University of Glasgow | School of Computing Science

Honours Individual Project Dissertation

Training the Baxter Research Robot using a VR interface based on the HTC Vive

Sean A. Skilling

January 22, 2018

Abstract

Every abstract follows a similar pattern. Motivate; set aims; describe work; explain results.

"XYZ is bad. This project investigated ABC to determine if it was better. ABC used XXX and YYY to implement ZZZ. This is particularly interesting as XXX and YYY have never been used together. It was found that ABC was 20% better than XYZ, though it caused rabies in half of subjects."

Education Use Consent

Contents

1 Introduction

Why should the reader care about what are you doing and what are you actually doing?

1.1 Guidance

Motivate first, then state the general problem clearly.

1.2 Writing guidance

1.2.1 Who is the reader?

This is the key question for any writing. Your reader:

- is a trained computer scientist: *don't explain basics.*
- has limited time: *keep on topic.*
- has no idea why anyone would want to do this: *motivate clearly*
- might not know *anything* about your project in particular: *explain your project.*
- but might know precise details and check them: *be precise and strive for accuracy.*
- doesn't know or care about you: *personal discussions are irrelevant.*

Remember, you will be marked by your supervisor and one or more members of staff. You might also have your project read by a prize-awarding committee or possibly a future employer. Bear that in mind.

Citation styles

- If you are referring to a reference as a noun, then cite it as: "? discusses the role of language in political thought."
- If you are referring implicitly to references, use: "There are many good books on writing (???)."

1.2.2 Plagiarism warning

WARNING

If you include material from other sources without full and correct attribution, you are commiting plagiarism. The penalties for plagiarism are severe. Quote any included text and cite it correctly. Cite all images, figures, etc. clearly in the caption of the figure.

2 Background

The main aim of this project is to take the existing implementation of an immersive VR interface for the the teleoperation of the Baxter robot and simplify it. In order to justify the approach taken in the context of the study, analysis of relevant work within available literature and conducted by the same institution is required. In particular, relevant work in Spacial Presence in VR, Systems for Tracking Human Movement and Human-Robot Mapping will be presented.

2.1 Spatial Presence in VR

Due to the relative newness of the commercial VR field, the majority of research into immersion in virtual reality presents it in context of video games, such as Bracken and Skalski (2010). Despite this specification, a great deal of insight can be drawn from these studies in regards to immersive techniques. In particular, these studies are the origin of the term "Spatial Presence" which describes the feeling of physically existing in a virtual space (Weibel and Wissmath (2011)). In some teleoperation contexts, this is often referred to as "telepresence".

This labelling of immersion as "spatial presence" or "telepresence" could be argued as being misleading. While telepresence has been identified as a fundamental component in driving immersion (Siebert and Shafer (2017)), there are several other factors that need to be taken into account such as the accuracy of movement tracking and the latency in the system. Failing to account for these key factors could ruin an otherwise perfectly realistic virtual world and break the immersion for the user.

It is also wise to compare how immersive virtual reality setups are in comparison to more standard setups. In research by Siebert and Shafer (2017), they compare a VR HMD with traditional monitors for creating immersion for the user in video games. The results show a significant increase in spacial presence when using a VR system compared to standard monitor displays. However, VR systems do have their own drawbacks. Specifically, VR interfaces have been known to cause some users to experience a feeling similar to motion sickness, dubbed "cybersickness" (Vinson et al. (2012)). While the focus of this project is not on cybersickness, it should be noted that it was cited as an obstacle for VR adoption and for the application of this technology in certain aspects.

2.2 Systems for Tracking Human Movement

There are multiple different ways of tracking and record movement, but the two covered here are full-body tracking and room-scale tracking. Full-body tracking devices, such as the Kinect produced by Microsoft, use depth cameras to create a real-time interface for teleoperation. The Kinect in particular has often been praised in studies for its tracking abilities without relying on complex sensors, achieving this through recognizing the joints and poses of the user captured from depth images. Figure 2.1 shows the joints which are tracked in this process. This isn't enough for the Kinect to be able to grant full telepresence on its own since it lacks a display, although some studies address this limitation by pairing it with a VR headset such as the Oculus Rift (Hsu et al. (2013)). The Kinect also has issues with tracking; Other studies, such as the study by Xu et al. (2016), encounter the same display shortcoming when using the Myo armbands and also require an additional VR headset to achieve telepresence. The skeletal tracking of the Kinect also has several flaws, such as the sun's rays interfering with the infrared sensors and an inability to accurately track users who are either not facing the Kinect or too far away.

