THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

CONNECTING VIRTUAL ROBOTICS TO AN EXPERIMENTAL PLATFORM

by
Callum Rohweder

School of Information Technology and Electrical Engineering,
The University of Queensland.

Submitted for the degree of
Bachelor of Engineering
in the field of Mechatronics

June & 2018.

52 KENNIGO STREET
SPRING HILL, QLD, 4000
Tel. 0404 639 174

May 24, 2018

Professor Shazia Sadiq
Head of School
School of Information Technology and Electrical Engineering
The University of Queensland
St Lucia, Q 4072

Dear Professor Sadiq,

In accordance with the requirements of the degree of Bachelor of Engineering in the division of Mechatronic Engineering, I present the following thesis entitled 'Connecting Virtual Robotics To An Experimental Platform'. This work was performed under the supervision of Dr Surya Singh.

I declare that the work submitted in this thesis is my own, except as acknowledged in the text and footnotes, and has not been previously submitted for a degree at The University of Queensland or any other institution.

Yours sincerely,

Callum Rohweder.

Acknowledgments

I specifically would like to thank my fellow students in the school of ITEE for assisting me during this project; for listening to the software issues I faced and giving useful insight and direction.

The product of this thesis was created in complete self-sufficiency, with functionality specified by my supervisor.

Acknowledge your supervisor, preferably with a few short and specific statements about his/her contribution to the content and direction of the project. If you collaborated with another student, acknowledge your partner's contribution, including any parts of the thesis of which s/he was the principal author or co-author; this information can be duplicated in footnotes to the chapters or sections to which your partner has contributed. Briefly describe any assistance that you received from technical or administrative staff. Support of family and friends may also be acknowledged, but avoid sentimentality—or hide it in the dedication.

Abstract

This document is a skeleton thesis for 4th-year students. The printable versions (`skel.dvi, skel.ps, skel.pdf`) show the structure of a typical thesis with some notes on the content and purpose of each part. The notes are meant to be informative but not necessarily illustrative; for example, this paragraph is not really an abstract, because it contains information not found elsewhere in the document. The LaTeX 2_ε source file (`skel.tex`) contains some non-printing comments giving additional information for students who wish to typeset their theses in LaTeX. You can download the source, edit out the unwanted material, insert your own frontmatter and bibliographic entries, and in-line or `\include{}` your own chapter files. Of course the content of a particular thesis will influence the form to a large extent. Hence this document should not be seen as an attempt to force every thesis into the same mold. If in doubt about the structure of your thesis, seek advice from your supervisor.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Remotely controlling a robot or robotic manipulator is a desirable objective with large complications to still be overcome. It is proposed that doing this through a virtual environment can eliminate collisions or undesirable actions that may incidentally occur due to the nature of long distance control, whilst providing the benefits of virtual simulations. Specifically, this thesis focuses on the development a remote interface to a 'Virtual Robotics Environment Platform', VREP. VREP can be used to simulate robotic arms and processes, and includes the commonly used wheel-chair based Kinova Jaco arm which is available at the University of Queensland. The remote interface, otherwise known as the 'client program' (given that it treats VREP as the server), was designed to take input from a user by either keyboard or joystick, and have the motion played out in VREP, corrected for any obstructions, and then control the movement of the Jaco arm through the Kinova Software Development Kit.

In all engineering disciplines, virtual simulations allow one to view and interact with an environment or process in a non-destructive manner. Simulations can provide a realistic rendering of an event, whilst providing further detail into physical phenomena that establish design constraints, and optimization techniques. Robotics makes use of lumped electromechanical components to interact with an environment in a desirable manner. Thus, virtual robotics is a necessary field for growth in engineering, as it allows the testing of interactions with an environment whilst giving unforeseen insight.

Companies such as those in manufacturing, technical experts in the fields of medicine and surgery, and persons with disabilities all benefit from the capability of robotic manipulators. In most circumstances it is expected that these manipulators can be controlled by a user in real time, however this ability is restricted by inherent delay in the process of receiving an input, calculating an action, and actuating. Further expectations of robotics include optimality and customisation; where it may be desirable for movements of a robot to minimize the energy used in a given

process or a robotic arm to pick up a glass a in a certain manner. This provides interconnected layers of desired functionality for a robot; a layer dedicated to moving the manipulator, a layer designed to create movements that meet the user's needs whilst minimizing design criteria, and a monitoring layer that focuses on aspects such as physical constraints and robotic learning.

With the invention of the internet to provide long range data-resourcing, came a desire to move the control of manipulators and processes to a remote location. The concept of remote robotics is no different to the typical method of controlling manipulators, an input has a desired output, however at some time in between, data is processed and sent through the internet. Given factors such as time to send, packet loss, processing time, and internet traffic, a significant amount of undesirable delay is added and decreases the satisfaction of real-time control. This produces large complications in areas such as remote robotic surgery, where reaction delay may have harmful effects.

Virtual robotic environments can simulate the true movements of a manipulator given its physical attributes. VREP in particular can calculate the joint angles required to be able to move the hand from one position to another using a physics engine and accuracy of choice. It is believed that allowing a virtual environment to compute the movements required by a remote user will increase the accuracy in movement and decrease the chance of collision of the physical robot.

