Bachelor of Engineering Thesis Project Proposal

Callum Rohweder

Connecting Virtual Robotics to an Experimental Platform

August 24th, 2017

METR4901

Under supervision of Dr. Surya Singh

Faculty of Engineering, Architecture and Information Technology
The University of Queensland

Project Summary

V-REP is a simulation platform which models' robot actuators and processes, providing realistic visualisations of object interference and dynamics. This program can be accessed externally via its remote API functionality, which allows a client to send commands to a simulation scene. There are over 100 of these commands, with some that manipulate the simulation through actuating a robotic model or by requesting kinematic or scene related information from V-REP. With this, it is proposed that the client user has the capability of replicating a physical robot's movements and interactions with simulation, and vice versa.

This could be the first step in a range of different advances around remote interactions using a robotic arm. For instance, users will be able to see, through simulation, their robotic arm working on a process, and provide modifications in real time; like assisting in grasping objects. Remote interaction using a robotic arm simulation, controlling a physical arm remotely, extends being distance from something from being seen on a screen to being interactive, 3D; given the environment of which the robot is in, is replicated in the scene. Advancements could improve distance education, allowing teachers to interact with their student's work and point things out. Or, allow simple laboratory work to be completed whilst being out of the lab.

For this project, a Kinova Jaco arm and simulation model in V-REP will be interfaced such that they are made to mimic each other's movements in real time, remotely. Progress in this could lead to manipulation tasks, where a client user clicks on an object in the scene, and both the virtual and real Jaco arms move and grasp the object. If the opportunity comes to advance this interface further, a video could be set up in the laboratory or which the client user can remotely click on an object, which translates to an object in the scene, thus actuating both arms to grasp the object.

To achieve this task, an interface will be executed on the physical arm end, which remotely connects to the V-REP API client for the user to see and interact. The Jaco arm's API will be configured to the software running computer, allowing direct access to via a USB-2.0 cable.

The greatest difficulties to overcome in this project will be in providing a reliable interface, whilst aiming to achieve real time replication between the physical and virtual models. This will depend mainly on the networks being used, however is also dependent on the ease of retrieving data from the physical arm. Although Kinova offers a software development package to assist in retrieving information from and actuating the arm, it is not yet known the simplicity of this and response time; it is the same API that the company uses to turn joystick operation into arm manipulation in disabled person's use, thus it is expected to be particularly responsive.

No catastrophic risks were identified, being a simulation based project. The most hazardous being data corruption or loss, of which there is a process in place to back up progress in both the cloud and on a local system. Other risks include loss of network connection or access to the Jaco arm, these are however only temporary and can be mitigated using a virtual machine and working on simulation interface tasks respectively.

Table of Contents

Introduction

From the first robotic arms of Da Vinci's mechanical knight in 1495 to the recent developments in remote robotic surgery, people have been trying to replicate human actions with autonomous systems for centuries. The current norm is that virtual communication only extends to two dimensional interaction through video, where someone can merely oversee an activity or speak to a colleague. Simulation environments which remotely control actuators, like that of a typical robotic arm, have the potential to allow the interaction on a physical level. This paper looks into the benefits this can possess and the current advancements. Additionally, it is proposed that using V-REP's remote API, a simulated environment of a test laboratory with a Jaco Kinova arm can be generated, and movement replicated in the physical space. Thus, allowing physical interaction occur for real-time or predefined processes.

Background Advancements

Manufacturing and Process Lines

Simulating a process before implementation in the manufacturing industry has led to significant reductions in failure and cost. With the help of 3D CAD models, manufacturers of large capital equipment are able to test a procedure and optimize accordingly. Thus, it is possible to view the limitations of their systems before an error occurs, and design accordingly [1]. From a manufacturing prospective, it allows the (physical) machining of materials to be done in such a way that excess material and coolant flow be controlled, and reduced, with precision beyond human capabilities.
As with the size of the process industry, there are many simulation programs which allow plants and robotics be modelled and planned; these programs can even allow for the modelling of human manipulation of a plant. An advantageous programming software package for this is Siemens PLM Software [2], however it possesses limitations. These are with respect to the limited functionality associated with human manipulation and the program does not allow for remote, real-time control. If it did however, and there was real time replication between physical and simulation, an operator would be able to define or edit a process in real-time by recording the movements of the physical arm, thus correcting for scaling and mechanical differences between simulation and reality.

Remote Surgery

Hospitals are full of strict processes and guidelines of which nurses and doctors must follow. This poses as a major opportunity for automation, where robotic arms can complete tasks to a high degree of accuracy, thus allowing nurses to participate more timely on complex interactions. Similarly, compared to manufacturing, the scale of the number of processes in a hospital means different objectives are being obtained through automation, and thus require different techniques.
In 1995 robotic arms were designed to prepare medication and pharmaceutical products within a hospitals [3]. This method proved quicker than human preparation as the processes and product locations were predefined. Following to now, it is possible to remotely perform surgery, robotic arms are able to perform open surgery, pre-operative planning can take place using patient images, and training and pre-operative warmups can occur with the help of remote simulations [4]. Robotic actuators can have real time tool changes, steady movement and the accuracy required to perform surgery related tasks; in prototypes, such as 'the surgical robotic cell' [5] , and in practice with the use of the surgeon assisted da Vinci Surgical System which allows remote control surgery to take place [6]. This can too include simple tasks such as retrieving equipment for a surgeon, thus freeing up nurses time to engage in more intellectual tasks around the hospital.
The da Vinci Surgical System is capable of remote telesurgery [7], where the surgeon (host) can be in a different location to the patient (client); it was used in close to half a million operations in the U.S. during 2015 [8].