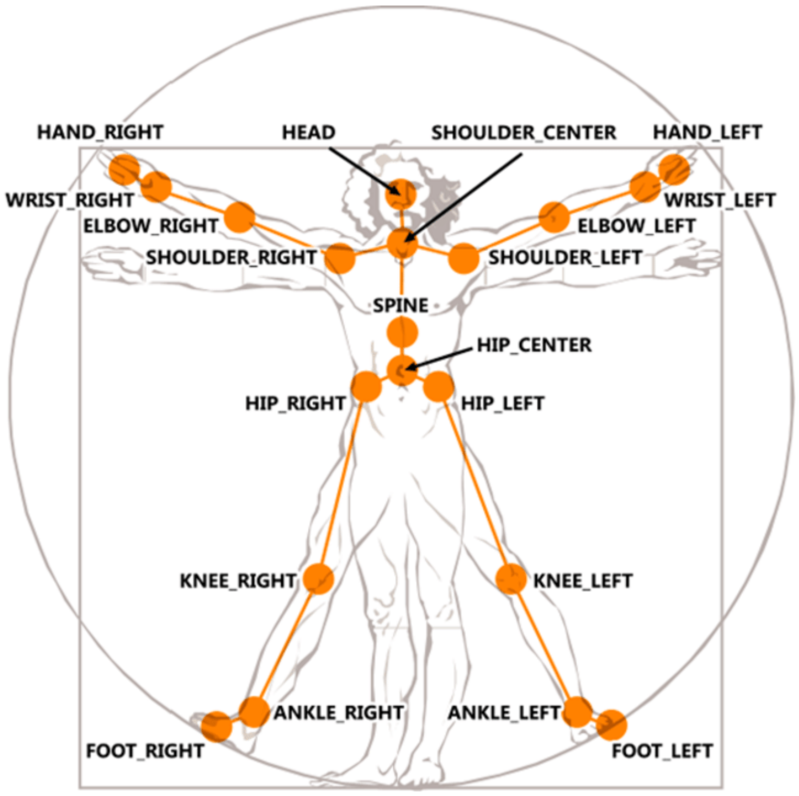


Figure 2.1: *The joints tracked by the Kinect's skeletal tracking system. Taken from Hsu et al. (2013)*

VR systems such as the HTC Vive, on the other hand, utilises room-scale tracking through the use of stationary sensors which, along with sensors built in to the various components of the system (discussed in more detail in Section 3.1) create a 3D co-ordinate system that allows the system to represent the user's orientation and position within a "play area" with diagonals of 5 metres. The HTC Vive system also already contains a display in the form of the Vive HMD, negating the reliance on an additional system for the display that the full-body systems discussed all had. The Oculus Rift, the Vive's main competitor, is another VR system which can be similarly be used for teleoperation purposes. Such is the case with a study conducted at the University of Glasgow by Vanik (2017), which justified the system's usability for tracking and mapping to the Baxter robot. However, the study fails to provide details for the system's accuracy.

A study by Niehorster et al. (2017) was conducted analyse the HTC Vive and its capabilities in tracking human movements. It found that the Vive combines efficient and accurate tracking with low latency, but also that there were several shortcomings with the system. Notably, the Vive tracks its virtual space by the "ground plane" which has its distance from the headset calibrated during Room Setup and is always taken from the base of the headset which creates innaccuracies when reading the roll and pitch data. While this can be fixed by hardcoding in an offset, this is lost whenever the tracking has to be re-calibrated. However, this is a result of moving whilst operating the system; seated and standing users can mitigate this problem to an extent.

2.3 Human-Robot Mapping

There are multiple models of mapping the movements of a human user to a robot. In the context of this project there are two particular models which bear further analysis: robocentric and egocentric. The robocentric model has the user co-existing in a virtual environment along with the robot as they control it, with the robot and the user existing as separate entities in this space. However, the egocentric model describes the system where the user and robot exist as a single entity in the virtual space - the user essentially "becomes" the robot when interfacing with it. This model minimises the coupling between the user and robot in an attempt to mitigate the discomfort and limitations caused by the robocentric model of control.

In a study by Lipton et al. (2018), which compared an egocentric approach they dubbed the "homunculus model" against several other models, the mimicking model was found to have the highest degree of immersion due the virtual space superimpoding the user's state and movements onto the robot. This is partly why previous projects in this field by this institution used the mimicking model such as the study by Vanik (2017). This model works best on robots with a humanoid form such as Baxter. This is due to the similarity in shape and resulting ease in mapping movements, being mostly direct with some differences due to differing numbers of joints in the robot's limbs.

2.4 Summary of Background

Virtual reality systems, though primarily designed for playing video games, are beginning to be introduced into robotics projects and while the findings aren't all applicable to robotics several results can be helpful to the field. In particular, the research into immersion and cybersickness can be used to better understand these aspects of the field, though this is mostly outside the focus of this project. In the particular case of cybersickness, the benefits of VR vastly outweigh the drawbacks and as such we accept it as a risk.

When using virtual reality systems to track user's movements, devices with integrated displays such as the HTC Vive and Oculus Rift are far more useful due to not needing to employ a second device with a screen even when taking into account full-body tracking devices compared to room-scale tracking. In addition, since the Baxter robot used in the project consists of just a torso and head and is incapable of movement, full-body tracking is excessive for this project and the room-scale devices are acceptable.