Although this thesis does not go into large detail on the testing of the complete product, remote user to physical robotic arm movement, it does go through the design of the remote interface. The problems faced in controlling a robotic arm, design strategies tested, and the future improvements of the client program are presented. This interface was crucial to the success of the whole system, and it was important to refine before interfacing with a physical arm. Included in this document is:

- A comparison of different approaches for moving a robotic arm, along with the calculations and methods for refining accuracy and decreasing processing time.
- An overview of VREP, and how its features were used to replicate the true movements of the Jaco arm. This includes a comparison of the available physics engines, the use of PID motor control for arm joints and Inverse Kinematics (IK) functionality.
- Validity of using VREP as a real-time simulation tool, looking into aspects such as rendering delay and communication techniques.
- The features included in client program and how its functionality was shaped,

and the limitations associated with a real-time program performing large amounts of calculations based off user inputs and a remote server.

- An overview of the Jaco Kinova arm and it's software development kit, which can be used to control the Jaco arm from VREP or the client program.

To accomplish the task of remotely controlling the Jaco arm with collision avoidance, the project was broken into four main stages. The long-term objectives to acheive the aforementioned goals, presented in , are:

- to configure the external simulation control of VREP and the Jaco arm within. This included retrieving all of the arm's joint information on start-up, so the correct joints can be referenced during connection. The central purpose of this objective was to achieve control of the arm in simulation, getting it to move its gripper to desired coordinates.

- using the information from VREP of the arm, and the current joint angles from the physical Jaco arm using the SDK, achieve movement replication between the the physical and virtual systems. The purpose of this was that it would allow tuning of the VREP simulation to mimic the way the physical arm moves and responds to commands.

- with VREP tuned appropriately, allow the movement of the physical arm by controlling movement in the simulation. It was predicted that movement in the simulation would come from keyboard commands or on-screen mouse presses.

- detect collision, move objects in the laboratory and interact with the environment in simulation, with the Jaco arm replicating the motion.

Due to unforeseen circumstances, the objectives presented in this project changed to focus on the different ways of interacting with the virtual arm through VREP's remote API. It's primary focus is the validity of using a remote client to interact with VREP, and whether a good interface with the aforementioned requirements can be created. The objectives of this thesis were to:

- Create a user friendly interface for controlling a robotic arm in VREP, robust to arm location within a scene and mode of the virtual arm (forward kinematics or inverse kinematics modes)

- Allow the user to give input types as they desire, such as moving the Jaco arm gripper from point to point, or controlling the joint angles individually; with keyboard input or joystick input.

- Experiment with different mathematical solutions and physics engines to achieve accurate and fast movement of the virtual robotic arm
- Experiment and outline the different functionality VREP has to offer, which may be utilised in the development of the over-arching objectives.

From here, a variety of challenges will be pondered as mathematical formulas are derived to describe the motion of a 6 degree of freedom (DOF) arm, and a remote interface is created for allowing real-time movement within VREP.

Chapter 2

Programming Robots to Move

In order to move a robotic arm, two layers of control are required, the motor position control for each joint, and the system that depicts how much the joints need to change angle when moving from one arm configuration to another. Following are some case studies that present methods of controlling a robotic arm, where the maths behind them are left for the theory section of this report.

2.1 MATLAB Toolboxes

Matlab provides an extraordinary amount of support for calculating the important properties required for the control of a robotic manipulator. Below is a sample of some of the useful libraries available that were, and further could be, beneficial to creating an interface for controlling the virtual Kinova Jaco arm remotely.

2.1.1 Robotics Toolbox

Professor Peter Corke, the Director of the Australian Centre for Robotic Vision (ACRV), has designed a Matlab specific toolbox for computing the mechanics associated with robotic manipulators. Included is easy to use functions for creating robotic arms, and calculating parameters such as forward and inverse kinematics, trajectory generation and control systems for motor control. Within the Queensland University of Technology's Robot Academy is 'Master Classes' on using the toolbox through examples of simple manipulators and the theory behind the complex calculus being performed.

There are dozens of object files containing the parameters of robotic manipulators; including the 6 DOF puma560 and limited information on the 6DOF Kinova Jaco arm. The parameters within these files contain the link times of the arm, the Denavit-Hartenberg parameters (DH parameters - theory section of this report), and

can contain inertial and angle properties for each link. These properties are helpful in computing the forward kinematics, inverse kinematics, and dynamic equation representing the motion and torques on the arm.

With this, the toolbox allows the viewing of the manipulator in desired angles, the movement of the arm between two desired end-effector positions, and the building of the control system controlling the torque at each motor given the current arm configuration. The latter of which is crucially important in providing accurate control of the arm, in ensuring the appropriate power is provided to all of the joint links as it moves over a given trajectory. Unfortunately, it is extremely difficult to compute, and for each degree of freedom contains 10 constants that need finding; mostly inertia terms which for each link depends on its orientation and position relative to the arm's base; thus relying on all link angles between the base and the link of interest. The mechanics of this is presented in the theory section of the report, but the Robotics Toolbox provides functions which compute the inertial, coriolis and gravitational torques acting on the arm. Professor Corke also provides a method of motor control for each link given the aforementioned parameters are known. The method uses the typical motor position control strategy of a Proportional Derivative Controller (details in the theory section) to compute the required motor torque from the error in angle position and velocity, however also includes a feed-forward component of the calculated disturbance torque due to the coupled dynamics between links.