Event Based Control

Tele-control refers to the communication between two systems, ran by different processors, of which the data could be transported over the internet. Such communication has two draw-backs which can impinge on real-time remote control of a robotic arm, data loss and time-variable time delay. Data loss can be accounted for by ensuring there is a reliable internet protocol in place, i.e. the Transmission Control Protocol (TCP) which checks that data packets have been transmitted. Time-variable time delay (TVTD), however is non-linear as the host and client networks are operating on systems of differing speeds. Event based control theory has proven to reduce TVTD, and it outlined in the figure below from [9].

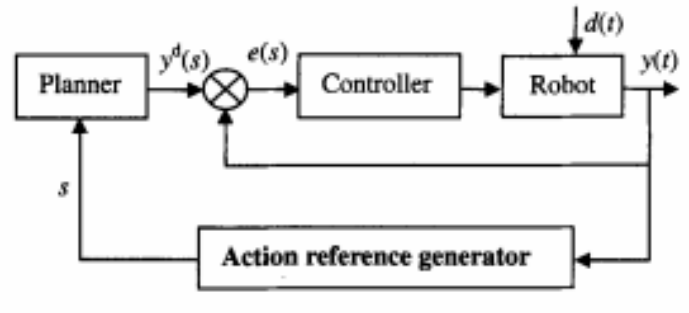

Figure 1: Event based control block – [9]

In this figure, 's' denotes the true state of the robotic actuator, after a controller manipulates it to a goal state. Due to the time delay, without the feedback reference state which changes the desired input, uncertainty and unexpected results of the actuator cannot be corrected for by a change of plan. This block diagram outlines that if a desired state is sent, a controller can be used to correct for disturbances and achieve the goal actuation position, but the result of this action is required to check that if the final state is that of which is desired. In this way, asynchronous, or multiple, commands can be sent to a robotic arm remotely, and the actuator's end state changes the next desired state.

Remote Application Manipulation

Remote application programming interfaces (API) allow the manipulation of a program from a different location / system. This is typically done through a socket connection, which allows functions to access parts and features of an application; an example of an API is, google maps planted on a webpage (external to google). A distinct benefit of remote API's from a programmer's prospective is that the client platform can be built by any language, as long as the data is readable by the host machine; the host is the system of the API engine, where the application runs, and the client is the program sending commands. A noticeable drawback with remote API's comes from the client different network configurations, and the permissions the operating system's firewall or network adapter has with regard to retrieving information from a network of different configuration.

A Brief on Robotic Simulators – V-REP and Gazebo

There is a strong demand for dynamic simulators, for the aforementioned tasks and more. These simulators are more complex than video-game platforms as they need to adhere more strongly to the laws of physics, especially when collisions or human manipulation is included. It is for this reason that simulators come with a variety of dynamics / physics engine add-ons; such as ODE and Bullet. Not all platforms are open-source, and source code or documentation not easily accessible, thus it was for this reason that a comparison between simulators is dependent primarily on user feedback [10]. Through a survey of 119 participants, cited by [10], it was concluded that 39% of them hadn't heard of V-REP compared to that of 15% for Gazebo and 10% for ODE, yet 5% of participants current use V-REP compared to 13% and 11% for Gazebo and ODE respectively. In addition to this, it was found

that V-REP was most preferable for research use. Gazebo is similar to V-REP as it is a multi-robot simulator with built-in support of ODE and Bullet dynamic engines, however, it's used to model outdoor environments. Thus, the survey concluded that Gazebo was the most used simulation platform, however V-REP was the most preferred by the users who had tried it, and was rated higher for its extensive tutorials, support and documentation. Similarly to these findings, a comparison between Gazebo and V-REP simulators was performed by [11], who concluded that V-REP was preferred due to its lower CPU demand, usability and vast external functionality, however Gazebo was found to be more heavily supportive of ROS.

Background of Project Related Material

V-REP

V-REP is a robotic simulator by Coppelia Robotics, which provides manipulation and physical simulation of all objects in a scene by embedded scripts or API engine. Below is a brief description of V-REP's extended functionality:

- There are seven different methods to externally control a scene; embedded scripts, add-ons, plug-ins, ROS node, remote API clients, ROS node and custom client/server configurations. Each of these have their own limitations in terms of delay and control [12].
- Compatible remote API languages include C/C++, Python, Java, Matlab, Octava, Lua and Urbi; each have their own libraries (100+ functions), example code and tutorials to ensure the user has the required support to connect to VREP remotely [13].
- For sending information via remote API, there are four different operation modes; blocking function calls, non-blocking calls, data streaming, and synchronous operation. Each has their own data flow that affects delay and reliability.
    - Blocking function calls – when a command is sent to the remote API server from a client, V-REP receives the command and sends back a return status. This enables the client program to identify if a problem has occurred with respect to the function call.
    - Non-blocking function calls – when a return status isn't required from the API client, and thus there are no execution delays between requesting information, or calling a function. This is useful for when the client needs to send several requests of information in one message; for example, requesting for all the joint positions on a robotic arm can happen in one transmission rather than individual transmissions.
    - Data streaming – Data can be streamed from the server for a certain length of time or continuously, thus giving the client updated information on an object within the simulation, as requested prior.
    - Synchronous operation – remote API functions are executed asynchronously during a simulation by default, thus the simulation will continue without the client on an irregular basis. Thus, the progression of the simulation can be coordinated with the client via the synchronous stepping mode. Controlling the environment in this manner can lead to complications in data reading, if the client and simulation become out-of-sync, such that the wrong data is read by the client.
- Bullet Physics Library, Open Dynamics Engine, Vortex by CM Labs, and Newton Dyamics are the four dynamic engines currently supported by V-REP. Each one has its own method of calculating dynamics to mimic the real world, and thus it may be more advantageous to use one engine over another depending on the type of task being performed. By default, Bullet Physics Library is enabled, which is similar to gaming dynamic engines, allowing 3D collision detection and rigid body dynamics. Vortex Dynamics engine however can be used to model more realistic physical interactions, offering a closer depiction of real-world parameters. It is recommended to use V-REPS built-in kinematics ability with Bullet Physics

