some defined goal positions where it can perform specific tasks. It will also be possible to determine exactly which objects the robot should manipulate, and even place identifiable markers on them, such as RFID[2] tags or visual markers. With these assumptions on the environment, the demands for the intelligence of the robot is reduced significantly and it can then be possible to create relative simple control programs. More about possible applications for mobile robots can be found in the FORK report[46].

## 1.2 Masonry robots

Construction sites can also be considered as semi-structured environments. Although the environment changes during the course of the construction process, many locations on the site can be told to the robot by using markers or a positioning system. Because of that it may be possible to develop robots able to act autonomously in such environments. A task that is relatively simple is the task of laying bricks. The task resembles pick and place operations on box-like objects very much, a task for which standard solutions exist for stationary robots. The most significant difference and difficulty compared to pick and place operations is that the platform will have to move between the pick and place operations.

In traditional construction brick laying is a time consuming task, and because of that it is more expensive to build brick walls than to use other material such as concrete. Brick walls are due to

[1] International Conference on Robotics and Automation 2010.
[2] Radio frequency identification.

aesthetically and indoor climate considerations still preferred as material in new buildings and if the cost of building brick walls can be reduced further by introducing automation, the market for brick walls may rise. The introduction of robots will also have positive effects on the bricklayers as it is possible for the mobile manipulator to perform the trivial, monotonous and repetitive work, that may seem tedious to and cause work-related injuries. This is so well known that the Japanese have a saying that captures the essence of the construction workplace: “Kitanai, Kiten, Kitsui” (Dirty, Dangerous, and Difficult)[55]. The dissertation from which this saying originates, contains a very thorough investigation of the topic of robotized masonry in terms of a critical success factor analysis.

The use of robots in this field also opens new perspectives. The possibility of very accurate placing of bricks makes it possible to construct brick walls that curve or form special patterns, yielding a wide range of new possibilities for architects and engineers. See [7, 16] for a project on the architectural benefits and possibilities of robotized masonry. This approach even enables the customer to take part in the design of the building, moving the field of automated construction of houses from mass production to mass customization by offering increased choice and design flexibility for the customers, see[1]. Also the project TailorCrete[30] is an example of how robots can meet the demand for more flexibility in the design possibilities for architects.

## 1.3 Related research and projects

Robots are already used in some parts of the masonry industry. Robot- and automation systems exist that are able to fabricate large units of brick wall in production halls. The units can then be transported to a construction site and assembled there. In this way cheap and durable brick walls can be built in a very short time. Carlsberg Mur[3] is an example of a company that delivers such a product. The approach of prefabricating wall segments in production halls have been pursued by many other projects, including [7, 54]. This approach simplifies the task a lot, as it is possible to control the environment as much as environments for other stationary industrial robot applications, and the difficulty of programming the robots is therefore decreased significantly. There is however one big disadvantage - when prefabricated wall segments are to be assembled on the construction site, the interconnections will be visible and disturbing to the viewer of the wall, this is also the reason why Carlsberg Mur are not used for outer walls in residential buildings. A way to avoid this is to build the wall at the construction site.

A solution to autonomous brick laying on the construction site, is to use a manipulator placed on a mobile platform. This enables the robot to be easily moved to the place where the wall is to be built. For around two decades, proposals have been made to automated brick laying systems, these include [51, 6, 19]. References to more projects can be found in [55]. These projects have all served as fine inspiration on how to approach the tasks in this project. An interesting research activity on the topic is carried out at Hanyang University of Korea, where a study has been made in the field of determining the robot movements when supposed to lay bricks in a specific pattern, see [68]. The approach presents a very promising mechanical design with a manipulator placed on a platform with caterpillar tracks, able to carry a pile of bricks. Also the strategy for moving the manipulator and placing the bricks can serve as a good inspiration for optimizing performance. The approach however restrains from covering one central topic - the uncertainty in the placement and movement of the platform. When taking this into consideration, the task of placing bricks in the wall becomes significantly more complicated as the transformation between the platform and the desired position is not perfectly known. The approach presented in this report will pursue more autonomy in correcting for the imprecise movement of the platform.

## 1.4 Starting point

In the FORK project[46] the basis for this project was established. Therefore, this statement on the state of this mobile manipulator project at the start of the master thesis project period is provided. This should give the reader a short introduction to what has been done during the master thesis and what has been done in the FORK project.

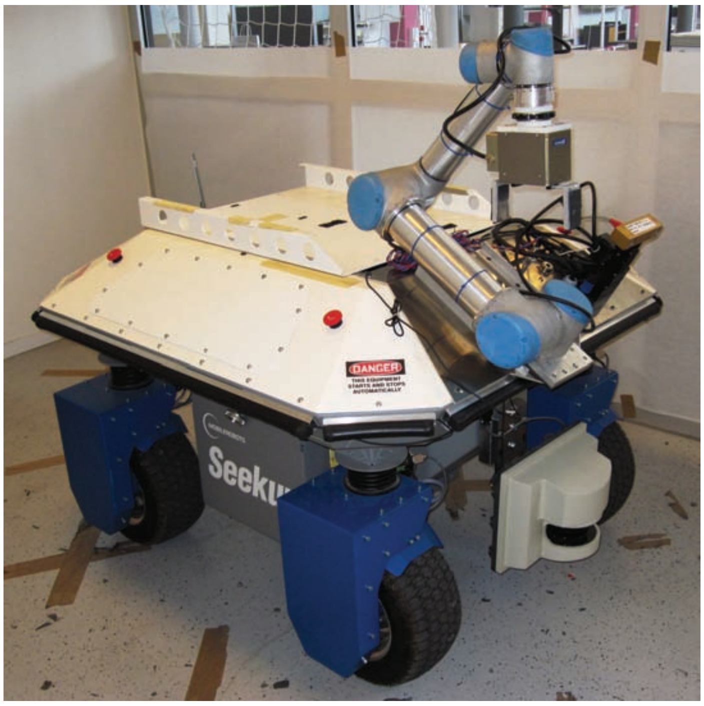


**Figure 1.1:** The mobile manipulator for which basic control was developed in the FORK project before this master thesis project.

The robot made available and customized for this project can be seen in figure 1.1, and the most important tasks done in the FORK project are the following:

- **Knowledge screening** - An investigation of the algorithms, applications, hardware and software available for mobile robots. This work served as the basis for developing the system and choosing the brick laying application in the master thesis.
- **Hardware setup** - The design and assembly of the mobile manipulator.
- **Hardware/software interface** - Interfacing the various devices, implementation and comprehension of control algorithms for the devices, especially the algorithm for making blended paths for the manipulator, see section 7.3, and calibration of the system. A calibration algorithm was developed in the FORK project, but a new improved algorithm is developed in this project in order to reach a satisfying accuracy.

Although these tasks were carried out in the FORK project, significant changes and improvements in all fields have been necessary as the development of the system continued. Therefore, the status at the end of the master thesis project for these parts of the system is also presented in this report.

## 1.5 Problem statement

In the FORK project, a mobile manipulator capable of performing object handling tasks has been developed. This includes sensing the environment with cameras and a laser scanner, controlling the manipulator with respect to what is perceived and navigating the platform around with respect to an intrinsic map of the scene. In the master thesis, the preceding development of the robot will form the basis for performing a specific type of handling tasks, namely automated masonry.

Masonry is the building of structures from individual units, bricks. Often, houses are manually built this way by skilled personal trained for the job. This thesis will focus on the development of a control scheme for the mobile manipulator which makes it capable of performing masonry at the construction site of e.g. a house or a wall in general. A high degree of autonomy is desired for the robot, meaning that when it has been specified where and how a wall should be built, the robot can then complete the task without further human involvement. In order to obtain this, several issues appear, including:

- Development of a user interface for simple specification of a building task.
- Recognition and grasping of bricks.
- Determination of desired position of each brick.
- Navigating the platform around the construction site without colliding with the structure being built or other possibly dynamic obstacles, such as human personal.
- Planning the manipulator trajectory needed to place the building unit in the structure.
- Schedule the tasks of fetching bricks and placing these, such that efficiency is kept high. This should take into account the desired bond of the wall to be build, as this will e.g. introduce constraints on the order of the bricks to be placed.
- Capability of evaluating the quality of a wall constructed by the robot with respect to quality measures of a conventionally constructed wall. This will solely regard the placement of the bricks, as mortar or other gluing technologies are not expected to be incorporated in the project.

## 1.6 Scope and delimitation

The task of performing masonry with a mobile manipulator is very complex, and there are many problems that need to be solved in order to make such a system. Therefore, not all problems can be treated with equal effort, i.e. the focus and scope of this project must be determined.

The main goal of this project is to create a working prototype of an autonomous brick laying mobile manipulator. The word autonomy does in this context mean that the robot should be able to execute a complete wall building task with minimal intervention from the user. The robot should be able to operate in an dynamic environment where objects may be moved, obstacles may occur and no exact model of the scene exists. This means that the robot must use its sensors to update the scene to make successful grasps of bricks, navigation with obstacle avoidance and placing of bricks satisfactory in the wall. With these capabilities the system is able to perform the wall construction with only a minimal input from the user.

The project is a master thesis in robotics, and therefore the focus is on programming the algorithms for controlling the robot. These algorithms include motion planning for the robotic arm, grasping and placing of objects, 3D reconstruction of objects detected by vision, robot self calibration and overall control software design. The design, implementation and analyses of these algorithms will be central to this report.

The project is delimited to utilize the existing hardware, meaning that evaluating the hardware and proposing design improvements will not be prioritized. Developing the mechanical design will primarily improve the productivity, which is not a central parameter for this proof of concept prototype, where the ability to build a wall of a satisfying quality is more important.

The robot is strong and heavy and in order to reduce the risk of damaging the robot or personnel, safety and collision avoidance must be taken into account when testing software on the real robot.

In order to successfully develop a working prototype, some assumptions on the environment in which the robot should act or on the task the robot should perform may be necessary. A sketch of how to improve the system to handle situations in which the assumptions cannot be considered valid, will also be presented.

## 1.7 Reading guide

This report is divided into chapters as follows. First, chapters 3 and 4 present the methodology used for solving the masonry application and gives a brief insight to how the project has been regarded, also with descriptions of peripheral activities. Chapter 2 gives an introduction to the hardware used in the project and also include analyses of the usability with respect to the masonry application. Chapters 5 to 7 are core chapters of the report with detailed presentations of the calibration issues regarded, the complete software system developed in the project and an assessment of all algorithms and functionality developed and used for making the robot perform masonry. Chapter 8 presents the results which have been obtained by testing the robot control system for the autonomous construction of a wall. Finally, chapter 9 discuss, concludes and puts the project into perspective with respect to the objectives, what is achieved and the problem statement.

### 1.7.1 Enclosures

On the enclosed DVD the following directories are placed in the root.

- **Report** - A copy of the pdf-version of this report and the FORK report.
- **Source** - The complete source code developed in this project. The reader is referred to the authors for help on compiling and running.
- **Data** - Files containing the main datasets used for analyzing and presenting the results in the report.
- **Data analysis** - Scripts written to load data files, analyze the data and create plots.
- **Mathematics** - Scripts developed to assess mathematic issues.
- **Media** - Pictures and videos of the robot.

Additionally the wiki, Brick laying mobile manipulator[47], at RoboLab wiki, contains text, pictures and videos presenting the system and its capabilities. Especially the videos can be recommended in order to see how the system actually works. These include videos of platform navigation, data acquisition for calibration, the robot playing tic tac toe, and of course the robot performing masonry. It should be noted that the information on this page is not complete and may therefore differ from what is stated in the report, in this case, the word of the report has precedence.

### 1.7.2 Notation

The notation given below applies throughout the report unless stated otherwise. A homogeneous transformation matrix used to describe the position and orientation of a frame $A$ with respect to a frame $B$ is given by

$$T_B^A = \begin{pmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.1}$$

To reach elements in a matrix or vector, the () operator is used, with indices starting at 1, e.g. $T_B^A(2,3) = r_{23}$. In order to extract several elements, a : is used to specify all values including and in between, e.g. $T_B^A(1:3,3) = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T$. By this either a submatrix or a subvector is gotten.

As $T$ denotes homogeneous transformation matrices with six degrees of freedom, the six dimensional *pose* vector $P$ is also used to describe this same transformation. That will be done using $P = \begin{bmatrix} p_x & p_y & p_z & \phi & \theta & \psi \end{bmatrix}$, where the last three entries specifies a roll-pitch-yaw rotation, such that it is equivalent to a $3 \times 3$ rotation matrix $R$

$$R = R_x(\phi)R_y(\theta)R_z(\psi) \tag{1.2}$$

where $R_i(\theta_j)$ denotes a rotation of angle $\theta_j$ around the $i$'th axis. Angles are usually given in radians but may be specified in degrees as well. For any transformation of frame $A$ with respect to a frame $B$ we may then write

$$P_B^A = \text{pose}\left(T_B^A\right) \Leftrightarrow T_B^A = \text{transform}\left(P_B^A\right) \tag{1.3}$$

Finally, the indices of joints of manipulator start at 1 which will then refer to the base joint. $TCP$ refers to the tool center point, i.e. the end frame of a manipulator.

# Chapter 2

# Robot hardware presentation

In this chapter, the hardware devices of the mobile manipulator are presented. It is important to have a good knowledge and understanding of these and the capabilities of them in order to be able to see possibilities and limitations and develop intelligent solutions. The chapter will contain a presentation of the Seekur platform, a presentation of the Universal Robots UR-6-85-5-A manipulator along with a test of its accuracy and a presentation of how it is mounted on the platform in order to evaluate reachability and stability. The used gripper, stereo camera and laser scanner will also be introduced.

## 2.1 Mechanics of the platform

The Seekur platform is a standalone device manufactured by MobileRobots Inc. and this presentation is based on [42] and [27]. In figure 2.1 the Seekur platform is shown.

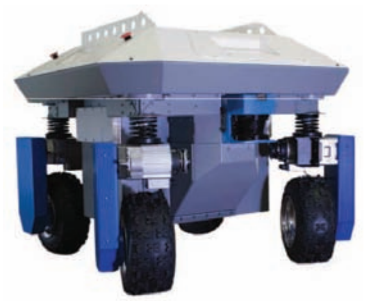


**Figure 2.1:** The Seekur platform as seen in [42].

The Seekur platform is equipped with four omnidirectional wheels which means it is capable of driving in any direction regardless its current orientation. The tires are robust and makes the robot drive well outdoors with speeds up to 2.2 m/s. It has a maximum payload of 70 kg and a battery capacity of seven hours of driving. It is 1.4 m long, 1.3 m wide and 1.1 m high and weighs 350 kg.

## 2.2 Manipulator mechanics

The manipulator used in this project is a Universal Robots UR-6-85-5-A manipulator[64] with six revolute joints all with a range of $\pm 2\pi$ rad, reachability of 85 cm and a maximum payload of 5 kg. All six joints are of the similar type, but the size of the three joints closest to the base is larger than the size of the wrist joints. Designing the manipulator this way has made it cost competitive compared to manipulators with different designs of each joint. The manipulator can be seen in an industrial application in figure 2.2.

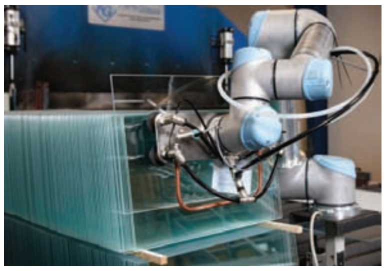


**Figure 2.2:** The Universal Robots UR-6-85-5-A in a working scene.[56]

As it might be noticed when comparing the manipulator in the figure with the one in this project, e.g. in figure 1.1, the manipulator has been changed. This is because, it has been necessary to switch to a zero series model of the UR-6-85-5-A manipulator, as the 2 series robot used in the FORK project was to be used in another project. Fortunately the interface to the two is similar, and only the mounting needed to be changed.

The zero series manipulator has a calibrated kinematic model available, including values for encoder decentralization[1] that should be used when reading the joint configuration and when moving the manipulator to a desired configuration. However, one disadvantage of this older zero series manipulator, is that the connection between the wires from the manipulator to the controller box plug can fail, resulting in the robot loosing the calibrated offset values for each joint. Therefore, a calibration procedure[2] estimating the joint value offsets from the kinematic model had to be performed. As switching of the manipulator happened several times during the project, it proved itself useful that the calibration could be performed automatically. In any case, since the zero series manipulator is mounted differently, a determination of the transformation between the manipulator base and the camera is necessary.

### 2.2.1 Configuration space

The configuration space of $\pm 2\pi$ rad does obviously contain many double configurations, meaning configurations where a joint is turned $2\pi$ rad compared to an identically looking configuration. This can be used to shorten paths for the manipulator, as the movement of it is less constrained. A problem is however that the gripper must be supplied with power and signals to function. The manipulator has an internal wire for this purpose but its maximal current limit of 600 mA is too little for operating the PG70 gripper. Therefore an external wire must be used. This wire has to be placed on the manipulator, where a wisely chosen mounting of it along the manipulator is necessary to avoid breaking the wire or having too much uncontrollable loose wire. This is practically impossible with the basic joint limits, so limiting the configuration space to about one round for all joints is a simple solution to that. The joint limits are presented in equation 2.1.

$$Q_{bounds} = \begin{pmatrix} -50^\circ & 340^\circ \\ -190^\circ & 10^\circ \\ -180^\circ & 180^\circ \\ -180^\circ & 180^\circ \\ -180^\circ & 180^\circ \\ -180^\circ & 180^\circ \end{pmatrix} \tag{2.1}$$

[1] If the encoder disc in a joint is not perfectly centered about the axis of rotation, it will result in the encoder distance measured between equally distanced joint configurations being configuration dependent.

[2] See chapter 5 for a complete description of the calibration procedure developed in this project.

The two first joints are limited such that configurations that will always result in collisions with the platform are not possible, also the limits are set such that the base joint can rotate in the area furthest away from the cameras. It is worth noting that limiting the configuration space this way can limit the possibilities for finding short paths, but as all joints ranges at least $2\pi$ rad, except the second where a large part of its configuration is blocked by the platform, all possible inverse kinematics solutions are applicable.

### 2.2.2 Manipulator accuracy

In the product sheet for the UR-6-85-5-A manipulator, Universal Robots claim to have a repetition accuracy of $\pm 0.1$mm[64]. This is very good and a far higher accuracy than necessary for manipulating bricks or manipulating objects located by vision, where uncertainties are magnitudes larger. It was however discovered that the manipulator sometimes did not reach goals with the specified accuracy. This means that when providing a path for the manipulator to move, the manipulator will not end up exactly at the end position of the path. Therefore a simple test of the repetition accuracy of the manipulator is carried out in order to investigate the actual accuracy.

The test is simply to make the manipulator move a linear path in a random direction in its configuration space and then compare the desired end configuration of that path to the actual achieved configuration. To make the results comparable to the reference value, the forward kinematics for the transformation from base to end of the manipulator is calculated for the configurations, making it possible to evaluate the precision in Cartesian coordinates.

The random linear paths are made up by the actual configuration of the manipulator and a configuration placed at an unit step in a uniformly distributed direction away from the actual configuration. The direction is generated by sampling a vector uniformly inside a six dimensional[3] unit sphere. As an algorithm for uniform sampling inside a hypercube bounded by $[-1;1]$ is available, this is simply done by sampling new vectors in the hypercube, until the distance to origo is less than one, and the configuration thus is inside the sphere. The paths are necessarily constrained by the bounds of the manipulator and configurations that will cause the manipulator to be in collision. With these constraints, we cannot expect a perfectly uniform distribution of directions and coverage of the configuration space, but we can assume that the generated paths provide sufficiently representative observations of such.

The results from creating 100 random linear paths can be seen in figure 2.3. It can be seen that errors are of size up to 0.002 rad. This does seem as a small value, but considering the length of the robotic arm, which is about one meter with the gripper mounted, an error in this size will give rise to an error of approximately 2 mm in position at the end of the manipulator.

Figure 2.4 shows the errors in the transformation $T_{base}^{TCP}(q)$, the transformation of the TCP frame with respect to the base frame of the manipulator with the specified joint configuration. It can be seen that the errors for the position part is in the expected range of up to 2 mm. It is interesting to note that especially for the position part, the errors do not seem to have zero mean. When ignoring the outliers containing errors of up to 0.02 rad, it can still be seen that the errors for the rotation is about 0.005 rad. These errors are quite large, and the position error is more than 20 times what is stated in the product sheet. It is worth considering this error when addressing uncertainties in performing precision tasks such as pick and place operations.

It is hard to determine the cause of this error but an interesting plot can be seen in figure 2.5. That plot shows the angle, $\theta$ , between the error vector, $\epsilon$, and the direction in joint space, $q_{dir}$, a goal is approached, computed as in equation 2.2.

$$\theta = \cos^{-1}\left(\frac{q_{dir}\cdot\epsilon}{|q_{dir}||\epsilon|}\right) \tag{2.2}$$

[3]Due to the six degrees of freedom of the manipulator.

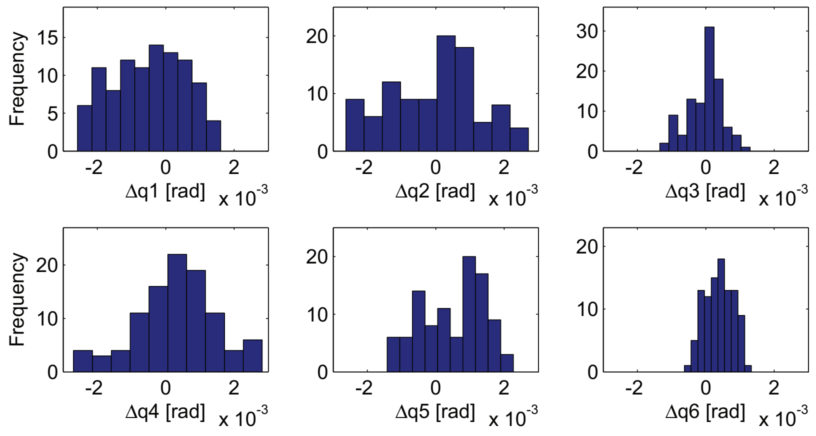


**Figure 2.3:** The error of the manipulator when moving to a goal joint space.

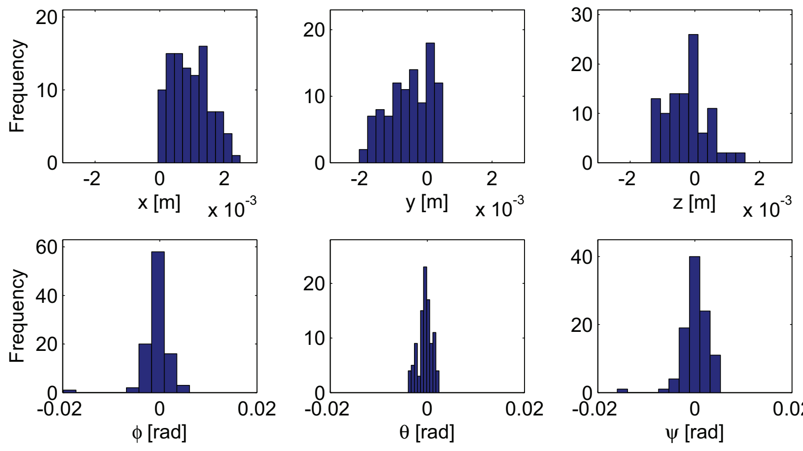


**Figure 2.4:** The error of the manipulator when moving to a goal Cartesian space.

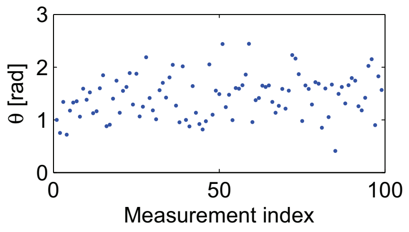


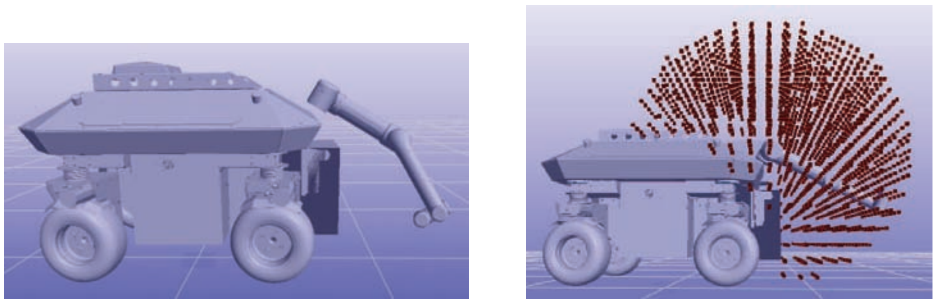
**Figure 2.5:** The angle between the error and the direction in which the manipulator has moved, calculated in configuration space by equation 2.2.

This angle is not uniformly distributed within its range of $[0;2\pi]$, but shows a high probability of occurrences around 1.5 rad. In the equation, 2.2, $\epsilon$ is calculated as $\epsilon = q_{desired} - q_{actual}$. The interpretation of this analysis must be that the error is most likely close to orthogonal to the direction of the manipulator movement, meaning that the manipulator stopping too early or late on its path cannot be the source to this error. Instead, it may be risked that the manipulator traverses a path that is slightly displaced from the desired. This should be taking into account when checking for collisions.

It is difficult to tell if this high inaccuracy is due to the fact that the manipulator is a worn zero series model, or if the same will stand for a new manipulator, but the fact that the error is visible just by looking at the joint encoders indicates that it may be possible to tune the joint controllers to reduce this error. It is however not possible to access and change these, as they are proprietary to Universal Robots.

## 2.3 Mounting of the manipulator

It is desirable for the manipulator, when placed on the platform, to have as much reachability and maneuverability as possible. Since it is important for the robot to be able to manipulate objects placed on the ground, e.g. when laying the first tier of bricks in a wall, the manipulator is placed at a sufficiently low level in order to accommodate this. It has been decided to give the manipulator base a sloped mounting, since this improves the ability for the robot to also handle or place objects on top of the platform. This area on the robot may be used as a kind of storage for e.g. bricks that must be carried to the wall building location. In figure 2.6 the placement of the manipulator together with the resulting reachability is shown.



**(a)** Side view showing the manipulator base mounting aligned with the slope of the Seekur chassis.

**(b)** Resulting reachability for the manipulator indicated by the red cubes.

**Figure 2.6:** Design choice for the placement of the manipulator on the platform.[46]

### 2.3.1 Stability of a mobile manipulator

In the FORK project a simple model for evaluating the stability of a mobile manipulator was developed, see figure 2.7.

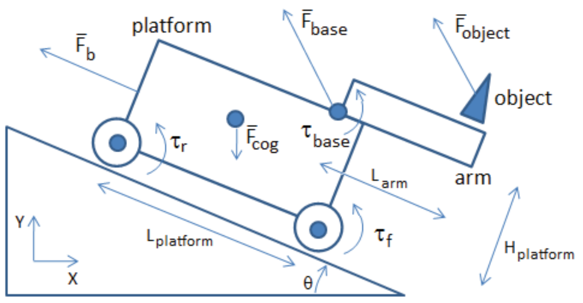


**Figure 2.7:** Model of mobile manipulator for evaluating dynamic stability.

By determining the net torques at the front and rear ends of the mobile manipulator, $\tau_f$ and $\tau_r$, it is possible to evaluate the risk of the robot tipping over during operation. In the FORK project it was shown that the chosen placement of the manipulator, even with maximum forces acting on the robot, still accommodates a stable state of the robot. Additionally, this result of the model is more conservative than one would experience in reality. This is mainly because the Seekur platform specifically has been designed to have a low center of mass to reduce the risk of tipping over, which is not incorporated very well into the model used.

## 2.4 Gripper

A Schunk PG70 parallel gripper is mounted on the manipulator TCP using a set of adapter flanges. The maximal distance between the two fingers is not sufficient for brick handling, so some customized fingers have been made to increase the gripper range. The gripper with the adjusted fingers can be seen in figure 2.8. The weight of the gripper is 1.4 kg which does not compromise

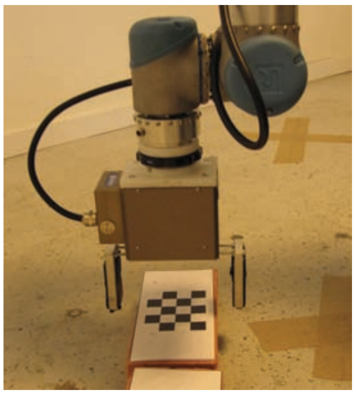


**Figure 2.8:** The Schunk PG70 parallel gripper used in the project.

the payload limit for the manipulator, even when handling bricks, as these usually weigh a bit less than 1 kg.

The Schunk PG70 gripper has previously been used in other projects on the institute, and therefore a control interface implemented in RobWorkHardware[57] is available. However, this specific gripper did not respond as intended to this, and no clear indication of why that was the case was apparent. Through systematic and thorough tests, very time consuming to the project, it became clear that a small firmware difference caused this malfunction. The resulting control interface is therefore inspired by the RobWorkHardware implementation, but with adjustments accounting for the firmware difference.

## 2.5 Vision system

At the front of the Seekur platform, a stereo camera is mounted on a pan tilt unit. The pan tilt unit has the possibility of increasing the field of view drastically, and with precise encoders it is still possible to relate positions viewed in the rotated cameras to coordinates in the fixed frames of the robot. In order to do this, a calibration of the system, including the dynamic pan and tilt joint of the pan tilt unit must be developed. So far, the pan tilt unit has however not been used actively, as the field of view of the mounted cameras is considered sufficient. The pan tilt unit can be controlled by ARIA[40].

### 2.5.1 BumbleBee2 stereo vision system

On the pan tilt unit, a BumbleBee2 BB2-08S2C-38 stereo vision system by Point Grey is mounted, it consists of two color cameras with a $1024 \times 768$ pixel resolution and a HFOV[4] of 100 degrees, [24]. In

[4] Horizontal Field Of View.

figure 2.9 the BumbleBee2 stereo camera is shown. An important advantage of this camera, is that it is already used in many projects on the institute, so knowledge on how to interface with it in the best way was easily available. When used in Linux, as is necessary on the onboard microcontroller on the Seekur platform, the camera is controlled by the open source general firewire camera library, libdc1394[15].

**Figure 2.9:** BumbleBee2 BB2-08S2C-38 stereo camera by Point Grey.

The camera system is able to acquire images in either single-shot or continuous mode using DMA[5] access for gaining high frame rates without using the processor. The cameras come in a stable metal case, making it almost impossible for the cameras to lose their internal calibration. For Windows users, the software Triclops SDK performs undistortion and rectification of acquired images using calibration data provided by Point Grey stored in the camera. It is however not possible to use this software in Linux, so a stereo calibration has to be made on the cameras before image processing can be done optimally, as undistortion of images is needed for the camera to fit the pinhole model, see section 5.1.1.

The very wide field of view of the camera is good for locating objects and for viewing a large area at once. However, the trade off is a reduction in precision, yielding a larger uncertainty in estimating 3D positions of objects using stereopsis. It was also discovered during the calibration that such camera is very distorted, making it difficult to fit a model to this large and uneven distortion in order to undistort images. It could be possible to use this setup together with a camera with a larger focal length and thus a more precise depth estimation.

## 2.6 Laser range finder

The Seekur platform is equipped with a Sick LMS2xx laser scanner which is used by the localization software on the onboard computer. The laser scanner only scans in 2D, which means it only detects objects at a certain height, i.e. a disk of the surroundings represented by distances to objects is mapped. It is used as primary source of information to the localization and navigation software, ARNL[41]. The data from it can also be accessed directly, in order to be used for other purposes, such as collision avoidance for the manipulator or localization using reflective markers.

[5]Direct memory access.

# Chapter 3

# Solution method

This chapter will contain an introduction to the overall approach for creating a mobile brick laying robot. When an overall approach is determined and specified, it will be possible to divide it into solving a number of subtasks. The detailed presentation of the solution of these subtasks will be the content of the following chapters. This chapter, however, describes the overall approach, the tasks needed to be solved, and gives a short introduction to the problems that may arise when solving these tasks.

It is written in the problem statement that the robot should be able to build walls on a construction site. This is opposed to other approaches where robots build entire wall units remotely in a production hall and only the assembly of these units is done on site. Having the robot on the construction site, it cannot be entirely preprogrammed, and therefore it is necessary to have a control interface for the robot that makes it possible for the workers on the construction site to specify the task the robot should perform. The whole system must then be intelligent enough to do the task autonomously based on the information given to it. In this regard it is important for the robot to use its sensors to perceive the important objects such as bricks and humans in the real world and act accordingly.

## 3.1 Approach

The task of performing masonry with a robot can actually be seen as a sequence of pick and place tasks, with some movement of the platform in between. When using a mobile platform in an outdoor scene that is not fully known, the robot must be able to act according to how its sensors recognizes the scene. This means that the actions of the robot must be guided and verified by feedback from sensors in order to be successful in this environment.

At the most abstract level, the task of bricklaying is very simple, as it is just to pick up some bricks, move to the wall and then place the bricks in the desired pattern, and then move back to pick up more bricks until the wall has been built. The approach to solving these tasks with the mobile manipulator is constrained by the available tools on the robot in terms of both sensors and actuators, as it is not part of this project to develop new tools for the robot. The robot is equipped with the manipulator and gripper as the only tools to pick up bricks, so the robot will have to pick bricks one by one, using the gripper. When moving to the area where the wall is intended to be build, the robot will have to use the sensors, especially the stereo camera and the laser scanner in order to localize itself relative to the wall.

Based on these constraints, an overview of the tasks involved in brick laying for the mobile manipulator containing identified subtasks can be seen in figure 3.1. It contains the three tasks of brick laying, picking up bricks, moving to the desired placing position, and placing the bricks, along with the tasks involved in developing the control system. The tasks and their subtasks will be described in the following sections.

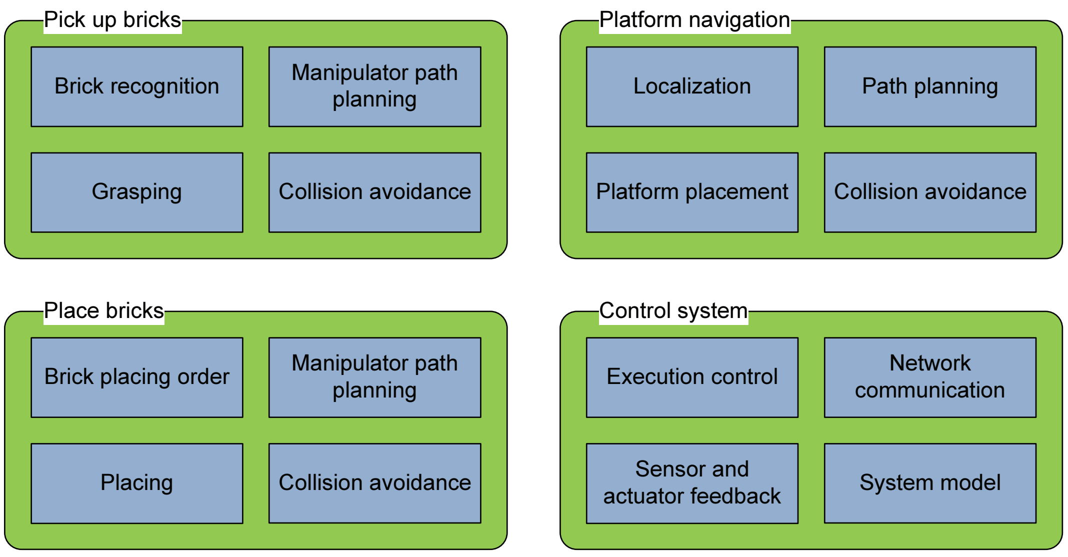


**Figure 3.1:** The main components and tasks of the brick laying robot system.

### 3.1.1 Picking bricks

The robot must be able to pick up bricks in order to transport these to the area where the wall is to be constructed. As the robot will have to move to a position where it can gather bricks from, there will be some inaccuracy in the position of the bricks relative to the platform. Therefore, sensor readings must be used in order to pick up the bricks successfully. The setup will use the robotic arm and a parallel gripper to pick up bricks.

A crucial part of the ability to pick up bricks, is how the position of these is determined when the platform placement may vary. The position of the bricks relative to the robot must therefore be determined in order to grasp the bricks. It is very important to be able to grasp the bricks precisely, because the difference between the desired and the actual grasp will cause an error in the position to be present when placing the brick with respect to a desired transformation.

The way the bricks are picked is by finding their 3D transformation using stereo vision. Knowing the shape of the bricks it is then possible to find a stable grasp and pick up the bricks. When picking up bricks it is important to be able to avoid collisions with other bricks or objects while moving the manipulator during the grasping process. An aspect of this is to add to the collision checking model that a brick is held by the gripper.

### 3.1.2 Platform navigation

The solution to this task has the responsibility of moving the platform such that the manipulator is able to pick up and place objects. The navigation system will rely on the existing navigation system on the platform. It uses a laser scanner and the encoders of the platform to localize itself in a map created from readings from the laser scanner. It is to be determined if the navigation system is sufficiently for navigating the big platform precisely in tight spaces, as a construction site may be. A very important aspect of the navigation system is to design the system and algorithms such that collisions are avoided with both static obstacles like walls, but also dynamic obstacles like human beings. This causes the need for dynamic updates of the planned path for the platform as obstacles are detected.

An important part of the platform navigation is to choose the desired position of the platform in order to execute a task with the manipulator. To chose this properly, one must have in mind the reachability of the robot, the area visible by the stereo vision system if that is needed, and the fact that the platform movement is imprecise. Knowledge of the platform movement uncertainty can lead to a smaller risk of tasks for the manipulator ending up impossible, as some goal is not reachable. It is considered relative time consuming to adjust an imprecise placement of the platform. This is because it will have to avoid collisions in this adjustment movement, and thus will not be able to use its capability of moving sideways in such a situation as the laser scanner can only see the area in front of the robot. Instead it will have to rotate, move forwards in the desired direction and rotate back to the desired orientation.

### 3.1.3 Placing bricks

The task of placing a brick in a specified position is not very difficult as long as the desired position is known, the problem is to specify that position in an internal coordinate system of the robot. Therefore, this task deals with handling the uncertainty of the platform movement by using sensors to find the transformation between the platform and the desired placing position. The accuracy of this transformation will be reflected in the ability of the robot to place the bricks correctly in the wall with a satisfying accuracy.