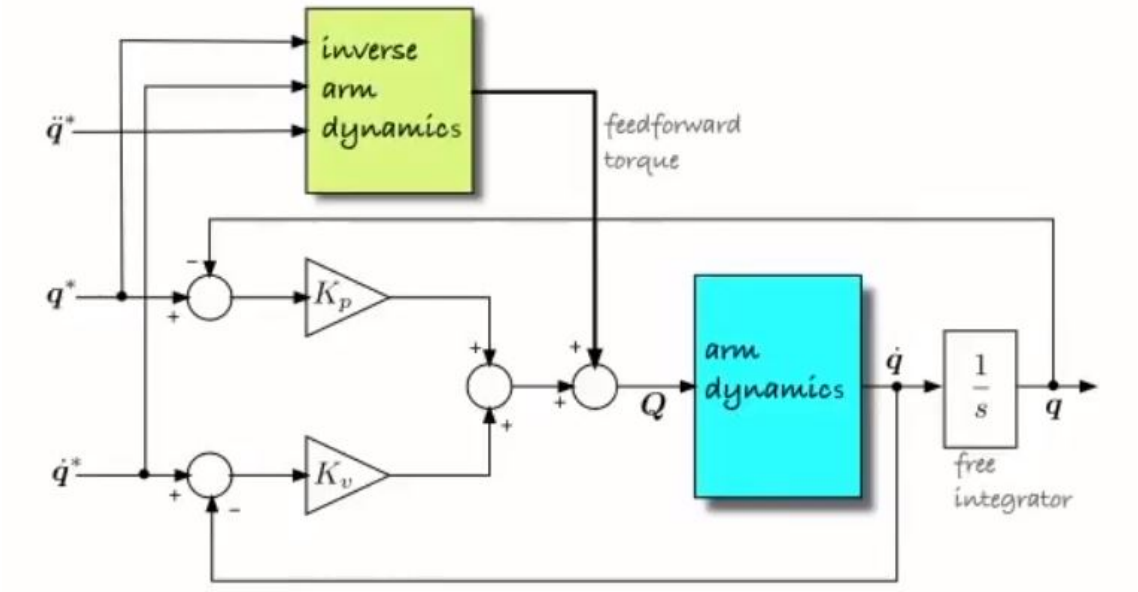

Figure 2.1: Motor Torque Control System

Thus the torque provided at each link can be expressed as:

$\tau = K_p(q^* - q) + K_v(\dot{q^*} - \dot{q}) + M(q)\ddot{q^*} + C_j(q, \dot{q})\dot{q} + g_j(q)$

Where the change in angle required by each joint when moving the end-effector from one coordinate to another can be computed using the variation of the 'ikine' function for the given robotic arm; the forward kinematics can be computed similarly using the 'fkine' function.

2.1.2 Robotic Arm Models

Separate to the custom designed toolbox of Professor Corke, MATLAB has its own robot manipulator modelling sub-program and alternative methods of control. Simscape allows the creation of physical systems within MATLAB's simulation tool Simulink. An example of 6 DOF robotic arm is presented below; the base of the arm represents the origin, and the lines span out from this with rigid bodies followed by links. Each rigid body can be shaped within MATLAB or imported from a CAD model to produce a replica of the real robotic arm; where properties such as material and interia can be defined.

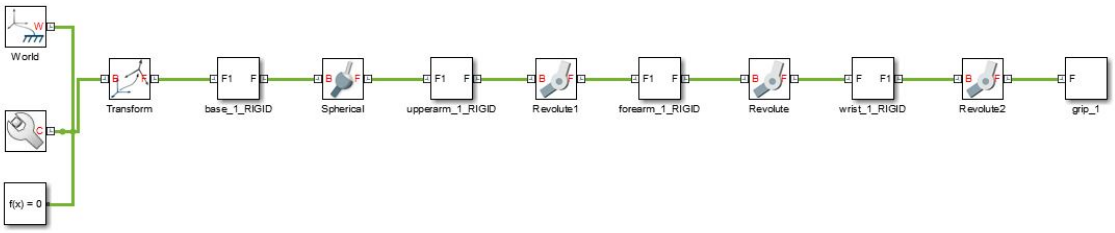

Figure 2.2: smrobot Simscape Model

Simulating the model shown above will provide a visualisation of the arms and any motion set on it by external forces or motor torques; not included in the above model.

From here, it is a matter of controlling the robotic arm to move from one joint configuration to another. For a custom robotic arm, any of the strategies outlined in the theory section of this report can be used, however MATLAB provides another means which is described in the section below. In order to move the joints from one position to the next, a strategy similar to that shown in Figure 2.1 is required. In simulink, the layer of controlling the motor position can be formatted as follows:

