Task Programming of Redundant Industrial Robots

A Virtually Extended Null Space Formulation
Verified Through Obstacle Avoidance

Audun Rønning Sanderud

Master of Science in Product Design and Manufacturing
Submission date:  June 2012
Supervisor:      Trygve Thomessen, IPK

Norwegian University of Science and Technology
Department of Production and Quality Engineering
MASTER THESIS

2012

for

stud. techn. Audun Rønning Sanderud

TASK PROGRAMMING OF REDUNDANT INDUSTRIAL ROBOT
(Høynivåprogrammering av industrirobot med overtallige frihetsgrader)

TASK TO BE SOLVED

Industrial robots have nowadays normally six degrees of freedom to provide an arbitrary position and orientation of the tool inside its working space. However, during the last years, there have been developed industrial robots with more than six axes. This gives extra functionality, i.e., to change the internal reconfiguration of the robot arm to avoid singular positions, to move around obstacles and to optimize the use of energy during a predefined trajectory.

However, this functionality is dependent on the user’s ability to program the robot efficiently and skillfully which makes the programming complicated and time consuming for redundant robots.

This master thesis is focusing on how to enable this extra functionality automatically so the operator can simply focus on the programming of the tool position, similar to a normal six axes robot, while the control system automatically take care of the internal configuration of the robot arm. This includes:

- Automatic reconfiguration of the robot arm to avoid obstacles. Thus, provide sensors on the robot arm, which detects when any part of the robot arm is close to an obstacle. An automatic algorithm has to be developed to automatically change of the internal configuration of the robot arm to avoid the obstacle.
- Automatic reconfiguration of the robot arm to avoid singularities. This is to be done by continuously monitoring the internal configuration of the robot arm, and detect when the robot arm is close to any singularities. Then, an automatic algorithm has to be developed to automatically reconfigure the robot to avoid to run further into the singular area.
- During run time of the robot, automatically optimize the internal kinematic configuration according to less energy use.
- There has also to be developed an overall algorithm which automatically selects the best configuration of the robot arm in case there is any contradiction between the configuration of the robot arm to avoid obstacles and singularities. This algorithm should have a set of rules which can be defined and prioritised by the user.

The solutions shall be developed in PPM’s laboratory in Trondheim, for NACHI MR20 industrial robot, with PPM’s high speed USB interface to NACHI’s AX20 controller.

The implementation shall be done using a standard personal computer (PC) communicating with the OLIMEX card and the collision sensors attached to the MR20 robot. The detection of singularities is done by reading the MR20’s axes positions through the OLIMEX card which communicates with high speed with NACHI’s AX20 controller.

The following tasks have to be accomplished:

i. Introductory literature study about methodology for collision- and singularity avoidance in addition to optimization of energy use during running the robot along a predefined trajectory.


iii. Development of methodology for energy optimization when running the robot along a predefined trajectory.

iv. Development of hardware solution and instrumentation of the robot system.

v. Development of software solution for the robot system.
vi. Experimental testing of the MR20 robot system for collision- and singularity avoidance.

vii. Documentation of the experimental setup including hardware and software.

viii. Documentation of the user functions to operate the experimental setup.

ix. Documentation of the experimental results.

Within three weeks after the date of the task handout, a pre-study report shall be prepared. The report shall cover the following:

- An analysis of the work task’s content with specific emphasis of the areas where new knowledge has to be gained.
- A description of the work packages that shall be performed. This description shall lead to a clear definition of the scope and extent of the total task to be performed.
- A time schedule for the project. The plan shall comprise a Gantt diagram with specification of the individual work packages, their scheduled start and end dates and a specification of project milestones.

The pre-study report is a part of the total task reporting. It shall be included in the final report. Progress reports made during the project period shall also be included in the final report.

The report should be edited as a research report with a summary, table of contents, conclusion, list of reference, list of literature etc. The text should be clear and concise, and include the necessary references to figures, tables, and diagrams. It is also important that exact references are given to any external source used in the text.

Equipment and software developed during the project is a part of the fulfilment of the task. Unless outside parties have exclusive property rights or the equipment is physically non-moveable, it should be handed in along with the final report. Suitable documentation for the correct use of such material is also required as part of the final report.

The candidate shall follow the work regulations at the company’s plant. The candidate may not intervene in the production process in any way. All orders for specific intervention of this kind should be channelled through company’s plant management.

The student must cover travel expenses, telecommunication, and copying unless otherwise agreed.

If the candidate encounters unforeseen difficulties in the work, and if these difficulties warrant a reformation of the task, these problems should immediately be addressed to the Department.

The assignment text shall be enclosed and be placed immediately after the title page.
Deadline: June 11th 2012.

Two bound copies of the final report and one electronic (pdf-format) version are required.

RESTRICTIONS
PPM has the rights to use the results from the Master Thesis for any purpose.

Responsible supervisor at NTNU and PPM AS:

Professor Trygve Thomessen
Managing Director
PPM AS
Leirfossveien 27
NO-7038 Trondheim
Norway
Phone: + 47 92 24 21 89
E-mail: tth@ppm.no

DEPARTMENT OF PRODUCTION
AND QUALITY ENGINEERING

Per Schjølberg
Associate Professor/Head of Department

Trygve Thomessen
Responsible Supervisor
Summary

Industrial robots are an important part of modern automation and are used in a variety of applications such as handling, welding, painting and assembling. They have normally six degrees of freedom to provide an arbitrary location of the tool inside its working space. However, during the last years, there have been developed industrial robots with more than six axes, thus redundant robots. This gives extra functionality to avoid singular positions, to move around obstacles and to optimize the use of energy during a predefined trajectory, by changing the internal configuration of the robot arm while still maintaining the tool's location.

However, programming of redundant robots is complicated and time consuming due to the fact that both the tool location and the internal configuration of the robot arm have to be programmed.

Investigations on the use of redundant industrial robots in the industry reveal several advantages including highly increased flexibility and a significant reduction of space need. The flexibility can be attributed through obstacle avoidance, singularity avoidance and energy optimization. For small and medium enterprises (SMEs) and in High-mix Low-volume productions, this gives a great advantage in even more competitive markets.

This thesis presents an efficient approach for programming of redundant industrial robots. The system uses proximity sensors mounted on the robot arm to detect obstacles. By analyzing the sensor data, the system can automatically reconfigure the robot's arm to automatically, comply with environmental constraints. Enabling this functionality by an automatic system, simplified the programming of the redundant robot to be similar to a normal six-axes robot.

Studies into the subjects that constitute the theoretical basis for the practical implementation and kinematic resolutions have been done. This includes, among others, kinematic analysis, redundancy resolutions, task formulation and methods for obstacle detection. The studies led to a suggestion for an task description scheme based on an extension of Mason's task formulation for force controlled tasks. The formulation augments the robots self-motion ability to be based upon a virtual extension of the robot's Null space. The virtual extension allows the operator to select the priority of the secondary task, subsequently programming the robot as if it were a six-axis robot.

The system has been implemented and experimentally verified on a NACHI MR20 seven-axes industrial robot. The implemented system includes Cartesian velocity limiter, Joint space velocity limiter, Task Reconstruction algorithm, Default arm reconfiguration and path correction algorithm. The sensors system is based on ultrasonic and infrared proximity sensors, covering the greater part of the robot arm.

The experiments proved convincing performance and robustness of the implemented system. It was shown that the extended null space formulation can redistribute certain axes
from the primary task to the secondary task, and thus, provide automatic obstacle avoidance. The obstacle avoidance strategy was shown to be successful, and gave the desired evasive maneuver. Experiments also demonstrated the system's ability to reconfigure the primary task after deflection caused by the secondary task and the ability to reconfigure the arm to a default configuration when both the task is reconstructed and no obstacles are present.
Sammendrag

Industriroboter er en viktig del av moderne automatisering, og brukes i en rekke applikasjoner som håndtering, sveising, maling og montering. De har normalt seks frihetsgrader for å kunne oppnå alle mulige plasseringer av verktøyet innenfor arbeidsområdet. Men i løpet av de siste årene har det blitt utviklet industrielle roboter med mer enn seks akser, disse kalles redundante roboter. Dette gir ekstra funksjonalitet for å unngå singulære posisjoner, manøvrere rundt hinderinger og for å optimalisere bruken av energi gjennom en forhåndsmåned definert bane, ved å endre den interne konfigurasjonen til robotarmen og samtidig opprettetholde verktøyets plassering. Men programmering av redundante roboter er komplisert og tidkrevende, fordi både verktøyets beliggenhet og den interne konfigurasjonen av robotarmen må programmeres.

Bruk av redundante roboter i industrien viser flere fordeler, inkludert kraftig økt fleksibilitet og en betydelig reduksjon av plassbehov. Fleksibiliteten kan tilskrives energidisponering, hindrings- og singularitetsunngåelse. For små og mellomstore bedrifter og i høy-variasjon lavt-volum produksjoner, gir dette en stor fordel i et stadig mer konkurranseutsatt markeder.


Forsøkene viste overbevisende ytelse og robusthet i det implementerte systemet. Det ble vist at det utvidede nullromsformulering kan redistribuere angitte akser fra den primære oppgaven til den sekundære oppgaven, og dermed gi automatisk hindringsunngåelse. Hindringsunngåelsesstrategien viste seg å være vellykket, og ga den ønskede unnvikevei manøver. Eksperimenter viste også systemets evne til å rekonstruere den primære oppgaven etter for-
flytning forårsaket av den sekundære oppgaven. Når både oppgaven var rekonstruert og ingen hindringer var til stede viste systemet evnen til å rekonfigurere armen til en standard konfigurasjon.
Preface

This thesis is written as partial fulfillment of an integrated masters degree in mechanical engineering at NTNU, Trondheim. The research and thesis has been carried out by stud.techn. Audun R. Sanderud on behalf of PPM AS and NTNU, and will contain both theoretical studies as well as practical implementations.

The problem description has been developed by Professor Trygve Thomessen and concerns task programming of redundant industrial robots. After discussion with Prof. Thomessen were the problem description slightly modified compared to the original version enclosed in the beginning of this thesis. Studies into singularity avoidance and energy use optimization is no longer considered part of this master thesis.

The project will be evaluated on the basis of a written report, as well as other material pertaining to it.

Trondheim, June 11, 2012

Audun R. Sanderud

Approval about the changes in the problem description:

Trondheim, June 11, 2012

Prof. Trygve Thomessen
Acknowledgment

I would like to take the opportunity to thank Professor Trygve Thomessen for great supervision, support and encouragement during the project, and for valuable advice and feedback on drafts. His mood and enthusiasm rubs off on everyone. I would also like to thank him for giving me access to PPM AS’ robot lab, and for placing important materials at my disposal.

I would also like to thank stud.techn. Fredrik Reme, PhDc Balazs Daniel and MSc Kosuke Wada for technical discussions. We have had numerous interesting and informative conversations about the content of this master thesis.

I also thank my parents for their support and encouragement, and for reading my draft and giving valuable feedback. Thanks also to Thomas Eeg for valuable feedback on one of my drafts. A special thanks goes to Ida Moe Evensen for her patience, moral support and advice throughout the work with this thesis.

A.R.S
# Contents

Summary ................................................. v
Sammendrag .............................................. vii
Preface .................................................. ix
Acknowledgment ......................................... xi

1 Introduction 1
   1.1 Background ........................................ 1
      1.1.1 Industrial robots and Programming .......... 1
      1.1.2 High-mix Low-volume Production .......... 2
      1.1.3 Redundant Industrial Robots ............. 2
   1.2 Problem Formulation ............................. 2
   1.3 Limitations ........................................ 4
      1.3.1 Timespan and Workload .................... 4
      1.3.2 Hardware and Software .................... 4
   1.4 Available Hardware .............................. 4
   1.5 Software .......................................... 5
   1.6 Approach .......................................... 6
   1.7 Structure of the report .......................... 6

2 Historical Perspective and Related Work 9
   2.1 Programming Support ........................... 9
   2.2 Obstacle Avoidance and Sensor Systems ....... 10
   2.3 Singularity Avoidance ........................... 11

3 Theory of Kinematics And Task Formulation 13
   3.1 Kinematics ...................................... 13
   3.2 Kinematic Analysis .............................. 14
   3.3 Redundancy Resolution .......................... 15
   3.4 Denavit-Hartenberg Formulation ............... 19
   3.5 Task Formulation ................................ 21
   3.6 Redundant Robots in Industrial Tasks .......... 23
      3.6.1 Human Interaction ........................ 24
      3.6.2 Process Control ........................... 24
      3.6.3 Space Efficiency ........................... 26
# Task Formulation with Redundant Robot

- **4.1 Redundancy Resolution**
- **4.2 Virtually Extended Null Space**
  - **4.2.1 Weighted Virtually Extended Null Space**
  - **4.2.2 Extension of Masons Task Formulation**
  - **4.2.3 Selection Matrix Transformation**

# Obstacle Avoidance as a Secondary Task

# Applications to NACHI MR20 robot

- **6.1 Kinematics of the MR20**
  - **6.1.1 Model With All Joints**
  - **6.1.2 Model With Virtual Extension**
- **6.2 Sensor Placement and Kinematics**
  - **6.2.1 Determining Avoidance Vectors**
  - **6.2.2 Sensor Kinematics and Avoidance Vectors**
- **6.3 Sensory System Hardware Structure**
  - **6.3.1 Proximity Sensors**
  - **6.3.2 Controller**
  - **6.3.3 Computer**
  - **6.3.4 Olimex**
- **6.4 System System Software Structure**
  - **6.4.1 FPGA**
  - **6.4.2 Micro Controller**
  - **6.4.3 PC**
  - **6.4.4 Olimex**
  - **6.4.5 Modeling in LabVIEW**
- **6.5 Installation of Sensor System**
  - **6.5.1 Connector Block C1**
  - **6.5.2 Proto-Board**
  - **6.5.3 Infrared Sensors**
  - **6.5.4 Ultrasonic Sensors**

# User guide

- **7.1 Startup Procedures**
- **7.2 LabVIEW User Interface**

# Evaluation of the System

- **8.1 Shoulder sensor**
  - **8.1.1 No extended Avoidance**
  - **8.1.2 With extended Avoidance**
- **8.2 Elbow Sensor**
  - **8.2.1 No extended Avoidance**
Chapter 1

Introduction

1.1 Background

1.1.1 Industrial robots and Programming

Industrial robots are a very important part of modern automation and are used in a variety of applications such as handling, welding, painting and assembling (Figure 1.1). In the beginning the programming of the required functions were done by the operator of the robot. One of the most common ways was by use of a teaching pendant (programming pendant). The teaching pendant is the operators interface with the robot. The robot arm is moved directly from the pendant, and the pendant records the motion. After completed programming the program can be played back and the recorded motion is repeated.

Figure 1.1: The NACHI MC20 6-axis Robotic Arm (NACHI Robotic Systems Inc., 2012).
1.1.2 High-mix Low-volume Production

Industrial robots are useful for improving productivity, stability of quality and saving human cost. Especially in mass production are industrial robots widely used, e.g. welding and painting robots in an assembly plant of an automobile manufacturer. However, in high-mix low-volume production, they may also be effective. E.g. an assembly process for electronic components usually requires many kinds of dedicated machines to deal with its wide variety of work and change of their demands. Introducing industrial robots instead of the dedicated machines may achieve high-mix low volume manufacturing while suppressing the initial investment. In this way, the flexibility of industrial robots can bring the benefits in reducing manpower and capital investment. However, the programming is a cumbersome process and can be very time consuming if the tasks for the industrial robot change frequently.

Systems for supporting the programming have been researched in recent years, e.g. automatic control system with image processing technology, programming software using pre-mapping information and analytical approaches. However, users need to learn how to use the equipment and tools beforehand. It is desirable that the support system for robot programming is easy to handle for the user.

1.1.3 Redundant Industrial Robots

An object in the real world has six degrees of freedom (DOF): position along the x, y and z axes and rotation about the x, y and z axis. A robot arm is required to have six DOF to reach any given position and orientation in space (hereby referred to as pose). The six DOF in the robot is represented as joints. A given combination of the joint angles gives one certain position in space, and it is only this combination that will give that very pose. When introducing another joint to the robotic arm, the robot will have seven DOF and it is a so-called redundant robot as seen in Figure 1.2. The robot now has the ability to reach the same pose with several different joint angle combinations (hereby referred to as arm configuration). This gives extra functionality in practical use of the robot and has advantages such as obstacle and singularity avoidance. However, the programming of the redundant robots is more complicated and more time consuming than that of a normal six axes robot. Application of the extra functionality therefore depends on the user's ability to program the robot motion.

1.2 Problem Formulation

This master thesis is focusing on how to ease the programming of redundant industrial robots so that the extra functionality is enabled automatically. The operator should be able to simply focus on the programming of the end-effector position, similar to a normal six axes robot, while the control system automatically take care of the internal configuration of the
1.2. PROBLEM FORMULATION


Figure 1.2: The NACHI MR20 7-axis Robotic Arm (NACHI Robotic Systems Inc., 2012).

robot arm. This includes automatic reconfiguration of the robot arm to avoid obstacles. This will be achieved by providing sensors on the robot arm, which detects when any part of the robot arm is close to an obstacle. An automatic algorithm has to be developed to automatically change the internal configuration of the robot arm to avoid the obstacle.

Objectives

The following tasks have to be accomplished:

i. Introductory literature study about methodology for collision avoidance.
ii. Development of methodology for collision avoidance on the MR20 robot.
iii. Development of hardware solution and instrumentation of the robot system.
iv. Development of software solution for the robot system.
v. Experimental testing of the MR20 robot system for collision avoidance.
vi. Documentation of the experimental setup including hardware and software.
vii. Documentation of the user functions to operate the experimental setup.
viii. Documentation of the experimental results.

The solutions is developed in PPM’s laboratory in Trondheim, for a NACHI MR20 industrial robot, with PPM’s high speed USB interface to NACHI’s AX20 controller. The implementation is done using a standard personal computer (PC) communicating with the OLIMEX card and the collision sensors attached to the MR20 robot.
1.3 Limitations

1.3.1 Timespan and Workload

The defined start of the project was when the project description was handed out January 16\textsuperscript{th} 2012. The master thesis, thus the ending of the project, was due June 11\textsuperscript{th} the same year. Including one week holiday, this gave a total of 100 working days. The thesis was rewarded 30 credits, one credit is estimated to be the equivalent of 1.6 hours work per week. The total workload for the thesis is thus estimated to be 960 hours.

1.3.2 Hardware and Software

The hardware regarding the robot was limited to the industrial robot, olimex interface and computer described in 1.4. Other than that were there few limitations to the hardware. There were no limitations on the software.

1.4 Available Hardware

The anticipated system structure will be as shown in Figure 1.3. The available hardware for this system was at the beginning of the work with this thesis what’s described below.

![Figure 1.3: Principal structure of the system.](image)
1.5. SOFTWARE

**Industrial Robot Arm**
The industrial robot available for use during the project was a NACHI MR20 7-axis industrial robot with a NACHI AX20 Controller.

**Computer**
A computer with an Intel Core i7 950 processor clocked at 3.07GHz with 6GB of memory was used during all experiments.

**Olimex Interface**
The Olimex real-time interface is a Linux single board that is connected between the NACHI AX20s main- and servo board (Figure 1.4). The Olimex intercepts the signals and sends out new manipulated signals depending on the interpretations provided by code running on the computer. The software running on the Olimex is written by Johannes Schrimpf. See appendix E.5 for more information.

![Figure 1.4: The Olimex real-time interface.](image)

**Sensors**
An infrared and an ultrasonic sensor were available. The infrared proximity sensors were the GP2Y0AA0YK from Sharp. The ultrasonic distance sensors were the Parallax PING_TM. See appendix E.4 for more information.

1.5 **Software**

There were no limitations on what software could be used. However, a LabVIEW framework for communication with all relevant hardware made by Balazs Daniel was available. This, combined with the candidate's experience with LabVIEW, reasoned for coding the system using LabVIEW 2011 with National Instruments' Robotics Module for all code on the computer side.
The thesis is typeset with \LaTeX. Flow charts and block diagrams are made using tikz environment. The models and figures are made using Google sketchup, Adobe Illustrator CS4 and Photoshop CS4 and/or Blender. Photos are edited in Adobe Photoshop CS4

1.6 Approach

The project is based on an experimental environment and consider a redundant robotic manipulator that is installed in PPM’s laboratory in Trondheim.

After studying the theory of kinematic analysis and redundancy resolutions a simulation platform for testing and development the kinematics was created. This enabled simulation of the robot to confirm that certain aspects of the kinematic model was functioning before applying it to the MR20. When the simulations gave satisfactory results, it was implemented with the provided framework for communication with the Olimex and the gripper. The development of laws for the obstacle avoidance system was conducted consecutively. After initial testing of the sensors and the obstacle avoidance laws the full sensor system developed and installed on the MR20. The full system was tested thoroughly. After satisfactory laws were achieved the work was continued on to evolve the robustness and functionality of the program. This was done by including several algorithms to ensure the reliability of the program. Studies into subjects that made up the theoretical basis for the practical implementation and experiments were performed as the challenges arose.

1.7 Structure of the report

The rest of this thesis is structured as follows.

Chapter 1 presents background information, problem formulation and other formalities regarding this thesis.

Chapter 2 gives a historical perspective on the use of redundant industrial robots, and approaches to obstacle and singularity avoidance. It will give an introduction through summarizing the most important events in the development of redundancy resolutions from the early seventies up until today.

Chapter 3 explains the theoretical background of redundant kinematics and redundancy resolutions. It will describe some of the most important approaches and how to model the kinematics using the Denavit-Hartenberg convention. The chapter will then explain the theory for partially constrained force controlled tasks. Lastly the chapter will present some effects the implementation of redundant industrial robots in the industry may have.
Chapter 4 presents a resolution for the redundant kinematics and a way to utilize this for secondary tasks, i.e. obstacle avoidance. The chapter then describes the workings of the weighted extended null space formulation and how it is used.

Chapter 5 will regard obstacle avoidance as a secondary task. The necessary calculations and considerations will be presented.

Chapter 6 describes how the system was implemented on a NACHI MR20 7-axes robot. The chapter will thoroughly explain all hardware and software that was developed to conduct the experiments and verifications carried out in this thesis.

Chapter 7 is a user guide in how to use the proposed hardware and software solutions that was developed throughout this project.

Chapter 8 covers different experiments conducted to demonstrate and test the capabilities of the developed system for obstacle avoidance.

Chapter 9 will summarize the report through a conclusion, discussion of the results, and suggest a few topics for further work.
Chapter 2

Historical Perspective and Related Work

Over the last 15-20 years particular attention has been devoted to redundant robotics. Their unique flexibility in tasks requiring the dexterity comparable to the one of the human arm has been recognized as a major advantage. This chapter will give an introduction to the history behind, and some related work with, obstacle- and singularity avoidance. Topics regarding kinematic analysis and redundancy resolution will be thoroughly discussed in chapter 3.

2.1 Programming Support

There have been some approaches for supporting the programming of industrial robots. One of the approaches is informational support which gives the operator feedback such as status of the robot and recommendation of operation by using visual, haptic or auditory interfaces. For example, a remote control system for an industrial robot based on cognitive info communication has been proposed (Thomessen et al., 2011). This system uses sound in combination with video to communicate the complex state of the industrial robot system to the user who operates the robot remotely.

If operators can recognize the robot’s situation immediately, they are able to make a program of the robot motion efficiently. Another solution is to generate motion of a robot arm automatically (Iossifidis and Schöner, 2006). Their system achieved automatic collision avoidance of a robot arm using stereo cameras and an attractor dynamics approach to generate a collision free path for the robot arm (Figure 2.1). In general, sensors to detect obstacles and control algorithms are required to realize an automatic navigation system for a robot with collision avoidance. Observation of the working environment is one of the most important functions for the robot navigation. Therefore, a control system should have a sensor system or environmental maps to perceive its surroundings.
2.2 Obstacle Avoidance and Sensor Systems

In many studies of obstacle avoidance a map of the environment is required beforehand to generate a motion of a robot arm. When the workspace is limited and static, it is possible to prepare the environmental map in advance. However, the map should be adjusted to present situation when the workspace changes. This modification of environmental maps requires a certain skill other than the robot programming for users and also causes extra time to prepare robot programs. Exact algorithms to cases where all information about the environment is known \textit{a priori} have been considered by Reif (1979), Hopcroft et al. (1982) and Schwartz and Sharir (1983).

There are several methods to observe the workspace of a robot during operation, for example infrared sensors mounted on a robot arm (Gandhi and Cervera, 2003), stereo camera (Iossifidis and Schöner, 2006), and multiple cameras distributed around the robot (Thomessen et al., 2011). These kind of approaches can provide real-time information, which is more robust to environmental changes than using predefined information. Since a reliable sensor system to obtain environmental information effectively is critical it is important to consider location and arrangement of sensors. Gandhi and Cervera (2003) installed infrared range sensors on the surface of a robot to detect objects around the robot for collision avoidance. Many other researchers have also placed sensors to detect obstacles on the surface of the robot (Balasubramanian et al., 2009; Jincong et al., 2009; Chen and Juang, 2009; Borenstein and Koren, 1990; Fox, 1996; Arras et al., 2002; Fox et al., 1997).

One of the major approaches for robot control with collision avoidance is potential field method (Iossifidis and Schöner, 2006; Borenstein and Koren, 1990; Fox, 1996; Arras et al., 2002; Fox et al., 1997). The main idea of the method is using an attractive and repulsive force. In an obstacle avoidance system, the attractive force to the goal and repulsive forces to obstacles are adapted to the robot for the decision of its motion. However, the approach sometimes makes the robot go to a local minima which means the robot fails to reach its goal because the robot is at a dead end of road or locked by the forces

Lumelsky (1987) presented what he called the Dynamic Path Planning algorithm (DPP). The
algorithm disposes the need of *a priori* information about the environment and the information about the environment was purely sensor based. Nemec and Zlajpah (2002) presented a velocity level based method for obstacle avoidance, placing motion components on the robot arm at critical points. The obstacles were detected with a simple vision detection system. This method will be further investigated and implemented later in this thesis. Cheung and Lumelsky (1988) addressed the hardware and low-level control issues related to the DPP.

### 2.3 Singularity Avoidance

Wampler (1986) introduced the damped least square method to overcome singularities. But this damped least square method has its disadvantages in loss of performance and increased tracking error (Chiaverini, 1997). Egeland et al. (1991) demonstrated a singularity robust solution to the redundancy problem using extended Jacobian with weighted damped least-square method. Singularities were avoided using simple constraints and position transformations. The choice of damping constant must be balanced with the required performance and allowed error. Nakamura et al. (1987) introduced a variable damping factor and numerical filtering of the velocity component to overcome these drawbacks. Chiaverini (1997) proposed a modified inverse, using the damping factor only on the lowest singular values. Their results are shown to be more satisfying than those of the classical damped least square method, but there still is the problem of tuning the damping factor.
Chapter 3

Theory of Kinematics And Task Formulation

This chapter will present a kinematic analysis followed by a presentation of some of the most common ways to deal with redundant kinematics today. This is followed by the theory behind the Denavit-Hartenberg formulation which is used to model all necessary kinematics in this thesis. Lastly will Mason’s Task Formulation for partially constrained tasks be presented.

![Image of MR20 robot utilizing all seven axes](image)

Figure 3.1: The MR20 utilizing all seven axes to hold an orientation in a trajectory (Appendix E).

3.1 Kinematics

Using a seven-axes industrial robot as shown in Figure 3.1 gives new opportunities, but also new challenges compared with standard six-axes robots. Seven degrees of freedom simply
cannot be determined in a six dimensional space by conventional inverse kinematics. There will be infinitely many solutions for most poses. This, on the other hand, gives several new opportunities, i.e. regarding obstacle avoidance and avoiding singularities. Sections 3.2 and 3.3 are based on excerpts from Patel and Shadpey (2005).