Both the robocentric and egocentric models of teleoperation have their advantages as seen in the project by Lipton et al. (2018). However, due to the focus on immersion when the user is interacting with the robot, the egocentric model is more appropriate as it allows the user to "see" out of the robot's eyes (through the use of the integrated HMD) and become the robot. In addition to this, the mimicking model will also be applied when dealing with the mapping from user to robot because of its increased speed of development and ability to achieve telepresence in contrast to the homunculus model. This is in addition to the latter system having far more complexity, further justifying the mimicking model.

3 | Hardware and Software Dependencies

With the background details and history of the project covered in the previous chapter, this chapter will detail the hardware and software technologies that were used in the project as well as any issues that arise from using these technologies in combination with one another.

3.1 Baxter

Baxter is an industrial robot, built and initially released by Rethink Robotics in September 2011. Mimicking the appearance of a human torso, arms and head atop a stand, Baxter's primary purpose is in automating industrial sized processes at an affordable cost. It comes with its own Software Development Kit (SDK) containing the various tools for interfacing with the different components of the robot. In the context of this project the most relevant of these packages is the Inverse Kinematics (IK) Service, which calculates a path of movement for its arm from its current location to the desired point given just that end point.

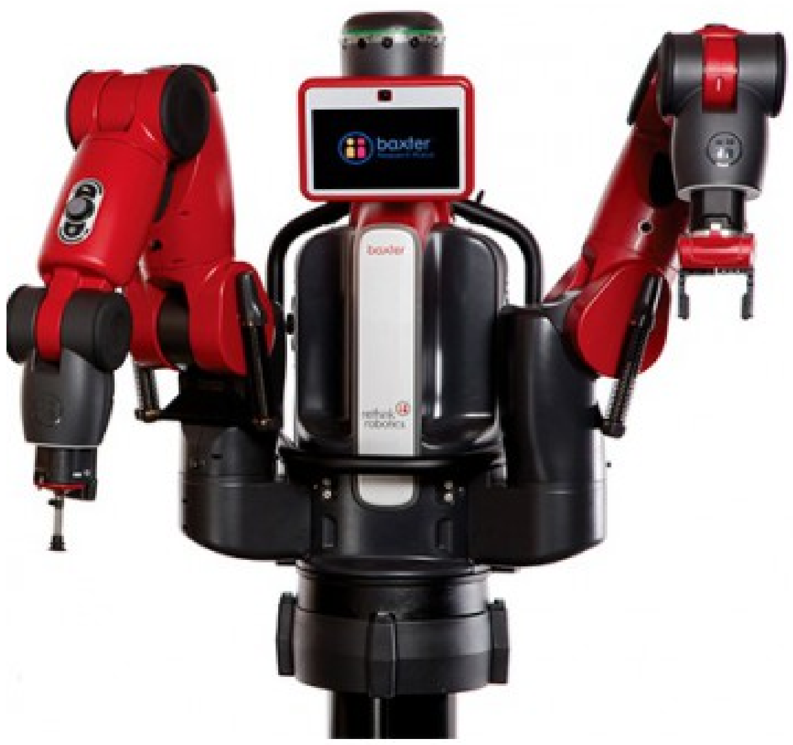


Figure 3.1: *The Baxter robot.*

For the purpose of mapping from human users to robots, Baxter isn't an perfect candidate due to a different number of joints and degrees of freedom to the user - human arms only have six degrees of freedom due to their joint structure, while Baxter's arms have seven degrees of freedom. These joints (show in Figure 3.2) are also positioned differently than those in a human, allowing it to reach poses that a human is incapable of reaching.