Library, as computing purely off a physics engine for a realistic simulation can be slow and in some cases be imprecise; this would only be done if collisions need to be simulated in a realistic manner, i.e. the interactions between a gripper and object [14].

- CAD models can be imported into VREP, in the format of .dxf. Thus any object can be created and added to the scene as a collidable object. These models can then be edited to remove their internal material, or reconstruct the mesh, which beneficially reduces simulation time.
- All objects are scalable and
- Some extra features include collision detection (mesh to mesh), minimum distance between objects, inverse and forwards kinematics calculations, vision and proximity sensors, building block concept (build any system, including robotic arms), motion planning and data recording.

A detailed summary of V-REP's features and functionality is available from [15], this as well as Coppelia's documentation pages are a good source of information; and outline how to utilise these V-REP's capabilities for a given task. For an understanding of V-REP's simulation processes, below is a flow diagram, figure 2, which describes the interactions of important functionality.

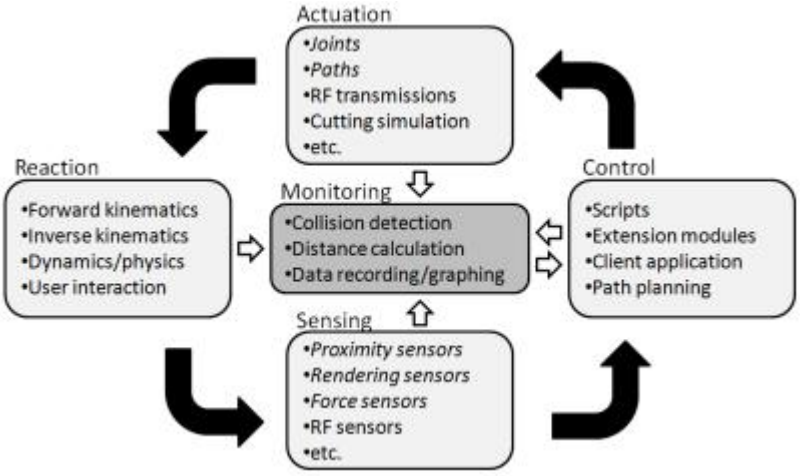

Figure 2 – V-REP's simulation flow diagram [16]

Kinova

Kinova is a Canadian based company whom develop robotic systems for personal use; more noticeably in rehabilitation, however also for research [17]. The Jaco arm is a lightweight 6 degree of freedom robotic arm designed by Kinova to assist people with upper body disabilities. Jaco^2 is carbon fibre, weighing 4.4Kg, and can be fixed to wheelchairs or almost any surface by its aluminium base [18]. The key difference between the new Jaco edition and its former is the gripper, which includes a friction pad which helps in grasping fragile objects. A Kinova joystick can be used to control the arm's translation, rotation and gripper, where the Singularity Avoidance System ensures the joints actuate in a desired manner. An additional feature is that the Jaco arm is compatible with Kinova's software development kit (SDK), of which the API is accessed through USB 2.0 [19]. This enables programmers to control the actuation and access the arm's advanced features. The SDK is compatible with Windows 7 and 8, as well as Linux Ubuntu, and a full programming guide is available for using the C++ API; with included tutorials and example code to get the programmer communicated and retrieving information from the arm [20].

From the arm's programmable documentation, it can be seen that the current position can be recorded, the arm can be made to move to a predefined position, it can rotate or translate about any of the 3D axis, and the gripper can open or close all of the gripper fingers. With the use of API functions, the arm can be made to move with respect to a Cartesian coordinate system, or by setting each joint angle

individually; this can happen similarly in reverse, where joint angles is given to the programmer. The functionality can be driven further with trajectory planning and force control; similar to V-REP. All of the information from the arm is stored in an activity log, accessible in the API directory.

Project Plan

Current Objectives

There are four main objectives associated with this project, as described in figure 3; extension objectives from these may occur, or the main objectives may change slightly, but the overall structure is defined here. The main goal is to be able to easily use the simulation to manipulate the physical laboratory environment using the Jaco arm. This is done, firstly, by constructing an interface that can manipulate a simulation environment; objective 1. Objective 1 needs to be configured in such a way that it automatically retrieves the object handles (or object names) associated with the Jaco arm and gripper, and stores them in an external file every time it is run. This allows objective 2 to retrieve such information, retrieve joint angle information from the physical Jaco arm, and pipe back joint handles (or simulation object joint names) with the corresponding angles. The program developed in objective one should be configured to receive this information and move the simulation (Jaco arm) to a position which mimics that of the (physical) Jaco arm. With enough delay reduction from objective 1, this should be close to 'real time' and allow the physical movement of the Jaco arm be mimicked in the simulation; where the scene environments are the same.