### 3.2 Kinematic Analysis

A manipulator is said to be redundant when the dimension of the task space, \( m \), is less than the dimension of the joint space, \( n \). Let the \((m \times 1)\) vector \( p \) denote the pose of the end-effector in task space, and the joint angles in joint space by the \((n \times 1)\) vector \( q \). The degree of redundancy is then defined as \( r = n - m \) \((r \geq 1)\). The forward kinematics is then solved with a forward kinematics function defined as

\[
p = f(q)
\]

The differential kinematics is then given by

\[
\dot{p} = J_e \dot{q}
\]

Furthermore can the solution at acceleration level be expressed as

\[
\ddot{p} = J_e \ddot{q} + \dot{J}_e \dot{q}
\]

where \( J_e \) is the \((m \times n)\) Jacobian of the end-effector. The Jacobian can be determined from the below equation.

\[
J_e = \frac{\partial f(q)}{\partial q}
\]

\( J_e \) can in any case be viewed as a linear transformation mapping from \( R^n \) into \( R^m \), in this case mapping \( \dot{q} \in R^n \) into \( \dot{p} \in R^m \). Linear transformations give two fundamental subspaces with some important properties (Figure 3.2). First of all the range, \( \mathcal{R}(J_e) \), is a subspace of \( R^n \) and defined by

\[
\mathcal{R}(J_e) = \{J_e \dot{q} | \dot{q} \in R^n\}
\]

The range defines in other words the set of \( \dot{p} \) vectors reachable with any possible set of \( \dot{q} \). The second and most important in redundant robotic control is the null space. The null space, \( \mathcal{N}(J_e) \), is a subspace of \( R^n \) and is defined by
3.3. REDUNDANCY RESOLUTION

The null space is thus a set of $\dot{q}$ vectors that will all give the same $\dot{p}$ vector. The dimension of $\mathbb{N}(J_e)$ is equal to $(n - \text{rank}(J_e))$, thus being equal to the redundancy when $J_e$ is full rank. The null space therefore only exists for redundant manipulators. Any joint velocity applied within null space, denoted $\dot{q}_n$, will not affect the task space velocities. This motion is called the self-motion of the robotic arm. This is where the great advantage of redundant manipulators lie, additional constraints can be satisfied while not interrupting the primary task specifying the pose of the end-effector.

![Figure 3.2: Geometric representation of the null space and range of $J_e$.](image)

3.3 Redundancy Resolution

Most robot controllers generates commands in Cartesian space, in case of controlling a redundant robot these control inputs must be projected into joint space. The redundancy can be solved on both position, velocity and acceleration level. This thesis will focus on a velocity level based redundancy resolution. The goal being to find $\dot{q}$ for a given $\dot{p}$ that exactly satisfies (3.2). This thesis will focus on solutions based on the pseudo inverse, $J_e^\dagger$, of the transformation matrix $J_e$.

$$\dot{q} = J_e^\dagger \dot{p}$$

(3.7)

This equation represents a general form of the widely used solution based on the least square problem defined by:
\begin{equation}
\text{min}_q = ||J_e q - \dot{p}||
\tag{3.8}
\end{equation}

The pseudo inverse is as mentioned denoted \(J_e^\dagger\) and can be expressed by:

\begin{equation}
J_e^\dagger = \sum_{i=1}^{m'} \frac{1}{\sigma_i} \hat{v}_i \hat{u}_i^T
\tag{3.9}
\end{equation}

Where \(\sigma_i\), \(\hat{v}_i\) and \(\hat{u}_i^T\) can be found from singular value decomposition of \(J_e\) (Golub and Van Loan, 1989). The \(\sigma_i\) are the non-zero singular values of \(J_e\). If \(J_e\) has full rank is its pseudo inverse given by:

\begin{equation}
J_e^\dagger = J_e^T (J_e J_e^T)^{-1}
\tag{3.10}
\end{equation}

The pseudo inverse has the ability to provide a meaningful solution to the least square problem in (3.8), regardless (3.2) being under-specified, square or over-specified. This makes it a very attractive solution to the redundancy problem. However, Maciejewski and Klein (1989) pointed out some weaknesses. As the robot approaches a singular configuration the singular value will approach zero \((\sigma_i \rightarrow 0)\). The norm of the solution of (3.7) then becomes very large. The pseudo inverse can therefore not guarantee non-singular configurations. Another drawback is that the extra degrees of freedom caused by redundancy is not utilized. To enable user defined secondary tasks the self motion, previously described, is added to the equation (Zghal, 1988):

\begin{equation}
\dot{q} = \dot{q}_p + \dot{q}_N
\tag{3.11}
\end{equation}

This will still satisfy (3.2) because \(\dot{q}_N\) will not affect \(\dot{p}\) from the definition in (3.6). \(\dot{q}_N\) can easily be obtained by projecting a \(n \times 1\) vector, \(\dot{\theta}\), into null space.

\begin{equation}
\dot{q}_N = (I - J_e^\dagger J_e) \dot{\theta}
\tag{3.12}
\end{equation}

The solution for \(\dot{q}\) is thus

\begin{equation}
\dot{q} = \dot{q}_p + \dot{q}_N = J_e^\dagger q + (I - J_e^\dagger J_e) \dot{\theta}
\tag{3.13}
\end{equation}

If \(\dot{\theta}\) is selected based upon the gradient of a cost function,

\begin{equation}
\dot{\theta} = \nabla \Psi = \frac{\partial \Psi}{\partial q} = \left[ \frac{\partial \Psi}{\partial q_1} \ldots \frac{\partial \Psi}{\partial q_i} \ldots \frac{\partial \Psi}{\partial q_n} \right]^T,
\tag{3.14}
\end{equation}
the method is defined as the *Gradient Projection Method* (GPA). Many have defined suitable cost functions to obtain different objectives, such as singularity avoidance (Nakamura and Hanafusa, 1986), acceleration minimization (Seraji, 1991) and obstacle avoidance (Baillieul, 1986; Colbaugh et al., 1989).

Egeland (1987) and Sciavicco and Siciliano (1988) presented the *Extended Jacobian Method* (EJM) where the Jacobian, \( J_e \), is augmented to include a user defined task:

\[
J_A = \begin{bmatrix} J_e \\ J_c \end{bmatrix}
\]  

(3.15)

Where \( J_A \) is the augmented Jacobian including the \( r \times n \) Jacobian, \( J_c \), defining the secondary task. The velocity kinematics are then given by:

\[
\dot{p}_A = \begin{bmatrix} \dot{p} \\ \dot{p}_S \end{bmatrix} = J_A \dot{q}
\]  

(3.16)

where \( \dot{p}_A, \dot{p} \) and \( \dot{p}_S \) are the extended-, primary-, and secondary task velocities respectively. The above equation is no longer redundant and redundancy resolution is achieved through extending the kinematics at velocity level. There are however drawbacks with this solution as well (Seraji and Colbaugh, 1990a). First of all the additional task must have the same dimensions as the degree of redundancy. This significantly limits possible secondary tasks. The second problem is the artificial singularities that occurs in addition to the kinematic singularities in the main task.

Both the GPA and EJM are referred to as exact solutions because of their ability to exactly satisfy (3.2). However, the problems related to these solutions motivated an alternative approach. The exact solution of (3.2) was extended to also take into account the accuracy. The method is referred to as the *Damped Least-Squares Method* (DLSM) and has been used in different forms by e.g. Wampler (1986) and Nakamura and Hanafusa (1986). They defined the least-square criterion as where a damping factor, denoted \( \lambda \), specifies the tracking accuracy and the norms of joint rates. The criterion is defined by:

\[
\begin{bmatrix} |J_e \dot{q}|^2 + \lambda^2 |\dot{q}|^2 \end{bmatrix}
\]

(3.17)

The original ((3.2)) can then be expressed:

\[
\begin{bmatrix} J_e \\ \lambda I \end{bmatrix} \dot{q} = \begin{bmatrix} \dot{p} \\ 0 \end{bmatrix}
\]

(3.18)

The solution for the above equation can then be found by solving:

\[
(J_e^T J_e + \lambda^2 I)\dot{q} = J_e^T \dot{p}
\]

(3.19)
where the least-square solution is given by:

\[ \dot{q}_\lambda = (J_e^T J_e + \lambda^2 I)^{-1} J_e^T \dot{p} \]  

(3.20)

The strength of this solution is its ability to give a unique solution that approximates the task velocities with joint velocities not exceeding \( \|\dot{q}_\lambda\| \). To compare this method to the Pseudo inverse based least square problem, the singular value decompositions (SVD) can be compared.

\[ (J_e^T J_e + \lambda^2 I)^{-1} J_e^T = \sum_{i=1}^{m'} \frac{\sigma_i}{\sigma_i^2 + \lambda^2} \dot{v}_i \dot{u}_i^T \]  

(3.21)

Setting the damping factor \( \lambda \) to zero in the above equation gives the exact same SVD as the one for the Pseudo inverse of \( J_e \).

\[ \frac{\sigma_i}{\sigma_i^2 + \lambda^2} \bigg|_{\lambda=0} = \frac{1}{\sigma_i} \]  

(3.22)

Secondly can it be seen that when the configuration is close to a singularity \( (\sigma_i \to 0) \) the damping factor will keep the norm from increasing to intolerable levels.

\[ \frac{\sigma_i}{\sigma_i^2 + \lambda^2} \bigg|_{\sigma_i \to 0} \approx \frac{1}{\lambda} \]  

(3.23)

Accordingly can it be seen that when the configuration is far from a singularity (the singular value \( \sigma_i \) is large) the damping factor will have little effect on the solution, and again the solution will be approximately the same as the Pseudo inverse based solution.

\[ \frac{\sigma_i}{\sigma_i^2 + \lambda^2} \bigg|_{\sigma_i \gg 0} \approx \frac{1}{\sigma_i} \]  

(3.24)

This means that the damping factor will only have effect if it is larger or of the same order as the singular values. If so it will reduce the norm of the joint velocities by

\[ \|\dot{q}_\lambda\| \leq \frac{1}{2\lambda} \|\dot{p}\| \]  

(3.25)

Based on the advantages of the Damped Least-Square method Seraji (1989, 1991) and Seraji and Colbaugh (1990b) proposed Configuration Control for redundancy resolution. The proposal was to augment the \((m \times 1)\) primary task vector, \(\dot{p}\), by a secondary task defined by the \((k \times 1)\) vector, \(\dot{p}_s\). The augmented \((m+k) \times 1\) task vector is denoted \(\dot{p}_A\) with the corresponding differential kinematics:
3.4. DENAVIT-HARTENBERG FORMULATION

A good model of the kinematics is a prerequisite before any calculations can be done. The most widely used way to express kinematic properties is the Denavit-Hartenberg (DH) convention (Denavit and Hartenberg, 1955, 1964). This method makes it possible to develop the

\[
\dot{p}_A = \begin{bmatrix} \dot{p} \\ \dot{p}_S \end{bmatrix} = J_A \dot{q}
\]

(3.26)

where

\[
J_A = \begin{bmatrix} J_e \\ J_s \end{bmatrix}
\]

(3.27)

is the augmented Jacobian with the \((m \times n)\) primary, and \((k \times n)\) secondary task Jacobians, \(J_e\) and \(J_s\), respectively.

The solution proposed by Seraji and Colbaugh (1990b) suggests the following solution using a weighted damped least squares method (WDLSM):

\[
\dot{q} = \left[ J_e^T W_e J_e + J_c^T W_c J_c + W_v \right]^{-1} \left[ J_e^T W_e \dot{p}_d + J_c^T W_c \dot{p}_s \right]
\]

(3.28)

which minimizes the cost function

\[
\Upsilon = \dot{E}_e^T W_e \dot{E}_e + \dot{E}_c^T W_c \dot{E}_c + \dot{q}^T W_v \dot{q}
\]

(3.29)

where \(W_e(m \times m)\), \(W_c(k \times k)\) and \(W_v(n \times n)\) are diagonal positive-definite weighting matrices assigning priority between the primary, secondary, and singularity robustness tasks. \(\dot{E}_e\) and \(\dot{E}_c\) are vectors representing the residual velocity errors of the tasks. The significant difference between this method and the extended formulation (equation (3.26)) is that there is no restriction on the dimensions, \(k\), of the secondary task. The solution gives the best joint speeds in the least-square sense when \(k > r\) and the joint velocities are minimized when \(k < r\). The weighting matrix \(W_v\) ensures, nonetheless, the boundedness of the joint velocities.

However, all the presented methods require expertise on redundant robots. For SMEs might it be to expensive to employ specialist for robot programming, and for High-Mix Low-Volume manufacturers is it to tedious to reprogram redundant robots using the aforementioned methods. A simplified method where a redundant robot can be programmed with only a basic knowledge about robots would therefore be a great advantage for SMEs and High-Mix Low-Volume manufacturers.
translation matrix based on four parameters per link, compared to the six needed in homogeneous transforms. In DH formulation one defines the links between joints, rather than the joints themselves. The rotation of the joint is always around the z-axis, and the x-axis is always normal on the current and last z-axis (see Figure 3.3). The remaining part of this section and the following section is based on excerpts from previous work done by the candidate in a specialization project (Sanderud and Reme, 2011).

Figure 3.3: The Denavit-Hartenberg Formulation.

Where the link parameters are defined as follows:

- **Offset Distance** $d_n$ displacement along $Z_{n-1}$ from $X_{n-1}$ to $X_n$ and perpendicular to both
- **Rotation** $\theta_n$ rotation about $Z_{n-1}$ from $X_{n-1}$ into $X_n$
- **Length** $r_n$ displacement along $X_n$ from $Z_{n-1}$ to $Z_n$ and perpendicular to both
- **Twist Angle** $\alpha_n$ rotation about $X_n$ from $Z_{n-1}$ to $Z_n$

Using the standard homogeneous rotation matrices for rotation about x-, y- and z-axis, see appendix B, the corresponding transformation matrices for the link parameters can be developed as follows:

$$T_{d_n} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ \hspace{1cm} (3.30)

$$T_{\theta_n} = \begin{bmatrix} \cos(\theta_n) & -\sin(\theta_n) & 0 & 0 \\ \sin(\theta_n) & \cos(\theta_n) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ \hspace{1cm} (3.31)
3.5. TASK FORMULATION

\[
T_r = \begin{bmatrix}
1 & 0 & 0 & r_n \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]  
(3.32)

\[
T_{\alpha_n} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \cos(\alpha_n) & -\sin(\alpha_n) & 0 \\
0 & \sin(\alpha_n) & \cos(\alpha_n) & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]  
(3.33)

Multiplying them together gives the transformation matrix for link \( n \) from joint \( n - 1 \) to \( n \).

\[
T_{n-1}^n = T_{dn} T_{\theta_n} T_r T_{\alpha_n}
\]  
(3.34)

\[
T_{n-1}^n = \begin{bmatrix}
\cos(\theta_n) & -\sin(\theta_n)\cos(\alpha_n) & \sin(\theta_n)\sin(\alpha_n) & r_n\cos(\theta_n) \\
\sin(\theta_n) & \cos(\theta_n)\cos(\alpha_n) & -\cos(\theta_n)\sin(\alpha_n) & r_n\sin(\theta_n) \\
0 & \sin(\alpha_n) & \cos(\alpha_n) & d_n \\
0 & 0 & 0 & 1
\end{bmatrix}
\]  
(3.35)

The total transformation matrix for the robotic manipulator with \( n \) links may then be formulated as:

\[
T_0^n = T_1^n T_2^n \ldots T_{n-1}^n
\]  
(3.36)

### 3.5 Task Formulation

Mason (1979) formalized the theory for partially constrained task. His work focused on force controlled tasks, but the formulations regarding partial constraints are nonetheless very interesting. In force controlled tasks, the robot’s end-effector is in physical contact with the environment. This means that the robot is free to move in some directions while, in other directions, it may be physically constrained by the environment of which it is in contact with.

Mason (1979) puts the model for general force control between the two concepts of: pure force control and pure position control. In pure force control the user provides \( f(t) \), which is the vector function of time for the forces to be exerted by the end-effector. Conversely, in pure position control, the user can specify the end-effector’s position trajectory by supplying the position vector of time, \( p(t) \).
The idea that the model for general force control is intermediate between these two, becomes apparent if one considers situations that illustrate these concepts taken to their extremes. The examples Mason (1979) used for this were the following: Consider the end-effector buried in an immobile stiff solid substance. As there is no positional freedom, pure position control is meaningless. In a pure force controlled situation however, the manipulation will have full force freedom, as any forces defined by the $\mathbf{f}(t)$ vector can be "absorbed" by the substance. The opposite can be seen in a situation where the end-effector is in free space, giving the pure position control full positional freedom while for pure force control there is no possible source in the environment for the forces to occur. In a sense, pure force control and pure position control can be considered to be dual concepts.

One can consider surfaces to be the intermediate between the two situations of solid- and free space. One can conclude that neither pure position- nor pure force control is appropriate to handle these partially constrained tasks. The freedom of motion on a surface occurs along its tangents, while the force freedom along the surface normals. In order to handle this, a hybrid control model, at a point between pure force- and position control, based on the tasks constraints has to be used.

Mason (1979) called the constraints imposed by the environment "natural constraints", and further suggested a control methodology based on expanding by using a set of "artificial constraints" appropriate for the task at hand.

A couple of examples of how this set of constraints can be used for task formulation will be presented based on Bruyninckx and Schutter (1996) and Thomessen (2012).

Below, a figure of the well-known peg-in-hole problem, accompanied by tables showing the natural constraints imposed by the environment and its appropriate artificial constraints, can be seen.

![Diagram of peg-in-hole problem](image)

<table>
<thead>
<tr>
<th>Natural Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_x = 0$</td>
</tr>
<tr>
<td>$F_z = 0$</td>
</tr>
<tr>
<td>$V_y = 0$</td>
</tr>
<tr>
<td>$M_z = 0$</td>
</tr>
<tr>
<td>$\omega_x = 0$</td>
</tr>
<tr>
<td>$\omega_y = 0$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Artificial Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_x = 0$</td>
</tr>
<tr>
<td>$V_z = V_{0z}$</td>
</tr>
<tr>
<td>$F_y = 0$</td>
</tr>
<tr>
<td>$\omega_z = 0$</td>
</tr>
<tr>
<td>$M_x = 0$</td>
</tr>
<tr>
<td>$M_y = 0$</td>
</tr>
</tbody>
</table>

Table 3.1: Peg-in-hole.

The peg's motion is constrained by the hole. From the position perspective, one can see that both $V_x$ and $V_y$ as well as $\omega_x$ and $\omega_y$ has to be zero if the peg is to not collide with the walls of the hole. From the force perspective, by idealizing the situation so that there is no friction, one can see that $F_z$ and $M_z$ should be zero if the peg moves perfectly into the hole. These constraints together constitutes the *natural constraints* of the peg-in-hole problem.
However, in addition to the constraints imposed by the environment, *artificial constraints* has to be introduced in order to specify the desired motion and force applied for the task. In order for the peg to run undisturbed into the hole, $F_x$, $F_y$, $M_x$ and $M_y$ has to equal zero, while the $z$-direction can be position controlled with $V_z$ equal to the programmed speed $V_{0z}$ and $\omega_z = 0$

The analysis of the above problem, has been performed in a coordinate system appropriate for studying the contact between the part and its environment during the task. This coordinate system is called the *task coordinate system*, and can be situated on the end of a peg or the end of a tool, depending on the task at hand.

In order to specify what directions are to be force- or position controlled in the task coordinate system, the *selection matrix*, $S$, is used in the robot control system. The selection matrix is a 6x6 matrix where the diagonal specifies the control method. A "1" signifies that the direction is force control, and a "0" that the direction is position controlled. Conversely, the matrix in (3.37), results in compliance for $F_x$, $F_y$ and $M_z$ while $V_z$, $\omega_x$ and $\omega_y$ is kept position controlled.

$$
S = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 
\end{bmatrix} \tag{3.37}
$$

### 3.6 Redundant Robots in Industrial Tasks

In many applications, there is no need for more than a six-axes robot, as it is sufficient to define any pose of the end-effector in space. The gains of having more than six axes can mostly be attributed to more flexibility in the robot arm configuration referred to as self-motion (see Figure 3.4). In terms of using a seven-axes robot, it can increase the space of which the robot is able to work, i.e. easily grind under a table or picking and placing in more complex environments. This also reduces the space needed for the industrial robot since it can reach further in a complex environment. In addition, by introducing more joints, one can distribute joint movements to the redundant joints, reducing cases where some joints would move considerably more than others.
3.6.1 Human Interaction

Human interaction with robots is to some extent in use in the industry today as seen in Figure 3.5. When an industrial robot is being used as a third hand for a worker use of a redundant robot will increase the flexibility of the system. The robot will be able to, for instance, hold a work piece in several different configurations as it can be seen in Figure 3.6. This would ease the workers accessibility to the workpiece. This may also reduce the number of required grippings, which in turn reduces time in production.

Figure 3.5: Two workers collaborating and working together with traditional six-axis robots.

3.6.2 Process Control

Since a redundant robot has more degrees of freedom than it needs to perform its primary task it is very flexible when it comes to implementation of secondary tasks. These may include the following.
Obstacle Avoidance

Obstacle avoidance that does not affect the task at hand is simply not possible with only six-degrees of freedom. The redundant robots self-motion ability gives it the flexibility it needs to keep the tool stationary while reconfiguring its arm to avoid an obstacle. This allows a redundant robot to reconfigure to reach places unreachable for a six-axis robot as seen in Figure 3.7.

Singularity Avoidance

The same reasoning may be applied to singularity avoidance as to obstacle avoidance. While jogging, the control system may detect a possible upcoming singularity, and then reconfigure the arm in a fashion so that the robots configuration never reaches its singular pose.
Energy Consumption and Torque optimization

To reduce the energy consumption of the larger joints the smaller joints may be prioritized when using redundant robots. If the robot is performing short-ranged tasks, it may use the smaller, and less energy consuming, ones. When moving heavier objects may the larger joints be used more to reduce the torque and stress in the smaller joints, thus prolonging their lifetime. This can also be achieved through keeping the greater component of the reaction forces parallel to the rotation axis of the smaller joints.

Joint Limit Avoidance

Due to the extra joint the robot is able to select such an arm configuration throughout a predefined path, so that it does not encounter any joint limits.

3.6.3 Space Efficiency

Figure 3.8: A six axis (left) and a seven axis (right) industrial robot used to load and unload parts.

Redundant industrial robots also have a great advantage when it comes to required space. A robot with seven axes used in a loading system can be placed on the machines side, as opposed to directly in front of the door as a traditional six axis robot would require (See Figure 3.8 and 3.9). This feature reduces the space required in front of the machine and gives easier access for the operator for maintenance and operation. According to NACHI, their MR20 seven-axes industrial robot can reduce the requirement for space in front of the machine with 70% (NACHI Robotic Systems Inc., 2012).
3.6. REDUNDANT ROBOTS IN INDUSTRIAL TASKS

Figure 3.9: A six axis (left) and a seven axis (right) industrial robot used to load and unload parts.

(a) (Makino Engineering Services, 2010).  (b) (NACHI Robotic Systems Inc., 2012).
Chapter 4

Task Formulation with Redundant Robot

4.1 Redundancy Resolution

The redundancy resolution used in this thesis is based on the self-motion ability for a redundant manipulator where a joint motion vector is projected in Null Space and the resulting velocity vector $\dot{q}_N$ is added to the joint velocity vector $\dot{q}_P$ calculated by solving the inverse kinematics for a Cartesian velocity $\dot{p}$. A secondary task is subject for projection and is thus the basis for the vector denoted $\vartheta$. When dealing with a velocity controlled system $\dot{q}$ is then given by (4.1) from section 3.3

$$\dot{q} = \dot{q}_P + \dot{q}_N = J_e^\dagger \dot{q} + (I - J_e^\dagger J_e) \vartheta$$  \hspace{1cm} (4.1)

If the secondary task is defined by a motion in task space on a given point on the manipulator, $\vartheta$ will be the corresponding motion in joint space for the joints required to accomplish the motion in task space. $\vartheta$ is thus a $1 \times n$ dimensional vector, calculated based on (4.2). Where $n$ denotes the number of dimensions in joint space.

$$\vartheta = J_s^\dagger \dot{p}_s$$  \hspace{1cm} (4.2)

Where $J_s$ is the Jacobian for the the point on the arm where the secondary task is present, and $\dot{p}_s$ is the velocity vector in task space.

A path correction algorithm will later be described. The redundancy resolution in this algorithm is a least square solution characterized by

$$\min_{\dot{q}} = ||J_e \dot{q} - \dot{p}||$$  \hspace{1cm} (4.3)

The solution is based on the pseudo inverse of the Jacobian, $J_e$, at the robots end-effector.
4.2 Virtually Extended Null Space

The theory of Mason (1979), states that many primary tasks are not fully naturally constrained. Based on his theory for force control of compliant tasks a method to enable certain dimensions in the secondary task was developed. If a manipulator has $n$ degrees of freedom, the redundancy is $r = n - m$, where $m$ is the dimensions of the task space. $r$ is then also the dimension of $\mathbb{N}$ if $J_e$ is full rank. This means that the dimensions available for the secondary task is equal to the degree of redundancy. When dealing with a 7 DOF robot the secondary task will only have one degree of freedom. This naturally limits the flexibility and possibilities of the secondary task. In cases where the primary task does not require a full set of six dimensions to perform its task, the degrees of freedom will not being used go to waste. The idea is therefore to enable these dimensions in the secondary task.

To do so, the null space was virtually extended with six dimensions. The virtual extension is modeled as three prismatic joints and three rotational joints mounted on the robots end-effector as shown in Figure 4.1. One joint corresponding to each of the dimensions in task space. The corresponding virtual extension Jacobian matrix is denoted $J_v$. The original Null Space being an $n \times n$ dimensional matrix makes the extended Null Space, $\mathbb{N}^V$, matrix $(n + 6) \times (n + 6)$ dimensional. A Denavit-Hartenberg matrix for the virtual extension is shown in table 4.1.

![Diagram of virtual extension](image)

Figure 4.1: The virtual extension were modeled as three prismatic axis and three rotational axis to represent the degrees of freedom in task space.
4.2. VIRTUALLY EXTENDED NULL SPACE

<table>
<thead>
<tr>
<th>Link</th>
<th>$\theta$</th>
<th>d</th>
<th>r</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>$d_{v1}$</td>
<td>$\pi$</td>
<td>$\frac{\pi}{2}$</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>$d_{v2}$</td>
<td>$-\frac{\pi}{2}$</td>
<td>$\frac{\pi}{2}$</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>$d_{v3}$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>$\theta_v4 + \pi$</td>
<td>0</td>
<td>0</td>
<td>$\frac{\pi}{2}$</td>
</tr>
<tr>
<td>5</td>
<td>$\theta_v5 + \frac{\pi}{2}$</td>
<td>0</td>
<td>0</td>
<td>$\frac{\pi}{2}$</td>
</tr>
<tr>
<td>6</td>
<td>$\theta_v6$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.1: Denavit-Hartenberg Matrix for the virtual joint extension

Which gives the virtually extended Jacobian:

$$J_{ve} = \begin{bmatrix} J_e \\ J_v \end{bmatrix}$$ (4.4)

The virtually extended self motion $\dot{q}_{vN}$ can now be found by:

$$\begin{bmatrix} \dot{q}_{vN} \\ \dot{q}_v \end{bmatrix} = \left( I - J_{ve}^\dagger J_{ve} \right) \begin{bmatrix} \dot{\theta} \\ 0 \end{bmatrix}$$ (4.5)

With the virtually extended null space, a projection of $\dot{\theta}$ into the null space will also give a velocity to the joints in the virtual extension. And a different set of joint velocities to the real joints than it would without the extended null space. The new suggested position calculated from $\dot{q}_S = \dot{q}_P + \dot{q}_{vN}$ are denoted $p_S$ (see Figure 4.2). And the position calculated from $\dot{q} = \dot{q}_P$ is the reference position and is denoted $p_R$. These two positions are then used to calculate the Cartesian velocity, $\Delta \dot{p}$.

![Figure 4.2: The Suggested point $p^S$ and the reference point $p^R$ on the robot with virtual extension.](image-url)
\[ \Delta \mathbf{p} = \mathbf{p}_R - \mathbf{p}_S = q_P J_e - q_S J_e \]  

(4.6)

By enabling projections in the virtually extended null space, one redistribute more dimensions to the secondary task space. However, this will obstruct the robot from realizing the primary task. A virtually extended null space based on a weighted Jacobian is therefore introduced. This will be more thoroughly explained in the next section.

### 4.2.1 Weighted Virtually Extended Null Space

To improve the flexibility and controllability of when, and to what degree, the extended null space is to be utilized a weighted extended null space is proposed. The null space is thus expressed based on a Jacobian matrix and a \((m \times m)\) weighting matrix denoted \(W\). The weighting matrix is a diagonal positive-definite matrix of the form

\[ W = \text{diag} \left[ w_1 \ w_2 \ \ldots \ w_n \right] \]  

(4.7)

where \(w_i\) is the diagonal element of \(W\) determined based on the selection matrix as it will be explained in the following section. The elements, \(w_i\), denote the secondary task’s priority in task space. If \(w_i = 1\) the secondary task will have full priority to manipulate the end-effector velocity along axis-\(i\). If \(w_i = 0.5\) is the secondary and primary task equally prioritized when performing tasks. One strength of this is that the system is fully determined even when a secondary task is not present because the Cartesian velocities, \(\dot{p}\) will always be defined as long as there is a primary task due to its least square approach.