When placing bricks, it is assumed that the transformation of the brick with respect to the gripper is known or at least contains only a small error. It is then desirable to place the bricks carefully, meaning that as the brick is laid and thus touches other bricks in the wall, this contact should be made soft. This means that the bricks are not dropped far from the desired position, nor crushed into the bricks already placed in the wall.

An important issue when placing bricks in a wall is to place the bricks in the right position. This task is to calculate the desired positions of the bricks from the input size of the wall and choice of bond, knowing the size of the bricks and the desired distance between them. It is then important to be able to determine the order of which, the placing of bricks should be executed. Obviously, a brick cannot be placed if the bricks on which it is supposed to rest have not been placed. There are also other considerations such as time consumption, as it may be possible to decrease unnecessary movement of the platform by wisely choosing the order of the brick placing. This will particularly be interesting when constructing larger walls and carrying many bricks.

### 3.1.4 Execution of tasks

An overall control algorithm must also be developed in order to let the robot know “what to do when”. It must be possible to get data from this algorithm on the quality and speed of the execution of the task. This algorithm must also be able to handle errors or unsuccessful execution of subtasks properly. The algorithm is initialized by the user and handles the entire construction process. An important part of that is to know the current state of the scene in which the robot acts, such that the robot knows where objects are and is able to avoid collisions.

## 3.2 Other facilities and capabilities

In order to execute the tasks described above, the robot has to be functional, i.e. the hardware parts must be found and installed both physically and electrically. Since most of the hardware equipment for the mobile manipulator has been decided upon prior to this project, it is more of a task to determine the assembly of these parts.

In addition to the physical installation of the hardware devices, interfaces to these in software must also be developed. This means that programs will have to be developed, which have the responsibility of sending the correct commands at the right time to the different hardware components when requested by the overall control system. As the robot is constituted of several devices and microcontrollers, the software controlling the entire system is spread out into several programs and services.

Finally, it is crucial that the kinematic modeling of the robot is calibrated. This calibration regards the specific transformation existing between the stereo camera and the manipulator base and also the one between the manipulator TCP and the gripper. In addition to these transformations, the kinematic model also relies on precise relationship between encoder values for the manipulator and the true joint angles. Not part of the kinematic model, but essential for making use of the observations by the stereo camera, is the calibration of the stereo camera. The calibration will accommodate the calculation of e.g. brick positions relative to the manipulator.

# Chapter 4

# Working process

This chapter is a description of how the tasks of the project have been regarded, how branches of the project objective have been pursued and what has influenced the project progress. First some overall statements on how the work has been carried out are made, and later descriptions of how work has been done to reach certain goals during the project are presented.

## 4.1 Goal directed development

To be able to succeed in a large project, where the outlook easily can be lost, it is important to work in a way such that it is always possible to find a purpose in the current task, this can be done by defining goals that must be reached during the project.

The working process should reflect the goals of the project, meaning that all work done during the process should lead to improvements in the developed prototype software. This means that it will be important to continuously evaluate the work done in order not to waste time on optimizing some arbitrary part of the system. Especially when working with a large system and a real robot it is important to keep the focus on the most important tasks. It must be stressed that describing the theoretical solutions to the subtasks involved in developing a mobile manipulator capable of performing masonry is not what is solely aimed for. Furthermore, the implementation of these solutions, the cooperation of the many subsystems of the robot and the actual conduction of the brick laying process by the robot is in focus. These issues, when building autonomous robots that work in the real world, are important and designing the system to act according to them is a central task to this project.

For the proposed solutions developed during the project, simulations have typically been made in order to verify the theoretical usability. This has worked as an important tool since it has then been possible to correct errors before experiencing these when using the physical robot. In general, simulating physical systems has a major potential for reducing the difficulties often appearing when starting to use a new system or employing a new algorithm in reality. The reason for that is that testing special situations as well as long time performance, can then be done in seconds rather than the long time needed to start up machinery in order to set up a specific situation. However, successful simulations alone have not been considered satisfactory, and throughout the project it has been important that proposed solutions could also be tested on the physical robot. Consequently, during the complete project this has placed demands on the hardware and control interfaces to be capable of executing these solutions, sometimes requiring issues to be resolved or improvements to be made beforehand. If the hardware has not fulfilled the requirements it has not been considered acceptable to pretend its functionality.

Throughout the project a wiki[47] has been updated with information about the state of the project. This includes descriptions of the work and results, figures and videos. Particularly, the ability to show videos of the robot performing some action, requires that the robot can actually do so, and therefore, it serves as motivation for pursuing towards the goal. The wiki has also served as a place for storing documentation of the work conducted, which has therefore also required it to be written in an understandable form. The actual act of describing methods and results is often a good way to get new ideas to improve solutions as one during the writing process must ask oneself questions like “Why did we do this?”, “Could we have done something else?” and “Is the result satisfying?”. The material produced for the wiki can be used later as a starting point for this report so the work on maintaining it is not wasted either way.

A way to direct the work and keep the progress running is to divide the task of making this robot prototype into a series of smaller projects all with a defined goal. The goals can preferably be the creation of a video or a demo application as this enforces the project participants to create solutions of a certain quality. The sub-projects are described in the following.

### 4.1.1 Working with real robots

From the beginning of this project, the focus is on the actual use of a real robot, and thus discovering the problems that will arise with such. An unavoidable aspect of this is to make devices work and to repair and maintain them during the project work. The maintenance and repairing tasks are especially big tasks when using devices that are in a development state or produced in relative small numbers. Examples of that are the Universal Robots manipulator, which is a zero series, and the Seekur platform has production number 13. Although repairs of these devices are not intended to be done by the project participants, these tasks cannot be avoided completely, as one must be able to identify things that go wrong or are broken and trace the problem such that it can be identified what has to be repaired. The advantage of having the support of Universal Robots situated 2 km away compared to the support of MobileRobots Inc. situated in USA, proved very convenient, as repair times then could be counted in hours instead of weeks. Although not described further in this report the experience in using, repairing and maintaining hardware gained during this work is considered very valuable.

### 4.1.2 Calibration of the system

In order to make any precise handling of object with the robotic arm, the system need to be properly calibrated. The process of creating the calibration procedure can be continued almost forever as it is always possible to find some part to tune in order to make the method faster or improving the result. Therefore one must be aware of when the demands for the accuracy of the calibration are met sufficiently. An important aspect of this is to make the calibration procedure easy to use as it becomes necessary several times during the project to move the camera and therefore recalibrate the system.

## 4.2 Pilot project

To get started using the mobile manipulator and the vision system as well as the calibrated model of the system, a pilot project was designed. The goal of that is to use many of the devices of the system to identify errors or possible improvements needed. In the pilot project the robot should be able to perform a simple pick and place operation. It was important that the pick-position was obtainable through the use of the vision system, since that would prove the correctness of the calibration result described above. Even more important, this kind of pick and place task should found the basis for later performing the complete masonry task. Although, not directly related to the masonry task, this branch of the project, was considered a suitable possibility for performing some simple action with the robot. As is the case with every attempt to make a robot do some action, problems was still expected to arise even for a simple task, and it would be simpler to trace the mistakes in a reduced scope. The experiences gained and improvements made in the pilot project would then become beneficial at a later stage of the project.

For the simple pick and place operation, small and equally sized cube objects were made from polystyrene and on top of them, a simple but distinct marker was drawn with color pens. The main principle of the pilot project was then to follow the steps below.

1. Place a polystyrene object in front of the robot, visible to the cameras and reachable by the manipulator.
2. Grab an image pair from the stereo camera and display these on the screen.
3. Click with the mouse on the top of the polystyrene object in the window showing the image from the left camera.
4. Extract a small pixel region around the clicked pixel, and perform a template matching with this region in the image from the right camera. The best match should then yield the pixel coordinate of the object in the image from the right camera.
5. Using developed vision algorithms, calculate the 3D coordinate of the object in the left camera frame.
6. Transform the object position to the frame of the manipulator base and calculate inverse kinematics for this position. When calculating the inverse kinematics, the desired manipulator TCP has been set to a position above the object corresponding to a gripper position making the object graspable.
7. Plan a collision free trajectory from the current manipulator configuration to the grasp configuration, move the manipulator according to this trajectory.
8. Close the gripper fingers and plan and move the manipulator back to a predefined configuration.

One important aspect not accounted for in the pilot project, is clearly the complete autonomous identification of the object to be grasped, but a guideline throughout the project has been to get something working before further development and in that process, corners will necessarily be cut.

### 4.2.1 Forskningens Døgn 2010

A fine opportunity to show the capabilities of the robot, when the pilot project had been completed, was the event Forskningens Døgn 2010 where the public is invited into the robotics laboratory, RoboLab, to learn what robots can do and see what projects are currently ongoing. In order to facilitate an inspiring experience for the guests, the pilot project was expanded to enable the robot to move the game pieces of a tic-tac-toe-game around a game board.

The board and pieces were again made from polystyrene and each brick was colored such that the risk of failing in the template matching procedure was minimized. When making a move, a game piece was chosen as described in item 3 of the enumeration in section 4.2 above. The game field at which to place the game piece was chosen as a number in the range 0-8 indexing the nine fields. The place configuration of the manipulator was then found by setting the desired TCP position in a transformation corresponding to the chosen index. That transformation could be found, since it was a requirement that before a game was started, the robot had to be shown the borders of the game board. This was also done very simply by clicking with the mouse on the pixel coordinates of the game board corners and then reconstructing these points to form the transformation between the camera frames and the game board. Again, since the camera-manipulator transformation is known, that was sufficient to arbitrarily moving the game pieces around on the game board.

Several school children visited the event and found the tic-tac-toe playing robot exciting. In figure 4.1 a picture of one of the many games which were played that day is seen.

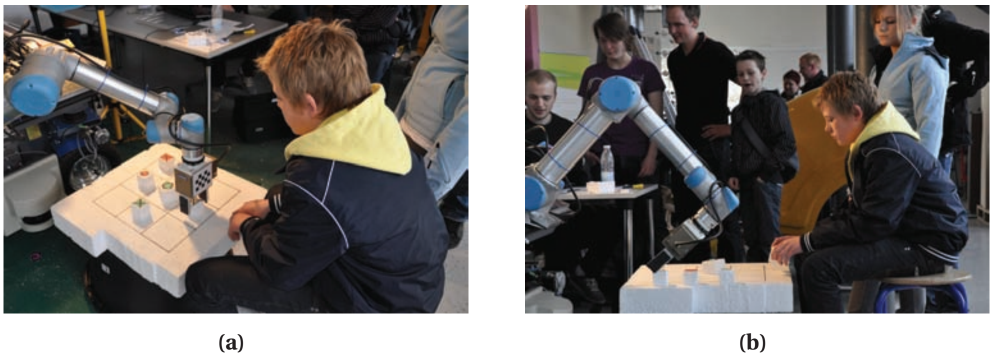


**Figure 4.1:** Children playing tic tac toe with the robot on Forskningens døgn 2010.

Participating in Forskningens Døgn yields experiences about how the robot can be controlled in an environment with persons not used to interact with robots. Since it is essential for performing masonry at a construction site, that the robot should be able to work in such environment, it is therefore necessary to perform these kinds of test along the way. One interesting observation, is that especially kids have very much faith in the robot not to be potentially harmful to anybody. Of course this is up to the developers of the control system to ensure, but robots are typically to some degree considered autonomous units, and still these kids often showed no fear of what the robot could possibly do. Another advantage of participating in this event, is that the robot must be functional at that specific point. Therefore, it is an exercise of being able to keep a deadline, and therefore ensure that the system is in a functional stage. This means that although the system is not perfect, one should rather concentrate on making the best of it, than considering and implementing improvements that cannot be tested in advance.

It would be a interesting task to make the robot able to play the game autonomously. The resulting application and algorithms of such a system would however not likely be useful in the further progress of the project and therefore this was not done. Such a system would require the logic to calculate proper moves in the game, which is a standard example application and solutions like state graph search or reinforcement learning can be found in many books on artificial intelligence, see [37] for example. The problem would however be to make the robot able to recognize the state of the game through its sensors and recognize when and what move the human opponent makes. These are also very interesting tasks, but beyond the scope of this project.

## 4.3 Performing the task of masonry

Performing masonry is a much more abstract task than the above mentioned tic tac toe task, as much more autonomy and planning is needed in order to succeed. Therefore this task is divided into two subtasks, first the creation of a simulation of a successful construction of a wall with the robot and then the task of making this work on the actual robot.

### 4.3.1 Simulations

The preliminary task in creating the simulation of the robot performing masonry is simply to create a pattern generator algorithm, i.e. an algorithm that determines where the next brick should be placed in the wall segment to be constructed. It is a relative simple task, but bugs in that task would definitely make the real masonry task fail, and bugs in this part of the system might be hard to trace.

The actual simulation task involves the development of the algorithm for performing masonry, i.e. an algorithm that creates the appropriate subtasks of moving the robotic arm, grasping and moving the platform, and ensures the execution of these in the correct order. This task has a very well defined goal, that is to be able to build a 1x1 meter wall successfully avoiding collisions. Especially the collision avoidance part is preferable to develop in simulation mode as errors are not fatal, and as many different situations can be tested in a short time. A simulator for the platform is available with the platform software, so simulations of the platform path planning and movement can preferably be tested separately.

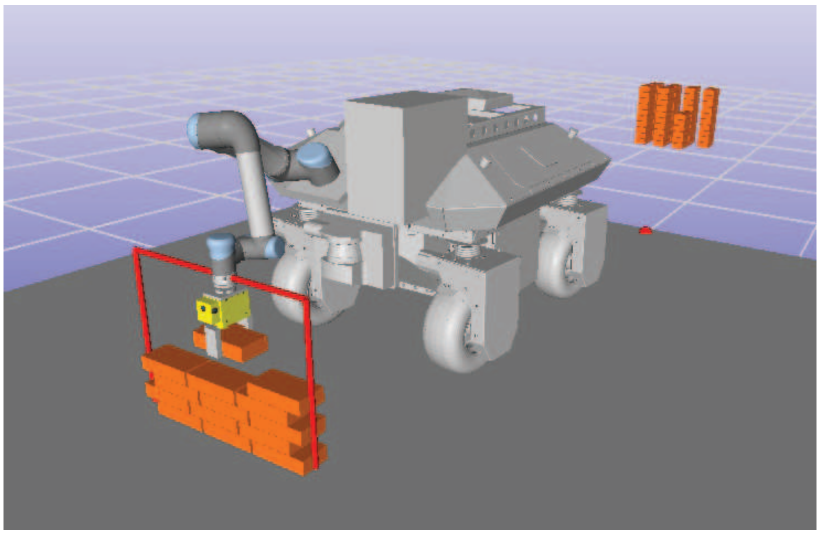


**Figure 4.2:** A screenshot from a simulation of the robot performing masonry.

The main result of this simulation application is the actual successful simulation of the masonry task, but also the ability to test individual algorithms is important. In that way, improvements to the algorithms for the real robot can be tested a priori in the simulation mode. The main drawback of simulations is that it is almost impossible to test all the exceptions, imprecisions and errors that can happen when using the real robot. Instead of spending a lot of effort on modeling these, the simulation mode is intended to be used along with tests on the real robot.

### 4.3.2 Real world implementation

With the sunny day scenario of the masonry properly working in simulation mode, the transition to executing the task on the real robot is much easier. The most important difference between the control of the real robot and the simulated control is the imprecision in the platform movement, and therefore algorithms for estimating the transformation between the platform and the bricks and wall must be developed. This development can however be kick-started, as algorithms for estimating transformations already have been used in the calibration and tic tac toe applications.

A second important task when using the actual robot is to be able to handle exceptions. The manipulator being a zero series model entails that both the hardware and controller software appears in a less mature state than the recent manipulators available from the provider. Also the blend algorithm for the controller, developed in the FORK project and described in 7.3 is expected to contain bugs or errors when exercised more thoroughly. Therefore this task also deals a lot with handling these situations properly as they occur.

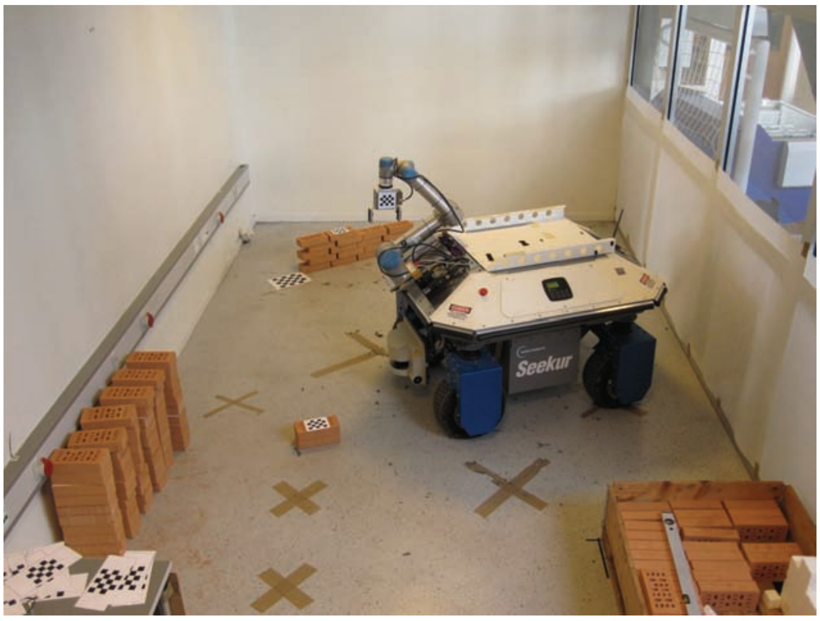


**Figure 4.3:** The testing environment for developing the masonry application.

In order to make testing easy, a testing area for the robot is also created in this part of the project. The testing environment can be seen in figure 4.3 The ability to make quick tests is very important for the speed of the development process, and especially the creation of this controlled scene where objects can be moved as desired made this possible. If too many changes in the scene occur, it will be necessary to create new maps for the platform movement which is a time consuming task that removes the focus from the specific property to be tested.

The development of logging facilities able to log the performance of the robot provided itself useful in diagnosing error causes and evaluating the performance in order to improve the system.

## 4.4 Attending the opening of Odense Havnekulturfestival 2010

A small promotion demo application is developed during the process to be used at the opening of the fair, "Odense Havnekulturfestival 2010". The aim of this application is to create interest for the project and the engineering educations at The University of Southern Denmark. The application itself has little to do with the masonry task. An important aspect of this task is that it should be very robust, meaning the risk of errors and exceptions should be minimal although the scene is unknown. This leads to the following task.

- Move the platform to an interview position on a scene.
- Make an interview with alderman Jan Boye of Odense Kommune by playing prerecorded sound files from a speaker on the robot. During the interview the robot moves a microphone back and forth between the interviewed and the speaker on the robot.
- Wave with the manipulator to the audience and move away.

The tasks involving movement of the platform are solved by tele-operating the platform, and the movements of the manipulator is made of predefined paths in order to perfectly control the flow of actions. Still, the demo should give an impression of an intelligent autonomous mobile robot.

# Chapter 5

# System calibration

This chapter will give insight to the necessity of calibration on different levels of the robot system, and then present the methods used and results obtained for each of the calibrations conducted. Both methods and results will be discussed in relation to the application of performing masonry with the robot. The performed calibrations are a calibration of the stereo camera, a calibration of the Denavit-Hartenberg parameters, describing the kinematics of the manipulator and a calibration of the transformation between the manipulator and the camera. The last calibration uses the two others, and with all three calibrations executed, it is possible to estimate the accuracy of the robot when guided by input from the vision system.

## 5.1 Motivation

Accordance is a keyword when developing robotic control systems. This is because these are typically dependent on some kind of modeling of the real world in which the robot is to be operating. The level of success with which the robot will do so will therefore depend on the level of accordance between the model used in the control system and the real world characteristics. Even though servo mechanisms can be applied in order to reduce the errors that these differences lead to, again, the effectiveness of these methods will also depend of some kind of modeling. Calibration can be said to be the art of adjusting a given model such that it optimizes its accordance with the real world.

For the particular robot system of this project three main models are used to determine the behavior of the robot in operation. The first is the modeling of how the appearance of the real world is mapped to pixel values in the two image planes of the stereo camera. Calibrating this model will lead to the best achievable image representation of the real world, which can then be used to extract information about the real world state. The second model is that of the kinematics of the manipulator which is essential for assuring the correct correspondence between the joint configuration and the transformation of the TCP with respect to the base.

The third model relates the two foregoing. It describes the placement of the manipulator base frame in the representation of the real world as obtained using the stereo camera. When it is possible to precisely describe this last relationship, it will also be possible to determine the manipulator configuration needed in order to correctly position the TCP relative to real world objects located by the vision system. This last aspect is of course essential considering the ability to place bricks precisely in a wall and it is described as a transformation between the frame of the stereo camera, in this case the left camera, and the base frame of the manipulator. When this transformation is known, one can easily transform coordinates of objects seen in the camera frame into the manipulator base frame. This calibration will also determine the transformation between the TCP of the manipulator and the gripper. The important frames for that calibration is seen in figure 5.1.

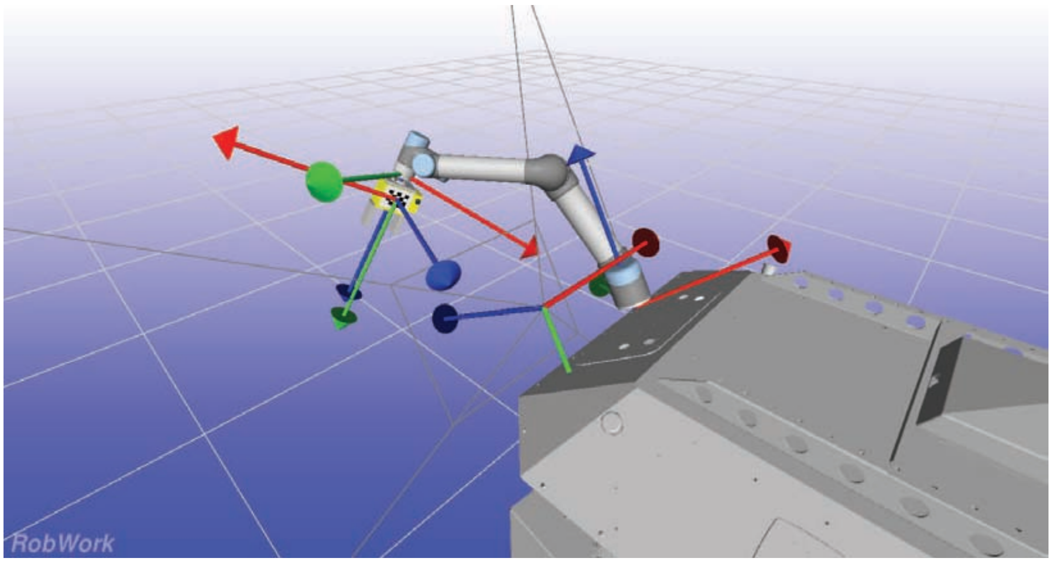


**Figure 5.1:** The four important frames for calibrating the transformations between the manipulator base and the stereo camera and between the manipulator TCP and the gripper. A red arrow points in the positive direction of the x-axis of a frame, a green along the y-axis and a blue along the z-axis. The shown frames are the manipulator base frame, the TCP frame of the manipulator, the frame of the marker on the gripper and the frame of the left camera. The lines originating from the camera frame visualize the approximated field of view of the camera.

The camera is mounted on a pan-tilt unit, placed on the same steel plate as the manipulator. The pan-tilt is controllable through ARIA, but it has not been used in the project, so the calibration procedure will determine the constant transformation between the camera and manipulator. Since it may be needed to unmount and remount or possibly move or adjust the placement of either the manipulator or the stereo camera, which will degrade the validity of the current calibrated transformation and therefore require it to be re-calibrated, an automated calibration procedure is developed for this transformation. The calibration of the stereo camera and the manipulator kinematics are both only needed to be performed once if the results are acceptable, and the methods used for doing so have been developed outside this project and these will therefore not be presented in the same level of detail as the manipulator-camera calibration.

### 5.1.1 Stereo camera calibration

Stereo camera calibration is the art of determining the intrinsic[1] and extrinsic parameters[2] of a stereo camera setup, using the pinhole camera model and a distortion model. Several software tools for camera calibration exist including OpenCV[45] and "Camera Calibration Toolbox for MATLAB"[8]. Both uses the Brown-Conrady[10] distortion model. Camera Calibration Toolbox for MATLAB is used in this project as we have prior experience with that software. The calibration procedure is performed as in the examples on the homepage, first for the left and right cameras separately and then for the stereo setup. An example of a pair of raw images is seen in figure 5.2. Due to the short focal length of the camera, the distortion becomes high near the edges, and thus a high order model is used for the distortion parameters. 116 image pairs of a chessboard were used in the calibration procedure and the result is seen in table 5.1.

[1] Focal length, skew, principal point, a ratio of the horizontal and vertical dimension of the pixels and the distortion model parameters.

[2] Transformation of the right camera placing the left camera in origo aligned with the coordinate system axes.

| Parameter | $f_h$ | $f_v$ | $\mu_x$ | $\mu_y$ | $k_1$ | $k_2$ | $k_3$ | $k_4$ | $k_5$ |
|---|---|---|---|---|---|---|---|---|---|
| Value | 548.93593 | 549.26726 | 518.23785 | 384.65478 | -0.34579 | 0.12787 | 0.00027 | 0.00022 | -0.02239 |
| Error | 0.71165 | 0.70702 | 0.55708 | 0.60303 | 0.00097 | 0.00101 | 0.00010 | 0.00008 | 0.00032 |

**(a)** Intrinsic parameters of left camera.

| Parameter | $f_h$ | $f_v$ | $\mu_x$ | $\mu_y$ | $k_1$ | $k_2$ | $k_3$ | $k_4$ | $k_5$ |
|---|---|---|---|---|---|---|---|---|---|
| Value | 547.61247 | 548.13413 | 511.54863 | 391.80996 | -0.33927 | 0.11929 | 0.00020 | 0.00050 | -0.01919 |
| Error | 0.70764 | 0.70186 | 0.55020 | 0.58846 | 0.00090 | 0.00087 | 0.00010 | 0.00008 | 0.00025 |

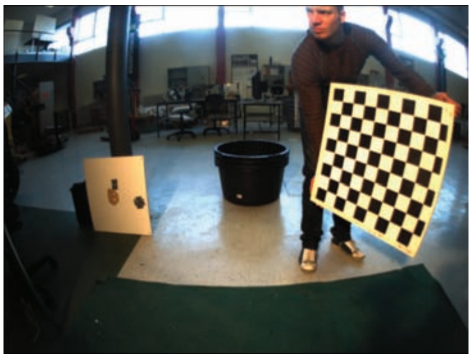
**(b)** Intrinsic parameters of right camera.

| Parameter | $x$ | $y$ | $z$ | $\omega_x$ | $\omega_y$ | $\omega_z$ |
|---|---|---|---|---|---|---|
| Value | -120.13556 | 0.27329 | 0.25384 | 0.00756 | 0.00616 | -0.00195 |
| Error | 0.17342 | 0.16371 | 0.39794 | 0.00045 | 0.00062 | 0.00008 |

**(c)** Transformation of the left camera frame with respect to the right camera frame.

**Table 5.1:** Result of the stereo camera calibration.

$f_h$ and $f_v$ are the focal lengths in the horizontal and vertical axes respectively, $\mu_x, \mu_y$ is the coordinates of the principal point in the image plane, $k_i$ is the i'th distortion parameter and $[\omega_x \omega_y \omega_z]^T$ is an angle-axis representation of the rotation aligning the right camera frame with the coordinate axes of the left camera. Positions of the transformation are in millimeters. The shown errors values correspond to approximately three times the standard deviation for each parameter. The errors are small for the parameters meaning that the confidence in the estimates is high.


**(a)** Left.

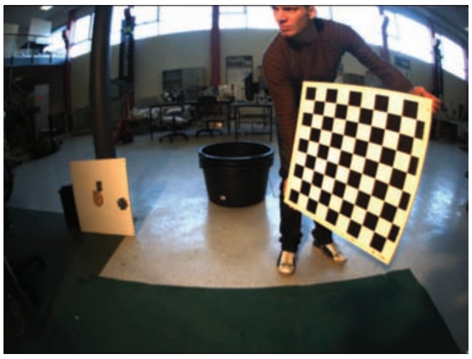

**(b)** Right.

**Figure 5.2:** Pair of raw images for the stereo calibration.

With the distortion model and the transformation between the left and right cameras obtained from the calibration, images can be undistorted and rectified as seen in figure 5.3. As can be seen in the figure, the transformation, which is applied to a raw image when performing undistortion and rectification, causes the middle part of the image to be down-sampled. Contrary, regions towards the periphery are stretched quite much, yielding blurry, erroneous and invalid pixels. In order to preserve the resolution in the middle region and to avoid unusable pixels, a transformation where the down-sampling of areas are avoided is used in this project.

This transformation makes a mapping from the raw image to a new one which is undistorted and rectified and will look like as if it was captured with a camera with specific characteristics. These characteristics are: A fixed focal length of 550 pixel, which has been found to preserve the image resolution of the raw image in the center. An image resolution of 1350 × 950 pixels, which is more than the raw image, but must be so in order to still contain the important and useful image

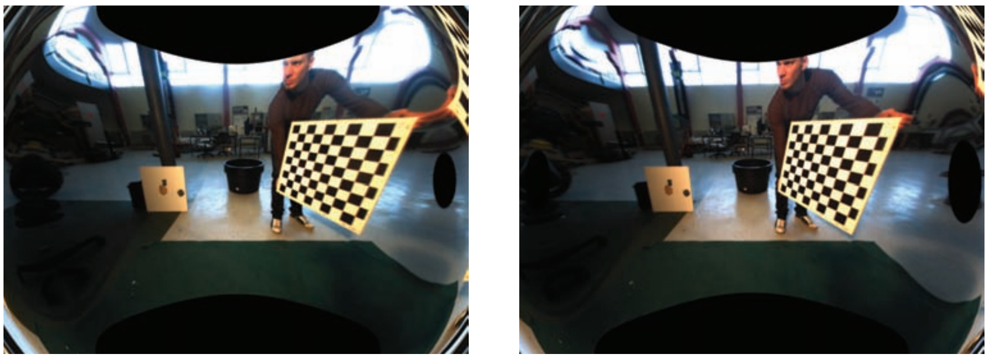

**(a)** Left. **(b)** Right.

**Figure 5.3:** The images of figure 5.2 after undistortion and rectification. The black areas represent invalid pixels.

region, when the focal length is changed. Principal point is placed in the middle of the image. An example of the result of applying this transform to the pair of images can be seen in figure 5.4.

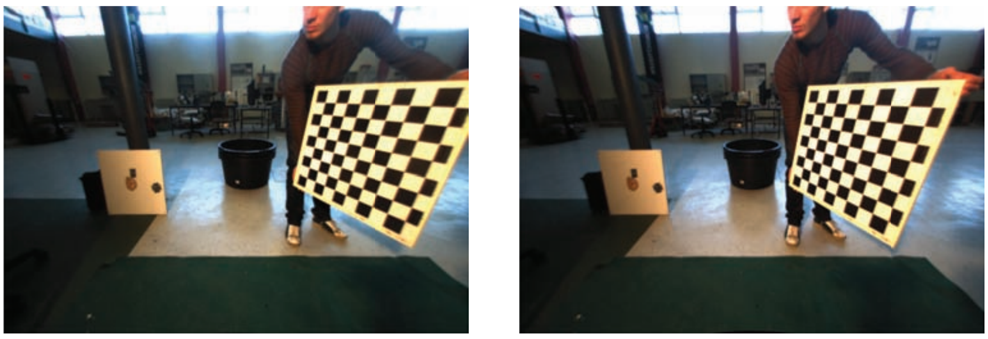

**(a)** Left. **(b)** Right.

**Figure 5.4:** The images of figure 5.2 after applying a transform that undistorts and rectifies the images while still preserving the resolution in the center of the images and discards invalid or too blurry image regions.

The presented transformation is used so that the chance of successful object recognition is increased, mainly by having objects appear more clearly in the middle of the image. One note to make is that OpenCV version 2.1, now incorporates functionality to extract valid pixels after undistortion and rectification, and also to adjust the new image size. This version was, however, not released at the time of implementing the procedure.

### Stereo calibration quality

The calibration procedure works by obtaining the image coordinates of the corner points of the chessboards, by knowing that for each chessboard observed, these points lie in a specific pattern in a plane, the desired coordinates of these points can also be calculated. it is then possible to project these desired points into the image plane, whereas the reprojection error can be calculated. The reprojection errors seen in figure 5.5 gives an indication of how well the calibration fit the data. These errors are the corner point residuals obtained from the calibration. The errors on up to 2 pixels are acceptable for a camera with a high field of view.[3]

[3]The few points at a larger distance are considered as outliers.

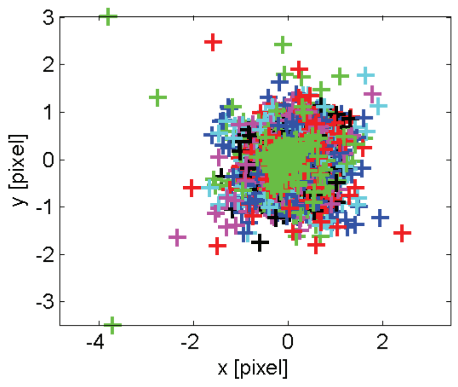


**Figure 5.5:** Reprojection errors from the stereo calibration.

With the short focal length and the relative high distortion near the edges, this also means that the correctness of the estimates of the distortion model parameters becomes highly significant. If the distortion model is poor, it will lead to a wrong mapping of the pixels into the image plane of the pinhole model. And again, that will lead to wrong interpretations about the state of the visible scene. The stereo calibration result presented above is the second one obtained in this project, since the first was not sufficiently accurate.

Although the errors for the parameter estimates of the original camera calibration were quite small, it was experienced that that does not necessarily imply the resulting camera model being accurate. This was observed by doing some simple tests of the 3D reconstruction performance using measuring tape. In the tests, the distance from the camera to a 3D point was measured by this tape and compared to the result for the same distance as obtained using the vision system. This showed differences of several centimeters, which was unacceptable considering the need of grasping bricks detected by the vision system accurately. Although the testing method was very unsophisticated, it still suggested that the camera calibration needed to be improved.

Using this kind of calibration procedure, it can be very hard to identify a problem like this, especially when the procedure on the homepage is followed rather strictly, which is expected to give reliable results. In the particular situation, the reprojection errors showed no signs of a bad model fit. However, this case suggests that with a camera giving highly distorted images, indicating that the pinhole model is relatively far from valid. Therefore a much larger dataset is required in order to estimate the true mapping to that model, when creating this data, attention was put on placing the chessboard in all areas of the images and at different distances and orientations. A convenient enhancement of the “Camera Calibration Toolbox for MATLAB” would be to include a way of supplying a validation set of image based on which, it could be evaluated if a calibration result is good enough, as these errors are hard to detect manually.

### Improving the reliability with a new calibration

For a calibration to be considered satisfactory, two goals were determined, see figure 5.6. First, the accuracy with which 3D distances, here called $x$, could be determined using the camera, should be within 0.2 % of the true distance, $\mu$. The value of this maximal deviation is called $\epsilon_1$. This choice of objective is based on the depth accuracy that is obtainable with the BumbleBee2 stereo camera.[4] A calibration of the BumbleBee2 stereo camera should then be found with which it is possible to reconstruct 3D points, such that the distances between these will not differ more than 0.2 %

[4]On the support pages of [28], an automated calculation of the depth accuracy for a specific camera can be downloaded, `http://www.ptgrey.com/support/kb/data/modifiedstereoaccuracy.xls`.

from the true distance between these 3D points. The intention is then to be able to detect the 3D position of objects with the highest possible accuracy.

The second goal for the calibration, is to reduce the risk of determining 3D position with too large uncertainty. If the calibration is poor it will lead to 3D positions of objects being determined incorrectly more often than for a good calibration. The objective in this matter is not to allow deviations from the true position of objects by more than 1 % in 95 % of all cases. This means that the precision of observations should be high, meaning that if the same object is to be identified several times, the variance in these observation should be equivalently low. The value of this maximal allowable variance is called $\epsilon_2$.

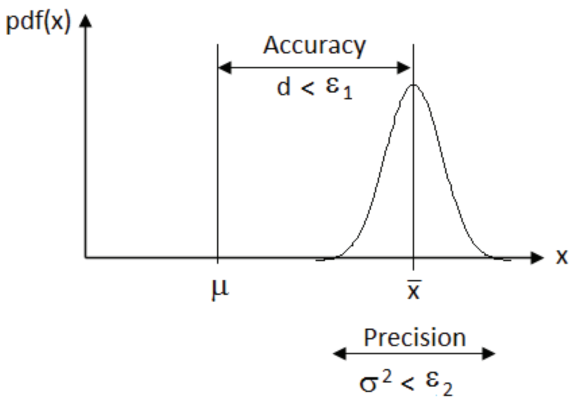


**Figure 5.6:** Illustration of parameters that must meet certain criteria for a stereo camera calibration to be satisfactory.

Both calibration goals may be reached partly by using a larger dataset for the calibration procedure, but it is crucial for high accuracy to have the calibration parameter estimates determined close to the true value, which will require a thoroughly sampled dataset.

An investigation of the calibration quality is performed in order to clarify that the old calibration should be discarded and to evaluate if a new one was acceptable. This analysis is based on the reconstruction of a chessboard with a known square size of 13.4 mm. By recognition of this chessboard in the images from both the left and the right cameras, using the stereo calibration, the transformation of the chessboard frame, which has its origo in the center of the chessboard, relative to the camera frames can be found. See section 7.4 for a detailed presentation of this technique.

Together with the parameters for the optimal pose of the chessboard, also the size of chessboard squares is left as a parameter to estimate minimization procedure. This will then also result in an estimate of the most likely square size of the observed chessboard. The interesting aspect of this analysis is then how well the minimization result of the square size matches the known true value of it. The analysis is based on a sequence of images, for which both the old and the new calibration is used. The results can be seen in figure 5.7.