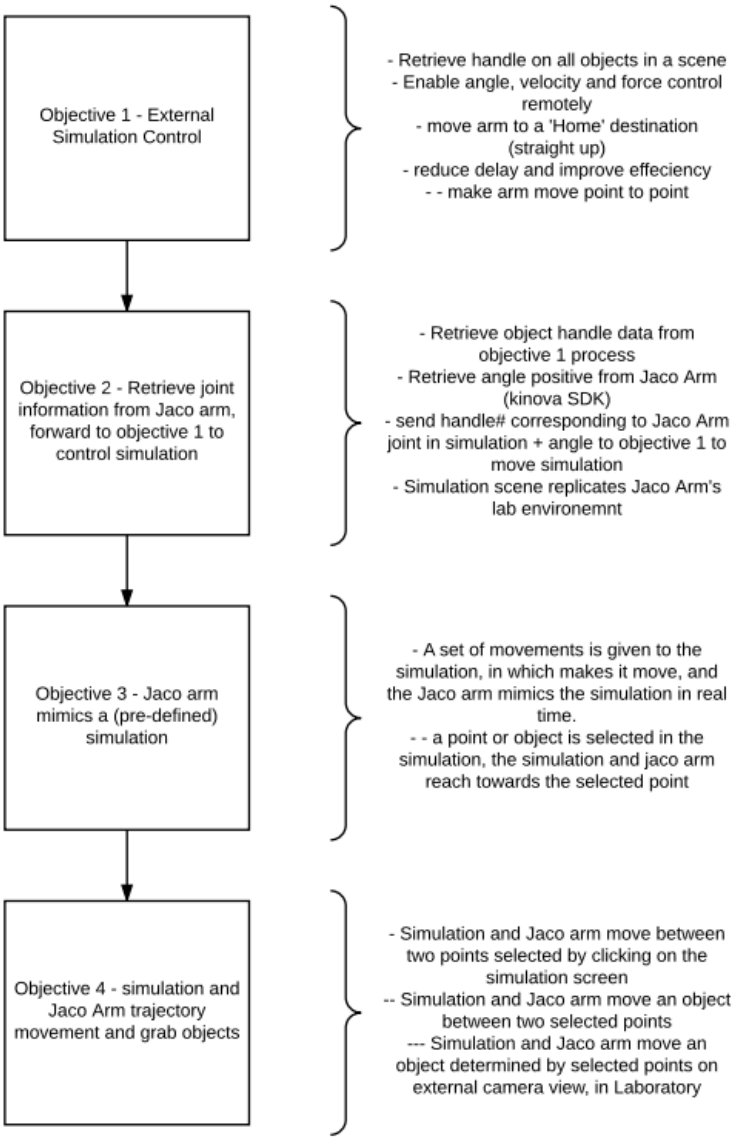

Figure 3 – List of project objectives and brief descriptions

Following from this, objective 3 is to allow the operation of objective 2 occur in reverse. This may require a script based of objective 1 in reverse be generated; such that simulation manipulation with a mouse is abled and recorded. Information from the simulation can be piped to a process which manipulates the physical arm accordingly. Objective 4 is an extension from this which enables the simulation to detect objects be selected and moved from one point to another, with the physical arm replicating the action. Furthermore, using a video of the real scene (and a camera placed in a similar manner in the simulation, allowing interface from objective 3) a point can be clicked on the video footage and the arm will appear to move to the points in both simulation and reality.

Possible Developments

Although this thesis is about creating an interface between a Jaco arm and V-REP simulator, to allow real-time movements be generated remotely, from these objectives there are further developments that could be made;

- Virtual learning and distance education can entail a more interactive experience. For instance, if a teacher wanted to point out something in a student's work, or potentially write something in front of them, they can do so by clicking on their video call window. This could enable a simulation arm platform to calculate the angles required by the appropriate to execute the task, and through remote API, a physical arm at the student end could interact with them. Alternatively, the teacher could move the simulation, which replicates the student's environment, thus allowing the physical arm to mimic the movement. This process is similar to how surgeons are able to perform tasks on patients, when they are in different locations, but instead of the surgeon's actions being recorded, a 2D to 3D mapping occurs with the help of the simulator. Thus, only the experience only requires two computers and a robotic arm; as the outcome is less complex in comparison with remote surgery of course.
- Physical laboratory work can be trialled without the physical arm to ensure the process will work. This is similar to the manufacturing industry, however could also allow real-time replication between simulation and physical environments. Thus, one could simulate a task, watch it be undertaken in the simulation knowing its working the same way in the laboratory.

Process

Objective 1

A key development to make with objective 1 is progress towards a reliable and responsive interface for retrieving information and controlling the simulated Jaco arm in V-REP. This will include testing different operating methods for the API functions; outlined in the background. As mentioned earlier, the reaction time between sending a command and it actuating in V-REP is dependent on network configuration and time delay. Therefore, in the interest of controlled testing, it is vital to perform the tests of operation mode incrementally (within the same time period), without the change of client and host networks. The more tests of each performed, the more likely a realistic outcome is obtained; thus it is estimated that each test should be performed incrementally, recording the time taken and resulting position against desired position, for 10 iterations. In this way, reliability and efficiency can be analysed, and obtained in the further objectives. A benefit for this is that an understanding of the operation modes can also be found, ensuring quick debugging in later experiments.

