Robot Manipulation with a Human in the Loop - Trajectory Recording and Playback

Sebastian Schweigert
Jiewen Sun
James Su
Sunil Srinivasan
Mark Jouppi

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2015-68
http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-68.html

May 13, 2015

University of California, Berkeley College of Engineering

MASTER OF ENGINEERING - SPRING 2015

Electrical Engineering and Computer Sciences

Robotics and Embedded Software

ROBOTIC MANIPULATION WITH A HUMAN IN THE LOOP – TRAJECTORY RECORDING AND PLAYBACK

SEBASTIAN SCHWEIGERT

This **Masters Project Paper** fulfills the Master of Engineering degree requirement.

Approved by:

1. Capstone Project Advisor:

Signature: _____ Date _____

Print Name/Department: PROFESSOR RUZENA BAJCSY, ELECTRICAL ENGINEERING & COMPUTER SCIENCES

2. Faculty Committee Member #2:

Signature: _____ Date _____

Print Name/Department: PROFESSOR RONALD S. FEARING, ELECTRICAL ENGINEERING & COMPUTER SCIENCES

University of California, Berkeley College of Engineering

MASTER OF ENGINEERING - SPRING 2015

Electrical Engineering and Computer Sciences

Robotics and Embedded Software

ROBOTIC MANIPULATION WITH A HUMAN IN THE LOOP – TRAJECTORY RECORDING AND PLAYBACK

SEBASTIAN SCHWEIGERT

Abstract

This project concerns the development of a system which allows tasks to be taught to a robotic manipulator using learning by demonstration. A subsystem calculates the accuracy required of the manipulator and determines which portion of the learned task it is able to perform. Therefore, the system can avoid performing tasks that require accuracy beyond the capabilities of the manipulator. The first sections of this paper are written by the entire project group and discuss the market and IP strategy suggested if the idea were commercialized. The final sections are written by myself and describe the trajectory recording and playback components of the project in technical detail.

1 PROBLEM STATEMENT

(Authors: Schweigert, Srinivasan, Sun, Su, Jouppi)

We live in an age of increasing automation, but while we have machines that can open a can, pour a glass of water, or assemble a piece of furniture, the world does not have a machine that is versatile enough to do *all* of these tasks.

Normally, when people think of automation, they think of robots designed to accomplish a very specific task, such as lifting a car-door into a car-frame, over and over again. The newer generation of robots though is the class of general-purpose robots. While such robots have yet to materialize commercially, general-purpose is a great concept. Imagine if families could have a robotic assistant to take care of household tasks or run daily errands. In short, human life would be much more convenient and efficient.

Unfortunately, a major limitation towards reaching such a milestone is the engineering trade-off between cost and performance: with a limited budget of resources, it is almost impossible to add additional levels of complexity without decreasing performance. As such, it has traditionally been challenging to use robots that are both low-cost and versatile in domestic environments because the applications of these robots are limited by their low performance – specifically, their inaccuracy. This is where we decided that the human should come in.

To address this barrier of limited resources, our capstone team has developed a system that is designed around robot-human interaction, where human instructors train and work with cost-effective robots to accomplish a broad range of tasks with high accuracy. Using a set of algorithms that we have developed, the robot learns how to perform a task from a human who teaches it through a series of demonstrations. Following this learning process, the robot evaluates the task and identifies the precision requirements using a mathematical model. And when the robot detects

that it is unable to achieve the accuracy required for a certain portion of the task, it requests human assistance. The final outcome is a system that excels across a vast range of duties, due to the combination of both the efficiency of robots working on a large-scale and the precision of humans working on a small-scale.

This revolutionary design of cooperation between man and machine succeeds at tasks that are otherwise impossible for the machine to accomplish alone. In essence, we added in the human as an additional resource to improve the overall performance of the system. This was the rationale behind our capstone project, for we saw an opportunity here to make an enormous technical stride in society's current usage of commercial robots: we took an otherwise unimpressive commodity – the low-cost and inaccurate robot – and engineered commercial value from it in the form of robotic adaptability.

2 INDUSTRY AND MARKET TRENDS

(Authors: Schweigert, Srinivasan, Sun, Su, Jouppi)

Before examining any technical details though, we first wanted to scope out the business potential of our project. Consequently, in an attempt to analyze our strategic position in the market, we evaluated the competitive forces outlined by Porter (Porter, 2008) because we felt that an in-depth analysis of the intensity of these forces will influence our marketing strategies. In other words, analyzing these five forces enabled us to have a better understanding of our industry and shaped our strategies to sustain long-term profitability. Before we begin our analysis however, let us first clearly define both our market and our product.

2.1 Market and Product

We defined our market to be consumer households with the intent that our algorithms accomplish household tasks, such as assembling furniture. We chose to target the elderly and the disabled as buyers of our product because this is a large, growing population with critical and largely unmet needs. Simply put, the elderly population in the United States is growing. While the current number of senior citizens in the US is roughly 40 million, that number is expected to grow to over 80 million by 2050 (Ortman et al., 2014). Additionally, according to US 2010 census data, about 30%, 38%, and 56% of the population aged 70 to 74, 75 to 79, and 80 or over, respectively, live with severe disabilities (Brault, 2012). To further narrow our market though, we chose to focus specifically on affluent elderly-and-disabled individuals as our target customers. This is a reasonable objective because many elderly people have amassed a wealth savings and investments cultivated over their lifetimes. Indeed, according to a Pew Research study, the median net worth of senior citizens in the US is $170,000, which is 47 times greater than that of people aged 35 and younger (Censky, 2011).

The definition of our product is a more complex matter because, at its core, our capstone project involved the research and development of an algorithm that allows a robot to learn a task and cooperate with a human to perform that task; it is not a complete software – or hardware – solution. Unfortunately, while software solutions usually have commercialization potential, algorithms alone do not. In order to take our robot-learning algorithm and relate it to a commercial application, we had to decide what form that application should take and how to take such a product to market. One option was to simply license out our algorithm for others to utilize; we would receive royalties as a result of these sold licenses, and companies could make products or provide services using our algorithm. One major caveat, though, is that our algorithm incorporates ideas

presented in externally-published research, so the intellectual property for this method may not lie entirely with us. We therefore chose not to investigate this option any further. Our next option to consider was to sell a software solution for users to install on devices that they already own. However, the "device" in this case would be a full-fledged robot, where, as a point of reference, a Baxter robot from Rethink Robotics – our current hardware-platform of choice – has a set price of approximately $35,000 (Rethink Robotics, 2015). Clearly, it would be ludicrous for people to purchase such costly technology without ensuring that it already comes with the necessary software to function. This left us with our final choice: a "full package", in which we offer a robotic apparatus preloaded and set up with our software such that a consumer only needs to buy one product, with installation services if necessary. This way, we can market our product directly to our target consumers and eliminate the customer's barrier-to-purchase that comes from setting up the technology. Thus, we decided on this "full package" as the form for our product: a physical robot bundled with software algorithms that we implement.

We must consider several factors with the decision to market this "full package". The first is price, and this is largely influenced by the suppliers since we must obtain the proper robotic hardware or components externally. After all, according to an IBISWorld report, the cost of mechanical manufacturing is increasing as the expenditure of raw materials increases, so we opt to purchase a whole robot setup instead of building our own robot from basic components (Crompton, 2014:25). As a result, we would look to Rethink Robotics as a supplier of our Baxter robot, a hardware platform. With a markup from our software and services, selling our product at around $40,000, or at about a 15% markup, is not an unreasonable price point – especially if we were to get an Original Equipment Manufacturer (OEM) discount for Baxter. This provides us with a defined pricing model.