First of all, by inspecting the two right plots of figure 5.7, the analysis shows that with the new calibration, the distribution of the square sizes determined is centered close to the correct value, $\mu$, of 13.4 mm with a mean of 13.39 mm and a standard deviation of 0.055 mm. Contrary, the data found by the old calibration has mean of 13.48 mm and standard deviation of 0.128 mm, which shows that the square size found using the old calibration must be expected to be wrong. It also shows that the minimization procedure has much more difficulty converging since the variation of the found square size is much bigger than with the new calibration. This last statement is verified by the lower left plot which shows the resulting minimized pixel errors. The upper left plot of figure 5.7 was constructed to investigate whether this error should be expected to be even greater as the distance to the cameras increases. This does, however, not seem to be the case.

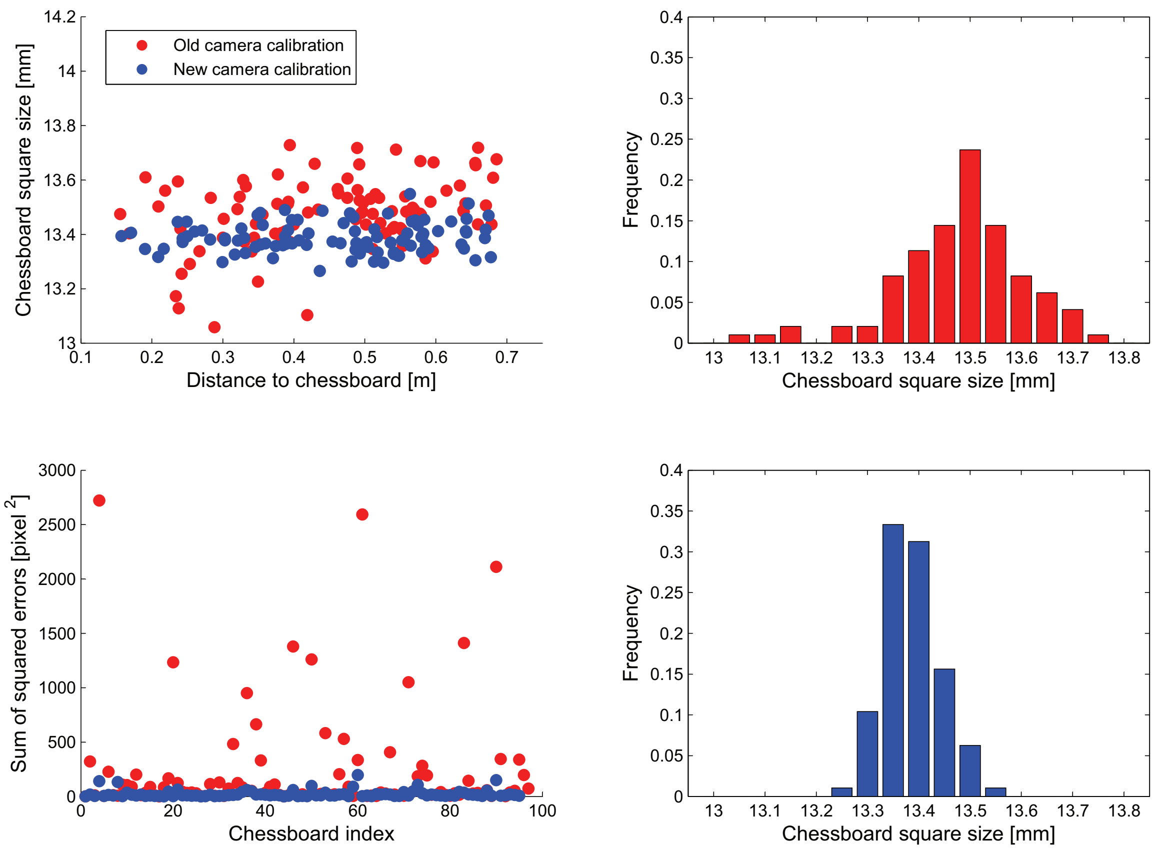


**Figure 5.7:** Results of comparing the old and the new calibrations for the BumbleBee2 stereo camera. Upper left: Found chessboard square size plotted against the distance to the chessboards. Upper right: Histogram of chessboard square size determined with the old calibration. Lower left: Minimized sum of squared pixel errors shown for all chessboards. Lower right: Histogram of chessboard square size determined with the new calibration.

The lower right histogram showing the distribution of the chessboard square size using the new calibration, gives an intuitive way of evaluating the goals for the calibration quality. $\epsilon_1$ will in this specific case evaluate to $0.2\,\% \cdot \mu = 0.2\,\% \cdot 13.4$ mm $= 0.027$ mm. The sample mean, $\bar{x}$, of the found chessboard square size of 13.39 mm gives a $d$ of $\|\mu - \bar{x}\| = \|13.4\text{ mm} - 13.39\text{ mm}\| = 0.01$ mm, which satisfies the objective of accuracy. The second goal is evaluated using a statistical prediction interval which gives the boundaries between which an additional sample with a specified significance level, $\alpha$, will be found. The chessboard square size, $x$, is assumed normally distributed and its mean and variance are estimated by the sample mean, $\bar{x}$, and the sample variance, $s^2$. This way a prediction interval with $\alpha = 95\,\%$ is given as

$$[PI_{lower}, PI_{upper}] = [\bar{x} + s \cdot \Phi^{-1}(0.025), \bar{x} + s \cdot \Phi^{-1}(0.975)] \tag{5.1}$$

where $\Phi$ is the cumulative distribution function of the standard normal distribution, $\sim N(0,1)$. Since it is required that observations should be expected to be within 0.1 % of the sample mean, the prediction interval, which can be solved for $s$ then reads

$$[\bar{x} - 0.01 \cdot \bar{x}, \bar{x} + 0.01 \cdot \bar{x}] = [\bar{x} + s \cdot \Phi^{-1}(0.025), \bar{x} + s \cdot \Phi^{-1}(0.975)] \tag{5.2}$$

which gives

$$\bar{x} - 0.01 \cdot \bar{x} = \bar{x} + s \cdot \Phi^{-1}(0.025) \Rightarrow s = \frac{-0.01 \cdot \bar{x}}{\Phi^{-1}(0.025)} \tag{5.3}$$

With $\bar{x} = 13.39$ mm the maximum allowable standard deviation, $\sqrt{\epsilon_2}$, is then 0.068 mm. The sample standard deviation for the chessboard square size found using the new calibration is 0.055 mm which then satisfies the second calibration goal. The data from the first calibration, that by inspection was considered inadequate, showed an deviance in the accuracy of 0.08 mm, which is far more than the tolerance of 0.027 mm, and a standard deviation of 0.128 mm, which is also higher than the tolerance.

Although it might seem farfetched to accept the new calibration using these goals and the detection of the chessboard square size, it is considered that this was the only way of evaluating the vision system output with respect to a known true quantity. Otherwise it should be possible to make comparisons with the position of a marker placed on the manipulator TCP for which the forward kinematic was 100 % accounted for. However, that will put some dependency on the calibration of the manipulator, and it was a clear desire to evaluate the camera calibration involving as little of other systems as possible. The major dependency of the used method lies on the chessboard recognition algorithm, which of course may give incorrect results if the illumination is bad or if the image is blurred. Still, these circumstances are assumed to have been present through the analysis, which is why the evaluation will also account for these cases.

**Discussion and further enhancements**

It can be discussed if the used BumbleBee2 stereo camera is suitable for the masonry application. The large field of view of course makes it possible to detect objects in the periphery of the robot front end. This is a good thing, since then is it not required that the camera must be pointing exactly in the direction of these objects. Instead, the dependency on the positioning capabilities of the platform is reduced, which is also beneficial since that positioning has shown to possess rather high variance, see section 7.1.2. Even so, the large field of view also implies that much data (light) need to be compressed onto the image plane making each pixel value more unreliable and will therefore be subject to blurriness since there is a higher risk that light originating from several object will affect a single pixel. Therefore, this is expected to cause degraded object detection performance, especially in the periphery of the image. A camera with a smaller field of view would produce sharper images representing more precisely the scene right in front of the camera than the used. Such a camera would therefore lead to more accurate depth estimates of the visible scene. Accurate depth maps (disparity maps) of the scene could be of great importance considering the ability to avoid obstacles in real time.

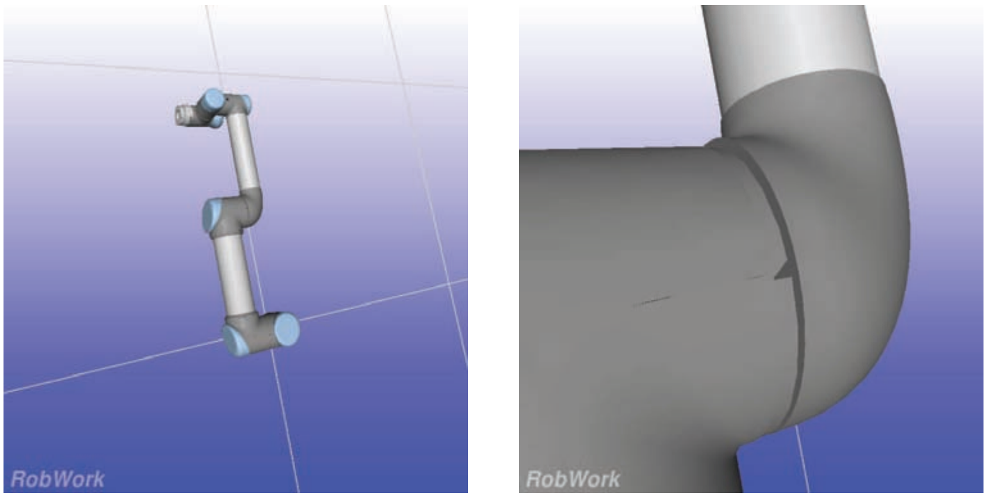
## 5.2 Manipulator kinematics calibration

The calibration of the manipulator kinematics is essential for the precision with which objects can be manipulated. This is because the kinematic model describes the construction of the real manipulator together with the relationship between the actuation of the joint motors and the corresponding movement of the manipulator links. Any error of the model means that the real manipulator will be positioned differently than what is expected when calculating its configuration.

In this project, the manipulator has been replaced several times, changing between different series of the Universal Robots UR-6-85-5-A model. Each of these manipulators came with a more or less precise kinematic model. Before the importance of calibrating the kinematic model was realized, errors were at that time detected when trying to match 3D positions obtained by the vision system with the position given by the kinematic model. Of course, this mismatch can also be related to imperfections of the other two models described in this chapter, either the camera calibration or the transformation between the manipulator and the camera. But it became clear that the calibration of the manipulator was also needed to be precisely calibrated in order to obtain an acceptable accordance between the different systems.

Fortunately, a calibration procedure for the Universal Robots manipulator has recently been developed at the Maersk Mc-Kinney Moller Institute. This procedure was applied to the manipulator currently used in this project. The result from performing this calibration is expected to have reliability since it is based on many samples, and that calibration procedure has previously been used with success. The current version of the calibration algorithm does not contain any direct way of calculating the resulting error in terms of position and rotation of the end effector. This would be convenient since, then would it be known what further uncertainties that will propagate to. [33]

One note to make about calibrating the kinematic model is to be aware that it influences the collision setup. The collision setup which is used for the manipulator in this project is defined as geometries corresponding to the manipulator links, placed in frames corresponding to the manipulator joints. These geometries are based on an ideal model of the manipulator where e.g. joint frames are exactly orthogonal to each other. When the kinematic model is changed slightly due to the calibration, it means that these geometries also will be moved relatively to each other. Although the displacements are small, it is important to be aware that these changes will result in incorrect collision free space in the model. Therefore, there will be a small risk, when using random based planning methods, that the robot is sent through this space, which then will result in self collision. An example of such geometry displacement is seen in figure 5.8.



**(a)** Universal Robots UR-6-85-5-A model. **(b)** The elbow joint at which the geometries are slightly displaced.

**Figure 5.8:** Example of ideal geometries not matching exactly a calibrated kinematic manipulator model.

In order to handle this risk of self collision, it is possible to assure a minimum clearance between geometries when evaluating if a specific configuration is considered to be in collision. Another possibility, which is partly used in this project, is to use geometries for collision checking which has a larger volume than the original geometry.

## 5.3 Manipulator-camera transformation calibration

This section will present the approach used for the calibration of the manipulator-camera transformation and the result obtained. The motivation for performing this calibration is to be able to carry out precise movements of the manipulator with respect to the surroundings of the robot. If an object to be grasped by the robot is recognized by the stereo vision system, it is crucial that the end effector of the manipulator can be placed precisely such that the object is graspable. This is only obtainable if the transformation between the camera frames and the manipulator base is known. The objective of the this calibration is therefore to determine the true value of the parameters in this transformation. In the described version, the pan tilt unit on which the camera is placed is held fixed. It would be possible to incorporate that into the model, but until the demand for active use of the pan tilt unit becomes necessary, this extra complexity is avoided.

### 5.3.1 Approach

In order to obtain a high quality calibration, the approach used in this project facilitate the gathering of many data points to base the inference on. The approach is based on the identification of a known marker in images from the stereo camera system and then compare these observations with the kinematic model. Below, this three-step approach is listed.

1. Locate all marker points in 2D in the two image planes by vision.
2. Calculate the projection of the real marker points onto the image planes using the known configuration of the manipulator and an estimated manipulator-camera transformation.
3. Adjust the manipulator-camera transformation such that distance between the located and reprojected points is minimal.

The vital part of the first step is to find a marker, satisfying the criteria, mentioned above, preferably one where feature identification already exists in software packages. A marker with such characteristics, used in many calibration applications is a chessboard. Each intersection of four squares on a chessboard can be used as a data point, and estimating the point can be precise down to subpixelar levels. This functionality is already implemented in OpenCV, and an example of the use of finding chessboard corners for calibration can be found in [9, chap. 11], in this case for stereo camera calibration. This chessboard identification will give the 2D point estimates $x_{i,j,n}$ for the $i$'th point in the $j$'th camera for the $n$'th observation. See section 7.4 for details of how the identification is done.

The second step is as mentioned to calculate the projection of the marker feature points onto the image plane. The formula for this can be seen in equation 5.4.

$$\hat{x}_{i,j,n} = P_j \left( \hat{T}^{base}_{camera} T^{TCP}_{base,n} \hat{T}^{gripper}_{TCP} T^{marker}_{gripper} X_i \right) \tag{5.4}$$

where $x_{i,j,n}$ denotes the projection onto the $j$'th image plane of the $i$'th 3D point in the $n$'th observation of the marker point $X_i$ with coordinates given in the marker frame. $P_j$ is the projection matrix of the $j$'th camera, found in section 5.1.1. The derivation of the above formula is based on the relationship seen in equation 5.5.

$$T^{marker}_{camera} = T^{base}_{camera} T^{TCP}_{base} T^{gripper}_{TCP} T^{marker}_{gripper} \tag{5.5}$$

$T^{marker}_{camera}$ is found by vision. $T^{base}_{camera}$ and $T^{gripper}_{TCP}$ are unknowns of the system and will be determined in the procedure presented here. $T^{TCP}_{base}$ is determined using the forward kinematics of the manipulator, using the actual joint configuration and the Denavit-Hartenberg parameters for which the calibration is described in 5.2. $T^{marker}_{gripper}$ is manually measurable.

It is desirable to be able to later recalibrate the manipulator-camera transformation, in case of e.g. change of manipulator base mounting or adjustment of the pan-tilt unit on which the camera is placed. Therefore, the marker to use for this calibration is placed on the side of the PG70 gripper, which means that the calibration procedure can be carried out without any disassembly.

Primarily, it is the transformation, $\hat{T}^{base}_{camera}$ we wish to find. Additionally, also $\hat{T}^{gripper}_{TCP}$ must be estimated since its value can only be measured imprecisely due to assembly flanges and mechanics. It would therefore bias the primary calibration result if not taken into account. Contrary, the transformation $T^{marker}_{gripper}$ is considered known since the gripper geometry is known and the marker has been placed carefully relative to this geometry. If $T^{marker}_{gripper}$ contains a little error, it will just cause that $\hat{T}^{gripper}_{TCP}$ compensates for that error without affecting the overall result of the $\hat{T}^{base}_{camera}$ transformation. It proved itself useful to also optionally be able to calibrate the offsets of joint 2, 3, 4 and 5, as these may need to be recalibrated if errors occur in the manipulator. The reason for only calibrating the joint offsets of the middle four joints, is that the base joint offset and the tool joint offset are both not independent parameters. The value of these offsets will namely be included in the determination of the rotation parts of $T^{camera}_{manipulator}$ and $T^{gripper}_{TCP}$ respectively.

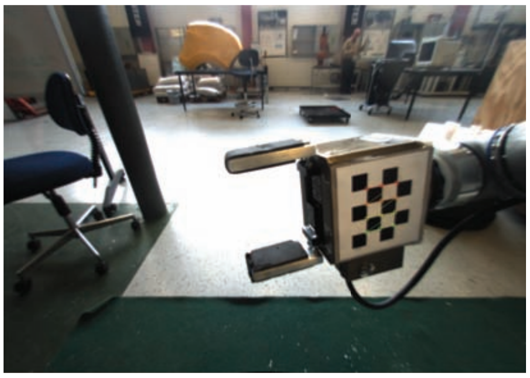
With the above defined projection function, it is then possible to formulate a quadratic cost function for estimates of the parameters:

$$\chi^2 = \Sigma_{n=1}^{N_{obs}} \Sigma_{j=1}^{2} \Sigma_{i=1}^{N_{corners}} \left(x_{i,j,n} - \hat{x}_{i,j,n}\right)^T C_i \left(x_{i,j,n} - \hat{x}_{i,j,n}\right) \tag{5.6}$$

Where $\hat{x}_{i,j,n}$ is calculated as in equation 5.4 and $x_{i,j,n}$ is found by vision. As we now define the cost in the image plane, we then minimize the geometric error which can often be seen as constant throughout the image planes, see [25, chap. 12], making $C_i$ a constant $2 \times 2$ identity matrix.

The cost function 5.6 can easily be transferred into a multidimensional least squares fitting problem. An algorithm capable of finding a minimum of such sum is the Levenberg-Marquardt algorithm [38, 50]. This algorithm is also capable of calculating the covariance matrix for the found parameters. An online free implementation of the algorithm is levmar[36]. To ensure finding the global minimum of the function, it could be possible to run the algorithm several times on the acquired data with different starting guesses. Again, the reader is referred to section 7.4 for more details.

As described in the preceding, a chessboard marker is placed on the side of the gripper. If the manipulator is in a configuration where the chessboard is visible by the cameras, images can be acquired in which the coordinates for the corners between the black and white squares can be determined. In this project, the mechanism for finding these corners is the one implemented in OpenCV. In figure 5.9 an example of a chessboard recognized in an image pair used for the calibration procedure is shown.


**(a)** Left camera view.

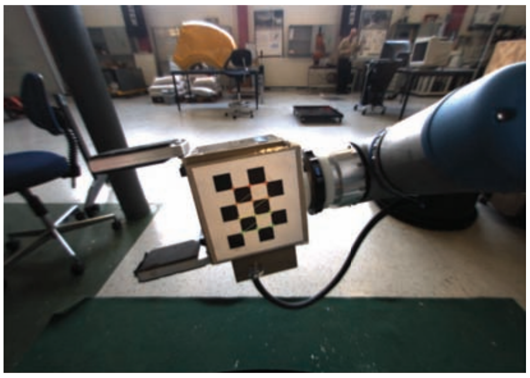

**(b)** Right camera view.

**Figure 5.9:** A pair of images from the stereo camera used to determine the manipulator-camera transformation. The chessboards in the images are identified by OpenCV and these image points are represented by the colored graphics.

What is to be described next is how the automated procedure for gathering the calibration data has been developed and then how this data is applied to the method described above.

### 5.3.2 Data acquisition

The calibration procedure is implemented to automatically collect data for the minimization of the cost function presented in the above section. In general it is desired, for the parameter estimation to be as stable as possible, to have the manipulator joint configurations sampled throughout its joint space. This is sought by generating random samples within this space. However, a large portion of these configurations will result in the chessboard not being recognizable by both the left and right cameras. The reasons for that include that it might not be entirely inside the field of view of the cameras, the orientation of the marker might be pointing away from the cameras, it might be too far for stable recognition or the manipulator links might occlude the chessboard in the images. Finally, a random configuration may be one which is not collision free, or there is no collision free path to this random configuration.

It is desirable to reduce the time spend moving the manipulator if no successful calibration data can be acquired. Therefore, a heuristic is applied before the manipulator is moved to a new configuration. This means that the physical manipulator is kept idle, while evaluating if a proposed manipulator configuration can be used for the calibration process. Until such one is found, the randomly generated manipulator configurations are only evaluated in a virtual environment. This is because intelligently moving the manipulator between reasonable configurations will speed up the data acquisition process by not moving it into intermediate fruitless configurations. The heuristic is described in the following subsections, it is based on a relatively good initial guess of $T_{camera}^{base}$.

#### Ensuring that the chessboard is within the camera field of view

When a random configuration has been generated, the resulting chessboard frame transformation is calculated. Then, in order to increase the speed of the algorithm, the chessboard visibility is first evaluated simply by considering the sign of third entry of the position part of $T_{camera}^{chessboard}$ which must be greater than zero. This makes sure that the chessboard is at all in front of the camera and not behind it.

Next, the 3D points of the corner points of the chessboard are determined for the sampled configuration. The four outermost of these corner points are then projected onto the image planes of the left and right camera, which will give the resulting theoretical pixel coordinates for these corners. These coordinates are then compared with the actual image dimensions, and if not all outermost corner points are within the limits of the image dimensions for both cameras, the present manipulator configuration is discarded and a new one is sampled.

Additionally, a criterion for a random configuration is that it must also be such that the orientation of the chessboard frame has its $z$-axis pointing towards the camera. This makes sure that it is the side of the gripper with the chessboard on it which faces the camera. The angle between the chessboard $z$-axis and a line from the left camera frame origo to the center of the chessboard, must be in the range 150° to 210°, in order for the chessboard recognition algorithm to be successful. This can be ensured by requiring

$$\theta_{max} = -\sqrt{\frac{3}{2}} > T_{camera}^{chessboard}(1:3,3) \cdot T_{camera}^{chessboard}(1:3,4) \tag{5.7}$$

Finally, it is evaluated whether the manipulator links occlude the chessboard. This is done by creating geometries corresponding to the space directly between the cameras and the outermost chessboard corners. These geometries are then added to the collision model, which will then return false if the manipulator (or something else) collides with these geometries, thus occluding the chessboard from the camera views. An example of these geometries not being in collision is seen in figure 5.10.

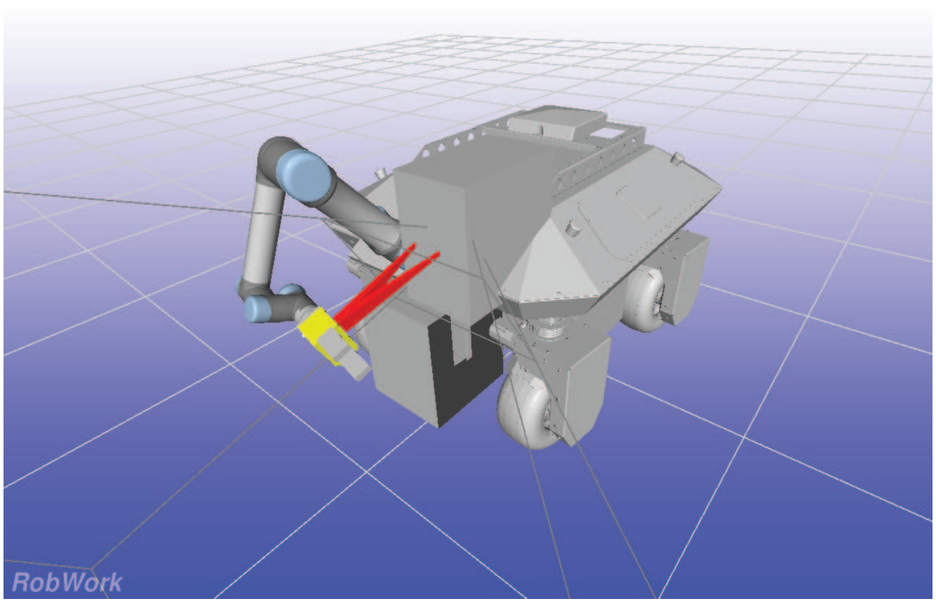


**Figure 5.10:** A random configuration of the manipulator in which the chessboard placed on the side of the gripper is considered visible be the stereo camera. The red geometries are used for checking that the manipulator does not occlude the chessboard.

### Ensuring manipulator movement

When the visibility has been tested and verified for a generated random configuration of the manipulator, the state of robot in this configuration is checked for collisions. Although it might seem weird to delay this check until after the visibility check, it makes the algorithm faster since collision checking is cumbersome compared to the other algorithms in the heuristic. If the configuration does not entail collisions, a path to this configuration from the current one is then searched for. A RRT[5], [35], planner is used for this job, and must also return successfully if the random configuration shall not be discarded. The planner may return unsuccessful if the maximum allowed planning time is reached, which then requires a new sampled configuration.

During preliminary tests of the calibration procedure it was observed, that the constrained joint limits of the manipulator, described in section 2.2.1, makes the planning step a lot more efficient. This is of no surprise since e.g. halving the joint range makes the volume of the configuration space $2^6 = 64$ times smaller. Also the configurations that must be planned to, will in general be closer to the current one, which also simplifies the planning task. Optimization of the path lengths has also provided itself useful for speeding up the data acquisition.

### Acquiring the chessboard data by vision

When the manipulator has reached the new configuration, the stereo camera is invoked to acquire new images. This is done after a wait stage of 3 s which ensures that vibrations in the system after manipulator movement have decreased sufficiently. In both the left and the right images the chessboard corners are then attempted recognized. If this is successfully done for both images, the configuration of the manipulator together with the complete set of chessboard corner coordinates

[5]Rapidly-exploring random tree.

for both images are saved.[6] If the chessboard recognition fails for one of the images, the manipulator must be moved to a new configuration by the procedure described above before a calibration data sample can be obtained. The procedure for data acquisition then runs automatically as long as it takes to successfully collect a predefined number of samples for the minimization and parameter estimation. In this project, 100 such samples were first acquired as the dataset for calibration and then afterwards, a second dataset of 20 samples were acquired for validation. More on validation in section 5.4.2.

### 5.3.3 Analysis of calibration data: Joint values

As described above, only a subset of the randomly generated configurations of the manipulator are valid data for the calibration procedure. To gain a model of the kinematic system which is valid in all configurations, it is necessary to use as many different manipulator configurations as possible in the calibration. This will increase the stability of the model, i.e. making it descriptive for a larger portion of the manipulator joint space.

Figure 5.11 shows that for the calibration dataset, all joints except $q1$ have their joint angles distributed approximately uniformly in the range valid for each joint. This is what was desired for obtaining a stable calibration result. It was expected that especially $q1$ would not comply with this, since only a subset of its joint range can result in the chessboard being visible to the stereo camera.

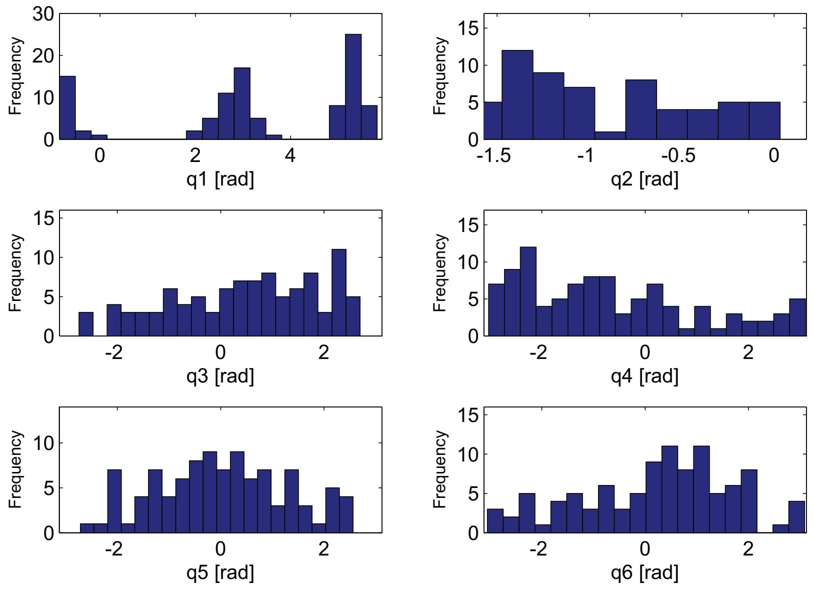


**Figure 5.11:** Distributions of the sampled joint values used for calibrating the manipulator-camera transformation.

If the configurations distribution was viewed in its original six dimensional space and not like in figure 5.11 as projections onto each axis, larger areas of the configuration space would be left unsampled, as only certain combinations of configurations of the joints can create valid calibration data. With the current placement of the cameras with respect to the manipulator, this incomplete sampling of the configuration space is unavoidable. It is, however, expected that the acquired configurations will be adequate for calibrating the system. This is however a reason why calibration of Denavit-Hartenberg parameters is not done this way, although that in theory could be possible.

[6]The configuration to be saved is the one, which can be read from the manipulator controller. Although it is send to the sampled configuration, it might not reached it perfectly and the calibration will therefore be more correct this way.

### 5.3.4 Analysis of calibration data: Image points

It is also interesting to verify that the chessboard is observed in the entire image plane. It is desirable to use the entire field of view of the cameras, as the sampling of the joint configurations is then limited as little as possible. In figure 5.12 it can be seen that the image planes are used fairly well, although the majority of the chessboards are observed close to the centre of the two fields of view of the cameras.

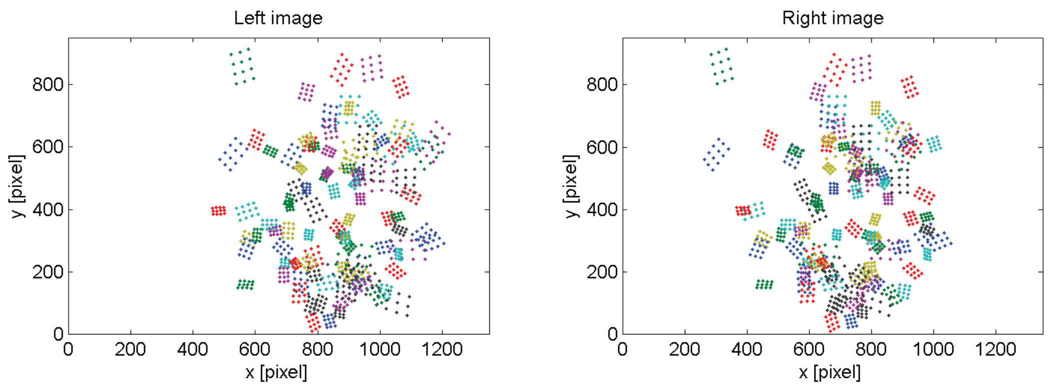


**Figure 5.12:** Distribution of the sampled image points for the calibration. The color coding is used to show corner points originating from the same view of the chessboard. Since 100 chessboards are showed, the same color is used for several chessboards.

It might seem as if the image points are not distributed throughout the image planes, but this is due to the fact that it is a requirement for collecting a calibration sample, that the chessboard must be visible in both cameras. This, together with relative short distance at which the chessboard can be placed, due to the reachability of the manipulator, reduces the range in which chessboard corner points will be observed.

### 5.3.5 Fitting the model

When a number of calibration samples have been acquired, a list containing all the observed chessboard corner point coordinates for both cameras for all samples is constructed. Since the chessboard in total has 12 corners to find, each with a $x$ and a $y$ pixel coordinate, this gives a total of 48 data points for each measurement. For N number of configurations with successful chessboard recognition, there will be N*48 data points as input to the minimization.

Along with this the Levenberg-Marquardt implementation, levmar, must also be supplied a pointer to a function which can generate the corresponding theoretical list of corner points coordinates. This function will use the saved manipulator configurations corresponding to each measurement, and this configuration list is also given as input. For each of these configurations, the theoretical corner points are then calculated and compared with the corresponding observations. Also a vector with the starting guess for the parameters, describing the two transformations to estimate and the optional joint offsets, is provided and these will then iteratively be adjusted until the quadratic cost of the pixel error is minimized. This starting guess is simply found by visual inspection and coarse measuring.

When levmar returns, the best fit parameter estimate together with its covariance matrix is available. Also, one has the opportunity to view the resulting minimum error, number of function evaluations or other information about the minimization. These can be useful for debugging purposes to e.g. confirm that the method converges.

As mentioned earlier, the first found set of parameters is slightly perturbated and then used as a starting guess for another minimization. This is done a few times, and the parameter estimate found with the lowest resulting error is then at last used as starting guess for a final run of the algorithm, but this time the threshold value for stopping at a small parameter change is decreased by a large factor. This is done to assure that the algorithm will not stop as long as the parameter estimates can be adjusted to further reduce the error.

Since *levmar* also gives the covariance for the parameter estimate it is then possible to evaluate the uncertainty of these parameters, which is important in order to make a statement about the precision of a robotic system, which is given in section 5.4.1. When running the developed calibration procedure one has the choice of calibrating the joint offsets or not, which may be beneficial if e.g. only the camera placement has been changed.

### 5.3.6 Results

Since the calibration procedure has been used to estimate both the two transformations and the manipulator joint offsets, the result for all these 16 parameters are presented in this section. The calibrated manipulator-camera transformation and the calibrated TCP-gripper transformation together with the initial starting guesses are seen below in table 5.2 as their respective 6D pose equivalents. The manipulator-camera transformation is the one between the manipulator base frame and the left camera frame. Also, the results for the calibrated joint offsets are seen.

| | | $p_x$ $[m]$ | $p_y$ $[m]$ | $p_z$ $[m]$ | $\phi$ [°] | $\theta$ [°] | $\psi$ [°] |
|---|---|---|---|---|---|---|---|
| $P^{camera}_{manipulator}$ | Start guess | -0.250 | -0.250 | 0.300 | 50.000 | 0.000 | -90.000 |
| | Result | -0.285 | -0.286 | 0.181 | 38.217 | -0.024 | -89.393 |
| $P^{gripper}_{TCP}$ | Start guess | 0.000 | 0.000 | 0.050 | -135.000 | 0.000 | 0.000 |
| | Result | 0.001 | 0.001 | 0.041 | -130.802 | 0.620 | -0.508 |

**(a)** Estimated transformations.

| | | $\Delta\theta_2$ [°] | $\Delta\theta_3$ [°] | $\Delta\theta_4$ [°] | $\Delta\theta_5$ [°] |
|---|---|---|---|---|---|
| Joint offsets | Start guess | 0.000 | 0.000 | 0.000 | 0.000 |
| | Result | -0.571 | 0.003 | 0.777 | 1.024 |

**(b)** Estimated joint offsets.

**Table 5.2:** Results from performing a full calibration of the system. The offsets for joints 1 and 6 are not estimated since these are dependent on $P^{camera}_{manipulator}$ and $P^{gripper}_{TCP}$, respectively. The offsets are therefore included in these quantities.

These final results of the calibration was reached through several iterations of development of the procedure, until an useful result was reached. In each of these iterations, the main concern was to identify the flaws of the procedure, causing the unsatisfactory results. The flaws have been improper stereo calibration, uncalibrated kinematic model for the manipulator, poor detection of the chessboard, vibrations in the platform arising from the manipulator movement giving unreliable data and imprecise placement of the chessboard on the gripper side.

## 5.4 Calibration result evaluation

First of all, it is interesting to inspect the residuals from the calibration. These are seen in figure 5.13, where they have been split into the $x$ and $y$ pixel errors both cameras, such that it can be identified if any systematic error is present.

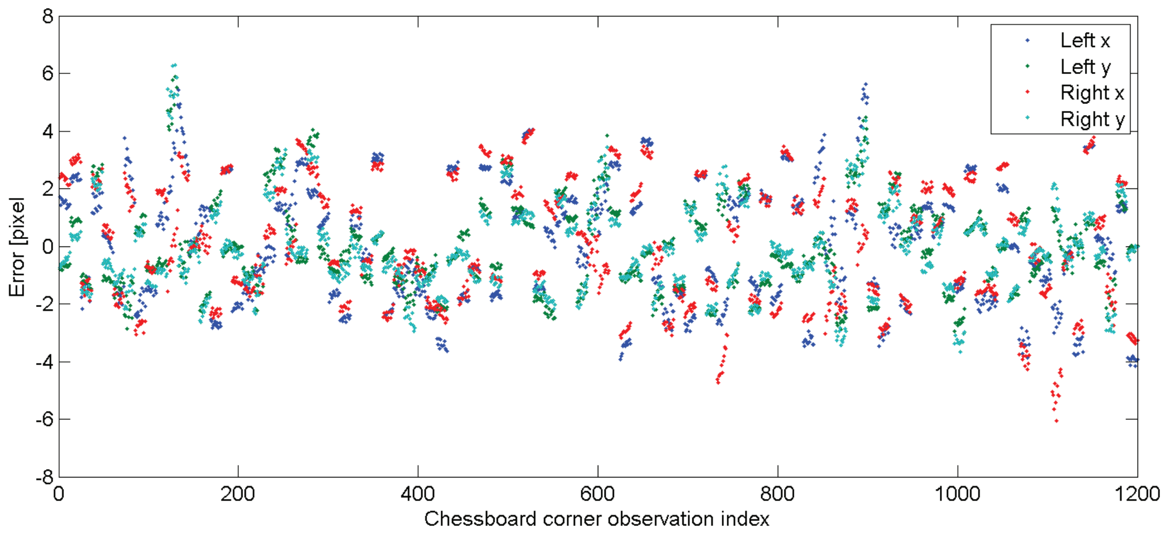
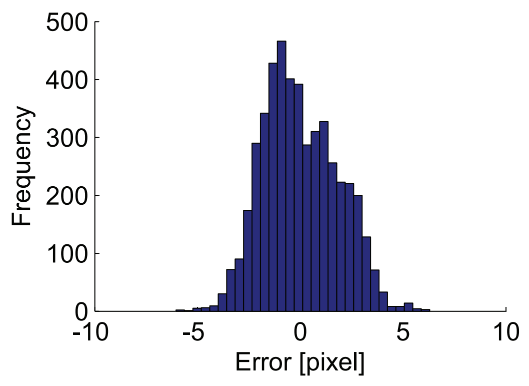
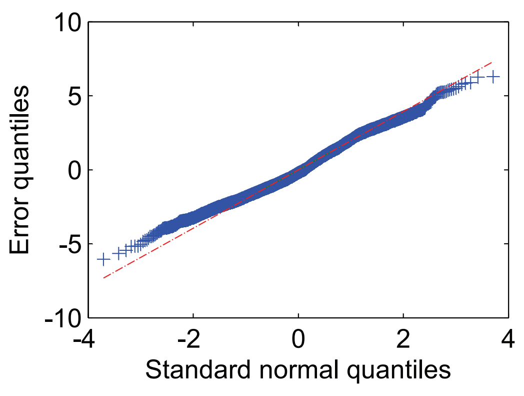


**Figure 5.13:** Residuals from the pixel error minimization of the calibration procedure. The colors indicate affiliation to coordinate and camera.