The weighted Jacobian can thus be determined by

\[ J_W = \begin{bmatrix} J_e \\ J_v W \end{bmatrix} \]  

(4.8)

And the pseudo inverse follows by

\[ J_W^+ = \begin{bmatrix} J_e \\ J_v W \end{bmatrix}^T \left( \begin{bmatrix} J_e \\ J_v W \end{bmatrix} \begin{bmatrix} J_e \\ J_v W \end{bmatrix}^T \right)^{-1} = J_W^T (J_W J_W^T)^{-1} \]  

(4.9)
This concludes with the following solution for the calculation of the joint velocities, $\dot{q}$:

\[
\begin{bmatrix}
\dot{q} \\
\dot{\tilde{q}}
\end{bmatrix} =
\begin{bmatrix}
J^e (\dot{p} + \dot{p}^v) \\
0
\end{bmatrix} +
\begin{pmatrix}
I - \begin{bmatrix} J_e \\ J_v W \end{bmatrix}^T \left( \begin{bmatrix} J_e \\ J_v W \end{bmatrix} \right)^T \begin{bmatrix} J_e \\ J_v W \end{bmatrix}
\end{pmatrix}^{-1}
\begin{bmatrix}
J_e \\
J_v W
\end{bmatrix} \begin{bmatrix}
\dot{\theta} \\
0
\end{bmatrix}
\]  

(4.10)

\[
\begin{bmatrix}
\dot{q} \\
\dot{\tilde{q}}
\end{bmatrix} =
\begin{bmatrix}
J^e (\dot{p} + \dot{p}^v) \\
0
\end{bmatrix} + (I - (J^e W (J^e W)^{-1})(J^e W)) \begin{bmatrix}
\dot{\theta} \\
0
\end{bmatrix}
\]  

(4.11)

The virtual extension used in this thesis allows the weighting matrix to be selected relative to the end-effector. By using the previously described virtual extension the weighting matrix relative to the task space is defined by:

\[
W = \text{diag} \begin{bmatrix} w_z & w_y & w_x & w_{oz} & w_{oy} & w_{oz} \end{bmatrix}
\]  

(4.12)

By using a fixed weighting matrix any secondary task will affect the end-effector right from the start. Conversely the weighting matrix can be manipulated and altered throughout a task, depending on varying conditions. One great advantage is allowing $W$ to be dependent on the performance of the secondary task. By using an initial $W$ close to zero one may prioritize the secondary task more if, and only if, needed. This will allow the end-effector to stay at task for longer than with a static weighting matrix.

### 4.2.2 Extension of Mason’s Task Formulation

A formulation based on Mason’s (1979) selection matrix, which is expanded to include the weighting matrix, is proposed. His way of defining force controlled task with a "1" and position controlled task with a "0" in the diagonal of a selection matrix is adapted to also include secondary task priority described in the previous section. If an axis is given full priority to the secondary task the corresponding value in the diagonal of the selection matrix is set to "−1". An axis with a lesser priority is assigned a negative decimal according to the priority, i.g. 50% controllability is assigned a "−0.5".

The weighting matrix $W$ can now be found from the extended selection matrix, $S_E$, from the following equation:

\[
W = \frac{1}{2} (\sqrt{S_E^2} - S_E)
\]  

(4.13)

where the original selection matrix for force control, $S$, can be found as follows

\[
S = S_E - W = S_E - \frac{1}{2} (\sqrt{S_E^2} - S_E)
\]  

(4.14)
Example

Let's again consider the Peg-in-Hole example. For the task to be achievable, the four natural constraints in Table 4.2 must be fulfilled. The axis, \( v_z \) and \( \omega_z \), on the other hand have no natural constraints, and can thus be manipulated freely by the secondary task. Manipulating \( \omega_z \) will not affect the task at all, since it is assumed that there is no requirements on the orientation about the z-axis that must be fulfilled. Translation along z-axis will delay the task or prematurely complete the task, but let's assume that it can be accepted to avoid an obstacle. Based on this a controllability level for the z-axis is chosen to 70%. In addition to the natural and artificial constraints, secondary task priority of the axis is selected. For the Peg-in-Hole case will the extended selection matrix, \( S_E \), thus be as shown in (4.15)

\[
S_E = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & -0.7 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 \\
\end{bmatrix}
\]  

(4.15)

From this the weighting matrix can be calculated by equation 4.14. This gives:

\[
W = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.7 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]  

(4.16)
And the selection matrix correspondingly:

\[
S = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]  (4.17)

By using this task description scheme and redundancy resolution may a redundant robot be programmed as if it was a six-axis robot. This lowers the threshold for using redundant robots significantly and can to an extent degree allow SMEs to implement redundant robots in their production plant.

### 4.2.3 Selection Matrix Transformation

In many cases it may be desired to select which coordinate system that the extended selection matrix should apply to. In the Peg-in-Hole example the selection matrix must be active in the same coordinate system as the task. But in a Pick-and-Place case, where it may be more important that the tool is constrained against its environment, it may be more suitable to use the selection matrix in base coordinates. If so the extended selection matrix must be transposed to match the desired space.
Chapter 5

Obstacle Avoidance as a Secondary Task

As it was shown in chapter 4, $\vartheta$ in equation 5.1 may be selected based on any given subtask. A method for determining $\vartheta$ with regards to obstacle avoidance will now be presented.

$$\begin{bmatrix}
\dot{q} \\
\dot{\tilde{q}}
\end{bmatrix} = J^\dagger_c (\dot{p} + \tilde{p}^V) + \mathbf{N}_W \begin{bmatrix} \dot{\vartheta} \\
0
\end{bmatrix}$$ (5.1)

The obstacle avoidance strategy presented in this section is based on Nemec and Zlajpah’s (2002) strategy. However, the proposed solution is different.

The obstacle avoidance strategy is based on first detecting an obstacle then identifying the critical point on the robot arm. The critical point being the point closest to the detected obstacle. Then, a velocity component is placed at the critical point, moving the robot away from the obstacle.

Let’s now consider a planar arm with 5 degrees of freedom. The arm in Figure 5.1 is currently moving its end-effector according to the velocity vector $\dot{p}$. At some point an obstacle is detected too close to the robot’s arm. The robot’s control system then determines the critical point. The control system subsequently place a velocity vector, $\dot{p}_a$ at the critical point. The velocity vector is calculated from a unit vector $\hat{u}$ perpendicular to the arm at the critical point which is multiplied to a gain, denoted $k_c$. Solving the least square with the pseudo inverse of the Jacobian at the critical point, $J_C$, related to the inverse kinematics gives a desired joint velocity $\dot{q}_a$.

$$\vartheta = \begin{bmatrix} \dot{q}_{avoid} \\
0
\end{bmatrix} = \begin{bmatrix} J_C^\dagger \dot{p}_{avoid} \\
0
\end{bmatrix}$$ (5.2)

The resulting velocity vector with length $a$ is extended with $n-a$ zeros, where $a$ is the number of dimensions in the robot’s joint space at the critical point. The resulting $n$ element vector is $\vartheta$, which can be applied to equation 5.1. This process is repeated for every iteration where
CHAPTER 5. OBSTACLE AVOIDANCE AS A SECONDARY TASK

Figure 5.1: A planar robot detects and avoids an obstacle while performing a task.

the robot arm is in the critical neighborhood of an obstacle. Once no obstacles are detected, \( \vartheta = 0 \).

One of the benefits of using this method combined with the previously described weighted extended null space formulation is the robot's ability to augment the primary task and prioritize the secondary task if an obstacle occur. This functionality does not only improve the robot's ability to avoid an obstacle but can improve the programmability of the robot significantly. If properly equipped with sensors, the operator may jog the redundant robot, as if it was a 6 DOF robot. Thus leaving the arm configuration problems to the algorithm executed in the background. The secondary task priority must be set as described in the previous chapter.

The flow of obstacle avoidance as a secondary task using the redundancy resolution described in chapter 4 can then be illustrated as in Figure 5.2.
Some Considerations

The gain, $k_c$ must be chosen and optimized based on several parameters. Amongst them one should consider at least the following:

- The velocity of the primary task being performed in both task space and joint space.
- Limitations in the dynamics of the robot.
- The proximity of the obstacle.
In cases where multiple obstacles are present simultaneously their corresponding extended \( \dot{q}_a \) may be summarized (Equation (5.3)).

\[
\dot{\theta} = \sum_{i=1}^{b} \begin{bmatrix} \dot{q}_{avoid} \\ 0 \end{bmatrix}_i
\]  

(5.3)
Chapter 6

Applications to NACHI MR20 robot

This section is divided into four parts that will describe the sensor system that was developed for this thesis. The first part will describe the kinematic models, avoidance vectors and other data that is a preliminary requisite for the system to function. Secondly is all the hardware that were used presented. This include properties and preliminary testing of the sensors. Thirdly is the software structure and code distribution on the different platforms describes. This includes the most important algorithms executed on the computer. Lastly will the installation of the system on the NACHI MR20 be described. A wiring diagram and photos are presented to document the final result.

![Figure 6.1: Principal structure of the system.](image)

The structure of the system considered in this chapter can be seen in Figure 6.1. It is built up from the hardware described in section 1.4. A NACHI MR20 7-axis robot is controlled by
a NACHI AX20 robot controller. The signals in the AX20 is manipulated via an Olimex USB interface which in turn is controlled from a PC. The PC treats sensor data gathered from multiple proximity sensors mounted on the robot arm. The data is analyzed and an appropriate joint velocity vector $\dot{q}$ is sent to the Olimex. All components of the system will be thoroughly explained in the remainder of the chapter.

### 6.1 Kinematics of the MR20

Models for performing both forward- and inverse kinematics was a prerequisite for starting work on implementation of an obstacle avoidance system. The next sections in this chapter will explore how the NACHI MR20 7-axis robot can be defined with and without a virtual extension.

#### 6.1.1 Model With All Joints

The values used in the DH-matrices is based on the datasheet for the MR20 (Appendix E). The model for the MR20 with all joints is therefore as follows.

<table>
<thead>
<tr>
<th>Link</th>
<th>$\theta$</th>
<th>$d$</th>
<th>$r$</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>0</td>
<td>0,3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>$\theta_1$</td>
<td>0,2</td>
<td>0,15</td>
<td>$-\frac{\pi}{2}$</td>
</tr>
<tr>
<td>2</td>
<td>$\theta_2$</td>
<td>0</td>
<td>0</td>
<td>$\frac{\pi}{2}$</td>
</tr>
<tr>
<td>3</td>
<td>$\theta_7$</td>
<td>0,6</td>
<td>0</td>
<td>$\frac{-\pi}{2}$</td>
</tr>
<tr>
<td>4</td>
<td>$\theta_3 - \frac{\pi}{2}$</td>
<td>0</td>
<td>0,1</td>
<td>$\frac{-\pi}{2}$</td>
</tr>
<tr>
<td>5</td>
<td>$\theta_4$</td>
<td>0,5</td>
<td>0</td>
<td>$\frac{\pi}{2}$</td>
</tr>
<tr>
<td>6</td>
<td>$\theta_5$</td>
<td>0</td>
<td>0</td>
<td>$\frac{-\pi}{2}$</td>
</tr>
<tr>
<td>7</td>
<td>$\theta_6$</td>
<td>0,175</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6.1: Denavit-Hartenberg Matrix for the MR20 with all joints.

The joints angular value $\theta_3$ in joint $q_3$ is offset by $-\frac{\pi}{2}$. This was done to get the desired L-shaped home pose. Similar offsets are added in joints later in this thesis to gain desired home poses for the given kinematics.
Figure 6.2: Wireframe model of the NACHI MR20.
6.1.2 Model With Virtual Extension

The virtual extension was, as described in section 4.2, modeled to correspond to the \( X, Y, Z, \omega_x, \omega_y, \omega_z \) dimensions in task space. Its three prismatic and three rotational joints together with the kinematics for the MR 20 give the DH-Matrix shown in Figure 6.2.

<table>
<thead>
<tr>
<th>Link</th>
<th>( \theta )</th>
<th>( d )</th>
<th>( r )</th>
<th>( \alpha )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>0</td>
<td>0.3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>( \theta_1 )</td>
<td>0.2</td>
<td>0.15</td>
<td>(-\frac{\pi}{2})</td>
</tr>
<tr>
<td>2</td>
<td>( \theta_2 )</td>
<td>0</td>
<td>0</td>
<td>( \frac{\pi}{2} )</td>
</tr>
<tr>
<td>3</td>
<td>( \theta_3 )</td>
<td>0</td>
<td>0.6</td>
<td>(-\frac{\pi}{2})</td>
</tr>
<tr>
<td>4</td>
<td>( \theta_4 )</td>
<td>(-\frac{\pi}{2})</td>
<td>0</td>
<td>(-\frac{\pi}{2})</td>
</tr>
<tr>
<td>5</td>
<td>( \theta_5 )</td>
<td>0.5</td>
<td>0</td>
<td>( \frac{\pi}{2} )</td>
</tr>
<tr>
<td>6</td>
<td>( \theta_6 )</td>
<td>0</td>
<td>0</td>
<td>(-\frac{\pi}{2})</td>
</tr>
<tr>
<td>7</td>
<td>( \theta_7 )</td>
<td>0.175</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( v_1 )</td>
<td>0</td>
<td>( d_{v1} )</td>
<td>( \pi )</td>
<td>( \frac{\pi}{2} )</td>
</tr>
<tr>
<td>( v_2 )</td>
<td>0</td>
<td>( d_{v2} )</td>
<td>(-\frac{\pi}{2})</td>
<td>( \frac{\pi}{2} )</td>
</tr>
<tr>
<td>( v_3 )</td>
<td>0</td>
<td>( d_{v3} )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( v_4 )</td>
<td>( \theta_{v4} + \pi )</td>
<td>0</td>
<td>0</td>
<td>( \frac{\pi}{2} )</td>
</tr>
<tr>
<td>( v_5 )</td>
<td>( \theta_{v5} + \frac{\pi}{2} )</td>
<td>0</td>
<td>0</td>
<td>( \frac{\pi}{2} )</td>
</tr>
<tr>
<td>( v_6 )</td>
<td>( \theta_{v6} )</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6.2: Denavit-Hartenberg Matrix for the MR20 with virtual extension.

6.2 Sensor Placement and Kinematics

Designing the sensor layout was based on work done by Wada (2012). The strategy being to place a sensor with its detection direction orthogonal to the last joint’s rotational axis and as close to the next joint as possible. The shoulder, upper-arm and elbow is covered by sensors related to each joint and their motion. The sensors detection direction is perpendicular to the fix-point on the robot arm. The under-arm is covered by sensors mounted with the detection direction parallel to the link between joint \( q_4 \) and \( q_5 \). This was done to cover a greater part of the underarm. The complete sensor placement plan can be seen in Figure 6.3.

The proximity sensors labels are decided based on the strategy shown in table 6.3
6.2. SENSOR PLACEMENT AND KINEMATICS

Figure 6.3: Sensor placements.

Figure 6.4: Sensor placements.

6.2.1 Determining Avoidance Vectors

There are two elements that need to be determined when selecting an avoidance vector; its direction, $\hat{u}_a$, and gain, $k_c$. The directions are determined based on the placement of
Table 6.3: Proximity sensor labeling strategy.

<table>
<thead>
<tr>
<th>First and second</th>
<th>Third</th>
<th>Forth</th>
</tr>
</thead>
<tbody>
<tr>
<td>US = Ultrasonic</td>
<td>L=Left</td>
<td>0=Higher</td>
</tr>
<tr>
<td></td>
<td>R=Right</td>
<td>1=Lower</td>
</tr>
<tr>
<td></td>
<td>F=Front</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B=Back</td>
<td></td>
</tr>
<tr>
<td>IR=Infrared</td>
<td>0 = Outer Circle</td>
<td>0...1 = Clockwise counter</td>
</tr>
<tr>
<td></td>
<td>1 = Inner Circle</td>
<td></td>
</tr>
</tbody>
</table>

a given sensor relative to the orientation of the robot’s transform at the given point. As shown in the example in Figure 6.5, if a sensor’s detection direction is along the y-axis, \( \mathbf{u}_v = [0 -1 0 0 0]^T \). The gains used in this thesis are small and are only meant to verify the method, rather than optimize it. Section 5 mentions a few concerns that should be regarded when setting the gain.

\[
k_c = \begin{cases} 
0, & \text{if } \delta_s > \delta_{\text{max}} \\
\frac{(\delta_{\text{max}} - \delta_s)^2}{K_d} + (k_{\text{cmin}}), & \text{if } \delta_{\text{min}} < \delta_s < \delta_{\text{max}} \\
k_{\text{cmax}}, & \text{if } \delta_{\text{min}} > \delta_s 
\end{cases}
\] (6.1)

Figure 6.5: A sensor’s detection area and its corresponding avoidance vector direction.

The sensors on the upper arm also detect the distance from the obstacle to the robot. The gain profile thus being as shown in figure 6.6. This is used to increase the gain up to a maximum as the obstacle is getting closer. The gain was calculated by equation 6.1 with the parameters shown in table 6.4. The maximal gain \( k_{\text{cmax}} \) for each sensor is given in section 6.2.2.
6.2. SENSOR PLACEMENT AND KINEMATICS

Figure 6.6: The relation between the gain and the proximity of the obstacle.

Where $\delta_{\text{max}}$ is the furthest away an obstacle can activate the avoidance algorithm. $\delta_{\text{min}}$ is the distance at which $k_c$ should reach its max. $k_{c_{\text{max}}}$ and $k_{c_{\text{min}}}$ are the max and min values of $k_c$ within the detection distance $\delta_{\text{max}}$. $K_d$ is a constant determined based on the required rise-over-run between $\delta_{\text{min}}$ and $\delta_{\text{max}}$, which gives $K_d = \frac{\delta_{\text{max}} - \delta_{\text{min}}}{k_{c_{\text{max}}} - k_{c_{\text{min}}}}$. $K_d$ and $k_c$ is given in the following section for each sensor. The constant parameters are shown in table 6.4,

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta_{\text{max}}$</td>
<td>200mm</td>
</tr>
<tr>
<td>$\delta_{\text{min}}$</td>
<td>50mm</td>
</tr>
<tr>
<td>$k_{c_{\text{max}}}$</td>
<td>See sec. 6.2.2</td>
</tr>
<tr>
<td>$k_{c_{\text{min}}}$</td>
<td>0.0005</td>
</tr>
<tr>
<td>$K_d$</td>
<td>See sec. 6.2.2</td>
</tr>
</tbody>
</table>

Table 6.4: Parameters used to determine $k_c$ for the upper-arm sensors.

6.2.2 Sensor Kinematics and Avoidance Vectors

Based on the sensor placements described in section 6.2, the following kinematic models for the sensor placements, $\hat{u}_a$ vectors and gains can be set.

An observation was that the velocity vectors in the second task required a higher gain depending on how close to the base they were placed. The shoulder sensors were given $k_{c_{\text{max}}} = 0.01$, the elbow sensors $k_{c_{\text{max}}} = 0.0015$ and the underarm sensors $k_c = 0.0008$. From this it can be interpreted that $\vartheta$ is dependent of the norm of $\dot{\mathbf{q}}_a$. The sensors that were placed closest to the base only produced a $\dot{\mathbf{q}}_a$ with the two first elements other than zero. Their magnitude had to be proportionally larger, compared to the velocities created by the sensors on the elbow or underarm to produce any significant self motion.
US-L1 and US-R1

\[
\hat{u}_{US-L1}^{avoid} = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \hat{u}_{US-R1}^{avoid} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}
\]

\[k_{cmax}^{US-L1} = k_{cmax}^{US-R1} = 0.01, \quad K_d = 16\]

<table>
<thead>
<tr>
<th>Link</th>
<th>(\theta)</th>
<th>d</th>
<th>r</th>
<th>(\alpha)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>0</td>
<td>0.3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>(\theta_1)</td>
<td>0.15</td>
<td>0.2</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6.5: Wireframe model of the kinematics for sensors US-L1 and US-R1 with avoidance unit vectors, gain and the corresponding DH-matrix.

US-B1 and US-F1

\[
\hat{u}_{US-F1}^{avoid} = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \hat{u}_{US-B1}^{avoid} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}
\]

\[k_{cmax}^{US-F1} = k_{cmax}^{US-B1} = 0.01, \quad K_d = 16\]

<table>
<thead>
<tr>
<th>Link</th>
<th>(\theta)</th>
<th>d</th>
<th>r</th>
<th>(\alpha)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>0</td>
<td>0.3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>(\theta_1)</td>
<td>0.15</td>
<td>0.2</td>
<td>(-\frac{\pi}{2})</td>
</tr>
<tr>
<td>2</td>
<td>(\theta_2)</td>
<td>0</td>
<td>0.27</td>
<td>(\frac{\pi}{2})</td>
</tr>
</tbody>
</table>

Table 6.6: Wireframe model of the kinematics for sensors US-B1 and US-F1 with avoidance unit vectors, gain and the corresponding DH-matrix.
6.2. SENSOR PLACEMENT AND KINEMATICS

US-L0 and US-R0

\[
\hat{u}_{US-L0}^{avoid} = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \hat{u}_{US-R0}^{avoid} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}
\]

\[
k_{US-L0}^{cmax} = k_{US-R0}^{cmax} = 0.0015, K_d = 150
\]

<table>
<thead>
<tr>
<th>Link</th>
<th>(\theta)</th>
<th>(d)</th>
<th>(r)</th>
<th>(\alpha)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>0</td>
<td>0,3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>(\theta_1)</td>
<td>0,15</td>
<td>0,2</td>
<td>(-\frac{\pi}{2})</td>
</tr>
<tr>
<td>2</td>
<td>(\theta_2)</td>
<td>0</td>
<td>0</td>
<td>(\frac{\pi}{2})</td>
</tr>
<tr>
<td>3</td>
<td>(\theta_7)</td>
<td>0</td>
<td>0,6</td>
<td>(-\frac{\pi}{2})</td>
</tr>
</tbody>
</table>

Table 6.7: Wireframe model of the kinematics for sensors US-L0 and US-R0 with avoidance unit vectors, gain and the corresponding DH-matrix.

US-F0 and US-B0

\[
\hat{u}_{US-F0}^{avoid} = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \hat{u}_{US-B0}^{avoid} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}
\]

\[
k_{US-F0}^{cmax} = k_{US-R0}^{cmax} = 0.0015, K_d = 150
\]

<table>
<thead>
<tr>
<th>Link</th>
<th>(\theta)</th>
<th>(d)</th>
<th>(r)</th>
<th>(\alpha)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>0</td>
<td>0,3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>(\theta_1)</td>
<td>0,15</td>
<td>0,2</td>
<td>(-\frac{\pi}{2})</td>
</tr>
<tr>
<td>2</td>
<td>(\theta_2)</td>
<td>0</td>
<td>0</td>
<td>(\frac{\pi}{2})</td>
</tr>
<tr>
<td>3</td>
<td>(\theta_7)</td>
<td>0</td>
<td>0,6</td>
<td>(-\frac{\pi}{2})</td>
</tr>
<tr>
<td>4</td>
<td>(\theta_3 - \frac{\pi}{2})</td>
<td>0</td>
<td>0,1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6.8: Wireframe model of the kinematics for sensors US-F0 and US-B0 with avoidance unit vectors, gain and the corresponding DH-matrix.
IR-00 to IR-07 and IR-10 to IR-17

Table 6.9: Wire frame model of the kinematics for the IR sensors with avoidance vectors.

<table>
<thead>
<tr>
<th>$k_c$</th>
<th>Link</th>
<th>$\theta$</th>
<th>$d$</th>
<th>$r$</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Base</td>
<td>0</td>
<td>0.3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>$\theta_1$</td>
<td>1</td>
<td>0.2</td>
<td>0.15</td>
<td>$-\frac{\pi}{2}$</td>
</tr>
<tr>
<td>2</td>
<td>$\theta_2$</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>$\frac{\pi}{2}$</td>
</tr>
<tr>
<td>3</td>
<td>$\theta_3$</td>
<td>3</td>
<td>$\theta_3 - \frac{\pi}{2}$</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td>4</td>
<td>$\theta_4$</td>
<td>4</td>
<td>0.27 + $d_{IR}$</td>
<td>0</td>
<td>$\frac{\pi}{2}$</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6.10: DH-matrix and $k_c$ values for the IR sensors mounted on the under arm.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$-1$</td>
<td>$-\frac{1}{\sqrt{2}}$</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}$</td>
<td>$\frac{1}{\sqrt{2}}$</td>
<td>0</td>
<td>$-1$</td>
<td>$-\frac{1}{\sqrt{2}}$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6.11: Unit avoidance vectors for the IR sensors.
6.3 Sensory System Hardware Structure

The structure of the sensor system is shown in Figure 6.7. All the sensors were connected to the sbRIO which in turn were connected to a PC. The different elements of this structure will be described in the following sections.

![Diagram of hardware structure](image)

Figure 6.7: The hardware structure of the sensor system.

6.3.1 Proximity Sensors

Two different types of proximity sensors were used in this thesis, ultrasonic and infrared. The ultrasonic sensors are digitally controlled and are therefore more robust towards noise from the robots servos. They have a wide detection field which makes them suitable to cover a greater area, the downside of this is that they are not suited to be placed close together. The sensor layout design required sensors to be placed close together, so an infrared sensor was chosen as the sensor that were to be placed on the sensor mount ring on the lower-arm. They use a line of infrared light rather than a sheet, compared to the ultrasonic sensors. As an introductory research project these sensors were regarded as a good choice.

Ultrasonic sensors

The ultrasonic sensors used in this project were the PING))) sensors from Parallax which can be seen in Figure 6.8 (see Appendix E.3). The sensors provide accurate, non-contact distance measurements and are an affordable alternative to industrial standard sensors.

They function by pulling the trigger line HIGH for 5 \( \mu sec \), which results in a short burst of ultrasonic sound. After a holding time of 18 \( \mu sec \) the input on the controller is enabled and listens for a HIGH which is emitted by the sensor when it "hears" the echo of the ultrasonic burst. The time it takes until the input pin is HIGH, denoted \( \Delta t \), is used to calculate the
distance from the sensor to the object. If the echo is not heard within a set time limit, it
goes to timeout and the process restarts. The distance is then calculated with the below
equation.

\[ \Delta d = \frac{\Lambda v_s}{2} \]  

(6.2)

Where \( \Lambda \) is the pulse width in \( \mu \text{sec} \) and \( v_s \) is the speed of sound. This thesis will use 0.00034342
\text{m/\mu \text{sec}} as the speed of sound at 20°C.

The trigger and echo line can be controlled by one I/O pin on the sbRIO, but when the sbRIO
reconfigures a pin’s direction, it causes the controller to stall for about 18\( \mu \text{sec} \). It was there-
fore necessary to design a circuitry to enable the use of two separate pins for input and out-
put, even though the PING))) only requires one I/O pin. The schematics for the circuitry can
be seen in Figure 6.9. The output voltage of the sbRIO’s DIO is limited to 3,3V and the in-
put range for the PING))) is 3,3V-5V. The circuitry was therefor also designed to step up the
voltage to 5V from an external power supply.

With a total of 8 ultrasonic sensors, the pin requirements were 8 DI pins and 8 DO pins.
The full circuitry for the ultrasonic sensors then becomes as shown in Figure 6.10.

![Figure 6.10: Schematics for the circuitry developed for the US sensors.](image)

Preliminary testing of the ultrasonic sensors showed very good measurements at long distance (Figure 6.11) with very low error, and good measurements at short range where the maximum error was 18 mm (Figure 6.12).

![Figure 6.11: The relation between measured distance and actual distance on long range for the ultrasonic sensors.](image)
Figure 6.12: The relation between measured distance and actual distance on short range for the ultrasonic sensors.

**Infrared Sensors**

The sensors mounted on the ring on the under-arm were placed too close together to use ultrasonic sensors. Infrared sensors were used instead. The Sharp GP2Y0A02 (Figure 6.13) functions well even if they are placed close together alongside each other. The sensor’s signal pin outputs a variable voltage between 0,5V and 3,3V, where the voltage corresponds to the amount of reflected infrared light measured by the light detector.

![Infrared Sensor](image)

Figure 6.13: The Sharp infrared proximity sensor.