Lastly, we must discuss promotion and place/distribution. As O'Donnell points out, 50% of seniors are not using the internet, so marketing is better achieved through conventional channels such as mail, television, and newspapers (O'Donnell, 2015). Interestingly, O'Donnell also predicts an increased use of social media by seniors in 2020, making social-media campaigns a possibility in the near future (O'Donnell, 2015). Distribution of this product, however, is complicated; while we would like to be able to sell our product online, providing setup services would require a trained professional to be present. As such, we will most likely have to either distribute through local partners that provide such services or create a local presence ourselves, incurring additional costs. With our product, price, promotion, and place now defined, we have all the significant facets of a commercialization strategy. Note that we do not analyze the minimum viable product (MVP) in detail. This is because our research specifically investigates the Baxter robot's ability to learn the task of assembling a coffee table, at which point we will have a decent MVP that performs table assembly. Thus, we have established a viable (if hypothetical) commercialization strategy for our research efforts.

2.2 Competitive Forces Analysis

2.2.1 Power of Buyers

With the market and product definition out of the way, we can begin to evaluate Porter's five forces, the first of which is the power of buyers (Porter, 2008). We deduce this force to be relatively weak, since the large population of potential buyers means that individual buyers do not have much leverage or bargaining power with us in our product offering. Moreover, as we will address later on, there are few – if any – direct rivals in our industry. Thus, a scarcity of competing products only elevates our power, as options are limited for the buyer. Furthermore, the switching

costs for complex, robotic solutions would be high; given that the price of these robots with our software would be roughly $40,000, it is not an expense to be made frequently. We imagine that a typical customer will only purchase one such robotic system in their life. Thus, it is not of great concern that customers would switch to using a competitor's domestic robot solution after purchasing our product. All in all, the power of buyers is assessed to be fairly weak, and we do not concern ourselves in mitigating this force.

2.2.2 Power of Competitors

Regarding rivalry within our industry, there are two main classifications of competitors: robotics companies and robotics research institutions. Some of these competitors offer products that are mildly similar to our envisioned product, and they also target similar markets. For example, Clearpath Robotics, a robotics company (Hoover's, Inc. "Clearpath Robotics Inc.," n.d.), offers support to the PR2 robot to perform household chores like fetching food from the refrigerator and cooking it. Alternatively, there are research institutions like the Information and Robot Technology Research Initiative (IRT) at the University of Tokyo working on developing software that allows the AR robot to accomplish household assignments such as cleaning floors, washing laundry, and so on. Fortunately, companies and research institutions like these will only indirectly compete with us because our product differs from theirs in the extent that humans are involved. The robotic systems these competitors are developing are meant to be fully autonomous – the robots execute their tasks independent of any human interaction – while our system is meant to be semi-autonomous, enabling a human to both work with and teach a robot to perform various tasks. This is an advantageously superior method because now the scope of the system is not limited to what the robot can accomplish independently; the scope is broadened to what the robot and human can accomplish together synergistically. Simply put, the generality of our method enhances a robot's

utility and flexibility. Apart from offering a unique product though, we also have some advantages over our competitors in terms of hardware costs. To illustrate, a two-arm PR2 robot is priced around $400,000 (Owano, 2011) while a Baxter robot, as mentioned previously, is priced around only $35,000 (Rethink Robotics, 2015), a relatively far-cheaper option. To summarize, since we are working in a fairly new field, there are no true established rivals in this specific area yet. Thus, we can conclude that the force of competitors is weak.

2.2.3 Power of Substitutes

Moving onto the next force listed by Porter, we realize that significant attention needs to be given to the force of substitutes since there are, broadly speaking, quite a number of substitutes to our product. For instance, alternative technologies, like the iRobot Roomba (Biesada, n.d.) – a popular floor cleaning robot, have existed in the consumer market for many years, and these established technologies have a large customer base. Customers are more comfortable with familiar products, so it will not be easy to encourage customers to migrate to a substitute product. Moreover, if we look past the technological substitutes, there are a variety of human-labor alternatives in regards to accomplishing household tasks, such as employing a live-in caretaker or residing in a nursing home. However, similar to our stance against the competitor force, we again have some advantages due to our functionality and low cost. Addressing the concern of alternative technologies, even though products like the iRobot Roomba are popular and functional, they tend to have a limited set of features, such as floor cleaning. Our product, on the other hand, is a more general solution which can be used to tackle a variety of household chores. Along that same line, for many tasks in this set, our robot can be more efficient than a human caretaker due to its autonomous nature. Furthermore, as mentioned previously, our pricing model markets our product at a cost of about $40,000 with an extensive lifespan, while most nursing homes cost up to $80,000

– and that is per year (Ellis, 2013). All of these arguments make our product competitive to existing substitutes, motivating us to divert attention from this force and concentrate on more pressing ones.

2.2.4 Power of New Entrants

In contrast to the mild nature of the forces mentioned previously, new-entrant competition looming over the horizon should be of great concern. For instance, some of the heavy-hitters in robotic research include companies like Clearpath Robotics (McMillan, 2015) and 3D Robotics (Bi, 2015), both of which were founded only six years ago in 2009. It seems that, unlike the issue of existing rivals and possible substitutes, there is indeed a strong force in regards to new entrants. To further illustrate this fact, large corporations with broader goals in the technological field can certainly seep into our industry, such as Amazon with its Amazon Prime Air drones or Google with its autonomous cars. Big players such as these would certainly have the resources to quickly create a new division within their company and fund research in alternative robotic avenues. Furthermore, even our suppliers can be considered possible new entrants, since they both already posses their own hardware and can additionally reverse-engineer our software algorithm that was, in large-part, acquired from public research papers. All in all, to summarize, we see that dangerous incoming players in this industry are: either startups or big companies with other additional focuses, or suppliers that provide our hardware. When combined with the fact that there are no true established rivals yet as mentioned previously, this danger reinforces both the notion that robotics is a relatively new field and that the threat of new entrants is high.

2.2.5 Power of Suppliers

The last of Porter's five forces to address is the threat of suppliers (Porter, 2008). This threat is a complex point that requires careful analysis in our business strategy. To first clarify, we

envision robotic-hardware-platform manufacturers as our suppliers. As per our product description, we would take the robotic hardware platforms from companies like Rethink Robotics and Universal Robotics, customize the robots with our specialized software that gives them practical intelligence to work alongside humans, and then sell them to customers. In particular, we would purchase from companies that produce innovative, low-cost robotic hardware platforms upon which we can then build our solution. Our smart software would make up for the inaccuracies in the cheaper hardware with better algorithms and human-in-the-loop collaboration. Since there are currently only a few firms producing such low-cost platforms, these few suppliers have high bargaining power, as we are left with fewer alternate firms from which to choose.

2.3 Market Strategy

We see that presently, of Porter's five forces, both new entrants and existing suppliers hold the most power (Porter, 2008). Knowing this, we can establish our market strategy to mitigate these two forces, strategically positioning ourselves in a superior situation.