Ideally, and for a model to be valid, all residuals shall be normally distributed with zero mean. This requirement is evaluated using figure 5.14.



(a) Histogram of residuals.



(b) QQ-plot of residuals.

**Figure 5.14:** The normality of the residuals is evaluated by inspection of their distribution and a QQ-plot of them.

The mean of the residuals is 0.003 pixel and by performing a $t$-test [34, chap. 23] it cannot be rejected at the 5 % significance level that the true mean of the sampled distribution is zero. This is because the confidence interval for that mean at the 5 % significance level is $[-0.049; 0.054]$, thus including zero. Since the QQ-plot compares the quantiles of the residuals with those of a standard normal distribution, it means that if there seems to be a linear relationship between the two, the sampled distribution is also normal. This is verified by the points approximately following a straight line.

By inspecting the plot of the residuals in figure 5.13, it can be seen that a relationship seems to be present between the residuals for the $x$ coordinate in the left camera and the residuals for the $x$ coordinate in the right camera (the blue and red dots follow the same pattern). Also, the residuals for the $y$ coordinate in the two cameras seems to be related. Such relationships are undesirable in residuals for a fitted model, since it indicates that the model does not describe the data sufficiently. This is because the errors can then not solely originate from Gaussian noise present in both the system to model and the measurements. It would be expected that the noise here ideally could be

Gaussian distributed and i.i.d.[7] if it was just measurement noise. To investigate these relationships further, pairs of these residual subsets are plotted against each other in figure 5.15.

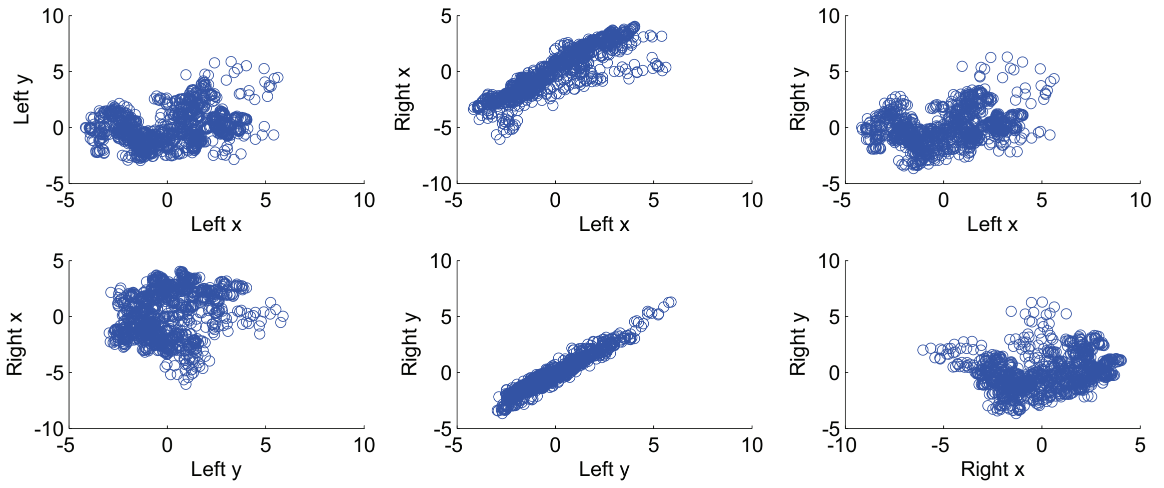


**Figure 5.15:** Residual subsets plotted against each other in order to detect any relationships.

Here, it is clearly seen that the residuals for both $x$ coordinates are linearly dependent as is also the $y$ coordinates. No relationships seem present for the other combinations. If we consider only the $x$ coordinate residuals, it means that the 24 projections, for a specific configuration of the manipulator, of the modeled corner points onto the image planes will most often be either to the left or to the right of the actual observations in both image planes. Thus, a model error or inaccuracy is present since the model cannot describe these patterns. Popular speaking, in those cases, the manipulator is not exactly where we think it is with the current model.

In order to investigate what causes the undesired dependencies in the residuals, these are compared with the independent variable of the dataset for the calibration, namely the configuration vector. We want to investigate if any dependencies are present between these configurations and the residuals. Since the residuals for $x$ in both cameras are highly correlated, as is also the residuals for $y$ in both cameras, this comparison will only split the total residual set in two, one for the $x$'s and one for the $y$'s. The result can be seen in figure 5.16.

The joints $q(3)$ to $q(6)$ show a fine random pattern around zero for the residuals for both $x$ and $y$. For $q(1)$ it must be kept in mind that this joint was not sampled throughout its space as shown in figure 5.11. In the areas for which calibration data was acquired, a slight clustering of the data points for $x$ and $y$ is visible, especially $x$ is mainly positive, negative and positive in the three areas from left to right respectively.

For $q(2)$ a rather clear linear relationship is visible between the residuals and the joint angle. In figure 5.17 this is shown by plotting also the least square regression line.

The confidence intervals for the slopes of the regression lines are given by:

$$\text{C.I.}\left(b_{2,x}\right) = [1.4825\,\text{pixel/rad}; 2.0167\,\text{pixel/rad}] \tag{5.8}$$

$$\text{C.I.}\left(b_{2,y}\right) = [0.2629\,\text{pixel/rad}; 0.8705\,\text{pixel/rad}] \tag{5.9}$$

Thus, given this data, it can be rejected at the 5 % significance level that the true value of either of these slopes is zero, thereby confirming that, indeed linear relationships exist. Since the residuals are dependent on the joint angle $q(2)$, it suggests that although the manipulator prior to this calibration had been calibrated alone, it might not have given a completely reliable result for

[7] Independent and identically-distributed

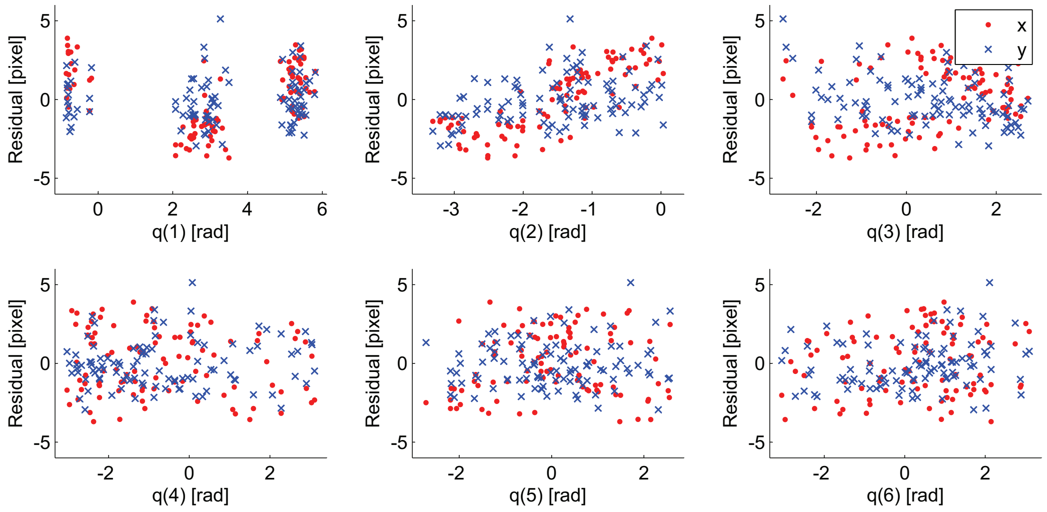


**Figure 5.16:** Residuals split into subsets for $x$ and $y$ coordinates and plotted against the independent variable of the calibration dataset, the configurations of the manipulator.

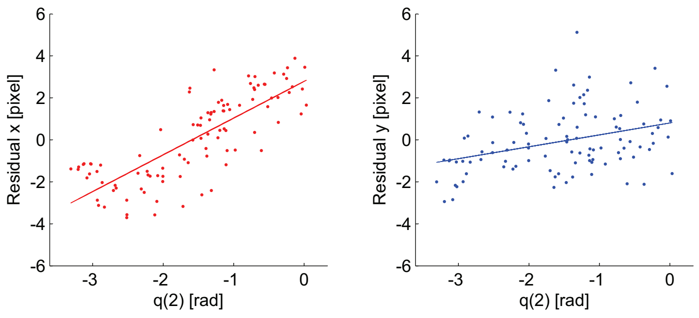


**Figure 5.17:** Residuals for $x$ and $y$ plotted against the sampled joint angles for $q(2)$.

the kinematic model of the manipulator. To further investigate if this is true or not, a method is proposed, which is expected to shed light on this case.

1. Pick a random configuration, $q_i$, ensured to be visible be both cameras, giving a $T^{tool}_{base,i}$.
2. Find the other $j$ inverse kinematics solutions to this transformation.
3. For each found solution, calculate by vision and record $\hat{T}^{tool}_{base,i,j}$.
4. Repeat for many random configurations $i = 1; N$ and observe variation for each set of $\hat{T}^{tool}_{base,i,j}$.

The variance within the sets $\hat{T}^{tool}_{base,i,j}$ will indicate if different configurations, expected to result in the same $T^{tool}_{base}$ will actually do so. Although, there might be differences between $T^{tool}_{base,i}$ and $\hat{T}^{tool}_{base,i,j}$, that will not affect this analysis. This investigation, however, has turned out to be inapplicable for the manipulator currently in use, since there is too much deviation from the desired configurations to which the manipulator should move to where it actually ends up. See section 2.2.2 for this analysis.

An alternative way of dealing with the imperfect calibration result, is to expand the calibration procedure developed in this project, to also estimate this kinematic model of the manipulator. However, based on the procedure by which the manipulator calibration was performed, this is expected to require a lot more data in order to obtain stable results. Furthermore, the narrow sampling areas for $q(1)$ would then be considered an even greater disadvantage. And still, it has not been the focus of this project to obtain a perfect calibration, so further development of the procedure was not prioritized in favor of continuing with other requirements to the system.

### 5.4.1 Parameter uncertainties

The 16 parameters have been estimated simultaneously during the multidimensional minimization process, and the best fit parameter vector $P$ is on the form

$$P = \begin{bmatrix} P^{camera}_{manipulator} \\ --- \\ P^{gripper}_{TCP} \\ --- \\ \bar{\Delta\theta} \end{bmatrix} = \begin{bmatrix} p_x\,[\mathrm{m}] \\ p_y\,[\mathrm{m}] \\ p_z\,[\mathrm{m}] \\ \phi\,[^\circ] \\ \theta\,[^\circ] \\ \psi\,[^\circ] \\ -- \\ p_x\,[\mathrm{m}] \\ p_y\,[\mathrm{m}] \\ p_z\,[\mathrm{m}] \\ \phi\,[^\circ] \\ \theta\,[^\circ] \\ \psi\,[^\circ] \\ -- \\ \Delta\theta_2\,[^\circ] \\ \Delta\theta_3\,[^\circ] \\ \Delta\theta_4\,[^\circ] \\ \Delta\theta_5\,[^\circ] \end{bmatrix} \tag{5.10}$$

The value of the best fit parameters are printed in table 5.2. However, in this section we are interested in the reliability of these best fit values. As described previously, an output of the *levmar* minimization routine is the covariance matrix for the fitted parameters. For the calibration including the joint offsets, this covariance matrix is given in appendix A. The parameter covariance can be used to evaluate the uncertainty of the parameter estimates since it describes the standard error for these parameters, see [50, chap. 15]. The covariance matrix is here used to construct a simultaneous confidence interval in order to evaluate the magnitude of these standard errors. Simultaneous means that with a confidence level of $\alpha$ all elements of the true parameter mean vector will lie inside a certain interval. The formula for the confidence interval for the i'th element of the parameter vector is reprinted below from [32, equation (5-24)].

$$\bar{x}_i - \sqrt{\frac{p(n-1)}{n-p}F_{p,n-p}(\alpha)}\sqrt{\frac{s_{i,i}}{n}} \le \mu_i \le \bar{x}_i + \sqrt{\frac{p(n-1)}{n-p}F_{p,n-p}(\alpha)}\sqrt{\frac{s_{i,i}}{n}} \tag{5.11}$$

Here $n$ is the number of observations, $p$ is the number of parameters, $F_{p,n-p}(\alpha)$ is the $\alpha$'th percentile of the $F_{\nu_1,\nu_2}$ distribution with $\nu_1 = p$, $\nu_2 = n-p$ and $s_{i,i}$ is the $i$'th diagonal element of the covariance matrix. This gives the following simultaneous intervals with $\alpha = 0.05$.

$$\text{C.I.}_P = \begin{bmatrix} -0.2850\,\text{m} & -0.2849\,\text{m} \\ -0.2864\,\text{m} & -0.2863\,\text{m} \\ 0.1806\,\text{m} & 0.1807\,\text{m} \\ -1.4969\,^\circ & -1.4967\,^\circ \\ -0.5190\,^\circ & -0.5188\,^\circ \\ 0.5237\,^\circ & 0.5239\,^\circ \\ --- & --- \\ 0.0007\,\text{m} & 0.0008\,\text{m} \\ 0.0008\,\text{m} & 0.0010\,\text{m} \\ 0.0406\,\text{m} & 0.0406\,\text{m} \\ 0.0069\,^\circ & 0.0085\,^\circ \\ 0.0148\,^\circ & 0.0168\,^\circ \\ -2.2843\,^\circ & -2.2814\,^\circ \\ --- & --- \\ -0.0100\,^\circ & -0.0099\,^\circ \\ 0.0000\,^\circ & 0.0001\,^\circ \\ 0.0134\,^\circ & 0.0137\,^\circ \\ 0.0177\,^\circ & 0.0180\,^\circ \end{bmatrix} \tag{5.12}$$

Representing the confidence interval as an ellipsoid would be more correct than a 16 dimensional hyperrectangle, but that result would be very hard to present in an intuitive way. Here each interval is the ellipsoid projected down on an axis.

It is observed that the largest uncertainty with respect to the positional entries of the parameter vector is 0.2 mm, which is a sufficiently small range to be acceptable for the masonry application. For the rotational entries of $P^{camera}_{manipulator}$ and $P^{gripper}_{TCP}$, the largest uncertainty is 0.003° which is also acceptable. The uncertainties for the joint offsets are all smaller. Therefore, the confidence in the fitted parameters is considered high. However, the fitted model should be tested on a new independent dataset, which is done next.

### 5.4.2 Validation

A validation of the calibration result is needed in order to check whether the result is reliable, meaning that the data to which the model was fitted, was a representative sample of the population. This is tested by acquiring a new dataset, and then using the previously fitted model, calculate the errors between this model and the new dataset. Then, it is possible to compare the residuals from the fitting of the model with the errors obtained with the new data. For the model to be acceptable, these two types of errors should be similar. A plot of the errors to the fitted model obtained with the 20 sample validation dataset is seen in figure 5.18.

The errors from the validation set are nicely distributed around zero, and seem to follow the straight line of the QQ-plot fairly well. Also a $t$-test verifies that the it may come from a zero mean distribution with a $p$ value of 0.267. However, as mentioned above it is important that these errors are comparable with the residuals from the calibration. This is investigated by performing a $F$-test, which test the hypothesis that two samples have equal variance. Doing this unfortunately rejects this hypothesis, by stating that the confidence interval for the fraction between the two variance is $[1.671; 2.033]$, which should desirably include 1.

This along with the systematic error in the residuals, reveal that the model has some flaw, as the dependency on especially joint angles of $q(2)$ also indicate. This means that the calibration cannot be perfectly valid, as it requires the distributions of both the residuals of the calibration data and the validation data to come from the same distribution. Although adding more data

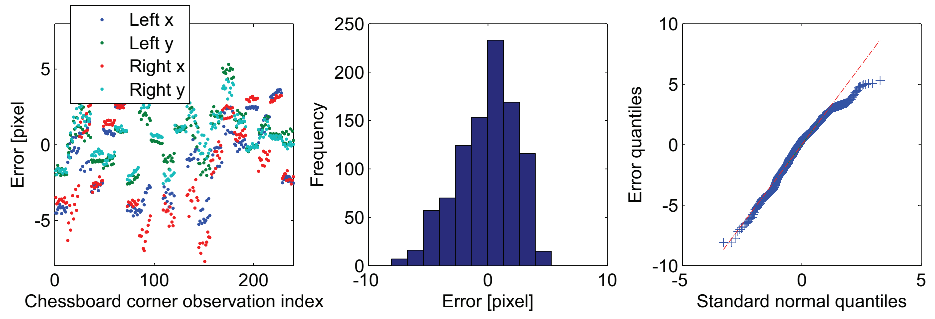


**Figure 5.18:** Errors from validating the calibrated model with a 20 sample dataset. Colors indicate affiliation to coordinate and camera.

to the calibration will not correct the model, it will in practice, however still possibly reduce the validation error, as the validation dataset can be expected to look more like the calibration data.

## 5.5 Evaluation of system accordance

Although the calibration of the manipulator-camera transformation etc. still contains some minor flaw, it is possible to evaluate the calibration results of the system as a whole. In order to do this, an investigation in which the accordance between visual observations of the chessboard marker used for the calibration and the expected pose of it, calculated by forward kinematics through the manipulator, is verified. The chessboard corner points which were used to obtain the calibration result are now used to estimate the chessboard frame pose using the methods of section 7.4. The found 6D pose for all chessboards in the dataset is then compared with the 6D pose stating the chessboard frame transformations given by the kinematic model. In figure 5.19 the frames to compare transformations of are shown together with the chessboard corner points they relate to.

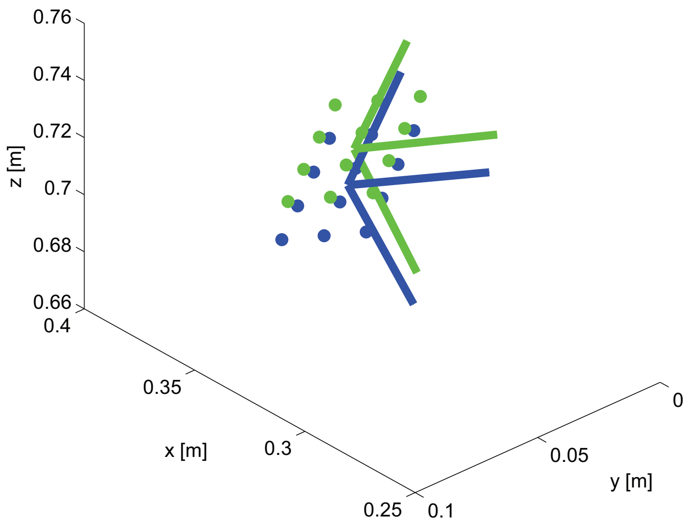
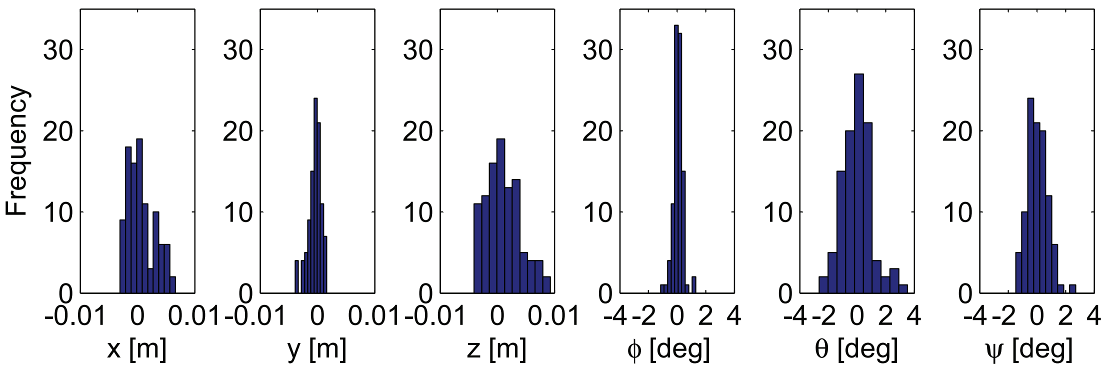
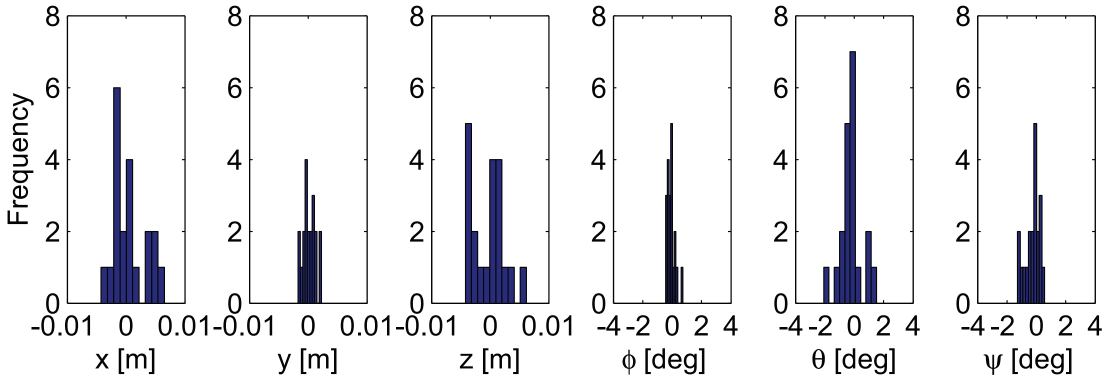


**Figure 5.19:** Example of two chessboards shown by their corner points and a frame. The pose of the green chessboard has been found as a least square fit to its reconstructed 3D corner points generated by vision and shown with red. The blue chessboard is positioned according to the state of the kinematic model.

Two equal analyses are made based on 1) the 100 calibration samples and 2) the 20 validation samples. The differences in each of the six pose coordinates between the two frames are recorded and below the result is visualized in figure 5.20 by histograms showing the distributions of these differences.



**(a)** Evaluation of reconstruction, calibration data.



**(b)** Evaluation of reconstruction, validation data.

**Figure 5.20:** Differences for each parameter of 6D poses between chessboard frame transformations determined by vision and the same transformations determined by the calibrated kinematic model.

As desirable, in this analysis the error distributions are comparable for the two datasets. By looking at the errors for the validation dataset, the standard deviation for the correspondence in the positional part is at most 2.9 mm and based on the foregoing evaluation of the calibration, it cannot be expected that better correspondence will be present without further enhancement of the calibration procedure. The rotational correspondence between the frames is also sufficient, in that the standard deviation at most $0.78°$.

The magnitude of these errors, has found the basis for not considering performing a recalibration or continuing the development of the procedure. This is because, errors of these sizes are not expected to cause failure of neither grasping objects, nor placing these.

# Chapter 6

# Software system design

The robot system contains many physical devices, each of these needing real-time software to control them. Therefore, a vital part of the project is to design the software controlling each device, and especially the software that in a reliable and fast way communicates with the devices. The software system is the thing that glues the whole system together to make it act as a whole. This chapter contains descriptions of the goals for the software design, stated as guidelines for the development process, the software architecture, some implementation specifics about the use of open source software and a more detailed description of the software components in the architecture. The aim of this description is to make it possible to use and evolve the software developed in this project as well as to discuss the design decisions made during the process.

## 6.1 Design goals

The goals of the software design is to make it possible to control a complex robotic system with many sensors, actuators and demanding algorithms in real-time. This means that the system should be able to respond quickly and wisely to events from devices while still processing high level planning and calculations for the actions of the robot. When making software for a development project, an important aspect for the choice of software architecture and design, is to structure it in a way that facilitates further evolution and application development. This demands that the developed source code is easy to comprehend such that it is easy to find and modify features in the code without breaking the entire program. An important aspect is also to make tests and simulations of the system possible with minimal or no changes to the algorithms, compared to when controlling the physical robotic system. This is because simulated tests are much faster and easier to perform than tests on the real robot. To use simulations efficiently, it should be easy to translate the control algorithm from an successful simulation into a control for the real robot.

## 6.2 Software architecture

A widely used pattern in software architecture design is to divide the software into tiers. With such an architecture it is possible to develop and modify the tiers independently without having to correct the other tiers as long as the interfaces between them are kept. The different tiers have different responsibilities, and an illustration of the tiered system architecture can be seen in figure 6.1.

### 6.2.1 Device control

Starting bottom-up the device control tier contains the drivers and interfaces for the actuators and sensors. The current system contains four device control servers for the platform, robotic

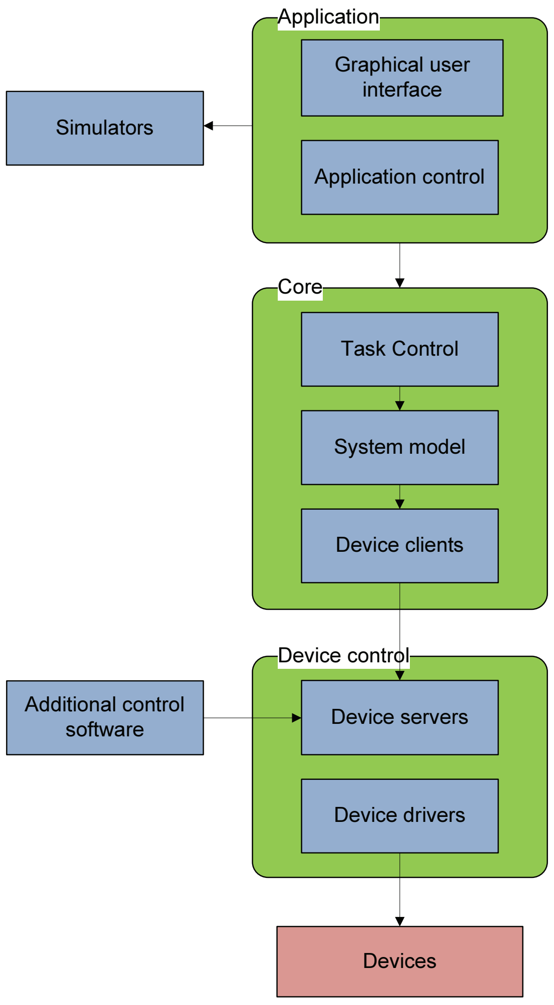


**Figure 6.1:** The software architecture of the software system controlling the mobile manipulator. The green boxes are the tiers of the architecture, the blue boxes are the software components and the red boxes represent physical devices.

arm, gripper and stereo camera, respectively. These all interface to the rest of the system by creating a networking server application that is accessible over TCP/IP. This design facilitates that applications running on different computers with different operating systems are able to connect to the servers and communicate with them. The servers vary much in complexity, as one must have in mind both the complexity of the device they must control and the computation power on the computer on which they are residing. The additional control software noted in figure 6.1 are clients for the robotic arm controller and the platform controller developed by the respective manufacturers.

### Gripper

The server for the gripper is just a simple algorithm that translates incoming requests into requests that can be sent over a CAN interface to the PG70 Gripper[58]. In the current implementation, only commands to close or open the gripper with a fixed force are available, but these are sufficient for performing grasping in the current applications.

#### Robotic arm

The controller for the Universal Robots UR-6-85-5-A robotic arm[56] runs a proprietary server[1] that is accessible over TCP/IP. The controller server continuously sends packets to connected client sockets containing status information about the current state of the manipulator. These info are the same as are used in the "PolyScope" program [65]. The robot is controlled by sending script, see [66], programs to the controller server which can be executed on the robot. The easiest way to dynamically update the program executed on the controller is then to send a new script to execute when necessary. In this application scripts usually contain instructions to move the robotic arm along specified path, using the blend algorithm, described in 7.3.

The "PolyScope" program is used to initialize, setup and to survey the performance of the robotic arm, and it can be run in parallel with the developed applications.

#### Platform

The Seekur platform does as standard contain several devices - the eight motors for controlling the platform movement, a pan-tilt unit for cameras, a laser range scanner and a inertial measurement unit (IMU). These can all be interfaced and controlled by the software, ARIA from MobileRobots Inc. A standard server using ARIA is distributed with the Seekur platform. In this project the software ARNL[41], which is capable of using the laser range scanner data for localization and path planning tasks, is also used, making the platform server able to support both interfaces. A server using both libraries is used in this project. The basic interface to the platform is then to specify a goal pose of the robot, meaning an x, y and orientation parameter which is sent to the server. The server then generates a path to move to that goal. The path is updated if obstacles are detected and the laser range scanner can be used to verify if the goal is reached. More information on how to use and extend this software can be found in [46].

#### Camera

The server for the BumbleBee2 stereo camera currently captures images in single-shot mode when requested by the client application. When images are captured they are compressed by a jpeg transform to decrease the amount of bytes to send per image and to decrease the delay until the client application receives the images.

It is due to the limited processing power on the onboard computer on the Seekur platform that further image processing algorithms and undistortion and rectification transforms are not executed there but on the remote computer hosting the client application. In a future version of the system, it would be preferred to place an extra computer, possibly with a fast graphics processing unit on it, on the platform with the responsibility of capturing images and making image processing. It would then be possible to make an interface that returns more abstract information such as the 3D position of an object or a depth map of the area reachable by the robotic arm.

### 6.2.2 Core

The core software of this project is the code that can be used in several applications, as it contains more general code for motion planning, grasping, vision algorithms, task execution and system state updating and representation. This part of the software is common to the applications that are and can be developed for the robot. This means that a lot of code is shared between for instance the robot arm calibration application, described in section 5 and the wall building application,

[1]The source code for the server has actually been made available in order to develop the blend algorithm, described in 7.3.

described in chapter 7. This has the advantage that during development of one application, the other applications are also improved.

#### Device clients

Each device server, described in the previous section needs a corresponding client, able to send requests and handle callbacks. The clients for the robotic arm and platform servers run in their own threads as they are responsible for retrieving the periodic status packets that are sent from these servers. The client threads are then able to notify the rest of the system of the status of the devices and if a task has been executed successfully or not.

#### System model

This part of the system contains abstract representations of the entities in the system, for example, the robotic arm, a chessboard pattern or the calibrated camera. The classes in the system model increases the level of abstraction for the hardware, as it for example implements planning algorithms for the robotic arm and vision algorithms for locating chessboards.

#### Control

This part of the system handles the tasks, in this system a task is a simple sequence of instructions that can be performed by a device, like moving the robotic arm along a specified path or closing the gripper. A task controller is implemented that handles the execution of tasks or creates data to visualize a simulation of the tasks. The task controller is designed to run in a different thread than the main application, in that way the main application can generate tasks for the task controller without having to wait until execution is done. For further details see section 6.5.

### 6.2.3 Application

This tier contains the user interface and algorithms specific to each application. The major applications are made with a graphical user interface, as a plugin to RobWorkStudio [57]. It helps the development as it is possible to use the capabilities of RobWorkStudio to display simulations of the robot and display the transformations of the frames defined in the program. The work cell, collision model and frames can be loaded into the simulator.

## 6.3 Implementation

A way to speed up the development process is to make use of existing software. This speeds up the development process by delivering the code needed, and it is also very likely that parts of existing software contain less bugs than new developed code. When using existing software and third party libraries it is favorable that the code is open source as it is then possible to inspect or even modify the used software. Much open source software is developed in C++, often for Linux environments, this is also an advantage in this project as the controllers for the platform and the robotic arm is running on Linux machines.

The software is in general implemented object oriented which makes the large number of code lines more comprehensive, such that it is easier to find and modify special features of the program. The program is compiled into several libraries, which makes it possible to use only a subset of the developed code for some application.

The software system for controlling the robot is distributed among several computers. The communication between these is done over a wireless TCP/IP connection. To make this work with reasonable response times, clients and servers are implemented as threads, able to read and send data asynchronous from the main application. Threading is implemented by the Qt[43] library which also uses “signals and slots” as a convenient way of message passing between threads. Threads are also used to avoid letting the whole system wait for some device to finish execution or some algorithm to return.

### 6.3.1 Open source and licensing

Open source libraries and frameworks are used almost whenever possible in this project. When used properly, this minimizes the amount of code needed to be written, and thus reduces the complexity of the code. The most important third party libraries and frameworks used in this project are listed in the following.

- **Qt** - The user interface framework Qt is used for GUI development, threading, inter-process communication and networking, as it has high level classes available for these topics. The framework has the advantage that it is cross platform, enabling the applications to work on both Windows and Linux based machines. Qt is already used by the simulator RobWorkStudio, so interfacing with that is also made easier.
- **RobWork** - The robotics library RobWork[57] is used for forward and inverse kinematics of the robotic arm, describing the work cell of the robot, collision checking and path planning for the robotic arm. The library is developed at the university so the possibilities for getting support or even getting demanded features added to the library are very good.
- **RobWorkStudio** - A robot simulator using RobWork and Qt. RobWorkStudio simulates and displays paths for robots, and it is possible to move the camera view as well as objects and devices inside the scene visualization.
- **OpenCV** - A multi-purpose computer vision library. OpenCV is used for chessboard recognition, undistortion and rectification of images and stereopsis.
- **levmar** - levmar is an implementation of the Levenberg-Marquardt optimization algorithm used for parameter-fitting and 3D reconstruction, used in several parts of the system.
- **ARIA** - The robot control library for the Seekur platform, ARIA, provides motor and sensor control as well as a networking interface.

An important topic when using third party libraries is licensing. Open source libraries come with many different software licenses stating terms and conditions on the usage of the software. These terms are important to know if the software is to be distributed. If for example the software is to be shipped with a commercial product, using a library with a copyleft license like GPL[2][17] entails the software to be released under the same license, and thereby making the source code public available. If however the open source software used has a permissive license, it is legal to release the software under a different license, making it possible to sell the software commercially. In this project only the libraries levmar and ARIA are under a copyleft license (GPL), the remaining libraries all have permissive licenses.[3] This means that levmar will have to be exchanged with a different one, for example Minpack[14], and another license must be acquired from MobileRobots Inc., the copyright holder of ARIA, in order to release the software of this project commercially. Keeping the software open source is however also an opportunity, as if it is released as such, it is easier for other people to add functionality to it. An interesting business model is that of MobileRobots Inc. where some software is public available and some is sold commercially with a robot.

[2]GNU General Public License.

[3]Qt is licensed under LGPL[18], RobWork and RobWorkStudio under Apache 2.0[2] and OpenCV under a BSD license.

## 6.4 User interface

An important part of the autonomous robot system is the interface with the user. As it is still a development project, most of the current graphical user interface is designed for that, but ideas on how the user interface will look to the end user will also be presented. A screenshot of the graphical user interface, based on RobWorkStudio can be seen in figure 6.2.

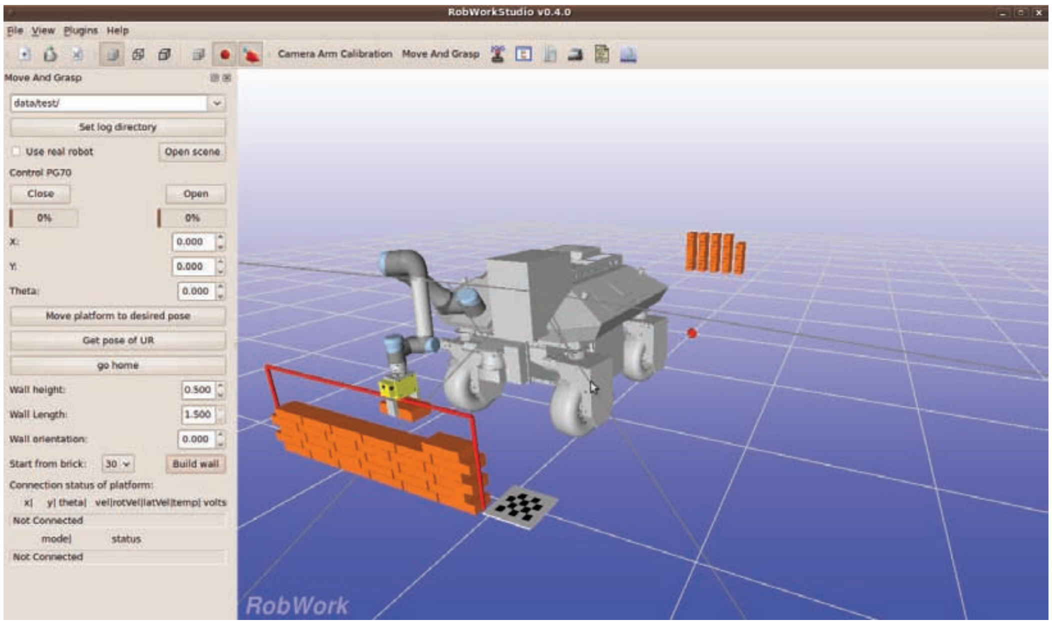


**Figure 6.2:** The graphical user interface of RobWokStudio with the plugin for construction of walls visible.

The main area of the user interface window is the simulator view, and in this window it is possible to see simulations of the building process. It is also possible to see the dimensions of the wall and where it is being build with respect to a chessboard marker. Making this visible should make it easier to check with this simulator that the chessboard marker indicating where the wall is to be build is placed in the correct place. The left part of the view is the plugin for controlling the building process. It is not interesting to explain all button and fields, but the most important of these are described. The checkbox named “Use real robot” is very important, it determines if the robot is going to build a wall in simulation or by connecting to the real robot. When checked and unchecked the robot connects and disconnects to the servers controlling the physical devices of the system.

The fields “Wall height”, “Wall length” and “Start from brick” as well as the button “Build wall” are the important buttons for deciding the size of the wall. When the size is changed, the red lines in the visualization move correspondingly. It is possible to continue an already started building process by setting the “Start from brick” correctly. The actual wall building process is started by pressing “Build wall”.