As an end result, the program should initially collect all the object handles corresponding to Jaco joints, and create a mapping within a file in the local directory. From this, it should read from the input to find a joint name and corresponding angle to move the simulation arm to; this using an internal mapping from C structures to determine the right object handle to use, not the created file. The generated text file can be used by a separate program designed for objective 2, which allows it to map the Jaco arm's physical joints to simulation joint names, thus proving an input into the program from objective 1. Having two programs, instead of one, allows the interface to be modular, and the simulation manipulation be separate to the physical arm manipulation; thus allowing both to be made separately and brought together. This assists with debugging and adding additional features, it also creates two process flows, and thus a faster run time.

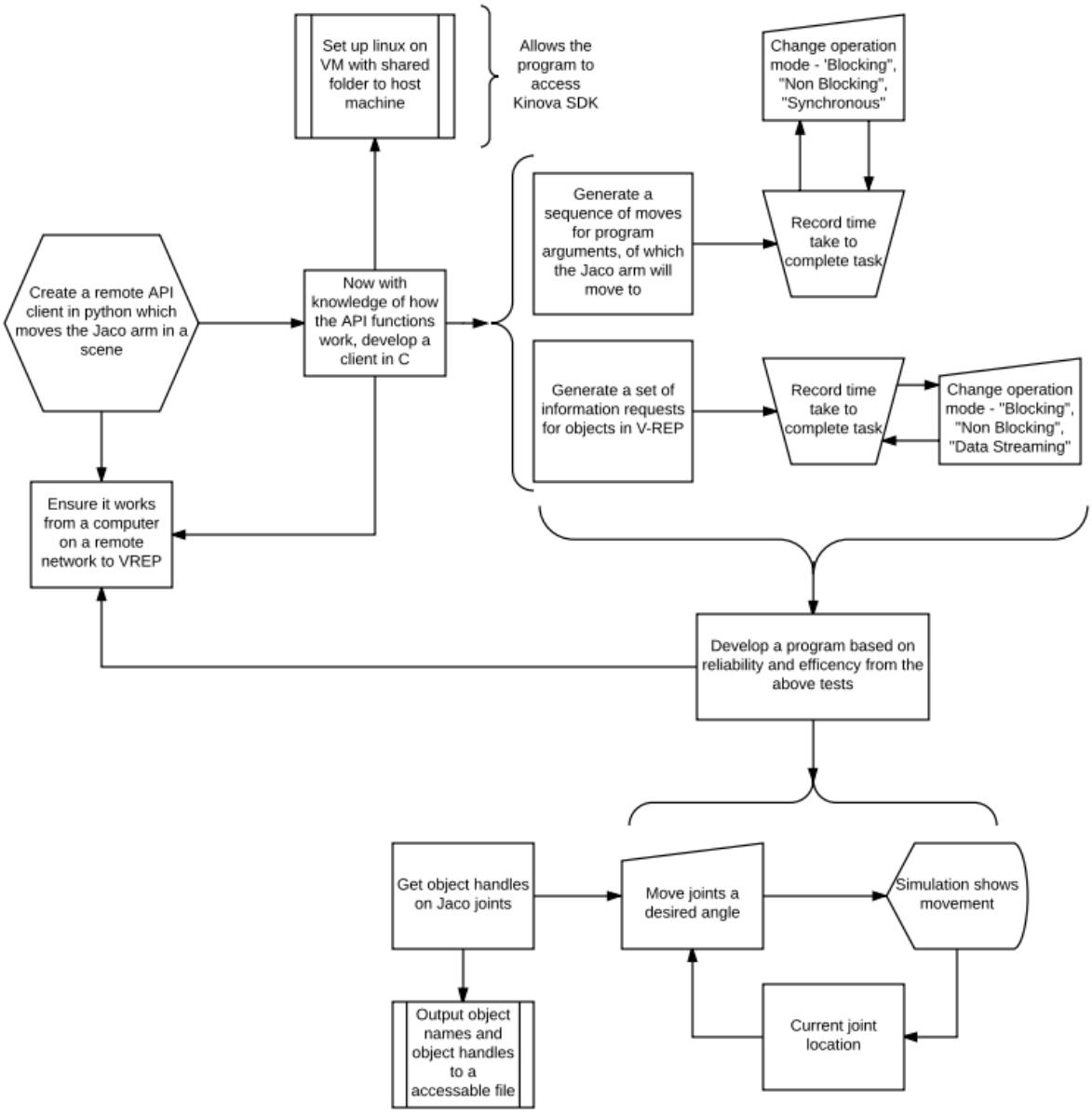

Figure 4 – Objective 1 flow diagram

Objective 2

Objective 2 includes receiving information from and manipulating the Jaco arm, utilising Kinova's SDK. A program should be created that reads the simulation object map and stores this, then retrieves joint angles and velocities from the Jaco arm. Using the map, it should be able to send out joint names and angles to the program generated from objective 1, enabling the simulation to mimic the position of the Jaco arm. The flow diagram presenting this is illustrated in figure 5.

In addition to this, the simulation scene should be made to replicate the laboratory environment the Jaco arm is placed in. This can be done predominantly from creating CAD models of the components in the laboratory and importing them into a mesh object; from a .dxf file as found in the background research. These models can be made in 3D modelling programs such as Autodesk Inventor or CREO Parametric.