To mitigate the threat of new entrants from the suppliers themselves (see Section 2.2.4), we can generalize our software to work across multiple platforms and disincentivize suppliers to enter the market, as they would only be encouraged to produce software across their own single platform. Additionally, to discourage new and small startups from forming, we can both establish strong relationships with suppliers to gain a leg up on others looking to pursue our method of utilizing existing hardware and maintain a high fixed cost – such as a high R&D cost by developing more proprietary algorithms – to deter incomers that have a small amount of seed funding. Finally, we can address the threat of entry from large corporations by realizing that these companies have more overarching goals, so focus on their robotics branch will not be as heavy as on their other

branches. As such, we can capture a niche market to detour focus and attention away from us. Fortunately, we have already positioned ourselves in such a situation, in which we target a niche group of customers – the elderly and the disabled. As a result, we see that our competitive landscape as it applies to new entrants can be classified as quite aggressive, but there are indeed routes we can take to dodge much of this aggression.

To mitigate the issue of being locked into a single supplier (see Section 2.2.5), the core strategy is still to generalize our software. This would considerably increase our power, since we would no longer be dependent on any one supplier. Note that as a trend, robotics startups are becoming increasingly common (Tobe, 2013), and we thus anticipate more suppliers coming into the market in the future. As of right now though, suppliers are a strong force that must be considered carefully in our strategy, and we must route efforts to ease this force.

2.4 Market Trends

With an evaluation of competitive forces complete, we end with a discussion of the major trends that influence our project strategy. Aside from the trends of both the changing age demographic of the US – affecting the power of both buyers and substitutes – and the increased interest in the robotic industry – affecting the power of both substitutes and new entrants, another trend to consider is the recent advancement in integrated circuit (IC) technology that has resulted in improved computing performance and reduced cost, resulting in reduced barriers-to-entry and thus further enhancing the threat of new entrants. IC technology has seen consistent improvements in computing power and consistent reductions in cost since their inception. Our industry is directly affected by these advancements; in recent years, more powerful computational devices have generated more robotic technology in the household arena, for engineers are allowed to easily

incorporate computing power into the chassis of the robot. This design contrasts with industrial robots, where the computational power is often located in an external computer. The trend is summarized with a concept known as Moore's law, stating that the computational power of the average IC doubles nearly every two years. This trend has been relatively consistent since the early history of ICs. However, there is disagreement among analysts about how much longer this trend will continue (Hoover's, Inc. "Home Health Care Services.," n.d.). The trend has the effect of making our products more functionally efficient and versatile, which reduces the power of substitutes. However, the lower cost of computing technology also reduces the barriers-to-entry in the industry, which increases the power of rivals. Only time will reveal the overall impact that this trend will have.

To summarize, from our strategy analysis, we have deduced that while some competitive forces are certainly in our favor, a few forces bring cause-for-concern and need to be addressed. With adequate industry analysis, we can plan our strategy in order to leverage ourselves into a better position within the market. Summarizing our findings, we have identified within the market both the power of new entrants and the power of suppliers to be strong forces. Consequently, to dampen these threats, we would generalize our software to work across multiple platforms, disincentivizing suppliers from entering the market as well as taking away supplier bargaining power. We would also encourage people to use our product instead of substitutes by having features and functionality that other products do not, at a price point that is not prohibitively expensive.

3 IP STRATEGY

(Authors: Schweigert, Srinivasan, Sun, Su, Jouppi)

Aside from a business standpoint though, we must also consider which legal avenues to take in order to protect our intellectual property (IP): in particular, whether or not our idea is patentable. After all, in many research scenarios such as ours, a patent is the most feasible way to safeguard any IP that is developed. Unfortunately, as this section will argue, patenting our work may not be the most practical path to pursue; however, we do have an alternative strategy better suited to our purposes, in the form of copyright.

We feel that in our more specific situation, the costs of attempting to obtain and enforce a patent far outweigh the benefits, for a number of reasons. One consideration is that the mathematics behind the algorithms we employ are pulled from published research papers, particularly those that deal with robot learning-by-demonstration (Billard et al., 2008). Therefore, the proprietary essence of such research is not ours to claim. By the same token, we cannot patent the ROS (Robot Operating System) software platform upon which we develop because it is open-source and thus, once again, publically available. Most importantly, we do not feel that it is pragmatic to patent the software code itself. This is because software, at its core, is the manifestation of logical deductions, and another group or individual may take a different route of logical deductions to arrive at the same conclusion. Following this train of thought, it is ordinarily quite difficult to obtain and/or protect a patent when the end result can be reached in various ways. As explained by Anthony Klein, an IP attorney at Latham & Watkins LLP, pure software patents remain controversial since "what would constitute patentable subject matter is unclear" (Klein, 2015).

Before investigating an alternative means at protecting our ideas however, it is important that we have the foresight to research whether existing patents overlap with our results. We

discovered that the closest patent to our project is entitled: “Method and system for training a robot using human-assisted task demonstration” (Bajaras, 2014). It describes a system for humans to train robots for pick-and-place tasks by moving the robot’s arm while recording trajectory-and-perception data through the robot’s sensory systems. At first glance, it may appear that our project directly infringes upon this patent. However, after delving into the details, this is not the case due to the limited scope of this patent. To give some background on the nature of patents, a patent consists of independent claims and dependent claims; if one does not violate the independent claims, then by definition, one does not violate the dependent claims (Brown & Michaels, PC 2006). Now, many of our project’s similarities with this patent lie in the dependent claims. However, if we can argue that our capstone project does not infringe upon any of the independent claims, then we can legally claim that we do not infringe upon the dependent claims as well – and thus the patent as a whole.

There are two independent claims mentioned in this patent. To quote the first independent claim (claim 1):

> A method for training a robot to execute a robotic task in a work environment, the method comprising: moving the robot across its configuration space … assigning, via the ECU, virtual deictic markers to the detected perceptual features (Bajaras, 2014).

We argue that we do not infringe this claim because our project does not use “virtual deictic markers” – markers are based on a representational paradigm that use “selective attention and pointers … to learn and reason about rich complex environments” (Ravindran, 2007). As for the second independent patent claim (claim 2):

> The method of claim 1, wherein moving the robot across its configuration space includes moving at least one of a robot arm and a robot manipulator attached to the robot arm (Bajaras, 2014).

Our project does not use a single "arm and a manipulator", but rather a dual-armed Baxter-robot. Hence, our project does not violate any of the independent claims and thus none of the dependent claims. Therefore, while this is the closest patent to our idea, we do not infringe upon it and are therefore not required to license from it. Since other existing patents are even less related, a breach of IP is of no worry to us.

With the threat of similar, existing IP out of the way, we can now begin to pursue an alternative strategy of IP protection. After much consideration, we believe that copyright is the most appropriate option – in fact, this happens to be the choice for many software companies. Of course, copyright does indeed present a few risks since, in general, patents protect ideas while copyright only protects the expressions of ideas.

The first risk is the risk of knock-offs: there are ways around copyright such that people can make products very similar to ours but are not in violation of copyright law. This includes implementing our algorithm through a different tactic – one example is converting our code to a different programming language – as well as merely adapting functions from our program. The point is that copyright does not protect our ideas, making it incredibly easy for others to take our ideas and tweak them to look slightly different in their end product. We would need to mitigate this issue by implementing our algorithm across multiple programming languages to prevent the scenarios where someone claims credit on our ideas based on simple modifications.

The second risk is the risk of undetected duplication. It is the first risk in reverse, where certain competitors are indeed copying our code directly, but we have no way of detecting that

they are doing so. The reason for this is that we will generally not have the source code of our competitors to compare to our own; all we will have is the compiled functionality that their code is capable of demonstrating. In that sense, it is near impossible to identify specifically if they have violated copyright. Consequently, it is quite difficult to mitigate this risk.