An important feature, especially when debugging, is to be able to monitor the status of the connected devices. This is done by watching the fields under the text “Connection status of platform” and by pressing the button “Get pose of UR”, which visualizes the actual configuration of the manipulator.

The top area of the view contains other plugins to RobWorkStudio. Most of them are used to move and visualize devices and objects in a model of a scene. The button “Camera Arm Calibration” enables the plugin, developed for calibrating the system.

## 6.5 Real time control

In this section it is described how the wall building application controls and supervises the execution of the tasks, that must be executed during the wall building process. Although it is relative simply described what the robot must do in order for it to build a wall from bricks, it is indeed much more difficult to ensure, that the robot also does as expected. And just as important it must also do these things at the right time, even though the execution of some process is not predictable. This means that the robot should at least to some degree be operating robustly, so that it will be capable of reacting intelligently to different situations, thereby being autonomous.

The task controller thread presented in section 6.2.2 holds a queue of tasks, and these are then executed sequentially, when it is signaled to begin task execution. It is therefore monitored by the task controller when the device servers finish their execution, which has been requested through the device clients. Also, the task controller is made aware of the success/failure of each task execution. This is important, such that the system will not continue task execution when a previous task, on which the following task is conditional, is not carried out successfully.

The necessary monitoring is realized by the device clients repeatedly receiving state information from the device servers[4]. The clients implement the protocol of the server communication and can then update in real time the application awareness about the state of the devices. Specifically, task execution calls are made blocking, such that the device client do not return until it has received information from the corresponding server that the task was either completed or not. The device clients then inform the task controller about the execution outcome and the task controller then reacts appropriately to this.

### 6.5.1 Evaluating device state information

The device clients for both the platform and the manipulator receive packages from respective servers at constant rates. These packages contain information about the current mode of the devices, this being e.g. either *idle*, *executing* or *error*. When a device client sends a request, the latest received package will most likely say *idle*. Then, after some time, when the device has accepted the request and begun execution, the client will receive the *executing* mode. When the client then, again receives *idle* mode it will know that the task was completed successfully, else the *error* mode is received. This information is then forwarded to the task controller which either continues with the execution of the next task in the queue or else stops completely because of an error.

An important detail to note is that the clients implement state machines such that it is handled, that a delay will be present from a request is sent till an *executing* mode is received. During this delay it may happen that the client will receive one or more packages containing the *idle* mode. In order to account for this, the state machines enters a wait state for a specified period. In the wait state the reception of *idle* mode packets is not acted upon. The wait state is left either by receiving *executing* or *error*, or by a timeout, which indicates that the device finished its execution before sending the *executing* mode.

### 6.5.2 Wireless connection monitoring

As the device clients are supposed to receive state information about the devices which is sent at a constant rate, this knowledge can be used to evaluate the connectivity to the robot from the remote computer running the application and controlling the robot via a wireless connection. However, at present this is only used as an indicator for the human operator presented in the GUI, which is there to show if a proper connection is established.

[4] Currently, only the device servers for the platform and the manipulator send this information.

As the control of the robot at present very much depends on the wireless connection, it would be a good idea to consider making the application itself capable of responding to varying connectivity conditions. This is also an important aspect when considering safety, since it should be ensured that the robot will not continue uncontrolled execution if the connection is lost. Instead the robot should rather be told to stop execution if connectivity reaches critical values.

### 6.5.3 Synchronization between planning and execution

As the wall building process is run in another thread than the task controller, some synchronization is needed between these entities. This is because e.g. the manipulator movements cannot be planned before, the platform is known to be in the right position. Qt facilitates such mechanisms, as the wall building application thread is simply send to sleep when it knows that it must wait for some external condition to be met. It is however, the task controller thread which knows about such condition, and when the condition is met, a special call is made to wake sleeping threads waiting for a certain criterion to be fulfilled. In practice the wall building process thread sleeps while the platform is moving, and on arrival the thread is waken up, and it will then continue by preparing for planning the manipulator task to be executed next.

# Chapter 7

# Algorithms for wall building

This chapter describes the algorithms involved in performing pick and place operations with a mobile manipulator. The algorithms can then be used in cooperation to perform more abstract tasks such as brick laying. First the navigation and planning capabilities of the platform is presented and evaluated, then the planning algorithms for generating collision free paths for the manipulator are presented. The chapter also describes the approach used to do the object recognition in the brick laying application. The tasks of grasping and placing bricks are also described, where the placing task involves the localization of a "wall" frame and determination of the place transformation with respect to this frame. Finally, an overall algorithm for brick laying is presented, using the features of the aforementioned algorithms.

## 7.1 Platform navigation

The task of navigating the Seekur platform, see description in section 2.1, is a dynamic task, as the scene is considered to be semi structured and containing dynamic obstacles such as humans. This means that the platform during its movement should be able to detect obstacles and then re-plan in order to find a path to its desired position that is not obstructed. In this project, the localization and navigation software shipped with the Seekur platform, ARNL[41], is used. The software uses a Sick laser scanner, described in section 2.6 to localize itself and to avoid collisions. In this section the precision of the platform navigation is investigated, in order to evaluate how reliable one can expect the platform to be when moving towards a goal. The quality of the localization software, using the laser scanner, is also evaluated.

To be able to navigate with the platform, a map of the environment in which the platform should move needs to be created. This is done by driving the platform around in the environment, ensuring that as much as possible of the environment has been inside the field of view of the scanner. The scanner software then logs all the obstacles detected as points relative to the position of the platform. When the scan is complete the software, Mapper3[39] from MobileRobots Inc., synthesizes the generated points by merging points that come from the same wall segments into lines. After that, it is possible for the user to add goals, a robot home position and forbidden areas to the map. The generated map is then loaded into a server using ARNL and ARIA. Using the map, the robot is able to localize itself, sometimes with the help of a user provided initial guess. A user or a user application can use the navigation software by specifying a goal position to where the platform should plan and move. The robot will then make a path and move along that path to the goal, updating the path if obstacles are detected during the movement. As the platform movement is not very precise, the platform stops and notifies that it has arrived when it is inside a small region around the goal position. The precision of the platform movement is evaluated in section 7.1.2. The navigation and path planning algorithm itself is proprietary developed by MobileRobots Inc., meaning that it is not possible to investigate or improve the algorithms used. The robot can also be controlled with more simple commands specifying a velocity to move in certain direction, making it possible to use other high level planning and control algorithms. The ability of the robot to move laterally would also be feasible for some applications, and adding extra laser scanners on the sides would make this movement as safe at the forward movement.

### 7.1.1 Collision avoidance

The navigation and control software also uses a parameter file, specifying the mechanics of the robot. In that file the parameters for the accelerations, velocities and for the size of the robot are adjusted to yield a reliable and secure navigation. It is especially important to adjust the parameters specifying the size of the robot, so that the planning algorithm and the collision avoidance algorithm knows in which situations the robot will collide with some obstacle. The model consists of a robot radius, which is the distance from the center of the robot to the outermost point when the robot is rotating around its own vertical axis, and a length and width of the robot, used when the robot moves forwards. The robot does not move backwards or sideways when it is in safe mode, meaning that the collision avoidance is on, as the robot will not be able to detect obstacles that do not appear in the map when moving in these directions.

The laser scanner scans in a plane, so obstacles above or under this plane, as well as holes, cannot be detected by the scanner. This also means that collisions with obstacles with a large extent in another height than that scanned plane are not guaranteed not to occur. In the current version of the system one assumes that obstacles with these properties do not occur in the working area of the robot.

The collision avoidance does not change as the manipulator moves outside the area described in the parameter file. This means that to ensure that the manipulator do not collide with anything under the assumption of the objects being largest in the plane of the scanner, the manipulator is moved to a part of its configuration space where all parts of the manipulator is above the platform.

### 7.1.2 Platform movement precision

The precision of the movement of the platform can be investigated in two ways - how well the robot is able move to a position specified in its coordinates, and how accurate its estimated pose is in correspondence to a ground truth. These two abilities are important to know when deciding where to place the platform in order to pick up or place bricks. It is then important to move the platform to a position where the manipulator can reach the desired pick and place positions with a high probability.

#### Test setup

The precision can be measured by setting up a small experiment on the robot. In the experiment, the robot moves back and forth between two goals, A and B, defined in its map. When arrived at the goal positions, the robot logs its actual position and calculates its transformation relative to a chessboard placed at a fixed transformation with respect to the goal. See illustration in figure 7.1. In front of the robot, when placed goal B, some obstacles not part of the map, are placed. In that way it will be possible to investigate if the uncertainty in the platform localization is larger when the map is less accurate.

The precision of the robot movement can be evaluated by examining the distribution of the logged positions. This result tells how precise the robot will move when repeating a movement to a specific goal pose. In this measurement two processes will cause the deviations between the desired pose and the reached pose of the robot. These are the ability of the robot to move precisely enough to reach a certain position, and the ability of the robot to localize itself according to the map it is using.

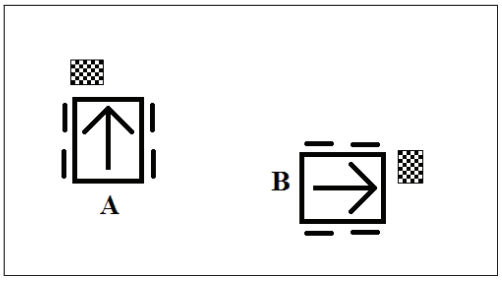


**Figure 7.1:** The platform movement precision test setup, with the platform placed at the two goals.

It is however also interesting to verify the localization against the real world. It can be done by calculating the transformation between the goal and the chessboard located near the goal, $T_{goal}^{chessboard}$. This transformation is constant for all measurements, so deviations will be due to measurement noise. $T_{goal}^{chessboard}$ is calculated by the transformations in equation 7.1.

$$T_{goal}^{chessboard} = T_{goal}^{world} T_{world}^{platform} T_{platform}^{camera} T_{camera}^{chessboard} \tag{7.1}$$

Where *world* is the coordinate system of the map used in the platform. $T_{goal}^{world}$ is the inverse of the known transformation of the goal.[1] $T_{world}^{platform}$ is the logged transformation of the platform, which will contain an error term due to the imprecision in the localization, and $T_{platform}^{camera}$ is the fixed transformation of the camera frame in the platform frame.[2] $T_{camera}^{chessboard}$ is the estimated transformation of the chessboard in the platform frame. This last transformation is calculated from the reconstructed pose of the chessboard in the camera frame, and the error term in this calculation is expected to be negligible compared to the error term in $T_{world}^{platform}$, see section 7.4.

For each observed $T_{world}^{platform}$ and $T_{camera}^{chessboard}$, it is then possible to calculate an estimate of the transformation $T_{goal}^{chessboard}$. Investigating the distribution of $T_{goal}^{chessboard}$, one can validate the correspondence between the map used by the platform and the real world, as this transformation should be constant for all measurements.

The performance of the robot is dependent on many parameters that will affect the result of this test. The surface on which the robot drives can be slippery such that the wheels will spin when accelerating, making the readings from the wheel encoders incorrect. If the surface has high friction or is steep, the robot may have problems reaching a certain position. It may also be the case that the friction is low or the robot is moving downhill such that it may have trouble braking hard enough to stop at a desired goal position. A thorough investigation of the performance would need tests on many different surfaces. In the current setup the surface on which the robot moves is a flat concrete floor, and the test will therefore be carried out on that surface.

The localization task of the robot matches the distances to obstacles observed by the laser scanner with the map of the environment provided by the user. It is therefore clear that the localization precision will depend on the quality of the map. The ability to localize the robot will also be varying depending on where in the map the robot is, as there must be some good landmarks to match the observed laser readings with. The map used in the test is the map used for the brick laying application. It is a map of a small room with straight walls and no small obstacles, the environment can be seen in figure 4.3.

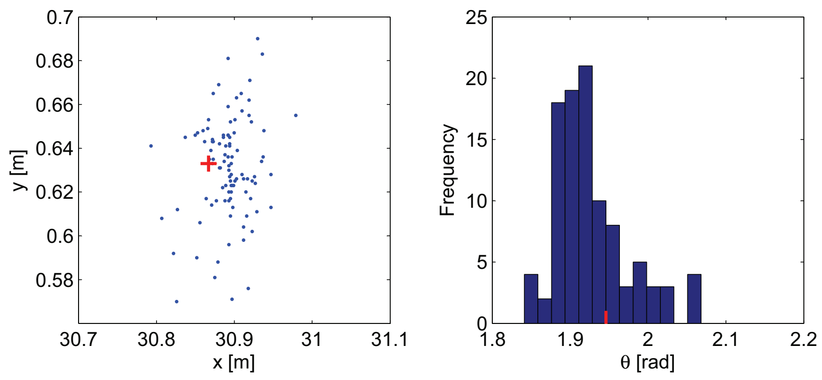
[1]The transformation of the goal is assumed to be known, making it possible to calculate its inverse without risking that small errors in the rotation creates a big error in the position part of the inverted transformation matrix.

[2]Noting that the pan tilt unit on which the camera is mounted is held fixed.

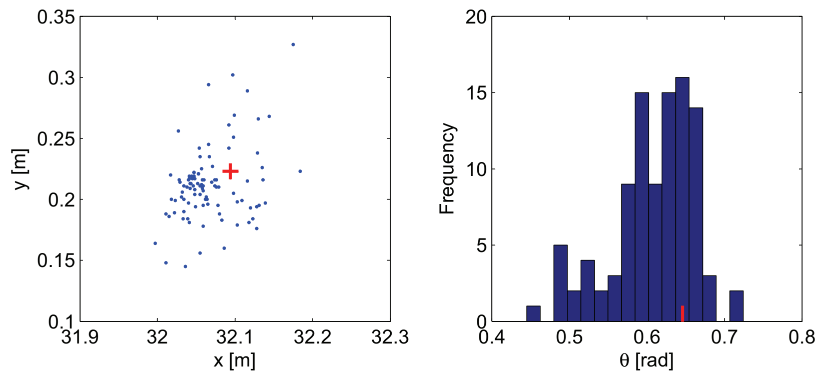
### Results

A test was carried out with the robot moving back and forth 100 times. Below the following three issues are regarded separately evaluating the performance of the platform. The precision of the platform when arriving at a goal as estimated by platform itself, the accuracy of the platform localization, and finally, the actual positioning precision.

**Precision based on localization** Figure 7.2 shows the pose of the platform, $T_{world}^{platform}$, when arriving at the two goals, as observed and logged by the localization software. It must be noted that these poses contain errors from two sources, both the error of not reaching the goal and the error from noisy position observations done by localization software. This means that knowing that the platform will not be able to stop precisely in the desired pose of the goals, it is expected that the logged position will deviate from the desired due to errors in the ability to reach the position and the ability to localize itself.



**(a)** Precision at goal A

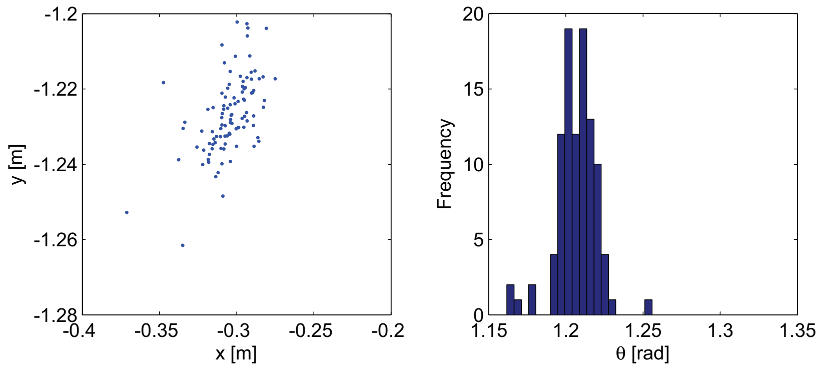


**(b)** Precision at goal B.

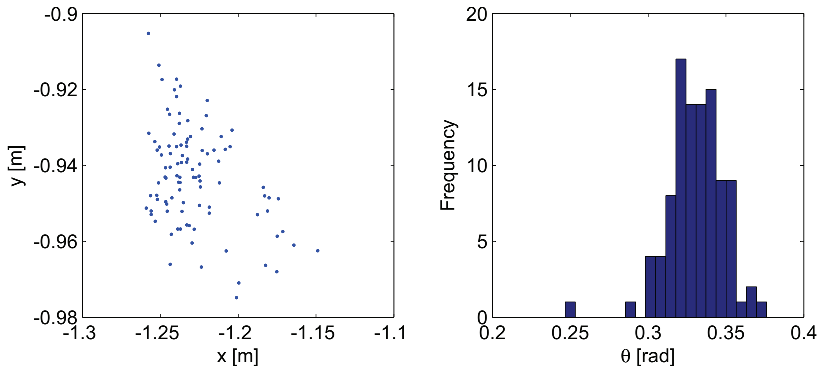
**Figure 7.2:** The poses, logged by localization software, of the platform when arriving at goal A and B, with the goal position and rotation indicated in red.

It can be seen that the observed arrival positions are not distributed symmetrically around the desired pose, meaning that the most likely achieved pose is not the same as the desired. With such a relative large number of samples it is assumed that the non-normal distribution of data is not caused by accident. An explanation for this is that the process producing the data, i.e. the imprecise movement and the localization of the platform, may be non-normal. In the experiment the platform moved back and forth between two goals, and in most situations, the robot then chooses almost identical paths to the goals. These paths usually consist of a path to the goal position and then a rotation around the vertical axis of the robot in order to reach the desired orientation. When approaching the goal position from the same direction in most cases, the resulting expected achieved position will then be slightly off the desired position if the robot systematically stops too early or too late.

**Localization accuracy** Figure 7.3 shows the accuracy of the robot localization, where the parameters of $T_{goal}^{chessboard}$ are estimated as in equation 7.1 and in reality this transformation is constant. As the chessboard and goal are stationary, deviances must be caused by the estimation of the other transformation matrices, as explained, mainly $T_{world}^{platform}$. The spread of the data is smaller than in figure 7.2 because the error of the platform not reaching its desired position is included in the model as $T_{world}^{platform}$.



**(a)** Localization accuracy at goal A.



**(b)** Localization accuracy at goal B.

**Figure 7.3:** The accuracy of the platform localization for goal A and B with a chessboard on the floor as ground truth.

From these data it is possible to get an estimate on the accuracy of the localization software of the platform. Applying a transformation matrix on a list of points will not alter the distances between the points, and in that way the statistical dispersion will not change. Because of that it is possible to calculate the covariance matrix for the independent parameters of $T_{world}^{platform}$ unaffected that the data is viewed in the frame of the chessboard. A 2D pose can be seen as a pair of 2D points with fixed distance where the direction towards the second point corresponds to theta.

The results of table 7.1 where subscript $L$ denotes localization, show that assuming normally distributed data, the platform will in 99 % of all cases be localized within $\pm 0.0633$ m and $\pm 0.0459$ rad. It is interesting to see that the variance is larger for goal B than for goal A. This may be due to the obstacles, placed in the map near that goal. From this result, it can be seen that the localization of the robot by laser is not accurate enough to make the robot place for instance bricks in walls, as the possible errors of 6 cm would be far too large.

| Goal | $\hat{\sigma}_{L,pos}$ [m] | $\hat{\sigma}_{L,\theta}$ [rad] | $\Phi^{-1}(0.995)\,\hat{\sigma}_{L,pos}$ [m] | $\Phi^{-1}(0.995)\,\hat{\sigma}_{L,\theta}$ [rad] |
|---|---|---|---|---|
| A | 0.0162 | 0.0130 | 0.0419 | 0.0335 |
| B | 0.0246 | 0.0178 | 0.0633 | 0.0459 |

**Table 7.1:** Platform localization uncertainty estimated for the two goals, estimated from the data seen in figure 7.3. $\hat{\sigma}_{L,pos}$ is the standard deviation of the position in the direction with largest variation in the x-y plane, calculated as $\sqrt{\lambda_{max}}$ where $\lambda_{max}$ is the largest eigenvalue of the position part of the covariance matrix of the data. The last two columns indicate the 99 % prediction interval extent.

**Actual precision measured using vision** Figure 7.4 shows the accuracy of the complete navigation system of the robot when arriving at a fixed goal position in a fixed transformation to a frame specified by a chessboard. The transformation shown is therefore $T^{platform}_{chessboard}$ given by equation 7.2.

$$T^{platform}_{chessboard} = \left(T^{chessboard}_{camera}\right)^{-1}\left(T^{camera}_{platform}\right)^{-1} \tag{7.2}$$

A ground truth value of the transformation of the platform with respect to the chessboard at the goals is not possible to obtain, so it is not possible to tell if the displacement of the achieved platform pose with respect to the desired platform pose also displays some tendency here.

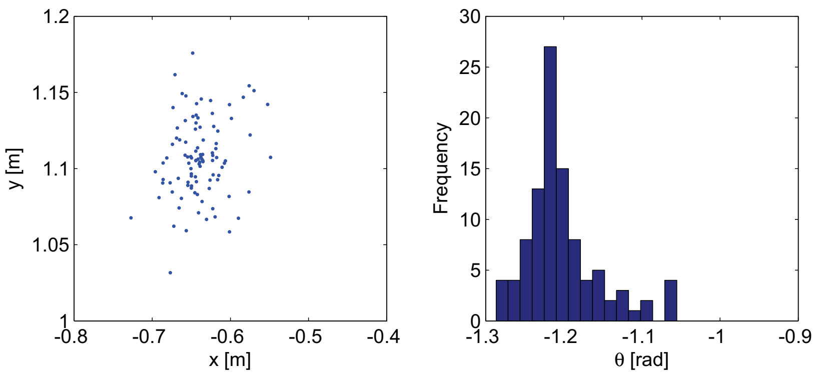
Assuming normally distributed data, the prediction intervals in table 7.1, where subscript $M$ denotes movement, show that the robot in 99 % of all cases will be positioned within a circle of radius 15 cm of the desired position specified by a marker in the real world. This result is important to take into account when choosing or calculating desired platform positions for manipulation tasks of the robotic arm, as one must take this uncertainty into account to avoid positioning the platform such that the manipulator cannot reach its goals.

| Goal | $\hat{\sigma}_{M,pos}$ [m] | $\hat{\sigma}_{M,\theta}$ [rad] | $\Phi^{-1}(0.995)\,\hat{\sigma}_{M,pos}$ [m] | $\Phi^{-1}(0.995)\,\hat{\sigma}_{M,\theta}$ [rad] |
|---|---|---|---|---|
| A | 0.0318 | 0.0483 | 0.0820 | 0.1246 |
| B | 0.0549 | 0.0533 | 0.1416 | 0.1375 |

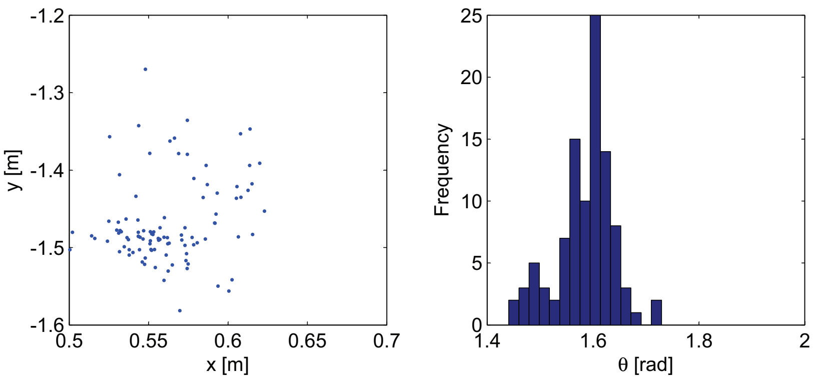
**Table 7.2:** Platform navigation uncertainty estimated for the two goals, estimated from the data seen in figure 7.4. The variables are defined as in table 7.1, here with subscript $M$ instead of $L$.

### 7.1.3 Evaluation and future improvements

This section shows that there are many possible improvements for the navigation of the platform. The introduction of some way of performing 3D obstacle detection would be very good for letting the robot act safely in less controlled environments. The data for this detection could be provided from for example a laser scanner mounted on a tilt unit, a time-of-flight camera or a disparity map calculated using stereo vision. For an simple implementation, these data could just be projected onto the plane of the map of the robot, such that the actual collision checking and for the platform is still kept in a two dimensional space.



**(a)** Navigation accuracy at goal A.



**(b)** Navigation accuracy at goal B.

**Figure 7.4:** The accuracy of the platform navigation for goal A and B with a chessboard on the floor as ground truth.

The Seekur platform is designed to drive around outdoor in terrain, and when using it in small spaces indoor, the motion planning and control algorithms have a hard time navigating the platform in a reasonable way over small distances. Because of that, the ability of the platform to reach a certain goal is not very good, as seen in figure 7.2. The results in figure 7.3 and table 7.1 about the localization software, using a laser scanner, show that this algorithm cannot be used as a basis for manipulating objects without other sensor input. The algorithm is however precise enough to avoid collisions, as no collisions between the platform and other objects have been encountered while using the algorithm. Figure 7.4 and table 7.2 show that the platform movement is very imprecise, such that a safety margin of 15 cm should be used in order to make sure that goals for the manipulator are inside its reach for a given platform pose.

Although many improvements to this part of the system can be addressed, the fact that it is easy to use and sufficient for many applications have made further development of this part of the system less crucial. A wise choice when positioning the platform and the long reach of the manipulator, combined with other sensors to localize objects have made the described functionality sufficient.

The use of a static map in the navigation algorithm also has some issues. The robot uses walls for determining its position, and at a construction site, walls for instance will be built, changing the map. The results showed that this caused a decrease in the localization performance of the manipulator, and when more walls are placed or other objects are moved, at some point then localization will fail completely. The most reliable way of ensuring that such situations cannot occur is to have an extra localization system installed, this could be a GPS system or the placement of markers, visible to the laser scanner or cameras. Alternatively the system must be able to dynamically update the map using SLAM[3] techniques. A global system will also reduce the risk of wrong localizations, which can occur if the map has many similar looking areas, such as straight walls and corners.

## 7.2 Path planning for the manipulator

The manipulator, used in this project is as described in section 2.2 a 6 DOF Universal Robots UR-6-85-5-A robotic arm. The manipulator is used in a dynamic way, meaning that new a priori unknown desired configurations and paths are to be created online, making it necessary to have algorithms available to find these, and not just a list of predefined goal configurations and paths between them.

The problem of controlling the manipulator can be divided into three subparts, the first part being finding a joint configuration that places the TCP in the desired transformation with respect to the base of the manipulator, and the second, finding a path through the collision free part of the configuration space of the manipulator from the current joint configuration to the desired. The last part is the actual execution of motions for the manipulator. This part is carried out by controlling the velocities of each joint in order to reach the waypoints and goal configurations. In order to speed up the execution of paths, minor deviations from the waypoint configurations can be permitted, making it possible for the robot to blend linear path segments instead of having to slow down to almost zero velocity when trying to change direction instantaneously when reaching each waypoint in a path.

In this project the two first tasks are solved with the help of algorithms implemented in RobWork. Therefore we will not describe the algorithms in detail, but instead we will focus on how the algorithms are used and what their advantages and disadvantages are. The third part is implemented by the blend algorithm, developed in the FORK project.

### 7.2.1 Inverse kinematics

Inverse kinematics is the art of finding a joint configuration of the manipulator bringing the TCP into a certain transformation with respect to the base frame of the manipulator. For manipulators with 6 degrees of freedom and a spherical wrist, there will always exist a solution to the inverse kinematics of the manipulator[48]. The manipulator used in this project does not have a spherical wrist, but an analytic inverse kinematics solution is nevertheless available.

When using this algorithm, it is possible to obtain the eight possible inverse kinematics solutions to a TCP transformation. The theoretical model of the manipulator used in the inverse kinematics algorithm, assumes that some of the Denavit-Hartenberg parameters, describing the kinematics of the manipulator are exactly zero or exactly $90°$. This is however not the case when using a calibrated arm, with these parameters calibrated. Because of that it is necessary to adjust the joint values to get the TCP transformation right with the calibrated arm.

This means that we have a good initial guess of a solution joint configuration, and it is then possible to use the inverse Jacobian of the TCP pose with respect to the joint configuration, to iteratively find the joint configuration resulting in the desired TCP transformation. In RobWork, this is implemented in the class *ResolvedRateSolver*. When a solution is calculated it is important to verify if it is inside the defined bounds of the configuration space of the manipulator.

#### Paths generated in Cartesian space

The inverse Jacobian solution method can also be used to produce paths directly in Cartesian space. If the path in Cartesian space is interpolated with a high resolution, it is possible to find

[3]Simultaneous localization and mapping.

the corresponding joint configuration with this solver using the joint configuration solved for the previous point on the path as starting guess. It is then preferable to investigate the distance between two consecutive joint configurations, to see if the solver has found a solution that is far from the preceding, either by accident or because the configuration is close to a singularity. The paths generated this way can for example be used when placing bricks as it would be preferable to place the bricks from above to avoid pushing other bricks sideways when placing a brick in a wall. For unconstrained path planning, it is however easier to find a collision free and short path between two joint configurations directly in the joint space.

### 7.2.2 Path planning

There exist several approaches to joint space path planning for finding collision free paths for a manipulator. When the joint configuration space is high dimensional, as in this situation when planning for a manipulator with six degrees of freedom, sampling based planning is often used. The overall strategy when searching for a path between two configurations, $q_a$ and $q_b$ is to sample the configuration space and then try to connect the sampled configurations until a path can be found from $q_a$ to $q_b$. A path is then described by a list of configurations with linear paths in between.

In this project a RRT planner, specifically RRT-Connect[35], is used for making paths for the manipulator. It is a single-query greedy planner, which means that the planner starts planning with empty trees of configurations. The planner is greedy in the way that it always seeks to connect the two trees as these are grown. The paths produced by a random planner like RRT can often be optimized as they tend to zig-zag through the configuration space. The fact that the planner is single-query makes it less affected by a scene model that is dynamic, as there is no need for bookkeeping of paths that have been checked for collision. In a completely static scene, a multiple-query might have advantages as it would be possible to reuse paths or path segments in different queries.

A general way to reduce the length of a path is to try to shortcut[22] configurations in the path produced by the planner. In this project this is done by sub sampling the path such that the distance between two consecutive configurations is less than 0.2 rad. Then, two random configurations are chosen and if the linear path in configuration space between them is collision free, the intermediate configurations are removed and the new path segment is sub sampled. This procedure goes on until the user is satisfied or a timer runs out, in this case the optimization goes on for 0.5 s. Following this optimization, the path is optimized by path pruning[22]. This algorithm has the benefit, apart from optimizing the length of the path, that it removes the sub sampled configurations on the path. This is practical when executing the paths on the robot with the blend algorithm, described in section7.3. The blend algorithm is designed such that the manipulator slows down when approaching a configuration in the path, not only in configurations where the direction of the path is changing such that some joints may have to stop and move in the opposite direction, but also in configurations with little or no change in direction. Because of that it is preferable to have few configurations in the paths in order to get a fast execution.

#### Collision checking

Collision free paths for the manipulator is ensured by checking all configurations on a path with a sufficiently small resolution if they cause some parts of the system to be in collision. This process involves the actual choice of collision checking algorithm, which is crucial for the speed of the planning algorithm as every path segment proposed by the planner will need to be checked for collision in order to be used in actual paths. The correctness of the collision checking depends on the model of the area in which collision may be encountered, and therefore it is very important to have a correct model of this area and know the state of it accurately.

Collision detection is done by the Yaobi[62] OBB[4] tree based collision checking algorithm. The model for collision checking is the scene that can be visualized in RobWorkStudio. It is therefore important to ensure by inspecting the scene and the files it is composed of, that everything is modeled correctly. The used collision checking algorithm is not capable of calculating clearance, and therefore it cannot be used to assure a certain clearance between objects. However, other collision checking strategies can do this, but anyhow these calculations are very cumbersome, thus increasing the time per collision check significantly. This is not considered beneficial, since collision checking is used extensively during path planning. With no minimum clearance required to exist between objects, small errors in the model of objects will make collisions, that cannot be detected, possible in the real world. A simple way to avoid this, is to make the models of objects a little larger than they actually are. A situation where this problem exists is in the model of the manipulator, since it is known from the calibration of it, section 5.2, that the model parameters are not completely correct, and because of that, the triangle soup models, representing it are also a bit incorrect or misplaced.

Objects are in a collision model often represented by a triangle soup. So reducing the number of triangles reduces the collision checking time significantly. This can be done by using more coarse models. It must however be noted that the configuration space of the robot is decreased when object models are enlarged. This is especially important when enlarging the model of the robot to avoid self collisions. A generic way to simplify collision models of objects and devices is to approximate their structure with convex hulls, this strategy is called convex decomposition [53]. The collision checking can also be speeded up significantly by increasing the distance between collision checks for path segments. Doing this, one must have in mind that the object models must preferably also be enlarged accordingly.

The approach of maintaining an always complete model of the environment for the manipulator has weaknesses, as the model will not be able to detect or act on unpredicted changes in the working environment. Therefore, the only way to make the system more resistant to these errors, is to add some automatic updating and verification of the scene by sensing the scene. This can be done in the same way and with the same sensors as proposed for the manipulator navigation, section 7.1.1, by detecting the objects that are in the vicinity of the robot. An important problem with the use of a scanner to detect obstacles inside the reachable area of the manipulator, is that the algorithm has to know which of the found 3D points do actually come from the manipulator, and which are parts of other objects for which collisions must be avoided. It would also be a difficult task to place sensors for this purpose, able to check the entire reachable space of the manipulator.

## 7.3 Algorithm for making blended trajectories

This section presents an algorithm which is implemented on the controller for the UR-6-85-5-A manipulator. The main efforts to do that were spend in the FORK project. The algorithm is used to determine joint trajectories for path execution. The manipulator controller did not contain functionality to execute paths in any other way than simply making a sequence of $q$ to $q$ movements. Therefore, in order for the manipulator to be faster and to look nicer during movements along a path, this algorithm was incorporated to the controller API[5]. Although the main development and implementation were done prior to the master thesis project, the algorithm is of main importance and therefore the principles behind it together with a brief analysis and evaluation is presented here. For a more thorough exposition consult either the deriver Henrik Gordon Petersen[6] or [46].

[4]Oriented bounding box.

[5]Application programming interface.

[6]`http://www.mip.sdu.dk/people/Staff/hgp.html`

### 7.3.1 Motivation

A path consisting of connected linear segments entails discontinuities where these segments meet. Therefore, if the execution of such a path must comply 100 % with the path, it is necessary for the robot to be stopped at the intermediate configurations, since otherwise, it would require infinite acceleration to change the direction of the robot movement instantaneously. This means that if the robot execution is not allowed to deviate from the path, the total path execution will simply be successive linear movements in which the robot accelerates to a maximum velocity and then, towards the segment end, decelerates completely. However, the appearance of the complete path execution will be jerky and in order to avoid this and thereby reduce wear on the mechanics and shorten the execution time, this algorithm will allow limited deviations from the path.

### 7.3.2 Algorithm principles

The main principle behind the algorithm is to determine a trajectory, in which the robot is allowed to blend between successive linear segments of a path. This means that within a given interval around every configuration of the path, each joint is allowed to deviate from the linear segment in order to make the transition to the next linear segment in a smooth manner. For a 2D configuration space, a blend example is seen in figure 7.5.

Between $q_I$ and $q_F$, the robot is allowed to deviate from the linear segments and may therefore start the transition to the next linear segment at some point before the intersection. The points at which the blend starts and end are given by the blend parameter, denoted $\beta$ and set by the user, which specifies the maximum deviation in radians from a configuration any joint may have. An important aspect of $\beta$ is that the value of it can be changed during a path, such that deviations from specific configurations can be controlled independently.

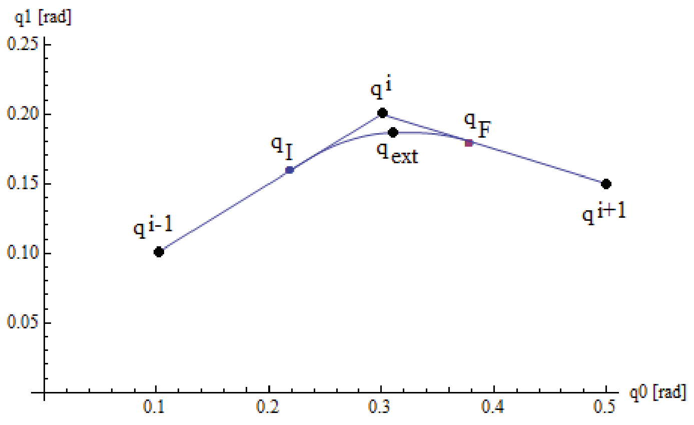


**Figure 7.5:** Blend example in 2D configuration.

One objective of the algorithm is to determine the configurations $q_I$, $q_{ext}$ and $q_F$ for all blends during the path and also, the algorithm explicitly specifies the velocities at the blend initiation point and finish point denoted $\dot{q}_I$ and $\dot{q}_F$. Furthermore, the algorithm also gives the trajectories for the individual joints during the blend. These are set independently for all joints, since joint movements are allowed not to be synchronized during the blend. Only is it required, and obeyed, that joint movements are synchronized in the linear segments between the blends.

The determination of the trajectories in the path is based on the synchronization criteria and on the determination of the minimum time each joint can possibly traverse each specific sub-part of the path. The idea is that given the maximal allowable deviation from a configuration, $\beta$, the algorithm finds the fastest path through the blend and linear segments, ensuring that the robot theoretically will end exactly on the line segment towards the next configuration after the blend.

### 7.3.3 Alternative approaches

Other approaches than the algorithm presented here exist for specifying how a robot must execute a path consisting of several configurations, and this brief summary is inspired by [12] and [59]. One simple option is to consider all segments between two configurations separately. Then the trajectory for these segments can be determined independently, by e.g. a cubic or higher order polynomial. However, it will be required for these segments that they force the robot to be stopped at both end points[7], which might not be desired if a fast execution of the path is sought.

Then, it is possible to consider the complete path, that is all segments, when determining the trajectory for each segment in the path. In this case, of course, the same approach as above is applicable if the robot is allowed to be at rest at each intermediate configuration of the path. However, we wish to travel the path fast, so this is a poor choice. As the complete path consisting of $N$ configurations giving rise to $N$ constraints for each joint, one may then determine a polynomial trajectory of order $N-1$ which meets these constraints. However, several disadvantages applies here including risk of oscillatory behavior and poor numerical accuracy for high $N$, and the resulting system might be heavy to solve.