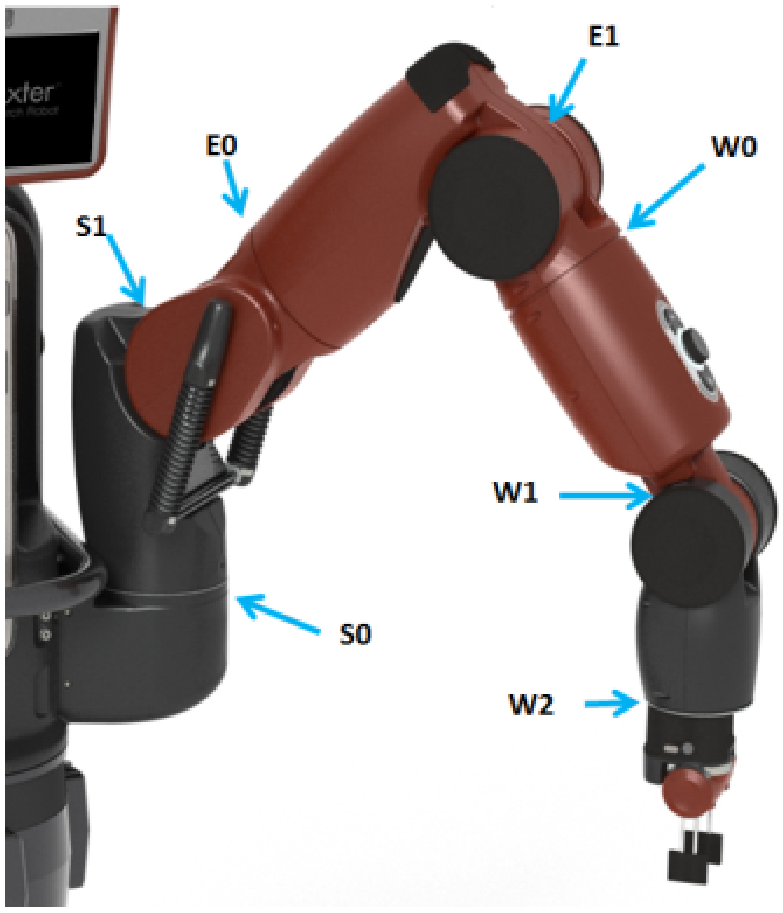


Figure 3.2: *The seven joints in Baxter's arm. The joints labelled S are the shoulder joints, E joints are the elbow joints and W joints are the wrist joints. Notably, Baxter has an extra degree of freedom in its wrist compared to a human*

This causes some issue with the mapping of human movements to Baxter. Using the IK Services, the movements are executed by calculating the path to the endpoint which could be inconsistent between the user and the robot. This may lead to instances where the point mapped doesn't match and fails to mimic movements exactly. There is also the consideration that Baxter wasn't designed for human-robot mapping, instead having industry tasks in mind.

Despite the above potentially causing issues during the project, Baxter was chosen for this project due to his affordability compared other similar robots (costing Âč19,000 compared to other industrial robots costing upwards of Âč37,000) and his design being close enough to shape of a human for accurate mapping. In addition, the Baxter robot was already available, saving a large sum of money purchasing a robot. Baxter also has a Gazebo simulation (a virtual environment containing a functioning replica of Baxter) which removes the need to have the Baxter robot on hand for development and allows work to be done away from his location.

3.2 ROS

ROS (**R**obotic **O**perating **S**ystem) is best described as robotics middleware with its main support base being for Ubuntu Linux systems. It contains code packages for various programming languages which allow many different varieties of robots to be driven and controlled, including Baxter. Since ROS (and by extension Baxter) is restricted mostly to Linux, this is the major limiting factor in our options for the other components of the project (there do exist "Experimental" packages for Windows, MacOS and Fedora Linux systems, but that is outside of the scope of this project).

A major component of working with ROS is the use of subscriber nodes and publisher nodes. Publisher nodes collect data from various functions and "broadcast" them into the system, while subscriber nodes are constantly listening for their specific function calls in these broadcasts. When the published nodes broadcast is picked up by the subscriber node, the data is then handled by any functions pertaining to them by the subscriber node before handing off that data to the target (Baxter in this case)

ROS subscribers and publishers can be written in a variety of programming languages, with the three main client libraries being C++, Python and Lisp. Python was chosen for this project due to its ease of readability and faster development due to simpler code. In addition, the developer has experience in Python but no experience in C++ or Lisp making the decision a straightforward one.

3.3 HTC Vive

The HTC Vive is a virtual reality headset system developed by smartphone developer HTC in partnership with Valve Corporation, a long-standing fixture in the videogame industry responsible for both highly popular games and one of, if not the most used digital storefronts in Steam. The system consists of three components which work in tandem to achieve accurate room-scale tracking of a user's movement in 3D space as seen in Figure 3.3: the HMD (**H**ead-**M**ounted **D**isplay), the controllers and the base stations. By having the HMD and controllers in view of both base stations, the systems is able to track the users movements with relatively high accuracy using infrared detectors and emitters respectively.

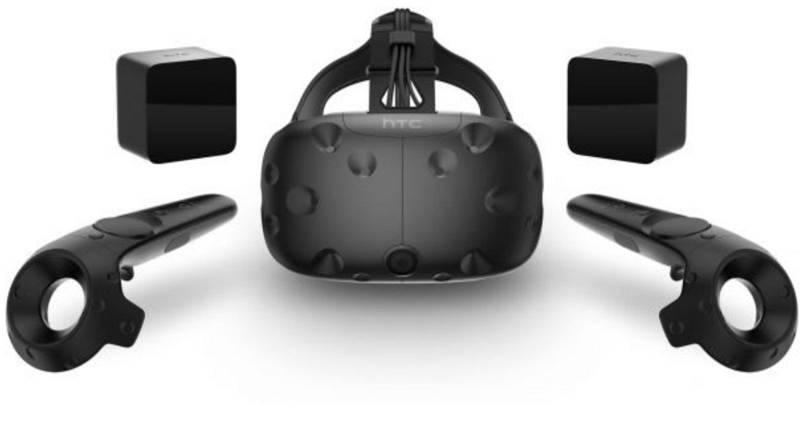


Figure 3.3: *The major components of the HTC Vive: the HMD (center), the controllers (front-sides) and the base stations (back-sides).*