While copyright does offer less protection than patents, it is nonetheless more feasible and realistic to acquire. For instance, copyright is granted automatically when an original work is created, so registration is not required. This simplified procedure immediately eliminates the time and money that we would otherwise need to spend to obtain a patent. Moreover, the duration of copyright is the life of the author(s) plus 70 years, which is plenty of time for us given the short life cycle of software. Furthermore, copyright offers authors the exclusive rights to reproduction, protects against public displays and derivatives of their work, and establishes a public credibility that can attract investment and customers. Licensing can also present itself as a way to increase profit and expand a business.

All in all, it appears that pursuing a patent is not the route for us to go. Instead, a more practical approach at protecting our IP is for us to pursue copyright, due to both the more lenient restrictions and more efficient timeline at obtaining copyright.

4 SYSTEM WORKFLOW

(Authors: Schweigert, Srinivasan, Sun, Su, Jouppi)

Shifting gears now to the technical details of our capstone project, let us clarify once more that we developed a general-purpose robotic system that incorporates robot-human interactions. Yet it is difficult to implement generality without first implementing and testing lower-level components. Therefore, as a stepping stone for a starting point, we defined a specific task that we

aimed to accomplish: having Baxter assembling a coffee table with the help of human. (Figure 1 shows the workflow of this system.) Having such an aim allowed us to physically manifest an implementation and test of our system.

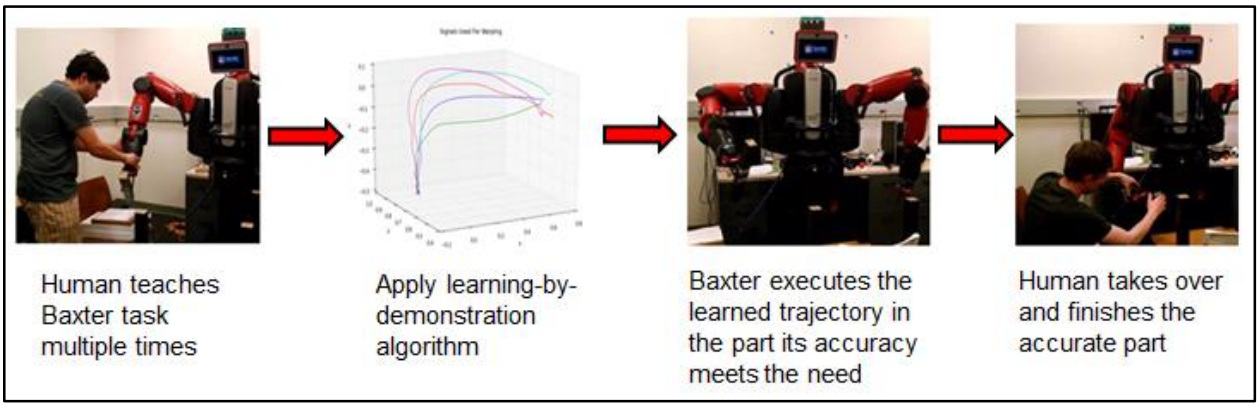

Figure 1: Workflow of System Process

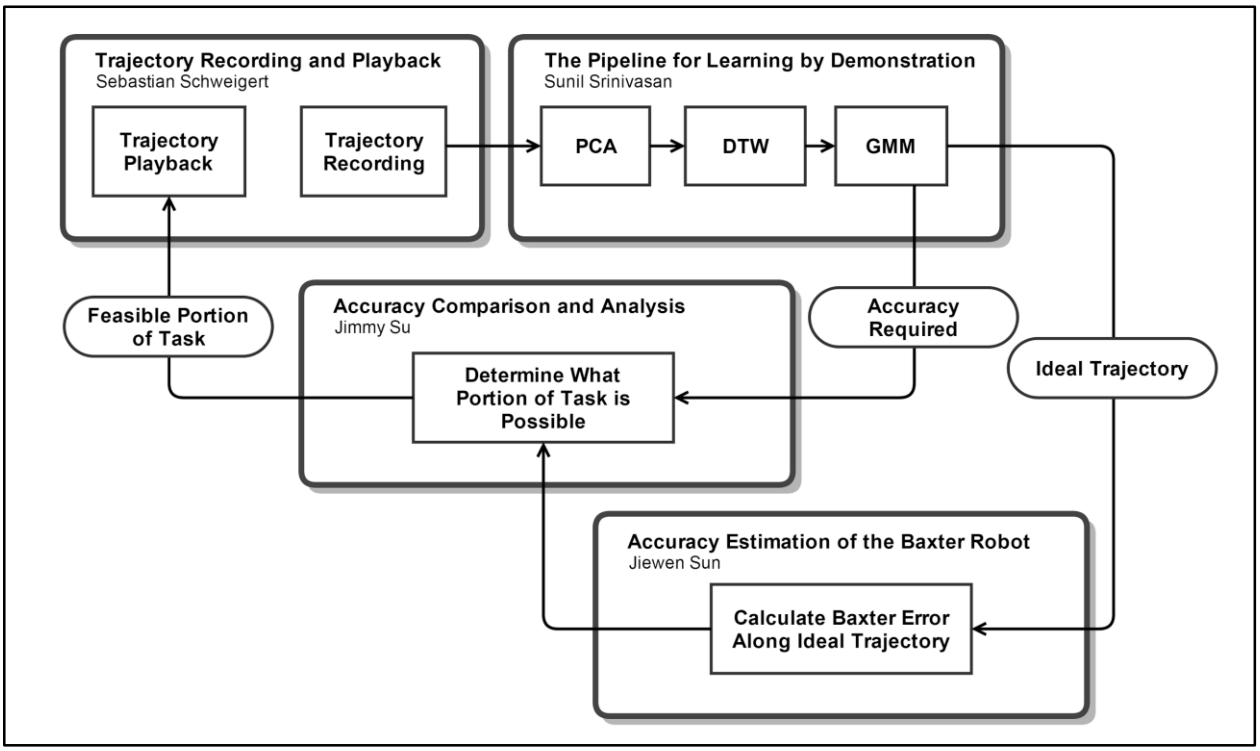

Figure 2: Paper Organization

5 TECHNICAL CONTRIBUTIONS

(Author: Schweigert)

5.1 Overview

The goal of the project is to develop a software system which enables inaccurate robotic manipulators to operate in a wider environment. This project focus is motivated by the fact that low-cost robotic manipulators are becoming more commonplace, but there is limited research being done to accommodate their relative inaccuracy.

Tasks are taught to the system using learning by demonstration, an increasingly common technique. Prior work for learning by demonstration can be found in (Abeel and Ng, 2004), (Calinon et al., 2007) and (Billard et al., 2008). The teaching process involves dragging the robot's arms manually in order to perform the task in question. Using these demonstrations, the software system generates an abstract model of the task. This model is referenced when a human requests that the robot performs the task on its own.

The learning by demonstration system derives some measure of the accuracy required of the robotic manipulator at each stage of the task. When the robotic system is asked to perform the task, it considers the accuracy required alongside its own limitations. Then, it identifies whether or not it is capable of completing the task. This system prevents the robot from failing to do tasks in a way that could break objects or damage the robot itself. For example, a robot might not be accurate to place a glass on a shelf. If it attempted to do so regardless, it could break the glass.