The correlation between the distance $D$ and the voltage $V$ is given by

$$\frac{1}{D + k} = mV + b \quad (6.3)$$

Where $k$, $m$ and $b$ are unknown parameters found in this case by using the IR calibration VI in LabVIEW. Experiments gave the data shown in table 6.12. Although the cabling used internally in the MR20 (see section 6.5.3) is not shielded the noise from the servos was primarily stationary. The solution was therefore to calibrate the infrared sensors with the servos active.

The given data samples then give the following parameters:
Due to noise in the analog readings from the infrared sensors a filter was implemented. The filter is a point by point filter that removes the highest and lowest 15% of a data series with a given length. It then averages the remaining values. The sample length used in this thesis was 20. A comparison of the received data with and without the filter can be seen in Figure 6.14.

Preliminary testing of the infrared sensors showed that the measurements were satisfactory in the critical range (100 – 400mm) from Figure 6.15. The measurements above 500mm and below 100mm were not satisfactory, but also not needed for the implementation on the MR20, and therefor considered irrelevant. The results of the short test can be seen in Figure 6.16.

Table 6.12: Voltage readings at given distances.

<table>
<thead>
<tr>
<th>Sample #</th>
<th>Distance [m]</th>
<th>Voltage [V]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.15</td>
<td>1.689</td>
</tr>
<tr>
<td>2</td>
<td>0.3</td>
<td>1.1167</td>
</tr>
<tr>
<td>3</td>
<td>0.7</td>
<td>0.4794</td>
</tr>
</tbody>
</table>

Table 6.13: Voltage to distance calculation parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>1.2205</td>
</tr>
<tr>
<td>m</td>
<td>0.473684</td>
</tr>
<tr>
<td>b</td>
<td>0.244444</td>
</tr>
</tbody>
</table>

Figure 6.14: Distance measured by the IR sensors without and with the filter with sample lengths 10 and 20.
The 16 infrared sensors were mounted on the upper-arm on a circular mount with the sensors detection direction alongside the arm (see Figure 6.4). Each sensor was assigned an Analog Input on the sbRIO.
6.3.2 Controller

Choosing the controller was based on two main criterion, it had to be capable of controlling all sensors, and it was desired to be programmable from LabVIEW. The preferable choice was then a Compact RIO system from National Instruments. This is a module based micro controller with easy connectivity which makes it great for a laboratory environment. Due to availability a Single Board RIO NI 9632 (sbRIO) was used instead (see Figure 6.17 and Appendix E). It offers a 400MHz processor, 128MB DRAM and 2M gate reconfigurable I/O FPGA. The sbRIO requires both the FPGA level and \( \mu \) controller level to be programmed separately.

![Figure 6.17: The NI 9632 Single Board RIO.](image)

6.3.3 Computer

As described in section 1.4 a standard Desktop PC was used in this thesis. The PC communicated with the sbRIO and the Olimex via the local network.

6.3.4 Olimex

The Olimex micro controller is the computer's interface with the NACHI AX20 Robot Controller (Figure 6.18). It is connected between the servo-board and main-board in the AX20, and has thus the ability to manipulate the data the main-board sends to, and receives from, the servos.
6.4 System Software Structure

The code is distributed over three levels plus the Olimex card for modification of the signals from the NACHI AX20 robot controller. Both the FPGA and the µ controller is part of the sbRIO. The software distribution has been illustrated in Figure 6.19. The remainder of this section will describe the code executed on each of the levels.

6.4.1 FPGA

The purpose of the code running at FPGA level was to operate, read and calculate the measured distances of all sensors. The distances were measured and calculated as described in section 6.3.1 and 6.3.1. Figure 6.20 shows the LabVIEW code developed to control one ultrasonic sensor. Each of the ultrasonic sensors were coded to execute in separate while loops.
Figure 6.20: LabVIEW code for the control of one ultrasonic sensor running on the FPGA.

Figure 6.21 shows the LabVIEW code developed to control one infrared sensor. There was one loop for each infrared sensor.

Figure 6.21: LabVIEW code for the control of one infrared sensor running on the FPGA.

The input to the FPGA is the 8 DI pins connected to the ultrasonic sensors, the 16 AI pins connected to the infrared sensors and a cluster containing the three parameters (k, m and b) for calculating the distance of the infrared sensors. The parameters are represented in double precision. The output from the FPGA was 8 DO pins connected to the ultrasonic sensors, 8 fixed-point values containing the distances measured by the ultrasonic sensors; 8 boolean values containing the timeout status of the ultrasonic sensors, and 16 fixed-point values containing the distances measured by the infrared sensors. The inputs and outputs of the FPGA level have been illustrated in Figure 6.22.

Figure 6.22: The inputs and outputs of the FPGA.
6.4.2 Micro Controller

The controllers main task was to process the distance data it receives from the FPGA. Distances received from ultrasonic sensors are set to an upper limit value to reduce the effective detection range. If an ultrasonic sensor has to timeout, the last registered distance will be used. All distances are arranged in a $1 \times 8$ dimensional double precision array. The distances measured by the ultrasonic sensors are also controlled against a set detection range and a $1 \times 8$ boolean array is built up based on the measured distance status. The boolean is set to TRUE if the measured value is lower than the detection range value.

The distance values obtained from the IR sensors are first off all filtered using the filter described in section 6.3.1 and modeled as shown in Figure 6.23. After filtering the distances are they controlled. If the value is lower than 0.30 cm, it means that an obstacle is detected and an avoidance maneuver should be put into effect. The information is organized in a $2 \times 8$ boolean array.

Figure 6.23: LabVIEW code for the filter used ultrasonic sensor data.

The output from the controller was four arrays. One double precision $1 \times 8$ array with the distances measured by the ultrasonic sensors, one $1 \times 8$ boolean array with the ultrasonic sensors timeout status, and one $2 \times 8$ double precision array with the distances measured by the IR sensors, and a $2 \times 8$ boolean array with the statuses of the IR sensors. The inputs and outputs of the micro controller have been illustrated in Figure 6.24.

Figure 6.24: The inputs and outputs of the controller code.
6.4.3 PC

The majority of the code was executed from the PC. Figure 6.26 shows the algorithm that governs the main process of this code. At first the code checks if the end-effector is at task plus any Cartesian delta position, if not the path will be corrected as described later in this section. Secondly, the code will investigate if there are any detected obstacles. If so the Avoidance algorithm will be activated and the suggested $\Delta p$ will be controlled to see if it is too great. If it is, it will be limited by the Position velocity limiter. If there are no detected obstacles the code will proceed and see if the end-effector is at task, $\Delta p = 0$. If not, the Task Reconstruction algorithm will be executed, and the end-effector will take one step closer to its task. If the end-effector is at task, and there are no detected obstacles the code will check if the robot has its default arm configuration. If not, it will take one step towards default arm configuration by self-motion. After completing either one of the aforementioned tasks, the suggested joint velocity will be compared to the maximum joint velocity. If it is found to be to great all joint velocities will be adjusted according to the joint velocity limiter. The implemented system uses a modified version of the presented Extended Selection Matrix where the selection matrix only regards the priority of secondary task axes.

The inputs and outputs of the computer can be summarized through Figure 6.25.

![Figure 6.25: The inputs and outputs of the computer code.](image)

The next sections will in other words describe the purpose and workings of the five blocks in figure 6.26 (excluding the "no action" block after "Obstacle detected?").
`q̇ = q̇\textsuperscript{P}
\Delta \dot{p} = 0`
Path Correction

A Pseudo inverse based least-norm solution to the inverse kinematics problem is proposed in a path correcting algorithm. The path correction was done as illustrated in Figure 6.27.

For each iteration the current position was corrected according to the reference data from the AX20, \( q_{AX20\text{ref}} \), and any desired delta position \( \Delta p \). The reference position, \( p_{\text{ref}} \) was calculated as shown in equation 6.4, using the Jacobian for the end-effector, \( J_e \).

\[
p_{\text{ref}} = J_e q_{AX20\text{ref}} + \Delta p
\]  

(6.4)

The error is then the Cartesian difference between \( p_{\text{ref}} \) and the current position, \( p_{\text{current}} \).

\[
p_{\text{current}} = J_e q_{\text{current}}
\]  

(6.5)
\[ \dot{e} = p_{ref} - p_{current} \quad (6.6) \]

A correctional joint velocity, \( \dot{q}_{corr} \), is calculated based on the pseudo inverse of the Jacobian for the end-effector (Equation (6.8)). The velocity is added to the current joint angles and the current position and error position is recalculated. The algorithm is terminated when the norm of \( \dot{e} \) is less than \( 10^{-12} \).

\[ \dot{q}_{corr} = J_{e}^{\dagger} \dot{e} \quad (6.7) \]

\[ ||J_{e}^{\dagger} q_{AX20ref} - J_{e}^{\dagger} (\dot{q}_{corr} + q_{current})|| = ||\dot{e}|| > 10^{-12} \quad (6.8) \]

**Avoidance Algorithm**

![Flow Chart of the avoidance process.](image-url)

Figure 6.28: Flow Chart of the avoidance process.
The avoidance has been thoroughly explained in section 5. The implementation of the code is also described through a flowchart in Figure 6.28.

**Position velocity Limiter**

Because of limitations in the hardware capabilities regarding the speed of the servos it was necessary to include a limitation to $\Delta \dot{p}$. It follows the algorithm shown in figure 6.29.

![Diagram](image)

**Figure 6.29:** The algorithm governing speed limitation.

$\Delta P(k)$ denotes the suggested delta position at iteration $k$. $\Delta P(k)$ is given by

$$\Delta P(k) = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}. \quad (6.9)$$

Similarly, $\Delta O(k)$ is the delta orientation suggested by the extended avoidance algorithm at iteration $k$. $\Delta O(k)$ is depending on the end-effector, since a change in orientation will keep
TCP stationary. The calculation of $\Delta O(k)$ is based on the arc lengths of the rotation, relative to TCP. It is therefore given by (6.10), where $T_i$ denotes the end-effector’s length in direction $i$, which is defined in the General Settings and Control panel, see section 7.2.

$$\Delta O(k) = \Delta AT_x + \Delta BT_y + \Delta CT_z.$$  \hspace{1cm} (6.10)

$\Delta M(k)$ denotes the longest movement, in either position or orientation. Thereby simply the greater of $\Delta P(k)$ and $\Delta O(k)$.

$$\Delta M(k) = \max(\Delta P(k), \Delta O(k))$$ \hspace{1cm} (6.11)

Then, if $\Delta M(k)$ is greater than $\Delta M_{\text{max}}$, $\Delta \dot{p}$ is compensated by limiting the contribution at iteration $k$. $\Delta \dot{p}$ at time $k$ then becomes as shown in (6.13). This will ensure that the direction of the movement is kept intact. In other words, the direction of the trajectory will not be affected if the speed is limited.

$\Delta M_{\text{max}}$ is given by:

$$\Delta M_{\text{max}} = \frac{v_{\text{max}}}{f_0}$$ \hspace{1cm} (6.12)

where $v_{\text{max}}$ denotes the maximal allowed speed which is set in the Advanced panel, see section 7.2, and $f_0$ is the frequency of the control system.

$$\Delta \dot{p} = \Delta \dot{p} \frac{\Delta M_{\text{max}}}{\Delta M(k)}$$ \hspace{1cm} (6.13)

**Joint Level Velocity Limiter**

The joint velocity control is the last check before $\dot{q}$ are sent to the Olimex. The purpose of the algorithm is to make sure no joints are rotating too fast. The limit used in this thesis is low compared to the capabilities of the robot, but this thesis will regard low speeds only. The algorithm simply works by reducing all joint velocities proportionately if one is too high. If none of the joints velocities are too high the velocities will not be affected by the algorithm. The algorithm is shown in Figure 6.30

If a joint, $q_i$, has to great velocity, $q_i > q_{\text{max}}$, then are all the joints velocities limited by (6.14).

$$\dot{q} = \dot{q} \frac{q_{\text{max}}}{q_{a} f_0}$$ \hspace{1cm} (6.14)

Where $q_{\text{max}}$ is the maximum joint velocity in $\text{rad/sec}$, $f_0$ is the frequency of the control system, and $q_{a} = \max(q_i)|_{i=0}$, where $n$ is the dimensions of joint space.
Task Reconstruction

When the avoidance algorithm acts with a weighting matrix $\neq \mathbf{0}$, it adds an offset distance to the original position value sent from the AX20 to the MR20. When the obstacle that caused an avoidance action disappears, it may be desirable to return to the original position. A trajectory generator was implemented to ensure a linear return. The trajectory generator is activated when the sensor no longer detects any obstacles. Since the return trajectory can be interrupted by a new avoidance maneuver at any time the last value of the interpolated return trajectory is stored and added to the offset caused by the avoidance algorithm. Figure 6.31 shows the algorithm that governs the task reconstruction mode.

The speed that the end-effector is returned at is accomplished by calculating the number of interpolation points that will be used in the trajectory generator, based on the Travel speed that is set in the Advanced panel in the GUI. The number of interpolation points, $N_I$, is calculated by first calculating the length from the current position to the original position. This distance corresponds to the accumulated delta position. The longest movement $\Delta M$ is calculated as described in the speed limiter in section 6.4.3. Once $\Delta M$ is achieved, it is multiplied with the inverse of the allowed length per iteration. Allowed length per iteration is calculated by dividing the return speed, $v_r$, by the operating frequency, $f_0$, for the system. $N_I$ is thus given by

$$N_I = \Delta M \left( \frac{v_r}{f_0} \right)^{-1}$$  \hspace{1cm} (6.15)
Default arm reconfiguration

When the primary task is fully reconstructed and there are no active secondary tasks, the arm may reconfigure to its default configuration. The algorithm that governs the reconfiguration is shown in Figure 6.32. After checking whether the reconfiguration should be activated or not, the number of steps ($N_A$) required to return to default, is calculated based on the joint with the greatest offset from default and the maximum step length. The step length is then calculated for each joint based on the number of steps and their current offset angle. These steps are repeated until the conditions activate reconfiguration are not met.

Where $q^N$ in Figure 6.32 is the total angular offset created by secondary tasks, $q_{max}^N$ is the joint with the greatest angular offset, $v_c$ is the maximum allowed joint velocity in rad/sec and $f_0$ is the operating frequency.
6.4. SYSTEM SYSTEM SOFTWARE STRUCTURE

Servo Encoder Values to Radians

Some parts of this subsection are either direct excerpts or based on previous work by the candidate in a specialization project (Sanderud and Reme, 2011). Since the AX20 communicates with its servos in encoder values there was a need for scaling all data to and from the Olimex. This is because the Olimex interfaces with the servo board, which operates with servo encoder values. It was therefore necessary to determine the relationship between the encoder values and the joint angles in radians.

This was done by programming the NACHI AX20 controller to set all the robots joints to its "home position", which is illustrated in figure 6.33.

Reading of the encoder values were performed in both the home position and a position where all joints were programmed to move an additional +10°. The results can be seen in table 6.14.

Table 6.15 shows the scaling factors for each joint, found using the delta of the long values and corresponding angles for each joint, from table 6.14.

![Flowchart](image-url)
As the direction of rotation for some of the joints from the robot system did not coincide with the directions in the kinematic model developed in LabVIEW, the signs for some of the scaling factors had to be altered.

### 6.4.4 Olimex

The Olimex intercepts the signals and sends out new manipulated signals depending on the interpretations provided by code running on the computer. The software running on the Olimex is written by Johannes Schrimpf. The principal workings of the software running on the Olimex is shown in Figure 6.34.
6.4.5 Modeling in LabVIEW

The LabVIEW 2011 Robotics Module offer a variety of VI’s (functions in LabVIEW) that are helpful when working with robotic arms. Amongst the ones utilized in this project are VI’s for forward kinematics and calculation of the Jacobian which is based on Prof. Peter Corke’s robotics toolkit for MATLAB. The Robotics Module also provide a number of VI’s for manipulating and converting between datatypes such as vectors, homogeneous transformations and rotation matrices, all of which are helpful when representing 3-dimensional pose.

6.5 Installation of Sensor System

This section will describe the practical aspects of the installation of the sensor system. This will include the necessary wiring, connectors, and circuitry that was made for this thesis. The complete cabling is shown in figure 6.35.

Figure 6.34: The working principals of the Olimex

Figure 6.35: Cabling Diagram of the Sensor System
### Cable Wire Type

<table>
<thead>
<tr>
<th>Cable</th>
<th>Wire Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>cIR0-cIR7</td>
<td>$6 \times 0.5^2$, Shielded</td>
</tr>
<tr>
<td>cUS0-cUS7</td>
<td>$4 \times 0.5^2$, Shielded</td>
</tr>
<tr>
<td>CNR010</td>
<td>MR20 Internal Cable</td>
</tr>
<tr>
<td>cIRM</td>
<td>$20 \times 0.75^2$, shielded</td>
</tr>
<tr>
<td>RB1-RB2</td>
<td>50pin Ribbon Cable</td>
</tr>
<tr>
<td>Power cables</td>
<td>$2 \times 2.5^2$</td>
</tr>
</tbody>
</table>

Table 6.16: Cables used in the sensor system.

### 6.5.1 Connector Block C1

The proposed hardware demanded some additional connector blocks and circuitry to be made. First of all was the pure connector block, C1, which was used inside the junction box in the robot’s wrist as can be seen in Figure 6.36. Male headers were used as connectors for the plugs CN61-CN64 mounted on the robot’s internal cable (CNR010). A spring terminal block functioned as connector for the cables cIR0-cIR7. The ground and 5V are connected in two 4-pos screw terminals.

![Figure 6.36: The C1 connector block mounted inside the robots wrist and the C1 connector block (inserted image).](image)

### 6.5.2 Proto-Board

The proto-board functions as a terminal block for the sbRIO and holds the necessary circuitry for the ultrasonic sensors (Figure 6.37). The circuitry was made after the schematics in Figure 6.10. The proto-board was powered from a 5V external power supply. Two $2 \times 25$ male headers functioned as connectors for the two 50pin ribbon cables connected to the P5 and J7 plugs on the sbRIO.
6.5. INSTALLATION OF SENSOR SYSTEM

6.5.3 Infrared Sensors

The IR sensors were, as previously mentioned, mounted on a circular plate encircling the robot arm, just after $q_4$ (see fig 6.3). The sensor mount ring was designed as a removable collar as shown in Figure 6.38, and the sensors were mounted as described in section 6.2. The sensors were wired in pairs, according to their clock position, and into the junction box in the robots wrist. There all the sensor cables were terminated in connector block C1 connected to the CNR010 cable in the robot. The BJ1 connector panel at the back of the robot’s base, was then connected to the proto-board functioning as a connector to the sbRIO.
6.5.4 Ultrasonic Sensors

The ultrasonic sensors were mounted and wired individually. Since the ultrasonic sensor's connector was a 3-pos male header the cables were equipped with 3-pos crimp-on female header connectors in both ends for easy handling. Each of the sensors were connected to a 3-pos male header at the proto-board.

The result of all sensors mounted on the robot is shown in the Figure 6.39.

![Figure 6.39: The front and side of the NACHI MR20 with ultrasonic and infrared sensors mounted (the collar have been colored blue for visibility).](image)

Figure 6.39: The front and side of the NACHI MR20 with ultrasonic and infrared sensors mounted (the collar have been colored blue for visibility).
Chapter 7

User guide

As a part of this thesis was a program developed in LabVIEW as described in chapter 6. The startup procedures and user guide for the Graphical User Interface that was developed will now be presented.

7.1 Startup Procedures

Before one can turn on the NACHI AX20 controller, the following has to be checked:

1. The Olimex interface is correctly connected between the controller- and servo- board.

2. The Olimex has power.

3. The Olimex is set up to intercept signals as well as receiving and broadcasting. This is done by connecting to it at IP: 192.168.212.30 via SSH. Subsequently logging in as "root" with the password "olimex". Then type the command for starting the process: "nasta".

At this point, the NACHI AX20 can be powered on.

Before starting the Obstacle Avoidance for NACHI MR20-VI, make sure the sbRIO is connected and powered up and that the Controller code v0.02-VI is running. The Master FPGA IO v1.00-VI must be compiled onto the FPGA before startup.
7.2 LabVIEW User Interface

This section will describe the purpose and functionality of all indicators and controls that can be found in the Graphical User Interface (GUI) developed for this thesis. The GUI was organized in a total of 5 tabs. Figure 7.1-7.5 show screenshots from the 5 tabs and are followed by a description of their respective indicators and controls.

![Figure 7.1: The General settings and Control-Panel in the GUI.](image)

**General settings and Control Panel**

All controls regarding activation of the code, selection matrix, and tool is done from the *General settings and Control* panel.

**CONTROLS**

**Activate** To start sending delta values to the Olimex, this button must be activated. It is important that the current *Computer -> Olimex* values are all zero when activating the program. And correspondingly, that the same values are zero when deactivated.

**Reset** Resets the *Computer->Olimex* values to zeros.

**Detect** Enables the detection mode in the system. If enabled the robot will attempt an evasive maneuver if an obstacle is detected.
**Ext. Avoidance system** Allows the operator to select whether the constraints and selection matrix should be defined in base coordinates, tool space, or a user defined task space.

**Selection Matrix** Allows the operator to set the desired selection matrix.

**Task Space** A manually selected task space can be selected for the selection matrix and constraints to function in.

**Extended Avoidance Constraints** Set constraints on the \( \Delta p \) resulting from the extended avoidance.

**Tool** Set the tool definition. The tool definition can be acquired from the AX20 Teach Pendant.

**STATUS LEDs**

**Activate** Indicates if the calculated delta position is being sent to the Olimex.

**Detection Active** The detection system is active and the robot will avoid an obstacle if detected.

**Stand-By** The system is currently in stand-by

**Obstacle Detected** An obstacle is detected and the robot is performing an evasive maneuver.

**Task Reconstruction** The system is currently reconstructing the primary task that was deflected from due to the secondary task.

**Arm Reconfiguration** The system is reconfiguring the arm back to the default configuration without interrupting the primary task.

**Detected - US** Indicates which of the ultrasonic sensors currently detecting an obstacle.

**Detected - IR** Indicates which of the infrared sensors currently detecting an obstacle.

**Delta Position**

Displays the delta position and orientation, \( \Delta p \), that is added to the reference position \( \Delta p_{ref} \) from the AX20

**Graphs**

**Computer -> Olimex** Allows monitoring of the values sent from the code to the Olimex.

**Delta Position** Allows monitoring of the Delta position created as a result of the extended avoidance.
Figure 7.2: The Sensor settings -> General and Monitoring-Panel in the GUI.

**Sensor settings -> General and Monitoring Panel**

This panel let the operator monitor sensor activity and enable or disable the individual sensors.

**CONTROLS**

**Enable IR Sensors** Selects which of the 16 infrared sensors that should be active.

**Enable US Sensors** Selects which of the 8 ultrasonic sensors that should be active.

**GRAPHS**

**Ultra Sonic Sensors** Allows monitoring of all distances measured by the ultrasonic sensors.

**Infrared Sensors 0x** Allows monitoring of all distances measured by the infrared sensors in the inner ring.

**Infrared Sensors 1x** Allows monitoring of all distances measured by the infrared sensors in the outer ring.
Sensor settings -> Advanced Panel

**CONTROLS**

- **$k_{c_{\text{max}}} \text{ US}** Set the maximum gain for the ultrasonic sensors according to the description in section 6.2.2.

- **US Direction** Set the axis the avoidance vector is active along for the ultrasonic sensors according to the description in section 6.2.2. (0=x, 1=y, ... 5=C)

- **Type** Set whether the avoidance vector is the same direction or the opposite direction of the axis it’s parallel to.

- **$\delta_{\text{max}} \text{ US}** Set $\delta_{\text{max}}$ described in section 6.2.2.

- **$p_a \text{IR}** Set the avoidance vectors for the infrared sensors. (The vectors are transposed)

**INDICATORS**

- **US Timeout** Indicates whether the individual ultrasonic sensors times out.
Gripper Control Panel

In this panel the operator may control the gripper action manually. The gripper is activated in two ways, either when GRIP or MOVE POS is pressed. In GRIP-mode does the Current bar indicate the current that will energize the gripper, a positive current will open the gripper, and a negative will close it. A greater current will allow the gripper to grip with greater force. Maximum Velocity sets the velocity that the gripper's fingers move. In MOVE POS-mode is the gripper operated by setting the desired position the gripper should be at by adjusting Position [mm], and then pressing MOVE POS. Velocity sett the velocity at which the gripper's fingers are moving, and Max Current [A] is defining the maximum allowed current. If at some point an error occurs, causing the gripper to be non-responsive when pressing GRIP, press QUIT ERROR to clear the error. Press FAST STOP to perform an emergency stop of the gripper.
7.2. LABVIEW USER INTERFACE

Figure 7.5: The Advanced-Panel in the GUI.

Advanced Panel

Max Speed Move  Is the maximum velocity in task space used in the position velocity limiter-algorithm.

Travel Speed [mm/s]  is the velocity in task space that the tool is returned to its task in the Task Reconstruction-algorithm.

Joint Speed [rad/sec]  Is the joint velocity which the joints are allowed to have in the Arm Reconfiguration-algorithm.

Joint speed MAX [rad/s]  Is the maximum joint velocity used in the joint level velocity limiter-algorithm.

Record data to file  will record to file the following data to file:

- Computer -> Olimex
- Servo -> Olimex
- Main -> Olimex
- Delta Position
- Reference Position
- Current Position
- $q^N$
- State
- Distances measured by the ultrasonic sensors
- Distances measured by the infrared sensors

**Olimex-card IP address** Set the IP-address for the Olimex-card.
Chapter 8

Evaluation of the System

This section will discuss and evaluate the sensor system. It will look into what effect detecting an obstacle with the different sensors will have on the robot. Three sensor groups were experimentally tested, shoulder sensors, elbow sensors and underarm sensors.

In the experiment was the end-effector placed a few centimeters above a table with a grid. The table’s surface was orthogonal to $Z_0$ and the table’s edges were parallel to the $X_0$ and $Y_0$ axes. In both experiments a dynamic obstacle was approached from negative $y$ in positive $y$ direction towards the robot. After a few seconds the obstacle was removed. The experiment to test the functionality of the extended avoidance calls for an appropriate selection matrix and tolerances. Since the table is located and defined by base coordinates is most appropriate to define the selection matrix and tolerances to the end-effector’s Cartesian offset in base coordinates. The experiment therefore uses a selection matrix which only allows an offset in $\pm y$.

The results are shown through a series of photos, and four graphs. The four graphs display the following:

I. **State**: The state of which the control systems were running (See section 6.4.3). The states are numbered as follows:
   1. Stand-by
   2. Obstacle detected: Avoidance
   3. Task Reconstruction
   4. Arm-Default Reconfiguration

II. **Computer to Olimex**: The added joint angle values sent to the Olimex from the computer. In other terms, the joint angle deflection form the AX20’s reference angles.

III. $\Delta p_{CP}$: The delta position to the Critical Point (CP) in base coordinates.

IV. $\Delta p$: The delta position added to the end-effector position in task space.
CHAPTER 8. EVALUATION OF THE SYSTEM

8.1 Shoulder sensor

8.1.1 No extended Avoidance

This section presents the experimental data gathered during experimental testing of the shoulder sensor, with No Extended Null Space. Figure 8.1-8.5 show photos, the state of the system, deflection of joint angles, delta position for the critical point and deflection of the end-effector respectively. The ultrasonic sensor US-L1 were approach with an obstacle, and the max gain were set to $k_{c_{\text{max}}} = 0.01$.

Figure 8.1: Frames from the experiment, No Program, No Extended null space.

Figure 8.2: The state at which the code runs over time from the experiment, No Program, No Extended null space.

Figure 8.3: Joint angles added to the AX20’s values from the experiment, No Program, No Extended null space.
From Figure 8.4 can it be seen that the the robot is only able to move $\approx 4\text{cm}$ away from the obstacle and it takes about $10\text{sec}$ to get there. The avoidance is only causing a deflection along the y-axis, which is natural due to the kinematic properties of the robot. This means that if an obstacle is detected by the robot’s shoulder will it have a very limited possibility to avoid it. Not only is the deflection very small, but the theoretical deflection is also very limited since the system only have one joint to manipulate to maneuver away from the obstacle. The rate at which the robot moves to avoid the obstacle is fairly low and linear.
8.1.2 With extended Avoidance

This section presents the experimental data gathered during experimental testing of the shoulder sensor, With Extended Null Space. Figure 8.6-8.10 show photos, the state of the system, deflection of joint angles, delta position for the critical point and deflection of the end-effector respectively. The state changes has been marked with a dashed line in all the presented graphs.