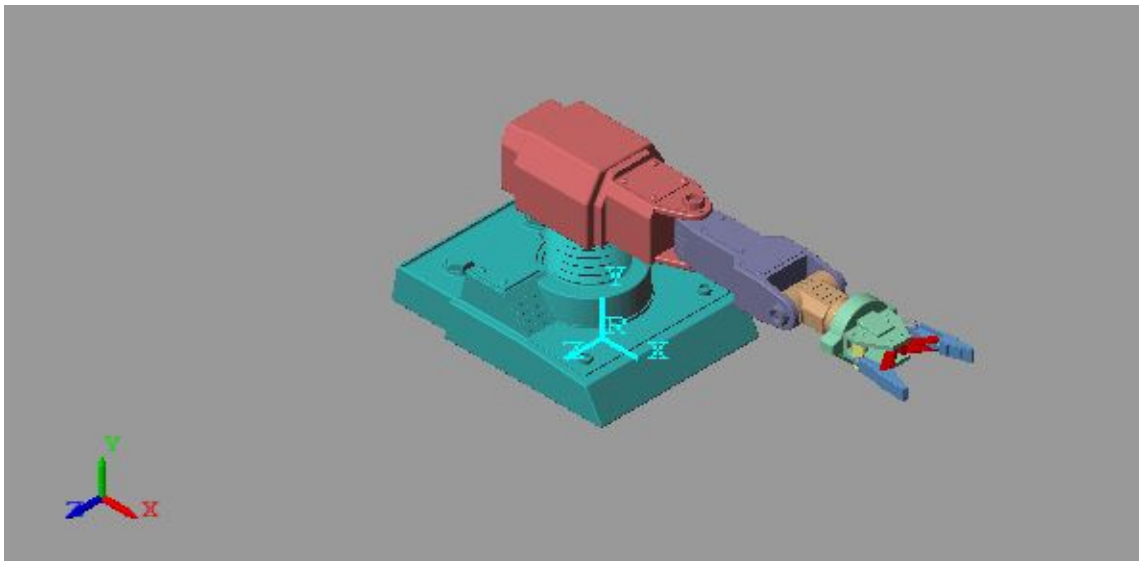

Figure 2.3: smrobot Simscape Model Visualisation

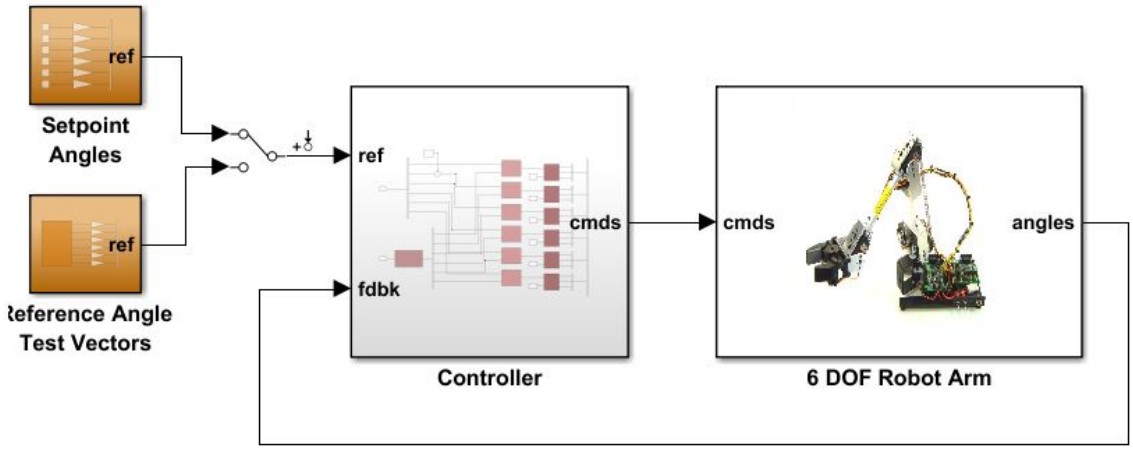

Figure 2.4: Simulink Model of Motor Control Loop

Where nested within the controller is the tuned PID controllers for each motor which drives it to the desired position. This is similar to Peter Corke's method but without pre-emptive knowledge of the disturbance torque due to the arm's dynamic coupling.

It will be seen that VREP operates in the same way, as it enables the importing of manipulator models and setting of each joint's controller coefficients. The figure above gives a good representation of the mechanics behind what the parameters are doing in terms of regulating each joint's error to a desired angle. A significant