The software system consists of several modular components. The first component allows trajectories to be recorded by guiding the robotic arms through tasks. These recorded trajectories are then combined and analyzed in order to generate a model of a particular task. A variety of techniques were used to analyze and combine the trajectories, including dynamic time warping

and Gaussian mixture models. More details on the techniques used can be found in in the paper written by group member Sunil Srinivasan. A module of the system is also required to determine whether the robotic is accurate enough to perform a task it has learned. Details on the techniques used to make this determination can be found in the papers written by group members Jiewen Sun and Jimmy Su. Finally, the trajectory playback system drives the manipulator to perform the task after it has been verified to be within the accuracy limitations of the robot.

Originally, Mark Jouppi and Sunil Srinivasan were assigned to the learning by demonstration module, while myself, Jimmy Su and Jiewen Sun were assigned to the trajectory recording and playback modules. Once trajectory recording and playback was completed, the three of us moved towards developing a system which generated a measure of the manipulator accuracy at a given manipulator configuration. The three of us worked in a collaborative way. We regularly met to discuss the progress being done on the software, as well as what work was required going forward.

This report discusses the design and implementation of the trajectory recording and trajectory playback modules. I chose to write about both these modules because I made large contributions to their development. Additionally, the two modules are similar in their purpose and design. They both interface to the Baxter robot, and are the only modules that do so.

The trajectory recording and playback modules represent basic functionality implemented on the robotic manipulator. As a result, the design was not guided by literature review. However, an understanding of the other modules was necessary during the design process because those modules are supported by the trajectory recording and playback modules. For example, the specific data recorded by the trajectory recording model was chosen so that the task modelling system had enough information to function correctly.

5.1.1 Methods and Materials: System Configuration

The trajectory recording and playback modules are software systems that interact with the robotic manipulator. Thus, the modules were designed in software. Specifically, they were written in Python and used a software framework called ROS (Robotic Operating System).

ROS was chosen as a platform on which the system would be developed for a variety of reasons. ROS is a software framework that was designed to ease development of highly modular robotic systems. The framework allows programmers to build many independent modules that can interact with each other using a form of communication called “messages.” ROS was a good option because the design of the system was modular, which is ideal for ROS. Additionally, Rethink Robotics (the company which sells the Baxter robot) has developed an SDK for the Baxter robot that is designed to work within ROS. The SDK provides many methods for communicating with and controlling the robot. Consequently, it was necessary to use ROS if the software was to be written for the Baxter robot.

A choice was made regarding which programming language to use in the software system. Three languages are supported by ROS: Python, C++ and LISP. Python was eventually chosen because everyone in the group had used it before and it is a language designed for fast prototyping. The project is only a year in length, and therefore quick implementation is important. Combining multiple programming languages within ROS was also considered. However, this would make the development process more difficult due to a lack of consistency. For example, it is difficult to consolidate code written in two different languages.

5.2 Trajectory Recorder

The purpose of the trajectory recording system is to facilitate bulk recording of trajectories. It is important that the module allows users to easily and quickly make large numbers of recordings. The choice of data to record is also an important consideration.

The implementation of the module is based on a code example provided by Rethink Robotics. This example provides useful methods which demonstrate the syntax for reading joint angles from the Baxter robot. It is important to point out that the example itself cannot act as a replacement for the module, as it doesn't fulfill the requirements. The example only records a single trajectory and has to be stopped by pressing a button on the keyboard. This is not ideal because the system must be capable of recording many trajectories, and doing so quickly.

Modifications were made to the example in order to match the requirements of the module. The key changes made were to re-organize the code so that the recording could be done multiple times in a row without having to re-run the software or touch the keyboard. Code was added which provided control to the buttons on the arm of the Baxter robot, instead of relying on the keyboard. The buttons on the arm could then be used to start and stop the recording process. As a result, the team members can record trajectories without having to remove their hands from the robotic arm.

The module was designed to record lots of extra data from the robot. This choice was made to improve flexibility in the event that the learning by demonstration system was modified to require different data sources. If that were to happen, then the trajectories would not have to be re-recorded. Instead, the additional data from the existing recording files could be re-used. The choice to record extra data was also motivated by the fact that there was very little overhead associated with doing so.

The data recorded for each arm is summarized in Table 1, and Figure 1 shows the names and locations of the joints in the robotic arm.

Table 1: Summary of Recorded Data

Data Recorded	Source
Position of all joints (see Figure 1)	Read from Baxter sensors using baxter_interface
Velocities of all joints (see Figure 1)	Read from Baxter sensors using baxter_interface
Position of end effector	Calculated using ROS tf package
Orientation of end effector (RPY)	Calculated using ROS tf package
State of gripper	Read from topic published by gripper_control
Displacement of end effector relative to first change of gripper state	Calculated after main recording process has finished

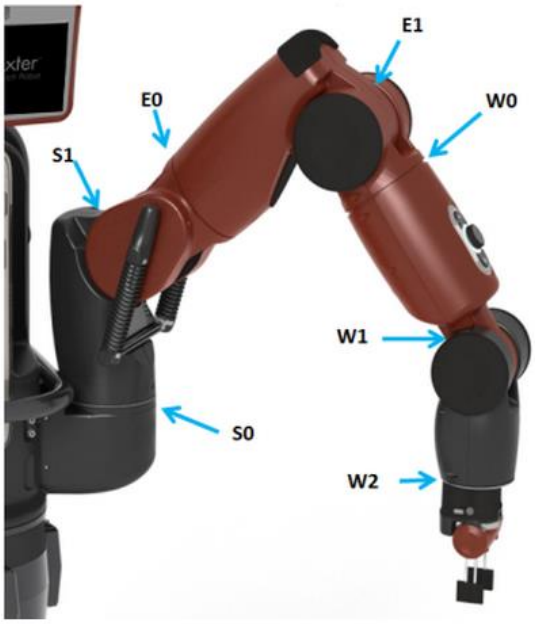

Figure 1: Joints on Baxter Robot (Rethink Robotics 2014)

In Table 1, the *State of gripper* field is a Boolean which is true as soon as the gripper has been given the signal to close, even if it is remains open for a short period of time while the actuators are working. Likewise, the variable changes to false as soon as the gripper has been given the signal to open.

The last variable in Table 1 is a special variable used for task modelling. It represents a vector pointing from the position where the gripper first changes state to the position of the end effector at a given time interval. This variable helps represent the task because it gives the displacement of the end effector relative to the object with which the gripper interacts. The variable has to be calculated after the entire trajectory has been recorded, because the location of the first change in gripper state is not known from the start of the recording.

A set of recording tools called Rosbag is also available within the ROS framework. The group decided not to use these tools when recording trajectories for several reasons. Firstly, they

were considered too inflexible. For example, the .bag files that Rosbag uses are difficult to import into MATLAB for analysis. Therefore, the use of Rosbag could have slowed progress since MATLAB was frequently used in the project to rapidly prototype algorithms. Additionally, some of the recorded data was processed before saving. In order to record the processed data using Rosbag, the data would have to be published to a topic after processing. Using csv files allows the data to be manipulated directly through Python instead of having to use the ROS framework. In this case, the use of Rosbag makes the system unnecessarily more complex.

5.2.1 Usage

Figure 2 shows the steps required to record trajectories using this module. The procedure outlined in Figure 2 facilitates fast and easy bulk recording of trajectories. The trajectories are saved as .csv files in the /data/ folder of the project. A batch of trajectory files will start with the same filename, but will have an ascending numerical suffix.