The main reason the HTC Vive was chosen over other VR systems such as the Oculus Rift used by Peter Vanik in his project is due to its compatibility with Linux systems. The Rift is only supported on Windows at this time and would require an additional system running on Windows alongside the system to run ROS (which is only supported on Linux systems), which goes against the point of the project. The Vive was updated to support Linux in early 2017 [Source], which makes it the ideal choice since one of the aims of this project is to limit the number of required systems to just a single Linux machine. Even so, the post-launch support for the Vive on Linux has been relatively poor - two years after launch performance is still worse on Linux than it is on a system with identical specifications that runs on Windows, and with only a handful of supported games and other software.

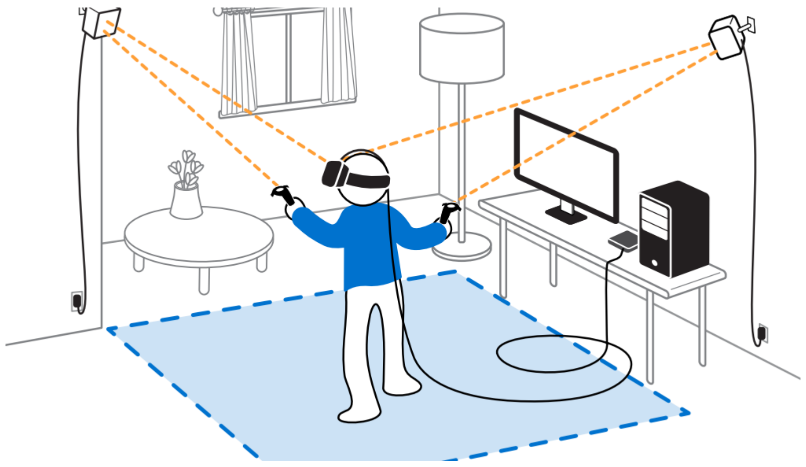


Figure 3.4: *The diagram above shows the intended layout of a room when using the HTC Vive, with both base stations raised into opposing corners of the room.*

The HTC Vive has several other limitations, some of which are more glaring in the context of this project than others. The base model of the Vive is limited in its range of motion by its dependence on the HDMI and USB cables connecting them to the computer unless the hefty cost for a wireless adaptor is deemed worthwhile. In addition, the HMD itself is fairly heavy at roughly 0.5 kilograms and could cause strain or injury over long user sessions. The tracking system also proved surprisingly easy to disturb, as any non-Vive system item that sits within the space between the controllers, the HMD and the base stations would cause the system to lose track of the various components and either desync them or lag heavily. The system is also rather inconvenient to set up as intended, as it requires drilling into the walls to attach the included wall mounts as shown in 3.4 although this can be done by attaching the mounts to microphone or lighting stands for a less permanent fixture.

3.4 SteamVR

Launched simultaneously with the HTC Vive, SteamVR is a virtual reality service created and distributed by Valve to interface with the Vive as well as other VR systems utilising their OpenVR API.

It would be an understatement to describe SteamVR on Linux as being unstable. Most of the defined issues were encountered when trying to set up the HTC Vive, aside from a few Windows-specific errors and crashes, as well as several error codes that were not defined anywhere in the support for SteamVR. Another major issue was that, due to VR on Linux still being in its infancy, most of the fixes and solutions for the issues were either only explained through Windows or didn't work on Linux at all. This led to repeated testing sessions cut short by having to uninstall Steam and SteamVR and reinstalling both, which could take upwards of two hours each time when factoring in various issues with installing Steam itself. In addition to this, a conversion between OpenVR's co-ordinate system and the HTC Vive's co-ordinate system is required due to their differences.

Despite the issues mentioned above, SteamVR was chosen as it was the earliest method of VR on Linux and it was hoped that it would be more stable than the alternative of Proton, which allows the Windows version of Steam games to be run on Linux through a combination of Wine and Windows partitions.

3.5 Summary of Hardware and Software

Due to the integration of Baxter being a key point of the project, choosing a suitable robot wasn't a factor. However, the use of Baxter creates a separate issue as it limits our choices for the other technologies available. ROS is only officially supported on Ubuntu Linux systems , which restricts our option of VR systems. There are two primary options for VR on Linux: using the HTC Vive through SteamVR or through the use of Proton. Due to the pre-existing access to a HTC Vive system and an uncertainty as to whether or not Proton would support the project, it was decided to use the HTC Vive and SteamVR. The following chapter will discuss how the technologies influenced and shaped the design and eventual implementation of the teleoperation system.

4 Design and Implementation

With the technologies covered in the previous chapter, this chapter will cover the iterations of design of the project and their implementation from the first ideas and experiments to the final product, as well as the various issues encountered along the way.