![Figure 8.6: Frames from the experiment, No Program, No Extended null space.](image)

![Figure 8.7: The state at which the code runs over time from the experiment, No Program, No Extended null space.](image)

![Figure 8.8: Joint angles added to the AX20's values from the experiment, No Program, No Extended null space.](image)
8.1. SHOULDER SENSOR

Figure 8.9: Cartesian Delta position at the critical point caused by the evasive maneuver from the experiment, *Shoulder Sensor, With Extended null space*.

Figure 8.10: Cartesian Delta position caused by virtual extension from the experiment, *No Program, No Extended null space*.

In this case were the extended avoidance activated so the robot was allowed to move its end-effector 10 cm along the y-axis as it can be seen in Figure 8.10. This cause the robot's evasive maneuver to go somewhat faster until the end-effector reach its maximum deflection. However, in Figure 8.9 can it be seen that the total deflection at the critical point is not of significant difference from the experiment without extended avoidance.
8.2 Elbow Sensor

8.2.1 No extended Avoidance

This section presents the experimental data gathered during experimental testing of the shoulder sensor, with No Extended Null Space. Figure 8.11-8.15 show photos, the state of the system, deflection of joint angles, delta position for the critical point and deflection of the end-effector respectively. The ultrasonic sensor US-L0 were approach with an obstacle, and the max gain were set to $k_{c_{\text{max}}} = 0.0015$.

![Figure 8.11: Frames from the experiment, No Program, No Extended null space.](image)

![Figure 8.12: The state at which the code runs over time from the experiment, No Program, No Extended null space.](image)

![Figure 8.13: Joint angles added to the AX20’s values from the experiment, No Program, No Extended null space.](image)
8.2. ELBOW SENSOR

Figure 8.14: Cartesian Delta position at the critical point caused by the evasive maneuver from the experiment, *Elbow Sensor, No Extended null space*.

In Figure 8.14 can it be seen that the elbow deflects ≈ 15 cm from its original position when an obstacle is approaching. It is assumed that it would have continued if the obstacle not were removed. The time it took to get there can also be seen to be ≈ 10 sec. This makes a much faster rate than for the shoulder sensor, even though the gain is \( \frac{1}{15} \) of shoulder sensor gain. The major component of the total deflection is seen along the y-axis, this is as expected because the sensors detection direction was along the robot’s y-axis. As the avoidance advances is the velocity component placed at the critical point rotated relative to base coordinates, the outcome can be seen as a deflection in along both z- and x-axis. It is assumed that this tendency would progress if the obstacle were not removed.
8.2.2 With extended Avoidance

This section presents the experimental data gathered during experimental testing of the shoulder sensor, With Extended Null Space. Figure 8.16-8.20 show photos, the state of the system, deflection of joint angles, delta position for the critical point and deflection of the end-effector respectively. The state changes has been marked with a dashed line in all the presented graphs.

Figure 8.16: Frames from the experiment, No Program, With Extended null space.

Figure 8.17: The state at which the code runs over time from the experiment, No Program, With Extended null space.

Figure 8.18: Joint angles added to the AX20's values from the experiment, No Program, With Extended null space.
8.2. ELBOW SENSOR

Figure 8.19: Cartesian Delta position at the critical point caused by the evasive maneuver from the experiment, Elbow Sensor, With Extended null space.

Figure 8.20: Cartesian Delta position caused by virtual extension from the experiment, No Program, With Extended null space.

This experiment were conducted with full priority for the secondary task along the y-axis. As it appears from Figure 8.20 is the end-effector deflected a total of \( \approx 14 \text{ cm} \), and the effect of that can be seen in Figure 8.19 where the elbow is deflected \( \approx 18 \text{ cm} \) as opposed to \( \approx 15 \text{ cm} \) without extended avoidance. Furthermore is the \( \approx 18 \text{ cm} \) of deflection reached in \( \approx 7 \text{ sec} \). It can also be seen that the evasive maneuver is faster in the beginning and proceeds at a lower rate at \( \approx 15 \text{ cm} \). Again can it also be seen that the major deflection is along the y-axis, and later also along x- and z-axis, much similar to the previous experiment without extended avoidance.
8.3 Underarm sensor

8.3.1 No extended Avoidance

This section presents the experimental data gathered during experimental testing of the shoulder sensor, with No Extended Null Space. Figure 8.21-8.25 show photos, the state of the system, deflection of joint angles, delta position for the critical point and deflection of the end-effector respectively. The infrared sensor IR-06 were approach with an obstacle, and the max gain were set to $k_{c_{\text{max}}} = 0.0008$.

![Figure 8.21: Frames from the experiment, No Program, No Extended null space.](image1)

![Figure 8.22: The state at which the code runs over time from the experiment, No Program, No Extended null space.](image2)

![Figure 8.23: Joint angles added to the AX20’s values from the experiment, No Program, No Extended null space.](image3)
This experiment show that if an obstacle is detected alongside the underarm is the robots ability to avoid it fairly limited. As seen in Figure 8.24 is the robot only moving the critical point \( \approx 7 \text{cm} \), in the 9 seconds the obstacle is within the critical neighborhood. This behavior can be expected as the detected obstacle is close the the end-effector, an evasive maneuver is barely possible if the end-effector are to be kept stationary.

The same effect as previously described where the deflection is mainly represented along the y-axis with an increasing component along z-axis can also be said to be present in this experiment.
8.3.2 With extended Avoidance

This section presents the experimental data gathered during experimental testing of the shoulder sensor, With Extended Null Space. Figure 8.26-8.30 show photos, the state of the system, deflection of joint angles, delta position for the critical point and deflection of the end-effector respectively. The state changes has been marked with a dashed line in all the presented graphs.

Figure 8.26: Frames from the experiment, No Program, No Extended null space.

Figure 8.27: The state at which the code runs over time from the experiment, No Program, No Extended null space.

Figure 8.28: Joint angles added to the AX20's values from the experiment, No Program, No Extended null space.
In this experiment were the extended avoidance enabled along the y-axis. When an obstacle were detected along the underarm would the end-effectors delta position (Figure 8.30) contribute to the evasive maneuver. In Figure 8.29 is it apparent that the avoidance is happening at a much higher rate as long as the end-effector is moving. In Figure 8.30 can it be seen that the end-effector reach its constraint (0, 1 m) after \( \approx 5 \text{sec} \), at the same moment is the avoidance rate significantly reduced in Figure 8.29. After 5 sec is the deflection in the critical point much similar to the one without extended avoidance as seen if Figure 8.24. The total deflection in this experiment was thus \( \approx 17 \text{cm} \).
Chapter 9

Experiments and Results

Several experiments were conducted to verify that the system was functioning as anticipated. The experimental plan and results from the experiments will be described in this chapter. All robot programs that was used in each of the cases were programmed on the AX20’s teach pendant as if the MR20 was a six-axis robot. One robot program was made for each case, the three experiments in each case were in other words controlled by the same robot program on the AX20.

9.1 Experimental Plan

The following chapter will describe the experiments that were conducted to verify and test the developed redundancy resolution and task formulation. One experiment with no program running on the robot and an approaching obstacle were carried out first. Then a series of three applicational cases were experimentally tested. The cases were a Peg-in-Hole, a Grinding and an Operation in constrained space situation. These will all be described in the following sections along with a presentation of their respective results. Some of the results for the No Program case have been repurposed from the shoulder sensor experiments in chapter 8.

A total of three experiments were carried out for each case. They are listed below along with their respective solutions for $\dot{q}$ ((9.1)-(9.3)). The first of the three experiments is the reference to how the task should be performed without the influence of the developed code or any obstacles. The third experiment was conducted with a simplified weighting matrix, represented as a selection matrix masking the axes not prioritized for the secondary task as shown in (9.3).

1. No collision avoidance.

\[
\dot{q} = \dot{q}_p + \dot{q}_N = J^T_\ell \dot{q} + 0 \quad (9.1)
\]
2. Collision avoidance without extended null space

\[ \dot{q} = \dot{q}_p + \dot{q}_\kappa = J^\dagger_c \dot{p} + \kappa_c \theta \]  \hspace{1cm} (9.2)

3. Collision avoidance with extended null space and appropriately set selection matrix and tolerances.

\[
\begin{bmatrix}
\dot{q} \\
\dot{q}^v
\end{bmatrix} = \begin{bmatrix}
\dot{q}_p + \dot{q}_\kappa \\
\dot{q}^v
\end{bmatrix} = J^\dagger_c (\dot{p} + S \dot{p}^v) + \kappa_v \begin{bmatrix}
\theta \\
0
\end{bmatrix}
\]  \hspace{1cm} (9.3)

The results from the experiments are shown as a series of photos and graphs displaying the data in the below list\(^1\). After each case the results will from the experiments be commented.

I. **State**: The state of which the control systems were running (See section 6.4.3). The states are numbered as follows:

1. Stand-by
2. Obstacle Detected: Avoidance
3. Task Reconstruction
4. Arm-Default Reconfiguration

II. **Computer to Olimex**: The added joint angle values sent to the Olimex from the computer. In other terms, the joint angle deflection form the AX20’s reference angles.

III. **p\textsubscript{ref}**: The reference position to the end-effector in Cartesian space.

IV. **\Delta p** : The delta position added to the end-effector position in task space.

Each graph has been marked with a dashed line at the moments where the control system changes state, according to point I. in the above list.

---

\(^1\)Some blank pages have been inserted to ensure the results from each experiment is presented on a double page.


9.2 No Program

9.2.1 Experimental Setup

Some of the results for this experiment have been repurposed from chapter 8. The results are presented again in this chapter because the case was investigated on other grounds than in chapter 8.

First of all a case where the end-effector was stationary was investigated. The experiment is as aforementioned much similar to the case study used to investigate sensor capabilities of the system in chapter 8. The setup is for order's sake repeated in this section. The end-effector was placed a few centimeters above a table (Figure 9.1) with a grid as shown in Figure 9.2. The table's surface was orthogonal to $Z_0$ and the table's edges were parallel to the $X_0$ and $Y_0$ axes. Only two experiments were carried out for the case since the static state of the case does not require a reference run. In both experiments a dynamic obstacle was approached from negative $y$ in positive $y$ direction towards the robot's elbow. After a few seconds the obstacle was removed. The experiment to test the functionality of the extended avoidance calls for an appropriate selection matrix and tolerances. Since the table is located and defined by base coordinates is most appropriate to define the selection matrix and tolerances to the end-effector's Cartesian offset in base coordinates. The experiment therefore uses the selection matrix shown in (9.4) which only allows an offset in $\pm y$. The constraint on the deflection was set to $\Delta y = \pm 0,2m$.

Figure 9.1: The experimental setup of the static experiment.
Figure 9.2: The grid on the table used in the No program case experiment.

\[ S = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \]  

(9.4)
9.2.2 Experimental Results

Collision Avoidance, No Extended Null Space

This section presents the experimental data gathered during the *No Program* case, with *Collision Avoidance, No Extended Null Space*. Figure 9.3-9.7 show photos, the state of the system, deflection of joint angles, reference position and deflection of the end-effector respectively. The state changes has been marked with a dashed line in all the presented graphs.

![Frames from the experiment, No Program, No Extended null space.](image)

**Figure 9.3:** Frames from the experiment, *No Program, No Extended null space*.

![The state at which the code runs over time from the experiment, No Program, No Extended null space.](image)

**Figure 9.4:** The state at which the code runs over time from the experiment, *No Program, No Extended null space*. 
Figure 9.5: Joint angles added to the AX20's values from the experiment, *No Program, No Extended null space*.

Figure 9.6: Reference position $p_{ref}$ from the experiment, *No Program, No Extended null space*.

Figure 9.7: Cartesian Delta position caused by virtual extension from the experiment, *No Program, No Extended null space*. 
Collision Avoidance with Extended Null Space

This section presents the experimental data gathered during the No Program case, with Collision Avoidance with Extended Null Space. Figure 9.8-9.12 show photos, the state of the system, deflection of joint angles, reference position and deflection of the end-effector respectively. The state changes has been marked with a dashed line in all the presented graphs.

Figure 9.8: Frames from the experiment, No Program, With Extended null space.

Figure 9.9: The state at which the code runs over time from the experiment, No Program, With Extended null space.
Figure 9.10: Joint angles added to the AX20’s values from the experiment, *No Program, With Extended null space*.

Figure 9.11: Reference position $p_{ref}$ from the experiment, *No Program, With Extended null space*.

Figure 9.12: Cartesian Delta position caused by virtual extension from the experiment, *No Program, With Extended null space*. 
9.2.3 Comments to the Results

From the above results is it clear that the MR20 has the ability to reconfigure its arm while keeping the end-effector stationary when an obstacle approaches. Based on the sensor information a velocity component is placed at the critical point. The first experiment was done without extended null space, and it is apparent from both photos (Figure 9.3) and the delta position (Figure 9.7) from the reference position (Figure 9.6) that the end-effector is stationary while the robot uses self-motion to avoid the approaching obstacle. When the obstacle disappears the robot reconfigures its arm to the default arm configuration. In the second experiment the y-axis was enabled in the secondary task by using the selection matrix in equation 9.4. This results in, as is shown in Figure 9.12, that there is an added delta position in $y$. This delta position causes the system to return to stand-by via the Task Reconstruction algorithm before the arm is reconfigured to default. From Figure 9.12 and 9.10 is it apparent that the end-effector is completely returned to its reference position before the arm is reconfigured.
9.3 Peg-In-Hole

9.3.1 Experimental Setup

This classic experiment is great way to test the control systems ability to reconfigure and move precisely within the natural and artificial constraints. The peg (see Figure 9.13) is mounted in the robots gripper so that the peg’s tool constant, $p^T$, is according to Table 9.1. The tool constants, $p^T$, was found with the AX20’s tool definition software. The hole is fixed with the center of the entrance hole at $p^H$ given in Table 9.1. The setup was then as shown in Figure 9.14. A path was programmed using the AX20’s teach pendant where the peg was inserted and removed from the hole continuously. For each experiment the loop ran approx three times. The selection matrix used in the third experiment is based on the interpretation of the problem in Table 9.2, which resulted in (9.5).

$$p^T = \begin{bmatrix} -0.2206 \\ -0.5 \\ 0,1663 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \quad p^H = \begin{bmatrix} 1.204 \\ -0.412 \\ 0.718 \\ 0.269 \\ 1.267 \\ 2.214 \end{bmatrix}$$

Table 9.1: Peg end-effector position and hole position and orientation.

Figure 9.13: A peg-tool and a block with a hole used for the Peg-in-Hole experiment.
Figure 9.14: A peg-tool and a block with a hole used for the Peg-in-Hole experiment.

Table 9.2: Peg-in-hole.

<table>
<thead>
<tr>
<th>Natural Constraints</th>
<th>Artificial Constraints</th>
<th>Secondary task priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_x = 0$</td>
<td>$V_z = V_0z$</td>
<td>$X = 0$</td>
</tr>
<tr>
<td>$v_z = v_z(.)$</td>
<td>$v_y = 0$</td>
<td>$Y = 0$</td>
</tr>
<tr>
<td>$V_y = 0$</td>
<td>$\omega_z = 0$</td>
<td>$Z = 1$</td>
</tr>
<tr>
<td>$\Omega_x = 0$</td>
<td>$\omega_x = 0$</td>
<td></td>
</tr>
<tr>
<td>$\omega_y = 0$</td>
<td>$\omega_y = 0$</td>
<td></td>
</tr>
</tbody>
</table>

$$S = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$ (9.5)
9.3.2 Experimental Results

No Collision Avoidance

This section presents the experimental data gathered during the Peg-In-Hole case, with No Collision Avoidance. Figure 9.15-9.19 show photos, the state of the system, deflection of joint angles, reference position and deflection of the end-effector respectively. The state changes has been marked with a dashed line in all the presented graphs.

Figure 9.15: Frames from the experiment Peg-in-Hole, No Avoidance.

Figure 9.16: The state at which the code runs over time from the experiment, Peg-in-Hole, No Avoidance.

Figure 9.17: Joint angles added to the AX20’s values from the experiment, Peg-in-Hole, No Avoidance.
Figure 9.18: Reference position $p_{ref}$ from the experiment, Peg-in-Hole, No Avoidance.

Figure 9.19: Cartesian Delta position caused by virtual extension from the experiment, Peg-in-Hole, No Avoidance.
Collision Avoidance, No Extended Null Space

This section presents the experimental data gathered during the *Peg-In-Hole* case, with *Collision Avoidance, No Extended Null Space*. Figure 9.20-9.24 show photos, the state of the system, deflection of joint angles, reference position and deflection of the end-effector respectively. The state changes has been marked with a dashed line in all the presented graphs.

![Figure 9.20: Frames from the experiment, Peg-in-Hole, No Extended null space.](image)

![Figure 9.21: The state at which the code runs over time from the experiment, Peg-in-Hole, No Extended null space.](image)
Figure 9.22: Joint angles added to the AX20’s values from the experiment, Peg-in-Hole, No Extended null space.

Figure 9.23: Reference position $p_{ref}$ from the experiment, Peg-in-Hole, No Extended null space.

Figure 9.24: Cartesian Delta position caused by virtual extension from the experiment, Peg-in-Hole, No Extended null space.
Collision Avoidance with Extended Null Space

This section presents the experimental data gathered during the Peg-In-Hole case, with Collision Avoidance with Extended Null Space. Figure 9.25-9.29 show photos, the state of the system, deflection of joint angles, reference position and deflection of the end-effector respectively. The state changes have been marked with a dashed line in all the presented graphs.

![Fig 9.25: Frames from the experiment, Peg-in-Hole, With Extended null space.](image)

![Fig 9.26: The state at which the code runs over time from the experiment, Peg-in-Hole, With Extended null space.](image)
Figure 9.27: Joint angles added to the AX20’s values from the experiment, Peg-in-Hole, With Extended null space.

Figure 9.28: Reference position $p_{ref}$ from the experiment, Peg-in-Hole, With Extended null space.

Figure 9.29: Cartesian Delta position caused by virtual extension from the experiment, Peg-in-Hole, With Extended null space.
9.3.3 Comments to the Results

In this case the robot was performing a task while an obstacle appeared by its elbow. Again it is clear that the robot is able to reconfigure its arm so that it avoids the approaching obstacle. The first experiment, with no avoidance, shows that the control system is kept in standby (Figure 9.16). The second experiment demonstrates the system's ability to avoid an obstacle while keeping the end-effector at task. When the obstacle is no longer in the neighboring area, the arm is reconfigured (Figure 9.22), still without interrupting the ongoing task (Figure 9.24). In the third experiment the extended null space is enabled, and the selection matrix is set to allow a delta position in task space along, and rotation about, the $x$-axis. This causes the task, as it can be seen in Figure 9.29, that the tool is given a delta in $z$, which in turn causes the task to be slightly delayed. The possibility to rotate about $x$ was not particularly exploited, from Figure 9.29 can it be seen that the deflection is at its maximum only about -0.09 radians ($\approx -5^\circ$). After the obstacle is removed, the task again is reconstructed as shown in Figure 9.29, followed by arm reconfiguration (Figure 9.27).
9.4 Grinding

9.4.1 Experimental Setup

To demonstrate the ability to offset all rotational axes while keeping the tool position on task a grinding simulation was set up. Using a dome shaped grinding tool does not require position control in either of the rotational axes. These could therefore be prioritized by the secondary task as shown in Table 9.4. A workpiece with a three dimensional curvature was used shown in Figure 9.30. A simple path was programmed on the teach pendant along the curvature before the tool was lifted and returned. The program ran once for each experiment. The selection matrix required in the third experiment is shown in (9.6).

\[
\begin{bmatrix}
-0.2206 \\
-0.5 \\
0.1663 \\
0 \\
0 \\
0
\end{bmatrix} \quad \begin{bmatrix}
1.2 \\
0.3 \\
0.40 \\
0 \\
0 \\
0
\end{bmatrix}
\]

Table 9.3: Peg end-effector position and hole position and orientation.
### Table 9.4: Grinding.

#### Natural Constraints

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_x = V_{0x}$</td>
<td>$v_x = v_x(.)$</td>
</tr>
<tr>
<td>$V_y = V_{0y}$</td>
<td>$\omega_z = \omega_z(.)$</td>
</tr>
<tr>
<td>$v_z = V_{0z}$</td>
<td>$\Omega_y = 0$</td>
</tr>
</tbody>
</table>

#### Artificial Constraints

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_x = 0$</td>
<td>$V_z = V_{0z}$</td>
</tr>
<tr>
<td>$v_y = 0$</td>
<td>$\Omega_z = 0$</td>
</tr>
<tr>
<td>$\omega_x = 0$</td>
<td>$\omega_y = 0$</td>
</tr>
</tbody>
</table>

#### Secondary task priority

<table>
<thead>
<tr>
<th>Task</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X$</td>
<td>$\omega_x = 1$</td>
</tr>
<tr>
<td>$Y$</td>
<td>$\omega_y = 1$</td>
</tr>
<tr>
<td>$Z$</td>
<td>$\omega_z = 1$</td>
</tr>
</tbody>
</table>

$S = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}$

(9.6)
9.4.2 Experimental Results

No Collision Avoidance

This section presents the experimental data gathered during the Grinding case, with No Collision Avoidance. Figure 9.31-9.35 show photos, the state of the system, deflection of joint angles, reference position and deflection of the end-effector respectively. The state changes has been marked with a dashed line in all the presented graphs.

Figure 9.31: Frames from the experiment, Grinding, No Avoidance.

Figure 9.32: The state at which the code runs over time from the experiment, Grinding, No Avoidance.
9.4. GRINDING

Figure 9.33: Joint angles added to the AX20’s values from the experiment, *Grinding, No Avoidance.*

Figure 9.34: Reference position $p_{ref}$ from the experiment, *Grinding, No Avoidance.*

Figure 9.35: Cartesian Delta position caused by virtual extension from the experiment, *Grinding, No Avoidance.*
Collision Avoidance, No Extended Null Space

This section presents the experimental data gathered during the Grinding case, with Collision Avoidance, No Extended Null Space. Figure 9.36-9.40 show photos, the state of the system, deflection of joint angles, reference position and deflection of the end-effector respectively. The state changes has been marked with a dashed line in all the presented graphs.

Figure 9.36: Frames from the experiment, Grinding, No Extended null space.

Figure 9.37: The state at which the code runs over time from the experiment, Grinding, No Extended null space.
Figure 9.38: Joint angles added to the AX20’s values from the experiment, *Grinding, No Extended null space*.

Figure 9.39: Reference position $p_{ref}$ from the experiment, *Grinding, No Extended null space*.

Figure 9.40: Cartesian Delta position caused by virtual extension from the experiment, *Grinding, No Extended null space*.
Collision Avoidance with Extended Null Space

This section presents the experimental data gathered during the *Grinding* case, with *Collision Avoidance with Extended Null Space*. Figure 9.41-9.45 show photos, the state of the system, deflection of joint angles, reference position and deflection of the end-effector respectively. The state changes has been marked with a dashed line in all the presented graphs.

Figure 9.41: Frames from the experiment, *Grinding, With Extended null space*.

Figure 9.42: The state at which the code runs over time from the experiment, *Grinding, With Extended null space*. 
Figure 9.43: Joint angles added to the AX20’s values from the experiment, *Grinding, With Extended null space*.

Figure 9.44: Reference position $p_{ref}$ from the experiment, *Grinding, With Extended null space*.

Figure 9.45: Cartesian Delta position caused by virtual extension from the experiment, *Grinding, With Extended null space*. 
9.4.3 Comments to the Results

This simulated grinding task required the TCP to be at task at all times. The complex geometry of the edge to be grinded would give a good indication of the functionality of the developed system. The first experiment was again a reference and the system was at standby throughout the experiment. In the second experiment the avoidance algorithm was activated by an obstacle approaching at the robot’s elbow. The task is maintained throughout the experiment while the arm is reconfigured to avoid the obstacle (Figure 9.38). The arm also reconfigures properly after the obstacle is removed. In the last experiment rotation about all axes is enabled for the secondary task, but no translation. This results in a delta rotation, but as in the Peg-in-Hole case the deflections are rather small. The largest is about the x-axis and is at its greatest $\approx -0.125$ radians ($\approx -7^\circ$). The task is, on the other hand, intact throughout the experiment. After the obstacle is removed the Task Reconstruction and arm reconfiguration is properly executed (Figure 9.45 and 9.43).
9.5 Operation in Constrained Space

9.5.1 Experimental Setup

An operation in constrained space experiment was set up to demonstrate how the system can ease the programming of the robot in a complex environment. The robot’s elbow is obstructed by a wall when it tries to reach its destination. A traditional 6DOF robot would not reach the place position at all and a traditional control system of a 7DOF robot would be tedious to program. This experiment was set up to demonstrate that one can jog the robot, as if it was a 6DOF, and the robot automatically reconfigures based on sensor readings of the environment. After clearing the obstacle the intention is that the robot reconfigures its arm to its default pose. This experiment is meant to demonstrate the programmability through jogging on the teach pendant. The jogging was for this case constrained to motion along y-axis. The case also tests the capability to handle data from multiple sensors simultaneously, with opposing avoidance vectors. The selection matrix is therefore set as shown below for the third experiment.

| Natural Constraints | | | | | |
|---------------------|----------------------|
| $V_x = 0$           | $v_z = v_z(.)$       |
| $V_y = 0$           | $\omega_z = \omega_z(.)$ |
| $\Omega_x = 0$     | $\Omega_y = 0$      |

| Artificial Constraints | | | | | |
|------------------------|----------------------|
| $v_x = 0$              | $v_z = V_{0z}$       |
| $v_y = 0$              | $\Omega_z = 0$      |
| $\omega_x = 0$        | $\omega_y = 0$      |

| Secondary task priority | | | | | |
|-------------------------|----------------------|
| $X = 0$                 | $\omega_x = 0$      |
| $Y = 1$                 | $\omega_y = 0$      |
| $Z = 0$                 | $\omega_z = 0$      |

$$S = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}$$  \hspace{1cm} (9.7)
9.5.2 Experimental Results

No Collision Avoidance

This section presents the experimental data gathered during the Operation in Constrained Space case, with No Collision Avoidance. Figure 9.46-9.50 show photos, the state of the system, deflection of joint angles, reference position and deflection of the end-effector respectively. The state changes has been marked with a dashed line in all the presented graphs.

Figure 9.46: Frames from the experiment, Operation in Constrained Space, No Avoidance.

Figure 9.47: The state at which the code runs over time from the experiment, Operation in Constrained Space, No Avoidance.
Figure 9.48: Joint angles added to the AX20’s values from the experiment, *Operation in Constrained Space, No Avoidance*.

Figure 9.49: Reference position $p_{\text{ref}}$ from the experiment, *Operation in Constrained Space, No Avoidance*.

Figure 9.50: Cartesian Delta position caused by virtual extension from the experiment, *Operation in Constrained Space, No Avoidance*.
Collision Avoidance, No Extended Null Space

This section presents the experimental data gathered during the *Operation in Constrained Space* case, with *Collision Avoidance, No Extended Null Space*. Figure 9.51-9.55 show photos, the state of the system, deflection of joint angles, reference position and deflection of the end-effector respectively. The state changes have been marked with a dashed line in all the presented graphs.

Figure 9.51: Frames from the experiment, *Operation in Constrained Space, No Extended null space*. 

Figure 9.52: The state at which the code runs over time from the experiment, *Operation in Constrained Space, No Extended null space*. 
Figure 9.53: Joint angles added to the AX20’s values from the experiment, Operation in Constrained Space, No Extended null space.

Figure 9.54: Reference position $p_{ref}$ from the experiment, Operation in Constrained Space, No Extended null space.

Figure 9.55: Cartesian Delta position caused by virtual extension from the experiment, Operation in Constrained Space, No Extended null space.
Collision Avoidance with Extended Null Space

This section presents the experimental data gathered during the Operation in Constrained Space case, with Collision Avoidance with Extended Null Space. Figure 9.56-9.60 show photos, the state of the system, deflection of joint angles, reference position and deflection of the end-effector respectively. The state changes have been marked with a dashed line in all the presented graphs.

Figure 9.56: Frames from the experiment, Operation in Constrained Space, With Extended null space.

Figure 9.57: The state at which the code runs over time from the experiment, Operation in Constrained Space, With Extended null space.
Figure 9.58: Joint angles added to the AX20’s values from the experiment, *Operation in Constrained Space, With Extended null space*.

Figure 9.59: Reference position \( p_{ref} \) from the experiment, *Operation in Constrained Space, With Extended null space*.