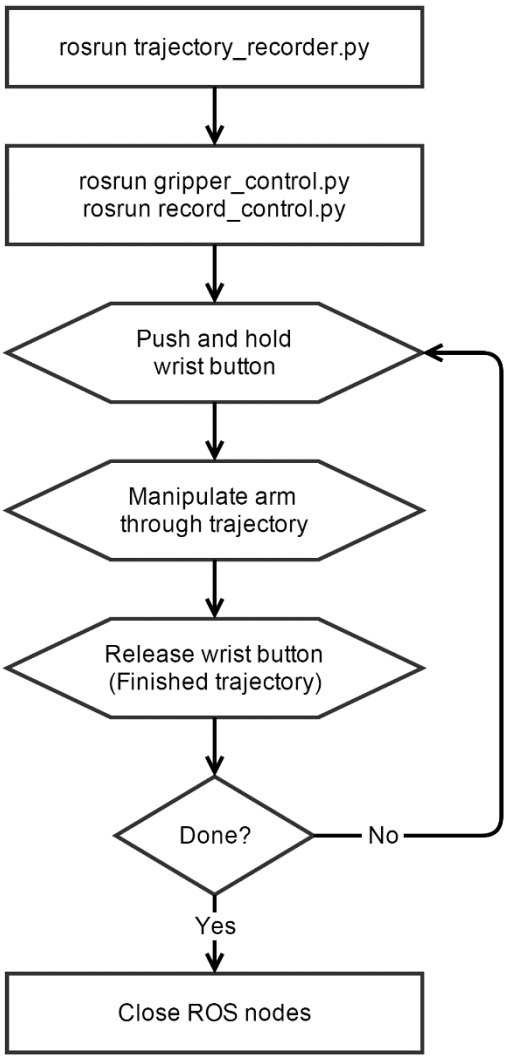

Figure 2: Trajectory Recording Usage Flowchart

5.2.2 Algorithm

The trajectory recording systems works on a relatively simple algorithm. The algorithm is illustrated in Figure 3.

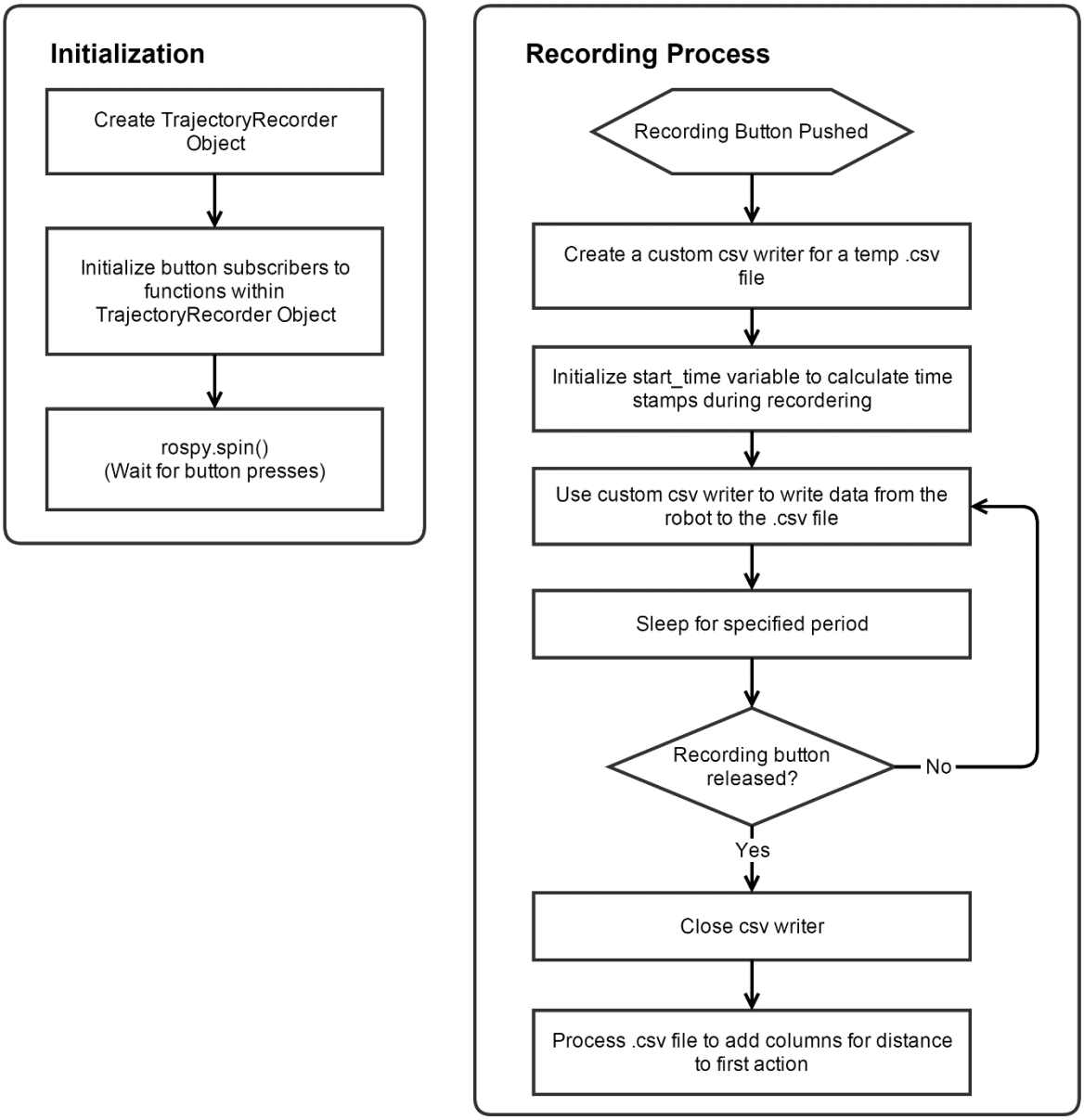

Figure 3: Trajectory Recording Algorithm

In Figure 3, the initialization step occurs as soon as the module is run, while the recording process only begins when the button event is detected. The last step in the recording process is to process the .csv file to show the distance to the first change of gripper state (ie. the first action). The algorithm for this step is shown in Figure 4.

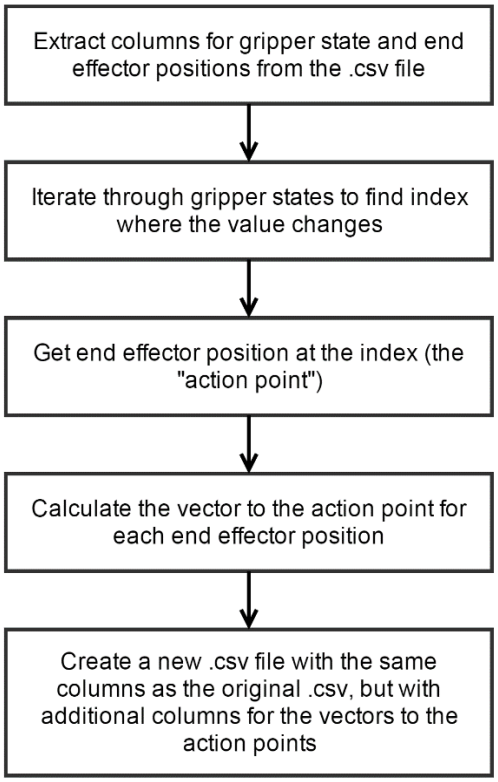

Figure 4: Algorithm for Inserting Vectors to Action Point

In Figure 4, “action point” refers to the point at which the gripper changes state. As mentioned previously, the location of this point is not known before trajectory recording begins. Therefore, the steps in Figure 4 must occur after the overall trajectory recording process has completed.