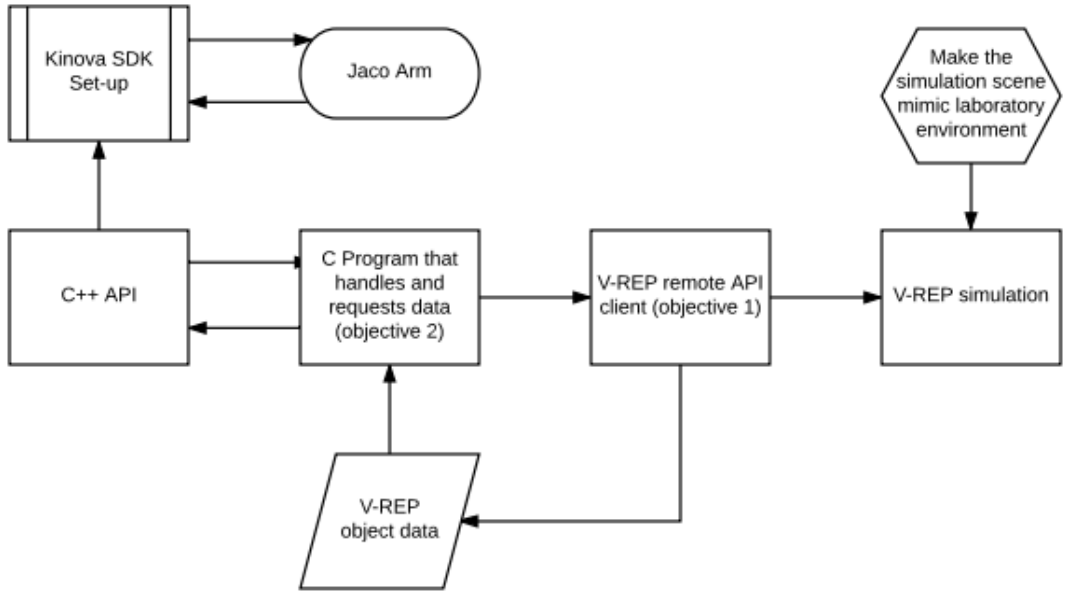

Figure 5 – Objective 2 flow diagram

Objective 3

The following objective is essentially that illustrated in figure 5, however reverse connections are made between V-REP and the API client, and to the program that handles the Jaco arm movement. Thus, information starts by retrieving the simulated Jaco arm angles and position, and ensuring the Jaco arm moves accordingly. The main outcome of this project is to obtain performance close to real-time, this is the objective that this outcome is supposed to be completed. This can be done using the relevant information from the event based control research in the background section of this report. In addition to this, a mouse click in V-REP should be translated to a set of coordinates of which both (virtual and physical) Jaco arms move to; the process is outlined in figure 6, below.

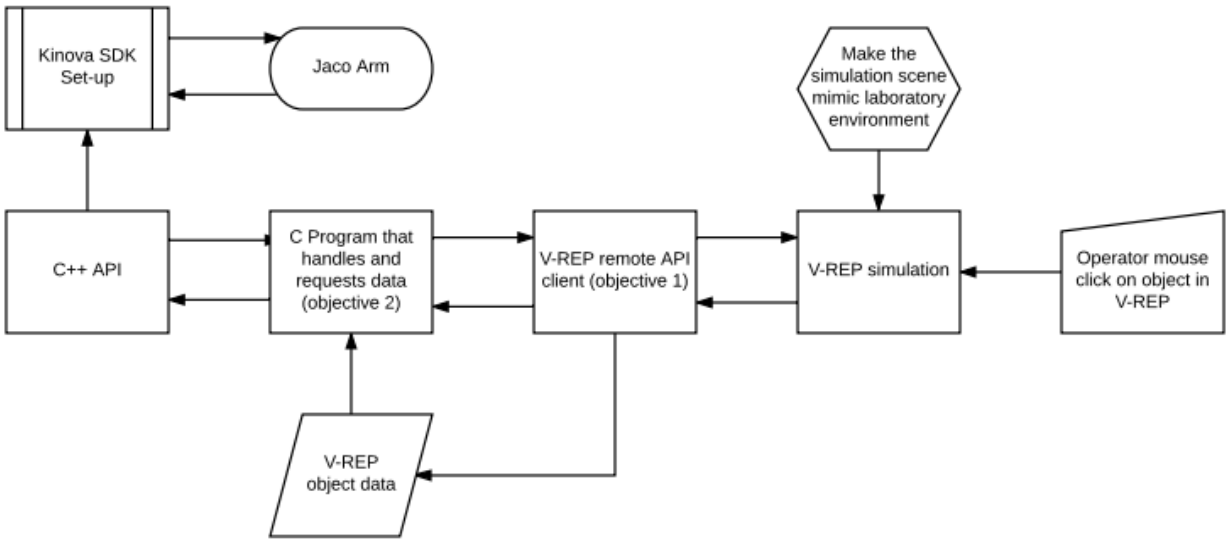

Figure 6 – Flow chart of objective 3

The path between moving the Jaco arm and the C program is dependent on how the arm is configured in the lab, however, from the documentation on Kinova's website it can be controlled as outlined above.