Figure 9.60: Cartesian Delta position caused by virtual extension from the experiment, *Operation in Constrained Space, With Extended null space*. 
9.5.3 Comments to the Results

In the last case that was investigated the elbow was placed in between two walls. The color of these walls have been made green to make them more visible in the photos. The effect was that the sensors were detecting an obstacle during the greater part of the experiment. Since the gains of the avoidance vectors were determined based on the proximity of the obstacle should the robot attempt to keep its elbow centered in the middle of the two walls. In the first experiment that was carried out the walls were removed to avoid collision. The second experiment demonstrated the robot's ability to handle multiple obstacles by keeping the elbow rather stationary centered between the two walls as shown in Figure 9.51. The same can be said about the third experiment as seen in 9.56. In this case the task was somewhat delayed due to the offset along y. The advantage was hence reduced risk of collision.
Chapter 10

Discussion

This section will first discuss the results from chapter 9 where the system, as a programming support system was experimentally tested. Then, major results that were seen in chapter 8 - Evaluation of the sensor system - will be discussed. This thesis will not consider the response time of the system since the gains have not been optimized. Rather a few observations on the difference in the robot’s behavior when an obstacle is detected at the different critical points tested in chapter 8. First the section will go through the behavior of the sensor system followed by a discussion on the MR20’s ability to avoid detected obstacles.

10.1 Programming Support

The virtually extended null space formulation was experimentally tested in chapter 9. As a programming support system for redundant robots was it shown that it require very limited additional knowledge to operate compared to a 6-axis programming system. The only parameters that need to be determined before programming is the suggested extension of Mason's task formulation. A correct task description may be the difference between a collision and a successfully completed task. Although, if the operator have sufficient knowledge about the task it is easy to determine the required parameters in the task description.

10.2 Sensor System

The ultrasonic sensors showed good performance, the circuitry which enabled the use of separate Output and Input on the sbRIO gave a stable and reliable sensor setup. However, a challenge was that the ultrasonic sensors could not be placed close together. This could easily give some blind spots. Although, the sensors detected the obstacles, a sensor system that covered the entire robot arm would be preferred in an industrialized installation.
The data from the infrared sensors required filtering. A filter removing upper and lower 15% and averaging the remaining data were implemented to remove white noise.

10.3 Avoidability

The NACHI MR20 is a 7-axis robot with thus normally one degree of freedom for obstacle avoidance. By enabling the virtually extended null space formulation was the collision avoidance even better. However, the self-motion is limited to one possible motion, moving the elbow side to side. If a sensor is mounted so that its detection area is perpendicular to this motion will the robot not be able to generate any self motion. The ability to avoid now fully relies on deflecting the end-effector from its primary task. This is possible with the proposed system, but it relies on the selection of the weighting matrix.

Experiments also showed that by using the virtually extended null space formulation was not only the robot’s ability to avoid an obstacle improved, it also contributed to the rate at which the evasive maneuver was performed. This can be seen in the experiments from all three sensor groups.

However, the robots ability to avoid an obstacle if detected close to its base or end-effector is limited. In chapter 8 it was shown that the MR20 only managed to move the critical point \( \approx 6 \text{cm} \) away from the obstacle when detected close to the base. This was seen both with and without extended avoidance. The limitation in the shoulder can be regarded as a kinematic limitation as shown in Figure 10.1.

![Figure 10.1: The avoidable and non-avoidable region of the robot’s shoulder. (Top view of the first link and base only)](image)

If an obstacle approaches the shoulder from the side the robot is only able to avoid obstacles in the avoidable region. If an obstacle is detected in the non-avoidable region will no
motion prevent a collision. On the contrary, a situation where the shoulder is obstructed by an obstacle can be regarded as fairly rare in the industry. Such a situation is most likely the result of an accident. In any case, to avoid collisions should it be considered to implement a system that gives a warning if the robot detects an obstacle close to the shoulder in the non-avoidable region.

The same behavior was seen if an obstacle was detected alongside the underarm without extended avoidance. However, the MR20 was able to avoid the obstacle by using the virtually extended null space formulation. The underarms limited avoidance possibility can be explained as a combination of the kinematic and physical properties of the MR20. As seen in Figure 10.2 can the area around the underarm also be represented by an avoidable and a non-avoidable region. If an obstacle is detected in the avoidable region can the obstacle be avoided by pure self-notion, whereas if detected in the non avoidable region is self motion not sufficient. As opposed to the shoulder example the obstacle can be avoided by deflecting the end-effector.

![Figure 10.2: The avoidable and non-avoidable region of the robot's underarm.](image)

In many situations may the obstacle not be purely in the avoidable or non-avoidable region. If an obstacle is detected very close to the non-avoidable region are the margins for error very small. The avoidability space for a robot is therefor introduced. As it was seen in the experiments in chapter 8 was the robots ability to avoid an obstacle detected by the elbow good and the avoidability is thus high. Conversely was the robots ability to avoid obstacles by the end-effector poor, the avoidability can therefore be said to be low. By analyzing a robots kinematic properties and physical proportions can the avoidability space for the robot be defined. This information can be used to determine the robot's type of evasive maneuver; pure self-motion or a combination with a deflection of the end-effector. In other words can the avoidability determine the parameters in the weighting matrix. The avoidability space
for the MR20 with the arm configuration used in the sensor experiments can be illustrated as shown in Figure 10.3.

Figure 10.3: The avoidebility space for the MR20.

The figure show the avoidability space for the MR20, where high avoidability is indicated with green. As the avoidability is reduced closer the the base and end-effector the color is gradually turning red. The edge of the avoidability space can be regarded as the robot’s critical neighborhood. As aforementioned can the avoidability space be determined based on kinematic and physical properties. However, the robot’s current arm configuration would also influence the avoidability space. The arm configuration shown in the above figure have high avoidability around the elbow, but if the arm straightens out to reach further will the avoidability be reduced, also around the elbow. This is because the linear independency of the columns in the Jacobian for the end-effector is reduced as a singularity is approached. By interpreting the singular values from the SVD of the Jacobian can these instances be exposed and the information can be used in determine the avoidability.

This method make the obstacle avoidance method more intelligent in its decision making process regarding the evasive maneuver. Based on an obstacles point of entry the system can determine the most appropriate evasive maneuver. This would further enhance the pro-graming experience and similarity to well known programming of 6-axis robots.
Chapter 11

Conclusion and Further Work

The results presented in the previous chapter lay the basis for the conclusion in this thesis. After the conclusion will some ideas for future work be presented.

11.1 Conclusion

As programming of industrial robots is still a major bottleneck in manufacturing, more productive methods for programming highly in demand. Investigations on the use of redundant industrial robots in the industry reveals several advantages including highly increased flexibility and a significant reduction of space need. The flexibility can be attributed through obstacle avoidance, singularity avoidance and energy optimization.

This thesis suggested a task description scheme for redundant industrial robots based on Mason's task formulation for force controlled tasks. The redundancy resolution is based on a weighted virtually extended null space formulation. The proposed solution enables the use of virtually extended self motion of the robot to perform secondary tasks. A system where the secondary task is utilized for obstacle avoidance is proposed and implemented on a NACHI MR20. A complete sensor system was developed and implemented on the robot using infrared- and ultrasonic sensors to cover the greater part of the robotic arm.

Several algorithms were implemented in the system to ensure the functionality and robustness of the system. The algorithms include:

- Path Correction
- Task Reconstruction
- Default Arm Reconfiguration
- Position Velocity limiter
- Joint Velocity Limiter

Experiments were carried out on the NACHI MR20 with the developed obstacle avoidance- and sensor system. All programs used in experiments were programmed using the AX20
teach pendant as if the MR20 was a six-axis robot. The experiments to test the functionality and performance included one static case, and three industrial cases:

- No program / Stationary Tool
- Peg-in-Hole
- Grinding
- Operating in tight space.

The experiments proved the robots ability to use both self-motion and virtually extended self-motion to avoid obstacles. The systems ability to reconfigure the primary task after deflection caused by the secondary task and the ability to reconfigure the arm to a default configuration when both the task is reconstructed and no obstacles are present was also proven to be successful. All of these secondary tasks were enabled automatically after the "operator" had simply programmed the redundant robot as if it was a six-axis robot. This proves the system's ability to make robot programming simpler and capable of performing more complex tasks than possible with a traditional position controlled system, without any extensive expertise on redundant kinematics.

11.2 Recommendations for further work

There are several directions this thesis can be expanded upon. The thesis, as an exploratory research project, have singled out four main topics that could be furtherer investigated. Firstly the tuning and optimization of the gain, $k_c$. Secondly, investigations on the use of the weighting matrix and thirdly, studies into avoidability spaces. Lastly could the sensor system be industrialized.

The avoidance gain, $k_c$, could be optimized to get a more responsive system. A highly responsive system is critical if the robot shall operate on any higher velocities than while jogging. If the gains are optimized may the robot be significantly more robust towards dynamic changes in the environment, which is a prerequisite if the system are to operate alongside humans, or unmonitored. The system may also benefit from variable gains, depending on e.g. the primary task and the robots current configuration.

A study on how to exploit the variable weighting matrix, $W$, is an intriguing topic because of the advantages that will bring. The primary task's availability could be considerably increased and the priority of the axes in the secondary task could be higher, since they would only be effective if needed. This could improve the performance of both the primary task and secondary tasks.

In chapter 10 was the MR20's ability to avoid obstacles detected in different places discussed. This could be extended into a study of a robot’s ability to avoid obstacles depending on the critical point and arm configuration. These avoidability spaces could lay the foundation for selection weighting matrix and be a factor in setting the gain. These spaces could also be of importance for the system to know whether the robot is able to avoid a given obstacle at
all. This could then be an indicator to enable the emergency stop, if the system detects an obstacle in a space where the robot can not avoid it.

The sensor system, as it was developed in this thesis, proved the capabilities and advantages of obstacle avoidance. However, the presented sensor system is only designed for research purposes, a study into a development of a industrial standard sensor system would therefore be necessary for an industrial implementation.


List of Figures

1.1 The NACHI MC20 6-axis Robotic Arm. .............................................. 1
1.2 The NACHI MR20 7-axis Robotic Arm. .............................................. 3
1.3 Prinsipal structure of the system. ...................................................... 4
1.4 The Olimex real-time interface. ....................................................... 5

2.1 The anthropomorphic robotic assistant CORA with stereo vision. ............... 10

3.1 The MR20 utilizing all seven axes to hold an orientation in a trajectory. .......... 13
3.2 Geometric representation of the null space and range of $J_e$. .................. 15
3.3 The Denavit-Hartenberg Formulation. ............................................... 20
3.4 The NACHI MR20 performing self-motion. ........................................... 24
3.5 Two workers collaborating and working together with traditional six-axis robots. 24
3.6 The flexibility of a seven axis industrial robot. ..................................... 25
3.7 The NACHI MR20 reaching into a tight spot. ....................................... 25
3.8 A six axis (left) and a seven axis (right) industrial robot used to load and unload parts. ................................................................. 26
3.9 A six axis (left) and a seven axis (right) industrial robot used to load and unload parts. ................................................................. 27

4.1 The virtual extension were modeled as three prismatic axis and three rotational axis to represent the degrees of freedom in task space. .................. 30
4.2 The Suggested point $p^S$ and the reference point $p^R$ on the robot with virtual extension. ................................................................. 31

5.1 A planar robot detects and avoids an obstacle while performing a task. ............ 38
5.2 The flow of the obstacle avoidance as a secondary task. ............................ 39

6.1 Prinsipal structure of the system. ...................................................... 41
6.2 Wireframe model of the NACHI MR20. ............................................... 43
6.3 Sensor placements. ........................................................................... 45
6.4 Sensor placements. ........................................................................... 45
6.5 A sensor's detection area and its corresponding avoidance vector direction. .... 46
6.6 The relation between the gain and the proximity of the obstacle. ................. 47
6.7 The hardware structure of the sensor system. ....................................... 51
6.8 The Parallax’s PING))) ultrasonic sensor. ........................................ 52
6.9 Schematics for the circuitry developed for the US sensors. .................. 52
6.10 Schematics for the circuitry developed for the US sensors. .................. 53
6.11 The relation between measured distance and actual distance on long range for the ultrasonic sensors. ........................................ 53
6.12 The relation between measured distance and actual distance on short range for the ultrasonic sensors. ........................................ 54
6.13 The Sharp infrared proximity sensor. ........................................ 54
6.14 Distance measured by the IR sensors without and with the filter with sample lengths 10 and 20. ........................................ 55
6.15 The relation between measured distance and actual distance on long range for the Infrader sensors. ........................................ 56
6.16 The relation between measured distance and actual distance on short range for the Infrader sensors. ........................................ 56
6.17 The NI 9632 Single Board RIO. ........................................ 57
6.18 The Olimex micro controller and a block diagram of its connections in the AX20. 58
6.19 The code distribution of the sensor system. ........................................ 58
6.20 LabVIEW code for the control of one ultrasonic sensor running on the FPGA. 59
6.21 LabVIEW code for the control of one infrared sensor running on the FPGA. 59
6.22 The inputs and outputs of the FPGA. ........................................ 59
6.23 LabVIEW code for the filter used ultrasonic sensor data. .................. 60
6.24 The inputs and outputs of the controller code. ........................................ 60
6.25 The inputs and outputs of the computer code. ........................................ 61
6.26 The main algorithm executed on the PC. ........................................ 62
6.27 The path correcting algorithm. ........................................ 63
6.28 Flow Chart of the avoidance process. ........................................ 64
6.29 The algorithm governing speed limitation. ........................................ 65
6.30 The algorithm governing joint velocity limitation. ........................................ 67
6.31 Algorithm governing the task reconstruction algorithm. .................. 68
6.32 Reconfigure arm to default configuration. ........................................ 69
6.33 Home position. ........................................ 70
6.34 The working principals of the Olimex ........................................ 71
6.35 Cabling Diagram of the Sensor System ........................................ 71
6.36 The C1 connector block mounted inside the robots wrist and the C1 connector block. ........................................ 72
6.37 The proto-board with cables connected. ........................................ 73
6.38 The front and back of the circular plate with IR sensors mounted. ........ 73
6.39 The front and side of the NACHI MR20 with ultrasonic and infrared sensors mounted. ........................................ 74

7.1 The General settings and Control-Panel in the GUI. .................. 76
7.2 The Sensor settings -> General and Monitoring-Panel in the GUI. .................. 78
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.3</td>
<td>The Sensor settings -&gt; Advanced-Panel in the GUI.</td>
</tr>
<tr>
<td>7.4</td>
<td>The Gripper Control-Panel in the GUI.</td>
</tr>
<tr>
<td>7.5</td>
<td>The Advanced-Panel in the GUI.</td>
</tr>
<tr>
<td>8.1</td>
<td>Frames from the experiment, No Program, No Extended null space.</td>
</tr>
<tr>
<td>8.2</td>
<td>The state at which the code runs over time from the experiment, No Program, No Extended null space.</td>
</tr>
<tr>
<td>8.3</td>
<td>Joint angles added to the AX20's values from the experiment, No Program, No Extended null space.</td>
</tr>
<tr>
<td>8.4</td>
<td>Cartesian Delta position at the critical point caused by the evasive maneuver from the experiment, Shoulder Sensor, No Extended null space.</td>
</tr>
<tr>
<td>8.5</td>
<td>Cartesian Delta position caused by virtual extension from the experiment, No Program, No Extended null space.</td>
</tr>
<tr>
<td>8.6</td>
<td>Frames from the experiment, No Program, No Extended null space.</td>
</tr>
<tr>
<td>8.7</td>
<td>The state at which the code runs over time from the experiment, No Program, No Extended null space.</td>
</tr>
<tr>
<td>8.8</td>
<td>Joint angles added to the AX20's values from the experiment, No Program, No Extended null space.</td>
</tr>
<tr>
<td>8.9</td>
<td>Cartesian Delta position at the critical point caused by the evasive maneuver from the experiment, Shoulder Sensor, With Extended null space.</td>
</tr>
<tr>
<td>8.10</td>
<td>Cartesian Delta position caused by virtual extension from the experiment, No Program, No Extended null space.</td>
</tr>
<tr>
<td>8.11</td>
<td>Frames from the experiment, No Program, No Extended null space.</td>
</tr>
<tr>
<td>8.12</td>
<td>The state at which the code runs over time from the experiment, No Program, No Extended null space.</td>
</tr>
<tr>
<td>8.13</td>
<td>Joint angles added to the AX20's values from the experiment, No Program, No Extended null space.</td>
</tr>
<tr>
<td>8.14</td>
<td>Cartesian Delta position at the critical point caused by the evasive maneuver from the experiment, Elbow Sensor, No Extended null space.</td>
</tr>
<tr>
<td>8.15</td>
<td>Cartesian Delta position caused by virtual extension from the experiment, No Program, No Extended null space.</td>
</tr>
<tr>
<td>8.16</td>
<td>Frames from the experiment, No Program, With Extended null space.</td>
</tr>
<tr>
<td>8.17</td>
<td>The state at which the code runs over time from the experiment, No Program, With Extended null space.</td>
</tr>
<tr>
<td>8.18</td>
<td>Joint angles added to the AX20's values from the experiment, No Program, With Extended null space.</td>
</tr>
<tr>
<td>8.19</td>
<td>Cartesian Delta position at the critical point caused by the evasive maneuver from the experiment, Elbow Sensor, With Extended null space.</td>
</tr>
<tr>
<td>8.20</td>
<td>Cartesian Delta position caused by virtual extension from the experiment, No Program, With Extended null space.</td>
</tr>
<tr>
<td>8.21</td>
<td>Frames from the experiment, No Program, No Extended null space.</td>
</tr>
</tbody>
</table>
8.22 The state at which the code runs over time from the experiment, *No Program, No Extended null space*. .......................... 92
8.23 Joint angles added to the AX20’s values from the experiment, *No Program, No Extended null space*. .......................... 92
8.24 Cartesian Delta position at the critical point caused by the evasive maneuver from the experiment, *Underarm Sensor, No Extended null space*. ................. 93
8.25 Cartesian Delta position caused by virtual extension from the experiment, *No Program, No Extended null space*. .......................... 93
8.26 Frames from the experiment, *No Program, No Extended null space*. .................. 94
8.27 The state at which the code runs over time from the experiment, *No Program, No Extended null space*. .......................... 94
8.28 Joint angles added to the AX20’s values from the experiment, *No Program, No Extended null space*. .......................... 94
8.29 Cartesian Delta position at the critical point caused by the evasive maneuver from the experiment, *Underarm Sensor, With Extended null space*. ................. 95
8.30 Cartesian Delta position caused by virtual extension from the experiment, *No Program, No Extended null space*. .......................... 95

9.1 The experimental setup of the static experiment. .......................... 100
9.2 The grid on the table used in the *No program* case experiment. ........ 101
9.3 Frames from the experiment, *No Program, No Extended null space*. .......................... 102
9.4 The state at which the code runs over time from the experiment, *No Program, No Extended null space*. .......................... 102
9.5 Joint angles added to the AX20’s values from the experiment, *No Program, No Extended null space*. .......................... 103
9.6 Reference position $p_{ref}$ from the experiment, *No Program, No Extended null space*. .......................... 103
9.7 Cartesian Delta position caused by virtual extension from the experiment, *No Program, No Extended null space*. .......................... 103
9.8 Frames from the experiment, *No Program, With Extended null space*. .................. 104
9.9 The state at which the code runs over time from the experiment, *No Program, With Extended null space*. .......................... 104
9.10 Joint angles added to the AX20’s values from the experiment, *No Program, With Extended null space*. .......................... 105
9.11 Reference position $p_{ref}$ from the experiment, *No Program, With Extended null space*. .......................... 105
9.12 Cartesian Delta position caused by virtual extension from the experiment, *No Program, With Extended null space*. .......................... 105
9.13 A peg-tool and a block with a hole used for the Peg-in-Hole experiment. .... 108
9.14 A peg-tool and a block with a hole used for the Peg-in-Hole experiment. .... 109
9.15 Frames from the experiment *Peg-in-Hole, No Avoidance*. .................. 110
9.16 The state at which the code runs over time from the experiment, Peg-in-Hole, No Avoidance. ................................................ 110
9.17 Joint angles added to the AX20’s values from the experiment, Peg-in-Hole, No Avoidance. ................................................ 110
9.18 Reference position $p_{ref}$ from the experiment, Peg-in-Hole, No Avoidance. ....... 111
9.19 Cartesian Delta position caused by virtual extension from the experiment, Peg-in-Hole, No Avoidance. ................................................ 111
9.20 Frames from the experiment, Peg-in-Hole, No Extended null space. .............. 112
9.21 The state at which the code runs over time from the experiment, Peg-in-Hole, No Extended null space. ................................................ 112
9.22 Joint angles added to the AX20’s values from the experiment, Peg-in-Hole, No Extended null space. ................................................ 113
9.23 Reference position $p_{ref}$ from the experiment, Peg-in-Hole, No Extended null space. ................................................ 113
9.24 Cartesian Delta position caused by virtual extension from the experiment, Peg-in-Hole, No Extended null space. ................................................ 113
9.25 Frames from the experiment, Peg-in-Hole, With Extended null space. ............ 114
9.26 The state at which the code runs over time from the experiment, Peg-in-Hole, With Extended null space. ................................................ 114
9.27 Joint angles added to the AX20’s values from the experiment, Peg-in-Hole, With Extended null space. ................................................ 115
9.28 Reference position $p_{ref}$ from the experiment, Peg-in-Hole, With Extended null space. ................................................ 115
9.29 Cartesian Delta position caused by virtual extension from the experiment, Peg-in-Hole, With Extended null space. ................................................ 115
9.30 The grinding tool and the “workpiece”. ................................................ 118
9.31 Frames from the experiment, Grinding, No Avoidance. ............................ 120
9.32 The state at which the code runs over time from the experiment, Grinding, No Avoidance. ................................................ 120
9.33 Joint angles added to the AX20’s values from the experiment, Grinding, No Avoidance. ................................................ 121
9.34 Reference position $p_{ref}$ from the experiment, Grinding, No Avoidance. ....... 121
9.35 Cartesian Delta position caused by virtual extension from the experiment, Grinding, No Avoidance. ................................................ 121
9.36 Frames from the experiment, Grinding, No Extended null space. ................. 122
9.37 The state at which the code runs over time from the experiment, Grinding, No Extended null space. ................................................ 122
9.38 Joint angles added to the AX20’s values from the experiment, Grinding, No Extended null space. ................................................ 123
9.39 Reference position $p_{ref}$ from the experiment, Grinding, No Extended null space. ................................................ 123
9.40 Cartesian Delta position caused by virtual extension from the experiment, Grinding, No Extended null space. ................................................ 123
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.41</td>
<td>Frames from the experiment, <em>Grinding, With Extended null space</em>.</td>
<td>124</td>
</tr>
<tr>
<td>9.42</td>
<td>The state at which the code runs over time from the experiment, <em>Grinding, With Extended null space</em>.</td>
<td>124</td>
</tr>
<tr>
<td>9.43</td>
<td>Joint angles added to the AX20’s values from the experiment, <em>Grinding, With Extended null space</em>.</td>
<td>125</td>
</tr>
<tr>
<td>9.44</td>
<td>Reference position $p_{ref}$ from the experiment, <em>Grinding, With Extended null space</em>.</td>
<td>125</td>
</tr>
<tr>
<td>9.45</td>
<td>Cartesian Delta position caused by virtual extension from the experiment, <em>Grinding, With Extended null space</em>.</td>
<td>125</td>
</tr>
<tr>
<td>9.46</td>
<td>Frames from the experiment, <em>Operation in Constrained Space, No Avoidance</em>.</td>
<td>128</td>
</tr>
<tr>
<td>9.47</td>
<td>The state at which the code runs over time from the experiment, <em>Operation in Constrained Space, No Avoidance</em>.</td>
<td>128</td>
</tr>
<tr>
<td>9.48</td>
<td>Joint angles added to the AX20’s values from the experiment, <em>Operation in Constrained Space, No Avoidance</em>.</td>
<td>129</td>
</tr>
<tr>
<td>9.49</td>
<td>Reference position $p_{ref}$ from the experiment, <em>Operation in Constrained Space, No Avoidance</em>.</td>
<td>129</td>
</tr>
<tr>
<td>9.50</td>
<td>Cartesian Delta position caused by virtual extension from the experiment, <em>Operation in Constrained Space, No Avoidance</em>.</td>
<td>129</td>
</tr>
<tr>
<td>9.51</td>
<td>Frames from the experiment, <em>Operation in Constrained Space, No Extended null space</em>.</td>
<td>130</td>
</tr>
<tr>
<td>9.52</td>
<td>The state at which the code runs over time from the experiment, <em>Operation in Constrained Space, No Extended null space</em>.</td>
<td>130</td>
</tr>
<tr>
<td>9.53</td>
<td>Joint angles added to the AX20’s values from the experiment, <em>Operation in Constrained Space, No Extended null space</em>.</td>
<td>131</td>
</tr>
<tr>
<td>9.54</td>
<td>Reference position $p_{ref}$ from the experiment, <em>Operation in Constrained Space, No Extended null space</em>.</td>
<td>131</td>
</tr>
<tr>
<td>9.55</td>
<td>Cartesian Delta position caused by virtual extension from the experiment, <em>Operation in Constrained Space, No Extended null space</em>.</td>
<td>131</td>
</tr>
<tr>
<td>9.56</td>
<td>Frames from the experiment, <em>Operation in Constrained Space, With Extended null space</em>.</td>
<td>132</td>
</tr>
<tr>
<td>9.57</td>
<td>The state at which the code runs over time from the experiment, <em>Operation in Constrained Space, With Extended null space</em>.</td>
<td>132</td>
</tr>
<tr>
<td>9.58</td>
<td>Joint angles added to the AX20’s values from the experiment, <em>Operation in Constrained Space, With Extended null space</em>.</td>
<td>133</td>
</tr>
<tr>
<td>9.59</td>
<td>Reference position $p_{ref}$ from the experiment, <em>Operation in Constrained Space, With Extended null space</em>.</td>
<td>133</td>
</tr>
<tr>
<td>9.60</td>
<td>Cartesian Delta position caused by virtual extension from the experiment, <em>Operation in Constrained Space, With Extended null space</em>.</td>
<td>133</td>
</tr>
<tr>
<td>10.1</td>
<td>The <em>avoidable</em> and <em>non-avoidable</em> region of the robot’s shoulder.</td>
<td>136</td>
</tr>
<tr>
<td>10.2</td>
<td>The <em>avoidable</em> and <em>non-avoidable</em> region of the robot’s underarm.</td>
<td>137</td>
</tr>
<tr>
<td>10.3</td>
<td>The avoidability space for the MR20.</td>
<td>138</td>
</tr>
</tbody>
</table>
# List of Tables

3.1 Peg-in-hole. .................................................. 22

4.1 Denavit-Hartenberg Matrix for the virtual joint extension .......... 31
4.2 Peg-in-hole. .................................................. 34

6.1 Denavit-Hartenberg Matrix for the MR20 with all joints. .......... 42
6.2 Denavit-Hartenberg Matrix for the MR20 with virtual extension. ... 44
6.3 Proximity sensor labeling strategy. ................................ 46
6.4 Parameters used to determine $k_c$ for the upper-arm sensors. .... 47
6.5 Wireframe model of the kinematics for sensors US-L1 and US-R1 with avoidance unit vectors, gain and the corresponding DH-matrix. .......... 48
6.6 Wireframe model of the kinematics for sensors US-B1 and US-F1 with avoidance unit vectors, gain and the corresponding DH-matrix. .......... 48
6.7 Wireframe model of the kinematics for sensors US-L0 and US-R0 with avoidance unit vectors, gain and the corresponding DH-matrix. .......... 49
6.8 Wireframe model of the kinematics for sensors US-F0 and US-B0 with avoidance unit vectors, gain and the corresponding DH-matrix. .......... 49
6.9 Wireframe model of the kinematics for the IR sensors with avoidance vectors. .............................................. 50
6.10 DH-matrix and $k_c$ values for the IR sensors mounted on the under arm. .............................. 50
6.11 Unit avoidance vectors for the IR sensors. ................................ 50
6.12 Voltage readings at given distances. ................................ 55
6.13 Voltage to distance calculation parameters. ................................ 55
6.14 Encoder values in home position to the left and encoder values for home+10° to the right. ................................................................. 70
6.15 Scaling factors for the joints with rad/encoderlong as unit. ............. 70
6.16 Cables used in the sensor system. ........................................ 72

9.1 Peg end-effector position and hole position and orientation. ........ 108
9.2 Peg-in-hole. .................................................. 109
9.3 Peg end-effector position and hole position and orientation. ........ 118
9.4 Grinding. ...................................................... 119
9.5 Operation in constrained space. .................................... 127
Appendix A

Acronyms

AI    Analog Input
DH    Denavit-Hartenberg Formulation
DI    Digital Input
DIO   Digital Input/Output
DO    Digital Output
DOF   Degrees of Freedom
ETSF  Extended Task Space Formulation
GPM   Gradient Projection Method
GUI   Graphical User Interface
IRB   Industrial Robot
LSM   Least Square Method
SVD   Singular Value Decomposition
VI    Virtual Instrument
WBS   Work Breakdown Structure
sbRIO Single Board Reconfigurable Input/Output
Appendix B

Rotation matrices

\[
\omega_x = \begin{bmatrix}
1 & 0 & 0 \\
0 & \cos(\omega_x) & -\sin(\omega_x) \\
0 & \sin(\omega_x) & \cos(\omega_x)
\end{bmatrix}
\]  
(B.1)

\[
\omega_y = \begin{bmatrix}
\cos(\omega_y) & 0 & \sin(\omega_y) \\
0 & 1 & 0 \\
-\sin(\omega_y) & 0 & \cos(\omega_y)
\end{bmatrix}
\]  
(B.2)

\[
\omega_z = \begin{bmatrix}
1 & 0 & 0 \\
0 & \cos(\omega_z) & -\sin(\omega_z) \\
0 & \sin(\omega_z) & \cos(\omega_z)
\end{bmatrix}
\]  
(B.3)
Appendix C

Attached Materials

The attached material includes:

- The LabVIEW code for obstacle avoidance for NACHI MR20
- Videos documenting the experiments, see below table for list of videos.