A way of overcoming these disadvantages is again to consider a cubic polynomial for each segment and then change the velocity constraint at each intermediate configuration of the path to some predefined value. This constraint must then be complied with by the trajectories for the segment before the configuration and the segment after. Alternatively, when solving the system of equations in order to determine the coefficients of these trajectories, it may be required that their velocity and acceleration being equal at the mid configuration. This will then require that one solves the complete system, hence determining all $4(N-1)$ coefficients of the cubic polynomials, at once. Fortunately, written in matrix form this system can be solved in $O(n)$ time as the matrix will be tridiagonal, see [50, chap. 2]. As each segment trajectory then depends on all configurations, they are required to be recomputed if the path is changed at runtime.

The approaches listed so far will entail the acceleration at all times being non-zero (except at zero-crossings), causing a high energy consumption and a long execution time, as many decelerations causing low velocities or even velocities in directions away from the goal. This is an undesired property in most applications. Therefore, linear segments are introduced in which a joint moves at constant velocity, thus at zero acceleration. Parabolic blends can then be added at the ends of a segment to accelerate/decelerate the joint. Following this approach, the linear segment is constrained to be parallel to the line $\bar{q}_j(t_j) - \bar{q}_i(t_i)$, where $q_i$ and $q_j$ are successive configurations that must be reached at times $t_i$ and $t_j$, respectively. This means that the blends will cause the robot not to reach the end-points exactly, which is the same as the algorithm presented here will allow. Traditionally, the blend trajectories are then determined by the amount of acceleration to use in them. As a consequence, if maximum acceleration is used, the robot will follow the straight line until close to the end configuration. However, the algorithm used in this project, then takes advantage of the fact, that it is not be necessary to get closer the end configuration then $\beta$ specifies. Contrary, it will use the specified allowed deviation from intermediate configurations to minimize the complete path execution time.

### 7.3.4 Implementation for RobWork

It became a subpart of this project, together with fellow Master Student Anders Glent Buch, to implement the presented algorithm for creating blended trajectories in the RobWork library. This makes it possible to simulate precisely these trajectories for visualization or for collision checking. When the calculations of the trajectories through the path have been performed, one may then request for the trajectory function value, of time $t$ which must be in the range from zero to the complete path execution time. In this range, both the joint values and their velocities and

[7]The robot must stop at the segment end when exclusively considering only one q-to-q movement.

accelerations can be accessed for arbitrary $t$. In this way, it will therefore also be possible to control other robots than the UR-6-85-5-A by this algorithm.

Many benefits of having this algorithm available for doing planning, path analysis, etc. are present. First, before executing a path on the real robot, it can be checked for collisions. This is exactly one hindrance of utilizing the algorithm at its full potential, since when blends are made, the robot leaves the linear segments between configurations, and these linear segments is all that is usually assured by path planning tools to be collision safe. Therefore, the risk of collision will be present as the linear segments are left.

Utilizing the RobWork implementation and calling this to construct the blended path, the resulting trajectories for the complete path can be passed to a collision detector, which will then also take into account the blended regions. Doing so, will significantly reduce the risk of encountering collisions, since these will then be detected in advance of execution.

**Choosing a good $\beta$**

Another issue related to the risk for collision, is the choice of the blend parameter to use. This is a choice which should be made for all configurations around which the path must blend. This is to allow for varying deviation from the path configurations at different points along the path. Depending on what is aimed for, the blend parameter can be chosen large to increase the probability of being able to traverse the complete path at higher speeds, thus reducing the execution time, or the robot can pass closely by configurations at a reasonable speed. One way of determining exactly how large a $\beta$ should be, is to calculate for each configuration in the path, the clearance to obstacles. This clearance can then be converted to joint offsets which correspond to the distance between the manipulator links when placed at these configuration and obstacles.

Another more simple approach is to test a blend for collisions for several values of $\beta$. It should be possible to increase any value of $\beta$ until it entail collision. If this is done for all blends, a good approximation to the path trajectory for which minimum execution time applies to, will be obtained. It will though require a good deal of collision checking, but these can of course be limited to only check the blend parts of the path which are the ones that will imply collision risk as the blends are increased.

### 7.3.5 Analysis of simulated trajectories

In this section a short analysis is made to investigate properties of the algorithm and explore the outcome of it. It has been conducted by utilizing the simulation facilities which the RobWork implementation of the algorithm provides.

An arbitrarily chosen path is in radians given by

$$\mathbf{Q} = \begin{bmatrix} \bar{q}_1 & \bar{q}_2 & \bar{q}_3 & \bar{q}_4 & \bar{q}_5 \end{bmatrix}^T \tag{7.3}$$

$$= \begin{bmatrix} 4.6512 & -0.0059 & 1.4745 & -0.0541 & 0.1358 & -3.0191 \\ 4.3309 & -0.1309 & -0.3774 & -2.4379 & 2.4854 & -0.4336 \\ 5.0364 & -0.1778 & 2.3398 & -2.5089 & -0.2001 & 0.4858 \\ 0.1947 & -1.8896 & 2.2040 & 1.0455 & -1.4941 & -2.2975 \\ 1.4030 & -1.8107 & -1.4089 & -0.1964 & 0.1894 & -1.7931 \end{bmatrix} \tag{7.4}$$

It is then analyzed what influence different values of the blend parameter will have on the execution of this path by a robot similar to the UR-6-85-5-A. In this analysis the blend parameter is kept constant, hence is the same used for all three blends. The resulting path in Cartesian space by following the stated joint space path is seen in figure 7.6.

It is seen that for increased blend parameter the actual path of the of the robot will be moved farther and farther from the waypoints that the path **Q** give rise to. As is natural, it is also observable that for small $\beta$ the Cartesian path followed by the robot is far from linear although it is in joint

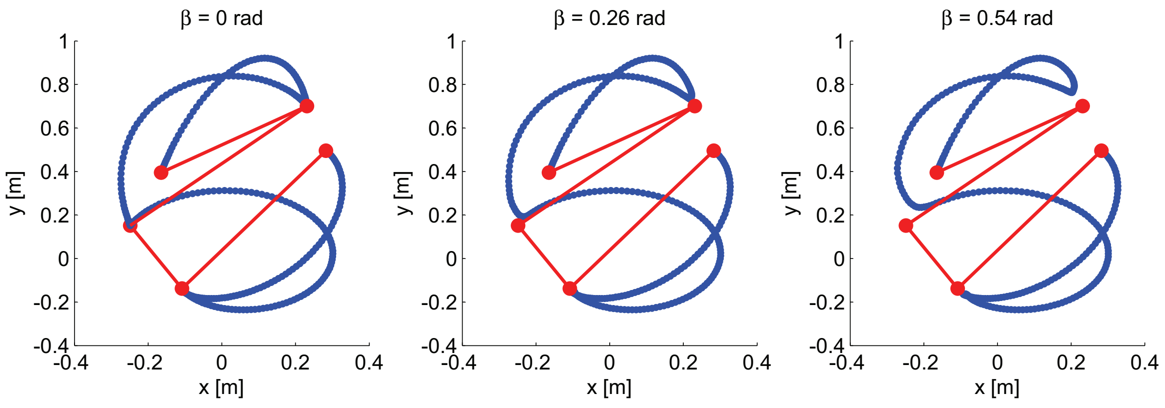


**Figure 7.6:** A view of the path of the manipulator TCP in Cartesian space which is projected down to the $x$-$y$ plane. The same path is shown for varying values of the blend parameter, $\beta$. The red lines indicate the linear path in this space between configurations. The blue lines are the trajectories for the manipulator TCP.

space. It must therefore be remembered that for all values of $\beta$, the red lines are not supposed to be followed only do they represent the waypoints and edges between them. It can be seen that the deviations from the points of the original path is different at the three blends. The point with smallest $y$ coordinate is hardly blended about, even at $\beta$ equal to 0.54 rad corresponding to 30.9°, which is a fairly high deviation in joint space. This illustrates that it is much configuration dependent, how large a deviation in Cartesian space, a certain blend parameter value will entail.

To visualize the actual output of the algorithm, which is used to control the robot movements, the joint position profiles for the path given by equation 7.3 are seen in figure 7.7 and the corresponding velocity profiles are seen in 7.8.

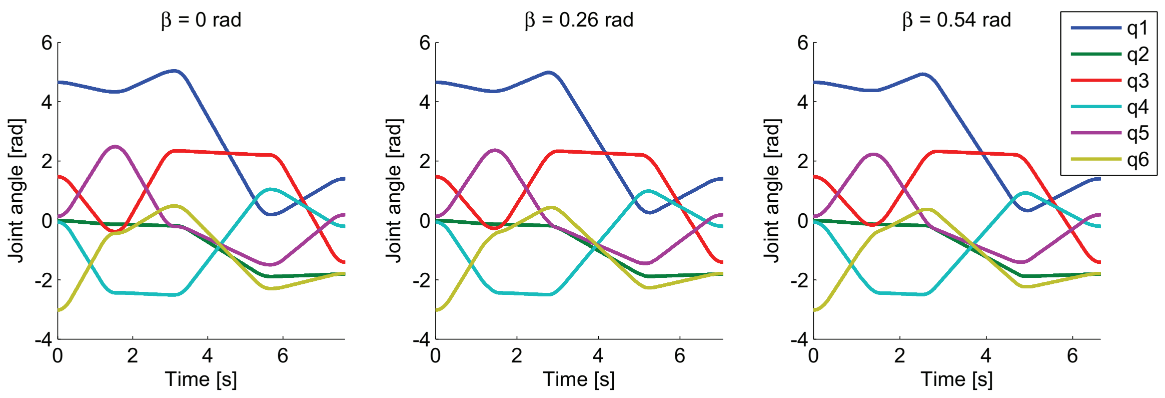


**Figure 7.7:** The joint position profiles for the path in equation 7.3.

What can primarily be seen, is that for zero $\beta$ all joints reach their respective configuration values and are stopped here, which is of course because no deviation from them are allowed and this scenario then ends up being equal to successive $q$ to $q$ movements. For the two other values of $\beta$, the blends may be hard to see in the position plot, but for the velocities it is noticed that these not need reach zero in these cases. The difference between the two non-zero $\beta$ values is that blends are extended in time, meaning that the non-synchronized segments are widened and thus begins earlier and ends later around each configuration.

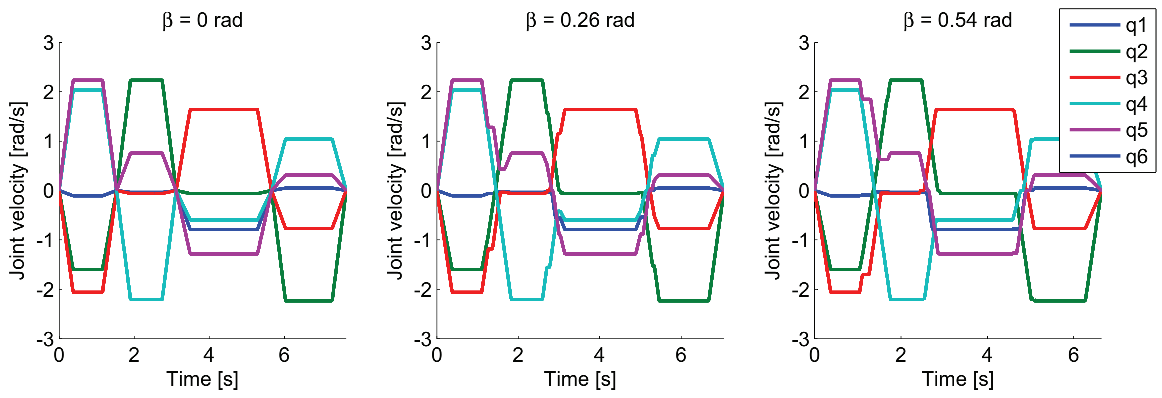
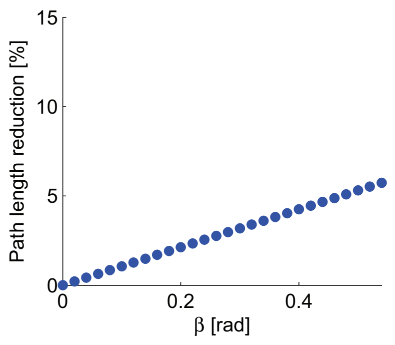
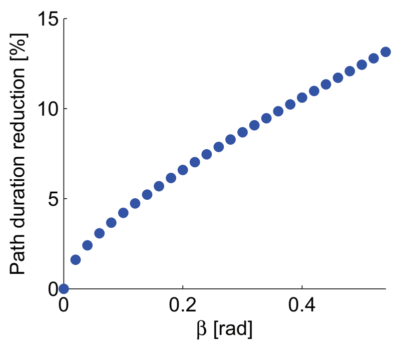


**Figure 7.8:** The joint velocity profiles for the path in equation 7.3.

In order to more thoroughly investigate what influence different values of $\beta$ has, the execution time for the complete trajectories together with the resulting path length[8] has been recorded for the same path for $\beta$ in the same range as above. The results are seen in figure 7.9.



**(a)** Path length.



**(b)** Path execution time.

**Figure 7.9:** Influence on resulting execution path by changing the blend parameter.

Clear linear relationships are observed for both parameters, but with steeper slope for the execution time, which does however not show linearity for small $\beta$. The steeper slope may be described by comparing a few of the previous observations. First, the appearance of the joint angle profiles do not change much with $\beta$ and it is these profiles which gives the resulting path length. Second, the velocity profiles changed a little more as more flexibility where given to attain speed. The result is that the executed path will more or less be the same, but it will be traversed faster as synchronization requirements are gradually let loose.

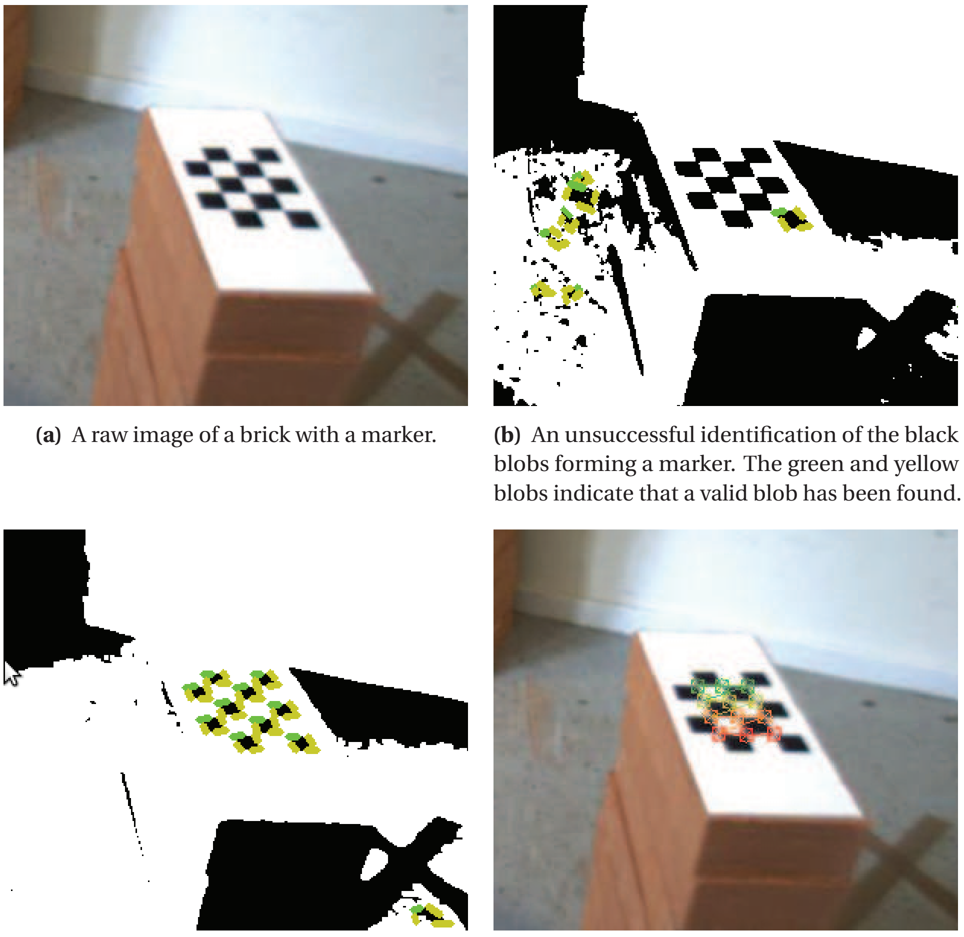
It should be remembered that this example is not considered a complete investigation of the algorithm, but it solely serves as a demonstration of how the algorithm may affect the path execution. As in this case the speed is increased up to 14 %, in some situations, that is expected to be of high value when e.g. performing many repititions of some task.

## 7.4 Object recognition

Using robots in environments that are not completely known requires input from sensors, and especially the task of grasping objects requires the sensors to be able to tell with a high precision where the object to be grasped is and how it is oriented.

[8]Path length here refers to the sum of 2-norms for all $\Delta q$ in the path.

In this project the objects to be recognized and grasped are bricks. These are known objects with a known size and shape. Some types of bricks are hollow with different shapes of hollows in the brick. It is possible to make algorithms that are able to detect and reconstruct objects that are known, but it is very likely that it would be very difficult and costly in terms of development time, to create algorithms that are reliable and precise enough. Instead, an approach that uses markers is chosen. A chessboard marker is placed on top of the brick, and using the known color pattern and size of the chessboard it can be located in a stereo image pair and its 3D pose reconstructed from its identifiable features, such as corner points. A brick with a marker can be seen in figure 7.10(a) and next it will be described how it is located and what is done with the information obtained from this.


**(a)** A raw image of a brick with a marker.

**(b)** An unsuccessful identification of the black blobs forming a marker. The green and yellow blobs indicate that a valid blob has been found.

**(c)** An successful identification of the black blobs forming a marker. The green and yellow blobs indicate that a valid blob has been found.

**(d)** The localized corner features of a chessboard marker.

**Figure 7.10:** Marker detection and localization using OpenCV.

It is the aim, that the markers are used to make the robot capable of determining the 3D grasp position of bricks, and the approach to do 3D reconstruction using the known chessboard markers is:

- **Locate chessboard** - In each image, locate the chessboard with the correct number of fields.
- **Find chessboard features** - For the located chessboards, get the precise pixel coordinates of some chosen features of the chessboard.

- **Calculate 3D transformation of chessboard frame** - Use three corresponding pairs of features to calculate a guess of the 3D transformation of the chessboard.
- **Find optimal 3D transformation** - Use all corresponding pairs of features to find the transformation of the chessboard that minimizes the reprojection error[9].

The algorithms involved will be investigated further in the following.

### 7.4.1 Object recognition in 2D

The object recognition algorithm to locate chessboard markers in an image is implemented in the function cv::findChessboardCorners() of OpenCV. Examples of the algorithm in use can be found in [9]. It is important to know the inner workings of the algorithm to be able to optimize its performance, and this can be done as the implementation of it is open source. Setting OpenCV in a debug mode, it is possible to investigate how the algorithm detects chessboards. Example views from this can be seen in figure 7.10(b) and 7.10(c).

**Algorithm 1** Find Chessboard OpenCV

```
1:  image := toBinary(inputImage, thresholdFlag);
2:  while Chessboard not found and attempt < maxAttemts do
3:      image := erode(image);
4:      Detect black square-like blobs.
5:      Connect blobs lying close to each other to components.
6:      for all component in components do
7:          if component is chessboard then
8:              return component
9:          end if
10:     end for
11: end while
12: return no chessboard found
```

A sketch of the chessboard detection algorithm can be seen in algorithm 1, and with that in mind, it is possible to use the algorithm wisely such that the rate of success is improved. From the use of the algorithm, it was discovered that the rate of false positives was very low, therefore the task of tuning the algorithm can be limited to increase the number of detections. The algorithm takes several flags, indicating how the algorithm should work, as input. Some flags indicate how the transform of the image to binary should be performed in line 1 of the algorithm. This does for instance set if a global or a local threshold is used in that transform, obviously the algorithm will only be successful if the chessboard fields are segmented correctly, which can be a problem in bad lightning conditions. Flags can also be set to adjust how to determine if the found components can be originating from the chessboard searched for, line 7. An often occurring error of the algorithm is that the black squares of the chessboard are overlapping, as seen in figure 7.10(b). To deal with this, the algorithm uses its erode step, line 3, to erode the image, hopefully resulting in the black areas to be split into square like blobs. A view of successfully found blobs constituting a chessboard component can be seen in figure 7.10(c) and the final result for the chessboard corner positions, found by also applying the technique of the next section, and shown on the original image is seen in figure 7.10(d).

Due to the varying lightning conditions and perspectives of the chessboards, it is difficult to find the flags that yield the highest success rate. It seems like that in difficult situations with the

[9]The word reprojection does in this context mean the act of projecting points reconstructed in 3D back into the image plane.

chessboard plane normal pointing in directions far from the direction towards the camera, it is random which flags that will make the erode step able to segment the blobs correctly into the chessboard pattern. Because of that the algorithm is improved by running the entire algorithm inside a loop with different flags set each time. Although these evaluations are time consuming, it is far faster than acquiring new images or trying to move the platform to another place from where the chessboard can be detected. It is however still important that the lightning conditions are good, as the algorithm is sensible to for instance direct sunlight either at the marker or into the camera.

#### Feature detection an localization

When the chessboard pattern has been found, a local search for the precise location of the internal corners in a chessboard can be executed. This is done by the function cv::cornerSubPix() in OpenCV, which, as the name indicates, is able to locate the image plane coordinate of the corners. An internal corner of a chessboard in a grayscale image can be identified as a point where, in a small neighborhood around it, the gradient is either close to zero (for points in a completely white or black area) or orthogonal to the vector from the corner center to that point (for points on the edges between the black and white areas). By visual inspection, the algorithm seems to work well when choosing the size of the neighborhood as the default value, and corners are found with a very high accuracy.

### 7.4.2 3D reconstruction

In this section we want to describe a way to determine the 3D reconstruction of a known object from a set of object points located in an image pair from a calibrated stereo camera. The simplest approach to 3D reconstruction, is the reconstruction of a point. Assuming that the point is identified perfectly in a pair of images as a point in each image plane, the lines from the origin of the cameras through these points in the image planes will intersect in the 3D point. If however the identified image point are perturbated with some error, the exact 3D point cannot be found.

Assuming that the observed points are distributed with a Gaussian distribution around the exact point, the maximum-likelihood estimate of the actual 3D point position can be found by minimizing a geometric cost function[25], minimizing the Euclidean distance in the image plane between the observed points in the image plane, $x_j$ for $j = 1, 2$, and the reprojections of the estimated 3D point $\hat{X}$ in the image planes, $\hat{x}_j$. The cost function can be stated as follows

$$\chi^2\left(\hat{X}\right) = \sum_j \sum_k \left(x_{j,k} - P_j \hat{X}\right)^T \left(x_{j,k} - P_j \hat{X}\right) \tag{7.5}$$

Where $j$ indices the two cameras, $k$ indices the x- and y-axes in the image planes, $P_j$ is the $3 \times 4$ projection matrix for the $j$'th camera and $\hat{X}$ is the homogeneous 3D coordinate of the point being reconstructed.

A starting guess for this minimization is obtained using Plücker coordinates to represent the lines, $L$, from the focal point of the cameras through the observed image plane points.

$$L \equiv \left(\nu; \mu\right) \tag{7.6}$$

where $\nu$ is the line orientation vector and $\mu$ is the line moment where

$$\mu = m \times \nu \quad \forall m \in L \tag{7.7}$$

It is then possible to calculate the closest point on one of these lines, $L_1 = (v_1; \mu_1)$, to the other line, $L_2 = (v_2; \mu_2)$, called $M_1$, using the formula

$$M_1 = \frac{v_1 \cdot (v_2 \times \mu_2) - (v_1 \cdot v_2)\, v_1 \cdot (v_2 \times \mu_1)}{\|v_1 \times v_2\|^2} v_1 + v_1 \times \mu_1 \tag{7.8}$$

Similarly the point $M_2$ can be calculated by switching the indices. The midpoint, $(M_1 + M_2)/2$ can then be used as an estimate of the 3D reconstructed point, [52].

For a list of $i$ points, with known transformations with respect to the object frame, $X_i$, constituting an object, a geometric cost function can be written.

$$\chi^2\left(T^{object}\right) = \sum_i \sum_j \sum_k \left(x_{i,j,k} - P_j T^{object} X_i\right)^T \left(x_{i,j,k} - P_j T^{object} X_i\right) \tag{7.9}$$

Here the optimal $T^{object}$, which describes the transformation of the object frame, minimizes the geometric error. A transformation matrix has only six independent variables, so the rotation part of the matrix is estimated by its equivalent angle axis representation, recommended in [25].

Solving the non-linear optimization problem of equation 7.9 can be done by the Levenberg-Marquardt algorithm, also used for calibration. The algorithm is standard for problems like the present one, [25], and also other multidimensional data fitting problems.[10] The algorithms combines the guarantee of convergence of gradient descent, and the quadratic convergence of the Gauss-Newton algorithm. In this project, the algorithm is used with approximate Jacobians.

As this is an iterative algorithm, a good initial guess will speed up the search significantly and reduce the risk of finding a local minimum.

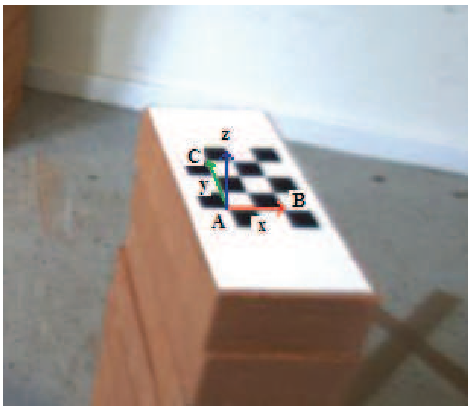


**Figure 7.11:** Visualization of calculation of frame transform for 3D reconstruction of chessboard markers.

A simple way to obtain an initial guess is by reconstructing three points of the object and then calculate a transformation of the object based on that. A point can be reconstructed by minimizing equation 7.5. With the points A, B and C localized as in figure 7.11, the position and orientation can quickly be estimated. The $x$ axis is an unit vector pointing in the direction of $B$ from $A$, the $y$ axis is a vector pointing in the direction of $C$ from $A$, and then $z$ is orthogonal to the two. The three vectors can be orthogonalized by a Gram-Schmidt process[23], to form a valid rotation matrix. The translation of the frame can then be set to the center of the chessboard by translating it along the found axes.

The algorithm of reconstructing object transforms using the Levenberg-Marquardt algorithm, has the benefit that it can also estimate the covariance matrix for the estimated parameters, and in

[10]It is used as default for minimizing sum of squares problems in Mathematica[29].

that way it is possible to investigate the quality of reconstructions. An evaluation of the algorithm can be found in section 8.4.

The algorithm has one weakness, in practice the geometric errors are not Gaussian distributed, most of them are, but vision algorithms also produce rare outliers that are not covered in the chosen distribution. When performing the minimization of equation 7.9, this leads to the possible outlier being given a large weight, dragging the result in a wrong direction. It would be feasible to be able to detect these outliers during the optimization and then exclude them from the further calculations.

### 7.4.3 Markerless reconstruction

It would be preferable that a marker was not necessary in order to detect bricks. Classical algorithms such as edge detection and line detection by Hough transform[13] could be ways to detect bricks, but also more generic algorithms based on learning of features, such as Viola-Jones classifiers[45] could be possible, using the edges of the bricks as well as the pattern of the hollows in hollow bricks as possible inputs. It is however worth noting that the production of bricks is not very precise, so bricks may vary in size, and the pattern of the hollows may be difficult to detect, even with bricks from the same production unit. In general bricks can vary also in color, and many types of bricks do not contain any pattern of hollows as they are massive. Therefore, it is not very likely that a reliable algorithm for precision grasping of bricks can be developed solely using vision. Other sensors might be necessary to obtain an acceptable performance.

### 7.4.4 Precision of a reconstructed pose

It is interesting to determine how precise we can estimate the pose of a marker using the proposed Levenberg-Marquardt algorithm to minimize equation 7.9. When picking up bricks we are interested in the precise position of it in order to know the transformation of it in the gripper when holding it such that it can be placed precisely. In that situation the precision of the rotation is not so crucial, as the flat parallel contact surfaces on the gripper will ensure that the object rotation when grasped is constrained in two dimensions. The third axis of rotation is orthogonal to that surface, and when a brick is placed in the wall, the rotations in that direction will be zero as the brick will end up lying flat on the top of the bricks below it.

When using the reconstruction algorithm to estimate the transformation of the wall frame, the rotation is very crucial, as an error in the rotation around an axis will cause a point to be translated proportionally to the distance to it. In other words an error of 0.01 rad around some axis will result in a difference in the position of a point at a one meter distance of up to $(1\ \mathrm{m} \cdot \sin(0.01)) \cong 1\ \mathrm{cm}$. This situation may occur when a brick is placed according to a frame, estimated by reconstructing a chessboard marker.

The Levenberg-Marquardt is able to calculate the covariance matrix of an estimated pose of a chessboard, and from the experiment of constructing a wall a series of covariance matrices are sampled from the tasks of locating the bricks and the wall frame. It is possible to use these covariance to estimate the precision in the pose reconstruction algorithm. It is worth noting that the pose estimates vary across the measurements, as the transform between the platform and the chessboard is not constant, the covariance is however assumed to be of equal magnitude across the measurements, although for instance the main direction of uncertainty in the position estimate is dependent on the actual estimate.

For a covariance matrix, the direction with most uncertainty is the same direction as of the eigenvector corresponding to the largest eigenvalue. In that direction the standard deviation is equal to $\gamma = \sqrt{\lambda_{max}}$ where $\lambda_{max}$ is the largest eigenvalue. This means that for each covariance matrix estimated, a $\gamma$ can be calculated.

When grasping bricks, the precision of the position of the brick is important. The $\gamma$'s for the position part of the covariance matrix, i.e. the upper left $3 \times 3$ matrix have been calculated for the covariance matrices obtained during the pick operations in the wall building test presented in chapter 8. The results can be seen in figure 7.12. It can be seen that $\gamma$ can be in the size of up to $1.4\,\text{mm} = \gamma_{max}$. A 95 % confidence interval for the actual position estimate will contain $\hat{p} \pm 1.96\gamma$, where $\hat{p}$ is the position, meaning that deviations from the actual value in 95 % of the estimates for $\gamma = \gamma_{max}$ will lie less than 3 mm away from the actual value. A typical value of $\gamma$ is 1 mm, it will then mean that error will be less than 2 mm in 95 % of the cases.

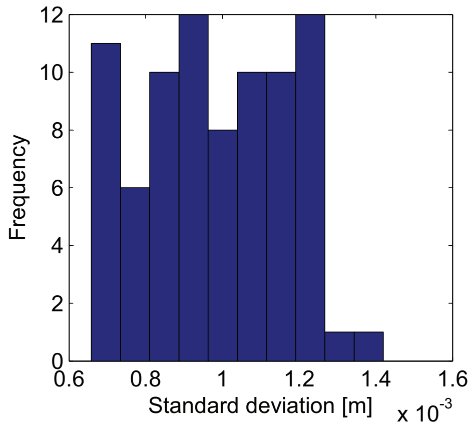


**Figure 7.12:** The standard deviation in the direction of most variability of the position of a reconstructed chessboard. The data come from estimating the pose of bricks for being picked up.

When placing bricks in a position relative to an estimated frame, as explained, the precision of the estimated rotation will be affecting the end position the most. Using the lower right $3 \times 3$ submatrix of the covariance matrix, a $\gamma$ can be calculated for the rotation part of the covariance matrix. The results can be seen in figure 7.13. Here $\gamma$ is between 0.002 and 0.005. Using the confidence interval as before, and knowing that $\sin(\theta) \cong \theta$ for small $\theta$, it can be seen that a point at a distance of one meter to the frame will have an error of less than 10 mm.

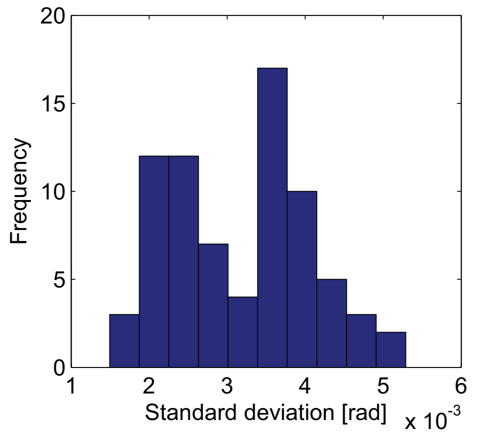


**Figure 7.13:** The standard deviation in the direction of most variability of the rotation of a reconstructed chessboard. The data come from estimating the transform of the marker specifying the wall frame.

It is worth noting that these confidence interval widths of 3 mm and 10 mm are very conservative guesses of the actual precision, due to the worst case guesses of the $\gamma$s. Because of that smaller errors than these can be expected in the real world.

## 7.5 Picking up bricks

Picking up bricks is an important aspect of the development of a brick laying robot, and it will be very difficult to be able to place bricks with a high accuracy if they are not also grasped accurately. Figure 7.14 shows an example grasp.

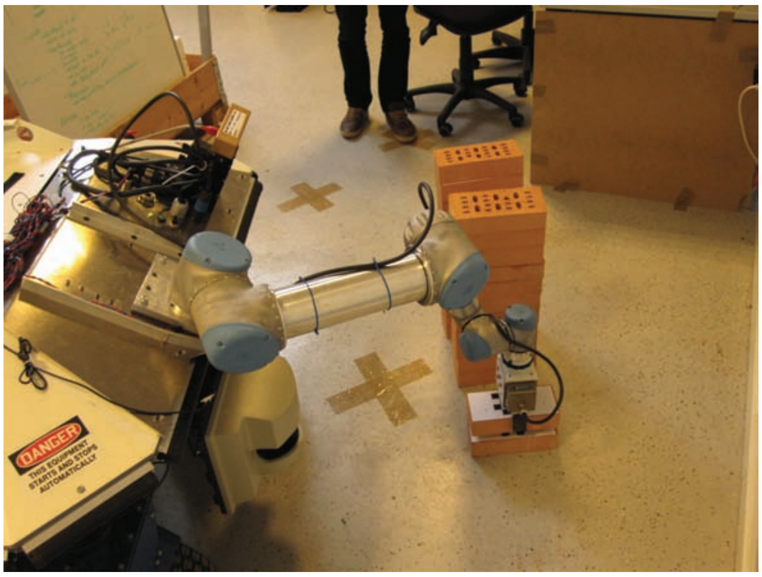


**Figure 7.14:** Example of grasping a brick.

The grasping procedure is explained as follows:

1. **Localize brick** - Reconstruct the 3D transformation of a graspable brick using stereo vision and the algorithm presented in section 7.4.
2. **Find grasp transformation** - Find the desired transformation of the manipulator such that the gripper can grasp the brick. Using a parallel gripper on the parallel surfaces of bricks, makes no need for calculations of the grasp quality; if the gripper closes around the center of the brick as in figure 7.14, the grasp will be acceptable with the used gripper.
3. **Find grasp configuration** - Solve the inverse kinematics of the manipulator, for the grasp configuration.
4. **Make path** - Make a path for the manipulator to a configuration a little above the desired grasp configuration, avoiding collisions with obstacles and other brick piles.
5. **Tessellate manipulator** - Move the manipulator linearly along the negative z axis of the object frame, see figure 7.11, to the grasp position.
6. **Grasp brick** - Grasp the brick by closing the gripper while applying a sufficient pushing force to keep the object fixed during movement.
7. **Move manipulator home** - plan a path to the transportation configuration of the manipulator, starting with a small tessellation along the positive z axis of the brick frame.

The presented grasping procedure is quite simple, as the bricks are easy to grasp using the parallel gripper with some high friction coating on the fingers. The linear tessellations are used partially to make it easier to make paths to the grasping position for the RRT planner, described in section 7.2. It also helps avoiding that the grasped object will slide along the brick on which it is resting when picked up. It is important to assure that movements of the manipulator or gripper are finished before a new movement is initialized, as that would result in imprecise or erroneous grasps.

When using this approach no unsuccessful grasps and no droppings of the grasped bricks has been encountered, as is also reflected in the experiments section, 8.4, therefore this algorithm is satisfactory with the current use of it.

#### Grasping without detecting markers

If it is necessary to grasp bricks without markers or in positions on the top of the platform, see figure 7.15, the precision is likely to be reduced. Therefore it would be preferable to use sensors to update the transformation of the brick with respect to the gripper TCP. It is also likely that when decreasing the grasp quality and increasing the speed of the manipulator, the bricks may slide during motion. Some ways to avoid or correct these problems are:

1. Using a parallel gripper makes the grasp especially sensitive to rotations around the axis connecting the two fingers. The risk of this flaw may be reduced by utilizing a constrained planner for the manipulator trajectories which prevents rotations of the held brick around that particular axis.

2. It might also be worth investigating other kinds of grippers, e.g. suction discs, customized for handling bricks. If it is not an option to purchase a new gripper, the design of the fingers may be reviewed. It is possible that with a wider gripping surface per finger, the grasp may be more stable. If the design of a finger could provide two contact surfaces, this would also reduce risk of unwanted brick rotations.

3. Pose refinement by vision can be used to update the actual position of the brick in the gripper. It can be done by moving the gripper in front of the cameras, enabling a precise pose estimation but also slowing down the process due to the extra movement.

4. Visual servoing may be applied such that an error in the placement position gives feedback to the control system, which will then iteratively adjust the brick position until satisfactory. This could in addition reduce necessary precision of the estimation of the reference frame to which bricks are placed relatively.

5. A force-torque sensor or tactile sensors may be added to determine how the brick is held by the gripper. It may even be possible to calculate the center of gravity of the brick with this sensor in order to refine the estimate of the brick pose. This sensor can also simply detect if a brick is held or not.

So far, however, these improvements have not been necessary, but one or more of them are likely to be needed in future development.