5.2.3 Future Work

The opportunities for future improvements of the module are listed in in this section.

Task:

Create a launch file which runs record_control.py, gripper_control.py and trajectory_recorder.py

Details:

This will simplify the task of recording trajectories because only one command will have to be entered into a single terminal. The group implemented this launch file, but it was found that trajectory_recorder.py would no longer save the .csv files. The files would simply not appear on the hard drive. The problem is probably caused by the way that ROS runs python files using a launch command. To solve the problem, the options used within the launch file must be chosen appropriately.

Task:

Generalize the add_distance_to_change function (described in Figure 4) so that it works for both the right gripper and the left gripper.

Details:

Currently this function only takes the right gripper into account. This limited functionality was sufficient because the project focus was on just the right gripper. However, the system should eventually be implemented so that it works for both grippers in combination. It is also simply good practice to keep the code as general as possible.

5.3 Trajectory Playback

The purpose of the trajectory playback system is to control the arms of the Baxter robot so that they move along the trajectory calculated by the task modelling system. Input to the module is a list of desired points along the trajectory which the robot must follow. The module has no output, but instead is required to ensure that the robot performs the task successfully. The module is used after the other components in the system have determined that the robot is sufficiently

accurate enough to perform the task in question. Thus, the module is quite simple since its only task is to relay the actions to the Baxter robot.

The trajectory playback module relies heavily on pre-existing functions provided by the ROS framework. These functions work closely with the Baxter SDK provided by Rethink Robotics in order to perform their task. The functions give the ability to execute Cartesian paths on the Baxter robot. First, an inverse kinematics function is used to produce the joint angles required in order to have Baxter trace out the path. Next, Baxter's joints are instructed to move through the required joint configurations, which will reproduce the Cartesian path that was originally requested. This action is performed using a control system running on the Baxter robot itself.

Despite the fact that these useful functions are provided for use by third parties, significant work still needed to be done in order to use them correctly. ROS is open-source software, and is very poorly documented as a result. Therefore, learning to use these functions so that they perform consistently was a difficult task. The data needed to be processed and formatted so that it could be played back properly. There were many problems associated with the implementation of this module, which will be discussed in Section 5.3.2.

5.3.1 Usage

Usage of this module is trivial. Plans can be generated using the plan_trajectory function and then executed using the execute_trajectory function. Both functions are found in Cartesian_playback.py. There is a main() function in the module which gives an example of its use. Please examine this main() function to get a better understanding of how to use the module.

5.3.2 Challenges

The ROS software package is generally poorly documented. As a result, it is not always easy to use the functions provided by the framework, even if they are useful. When the group implemented the trajectory playback module using the path planning and plan execution functions, they did not work as expected. The Baxter robot executed the desired path in a slow and jerky way. It could take the robot a minute just to move its arm 1m along a relatively straight path.

It was identified that the system generated too many points for the path execution system. The control system was designed to make sure that each point on the path was reached before moving on to the next. The system inevitably ran slowly because it was trying to ensure that all the points were reached exactly. Ideally, the system would interpolate between the points in order to generate a smooth path. To illustrate this, examine Figure 5.

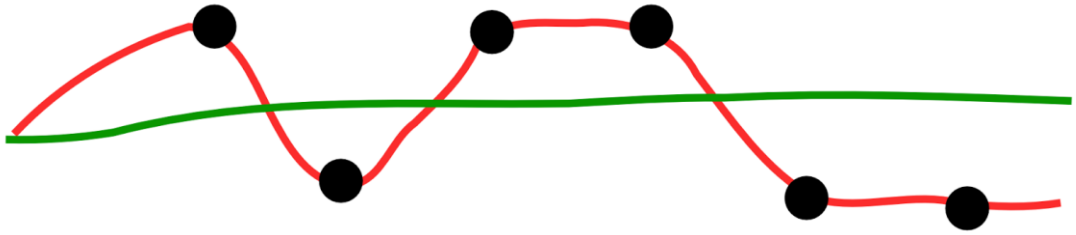

Figure 5: Path Execution Example

In Figure 5, the black dots represent the points given to the path planning system. These points are supposed to represent an approximately straight path, and the distance between each successive point is only a few centimeters. Random variations have caused the points to not line up perfectly, making them not look straight. The red line shows the path of the robot's manipulator when it executed the path. The green line shows a more ideal outcome. It is clear that the red line has a greater length, which means that the end effector has a further distance to travel. Additionally,

the velocity of the end effector changes frequently. For example, at the second point it takes a 90 degree turn in order to proceed to the third point. Both of these effects cause the playback to proceed slower than expected.

A temporary solution to this problem was to remove points that are very close to each other. The first and last points are never removed because they are likely important for the completion of the task. This method resulted in a sparser distribution of points in the path, which allowed the trajectory to be executed more quickly. In the future, smoothing algorithms should be used to process the path, thereby improving execution performance.

5.3.3 Future Work

The opportunities for future improvements of the module are listed in in this section.

Task:

Improve robustness of system

Details:

Currently the system is not very robust. Trajectory planning or trajectory execution may fail, resulting in errors. Generally, the errors are returned by MoveIt and indicate that the joint thresholds are exceeded. The errors indicate that the trajectory cannot be executed because it would require that one or more joints rotate to a disallowed angle. Often, these errors will disappear if the plan is re-generated. To solve this problem, the error should be handled appropriately instead of halting execution. Once the error is caught, the plan can be re-generated or a message could be sent to the user indicating what has occurred.

Task:

Improve the performance of plan execution by adjusting eef_step, and jump_threshold and by pre-processing the trajectory.

Details:

Part of the problem is the lack of ROS documentation in python (the C++ documentation was used as a replacement). The group does not fully understand how to use the eef_step and jump_threshold parameters given to compute_cartesian_path. The table below summarizes the documentation (ROS, 2015) for these parameters and recommendations for their use.

Table 2: Summary of Parameters for compute_cartesian_path

Parameter	Documentation	Usage Recommendations
eef_step	Max step size in meters between end effector configurations of consecutive points in the result trajectory.	This value should be kept relatively small (0.001 – 0.01). Experiment with and visualize the effect of different values.
jump_threshold	Maximum allowed change in distance in the configuration space of the robot.	This parameter seems to have no noticeable effect.

These two parameters may need to be set to default values, or changed depending on the desired performance of the playback system.

It was difficult to debug the system because the method used was trial and error on the Baxter robot. It is recommended that instead the system is debugged using visualization. For example,

trajectories can be visualized for different values of eef_step and jump_threshold in order to get a better understanding of their effects.

Task:

Create a launch file (or a python script) which runs enable_robot, joint_trajectory_action_server and move_group.launch

Details:

It is inconvenient to run so many commands before being able to use cartesian_playback.py. This process could be streamlined if it was centralized in a single launch file or python script.

Task:

Generalize the module to work with both arms.

Details:

Currently the module is designed to only work with the right arm. This limits the capabilities of the system. Ideally, the system should be generalized so that plans can be computed and executed easily on either arm.

5.4 Conclusion

The trajectory playback and recording systems are a fundamental component of the overall software system. These two modules provide the tools from which the algorithms are tested and demos are created. Although these modules are relatively simple, a large quantity of experimentation was required in order to get them working as intended. Work on these modules is a continuous process, and modifications have been ongoing over the entire course of the project. These modifications include bug fixes and functional improvements. Overall, the modules are a valuable contribution to the project and without them the project could not proceed.