<table>
<thead>
<tr>
<th>Video #</th>
<th>File name</th>
<th>Description</th>
<th>Related section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video 1</td>
<td>c01e01</td>
<td>Video of the No Program case, No extended Nullspace experiment.</td>
<td>Section 9.2.2 and 8.2.1</td>
</tr>
<tr>
<td>Video 2</td>
<td>c01e02</td>
<td>Video of the No Program case, With extended Nullspace experiment.</td>
<td>Section 9.2.2</td>
</tr>
<tr>
<td>Video 3</td>
<td>c02e01</td>
<td>Video of the Peg-in-Hole case, No Avoidance experiment.</td>
<td>Section 9.3.2</td>
</tr>
<tr>
<td>Video 4</td>
<td>c02e02</td>
<td>Video of the Peg-in-Hole case, No extended Nullspace experiment.</td>
<td>Section 9.3.2</td>
</tr>
<tr>
<td>Video 5</td>
<td>c02e03</td>
<td>Video of the Peg-in-Hole case, With extended Nullspace experiment.</td>
<td>Section 9.3.2</td>
</tr>
<tr>
<td>Video 6</td>
<td>c03e01</td>
<td>Video of the Grinding case, No Avoidance experiment.</td>
<td>Section 9.4.2</td>
</tr>
<tr>
<td>Video 7</td>
<td>c03e02</td>
<td>Video of the Grinding case, No extended Nullspace experiment.</td>
<td>Section 9.4.2</td>
</tr>
<tr>
<td>Video 8</td>
<td>c03e03</td>
<td>Video of the Grinding case, With extended Nullspace experiment.</td>
<td>Section 9.4.2</td>
</tr>
<tr>
<td>Video 9</td>
<td>c04e01</td>
<td>Video of the Operation in Constrained Space case, No Avoidance experiment.</td>
<td>Section 9.5.2</td>
</tr>
<tr>
<td>Video 10</td>
<td>c04e02</td>
<td>Video of the Operation in Constrained Space case, No extended Nullspace experiment.</td>
<td>Section 9.5.2</td>
</tr>
<tr>
<td>Video</td>
<td>Code</td>
<td>Title</td>
<td>Section</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>------------------------------------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Video 11</td>
<td>c04e03</td>
<td>Video of the <em>Operation in Constrained Space</em> case, <em>With extended Nullspace</em> experiment.</td>
<td>9.5.2</td>
</tr>
<tr>
<td>Video 12</td>
<td>c05e02</td>
<td>Video of the <em>Shoulder Sensor</em> case, <em>Without extended Nullspace</em> experiment.</td>
<td>8.1.1</td>
</tr>
<tr>
<td>Video 13</td>
<td>c05e03</td>
<td>Video of the <em>Shoulder Sensor</em> case, <em>With extended Nullspace</em> experiment.</td>
<td>8.1.2</td>
</tr>
<tr>
<td>Video 14</td>
<td>c06e02</td>
<td>Video of the <em>Underarm Sensor</em> case, <em>Without extended Nullspace</em> experiment.</td>
<td>8.3.1</td>
</tr>
<tr>
<td>Video 15</td>
<td>c06e03</td>
<td>Video of the <em>Underarm Sensors</em> case, <em>With extended Nullspace</em> experiment.</td>
<td>8.3.2</td>
</tr>
</tbody>
</table>
Appendix D

VIs made for this Thesis

calculate theta avoidance Calculate $\theta$ based on $p_{avoid}$.

delta TCP limit Constraints the deflection of the end-effector.

IR Create Avoid Creates the correct $p_{avoid}$ for infrared sensors.

Limit Delta Limits the deflection of the end-effector.

Mask avoid in TCP Mask out undesired axes in the deflection of the end-effector.

set p move Creates a $p_{avoid}$ vector for a ultrasonic sensor based on the proximity of an obstacle and $k_{cmax}$.

us Create Avoid Creates $q^N$ for a ultrasonic sensor.

controller code The code that is executed on the controller.

IR filter Filters the data from the infrared sensors.

delta pos rot Calculate the delta position and rotation between two poses.

joint velocity controller Limits the joint velocities if they are above a set threshold.

Limiter Governs the algorithm for limiting the speed as a result of the output from the PI-controller.

Long to Rad Converts the joint angle values received from the olimex from long to radians.

organize joint space Organize the $q$ vector to the order the olimex can interpret it.

p to transform Manipulate a $p$ vector into matrix form.

p_comp Summarize two translation matrices.

Rad to Long Converts the joint angles from radians to long.

reorganize joint space Reorganizes the joint vector $q$ so that it can be interpreted by the code executed on the computer.
**Set Selection Matrix 1D**  Sets the selection matrix based on the input from the manual control of the selection matrix.

**TCP to Base Transform**  Transforms a translation matrix from tool coordinates to base coordinates.

**Transform to p**  Manipulate a position and orientation on matrix form to a $1 \times 6$ vector $p$.

**Transpose P Avoid**  Transpose the $p_{avoid}$ vector to base coordinates.

**Master FPGA IO**  The code that is executed on the FPGA level of the sbRIO.

**Robot Definition NACHI MR20 tom q2**  Governs the robot definition, including DH-matrix, tool definition and base transform for the MR20 including only the first joints.

**Robot Definition NACHI MR20 tom q3**  Governs the robot definition, including DH-matrix, tool definition and base transform for the MR20 including the two first joints.

**Robot Definition NACHI MR20 tom q4 US**  Governs the robot definition, including DH-matrix, tool definition and base transform for the MR20 including the four first joints.

**Robot Definition NACHI MR20 tom q4**  Governs the robot definition, including DH-matrix, tool definition and base transform for the MR20 including the five first joints.

**Robot Definition NACHI MR20 tom q7**  Governs the robot definition, including DH-matrix, tool definition and base transform for the MR20 including the three first joints.

**Robot Definition NACHI MR20 virtual extension**  Governs the robot definition, including DH-matrix, for the virtual extension.

**Robot Definition NACHI MR20 virtually extended**  Governs the robot definition, including DH-matrix, tool definition and base transform for the full MR20 including the virtual extension.

**Robot Definition NACHI MR20**  Governs the robot definition, including DH-matrix, tool definition and base transform for the full MR20.

**Enable sensors**  Selects which sensors are to be enabled.

**Measure distance US (FPGA)**  Operates and measures a distance using a ultrasonic sensor.

**Pulse width to distance**  Calculates the distance from a given pulse width for the ultrasonic sensors.

**Interpolate Return**  Applies the algorithm for the return to main function.

**Interpolate Trajpt cur to main**  Generates a trajectory for the return to main mode.

**Calculate # Interpolation Points**  Calculates how many interpolation points is needed in the return to main to keep the desired speed set in the GUI.

**return joints to default**  Governs the code that return the arm back to its default configuration.
Appendix E

Datasheets

Datasheets for the following is included:

- NACHI MR20 industrial robot
- NI 9632 Single Board RIO
- Parallax PING))) Ultrasonic sensor
- Sharp GP2Y0A02 Infrared sensor
- Olimex SAM-L9260 Development Board
E.1 NACHI MR20

Flexible motion “Arm” robot with 7-axes

7-axes structure
- Flexible and complex positioning and motion can be available by 7-axes structure.

Compact body, powerful arm
- Minimizing installation space.
- Payload 20kg MAX 30kg (*1)

(*1) Limited envelope within 30kg
Flexible motion "Arm" robot with 7-axes

MR20/20L

<table>
<thead>
<tr>
<th>Item</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Payload</td>
<td>30kg</td>
</tr>
<tr>
<td>Maximum torque (Arm)</td>
<td>80.3Nm</td>
</tr>
<tr>
<td>Maximum speed (Arm)</td>
<td>44.1N/m</td>
</tr>
<tr>
<td>Maximum speed (Wrist)</td>
<td>6.0kgf/cm²</td>
</tr>
<tr>
<td>Maximum speed (Base)</td>
<td>2.0kgf/cm²</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating envelope</td>
<td></td>
</tr>
<tr>
<td>Fig. 2</td>
<td>[Diagram of operating envelope]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example loading system using the MR20</td>
<td>[Diagram of example loading system]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examples of use</td>
<td></td>
</tr>
<tr>
<td>Fig. 3</td>
<td>[Diagram of examples of use]</td>
</tr>
</tbody>
</table>

---

** железный перевернутый робот \n
** Flexible motion "Arm" robot with 7-axes \n
** MR20/20L \n
<table>
<thead>
<tr>
<th>Item</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Payload</td>
<td>30kg</td>
</tr>
<tr>
<td>Maximum torque (Arm)</td>
<td>80.3Nm</td>
</tr>
<tr>
<td>Maximum speed (Arm)</td>
<td>44.1N/m</td>
</tr>
<tr>
<td>Maximum speed (Wrist)</td>
<td>6.0kgf/cm²</td>
</tr>
<tr>
<td>Maximum speed (Base)</td>
<td>2.0kgf/cm²</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating envelope</td>
<td></td>
</tr>
<tr>
<td>Fig. 2</td>
<td>[Diagram of operating envelope]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example loading system using the MR20</td>
<td>[Diagram of example loading system]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examples of use</td>
<td></td>
</tr>
<tr>
<td>Fig. 3</td>
<td>[Diagram of examples of use]</td>
</tr>
</tbody>
</table>
9. Application wiring and piping diagram for standard

9.1. Wiring and piping system diagram

For additional axis encoder, it is necessary to use a bus-connection type encoder that is designated by NACHI. For more details, please contact our technical department.
9.2. Detailed diagram of the base frame

- When not using air, attach plugs to the air inlet holes so that water or oil cannot get into the robot arm. (The plugs are attached when the robot is shipped)
- When attaching air valves or cable brackets etc., please use "Tap holes for application (4-M5)" shown in the figure. Tightening those things with the cover fixing bolts or making holes on the box cover itself may cause water leakage or oil leakage into the arm and the parts of the robot may be damaged.

The application signal lines are in the box.
When using those, make a hole on the cover and attach a cable clamp. The cable clamp must be waterproof type.

Fig. 9.2 Application wiring and piping diagram on the arm

Fig. 9.3 Application wiring and piping diagram on the base
9.3. Detailed diagram of the connectors

1. BJ1 side (connector)

Connector for application cable
- Wire-side shell: JFM-NWA-4-A (JST)
- Guide plate A kit: FM-GPAK-4 (JST)
- Receptacle housing: JFM2FDN-ZV-K (JST)
- Receptacle contact: SJGF-01GF-P1.0 (JST) 0.20 ~ 0.50 sq
  Manual crimp tool: YRS-8861
- Receptacle contact: SJGF-21GF-P1.0 (JST) 0.30 ~ 0.75 sq
  Manual crimp tool: YRF-1120
- Cable diameter suitable for wire-side shell: 8.28 ~ 12.80

(This figure is drawn seeing from the backside of the robot.)

Fig. 9.4 Details of Connectors for Application

2. BJ3 side (junction connector)

Connector form (CN61, CN62 and CN63)
- Housing: J21DF-06V-KX (JST)
- Partner connector
  - Housing: J21DPM-06-KX (JST)
  - Contact: SJ2M-01GF-M1.0N (applicable wire: 0.20 ~ 0.50 mm²)
    Manual crimp tool: YRS-8861
  - Contact: SJ2M-21GF-M1.0N (applicable wire: 0.30 ~ 0.75 mm²)
    Manual crimp tool: YRF-1120
  - Contact: SJ2M-01GF-M1.0S (applicable wire: 0.20 ~ 0.50 mm²)
    Manual crimp tool: YRS-8861

Connector form (CN64)
- Housing: J21SF-03V-KX (JST)
- Partner connector
  - Housing: J21SPM-03V-KX (JST)
  - Contact: SJ2M-01GF-M1.0N (applicable wire: 0.20 ~ 0.50 mm²)
    Manual crimp tool: YRS-8861
  - Contact: SJ2M-21GF-M1.0N (applicable wire: 0.30 ~ 0.75 mm²)
    Manual crimp tool: YRF-1120
  - Contact: SJ2M-01GF-M1.0S (applicable wire: 0.20 ~ 0.50 mm²)
    Manual crimp tool: YRS-8861

(This figure is drawn seeing from the backside of the robot.)

Fig. 9.5 Details of Connectors for Application
NI Single-Board RIO Embedded Control and Acquisition Devices

**Overview and Applications**

NI Single-Board RIO devices are designed to be easily embedded in high-volume applications that require flexibility, high performance, and reliability. NI sbRIO-96xx devices feature an industrial Freescale MPC5200 real-time processor with speeds up to 400 MHz for deterministic real-time applications. The real-time processor is combined via a high-speed internal PCI bus with an onboard reconfigurable Xilinx Spartan-3 field-programmable gate array (FPGA). The FPGA is connected directly to all onboard 3.3 V digital I/O. Each onboard analog and digital I/O module has a dedicated connection to the FPGA as well.

All sbRIO-96xx devices contain 110 bidirectional digital lines. You can select an NI Single-Board RIO device that includes up to 32 analog inputs, four analog outputs, and 32 industrial 24 V digital inputs and digital outputs. In addition to the built-in I/O capabilities, each NI Single-Board RIO device has three connectors for adding board-only versions of NI, third-party, or custom C Series I/O modules.

The sbRIO-96xx devices accept a 19 to 30 VDC power supply and can operate within a -20 to 55 °C temperature range. With the 10/100 Mb Ethernet and serial ports, you can communicate with external devices and systems via TCP/IP, UDP, Modbus/TCP, and serial protocols. The built-in real-time controller also features Web (HTTP) and file (FTP) servers.

**Embedded Software**

The sbRIO-96xx devices are programmed using the NI LabVIEW graphical programming language. The real-time processor runs the LabVIEW Real-Time Module on the Wind River VxWorks real-time operating system (RTOS) for extreme reliability and determinism. You can integrate your C code libraries within LabVIEW Real-Time.

In addition, you can quickly program the onboard reconfigurable FPGA on sbRIO-96xx devices using the LabVIEW FPGA Module for high-speed control, custom I/O timing, and inline signal processing. LabVIEW contains built-in drivers and APIs for handling DMA or interrupt request (IRQ)-based data transfer between the FPGA and real-time processor. You can reuse your existing hardware description language (HDL) libraries and intellectual property (IP) blocks within LabVIEW FPGA.

**Ordering Information**

NI Single-Board RIO products are available in quantity 100 or higher volumes only. For complete product specifications and accessory information, go to ni.com/singleboard.

OEM Pricing Available!

Aggressive discounts are available for high-volume customers. For pricing information, call 800 813 3693 (U.S.).

BUY NOW!

For complete product specifications, pricing, and accessory information, call 800 813 3693 (U.S.) or go to ni.com/singleboard.
## Specifications

### Network
- **Network Interface**: 10BASE-T and 100BASE-TX Ethernet
- **Compatibility**: EEE 802.3
- **Communication rates**: 10 Mb/s, 100 Mb/s autonegotiated
- **Maximum cabling distance**: 100 m/segment

### Power Requirements
- **Power supply voltage range**: 19 to 30 V
- **Power consumption (internal, driving no loads)**
  - sbRIO-960x: 6.00 W
  - sbRIO-961x: 7.50 W
  - sbRIO-963x: 7.75 W
  - sbRIO-964x: 8.00 W

### Available embedded RAM
- Standard device: 320 kb
- sbRIO-960x: 46,080
- sbRIO-961x/963x/964x: 17,280

### Xilinx Spartan-3 Reconfigurable FPGA
- Number of logic cells
  - sbRIO-9611/9631/9641: 17.280
  - sbRIO-9612/9632/9642: 46.080
- Memory: 432 kb
- Configuration memory: 720 kb

### 3.3 V Digital I/O
- **Number of channels**: 110
- **Max current per channel**: 3 mA

#### Output characteristics
- **Output high voltage**: 2.7 V min; 3.3 V max
- **Output low voltage**: 0.07 V min; 0.54 V max

#### Input characteristics
- **Input high voltage**: 2.0 V min; 5.25 V max
- **Input low voltage**: 0 V min; 0.8 V max

### Analog Input (sbRIO-961x/963x/964x)
- **Number of channels**: 32 single-ended or 16 differential
- **ADC resolution**: 16 bits
- **Conversion time**: 4 µs (250 kS/s aggregate)
- **Nominal input ranges**: ±10, ±5, ±1, and ±0.2 V

### Analog Output (sbRIO-963x/964x)
- **Number of channels**: 32
- **DAC resolution**: 16 bits
- **Update time (one channel)**: 3 µs
- **Output range**: ±10 V

### 24 V Digital Input (sbRIO-964x only)
- **Number of channels**: 32
- **Input type**: Sinking

#### Digital logic levels
- **GPI input**: ≤5 V
- **GPO output**: ≤150 µA
- **ON state**: ≥10 V
- **OFF state**: ≥330 µA

### 24 V Digital Output (sbRIO-964x only)
- **Number of channels**: 32
- **Output type**: Sourcing
- **External supply voltage**: 6 to 35 VDC
- **Continuous output current on each channel**
  - No heat sinks: 250 mA max
  - External heat sink added: 1.5 A max [20 A max aggregate]

### Physical Characteristics
- **If you need to clean the device, wipe it with a dry towel.**
- **Torque for screw terminals (J3)**: 0.5 to 0.6 N·m (4.4 to 5.3 lb·in.)
- **Weight**
  - sbRIO-960x: 198.4 g (7.0 oz)
  - sbRIO-961x: 286.5 g (10.1 oz)
  - sbRIO-963x: 269.3 g (9.5 oz)
  - sbRIO-964x: 292.0 g (10.3 oz)

### Safety Voltages
- **Connect only to voltages that are within these limits.**
  - V-to-C: 35 V max, Measurement Category I
- **Caution**: Do not connect to signal or use for measurements within measurement category I, II, III, or IV.

### Compliance
- **National Instruments makes no product safety, electromagnetic compatibility (EMC), or CE marking compliance claims for the sbRIO-961x/963x/964x.** The end-product supplier is responsible for conformity to any and all compliance requirements.
- **Note**: For UL and other safety certifications, refer to the product label or visit [ni.com/certification](http://ni.com/certification), search by model number or product line, and click the appropriate link in the Certification column.

### Waste Electrical and Electronic Equipment (WEEE)
- **EU Customers**: At the end of their life cycle, all products must be sent to a WEEE recycling center. For more information about WEEE recycling centers and National Instruments WEEE initiatives, visit [ni.com/environment/weee.htm](http://ni.com/environment/weee.htm).

### Environmental
- **The sbRIO-96xx devices are intended for indoor use only.** The sbRIO-96xx devices are intended to be built into a suitable enclosure

#### Ambient temperature in enclosure
- **(IEC 60668-2-1, IEC 60668-2-2)**
  - 20 to 55 °C

#### Storage temperature
- **(IEC 60668-2-1, IEC 60668-2-2)**
  - -40 to 85 °C
  - 10 to 90% RH, noncondensing
  - 95% RH, noncondensing

#### Maximum altitude
- **(IEC 60668-2-56)**
  - 2,000 m

#### Pollution degree
- **(IEC 60664)**
  - 2

---

**BUY ONLINE** at [ni.com](http://ni.com) or **CALL 800 813 3693 (U.S.)**
NI Services and Support

NI has the services and support to meet your needs around the globe and through the application life cycle— from planning and development through deployment and ongoing maintenance. We offer services and service levels to meet customer requirements in research, design, validation, and manufacturing. Visit ni.com/services.

Training and Certification
NI training is the fastest, most certain route to productivity with our products. NI training can shorten your learning curve, save development time, and reduce maintenance costs over the application life cycle. We schedule instructor-led courses in cities worldwide, or we can hold a course at your facility. We also offer a professional certification program that identifies individuals who have high levels of skill and knowledge on using NI products. Visit ni.com/training.

Professional Services
Our NI Professional Services team is composed of NI applications and systems engineers and a worldwide National Instruments Alliance Partner program of more than 600 independent consultants and integrators. Services range from start-up assistance to turnkey system integration. Visit ni.com/alliance.

OEM Support
We offer design-in consulting and product integration assistance if you want to use our products for OEM applications. For information about special pricing and services for OEM customers, visit ni.com/oem.

Local Sales and Technical Support
In offices worldwide, our staff is local to the country, giving you access to engineers who speak your language. NI delivers industry-leading technical support through online knowledge bases, our applications engineers, and access to 14,000 measurement and automation professionals within NI Developer Exchange forums. Find immediate answers to your questions at ni.com/support.

We also offer service programs that provide automatic upgrades to your application development environment and higher levels of technical support. Visit ni.com/ssp.

Hardware Services
Calibration Services
NI recognizes the need to maintain properly calibrated devices for high-accuracy measurements. We provide basic or detailed services to recalibrate your products. Visit ni.com/calibration.

Repair and Extended Warranty
NI provides complete repair services for our products. Express repair and advance replacement services are also available. We offer extended warranties to help you meet project life-cycle requirements. Visit ni.com/services.
E.3 Parallax PING)))

PING)))™ Ultrasonic Distance Sensor (#28015)

The Parallax PING))) ultrasonic distance sensor provides precise, non-contact distance measurements from about 2 cm (0.8 inches) to 3 meters (3.3 yards). It is very easy to connect to microcontrollers such as the BASIC Stamp®, SX or Propeller chip, requiring only one I/O pin.

The PING))) sensor works by transmitting an ultrasonic (well above human hearing range) burst and providing an output pulse that corresponds to the time required for the burst echo to return to the sensor. By measuring the echo pulse width, the distance to target can easily be calculated.

Features
- Range: 2 cm to 3 m (0.8 in to 3.3 yd)
- Burst indicator LED shows sensor activity
- Bidirectional TTL pulse interface on a single I/O pin can communicate with 5 V TTL or 3.3 V CMOS microcontrollers
- Input trigger: positive TTL pulse, 2 μs min, 5 μs typ.
- Echo pulse: positive TTL pulse, 115 μs minimum to 18.5 ms maximum.
- RoHS Compliant

Key Specifications
- Supply voltage: +5 VDC
- Supply current: 30 mA typ; 35 mA max
- Communication: Positive TTL pulse
- Package: 3-pin SIP, 0.1” spacing (ground, power, signal)
- Operating temperature: 0 – 70°C.
- Size: 22 mm H x 46 mm W x 16 mm D (0.84 in x 1.8 in x 0.6 in)
- Weight: 9 g (0.32 oz)

Pin Definitions

<table>
<thead>
<tr>
<th>GND</th>
<th>Ground (Vss)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 V</td>
<td>5 VDC (Vdd)</td>
</tr>
<tr>
<td>SIG</td>
<td>Signal (I/O pin)</td>
</tr>
</tbody>
</table>

The PING))) sensor has a male 3-pin header used to supply ground, power (+5 VDC) and signal. The header may be plugged into a directly into solderless breadboard, or into a standard 3-wire extension cable (Parallax part #805-000012).
### Dimensions

![Dimensions Diagram]

### Communication Protocol

The PING sensor detects objects by emitting a short ultrasonic burst and then "listening" for the echo. Under control of a host microcontroller (trigger pulse), the sensor emits a short 40 kHz (ultrasonic) burst. This burst travels through the air, hits an object and then bounces back to the sensor. The PING sensor provides an output pulse to the host that will terminate when the echo is detected, hence the width of this pulse corresponds to the distance to the target.

![Output Pulse Diagram]

<table>
<thead>
<tr>
<th>Host Device</th>
<th>Input Trigger Pulse</th>
<th>t_{OUT}</th>
<th>t_{MIN}</th>
<th>t_{MAX}</th>
<th>t_{BURST}</th>
</tr>
</thead>
<tbody>
<tr>
<td>PING Sensor</td>
<td>Echo Holdoff</td>
<td>750 µs</td>
<td>115 µs</td>
<td>18.5 ms</td>
<td>200 µs</td>
</tr>
<tr>
<td></td>
<td>Burst Frequency</td>
<td></td>
<td></td>
<td></td>
<td>200 µs @ 40 kHz</td>
</tr>
<tr>
<td></td>
<td>Echo Return Pulse Minimum</td>
<td>t_{MIN}</td>
<td></td>
<td></td>
<td>115 µs</td>
</tr>
<tr>
<td></td>
<td>Echo Return Pulse Maximum</td>
<td>t_{MAX}</td>
<td></td>
<td></td>
<td>18.5 ms</td>
</tr>
<tr>
<td></td>
<td>Delay before next measurement</td>
<td></td>
<td></td>
<td></td>
<td>200 µs</td>
</tr>
</tbody>
</table>

---

Copyright © Parallax Inc.  PING sensor Ultrasonic Distance Sensor (#28015)  v1.5  2/19/2008  Page 2 of 12
Practical Considerations for Use

Object Positioning
The PING sensor cannot accurately measure the distance to an object that: a) is more than 3 meters away, b) that has its reflective surface at a shallow angle so that sound will not be reflected back towards the sensor, or c) is too small to reflect enough sound back to the sensor. In addition, if your PING sensor is mounted low on your device, you may detect sound reflecting off the floor.

![Diagram showing object position considerations](image)

Target Object Material
In addition, objects that absorb sound or have a soft or irregular surface, such as a stuffed animal, may not reflect enough sound to be detected accurately. The PING sensor will detect the surface of water, however it is not rated for outdoor use or continual use in a wet environment. Condensation on its transducers may affect performance and lifespan of the device. See the "Water Level with PING Sensor" document on the 28015 product page at www.parallax.com for more information.

Air Temperature
Temperature has an effect on the speed of sound in air that is measurable by the PING sensor. If the temperature (°C) is known, the formula is:

\[ C_{\text{air}} = 331.5 \cdot 0.6 \cdot T_{\text{c}} \text{ m/s} \]

The percent error over the sensor's operating range of 0 to 70 °C is significant, in the magnitude of 11 to 12 percent. The use of conversion constants to account for air temperature may be incorporated into your program (as is the case in the example BS2 program given in the Exemple Programs section below). Percent error and conversion constant calculations are introduced in Chapter 2 of Smart Sensors and Applications, a Stamps in Class text available for download from the 28029 product page at www.parallax.com.
Test Data

The test data on the following pages is based on the PING))) sensor, tested in the Parallax lab, while connected to a BASIC Stamp microcontroller module. The test surface was a linoleum floor, so the sensor was elevated to minimize floor reflections in the data. All tests were conducted at room temperature, indoors, in a protected environment. The target was always centered at the same elevation as the PING))) sensor.