## 7.6 Placing bricks

Placing bricks in a wall, is a task which is carried out as the robot after having picked up one or several bricks, arrives at the wall building location, where it will then put the bricks into the wall being build. Actually, the task of placing a brick, which is held by the gripper, in a desired transformation relative to some frame, is actually very similar to that of grasping objects described above. The first thing to do, is to determine the transformation of the robot relative to the frame, to which the brick must be laid relatively. This transformation cannot be expected to be known in advance, since that would require a very accurate movement of the platform to the goal position, and that is not possible with the current equipment. It is, however, also more appealing not to depend on such predefined transformation, as that does not accommodate a flexible and robust operation of the robot.

The robot determines, by the recognition of a known marker, its current transformation to this marker frame. The marker is required to be placed precisely relative to the desired wall position. Then, by bookkeeping where bricks that has been placed in the wall before the current one, the desired transformation of the current brick relative to the marker is easily calculated, and a manipulator joint configuration is found which places the brick in the desired position. The internal state of the robot must be up to date about where the preceding bricks have been placed in the wall, and therefore it can then search for a collision free path to the place configuration which takes into account the bricks that have been laid until then. Similarly, when the manipulator should be moved away after having placed the brick, the newly placed brick is added to the wall model, which then ensures that collisions with this brick are also avoided henceforward.

## 7.7 The brick laying task

The task of brick laying is presented and described in chapter 3. The more detailed algorithms used to solve the subtasks in the task of laying bricks are described in the previous sections of this chapter. Here, the algorithm for laying bricks, using these algorithms is presented in algorithm 2.

**Algorithm 2** Build wall

```
1:  $BP$ := GenerateBrickPile(stacks, bricksPerStack);
2:  $BT$ := GenerateBrickPattern(width, height, bond);
3:  while !empty($BP$) and !empty($BT$) do
4:      move platform to $BP$
5:      identify next graspable brick $b_i$ from $BP$
6:      estimate $T^{b_i}_{platform}$ by stereo vision
7:      pick brick $b_i$ at $T^{b_i}_{platform}$ with manipulator
8:      move manipulator to home
9:      move platform to wall
10:     estimate $T^{wall}_{platform}$ by stereo vision
11:     get $T^{b_i}_{wall}$ from $BT$
12:     move manipulator to $T^{b_i}_{wall}$ and place $b_i$
13:     move manipulator to home
14: end while
```

First step of the algorithm is to initialize the model of the brick pile. The model is initialized and updated in order to be able to make paths for the manipulator that do not collide with the pile, when picking up bricks. Before the robot starts execution, a list, $BT$, of desired transformations of each brick in the wall with respect to a reference frame is calculated. The input to this function is the desired width, height and bond of the wall.

The while-loop starting in line 3 of the algorithm contains the actual building of the wall. The first step here is to move the platform to the brick pile. The location of the brick pile is a priori known in the map used by the platform. The platform navigation algorithm is described in 7.1. The arrival position of the platform is not very precise, so before a brick can be grasped, the precise transformation of the brick with respect to the robot is estimated. This is done by identifying and reconstructing the pose of it, using stereo vision, with the algorithm described in section 7.4. It is then possible to move the manipulator to the brick and pick it up using the grasping strategy described in section 7.5, together with the path planning and execution strategies for the manipulator, described in section 7.2. It is in that situation important to know the locations of the other bricks in the brick pile, such that collisions between the manipulator and these are avoided.

Before moving the platform to the wall, the manipulator is moved to its home position. The path generated for the manipulator will then take into account that a brick is held by the gripper. The manipulator movement to the home configuration is done to reduce the risk of encountering collisions between the manipulator and obstacles in the scene while moving the platform. The position of the wall is, as the position of the brick pile, known to the robot, so the robot can move to that goal as soon as the manipulator is in place.

Upon arrival at the goal position in front of the wall, the robot updates the transformation between the platform and the wall frame, as described in line 10, with the algorithm of section 7.4. The robot can then calculate the desired transformation of the brick, $b_i$, using $T^{wall}_{platform}$ and $T^{b_i}_{wall}$ in the platform frame. It is then possible to make a path for the manipulator to place the brick in the desired transformation, using the strategy in section 7.5. The robot can then move the manipulator back to its home configuration in order to be ready to go and fetch the next brick.

### 7.7.1 Transportation of bricks

The brick laying algorithm presented in algorithm 2 has not been developed with the focus on execution speed, which is why the robot has to move back and forth between the brick pile and the wall once for each brick being laid. This method is good to test and examine the key aspects of brick laying, the ability to pick up the bricks and especially the ability to place them correctly and precisely in the wall. It is however slow and might cause the system to fail more often than necessary, as the planning for the platform sometimes fails, see section 8.4.2.

The robot does not have any means to pick up more than one brick at a time, but it can be possible for the robot to carry several bricks, by one at a time picking up a brick and then place it somewhere on the robot. When the desired number of bricks has been loaded, the robot can then carry these from the brick pile to the wall. There is a flat area on the top of the platform which can be used for such a purpose. An example of the robot placing bricks on its top surface can be seen in figure 7.15.

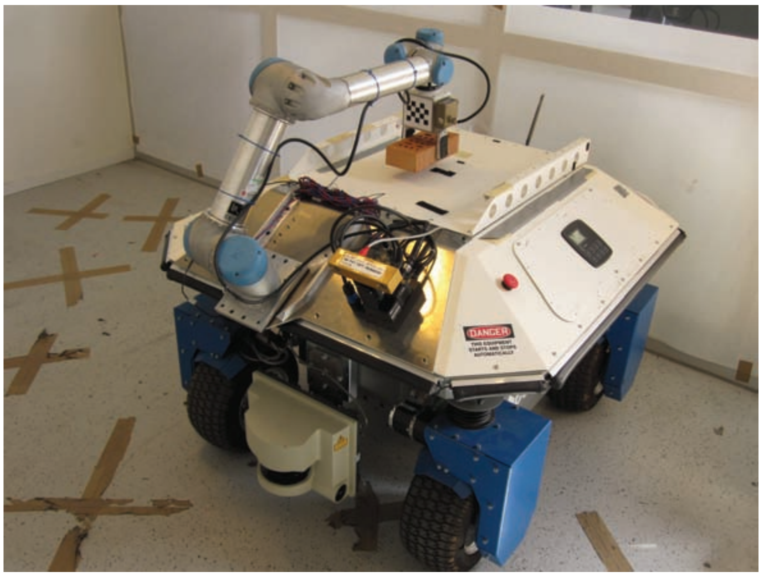


**Figure 7.15:** Transportation of bricks on the platform.

A problem arising when choosing this approach is not to lose the accuracy in the brick laying process. It is likely that this accuracy might fall as two grasps and placements of bricks are needed for each brick in this approach, first when placing the bricks on the robot carrying area, and second when placing the bricks in the wall. During the movement of the platform from the brick pile to the wall, it is likely that the bricks will slide or move a bit due to the vibrations of the platform. Because of that, the second grasp of the bricks will be inaccurate or maybe even unsuccessful, leading to a bad placement of the brick in the wall. It is in the current setup not possible for the robot to see this area with the cameras, making it difficult for the system to adapt to these errors as the position of the bricks cannot be re-estimated.

The simple solution to this is to make some kind of fixture for bricks on top of the robot, such that the bricks can be assured to be in a known position when grasped for placing in the wall. The fixture could be designed such that it actually improves the accuracy of the robot, as they can be used to let the brick slide from its inaccurate placing position to the accurate desired position in the fixture. The position of these fixtures will be static, so it may be feasible to make pre-calculated paths to and from these positions. It is likely that the area above the platform will be collision free, making these paths possible. It is also possible to use the suggestions of section 7.5 to improve precision of grasps.

# Chapter 8

# Experiments

This chapter will describe the experiments conducted in this project, which have mainly been the construction of a 1 m × 1 m wall using the robot and the developed control system. This is to evaluate the performance of the robot performing masonry, which is the algorithms presented in the previous chapter. First, good measures for the performance of the robot performing masonry are presented, then the experimental setup can be presented along with the measuring techniques. After that, the results concerning the quality of the constructed wall and the overall performance of the system are presented and evaluated.

## 8.1 Brick laying performance measures

As described in the problem statement, the performance of the robot performing masonry is to be evaluated in order to be compared to the performance tolerances of conventionally constructed walls. To do this, it is necessary to identify some measures that can be obtained consistently in the experiments. It is also important to identify measures presenting the performance of the execution in a way that makes it possible to evaluate the subtasks of the robotized masonry task.

### 8.1.1 Quality of a constructed wall

When a wall is constructed it is interesting to investigate the quality of it in terms of the precision in which the bricks are placed. Although the wall constructed in the experiment is not as big as a normal wall, it will still be possible to investigate it in the same ways as conventionally constructed walls. The Danish standards and measurement techniques are described in [63, 11]. Some interesting measures and tolerances applicable for this experiment are:

- **Planarity** - The planarity of a facade wall. This can be measured by placing a two meter straightedge at the wall and verify that all bricks must lie within ±6 mm/2m.
- **Head joint** - Bricks are made from natural materials and due to the drying process, their size may vary. This is compensated for when putting the right amount of mortar in the wall between the bricks. The reference value for the width of the mortar space between bricks is 12 mm. The tolerances are such that an acceptable mortar space must vary between $12-4$ mm and $12+8$ mm.
- **Deviation from vertical** - For a normal storey, with height of less than 3.4 m, the deviance from vertical from top to bottom must be less than 20 mm. This deviance can be measured both at the side of the wall and at the end of the wall.

Obtaining good data for these measures, it will be possible to evaluate the quality of the wall with respect to the tolerances. Along with these measures, it would be interesting to estimate the variance of the position of the placed bricks in the directions along the wall and across the wall. These values are interesting when investigating the covariances for the precision of the many subparts of the system, such as calibration and manipulator movement. These values can also be used as a reference for other projects involving mobile manipulation.

### 8.1.2 Performance of execution

The performance of the robot is measured in order to visualize the time consumption for the robot building a wall. These measures will be interesting if the system is to be compared to other similar systems[1] or human construction. As this is a development system, it will also be interesting to divide the performance measurements into measures for execution of the different subtasks of the system. In that way it is easier to detect which parts of the execution that can be optimized to gain most in the overall performance. The main interesting measure is the speed of execution. The process occasionally breaks down due to some errors, and to be able to minimize these occurrences and identify where and how often they happen, the failure rate of the subtasks is also measured. A failure in the execution of some task is in this context a situation where a human must interact with the robot in order to resume the execution.

The subtasks for which time consumption and failure rate is measured are the following:

- **Move platform to wall site** - This task is initialized when the application is ready to send a command to the platform to move to the wall site and ended when the platform software signals that it has arrived or has failed.
- **Move platform to brick pile** - This task is initialized when the application is ready to send a command to the platform to move to the brick pile and ended when the platform software signals that it has arrived or has failed.
- **Locate wall frame** - This task is initialized when the robot has arrived at the wall site and ends when the transformation of the wall frame with respect to the robot is estimated using the stereo vision system.
- **Locate brick frame** - This task is initialized when the robot has arrived at the brick pile and ends when the transformation of the wall frame with respect to the robot is estimated using the stereo vision system.
- **Pick brick with manipulator** - This task covers the picking of a brick starting when the brick is located and ending when the manipulator has moved to a transportation configuration, carrying the brick.
- **Place brick with manipulator** - This task covers the placing of a brick starting when the wall frame is located and thus the desired position of the brick can be determined, ending when the robot has moved back to its transportation configuration after the brick has been placed in the wall.

### 8.1.3 Performance of manipulation

The pick and place tasks depend on solving many smaller subtasks, therefore it is interesting to investigate the speed of execution and failure rate of these subtasks, in order to detect possible improvements. The manipulation tasks are the following:

- **Solve inverse kinematics** - The task of finding the closest collision free robot configuration constrained by a desired $T_{base}^{end}$ to the current configuration.

[1]Although it has not been possible to find such data for such systems so far.

- **Find path** - The task of connecting a start and goal configuration by a collision free path of joint configurations.
- **Find linear path** - The task of creating a path that is linear in Cartesian space between two $T_{base}^{end}$ transformations.
- **Optimize path** - A subtask of find path, optimizing the path length by shortcutting intermediate configurations, still avoiding collisions.
- **Move along path** - This task starts when a path is ready to be send to the manipulator, and is finished when the manipulator signals if the path was executed successfully or not.

When one is to improve the performance of one of these tasks, it must be kept in mind that it could be the performance of another task that causes the possible bad performance. For instance if the movement along a path takes a long time, it may be caused by the planner creating longer paths than necessary. It is important to note that movement of the platform and gripper can in many cases be executed at the same time as planning for a subsequent task is performed, making the actual execution time of the entire process shorter than the sum of these times.

## 8.2 Experimental setup

The experiment for evaluating the wall building mobile manipulator is to build a 1 m × 1 m wall in a semi-controlled environment. The environment can be seen in figure 4.3. In this environment the robot moves back and forth between a brick pile and the wall, picking and placing bricks one at a time according to the algorithm in section 7.7. The environment will be specified with an imperfect map, and it can be expected that humans move around in the environment during the construction. If the robot stops because of a failure during the execution, the task can be continued from there. The algorithm relies on marker detection to locate the precise position of the wall frame.

### 8.2.1 Assumptions

To make the experiment successful, the execution is based on the following assumptions for the environment and task.

- **Partially known map** - A map of the working environment has been created by the laser scanner of the platform. It is not perfect, as it is made from sampled laser readings, and some parts of it are incomplete. The desired platform goal positions according to the brick pile and wall frame are defined in that map such that bricks and desired place positions are reachable for the manipulator.
- **Detectable bricks** - Bricks are equipped with markers in order to be located, only one brick is visible in the brick pile at the time.
- **Wall frame indicated by marker** - A known marker is placed in a known transformation respective to the wall being build, making it possible to locate the frame in which the positions of the bricks in the wall are specified.
- **Partially controlled lightning conditions** - The test is conducted indoor in a room with both artificial lightning and some sunlight coming in through windows. No direct sunlight is permitted.

- **No additional obstacles when moving manipulator** - It is assumed that no object other than the robot, the floor, the wall or the bricks known in the environment model of the robot can cause collision with the manipulator.
- **Obstacles detectable by laser scanner** - The obstacles in the map are detectable by the 2D laser scanner of the platform meaning that these are assumed to be largest in any direction at the level of the scans.

With these assumptions in mind, it is expected that the robot can perform the brick laying task successfully.

## 8.3 Measuring techniques

### 8.3.1 Quality of a constructed wall

In practice the tolerances for a constructed wall are verified by investigating critical places in the wall, and it is only inspected if some part meets the requirements or not. In this situation we are interested in making more general inference on the data of the wall, that is not only obtaining an accepted/rejected statement. For the same measures as above, the measurement methods are:

- **Planarity** - The planarity is measured by placing a spirit level vertically at eight positions along the wall, and then measure the distance from this vertical line to the bricks in the wall. When evaluating the quality, the average, i.e. the average distance from the level to the wall, can be subtracted and a linear regression can be made to remove effect of the possible slope of the wall from this measure.
- **Head joint** - The head joint can directly be measured by inspecting the space between each brick. In that way it is also possible to investigate if the space has equal mean and variance in different parts of the wall.
- **Deviation from vertical** - The deviation from vertical can be verified by evaluating the slope of the linear regression, performed in the evaluation of planarity. The deviation can be investigated in both the direction transversal to the wall and in the direction along the longitudinal axis of the wall.

As mortar is not used, the height of the wall cannot be controlled by the robot, and it is therefore not interesting to measure. The variance for the position of the placed bricks can be estimated in the direction orthogonal to the wall plane from the samples of the planarity of the wall. The size of the head joint can be used to determine the variance for the position of bricks in the direction along the wall. The following model can be set up to describe the stochastic variable $H$ which is the width of a head joint between two bricks

$$H = B2 - B1 - c \tag{8.1}$$

where $B1$ and $B2$ are the stochastic variables describing the position of two bricks along the wall. $c$ is a constant corresponding to the length of a brick plus the desired head joint width. As it can be assumed that $B1$ and $B2$ have equal variance and are independent, the variance of a brick must then be half of the variance of $H$. This is because the variance of the sum (or the difference) of some stochastic variables is equal to the sum of the variances of the stochastic variables when they are uncorrelated.

### 8.3.2 Performance of execution

The easiest way to measure the performance of the robot executing tasks is simply to use software timers in the program. This enables high precision measurements with consistent determination of when one task is finished and another is started.

It is worth noting that the software approach to measuring time and failure rates, is not able to detect all software failures, as if the main application breaks down, it will not be possible to log the time and success or failure of tasks not finished.

## 8.4 Results

Data is acquired from a construction of a 1 m × 1 m wall. The wall was constructed with the mobile manipulator as specified in section 8.2. In order to reduce the risk of damaging equipment the speed of the platform and the manipulator was reduced, which will affect the performance of the robot, especially the manipulator movement. The constructed wall can be seen in figure 8.1. This wall has a reasonable size for evaluating its quality. As 72 bricks are used in the construction, sufficient data to analyze the performance in terms of speed and success rates is also available.

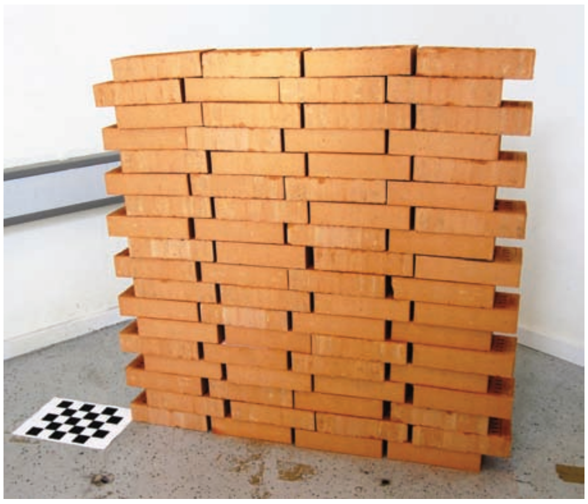


**Figure 8.1:** The constructed 1 m × 1 m wall.

An interesting observation from evaluating the wall visually is that the error in the positioning of each brick is independent. A human brick layer will place each brick relative to the closest bricks, resulting in a small difference in placement of the bricks, but not necessarily a small absolute error, he will however have difficulties avoiding the errors adding up, when placing more bricks. It is a matter of preferences what is best, but with the current setup there is a visual difference between the two ways of constructing walls, regardless of if tolerances are met or not.

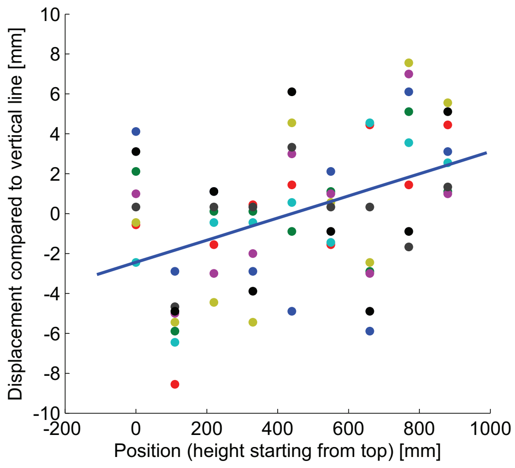
### 8.4.1 Quality of a constructed wall

The quality of the constructed wall and the evaluation of if it will meet the requirements for normal walls specified in 8.1.1 is presented here.

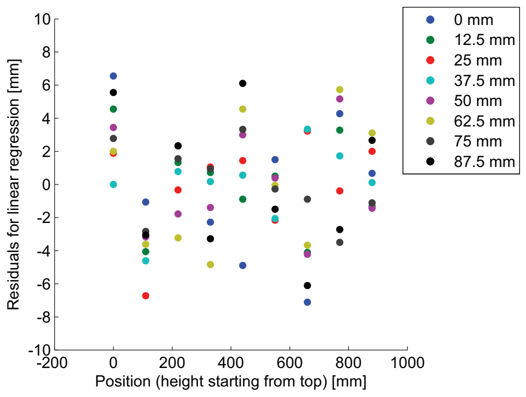
#### Planarity

The planarity can be evaluated by the residuals of the linear regression on the planarity measurements, in figure 8.2(b). The residuals show the deviation in the brick position in the direction normal to the wall plane, corrected for the possible slope of the wall. It can be seen in figure 8.2

that the data which is sampled at eight different positions, indicated by the colors, seem to come from the same distribution, modeled by the proposed linear regression model with Gaussian noise on the residuals. One pattern is however visible - the samples from the same height on the walls are correlated. This means that each layer of bricks is displaced in one direction compared to the wall plane. This fact is hard to explain, but the fact that the robot may swing in that direction when bricks are placed, due to the change in balance point, may be an explanation.



**(a)** The linear regression of the displacement of the bricks in the plane of the wall on the height at the wall.



**(b)** The residuals of the linear regression in figure 8.2(a).

**Figure 8.2:** The evaluation of planarity and deviation from vertical for the constructed wall. The colors indicate the eight different positions along the wall where the measurements are sampled.

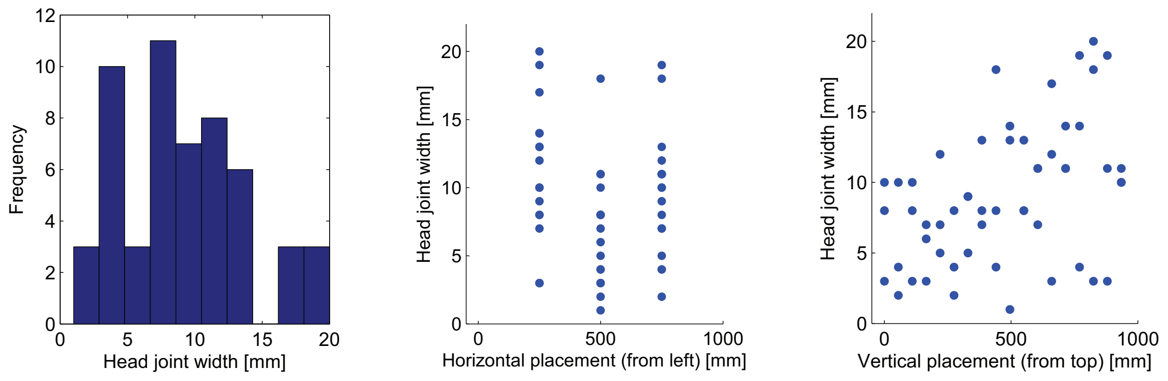
Assuming a normal distribution of the residuals, a prediction interval of these will reveal the maximum and minimum limits for the placement of new bricks. This can be seen in equation 8.2. It can be seen that the 99 % limits of ±8.29 mm is close to the tolerance of ±6 mm, but improvement would still have to be made in order to reach this limit.

$$P.I._{planarity}(0.99) = [-8.29\,\text{mm}; 8.29\,\text{mm}] \tag{8.2}$$

### Head joint

To make a brick wall look good it is important to have a uniform spacing between bricks. This space is evaluated in figure 8.3. It can be seen in figure 8.3(c) that the variation in the head joint space is larger at the bottom of the wall than at the top. The bottom of the wall is further away from the base of the manipulator and therefore errors in the calibration of the manipulator kinematic parameters, both base frame transformation with respect to the camera and the Denavit-Hartenberg parameters of the manipulator, may cause larger errors there. Another cause of this error may be the suspension at the platform wheels that may change when the manipulator is moved and therefore changes the center of gravity.

When not taking this pattern in the variance of the distribution of the head joint into account, the data forms a normal distribution around a center of 9 mm, as seen in figure 8.3(a), and a prediction interval can be calculated to verify if the head joint space meets the requirements. The prediction interval can be seen in equation 8.3 It is much wider than the tolerance, meaning that this requirement is not met. The fact that the average of the head joint is smaller than the expected 12 mm is due to the fact that the bricks are a bit larger than their nominal size, actually about 3 mm. Zero is included in the interval, and because of that some bricks collides with the already



**(a)** The distribution of the head joint between bricks.

**(b)** The sampled head joint values plotted against the position along the wall.

**(c)** The sampled head joint values plotted against the height at the wall.

**Figure 8.3:** Evaluation of the head joint between bricks sampled in different places at the wall.

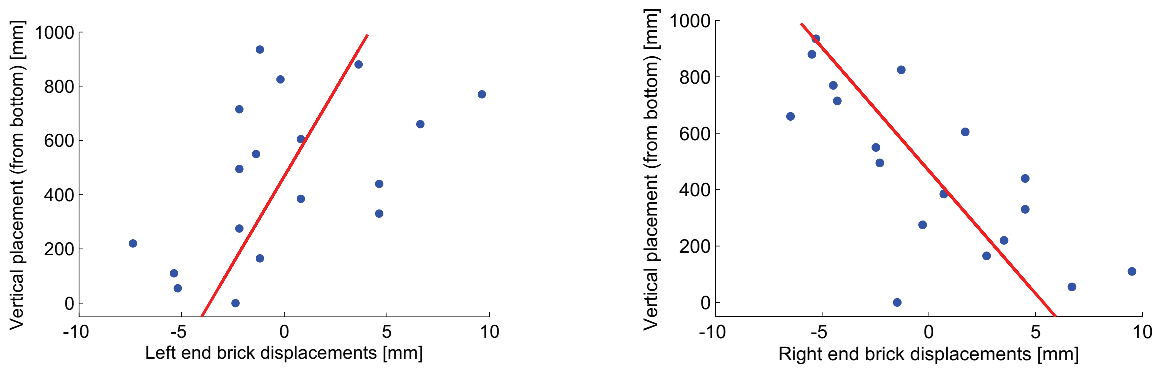
placed bricks, when placed. This was also observed during the actual construction process, as some bricks ended resting with one end on top of the brick it was supposed to be placed next to. These bricks were manually moved to a correct position such that the construction could be carried on.

$$P.I._{Head joint}(0.99) = [-3,52\text{ mm}; 21.63\text{ mm}] \tag{8.3}$$

## Deviation from vertical

The deviation of the wall from the vertical axis is evaluated by investigating the regression in figure 8.2(a). The confidence interval for the slope can be seen in equation 8.4. This value is comparable to the tolerance value of 20 mm/3m = 6.67 mm/m, although the tolerance is not met. The placement of the chessboard pattern which serves as reference frame for the wall is crucial for this value as the estimated z-axis deviation from vertical will directly result in this error.

$$P.I._{transversal\ slope}(0.99) = [2.87\text{ mm/m}; 8.23\text{ mm/m}] \tag{8.4}$$



**(a)** The deviation at the left end of the wall

**(b)** The deviation at the right end of the wall

**Figure 8.4:** The longitudinal deviation from vertical of the wall

The estimation deviation from vertical in the longitudinal axis of the wall is less precise, as less data can be produced. The data can be seen in figure 8.4. A clear pattern can be seen, showing that the wall is wider at the bottom than at the top. This can possibly be explained by the larger uncertainty in the brick placement at the bottom of the wall. The confidence interval for the regression lines can be seen in equation 8.5. It can be seen that these intervals are large and they show that the wall can be much tilted.

$$P.I._{longitudinalslopeleft}(0.99) = [0.1\,\mathrm{mm/m}; 14.6\,\mathrm{mm/m}] \tag{8.5}$$

$$P.I._{longitudinalsloperight}(0.99) = [-17.0\,\mathrm{mm/m}; -5.9\,\mathrm{mm/m}] \tag{8.6}$$

In the measurements for the deviation from vertical, the measurement technique is very imprecise, as it is done by placing a spirit level vertically and then measuring the distance to this at different points in the wall. This highly affects the results as the accuracy of this device is about 10 mm/m. It is especially a problem for the measurements at the ends of the wall as fewer data can be acquired from that.

**Variance of brick position**

The variance of the placed bricks in the direction across the wall can be seen equation 8.7. The value assumes that any linear tendency of the wall from being not vertical is subtracted.

$$\sigma^2_{across} = (3.2163\,\mathrm{mm})^2 = 10.3448\,\mathrm{mm}^2 \tag{8.7}$$

The variance of the brick position along the wall can as described earlier be calculated from the variance of the head joint space. The variance can be seen in equation 8.8.

$$\sigma^2_{along} = (3.4461\,\mathrm{mm})^2 = 11.8758\,\mathrm{mm}^2 \tag{8.8}$$

It can be seen that the variances in the two directions are almost same size, which indicates that the sources of the uncertainty of the brick position can be expected to have the same distribution in these directions. The values are not very large, but still apparently not small enough to let the constructed wall meet the normal tolerances. All in all the results show that the constructed wall has a quality that is comparable to that of conventional masonry, although the tolerances are not entirely met. It is expected that by optimizing accuracy in the different subtasks, the tolerances can be met.

### 8.4.2 Performance of execution

The pie chart in figure 8.5 displays how the total time of the wall construction is distributed among the subtasks. It reveals that more than 50 % of the total execution time is spent on navigating around with the platform. This is not very surprising as the platform has to move each time a brick has to be picked or placed. An interesting observation here is that the time spent navigating to the brick pile is larger than the time spent on moving back. An explanation for that is simply that the manipulator moves along different paths in each direction. It is also seen that the pick and place tasks that are quite similar also takes about the same time. The reason for the more time executed on picking may be that a few more pick operations are executed, as it is necessary to start by picking up a brick when restarting the application after an error. The pick and place tasks also consume a very large fraction of the total time spent, meaning that the reduction in total execution time from improving the locating tasks will be almost undetectable.

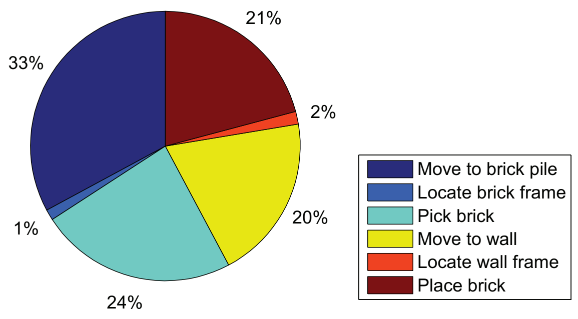


**Figure 8.5:** A pie chart of the time consumption of different subtasks of wall building.

Figure 8.6 shows how much time is spent on each execution of one of the tasks. Again, the locate tasks take very little time. The outliers are expected to come from the situations where the vision system was not able to locate the chessboard in the first pair of images.[2] The large spread in the task of moving the platform to the brick pile indicates that the paths followed by the platform are diverse. It was seen that during execution of these paths, the robot ended in a position too far from the goal, making it necessary for it to turn around and move a little back in order to reach that goal. The figure also shows that the average time to pick a brick and the time to place it are almost the same. The larger spread in the time to pick is difficult to explain, although an explanation may be that it is sometimes hard to find a path for the manipulator when it is needed to move the manipulator around some brick piles to grasp the next brick.

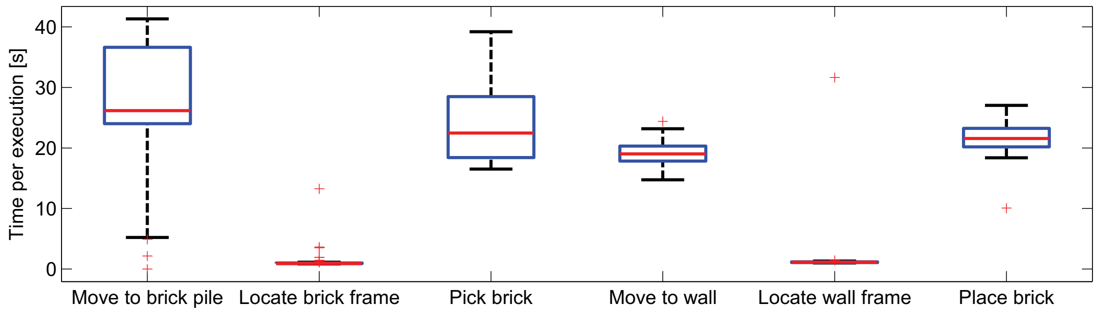


**Figure 8.6:** Box plots of the time it takes to execute subtasks.

A histogram of the total time for a loop, meaning the picking and placing of one brick, can be seen in figure 8.7. It can be seen that the times are distributed between 70 s and 120 s with a multimodal distribution. This interval of times is very high, and it reveals that large improvements are definitely possible, both when considering the navigation of the manipulator and the platform. The platform navigation is carried out by ARNL, as explained in section 7.1, which cannot be changed as it is closed source, therefore a more detailed investigation on that is not carried out. The performance of the manipulator subtasks can however easily be changed and improved, and thus it is interesting to investigate further.

The failure rates for the different tasks can be seen in table 8.1. It can be seen that during the entire wall construction process, only few failures were detected. The failures for the platform navigation when moving to the brick pile are hard to control, but ensuring that obstacles do not appear near the goals is a way to reduce this rate, as the robot in rare occasions moves to a position that is too close to the obstacle, resulting in that the robot controller goes into an "in collision" state, disabling further movement.

[2] If chessboard localization fails, a new image pair is grabbed and the chessboard localization algorithm is again run. This is done three times at maximum.

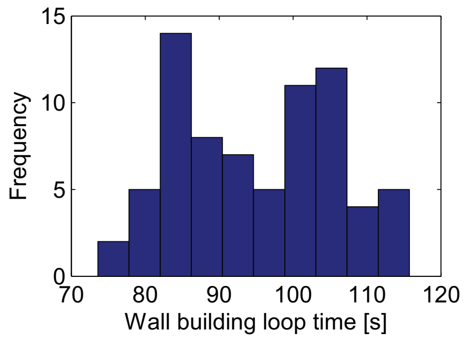


**Figure 8.7:** A histogram of the time it takes for a loop, meaning the time from the robot is ready to pick a brick, until it has picked and placed it and is ready to pick the next.

| | Move to brick pile | Locate brick frame | Pick brick | Move to wall | Locate wall frame | Place brick |
|---|---|---|---|---|---|---|
| Failure rate [%] | 4.6 | 0.0 | 2.7 | 0.0 | 0.0 | 2.7 |

**Table 8.1:** Failure rates for the tasks of building a wall.

A few errors were encountered for the pick and place task, these will be investigated further when investigating the platform movement in detail. The overall result is however, that the reliability of the remaining parts of the system is very high, and quite long runs without human intervention can be possible if just a few more failures are avoided.

### 8.4.3 Performance of manipulator movement

Figure 8.8 reveals that with the current setup, almost no time is spent on creating linear paths and solving the inverse kinematics for the robot, it is not surprising as both tasks primarily consist of solving the analytical inverse kinematics, which is a process that can be done in $O(1)$, whereas the planning and collision checking tasks in high dimensional spaces have a much higher time complexity.

It can be seen that much time is spent on creating paths for the manipulator, the box plot of the task “Find path”, and on executing it, the box plot of the task “Move robot”. The task “Optimize path” is a part of “Find path”, so it can be seen that about half of the path planning time for the manipulator is actually spent on optimizing paths that are collision free from start to goal. It is worth noting that in this test the robot did only move with 30 % of its maximal speed and 20 % of its maximal acceleration. If the maximal values was used, the actual execution time will in fact be significantly faster than the planning time. It will in many cases not be desirable as it would then be faster to reduce the amount of planning and optimization and instead move along less optimal paths.

In some situations where many paths are to be planned, it is possible to plan asynchronously of the moving of paths. In that way, the time available for planning while moving the robot in another path, or executing another task, is “free”, meaning that this planning will not affect the overall performance.

In table 8.2, it can be seen that the failure rates are very low for the manipulator, both in terms of planning and moving. As an analytical solution to the inverse kinematics exists, and since failures in solving i.k. occur, it means that either the transformations for which the algorithm fails are in collision or they are outside the working area of the manipulator. This means that this error is due to other parts of the system not functioning properly. Most likely the failures occur because

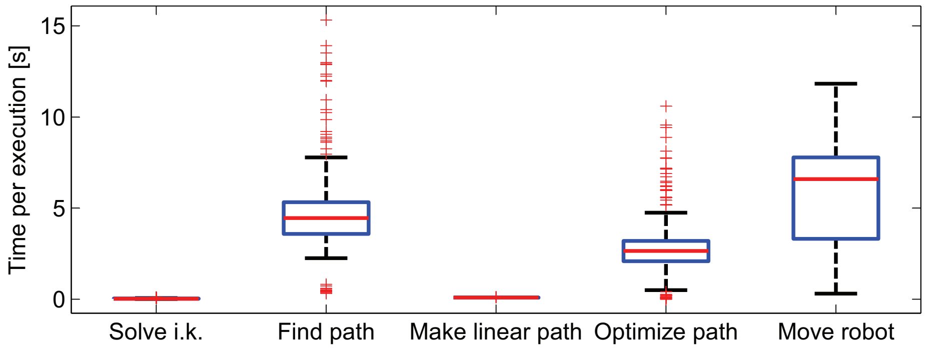


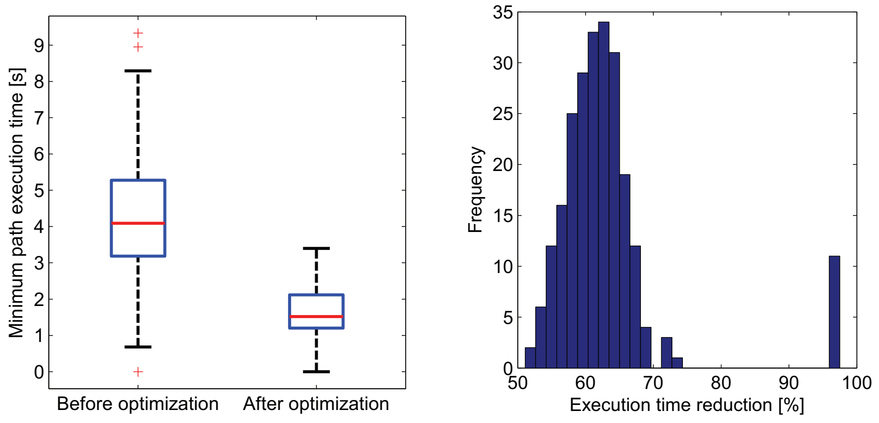
**Figure 8.8:** Box plot of the times for calculating and execution tasks when controlling the manipulator.

| | I.k. | Find path | Make linear path | Optimize path | Move robot |
|---|---|---|---|---|---|
| Failure rate [%] | 0.98 | 0.0 | 0.0 | 0.0 | 0.26 |

**Table 8.2:** Failure rates for the manipulator planning and movement.

the platform was not placed correctly in order to make the goal transformation of the manipulator reachable.