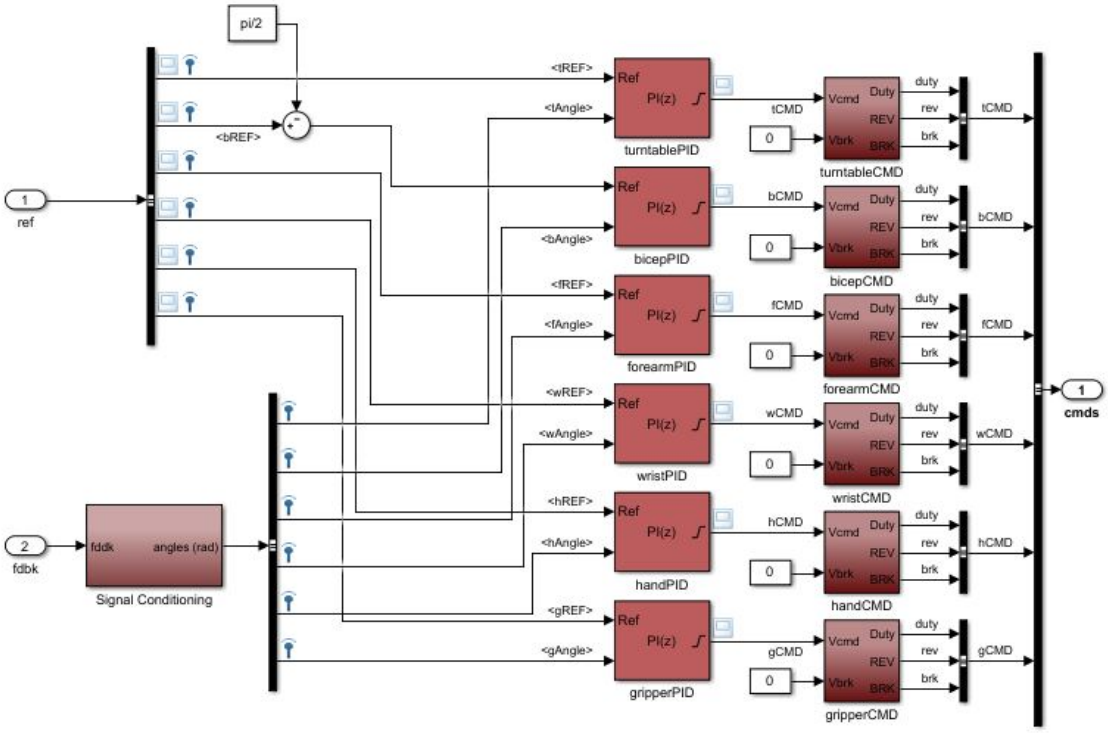

Figure 2.5: Simulink Model of Motor Controllers

advantage of MATLAB is the auto-tuning functionality of the PID controllers for each joint, however a controller on it's own would not be as robust as that described in the Robotics Toolbox section.

2.1.3 Adaptive Neuro-Fuzzy Inference System

An Adaptive Neuro-Fuzzy Inference System (ANFIS) network can be used to deduce the joint angles required for an arm's end-effector to reach a desired 3-D position. Calculating the required angles analytically for an arm of greater than 4 DOF can become challenging and result in infinitely many solutions. Like any neural network, with a sufficiently large dataset the internal weights will learn what angles will result in the desired end-effector position. Thus, in training an ANFIS to control each joint for a training dataset, it is able to interpolate angles to sufficient accuracy and move the joints to reach a desired end-effector position.

To train the network, input to output mapping data is required. Thus, given the forward kinematics for a robotic manipulator is known, a dataset of angles to end-effector position can be created over the space that it is desired for the end-effector to move. After the network gains its own understanding of the mapping, any point within the testing region can be given and the network return the required joint angles. In a MATLAB example, each joint is given its own network to be trained such that for each joint an angle will be provided given a desired position. Outside

the range of training, the manipulator will not respond promisingly, and thus for a large control range, a sufficiently large dataset is required. Overall neural networks respond quickly to providing results once trained, however the training process can be time consuming.

2.2 Programmed Robotic Arms

Volume 58 of the 2012 International Journal of Computer Applications, titled Software Development for an Inverse Kinematics of Seven-Degrees of Freedon Newly Designed Articulated Inspection Robot, details on previous software development tools used to control robotic manipulators. It is detailed that the inverse kinematics can be solved using an Adaptive Neuro Fuzzy Network (ANFIS), a neural network trained to reduce error in end effector position, non-linear optimization methods, geometric approaches, and through the manipulation of the transformation matrices. In the ANFIS approach, it was noted that MATLAB was used, however C / C++ programming has been used to develop programs that compute the more mathematics based inverse kinematics methods. A program is proposed in this paper which uses Visual-Basic to allow the calculation of forward and inverse kinematics, and trajectory planning.

A thesis paper from the University of South Florida comments on the fact that programming in a low level language such as C is closer to the hardware level which reduces computation time in processing and communicating with external systems (sensors and motors). High-level languages, such as MATLAB, provide an user-friendly interface for solving complex maths operations, however lack capabilities found in C / C++ such has multi-threading and easy communication between concurrently running processes. It is explained that the complexity in configuring MATLAB to communicate with a separate C++ program lead to failure of the wheel-chair mounted robotic arm's program on multiple occasions. The software system in this paper refers to using a C++ program used to communicate with sensors and other hardware, whilst using MATLAB for the control algorithm computations, and simulation. Due to the aforementioned advantages of C++, a new C++ program, with an external library for matrix manipulation, was created to replace the MATLAB program. The concluding results being that the new software system reduced delays in the control of the robotic arm, leading to better stability and control.
The primary purpose of using MATLAB in this experiment was to simulate a virtual model of the robotic arm, and using the angles of this to control the physical robotic arm through the C++ platform. The complexity and delay produced by MATLAB

lead to slow and unreliable control.

The Kinova Jaco arm's Software Development Kit (SDK) is written in C++, with the handling of communication and calculations stored in an human-unreadable dynamic link library. There is no documentation into how the SDK calculates the joint angles required for a desired gripper position, however there are usage examples in C++ of all the available functions; one of which is used to setting the gripper position and angle in cartesian coordinates. This SDK allows direct tuning of crucial arm parameters such as maximum applied torque, PID coefficients and velocity control. Predefined structures can be filled with this information and sent to the arm's FIFO buffer to interpret and act accordingly, making the control of the arm from the developer's perspective effortless.
Controlling the robotic arm in this way however, means the arm needs to be completely set-up in order to use the SDK. It may also be hard to understand how the parameters affect the movement of the arm; for example, the coordinate system isn't well defined and will take some trial and error to understand. This could be done with the help of the 'get' functions, which can return torque, velocity and position values of the arms current configuration. To further understand how the settings can affect the arm, Kinova have supplied the 'Development Center' with the SDK (which back-end C++ code is accessible, and can provide knowledge into how the arm can be controlled) what allows the user to move the arm, set trajectories, and review current parameters.