Test 1

Sensor Elevation: 40 in. (101.6 cm)
Target: 3.5 in. (8.9 cm) diameter cylinder, 4 ft. (121.9 cm) tall – vertical orientation
Test 2

Sensor Elevation: 40 in. (101.6 cm)
Target: 12 in. x 12 in. (30.5 cm x 30.5 cm) cardboard, mounted on 1 in. (2.5 cm) pole
Target positioned parallel to backplane of sensor
E.4 Sharp GP2Y0A02

**Features**
1. Less influence on the colors of reflected objects and their reflectivity, due to optical triangle measuring method
2. Distance output type
   - (Detection range: 20 to 150 cm)
3. An external control circuit is not necessary
   - Output can be connected directly to a microcomputer

**Applications**
1. For detection of human body and various types of objects in home appliances, OA equipment, etc

**Absolute Maximum Ratings** *(T_a=25°C)*

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Rating</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply voltage</td>
<td>Vcc</td>
<td>-0.3 to +7</td>
<td>V</td>
</tr>
<tr>
<td><em>Output terminal voltage</em></td>
<td>V_D</td>
<td>-0.3 to Vcc +0.3</td>
<td>V</td>
</tr>
<tr>
<td>Operating temperature</td>
<td>T_{op}</td>
<td>-10 to +60</td>
<td>°C</td>
</tr>
<tr>
<td>Storage temperature</td>
<td>T_{st}</td>
<td>-40 to +70</td>
<td>°C</td>
</tr>
</tbody>
</table>

*Open-collector output

**Recommended Operating Conditions**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Rating</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating Supply voltage</td>
<td>Vcc</td>
<td>4.5 to 5.5</td>
<td>V</td>
</tr>
</tbody>
</table>

*Notice: In the absence of confirmation by device specification sheets, SHARP takes no responsibility for any distortion that may occur in equipment using any SHARP device shown in catalogs, data books, etc. Contact SHARP in order to obtain the latest device specification sheets before using any SHARP devices.

Internal: Internal address for Electronic Components Group http://sharp-world.com/icpg
### Electro-optical Characteristics

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Conditions</th>
<th>MIN.</th>
<th>TYP.</th>
<th>MAX.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance measuring range</td>
<td>ΔL</td>
<td></td>
<td>20</td>
<td></td>
<td>150</td>
<td>cm</td>
</tr>
<tr>
<td>Output terminal voltage</td>
<td>V_{DO}</td>
<td>L=150cm</td>
<td>0.25</td>
<td>0.4</td>
<td>0.55</td>
<td>V</td>
</tr>
<tr>
<td>Difference of output voltage</td>
<td>ΔV_{DO}</td>
<td>L=150cm to 20cm</td>
<td>1.8</td>
<td>2.05</td>
<td>2.3</td>
<td>V</td>
</tr>
<tr>
<td>Average dissipation current</td>
<td>I_{EC}</td>
<td></td>
<td>–</td>
<td></td>
<td>33</td>
<td>50</td>
</tr>
</tbody>
</table>

Note: L: Distance to reflective object

* Using reflective object: White paper (Made by Kodak Co. Ltd., gray cards R-27, white face, reflective white 90%)

*1 Distance measuring range of the optical sensor system

---

**Fig.1 Internal Block Diagram**

![Internal Block Diagram](image)

**Fig.2 Timing Chart**

![Timing Chart](image)
Fig. 3 Analog Output Voltage vs. Distance to Reflective Object

- White Reflectivity 90%
- Gray Reflectivity 18%
NOTICE

- The circuit application examples in this publication are provided to explain representative applications of SHARP devices and are not intended to guarantee any circuit design or license any intellectual property rights. SHARP takes no responsibility for any problems related to any intellectual property right of a third party resulting from the use of SHARP's devices.

- Contact SHARP in order to obtain the latest device specification sheets before using any SHARP device. SHARP reserves the right to make changes in the specifications, characteristics, data, materials, structure, and other contents described herein at any time without notice in order to improve design or reliability. Manufacturing locations are also subject to change without notice.

- Observe the following points when using any devices in this publication. SHARP takes no responsibility for damage caused by improper use of the devices which does not meet the conditions and absolute maximum ratings to be used specified in the relevant specification sheet nor meet the following conditions:

  i) The devices in this publication are designed for use in general electronic equipment designs such as:
     - Personal computers
     - Office automation equipment
     - Telecommunication equipment [terminal]
     - Test and measurement equipment
     - Industrial control
     - Audio visual equipment
     - Consumer electronics

  (ii) Measures such as fail-safe function and redundant design should be taken to ensure reliability and safety when SHARP devices are used for or in connection with equipment that requires higher reliability such as:
     - Transportation control and safety equipment (i.e., aircraft, trains, automobiles, etc.)
     - Traffic signals
     - Gas leakage sensor breakers
     - Alarm equipment
     - Various safety devices, etc.

  (iii) SHARP devices shall not be used for or in connection with equipment that requires an extremely high level of reliability and safety such as:
     - Space applications
     - Telecommunication equipment [trunk lines]
     - Nuclear power control equipment
     - Medical and other life support equipment (e.g., scuba).

- Contact a SHARP representative in advance when intending to use SHARP devices for any "specific" applications other than those recommended by SHARP or when it is unclear which category mentioned above controls the intended use.

- If the SHARP devices listed in this publication fall within the scope of strategic products described in the Foreign Exchange and Foreign Trade Control Law of Japan, it is necessary to obtain approval to export such SHARP devices.

- This publication is the proprietary product of SHARP and is copyrighted, with all rights reserved. Under the copyright laws, no part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, in whole or in part, without the express written permission of SHARP. Express written permission is also required before any use of this publication may be made by a third party.

- Contact and consult with a SHARP representative if there are any questions about the contents of this publication.
E.5 Olimex SAM-L9260

Olimex - SAM9-L9260 - MCU - Development Kit

Product Overview:

SAM9-L9260 is a low cost development platform with ARM9 microcontroller, SAM9 SDRAM and 512MB NAND Flash. The board has Ethernet 100Mbit controller, USB host, USB device, RS232 and 40 pin extension port with all unclassed SAM9260 ports available for add-on boards.

SAM9-L9260 has waste amount of Flash and RAM and runs Linux, WindowsCE and other RTOS natively. The on-board RTC clock is equipped with a 3V Li backup battery.

Kit Contents:

- SAM9-L9260 Development board
- AT91SAM9260 microcontroller

Key Features:

- MCU: AT91SAM9260 16/32 bit ARM™ 133Mhz operation
- Standard JTAG connector with ARM 2x10 pin layout for programming/debugging with ARM-JTAG
- 64 MB SDRAM
- 512MB NAND Flash (seen in Linux as silicon drive)
- Ethernet 100Mbit controller
- USB host and USB device connectors
- RS232 interface and drivers
- SD/MMC card connector
- One user button and one reset button
- One power and two status LEDs
- On-board voltage regulator 3.3V with up to 300mA current
- Single power supply 5V DC required
- Power supply filtering capacitor
- 10.432 MHz crystal on socket
- Extension header
- P.C.B. FR-4, 1.5 mm (0.062"), solder mask, silkscreen component print
- Dimensions: 100 x 60 mm (3.94 x 3.15")
Ordering Information:

Products:

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Manufacturer</th>
<th>Farnell P/N</th>
<th>Newark P/N</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAM9-L9260</td>
<td>Climex</td>
<td>1703512</td>
<td>20R5022</td>
</tr>
</tbody>
</table>

Associated Products:

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Manufacturer</th>
<th>Description</th>
<th>Farnell P/N</th>
<th>Newark P/N</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT91SAM9260</td>
<td>Atmel</td>
<td>MCU, 32BIT, ARM9, BK, SRAM, 217CEGA</td>
<td>1715466</td>
<td>15R0331</td>
</tr>
<tr>
<td>MT46LC4M16A2F-7E:G/G</td>
<td>Micron</td>
<td>SRAM 64MB, SMD, 46LC4, TSOIC64</td>
<td>1216285</td>
<td>97K6123</td>
</tr>
<tr>
<td>NAND5I2V3A2CN6E</td>
<td>Numonyx</td>
<td>FLASH NAND 512MB, SMD, TSOIC48, 512</td>
<td>1507119</td>
<td>01P0405</td>
</tr>
</tbody>
</table>

Similar Products:

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Manufacturer</th>
<th>Description</th>
<th>Support Device</th>
<th>Farnell P/N</th>
<th>Newark P/N</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAM9-L9260</td>
<td>Climex</td>
<td>BOARD, DEV, FOR AT91SAM9260, WFTouch-TFT</td>
<td>AT91SAM9260</td>
<td>1776245</td>
<td>52R3271</td>
</tr>
<tr>
<td>AT91SAM9260-EK</td>
<td>Atmel</td>
<td>BOARD, DEV, FOR AT91SAM9260</td>
<td>AT91SAM9260</td>
<td>1629537</td>
<td>02F5235</td>
</tr>
</tbody>
</table>

Document List:

Datasheets:

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Description</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAM9-L9260</td>
<td>SAM9-L9260, pdf, users manua</td>
<td>519k</td>
</tr>
</tbody>
</table>
### Application Notes:

<table>
<thead>
<tr>
<th>File Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog-to-Digital Converter in the AT91SAM9260A</td>
<td>153kB</td>
</tr>
<tr>
<td>AT91SAM9260 Microcontroller Schematic Check List</td>
<td>122kB</td>
</tr>
<tr>
<td>AT91SAM92 - Speed Integrity and AT91 Products</td>
<td>71.3kB</td>
</tr>
</tbody>
</table>

### Hardware & Software:

<table>
<thead>
<tr>
<th>File Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAM9450C schematic</td>
<td>6.4kB</td>
</tr>
<tr>
<td>SAM94520 REV B schematic</td>
<td>63.7kB</td>
</tr>
<tr>
<td>Linux-2.6.28 atmel7232.zip Linux kernel patches for SAM92</td>
<td>153kB</td>
</tr>
<tr>
<td>u-boot-atmel7232patches-3008017.tar.gz</td>
<td>122kB</td>
</tr>
<tr>
<td>nano-sam9450c.zip NanoX 1.0 patch for SAM92</td>
<td>71.3kB</td>
</tr>
<tr>
<td>SAM9450C test code for EY</td>
<td>1.5kB</td>
</tr>
</tbody>
</table>

### Others Resources:

<table>
<thead>
<tr>
<th>File Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strelk Linux 2.6.28 kernel</td>
<td>41.7kB</td>
</tr>
<tr>
<td>Full list of supported peripherals</td>
<td>5kB</td>
</tr>
</tbody>
</table>
Preliminary study report
Task Programming of Redundant Industrial Robot

stud.techn. Audun Sanderud

Spring 2012

Department of Production and Quality Engineering
Norwegian University of Science and Technology

Supervisor: Professor II Trygve Thomessen
Preface

This thesis is a part of the fifth year specialization course TPK4510, which in turn is a part of an integrated masters degree in mechanical engineering at NTNU, Trondheim. The course is rewarded 30 credits, which is equivalent to a workload of approx. 48 hours pr week. The project will be evaluated on the basis of a written report, as well as other material pertaining to it, which will be submitted to the Department of Production and Quality Engineering (IPK) within June 11th 2012.

The project will be carried out by stud.tech. Audun Sanderud on behalf of PPM AS and NTNU, and will contain both theoretical studies as well as practical implementations.

The problem description has been developed by Prof. II Trygve Thomessen and concerns task programming of redundant industrial robots.

Thorugh this the candidate will learn about the subject matter, practical work in a research environment as well as project management.

The preliminary study is performed in order to gain an overview of the project’s tasks and scope at an early stage. The documents in the preliminary report will be used as tools for providing pointers for both time and resources needed to retain an appropriate progression in the main project.
# Contents

1 Project description  
1.1 Background .................................................. 1  
1.2 Problem ......................................................... 1  
1.3 Approach ....................................................... 2  

2 Project Partners  

3 Project planning  
3.1 Work Breakdown Structure ................................ 7  
3.2 Work Packages ................................................ 8  
3.3 Resource Distribution ....................................... 11  
3.4 Gantt .............................................................. 12  
3.5 Milestones ....................................................... 13
CONTENTS

**Acronyms**

**DOF**  Degrees of Freedom  
**ETSF**  Extended Task Space Formulation  
**DH**  Denavit-Hartenberg Formulation  
**VI**  Virtual Instrument  
**GUI**  Graphical User Interface  
**IRB**  Industrial Robot  
**MR20**  NACHI MR20 Industrial Robot  
**WBS**  Work Breakdown Structure  
**F/T**  Forces and Torques
1 Project description

1.1 Background

Industrial robots have nowadays normally six degrees of freedom to provide an arbitrary position and orientation of the tool inside its working space. However, during the last years, there have been developed industrial robots with more than six axes. This gives extra functionality, i.e., to change the internal configuration of the robot arm to avoid singular positions, to move around obstacles and to optimize the use of energy during a predefined trajectory.

However, this functionality is dependent on the user's ability to program the robot efficiently and skillfully which makes the programming complicated and time consuming for redundant robots.

1.2 Problem

This master thesis is focusing on how to enable this extra functionality automatically so the operator can simply focus on the programming of the tool position, similar to a normal six axes robot, while the control system automatically take care of the internal configuration of the robot arm. This includes:

- Automatic reconfiguration of the robot arm to avoid obstacles. The work done by Kosuke Wada will be used as a groundwork on obstacle avoidance. An automatic algorithm has to be developed to automatically change of the internal configuration of the robot and to avoid the obstacle.

- Automatic reconfiguration of the robot arm to avoid singularities. This is to be done by continuously monitoring the internal configuration of the robot arm, and detect when the robot arm is close to any singularities. Then, an automatic algorithm has to be developed to automatically reconfigure the robot to avoid to run further into the singular area.

- Based on the path generated by obstacle- and singularity avoidance, automatically optimize the internal kinematic configuration according to less energy use.

- There has also to be developed an overall algorithm which automatically selects the best configuration of the robot arm in case there is any contradiction between the configuration of the robot arm to avoid obstacles and singularities. This algorithm should have a set of rules which can be defined and prioritized by the user.
1.3 Approach

The solutions will be developed in PPM’s laboratory in Trondheim, for NACHI MR2O industrial robot, with PPM’s high speed USB interface to NACHI’s AX2O controller. The implementation will be developed in LabVIEW 2011 using a standard personal computer (PC) communicating with the OLIMEX card and the collision sensors attached to the MR2O robot. The detection of singularities is done by reading the MR2O’s axes positions through the OLIMEX card which communicates with high speed with NACHI’s AX2O controller. See Figure 1.

![Layout of the hardware setup](image)

Figure 1: Layout of the hardware setup

The following tasks have to be accomplished:

i. Introductory literature study about methodology for collision- and singularity avoidance in addition to optimization of energy use during running the robot along a pre-defined trajectory.


iii. Development of methodology for energy optimization when running the robot along a predefined trajectory.

iv. Development of hardware solution and instrumentation of the robot system.

v. Development of software solution for the robot system
1.3 Approach

vi. Experimental testing of the MR2O robot system for collision- and singularity avoidance.

vii. Documentation of the experimental setup including hardware and software.

viii. Documentation of the user functions to operate the experimental setup.

ix. Documentation of the experimental results
2 Project Partners

NTNU
The Norwegian University of Science and Technology is the main provider of higher education in technological and natural science in Norway. Amongst the 20,000 students, more than 10,000 are studying technological subjects. The department of production and quality engineering is situated under the Faculty of Engineering Science and Technology. The department is focusing on education and research in three areas: Production Systems, Product Management and Reliability, Availability, Maintainability and Safety. The department has extensive experience with project- and master thesis in close cooperation with the industry.

PPM AS
PPM (Productive Programming Methods) was founded in December 2000 by Dr.ing Trygve Thomessen and Siv.ing Per Kristian Sannæs. PPM has it’s main focus on R&D projects on productivity improvement and robotics in low batch production.
3 Project planning

In this chapter we will utilise various tools for project planning and control described by
This is done in order to gain an overview of the project's tasks and scope at an early stage.
The following documents will be presented:

**Project overview Statement**
Overview over the projects problem, goals, success criteria, conditions, risks and obsta-
cles.

**Work Breakdown Structure**
Is a one-dimensional breakdown of the work. The project is broken down into smaller ele-
ments in a logical and systematical manner. A WBS can assist in identifying the most impor-
tant parts of the project and their relationship to each other and the project as a whole.

**Resource Distribution**
Provides an overview over how much time is allocated to the various activities.

**Gantt**
Is a well-known planning and scheduling tool which shows the various activities vs. time in a
diagram. It provides a good visual presentation of all the activities and their duration as well
as sequence.

**Milestones**
Table of all milestones and the planned date of reaching them.
### Project Overview Statement

<table>
<thead>
<tr>
<th>Project:</th>
<th>Task Programming Of Redundant Industrial Robot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Responsible:</td>
<td>stud. techn. Audun Rønning Sanderud</td>
</tr>
</tbody>
</table>

#### Problem:
Industrial robots have nowadays normally six degrees of freedom to provide an arbitrary position and orientation of the tool inside its working space. However, during the last years, there have been developed industrial robots with more than six axes. This gives extra functionality, i.e., to change the internal reconfiguration of the robot arm to avoid singular positions, to move around obstacles and to optimize the use of energy during a predefined trajectory. However, this functionality is dependent on the user's ability to program the robot efficiently and skilfully which makes the programming complicated and time consuming for redundant robots.

#### Main Goal:
Develop a system that automatically reconfigures the internal configuration of the redundant robot manipulator to avoid obstacles and singularities. And to develop an algorithm that reconfigures the joint motion in a predefined path to optimize the energy consumption. The system must be implemented on a NACHI MR20 iRb with an AX20 controller and an Olimex cad as a high-speed interface.

#### Secondary Goal:
- Achieve sufficient understanding of the theory
- Model the kinematics of the robot with all possible joint as the redundant joint.
- Describe and find a solution to all possible singular configurations
- Develop an algorithm to optimize the joint motion in a predefined path.
- Write a paper on force control of redundant industrial robots.
- Write a Preliminary report
- Write a Final Report, with sufficient documentation

#### Success Criteria:
- The produced material meet the expectations of the partners in the projects
- Get top grade
- The report is adaptable to a publishable paper

#### Conditions, Risks and Obstacles:

<table>
<thead>
<tr>
<th>Conditions:</th>
</tr>
</thead>
<tbody>
<tr>
<td>The candidate can get sufficient knowledge about the subject</td>
</tr>
<tr>
<td>The candidate can cooperate with the supervisor</td>
</tr>
<tr>
<td>The project is sufficiently planned</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Risks:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Illness</td>
</tr>
<tr>
<td>The magnitude of the project is to comprehensive</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Obstacles:</th>
</tr>
</thead>
<tbody>
<tr>
<td>The workload as a student assistant is greater than expected.</td>
</tr>
</tbody>
</table>
3.2 Work Packages

1. Master Thesis
   1.1. Writing paper on force control of redundant robot
      1.1.1. Literature search and introduction
      Search for references and supporting literature for the redundant kinematics in the force control system. Get familiar with the form of the papers previously submitted to SYROCO (Symposium on Robot Control). Then write an introduction based on this and the results form the "sensor based control of redundant industrial robots"-report.
      1.1.2. Reproduce necessary experimental data
      Reproduce and document the necessary experiments for the report. This includes photos, data for F/T and position in both base- and joint space coordinates.
      1.1.3. Write paper
      Write the remainder of the paper.

   1.2. General and Theory
      1.2.1. Literature search
      Search for, and study the necessary literature. Define redundancy and singularity. This includes how obstacle avoidance, singularity avoidance and energy optimization might have been solved in previous projects. It shall also involve a study on how inverse kinematics may be solved traditionally. The theory is to be presented in the masters thesis.

      1.2.2. Kinematic models
      Describe the different kinematic models required to utilize ETSF for all possible redundant joint. The work shall result in a set of figures and parameters to describe the possible kinematic models.

   1.3. Obstacle Avoidance
      1.3.1. Avoid static obstacles while programming
      Develop a system that enables the user to program a path, while the robots internal configuration automatically reconfigures to avoid static obstacles. Investigate Virtual Extended Task Space.
3.2 Work Packages

1.3.2. Dynamic obstacles

Develop a system that gives the robot the ability to reconfigure its internal configuration to avoid obstacles while performing a task. The robot must keep performing its given task as long as it is possible. The work shall also include a task reconstructing algorithm. The user should be notified when approaching an obstacle.

1.4. Singularity Avoidance

1.4.1. Reveal and describe all possible singularities

Reveal and describe all possible singularities for the NACHI MR20. The description must include the states of the joints contributing to the singularity and which states might be used to detect possible singularities approaching. This shall result in a set of figures and joint configurations where a singularity occurs, as well as what rules might be used to limit the joints causing the singularity.

1.4.2. Investigate and implement solutions

Investigate whether or not switching ETSF or Masked Jacobian Method might be utilized to avoid singularities. Research what other ways to implement singularity avoidance that utilizes the redundancy in the MR20. How can user defined rules be implemented? Implement in the force control scheme developed in the specialization project from fall 2011.

1.5. Joint Angle Optimization

1.5.1. As a Post processor on a preprogrammed path

Develop and implement a control that will optimize the different axes motion through a preprogrammed path. This must not affect the change in the internal configuration caused by singularity or obstacle avoidance.

1.5.2. Adapting to applied forces

Research the opportunity and implement an algorithm that use the seven axes to ensure as low force as possible perpendicular to $q_5$’s rotational axis.

1.6. Report Work

1.6.1. Preliminary report

Write and edit a preliminary report containing problem description, WBS and Gantt diagram of the planned work.
1.6.2. **Documentation and demonstration**

Creating necessary documentation to demonstrate the achievements of the work. This may include video, photos and the developed code.

1.6.3. **Final report**

Write and edit the final report.
3.3 Resource Distribution

The resource distribution of the work packages in days.

<table>
<thead>
<tr>
<th>Subproject</th>
<th>1.1 Write paper on force control of redundant robots</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work Package Hours</td>
<td>1.1.1 1.1.2 1.1.3</td>
</tr>
<tr>
<td>Hours</td>
<td>3 2 5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subproject</th>
<th>1.2 General and Theory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work Package Hours</td>
<td>1.2.1 1.2.2</td>
</tr>
<tr>
<td>Hours</td>
<td>5 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subproject</th>
<th>1.3 Obstacle Avoidance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work Package Hours</td>
<td>1.3.1 1.3.2</td>
</tr>
<tr>
<td>Hours</td>
<td>10 10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subproject</th>
<th>1.4 Singularity Avoidance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work Package Hours</td>
<td>1.4.1 1.4.2</td>
</tr>
<tr>
<td>Hours</td>
<td>3 10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subproject</th>
<th>1.5 Joint Angle Optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work Package Hours</td>
<td>1.5.1 1.5.2</td>
</tr>
<tr>
<td>Hours</td>
<td>10 5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subproject</th>
<th>1.6 Report Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work Package Hours</td>
<td>1.6.1 1.6.2 1.6.3</td>
</tr>
<tr>
<td>Hours</td>
<td>10 5 20</td>
</tr>
</tbody>
</table>

| Total                        | 100             |
## 3.4 Gantt

<table>
<thead>
<tr>
<th>Task</th>
<th>Name</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Master Thesis</td>
<td>106 days</td>
</tr>
<tr>
<td>1.1</td>
<td>Writing Paper on force control of redundant robot</td>
<td>8 days</td>
</tr>
<tr>
<td>1.1.1</td>
<td>Literature search</td>
<td>3 days</td>
</tr>
<tr>
<td>1.1.2</td>
<td>Reproduce necessary experimental data</td>
<td>2 days</td>
</tr>
<tr>
<td>1.1.3</td>
<td>Write Paper</td>
<td>3 days</td>
</tr>
<tr>
<td>1.2</td>
<td>General and Theory</td>
<td>7 days</td>
</tr>
<tr>
<td>1.2.1</td>
<td>Literature search</td>
<td>5 days</td>
</tr>
<tr>
<td>1.2.2</td>
<td>Kinematics model</td>
<td>2 days</td>
</tr>
<tr>
<td>1.3</td>
<td>Obstacle Avoidance</td>
<td>30 days</td>
</tr>
<tr>
<td>1.3.1</td>
<td>Avoid static obstacles while programming</td>
<td>10 days</td>
</tr>
<tr>
<td>1.3.2</td>
<td>Dynamic obstacles</td>
<td>10 days</td>
</tr>
<tr>
<td>1.4</td>
<td>Singular Avoidance</td>
<td>13 days</td>
</tr>
<tr>
<td>1.4.1</td>
<td>Reveal and describe all possible singularities</td>
<td>3 days</td>
</tr>
<tr>
<td>1.4.2</td>
<td>Investigate and implement solution</td>
<td>10 days</td>
</tr>
<tr>
<td>1.5</td>
<td>Joint Angle Optimization</td>
<td>15 days</td>
</tr>
<tr>
<td>1.5.1</td>
<td>As a postprocessor on a preprogrammed path</td>
<td>10 days</td>
</tr>
<tr>
<td>1.5.2</td>
<td>Adapting to applied forces</td>
<td>5 days</td>
</tr>
<tr>
<td>1.6</td>
<td>Joint Angle Optimization</td>
<td>15 days</td>
</tr>
<tr>
<td>1.6.1</td>
<td>Preliminary Report</td>
<td>10 days</td>
</tr>
<tr>
<td>1.6.2</td>
<td>Documentation and demonstration</td>
<td>5 days</td>
</tr>
<tr>
<td>1.6.3</td>
<td>Final Report</td>
<td>20 days</td>
</tr>
<tr>
<td>2</td>
<td>Report</td>
<td>103 days</td>
</tr>
<tr>
<td>2.1</td>
<td>Preliminary Report</td>
<td>10 days</td>
</tr>
<tr>
<td>2.2</td>
<td>Documentation and demonstration</td>
<td>5 days</td>
</tr>
<tr>
<td>2.3</td>
<td>Final Report</td>
<td>20 days</td>
</tr>
<tr>
<td>3</td>
<td>Paper Finished</td>
<td>0 days</td>
</tr>
<tr>
<td>4</td>
<td>Status Report Finished</td>
<td>0 days</td>
</tr>
<tr>
<td>5</td>
<td>All Programming Finished</td>
<td>0 days</td>
</tr>
<tr>
<td>6</td>
<td>Final Report Finished</td>
<td>0 days</td>
</tr>
</tbody>
</table>
### 3.5 Milestones

<table>
<thead>
<tr>
<th>Event</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preliminary report finished</td>
<td>03.02.12</td>
</tr>
<tr>
<td>Paper finished</td>
<td>15.02.12</td>
</tr>
<tr>
<td>Status Report Finished</td>
<td>13.04.12</td>
</tr>
<tr>
<td>Programming and Implementation completed</td>
<td>20.04.12</td>
</tr>
<tr>
<td>Final Report Finished</td>
<td>08.06.12</td>
</tr>
</tbody>
</table>
Preliminary Literature Index


Mayorga, R. V. An Artificial Neural Network Approach for the Obstacle Avoidance of Redundant Robot Manipulators. *Neural Networks*, page 5.


Appendix G

Progress Report
Progress Report

Task Programming of Redundant Industrial Robot

Stud.techn. Audun Sanderud

Spring 2012

Department of Production and Quality Engineering
Norwegian University of Science and Technology

Supervisor: Professor II Trygve Thomessen
1 Progress Report

This report will cover the progress and developments of the work related to the master thesis, Task Programming of Redundant Industrial Robot, since project start in January 2012. The previously submitted preliminary study report presented various topics for further investigation. During the work, however, have the candidate in consultation with the supervisor decided to remove the section regarding energy efficiency from the master thesis. It was also decided to deviate from the original plan to use the existing proximity sensor system, and to develop a new system, although based on the existing one.

2 Project Status

The work has now led to a task formulation based on sub-task controlled extended null space. Task Formulation of Compliant Tasks and the Gradient Projection Method for redundancy resolution has been some of the most important base theories. The result so far is a generalized model for task formulation of compliant task with redundant robots. A proximity sensory system has been developed and implemented on a NACHI MR20 7-axis industrial robot. This system has been basis for a verification of the developed task formulation where the sub-task is based on obstacle avoidance. So far have only a few unrecorded experiments been conducted, but the results are promising.

3 Further Progress

The next step is to experimentally test the the task formulation through three cases with obstacle avoidance. A classic Peg-In-Hole, a grinding case and a Pick-And-Place will be subject to experimentation. The two first cases will deal with dynamic obstacles and the last will deal with static obstacles. Work on the thesis is also a great part of the remaining work.
The above Gantt chart has been updated so that work packages located behind the "current date"-line has been finished and tasks in front of the line are still to be done. This gives an insight into the current progress in the project and provides a schedule for the remaining tasks.

Compared to the original schedule\(^1\), the main differences are that the "Joint Angle Optimization" work package has been removed, The "Singularity Avoidance" section has been minimized and mainly replaced with "Proximity Sensor System". This is due to the generalized task formulation that were developed during the work with obstacle avoidance and the decision to make a new sensor system.

\(^1\)The original schedule can be found in the preliminary study report.