The path optimization task is quite time consuming, so in order to see if it improves overall performance, an analysis of it is conducted, and the results are seen in figure 8.9.



**(a)** Box plots of expected path execution times.

**(b)** Histogram of achieved reductions in expected path execution times.

**Figure 8.9:** The performance of the path optimization algorithms form the manipulator movement. The examined paths are the paths executed by the robot during the wall construction where the task "Find path" was executed in order to generate the path.

The figure shows that for the robot moving at full speed and acceleration, the execution time is more than halved. This substantially advocates for the use of the optimization, as other advantages such that less power consumption and less jerky movement is also achieved. The bar indicating occurrences of time reductions very close to 100 % in the figure are expected to be due to some outlier paths being reduced to almost zero length. It is possible to choose how much time is spent on the shortcut algorithm, this time can also be tuned to yield a large reduction in execution time without having to wait for that algorithm too much.

# Chapter 9

# Epilog

This chapter contains the final evaluations on the complete project and it is split into three sections. The discussion section is an analysis and discussion of the methods used in the project and the results obtained. The analysis will be made with respect to the main objectives of the project and the problem statement. The conclusion section will summarize from the discussion and present the results of the project together with a statement about the quality and usability of these results. The perspectives section will look at the project results in the light of the surrounding environment, thereby analyzing the future possibilities for the product. In connection with that, proposals for further evolution of the system are presented.

## 9.1 Discussion

The compliance of the project results to the objectives of the problem statement is evaluated in the following.

- **User interface** - The developed user interface for the specification of a wall building task, is simple as was desired. However, the variety of walls that can be specified with it is similarly limited. This is because the testing purposes of this project did not require more advanced features than specification of the dimensions, and which number of brick of the wall should be the next to place in it. The user interface as of now, must actually be accompanied by a tool for giving inputs to the platform about where to go and pick up bricks and where to go for laying these. These locations are presently build into the platform map, meaning that these cannot be changed by the user through the user interface.

- **Brick recognition and grasping** - In this project, bricks are located with the help of markers. This provides an easy way of detecting them and finding known features, such that an optimal 3D transformation them can be estimated, minimizing reprojection errors in the image planes of the stereo vision system. This is a good approach to make something work without too much work on difficult vision algorithms, and it provides a reliable tool for object recognition with accuracies of less than 3 mm. In this way, stable and reliable grasping can be performed, such that focus can be put on other aspects of the system. It is unlikely that the approach of placing markers on bricks can be used in a final product, but so far, it serves as a practical solution. Having the bricks grasped with good accuracy, also facilitates the task of placing them as this error will be a smaller contribution to the final result.

- **Brick positioning** - It is a simple task to determine the desired position of each brick in an ideal wall. Therefore, the simulation of the wall building process also gives a very nice result. This is because in the simulation, we are sure of the accurate position of all objects, and therefore the robot will always position bricks perfectly. A system capable of determining the desired world coordinate of a brick has been developed. It is based on recognition and reconstruction of the transformation of a chessboard marker, which with the developed algorithms can be done with a very high accuracy. In this way it is also possible to compensate for the imprecise movement of the platform, which was one of the goals for the system.

- **Collision free platform navigation** - The navigation functionality for the platform used in the project is the one which was originally shipped with the Seekur platform. The collision safety is therefore rather limited, as it is only based on 2D range readings from the laser scanner. Therefore, it cannot be guaranteed that human interference with the environment will be detected. This is also the case for controlling the manipulator, since we are not able to detect dynamic obstacles in order to include these in the collision model to use for path planning. Also, the configuration of the manipulator is not taken into account by the platform navigation system, which is only based on a very low level model of the extent of the robot. Nevertheless, it is so that the manipulator is always placed in a "home" configuration when the platform is moved. Additionally, it is ensured, that if some movement of the manipulator fails, such that it does not end up in the home configuration, the application controller will halt execution and therefore the platform is not moved in such situation.

  As the wall which the robot is building reaches the level of the laser scanner, the robot will avoid collisions with it. This will, however, compromise the localization accuracy since the platform navigation system is not aware that the wall is supposed to be where it is observed. As long as it is still capable of reaching its goal position, that is not of main importance, because the stereo camera is used to compensate for the positioning errors it will lead to. Therefore, under the assumptions outlined and present for the restricted testing environment, this system has proved to be sufficient. This means that the robot is capable of autonomously driving between predefined locations as requested by the main application controller. A choice of collision model of the platform as well as a scaling of its acceleration did however improve reliability significantly.

- **Planning to place bricks** - In this project the RRT planner has been used successfully to make collision free paths in the scene. It means that the model of the scene has been made correctly, and that the planner is able to find paths in it. Optimization algorithms have been applied to reduce the length of the paths significantly, although the algorithms for that still are relatively slow. Simplifications of the collision model may serve as the easiest way of improving that. The constructed paths can be executed smoothly with high speeds and controlled errors using the developed algorithm for making blended trajectories.

- **Scheduling of building process** - The building strategy used in the project is a naive one, since one brick at a time is fetched and placed in an order simply corresponding to a brick index in the wall. Little concern has been given to the development of advanced strategies, that can determine the brick placement order differently. It was the aim to be able to consider different brick laying patterns and then develop methods for the robot to independently choose which bricks to place where under consideration of efficiency. The resulting building strategy has though, focused on one simple building task without patterns or different kinds of bricks, instead the efforts have been put into making the robot follow this strategy as robustly and autonomously as possible.

- **Evaluating wall quality** - A thorough investigation of the quality goals presently valid for walls built conventionally in construction has been performed. This has made it possible to determine how the quality of the wall built by the robot should be measured and presented. Since simplifications of the building task have been made, not all quality goals are relevant for this project. Although the quality of the wall built by the robot, is in general below conventional standards, they are indeed acceptable for this prototype. Many improvements to the robot can be thought of, so since the quality is comparable, it is indeed considered possible to reach the standards of conventionally built walls.

### 9.1.1 Brick placement accuracy

In this subsection consideration is given to the uncertainties, noise and errors of the system, seen in relation to the accuracy of the brick placement. It has been shown that the uncertainty in the chessboard recognition on which the brick placement is based, propagates to approximately ±1 cm at a 1 m distance. As bricks are laid at distances comparable to that, this is an estimate for the deviation of brick placement that the uncertainty in the chessboard recognition will lead to. As explained in section 7.4, this is a conservative estimate, so the experienced deviation from this will most likely be less.

Adding to the chessboard recognition uncertainty, is the system calibration uncertainty. The calibration evaluation of section 5.5 presented errors of up to ±0.5 cm on the position between a modeled frame and an observed frame. This means that errors of this size will also be present when bricks are placed since it is based on the accordance between the stereo vision observations and the manipulator movements. The uncertainty of the calibration result cannot be stated as noise, since the result will keep being what it is found to be. Therefore, it is rather a constant offset between the calibration result and the true model. However, this constant offset will manifest itself somehow unpredictable in different positions, as was seen in figure 5.20. Therefore it may anyhow be regarded as noise, which will then also be present on the brick placement positions.

In section 8.4.1 the measured standard deviation of the actual brick placement in the wall built was found to be 0.3 cm in both the x and y directions. As this means that all bricks are then expected to be positioned within a range of approximately 2 cm, the observed variation in brick placement can possibly be described by the uncertainties in the used methods and models. In that way, it can also be possible to predict what effect, an improvement in one of the other uncertainties, will have on the variance in the placed bricks,. This facilitates further development, as it can easily be detected if a certain part of the system is improved and how much that is expected to affect the final result.

### 9.1.2 Assumptions

Several assumptions have been made in order to realize the objective of developing a robot system capable of building a wall within the time frame of this project. In this section, the validity of these are evaluated.

Already at the beginning of the project, it was decided that dealing with mortar should not be part of this project. It is however a central part of the brick laying task, and a development of systems to handle that will be necessary before a product can be said to be ready. The handling of bricks and the placing of them in the mortar is also areas where the abilities and techniques of a bricklayer can be applied. This makes it possible to benefit from using mortar by compensating for brick imperfections by accurately adjusting the amount and level of mortar for each brick to lay, such that the quality of the wall continuously is upheld. These tasks are however difficult by themselves, and in order to focus on other aspects of brick laying this is completely omitted.

Other assumptions mainly concern the environment in which the robot is operating. As also mentioned in the entries of the bullet list in the beginning of the discussion section, obstacles must be detectable by the laser scanner when driving the platform and no unknown obstacles may be present when moving the manipulator. With respect to the manipulator this is a reasonable assumption, since it does not involve the brick laying process directly. This means that either it must be possible to assure that no obstacles can occur where the manipulator will be moving or else should one install sensors, e.g. ultra sonic or time-of-flight camera, which continuously can monitor the working area and provide information to the path planner about the scene and obstacles therein.

The extensive utilization of markers to provide information to the robot about the location of bricks and the wall reference frame, is also worth considering. It is of course unrealistic that chessboards can be printed on all bricks, but either there must be some kind of other identification mechanism as mentioned in 7.4.3, or else a complete other approach should be followed where each brick is not needed to be detected visually, but instead known to be in some kind of fixed container, possibly placed on the robot. The grasping based on vision is however a good starting point, and the algorithms developed as well as the evaluation of them can be used and compared to other robotic applications. It is however likely that a completely different approach to gathering bricks will be pursued in the future development of the system, as the time consumption when picking bricks one by one will unavoidably be large.

## 9.2 Conclusion

In this project the control system for a mobile manipulator capable of autonomously building a wall has been developed. This means that a user via an interface has the ability to specify the dimensions of a wall to be build and start the robot execution of the task. The robot will then move between two work stations during the building process until the wall is completed, one at which bricks are picked up and the other at which the bricks are placed according to the specified dimensions of the wall. Bricks are placed without the use of mortar, so the system works like performing several pick and place tasks, in between which, movement and re-localization of the platform are also taking place.

The subsystems needed by the application, and also developed or customized in this project, include governing of the different physical devices, platform, manipulator, gripper and stereo camera. For all of these, interfaces has been established which supports the functionality needed by the brick laying application. This both include requesting tasks to be executed and monitoring their execution states.

Also, the identification of bricks is highly important, and this has been facilitated using chessboard markers placed on the bricks. The precision of the position of this object recognition technique has shown on average to be within 2 mm of the actual position, and in the worst case within 3 mm. The precision of the rotation part has been evaluated to being within $0.6^\circ$.

In order to be able to pick up an identified brick, a calibration of the transformation between the camera and the manipulator has been conducted. Although, it is based on many data points from identified chessboard corners, it is found that an exact accordance between observations of the chessboard and the calibrated model was not obtained. For an independent validation dataset, the error between the two, in the position part contains errors of up to $\pm5$ mm and the rotation errors within $\pm2^\circ$. The uncertainty arising from the calibration error, and the one arising from the chessboard localization will add together on the accuracy by which a brick can be grasped by the robot, thus entail noise on the exact pose of the brick when held by the gripper.

The placement of bricks in the wall is based on the recognition of a wall frame given by a chessboard placed on the ground. The bricks are then placed relatively to this chessboard, and that will induce noise on the placement because of the dependency on the recognition uncertainty. When bricks are placed relatively far from the localized frame, especially the orientation of this frame becomes critical. At a 1 m distance from the frame, the precision of the brick placing position has been shown to be within 10 mm.

The quality of a wall, built by the robot, is dependent on the precision of the subsystems, and by measuring the actual position of the bricks in the wall, the standard deviation of the brick placement has be found to be 3 mm both across and along the wall. Additionally, the quality of the wall has been measured with respect to standards used for conventionally built walls in Denmark. By doing that the following is concluded about the wall built by the robot:

- The planarity does not meet the standards, as the brick position deviation has a range approximately 4.5 mm larger than the tolerance limit.
- The head joint space has variation much larger than the tolerance limits. The reason for this will be the same as for the above failure to meet the requirement, namely that the uncertainties within the system adds a too large noise to the placement of bricks.
- The deviation from vertical is measured both in the longitudinal and transversal directions, and in both of these there is a clear tendency, that the average slope is not zero as desired. Furthermore, the confidence intervals calculated for these slopes, show that the wall can both be constructed such that it lives up to the tolerances, but it can also be up to 2.5 times more oblique than these allow.

The non-fulfillment of the building standards is not critical to the project. Instead, the objective of actually building a wall, comparable to conventionally built walls, has been fulfilled. And based on the analyses conducted, it is possible to identify what improvements can be made in order to meet the masonry standards.

In order to make the robot build walls, a user interface for specifying how this must be done is created. For the simple tests, performed in the project, this has shown sufficient, but it is also clear that much more development must be put into this, when considering real wall applications.

The optimization of the efficiency of the robot performing masonry has not been pursued, and thus a simple approach for building walls, by picking and placing bricks one at a time is used. It has proven sufficient for the test setup, but again, it is a topic subject to apparent improvement possibilities. An evaluation of it is however present, such that it is possible to detect where improvements can be made.

Despite the simplifications decided upon, the overall objective of the project, to make up a mobile robot from several separate units and simultaneously controlling these, such that the robot is capable of autonomously moving around in a simple environment, while by itself identifying, picking up and placing bricks into a wall, has been successfully fulfilled. This verifies that the methods used are usable for the masonry application. Furthermore, since the robot is a prototype, and thus of course has high potential for enhancements, it is an important result of the project, that the effect of using these methods, has also been thoroughly investigated. This is because it will then be a much more simple task, when considering a certain increase of performance, to identify which subsystem or component need to be improved, renewed or added.

## 9.3 Perspectives

This section will try to set the realized prototype robot into a larger context, this means how such a system can be transformed from a prototype to an actual product. By doing this, the topics that are not covered in this report, such as mortar placement and mechanical design, are discussed such that sketches of how these issues can be approached can be drawn. Finally we will also try to describe how this system can participate in the work on future construction sites.

### 9.3.1 A possible application

Starting out, one can focus on what this robot actually is capable of - retrieving bricks and placing these in a precise position with respect to some frame. Without much further development, the robot can therefore serve as a bricklayer's assistant, automatically placing bricks in a favorable position with respect to where the bricklayer is to place the brick. This could in practice be done in the way that the robot simply places bricks with a little larger distance to the last placed brick than desired, such that the bricklayer then has some free space to put mortar on the brick and wall and squeeze the bricks into place. The robot can then place bricks in the desired positions many bricks ahead, such that the bricklayer will not need to wait for the robot. During this work, the robot could also take care of making mortar ready for the bricklayer. The bricklayer will then have the opportunity to focus on his most important job, the actual placing of the bricks, where the squeezing and fitting of the brick into place such that mortar is pressed evenly out into the joint space[1], a task that will not be trivial to program for a robot. We hereby also avoid the problem that the robot is not entirely able to meet the quality tolerances for the accuracy of brick placement. In this scenario, it can be preferable to let the robot and the bricklayer work on each side of the wall, making the risk of the two moving in the way of each other smaller. It is however to be determined what the benefits in terms of speed for such a system actually are, but it is sure that the wear of the bricklayer due to the many lifts can be reduced drastically. Making a prototype for this particular task would also be a feasible way of getting the robots out on real construction sites, such that experiences of using the system there can be obtained.

### 9.3.2 Handling mortar

The problem that is avoided in the above proposed application, where the bricks are not placed by the robot alone, is mortar. It is a completely different task to place mortar on bricks than to move the rigid bodies that bricks are. When considering how to make the system capable of handling mortar, one must be aware that it is not necessarily the best way to try to imitate how human bricklayers handle it, by applying it with a trowel, instead, using a more fluid type of mortar that could be pumped onto the bricks could be an approach, as this is already used in some parts of conventional brick laying, [55]. Using mortar that is more solid could also be possible, as it would then be easier to have the mortar placed on the bricks before the brick is moved to the wall, using some other kind of glue for the actual fastening of the bricks could also be possible. It is however likely that the appearance of the wall with mortar between the bricks is one of things people desire when choosing to use brick walls, so omitting the mortar space between bricks is not considered a feasible solution.

The robot could also be used in laying other types of bricks than standard bricks, i.e. leca[2] blocks, where conventional mortar may not be necessary.

### 9.3.3 The mechanical design

So far in the project, the mechanical design of the robot has not been evaluated very carefully, as the parts were provided at the beginning of the project without funds available for acquiring new equipment. It is however evident that the current design is not optimal for the purpose of masonry. The big and heavy platform with soft wheels and much suspension is made for driving with relative high speed through terrain and not for precise positioning. When the load distribution of the robot is altered, by for instance moving the manipulator, the suspension will give in, resulting in a displacement that may be large at the end of the manipulator.

A feature that can also be improved is how bricks are picked and transported. In the current version, the bricks are picked and transported one by one, this will of course not be feasible in any application, as the loop time as seen will be very long. Instead, it is preferred that the robot is able to carry many bricks, one possibility is that an entire pallet of bricks can be loaded onto the robot, whereas some of the design can be inspired by pallet transporting AGV's, noting that maneuverability on the construction site must still be kept at a reasonable level. This also alters

[1]Obviously, here we do not mean the joint configuration space of the robot, although it is likely that an early prototype mortar dispenser robot may show such behavior.

[2]Lightweight expanded clay aggregates.

the way the manipulator should pick bricks, as the position of these now can be known to a much higher degree. Normally bricks are stacked very tight on pallets when purchased, but it should be possible to stack these differently such that a parallel gripper can pick them, as a starting point this could be done manually. Another idea is to mount another manipulator or device able to pick the bricks from the pallet and place them in a fixture on the platform.

An issue that may be necessary for the robot to handle is also hacking of bricks, as some bricks near ends of walls may need to be shortened in order to obtain a straight edge of the wall. This can be done either by the robot or by having bricks of different sizes ready to the robot.

### 9.3.4 Safety

Safety is an important issue when dealing with robots, and most likely it will not be feasible to use a brick laying robot that is only certified to work behind a fence. If this shall not be the case restrictions are put on both the speed and power of the platform and manipulator. A way to approach this issue is simply to power off the motors of the manipulator while moving the platform and vice versa. Currently this would be difficult to do with the manipulator of Universal Robots, as it needs a manual initialization by jogging the joints for calibrating the encoders every time it is started. Alternatively, one must extend the obstacle detection system, from the current 2D laser scanner to a 3D mapping of obstacles in the vicinity of the robot. Devices for that, such as depth maps calculated by stereo vision, time of flight cameras or laser scanners mounted on a pan tilt unit, are available, but can to a greater extent be considered as experimental equipment than certifiable security solutions. It must however be noted that the standard 2D laser scanners are sufficient for standard AGV's.

### 9.3.5 Accuracy

The current approach to finding the transformation of the wall, the reconstruction of chessboard markers has served as a sufficient tool for placing bricks in a small wall. This technique however is difficult to scale to construction of larger walls, as the uncertainty in the desired position of a brick is proportional to the distance to the marker. Therefore, regardless the accuracy of the chessboard reconstruction algorithm, at some point the accuracy will be too low. A solution to that is to place several markers along the wall, this is possible but it will then be very difficult to place these accurately enough, since this should be done relative to the other markers. In general, it will be hard not to lose accuracy as dimensions increase. The current system also has the disadvantage that it is very sensitive to that the entire system is modeled and calibrated accurately enough.

#### A laser tracking approach

In the construction industry, laser solutions have become a popular tool to help construction workers make big machines pave in a straight line or dig holes of an accurate depth. These systems are extremely accurate in measuring depth and position, and the fact that they are already used in this sector, possibly makes acceptance of this technology much easier than the many other technologies for robots, presented in this report. Especially a new product from Leica Geosystems, the Laser Tracker Systems[21], is of keen interest. It consists of an absolute laser tracker and some devices with reflective markers. Concerning robotics, the system becomes interesting because it is possible to mount a so called "T-Mac" on the robot, making it possible to track its position and orientation with uncertainties of 60 $\mu$m ($2\sigma$) at a distance of 15 m. This would by far make it possible for the robot to meet the tolerances of construction work. Using this as input for servoing mechanisms, makes the possible accuracy of mobile manipulators even better than todays stationary robots. In addition, the system can also be used for robot calibration.

One issue existing when using this approach, is that the tracked object, the T-Mac in this case, must be orientated such that its reflectors does not point too far away from the tracker and obviously occlusions must not occur. The system is however able to find the tracked object again after an occlusion. This puts constraints on the placing of the tracker. When placed on the manipulator of a mobile robot for the most accurate control, it constraints its configuration space significantly. If it is placed on the platform the accuracy is still dependent on the robot calibration, which will also be good enough for most types of applications. Still, occlusions must not be encountered while executing the tasks, and therefore finding a suitable position of the tracker on a construction site is non-trivial.

The system can also act as a help for the planning for the platform, which will encounter difficulties in the changing environments of a construction site. In such an environment, it can be difficult to localize the robot, making also navigation impossible, when walls, which are the objects that are easiest to detect for the laser scanner, are being built during the time the robot is going to be on the construction site. Also objects like carts, piles of bricks and other building material tend to be moved around during the working period. A laser scanner also assumes a flat surface which might not be the case.

### 9.3.6 The future of automated construction

So far we have discussed and proposed solutions to how the robot can be evolved in order to increase performance and thus be a more and more competitive product compared to conventional brick laying. There is no reason why the proposed improvements cannot be implemented and then it is just a matter of will and enterprise to get the mobile brick laying robot out on the construction sites.

An important requirement for the success of the robot is that it can interact with the systems and people already present at the construction site, as it cannot be expected that robots will be able to take over all tasks for the time being. This interaction starts already at the desktop of the architect or engineer that designs the building. They use 3D CAD[3] programs to design the buildings, so it is preferable that the robot can read files from such a program in order to decide how to create the building. In that context it can be preferable that the actual design of where each brick exactly is supposed to be placed, is done beforehand on the computer, in many ways similar to how CAM[4] programs control robots in manufacturing.

When the robot then arrives on a construction site with the model of what it is going to construct, a construction site worker just has to set the position of a reference frame, in order to let the robot start the construction process. This can either be done with markers or by placing for instance a laser tracker at a fixed location at the construction site. An upcoming technology that could be used for placing this reference frame and verifying the placement of the building desired to construct is augmented reality. By using this technique, a user can see and move around in the building before it is built. This can both be used as a way to improve the design of the building, as it for instance is possible to experience the outlook from a window or how the building fits into the environment around it. In this context, it can be used to verify that the marker or tracker is placed correctly, such that the building will be built in the exact desired position.

The robot can then execute the placing of bricks by itself, or cooperate with other robots in doing it. In both ways, it is important that some CAM-like program is running on the robot or remotely controls the progress of the task. In connection to that, it will also be feasible that there exists some interface for the construction site worker to pause or modify the behavior of the robot.

[3]Computer-aided design.
[4]Computer-aided manufacturing.

# Bibliography

[1] Construction robotics. *Autonomous Robots*, 22:201–209, 2007.

[2] Apache. Apache license version 2.0 [online]. Available from: `http://www.apache.org/licenses/LICENSE-2.0` [cited 31/5 2010].

[3] Carlsberg Mur A/S. Præfabrikerede teglvægge [online]. Available from: `http://www.carlsbergmur.dk/Default.aspx?ID=1` [cited 31/5 2010].

[4] Department Robotics at GPS GmbH. Welcome to neobotix! [online]. Available from: `http://www.neobotix.de/en/index.html` [cited 31/5 2010].

[5] Simon Bøgh, Mads Hvilshøj, Christian Myrhøj, and Jakob Stepping. Fremtidens produktions-medarbejder - udvikling af mobilrobotten; "lille hjælper". Master's thesis, Aalborg Universitet, 2008.

[6] T. Bock, D. Stricker, J. Fliedner, and T. Huynh. Automatic generation of the controlling-system for a wall construction robot. *Automation in Construction*, 5(1):15–21, March 1996.

[7] T. Bonwetsch, F. Gramazio, and M. Kohler. Digitally fabricating non-standardised brick walls. In *ManuBuild, conference proceedings*, pages 191–196, 2007.

[8] Jean-Yves Bouguet. Camera calibration toolbox for matlab [online]. Available from: `http://www.vision.caltech.edu/bouguetj/calib_doc/` [cited 31/5 2010].

[9] Gary Bradski and Adrian Kaehler. *Learning OpenCV*. O'Reilly Media Inc., 2008.

[10] D.C. Brown. Decentering distortion of lenses. *Photometric Engineering*, 32(3):444–462, 1966.

[11] Dansk Byggeri. *Hvor går grænsen? Murerfaget*. Dansk Byggeri, 2007.

[12] John J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[13] E. R. Davies. *Machine Vision: Theory, Algorithms, Practicalities*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[14] Frédéric Devernay. C/c++ minpack [online]. Available from: `http://devernay.free.fr/hacks/cminpack.html` [cited 31/5 2010].

[15] Damien Douxchamps. libdc1394 homepage [online]. Available from: `http://damien.douxchamps.net/ieee1394/libdc1394/` [cited 31/5 2010].

[16] Storefront for Art and Architecture. Pike loop, a robot-built installation in nyc [online]. Available from: `http://www.storefrontnews.org/exhib_dete.php?exID=152` [cited 31/5 2010].

[17] Free Software Foundation. Gnu general public license [online]. Available from: `http://www.gnu.org/licenses/gpl.html` [cited 31/5 2010].

[18] Free Software Foundation. Lesser gnu general public license [online]. Available from: `http://www.gnu.org/licenses/lgpl.html` [cited 31/5 2010].

[19] E. Gambao, C. Balaguer, and F. Gebhart. Robot assembly system for computer-integrated construction. *Automation in Construction*, 9(5):479–487, September 2000.

[20] Willow Garage. Willow garage webpage [online]. Available from: `http://www.willowgarage.com/` [cited 31/5 2010].

[21] Leica Geosystems. Laser tracker systems [online]. Available from: `http://www.leica-geosystems.dk/dk/Laser-Tracker-Systems-eng_69045.htm` [cited 31/5 2010].

[22] Roland Geraerts and Mark H. Overmars. Creating high-quality paths for motion planning. *The International Journal of Robotics Research*, 26, 2007.

[23] Gene H. Golub and Charles F. Van Loan. *Matrix Computations (3rd ed.)*. Johns Hopkins, 1996.

[24] Point Grey. Bumblebee2 getting started manual [online]. Available from: `http://www.ptgrey.com/support/downloads/documents/Bumblebee2%20Getting%20Started%20Manual.pdf` [cited 31/5 2010].

[25] Richard Hartley and Andrew Zisserman. *Multiple view geometry*. Cambridge university press, 2003.

[26] MobileRobots Inc. Mobilerobots homepage [online]. Available from: `http://www.mobilerobots.com/Mobile_Robots.aspx` [cited 31/5 2010].

[27] MobileRobots Inc. Seekur information webpage [online]. Available from: `http://www.activrobots.com/ROBOTS/Seekur.html` [cited 31/5 2010].

[28] Point Grey Research Inc. Point grey [online]. Available from: `http://www.ptgrey.com/` [cited 31/5 2010].

[29] Wolfram Research. Inc. Mathematica some notes on internal implementation [online]. Available from: `http://reference.wolfram.com/mathematica/note/SomeNotesOnInternalImplementation.html#13028` [cited 31/5 2010].

[30] Danish Technological Institute. Tailorcrete [online]. Available from: `www.tailorcrete.com` [cited 31/5 2010].

[31] IRobot. PackBot [online]. Available from: `http://www.irobot.com/sp.cfm?pageid=109` [cited 31/5 2010].

[32] Richard A. Johnson and Dean W. Wichern. *Applied Multivariate Statistical Analysis*. Pearson Prentice Hall, 2007.

[33] Rune Søe Knudsen and Daniel Schlichting Kirkegard. Automated robot calibration. Master's thesis, University of Southern Denmark, 2010.

[34] Erwin Kreyszig. *Advanced Engineering Mathematics*. Wiley, 8 edition, 1999.

[35] James J. Kuffner and Steven M. LaValle. Rrt connect: An efficient approach to single query path planning. *In Preceedings of the International Conference of Robotics and Automation*, 2000.

[36] Manolis Lourakis. Levenberg-marquardt nonlinear least squares algorithms [online]. Available from: `http://www.ics.forth.gr/~lourakis/levmar/` [cited 31/5 2010].

[37] George F. Luger. *Artificial Intelligence Structures and Strategies for Complex Problem Solving, 5. ed.* Addison Wesley, 2005.

[38] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963. Available from: `http://link.aip.org/link/?SMM/11/431/1`, `doi:10.1137/0111030`.

[39] MobileRobots. Mapper3 software at mobilerobots wiki [online]. Available from: `http://robots.mobilerobots.com/wiki/Main_Page` [cited 16/5 2010].

[40] MobileRobots Inc. *Aria Documentation.*

[41] MobileRobots Inc. *ARNL Documentation.*

[42] MobileRobots Inc. *Seekur Quick Start.*

[43] Nokia. Qt - cross-platform application and ui framework [online]. Available from: `http://qt.nokia.com/` [cited 31/5 2010].

[44] Institute of Automation at Bremen University. Research projects [online]. Available from: `http://www.iat.uni-bremen.de/sixcms/detail.php?id=97` [cited 31/5 2010].

[45] OpenCV. OpenCVWiki [online]. Available from: `http://opencv.willowgarage.com/wiki/` [cited 31/5 2010].

[46] Emil Pedersen and Oluf S. Nielsen. Fork report: Control of a mobile manipulator. 2010.

[47] Emil Pedersen and Oluf Skov Nielsen. Brick laying mobile manipulator, robolab wiki [online]. Available from: `http://robolab.tek.sdu.dk/mediawiki/index.php/Brick_laying_mobile_manipulator` [cited 31/5 2010].

[48] D. L. Pieper. *The kinematics of manipulators under computer control.* PhD thesis, Stanford University, Department of Mechanical Engineering, 1968.

[49] Intel Labs Pittsburgh. Personal robotics [online]. Available from: `http://personalrobotics.intel-research.net/index.php` [cited 31/5 2010].

[50] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes (3rd ed.).* Cambridge university press, 2007.

[51] G. Pritschow, M. Dalacker J. Kurz, and J. Zeiher. A mobile robot for on-site construction of masonry. In *IEEE/RSJ/GI International Conference on Intelligent Robots and Systems, vol. 3*, pages 1701–1707, 1994.

[52] Nicolas Pugeault, Norbert Krueger, and Emre Baseski. Camera geometry and 3d-reconstruction. Available from: `http://homepages.inf.ed.ac.uk/npugeaul/ROB02/` [cited 31/5 2010].

[53] John Ratcliff. John ratcliff's code suppository - convex decomposition [online]. Available from: `http://codesuppository.blogspot.com/2009/11/convex-decomposition-library-now.html` [cited 31/5 2010].

[54] R.A. Rihani and L.E. Bernold. Methods of control for robotic brick masonry. *Automation in Construction*, 4(4):281–292, January 1996.

[55] Rami Awni Rihani. *An Investigation of Critical Success Factors for Robot Masonry*. Dissertation, North Carolina State University, RALEIGH, 2006.

[56] Universal Robots. Industrirobot UR-6-85-5-A [online]. Available from: `http://www.universal-robots.com/` [cited 31/5 2010].

[57] RobWork. Robwork webpage [online]. Available from: `http://www.mip.sdu.dk/robwork/joomla/` [cited 31/5 2010].

[58] Schunk. *PG70 Product sheet*.

[59] Lorenzo Sciavicco and Bruno Siciliano. *Modeling and Control of Robot Manipulators*. McGraw Hill, 1996.

[60] Lars Erik Skovgaard. Arbejdsstyrke rasler ned - stik mod hensigten [online]. Available from: `http://www.business.dk/oekonomi/arbejdsstyrke-rasler-ned-stik-mod-hensigten` [cited 31/5 2010].

[61] Bill Smart and Mark Yim. Virtual manufacturing automation challenge (vmac) [online]. Available from: `http://icra.wustl.edu/?q=2010vmac` [cited 31/5 2010].

[62] Morten Strandberg. Yaobi webpage [online]. Available from: `http://yaobi.sourceforge.net/` [cited 31/5 2010].

[63] Murværk Teknologisk Institut. Murtag.dk [online]. Available from: `http://www.mur-tag.dk/` [cited 31/5 2010].

[64] Universal Robots. *UR-6-85-5-A produktblad*. Available from: `http://www.universal-robots.com/Produkter/Produktblad.aspx` [cited 31/5 2010].

[65] Universal Robots. *UR-6-85-5-A User Manual*.

[66] Universal Robots. *The URScript Programming Language*.

[67] Michael D. Wagner, Dimitrios Apostolopoulos, Kimberly Shillcutt, Benjamin Shamah, Reid Simmons, and William (Red) L. Whittaker. The science autonomy system of the nomad robot. In *2001 IEEE International Conference on Robotics and Automation*, pages 1742–1749, May 2001.

[68] Seung-Nam Yu, Byung-Gab Ryu, Sung-Jin Lim, Chang-Jun Kim, Maing-Kyu Kang, and Chang-Soo Han. Feasibility verification of brick-laying robot using manipulation trajectory and the laying pattern optimization. *Automation in Construction*, 18:644–655, 2009.

# Appendix A

# Covariance of the estimated calibration parameters

The parameters estimated in the calibration are

$$P^{camera}_{manipulator} \equiv \begin{bmatrix} p_{x,1} & p_{y,1} & p_{z,1} & \phi_1 & \theta_1 & \psi_1 \end{bmatrix}^T \tag{A.1}$$

$$P^{gripper}_{TCP} \equiv \begin{bmatrix} p_{x,2} & p_{y,2} & p_{z,2} & \phi_2 & \theta_2 & \psi_2 \end{bmatrix}^T \tag{A.2}$$

$$\bar{\Delta\theta} \equiv \begin{bmatrix} \Delta\theta_2 & \Delta\theta_3 & \Delta\theta_4 & \Delta\theta_5 \end{bmatrix}^T \tag{A.3}$$

They are concatenated to the complete $16 \times 1$ parameter vector $P$:

$$P = \begin{bmatrix} p_{x,1} \\ p_{y,1} \\ p_{z,1} \\ \phi_1 \\ \theta_1 \\ \psi_1 \\ p_{x,2} \\ p_{y,2} \\ p_{z,2} \\ \phi_2 \\ \theta_2 \\ \psi_2 \\ \Delta\theta_2 \\ \Delta\theta_3 \\ \Delta\theta_4 \\ \Delta\theta_5 \end{bmatrix} \tag{A.4}$$

For the parameter vector $P$ the upper left $8 \times 8$ submatrix of the covariance matrix $\Sigma_P$ is:

$$\Sigma_P(1:8, 1:8) =$$

$$10^{-7} \cdot \begin{bmatrix} 0.0918 & -0.0211 & 0.0075 & 0.0073 & -0.0907 & 0.0967 & 0.0249 & 0.0267 \\ -0.0211 & 0.0958 & -0.0103 & 0.0348 & -0.0325 & 0.0351 & -0.0375 & -0.0444 \\ 0.0075 & -0.0103 & 0.0790 & -0.1266 & -0.0137 & 0.0055 & -0.0002 & 0.0056 \\ 0.0073 & 0.0348 & -0.1266 & 0.2545 & -0.0121 & 0.0310 & -0.0041 & -0.0053 \\ -0.0907 & -0.0325 & -0.0137 & -0.0121 & 0.2457 & -0.0287 & -0.0006 & 0.0005 \\ 0.0967 & 0.0351 & 0.0055 & 0.0310 & -0.0287 & 0.3426 & 0.0003 & -0.0052 \\ 0.0249 & -0.0375 & -0.0002 & -0.0041 & -0.0006 & 0.0003 & 0.1025 & -0.0155 \\ 0.0267 & -0.0444 & 0.0056 & -0.0053 & 0.0005 & -0.0052 & -0.0155 & 0.1167 \end{bmatrix} \tag{A.5}$$

The lower right 8 × 8 submatrix is

$$\Sigma_P(9:16, 9:16) =$$
$$10^{-5}\cdot\begin{bmatrix} 0.0004 & -0.0068 & -0.0056 & -0.0007 & 0.0000 & -0.0000 & 0.0002 & 0.0001 \\ -0.0068 & 0.2033 & 0.0544 & -0.0028 & 0.0000 & 0.0001 & -0.0002 & 0.0007 \\ -0.0056 & 0.0544 & 0.2960 & 0.0374 & -0.0002 & 0.0001 & 0.0021 & 0.0033 \\ -0.0007 & -0.0028 & 0.0374 & 0.6173 & 0.0003 & 0.0002 & -0.0035 & -0.0020 \\ 0.0000 & 0.0000 & -0.0002 & 0.0003 & 0.0004 & -0.0000 & -0.0001 & -0.0000 \\ -0.0000 & 0.0001 & 0.0001 & 0.0002 & -0.0000 & 0.0001 & -0.0001 & -0.0000 \\ 0.0002 & -0.0002 & 0.0021 & -0.0035 & -0.0001 & -0.0001 & 0.0072 & 0.0025 \\ 0.0001 & 0.0007 & 0.0033 & -0.0020 & -0.0000 & -0.0000 & 0.0025 & 0.0071 \end{bmatrix} \tag{A.6}$$

Finally, the upper right 8 × 8 is

$$\Sigma_P(1:8, 9:16) =$$
$$10^{-6}\cdot\begin{bmatrix} -0.0002 & -0.0162 & -0.0096 & -0.0068 & -0.0007 & 0.0000 & -0.0037 & -0.0004 \\ -0.0009 & 0.0119 & 0.0051 & 0.0027 & 0.0001 & -0.0000 & 0.0011 & -0.0043 \\ -0.0008 & 0.0003 & -0.0035 & 0.0087 & -0.0002 & 0.0000 & -0.0018 & 0.0004 \\ 0.0009 & -0.0027 & 0.0008 & -0.0066 & 0.0002 & -0.0000 & 0.0012 & -0.0020 \\ 0.0007 & 0.0125 & -0.0013 & -0.0015 & 0.0009 & 0.0000 & 0.0017 & 0.0040 \\ 0.0006 & -0.0038 & -0.0211 & -0.0096 & -0.0007 & 0.0001 & -0.0013 & 0.0014 \\ -0.0010 & 0.0402 & -0.0173 & 0.1723 & -0.0005 & 0.0001 & -0.0005 & 0.0022 \\ -0.0022 & 0.0350 & 0.0772 & -0.1483 & 0.0004 & 0.0000 & -0.0029 & -0.0018 \end{bmatrix} \tag{A.7}$$