The Robot Operating System (ROS) is a modular collection of commonly used software in low-level device control, hardware abstraction and functionality seen in robotics. ROS can be used in Python, C++ and Lisp programming languages to build a framework around robotic software kits that directly control a robotic actuator. Thus, it has already been used as a cover of the Kinova SDK for the Jaco arm. Further, simulation platforms such as Gazebo and MoveIt can be controlled using ROS, with the software for controlling a Jaco arm in Gazebo being available. Together, ROS can be used to control a virtual model of the Jaco arm in Gazebo, as well as the physical arm through the Kinova SDK. Control of robotic manipulators in VREP is also available using ROS through the VREP remote API.
ROS offers a package called Rvis, which provides a method of interacting with imported models in 3D. Similar to a robotics simulation package, Rvis can provide information about the centre of mass of objects, collisions, and joints of robotic manipulators and objects in a scene. It however lacks the real-world modelling that comes with simulation programs such as V-REP or Gazebo (running a physics engine). Thus, it is typically used as a wrapper for these programs, such that the user

can retrieve key information about objects in a scene whilst simulating properties such as gravity and friction.

In Professor Peter Corke's Robot Academy series, he goes through an intuitive explanation for programming robotic arms using the arm's Jacobian matrix (referenced in Theory). As the Jacobian is a function of joint angles, it only provides a truthful reflection of the small change in angles required for a small change in end effector position, local to the current arm configuration. Thus, the Jacobian matrix can give a solution for small angle changes, but after the arm moves to this new position, it must be recalculated. This can be done in programming using discrete time steps as outlined in the figure below. Unfortunately, as this is an approximation technique, if any of the spacial or time parameters are too large, the result will no longer hold, and thus it should be used with caution.

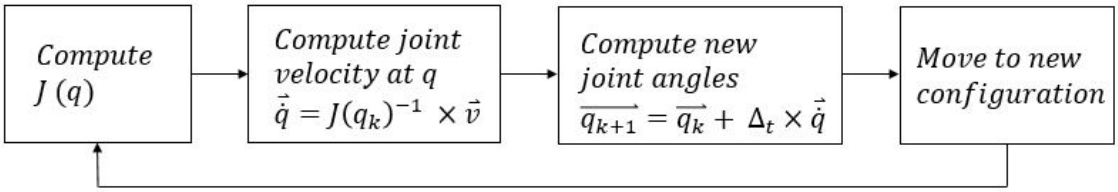

Figure 2.6: Flow Diagram of using Jacobian in Programming

A similar technique that can reduce the error between the end effector's position and a small change in position is the PID controller. With the appropriate weightings for each joint, the joints can be moved to reduce the error in end effector position. Note that there is no control of the joint-space in this case, and could lead to undesired joint configurations. This gives the Jacobian an advantage, as it is non-invertible at a joint singularity; when the arm looses a degree of freedom.

2.3 Simulation Platforms and Physics Engines

There are numerous platforms available for simulating the dynamics and interactions of robot's with a scene. V-REP is particularly useful for this thesis due to its ability to remotely control the simulation and the objects within, whilst providing a realistic environment for said interactions. Other popular robotic platforms have similar functionality, but may come at a cost, include Webots, Robot Studio, ARGoS, labVIEW, Visual Components, Virtual Robotics Toolkit, Gazebo, Actin Simulation and Workspace. Although the objective of these platforms is to simulate a robotic manipulator in a scene, they are customized in the robots that can be

simulated, the physics engine they use to simulate the dynamics, and the requirements by the users using the product. A good example of this is the Robot Virtual Worlds platform, which has comprehensive capabilities in simulating and programming LEGO robotics. On the contary, Workspace is used for simulating industrial production lines with multiple robotic manipulators acting together; and is used by large companies such as ABB and Mitsubishi. The scope of this thesis focuses on the use of free simulators that provide realistic and programmable simulations for educational use. Thus, simulation platforms such as Webots, Gazebo and V-REP are of particular interest as they equally provide the required features to meet the objectives.

An online survey was conducted by researchers Serena Ivaldi, Vincent Padois and Franceso Nori to determine which simulation platform and dynamics solver was most preferred. The survey was completed by 119 participants with 62% possessing a PhD, and of the remaining, 32% had a university level degree. It was found that 39% of the participants hadn't heard of V-REP, compared to 27% and 15% for Webots and Gazebo respectively. However, it was found that V-REP was the best rated for usability, documentation and set-up guides, whereas Gazebo was the most used. From the findings, it was noted that V-REP was the most likely for applicants to try, and keep using, compared to Webots and Gazebo.

A full comparison between Gazebo and V-REP was conducted by Lucas Noguira from the Universidade de Campinas. This article goes through implementing features common to both platforms and the process required for setting them up to do a given task; i.e. connect a sensor to a robot and read it through ROS. It was found that V-REP offered a much better interface for adding components, and used less computation time when compared.

2.3.1 V-REP

The robotics simulation platform used in this thesis is the Virtual Robotics Experimental Platform (V-REP) by Coppelia Robotics. V-REP offers a means of realistically simulating mobile robotics and actuators with an easy to use interface. In a simulation, V-REP continuously reads an internal 'main' Lua script, that may read other scripts describing properties associated with robotics in the scene, which are customisable by the user. On the side pane is a large list of available robots that can be imported, and once they are, a script customised by Coppelia is loaded to the list of running scripts; i.e. in the case of the Jaco arm, a script is imported that moves the arm around in the scene to a pre-configured trajectory. The simulation process is described by the figure below, where it can be seen that the main flow of code goes between actuation, reaction, sensing and control, with monitoring

collisions being the main priority at each stage.