6 CONCLUDING REFLECTIONS

(Author: Schweigert)

Our project proceeded a bit slower in the beginning than we had expected. Part of the reason for this was that we needed to complete literature review before finalizing our topic. Additionally, a large amount of time was allocated to configuring an environment in which we could develop our software. We also had to become familiar with the ROS framework and the Baxter SDK. Although all of these tasks resulted in a bit of a slow start to the project, they also forced the team to improve our skills. Through literature review, for example, we obtained a better understanding of the state of the art as well as stronger scientific reading skills. While configuring our environment, we strengthened our skills in robot-interfacing and Python programming.

The team also learned a great deal about how to manage a technical project. We learned that it was very important to have a person overseeing the completion of each upcoming individual task. If a group of people were assigned to oversee a task, there is a tendency for them to assume that the other members of the group will be working towards completion of the task. As a result, completion of the task might be delayed. In contrast, if a single individual is in charge of a task, they cannot make assumptions about other people doing the required work. Therefore, centralizing the management of a task can make individuals more motivated to work and this helps the project proceed more quickly.

A similar realization was the fact that each group member should have a concrete task that they are working towards at any given time. If any group member did not have an action item, then they should be given one as soon as possible because otherwise they will inevitably stop working. The result is that time is wasted, and the project proceeds more slowly.

Going forward, the software we have developed in our project should be used for future projects with similar research goals. Core functions have already been implemented, so a new group can proceed more quickly with implementation and analysis of algorithms. These core functions concern interfacing with the robot and basic ROS utilities. For example, they allow trajectory data to be easily be recorded or executed on the robot. We have also implemented and tested dynamic time warping and Gaussian mixture models. A new group will not have to re-implement these algorithms, and can instead focus on the implementation of novel algorithms.

In order to continue the project and reuse the code, the next group must first get a good understanding of the existing system that we have developed. Luckily, this task is made easy. Care has been taken to ensure that the code is well documented and the design of the system is clear and easy to understand. The code is split up into many different Python files, which allows for clear organization. All of these design techniques facilitate easy interpretation of the software.

References

(Strategy References:)

Bi, F. & Mac, R. (2015). Drone Maker 3D Robotics Raises $50 Million in Latest Round. *Forbes*. Retrieved from http://www.forbes.com/sites/frankbi/2015/02/26/drone-maker-3d-robotics-raises-50-million-in-latest-round/

Biesada, A. (n.d.). Hoover's Company Profiles: iRobot Corporation. *Hoover's, Inc.* Retrieved from http://subscriber.hoovers.com/H/company360/fulldescription.html?companyId=132607000000000

Brault, M. (2012). Americans With Disabilities: 2010 - Household Economic Studies. *Economics and Statistics Administration, US Department of Commerce*. Retrieved from http://www.census.gov/prod/2012pubs/p70-131.pdf

Censky, A. (2011). Older Americans are 47 times richer than young. *CNN Money - The New American Dream*. Retrieved from http://money.cnn.com/2011/11/07/news/economy/wealth_gap_age/

Crompton, J. (2014). IBISWorld Industry Report 33399: Power Tools & Other General Purpose Machinery Manufacturing in the US. *IBISWorld.* Retrieved from http://www.ibis.com

Ellis, B. (2013). Nursing home costs top $80,000 a year. *CNNMoney (New York)*. Retrieved from http://money.cnn.com/2013/04/09/retirement/nursing-home-costs/

Hoover's, Inc. (n.d.). Hoover's Company Profiles: Clearpath Robotics Inc. *Hoover's, Inc.* Retrieved from http://subscriber.hoovers.com/H/company360/overview.html?companyId=244260399

Hoover's, Inc. (n.d.). Industry Report: Home Health Care Services. *Hoover's, Inc*. Retrieved from http://subscriber.hoovers.com/H/industry360/overview.html?industryId=1383

McMillan, R. (2015). Now We Can Build Autonomous Killing Machines. *Wired*. Retrieved from http://www.wired.com/2015/02/can-now-build-autonomous-killing-machines-thats-bad-idea/

Owano, N. (2011). Willow Garage slashes price (and arm) of PR2 robot for research. *PhysOrg*. Retrieved from http://phys.org/news/2011-08-willow-garage-slashes-price-arm.html

O'Donnell, F. (2015). Issues and Insights. *Mintel: Senior Lifestyles - US - December 2013*. Retrieved from http://academic.mintel.com/display/689700/

Ortman, J. M., Velkoff, V. A., & Hogan, H. (2014). An Aging Nation: The Older Population in the United States. *Economics and Statistics Administration, US Department of Commerce*. Retrieved from http://www.census.gov/prod/2014pubs/p25-1140.pdf

Porter, M. E. (2008). The Five Competitive Forces That Shape Strategy. *Harvard Business Review 86*(1), 25-40.

Rethink Robotics (2015). Build a Baxter. *Rethink Robotics.* Retrieved from http://www.rethinkrobotics.com/build-a-bot/baxter/

Tobe, F. (2013). A glimpse at our robotic future: 235 start-ups reviewed. *Robohub*. Retrieved from http://robohub.org/a-glimpse-at-our-robotic-future-235-start-ups-reviewed/

(IP References:)

Barajas, L. G., Martinson, E., Payton, D. W., & Uhlenbrock, R. M. (2014). Method and system for training a robot using human-assisted task demonstration. *U.S. Patent No. 8,843,236*. Retrieved from https://www.google.com/patents/US8843236

Billard, A., Calinon, S., Dillmann, R., & Schaal, S. (2008). Robot Programming by Demonstration. *Springer Handbook of Robotics*, 1371-1394.

Brown & Michaels, PC. (2006). How do I read a patent? - the Claims. *Brown & Michaels*. Retrieved from http://www.bpmlegal.com/howtopat5.html

Klein, A. R. (2015). Intellectual Property Basics for Technology Companies. *Latham & Watkins*.

Ravindran, B., Barto, A. G., & Mathew, V. (2007, January). Deictic Option Schemas. In *IJCAI* (pp. 1023-1028).

(Technical Contributions References:)

Abbeel, P. & Y. Ng, A. (2015). Apprenticeship Learning via Inverse Reinforcement Learning. International Conference on Machine Learning 21. Retrieved from http://ai.stanford.edu/~ang/papers/icml04-apprentice.pdf.

Billard, A., Calinon S., Dillmann, R., & Schaal, S. (2008). Robot Programming by Demonstration. Springer Handbook of Robotics. (pp. 1371-1394).

Calinon, S., Guenter, F., & Billard, A. (2007). On Learning, Representing, and Generalizing a Task in a Humanoid Robot. IEEE Transactions on Systems, Man, and Cybernetics 37 (2). (pp. 286-298).

Open Source Robotics Foundation. (2015). MoveIt documentation. Retrieved from http://docs.ros.org/api/moveit_ros_planning_interface/.

Su, J. (2015). Accuracy Comparison and Analysis. Unpublished manuscript, Department of EECS, University of California, Berkeley.

Sun, J. (2015). Accuracy Estimation of the Baxter Robot. Unpublished manuscript, Department of EECS, University of California, Berkeley.

Srinivasan, S. (2015). The Pipeline for Learning by Demonstration. Unpublished manuscript, Department of EECS, University of California, Berkeley.