4.1 Initial Work

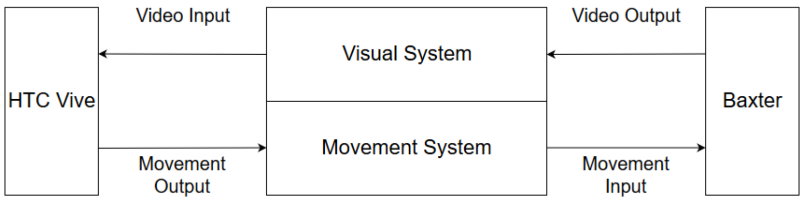


Figure 4.1: The proposed system in its earliest and simplest iteration. As of writing the HTC Vive is the only Linux-compatible VR system – hence why it is specified in the diagram. A two-computer solution could replace the HTC Vive with a different headset.

Since Baxter is only capable of moving its upper body, a full-body tracking system would be excessive even without taking into account the lack of an immersive display in such systems.

4.2 Design

The architecture of the system is shown in a simplified form in Figure 4.1, which details the

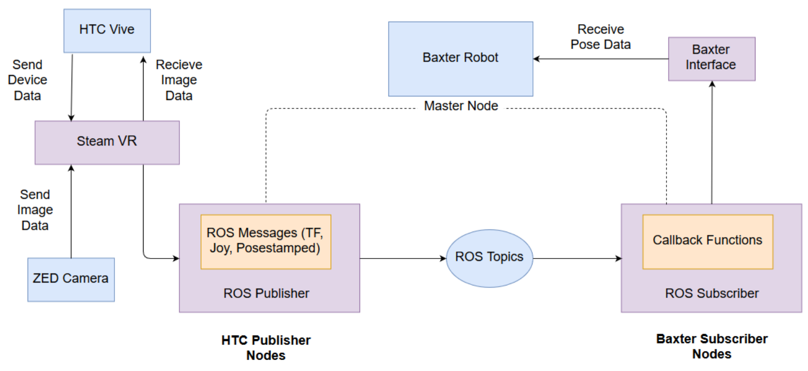


Figure 4.2: The proposed system in its earliest and simplest iteration. As of writing the HTC Vive is the only Linux-compatible VR system – hence why it is specified in the diagram. A two-computer solution could replace the HTC Vive with a different headset.

4.3 Implementation

The issues covered in Chapter 3 were not the only issues that were encountered during the course of the project. Our initial plan was to develop the system on a laptop running Linux for convenience, which would allow work to be done outside of the Baxter lab and increasing the ease of production. However, Steam proved far more difficult to install than we expected due to various inconsistencies between packages that were updated between releases. In addition to this, the ROS package and Baxter simulator also encountered similar issues when installing which we believe was due to the open-source nature of ROS and Gazebo leading to software packages that could easily "tangle themselves up" for lack of a better term. By the time we came to the conclusion that developing on the laptop was too much work, two months had passed without any code having been properly run and tested.

The layout of the Baxter lab proved an issue when it came to setting up the HTC Vive. As detailed in Figure 3.4, the intended setup is to have both sensors in opposite corners of the room in order to provide as close to complete coverage with the sensors as possible. This wasn't possible in the Baxter lab, as two of the four walls of the room are made of glass and therefore are unsuitable for attaching the wall mounts to. The mounts can be used to attach the base stations to C-stands or similar pole-based stands, which was not possible since the mounts were missing from the HTC Vive's box. As a result of this, an alternative setup had to be found. It was discovered that the base stations would work as long as they had line of sight with one another, and so the setup detailed in Figure 4.3 was settled on. It is unknown if this setup is the cause of any of the technical issues experienced during the course of the project.

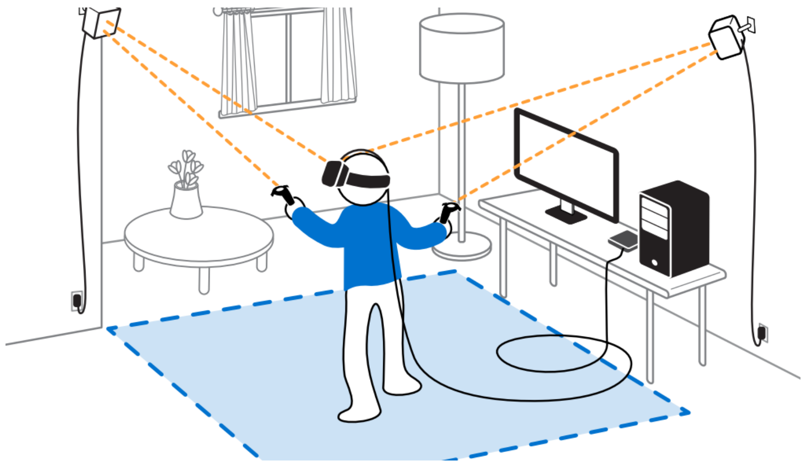


Figure 4.3: *The figure above shows the finalized room layout that was settled upon during the project. With the base stations placed at an angle to the edge of the desk so that they have line of sight with each other, the HTC Vive appears to work as normal. Of important note is that the base stations sit at the edge of the desk, as otherwise the desk blocks any signals that would occur past it.*

Another issue that caused problems for the project was the shutdown of the university network over the Christmas period. This lead to Baxter being disconnected and shut down for most of the development cycle. The obvious issue with this is that there was now no way to run code on the robot as opposed to the simulation and no way to check if any additional quirks with the code would arise. The main place where this caused problems is in the vision aspect, as the simulation does not support the standard camera functions in ROS.