Objective 4

This objective aims to progress towards the potential uses outlined in the Possible Development section of this report. The process and success in this will depend on the limitations found in any of the above objectives. Given that they are successfully implemented there could be up to 4 programs used to manipulate and retrieve information from V-REP and the Jaco arm, respectively. Once time delay is reduced, the project will be in a position to build a master program that sets up the others to undertake task; grasp an object if it is clicked on, and drag it to another clicked point. The clicking may take place in simulation, V-REP, or more advanced, from video footage of the arm in the lab. This methodology undertaken to perform this task will be more properly defined after the completion of objectives 1 and 2, as it is the most complex.

File Storage

Most of the software will be written on the University of Queensland's Unix server, MOSS, accessible by students and staff. V-REP is free to download for education purposes, and the program will always be run external to the MOSS server; on a computer or laptop. In addition to this, a Linux virtual machine has been set up to allow testing on home networks without Windows Firewall or home router causing connection problems; between V-REP and the remote API client. File storage is predominantly handled using GitHub, thus allowing code to be easily backed up and accessible on any network connected device. Progress can be overseen at https://github.com/Calzatron/METR4901.

Milestones

Each objective is designed to be detrimental progress for the next objective to work, and thus they must be completed in order. The total time duration of an undergraduate thesis is almost 11 months (8 if compulsory work placement stops progress over the summer), split into two semesters of 13 weeks each. Therefore, it is desirable to make each objective equally spaced between the two semesters; each objective increases in complexity, but decreases in the amount of setup required. A layout of the project is presented in table 1.

Table 1 – Assessment deadlines

Task	Due Date (2017/18)
Project Proposal	Thu 24 Aug - 4pm
Objective 1	Thu 14 Sept
Objective 2	Thu 5 Oct
Seminar	9-13 Oct (to be scheduled)
Objective 3	Thu 5 April
Objective 4	Thu 17 May
Poster and Demonstration	Fri 25 May (to be scheduled)
Thesis Paper	Mon 11 Jun - 4pm

In addition to these assessment deadlines, weekly commitments to the project has been organised. Monday is a dedicated thesis project day, with any overflowing work being completed on Thursday mornings. This strategy allows weekly goals to be completed, ensuring progression of the aforementioned objectives and assessment.

Risk Assessment

The following risk assessment of the project is based off the 'tolerable risk matrix', shown in figure 7 below. Table 2 contains the risk assessment, where it can be seen that there is no clear event within the project that could cause catastrophic failure of the project; a result of being having adaptable software tasks. The objective of completing this risk assessment is to become aware of the problems in the project development, and mitigate their likelihood to be as low as practically possible (ALARP).

Likelihood		Consequence				
		1	2	3	4	5
		Insignificant	Minor	Moderate	Major	Catastrophic
A	> 100 / year	ALARP	Intolerable	Intolerable	Intolerable	Intolerable
B	> 10 / year	ALARP	Intolerable	Intolerable	Intolerable	Intolerable
C	> 1 / year	Tolerable	ALARP	ALARP	Intolerable	Intolerable
D	> 1 / 10 years	Tolerable	Tolerable	ALARP	ALARP	Intolerable
E	> 1 / 100 years	Tolerable	Tolerable	Tolerable	Tolerable	ALARP
F	< 1 / 100 years	Tolerable	Tolerable	Tolerable	Tolerable	Tolerable

Figure 7- Tolerable Risk Matrix, from METR3100 lecture notes, UQ, prepared by Dr Michael Kearney. [21]

Table 2 Risks Analysis based on Tolerable Risk Matrix

Hazards	Causes	Consequence	Likely hood	Existing Protection	Tolerability after Mitigation
Natural hazards *- flooding* *- draughts* *- economy lows*	Could be a result of global warming, but generally unpredictable	Insignificant	1/100 years	Software is cloud stored, as well as on local computers	Tolerable
Accidents *- Personal*	Unpredictable, could result in injury	Moderate	1/ year	Project is software based, progress can be made remotely	Tolerable
Accidents *- Software*	Deletion of software files or data	Minor	3/year	Documents and data is stored on hardware and in the cloud; version control. Can always revert back to previous efforts; V-REP and Jaco API are well documented	Tolerable
Accidents *-Hardware*	Unable to use or retrieve information from Jaco Arm	Major	1/year	Objectives 2, onwards, could be redesigned to be more simulation based.	ALARP
Technological failure *- Network* *- Computer*	Loss of connection from network or firewalls	Insignificant	100/year Could happen during	Can run client API and V-REP on virtual Linux machine with bridged network;	ALARP

	interrupting connection		any test where V-REP and client API are on different networks	does not require network connection.	
Compatibility	Hardware and software advances. Usability by anyone	Minor	1/10 years	Applications written in C can be compiled on a Linux or windows machine (differently, Linux make file included), and executed on any thereafter. All software used are current standard.	Tolerable
Funding	Robotics Design lab shuts down, or Jaco arm is sold	Minor	1/10 years	There is no protection from this, the thesis topic would change to be more simulation based.	Tolerable
Planning	A lack of planning of each project objective	Major	1/year	A log book is kept for all note taking during the project; with time set aside for development and planning.	ALARP
Time + other courses	There may be times where other courses or work take priority	Moderate	10/year	A plan has been made with deadlines and expected times to work on development.	ALARP
User friendly	Users of this interface may be unable to start the program	Minor		The code is available in GitHub with a how-to for starting the program. Unfortunately compiling the software is different between operating systems.	ALARP
Resources and help	Unable to find information on a particular	Moderate	Is a possibility, but very	V-REP offers a large range of documentation, online help and	ALARP

	phase of the project		unlikely 4/year	tutorials for remote based API clients. Kinova however offers one extensive document for their API. It is always possible that there is something required to complete the project that isn't documented, which could result in a change of strategy.	
Being hit by a moving object	Standing too close to the Jaco arm whilst it's in motion	Minor	1/year	The Jaco arm is a torque limited device, it shouldn't inflict injury. However, only one cautious person may stand near the arm whilst it is in motion, if not avoidable, with supervisor's permission.	Tolerable