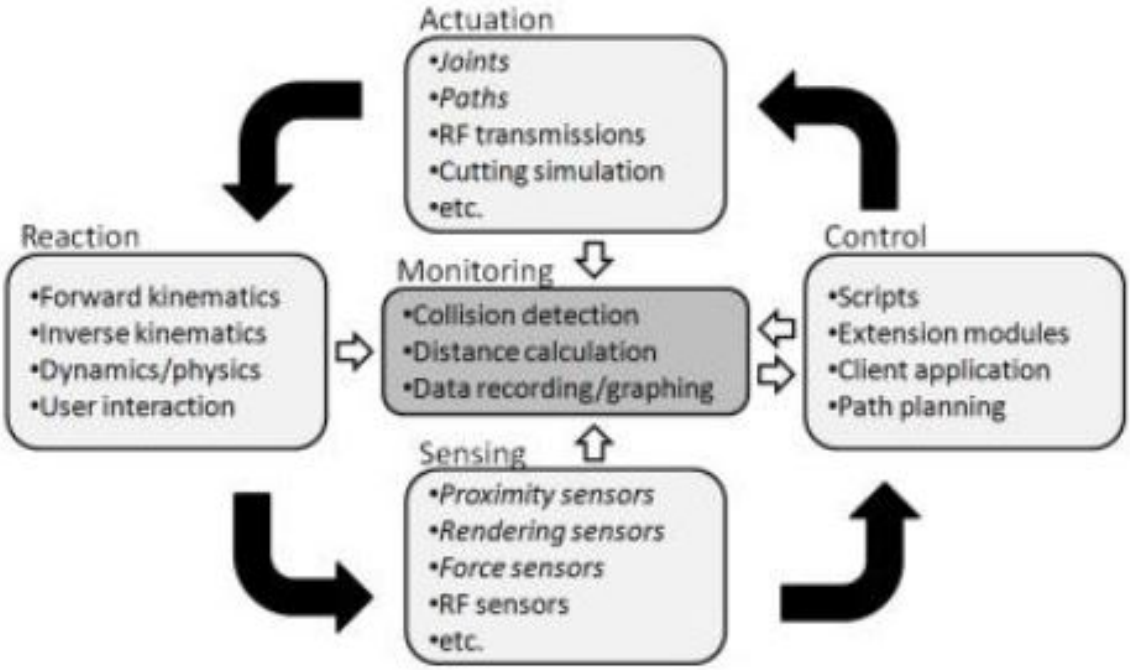

Figure 2.7: Flow Diagram of the V-REP Simulation

Communication

Fortunately, the parameters within these scripts and objects within the scene can be controlled using the remote API, which is available for C/C++, Python, Java, Matlab, Octava, Lua and Urbi. The documentation for the remote API contains extensive descriptions of over 100 functions, where the usage is available for all supported languages. Additionally to the remote API and internal Lua scripts, there are 4 other methods of controlling a scene in V-REP. The most obvious is plugins, which can provide custom Lua commands for interacting with the internal Lua scripts, it is effectively an external library of functions. Similarly, ROS can provide a wrapper of V-REP, enabling control from any ROS supported coding language, and further can enable the control of an external robot through a ROS node, using the information from V-REP; which is the aim of this thesis to a large extent. ROS. There is no limitations to utilizing V-REP's capabilities between using a remote API client and a ROS node, as seen in the connection methods table on the V-REP website; both will contain lag due to communication over the internet through a socket connection. Other methods of controlling V-REP include through the use of an add-on, or a BlueZero node.

Communication using the remote API is through a socket connection between the client API and VREP. To enable connection, V-REP needs to be configured to allow external connections through a specified port. As scripts are associated with objects within the scene, this configuration is typically set in a threaded child script of an invisible dummy object in the V-REP scene; an example of this can be seen below.

```
threadFunction=function()
end


simSetThreadSwitchTiming(2) -- Default timing for automatic thread switching

simExtRemoteApiStart(19999) -- port of which remote API server will run

-- Here we execute the regular thread code:
res,err=pcall(threadFunction)
if not res then
    simAddStatusbarMessage('lua runtime error: '..err)
end
```

Figure 2.8: Sample Lua Code for Connecting V-REP to a port

On the client side, connecting to V-REP waits for a given length of time for a client identification number. Once this number is obtained, the number is used for all communication thereafter in a manner noted by Coppelia for the language chosen. As an example, the functions could return parameters such as joint angles, velocity and sensor outputs, alternatively the functions could also be used to set or change these parameters.

Communication between the client program and V-REP is undertaken in four different modes; blocking function calls, non-blocking function calls, data streaming and synchronous operation.

To be able to interact with a particular object in V-REP, one must refer to it in function calls using its ‘object handle‘. The object handle is a unique number given to every object in a scene, typically in order of when the object was added to the scene, known by its appearance in the scene hierachy; a list containing all of the objects in the scene, starting with objects like the default camera, floor, lights, XYZ coordinate frame, and then any objects that were placed in the scene after. Once an object has been deleted from the middle of the hierachy, the other objects retain their object handle; thus once an item is added its object handle won’t change. When calling a missing object in a scene, NULL will be returned.

Operation Mode	Summary
Blocking	When a command is sent in blocking mode, the client will wait until the V-REP API response before moving to the next line of code.
Non-blocking	When data needs to be updated in the client program, that isn't required straight away. The client can make a call for the data to be copied to a particular address and move on to the next line. The API will respond in due time.
Data Streaming	When the same data is constantly required from V-REP, the data-streaming method allows a way for V-REP to send updates of a particular value without calling for it each time. For instance, joint angles of a robotic arm may be required by the client program with large frequency, and could be updated without making direct calls and waiting for a response each time once a data-streaming call is given. This is a continuous mode, and the client will be stuck receiving data.
Synchronous	When it is desired to have the simulation and client program run concurrently in time, such that the simulation advances with the client. Synchronous mode allows the client to control the advancement of the simulation.