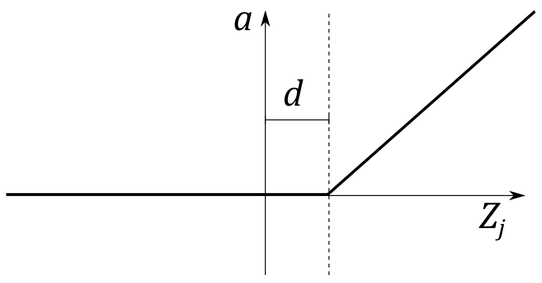


Figure 4.4: *In figure captions, explain what the reader is looking at: "A schematic of the rectifying linear unit, where a is the output amplitude, d is a configurable dead-zone, and Z_j is the input signal", as well as why the reader is looking at this: "It is notable that there is no activation* at all *below 0, which explains our initial results."* ***Use vector image formats (.pdf) where possible****. Size figures appropriately, and do not make them over-large or too small to read.*

4.4 Design

How is this problem to be approached, without reference to specific implementation details?

4.4.1 Guidance

Design should cover the abstract design in such a way that someone else might be able to do what you did, but with a different language or library or tool.

4.5 Implementation

What did you do to implement this idea, and what technical achievements did you make?

4.5.1 Guidance

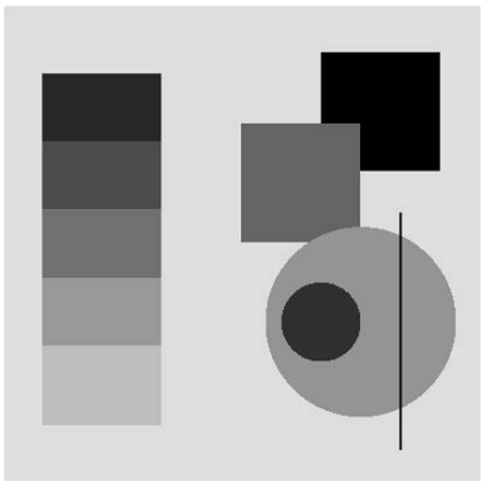
You can't talk about everything. Cover the high level first, then cover important, relevant or impressive details.

4.6 General points

These points apply to the whole dissertation, not just this chapter.

4.6.1 Figures

Always refer to figures included, like Figure 4.4, in the body of the text. Include full, explanatory captions and make sure the figures look good on the page. You may include multiple figures in one float, as in Figure 4.5, using `subcaption`, which is enabled in the template.


(a) *Synthetic image, black on white.*

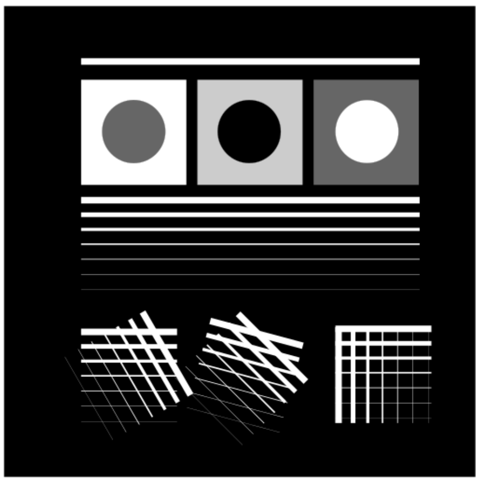

(b) *Synthetic image, white on black.*

Figure 4.5: *Synthetic test images for edge detection algorithms. (a) shows various gray levels that require an adaptive algorithm. (b) shows more challenging edge detection tests that have crossing lines. Fusing these into full segments typically requires algorithms like the Hough transform. This is an example of using subfigures, with* `subref`*s in the caption.*

4.6.2 Equations

Equations should be typeset correctly and precisely. Make sure you get parenthesis sizing correct, and punctuate equations correctly (the comma is important and goes *inside* the equation block). Explain any symbols used clearly if not defined earlier.

For example, we might define:

$$\hat{f}(\xi) = \frac{1}{2}\left[\int_{-\infty}^{\infty} f(x)e^{2\pi i x\xi}\right], \tag{4.1}$$

where $\hat{f}(\xi)$ is the Fourier transform of the time domain signal $f(x)$.