Conclusion

Through the information and processes outlined in this document, the development of software which will enable V-REP to manipulate a physical Kinova Jaco arm can begin. To achieve this goal, the project was broken into four main sections, each modular and equal in difficulty. For support, V-REP and Kinova offer vast amounts of documentation, and their software is entirely open-source for educational users, thus ensuring progression. Time has been allocated each week to work on this project specifically, where workload is determined by progression leading towards the defined due date for each objective; however allowing time towards the project each week should ensure it moves steadily. Notes will be taken during all project times in a log book, with the pages titled and dated, outlining any progress or experiments. To ensure all of the data from any experiments, and the developed software or progress is safe, all information will be stored on GitHub and in a local directory. In doing this, risk of losing information or progress is mitigated.

Bibliography

[1] "Benefits and Costs of Process Simulation," Modern Machine Shop, 15 February 2000. [Online]. Available: http://www.mmsonline.com/articles/benefits-and-costs-of-process-simulation.

[2] "Process Simulate," Siemens PLM Software, 2017. [Online]. Available: https://www.plm.automation.siemens.com/en/products/tecnomatix/manufacturing-simulation/assembly/process-simulate.shtml.

[3] S. Fred M. Valerino, D. Sweetak and J. Osborne, "Parenteral products automation system (PPAS)". Patent US5805454 A, 8 September 1998.

[4] J. Rosen, B. Hannaford and R. M. Satava, "Surgical Robotics - Systems Applications and Visions," Springer, University of California Santa Cruz, 2011.

[5] P. Garcia, *Surgical Robotic Cell - Operating room without people*, Menlo Park, California: SRI International, 2008.

[6] "The da Vinci Surgical System," Intuitive Surgical Inc, 2017. [Online]. Available: http://www.davincisurgery.com/da-vinci-surgery/da-vinci-surgical-system/.

[7] D. Obenshain and T. Tantillo, "Distributed Systems and Networks Lab - Remote Telesurgery," [Online]. Available: http://www.cnds.jhu.edu/~dano/RemoteTelesurgery.pdf.

[8] T. Simonite, "The Recipe for the Perfect Robot Surgeon," Technology Review, 14 October 2016. [Online]. Available: https://www.technologyreview.com/s/602595/the-recipe-for-the-perfect-robot-surgeon/.

[9] L. Xiao-ming, Y. Can-jun, C. Ying and H. Xu-dong, "Hybrid event based control architecture for tele-robotic systems controlled through Internet," 11 September 2003. [Online]. Available: https://link.springer.com/article/10.1007%2FBF02841014?LI=true.

[10] S. Ivaldi, V. Padois and F. Nori, "Tools for dynamics simulation of robots: a survey based on user feedback," 27 February 2014. [Online]. Available: https://arxiv.org/pdf/1402.7050.pdf.

[11] L. Nogueira, "Comparative Analysis Between Gazebo and V-REP Robotic Simulators," [Online]. Available: http://www.dca.fee.unicamp.br/~gudwin/courses/IA889/2014/IA889-02.pdf.

[12] "Writing code in and around V-REP," Coppelia Robotics, [Online]. Available: http://www.coppeliarobotics.com/helpFiles/en/writingCode.htm#sixMethods.

[13] "Remote API," Coppelia Robotics, [Online]. Available: http://www.coppeliarobotics.com/helpFiles/en/remoteApiOverview.htm.

[14] "Dynamics," Coppelia Robotics, [Online]. Available: http://www.coppeliarobotics.com/helpFiles/en/dynamicsModule.htm.

[15] E. Rohmer, S. P. N. Singh and M. Freese, "V-REP: a Versatile and Scalable Robot Simulation Framework," [Online]. Available: https://www.researchgate.net/profile/Eric_Rohmer/publication/261352390_V-

REP_A_versatile_and_scalable_robot_simulation_framework/links/54720e120cf216f8cfadb08b/V-REP-A-versatile-and-scalable-robot-simulation-framework.pdf.

[16] M. Freese, S. Singh, F. Ozaki and N. Matsuhira, "Virtual Robot Experimentation Platform V-REP: A Versatile 3D Robot Simulator," [Online]. Available: http://robotics.itee.uq.edu.au/~spns/pubcache/simpar2010.vrep.pdf.

[17] "Kinova," Robotnik, [Online]. Available: http://www.robotnik.eu/kinova/.

[18] "KINOVA JACO² arm," Robotnik, [Online]. Available: http://www.robotnik.eu/robotics-arms/kinova-jaco-arm/.

[19] "Kinova Robotics - Jaco 6 DOF Technical Specifications," 2014. [Online]. Available: http://www.robotnik.es/web/wp-content/uploads/2014/04/JACO2_Brochure1.pdf.

[20] Kinova, "Jaco API Programming Guide," 10 April 2014. [Online]. Available: http://www.ee.oulu.fi/~sunday/jaco/JacoAPI_ProgrammingGuide.pdf.

[21] D. M. Kearney, "METR3100," 2016. [Online]. Available: learn.uq.edu.au.