Table 2.1: V-REP remote API operation modes

V-REP Features

2.3.2 Alternative Simulation Tools

Webots

Webots is a robotics platform that allows the simulation of popular robot models, that can be interacted with using external software scripts, of which can be then programmed onto real robots; given they are compatible. Similar to V-REP, one can communicate with the simulation using TCP/IP from a program written in C/C++, MAtlab, Python or Java. The simulation also utilises Open Dynamics Engine (ODE) to give a realistic representation of physical interactions between objects. Motors and sensors can be placed in the scene or within robots, and tuned to mimic their real-time capabilities; maximum torque, PI control of servos, and sampling of sensors to name a few parameters that can be varied. There is little information on the comparison between this platform and V-REP, but from the

information found on Webots and in the survey outlined above, it appears to have equivalent functionality, if not less, as it relies only on one physics engine whereas V-REP offers four. Webots could be used for this thesis, as it provides a means of real-time (or even speed-up) interaction with a scene and the objects within. The use of ODE is no downfall, as outlined in the ODE section, it provides a realistic rendering of collisions within a scene.

Gazebo

Gazebo offers a very similar platform to V-REP, where a model can be placed in a scene and have the simulation controlled by a remote API. As an open source platform, full integration with ROS has been created, where the internal parameters of a simulation can be controlled with ROS. It is offered on any OS with a command line interface to help control simulations. Simulations are saved to an .xml script, which can be edited, and rerun; a feature V-REP lacks with it's .ttt saving format. A thorough comparison was made between Gazebo, V-REP and ARGoS, which demonstrates that they both contain the key features required for this thesis. It can be seen however, that V-REP has more positive features when compared to the other too. Gazebo on start-up is build with ODE, however can be built from source with any physics engine.

Bullet Physics Library

Open Dynamics Engine

Vortex Dynamics

Newton Dynamics

Chapter 3

Theory

3.0.1 Equations of Motion

A scientific paper is likely to be read by people who are not specialists in the same field as the author(s), but who nevertheless may need to use the results of the paper in their own fields. Similarly, the examiners of your thesis will probably include at least one academic who does not teach or conduct research in the subject area of your thesis. In an early chapter of your thesis, therefore, you should quote any theoretical results which are necessary for the understanding of later chapters. Examiners who are not specialists in your area will know whether you have given sufficient theoretical information. They will also know whether you have insulted their status by presenting material which is familiar to every half-competent graduate in every field of ECE.

Chapter 4

Methodology, procedure, design, etc.

This may be one chapter or several. Again, titles should be more informative than the above.

You will almost certainly need diagrams to clarify your meaning. The LaTeX 2_ε `graphics` package allows the inclusion of PostScript graphics, as in Fig. **??**. The inclusion of LaTeX `picture` graphics, as in Fig. **??**, requires no auxiliary packages and allows the mathematical formatting features of LaTeX to be used in diagrams; but the `picture` files, unlike PostScript files, usually require manual editing.

Chapter 5

Results and discussion ...

... or perhaps the discussion should be a separate chapter.

In any case, you will probably need to include tabulated results. Table 5.1 illustrates the use of various LaTeX environments to include a computer printout (plain text file) in a document. The `verbatim` environment, which encloses the formatted text, is also useful for program listings.

Table 5.1: *Fraction of air volume involved in heat exchange for second mode (right column) vs. filling factor (left column). The plain-text headings represent f, m, μ_2 and f_2.*

f(%)	m	mu2	f2(%)
0.016	80.00	0.05400	4.874
0.031	56.57	0.07732	5.438
0.062	40.00	0.11103	6.125
0.125	28.28	0.16001	6.970
0.250	20.00	0.23175	8.020
0.500	14.14	0.33799	9.329
1.000	10.00	0.49789	10.967
2.000	7.07	0.74444	13.008
4.000	5.00	1.13919	15.525
8.000	3.54	1.81095	18.568
19.237	2.28	3.61958	23.174
37.180	1.64	7.28635	27.094
57.392	1.32	14.63631	29.813
74.316	1.16	29.35160	31.453
85.734	1.08	58.79364	32.360

Chapter 6

Conclusions

6.1 Summary and conclusions

6.2 Possible future work

Appendix A

Dummy appendix

Appendices are useful for supplying necessary details or explanations which do not seem to fit into the main text, perhaps because they are too long and would distract the reader from the central argument. Appendices are also used for program listings.

Notice that appendices are "numbered" with capital letters, not numerals. When the `\appendix` command in LaTeX [1, p. 175] is used with the `book` document class, it causes subsequent chapters to be treated as appendices.

Appendix B

Program listings

B.1 First program

Some initial explanatory notes may precede the listing.

B.2 Second program

B.3 Etc.

Appendix C

Companion disk

If you wish to make some computer files available to your examiners, you can list and describe the files here. The files can be supplied on a disk and inserted in a pocket fixed to the inside back cover.

The disk will not be needed if you can specify a URL from which the files can be downloaded.

Bibliography

[1] L. Lamport, *LaTeX: A Document Preparation System*, 2nd ed. (Addison-Wesley, 1994).

[2] REFERENCE 2

[3] Etc.