4.6.3 Algorithms

Algorithms can be set using `algorithm2e`, as in Algorithm 1.

```
Data: $f_X(x)$, a probability density function returing the density at $x$.
$\sigma$ a standard deviation specifying the spread of the proposal distribution.
$x_0$, an initial starting condition.
Result: $s = [x_1, x_2, \ldots, x_n]$, $n$ samples approximately drawn from a distribution with PDF $f_X(x)$.
begin
    $s \longleftarrow []$
    $p \longleftarrow f_X(x)$
    $i \longleftarrow 0$
    while $i < n$ do
        $x' \longleftarrow \mathcal{N}(x, \sigma^2)$
        $p' \longleftarrow f_X(x')$
        $a \longleftarrow \frac{p'}{p}$
        $r \longleftarrow U(0, 1)$
        if $r < a$ then
            $x \longleftarrow x'$
            $p \longleftarrow f_X(x)$
            $i \longleftarrow i + 1$
            append $x$ to $s$
        end
    end
end
```

Algorithm 1: The Metropolis-Hastings MCMC algorithm for drawing samples from arbitrary probability distributions, specialised for normal proposal distributions $q(x'|x) = \mathcal{N}(x, \sigma^2)$. The symmetry of the normal distribution means the acceptance rule takes the simplified form.

4.6.4 Tables

If you need to include tables, like Table 4.1, use a tool like https://www.tablesgenerator.com/ to generate the table as it is extremely tedious otherwise.

4.6.5 Code

Avoid putting large blocks of code in the report (more than a page in one block, for example). Use syntax highlighting if possible, as in Listing 4.1.

Table 4.1: *The standard table of operators in Python, along with their functional equivalents from the* `operator` *package. Note that table captions go above the table, not below. Do not add additional rules/lines to tables.*

Operation	**Syntax**	**Function**
Addition	`a + b`	`add(a, b)`
Concatenation	`seq1 + seq2`	`concat(seq1, seq2)`
Containment Test	`obj in seq`	`contains(seq, obj)`
Division	`a / b`	`div(a, b)`
Division	`a / b`	`truediv(a, b)`
Division	`a // b`	`floordiv(a, b)`
Bitwise And	`a & b`	`and_(a, b)`
Bitwise Exclusive Or	`a ^b`	`xor(a, b)`
Bitwise Inversion	`~a`	`invert(a)`
Bitwise Or	`a | b`	`or_(a, b)`
Exponentiation	`a ** b`	`pow(a, b)`
Identity	`a is b`	`is_(a, b)`
Identity	`a is not b`	`is_not(a, b)`
Indexed Assignment	`obj[k] = v`	`setitem(obj, k, v)`
Indexed Deletion	`del obj[k]`	`delitem(obj, k)`
Indexing	`obj[k]`	`getitem(obj, k)`
Left Shift	`a <<b`	`lshift(a, b)`
Modulo	`a % b`	`mod(a, b)`
Multiplication	`a * b`	`mul(a, b)`
Negation (Arithmetic)	`- a`	`neg(a)`
Negation (Logical)	`not a`	`not_(a)`
Positive	`+ a`	`pos(a)`
Right Shift	`a >>b`	`rshift(a, b)`
Sequence Repetition	`seq * i`	`repeat(seq, i)`
Slice Assignment	`seq[i:j] = values`	`setitem(seq, slice(i, j), values)`
Slice Deletion	`del seq[i:j]`	`delitem(seq, slice(i, j))`
Slicing	`seq[i:j]`	`getitem(seq, slice(i, j))`
String Formatting	`s % obj`	`mod(s, obj)`
Subtraction	`a - b`	`sub(a, b)`
Truth Test	`obj`	`truth(obj)`
Ordering	`a <b`	`lt(a, b)`
Ordering	`a <= b`	`le(a, b)`

```python
def create_callahan_table(rule="b3s23"):
    """Generate the lookup table for the cells."""
    s_table = np.zeros((16, 16, 16, 16), dtype=np.uint8)
    birth, survive = parse_rule(rule)

    # generate all 16 bit strings
    for iv in range(65536):
        bv = [(iv >> z) & 1 for z in range(16)]
        a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p = bv

        # compute next state of the inner 2x2
        nw = apply_rule(f, a, b, c, e, g, i, j, k)
        ne = apply_rule(g, b, c, d, f, h, j, k, l)
        sw = apply_rule(j, e, f, g, i, k, m, n, o)
        se = apply_rule(k, f, g, h, j, l, n, o, p)

        # compute the index of this 4x4
        nw_code = a | (b << 1) | (e << 2) | (f << 3)
        ne_code = c | (d << 1) | (g << 2) | (h << 3)
        sw_code = i | (j << 1) | (m << 2) | (n << 3)
        se_code = k | (l << 1) | (o << 2) | (p << 3)

        # compute the state for the 2x2
        next_code = nw | (ne << 1) | (sw << 2) | (se << 3)

        # get the 4x4 index, and write into the table
        s_table[nw_code, ne_code, sw_code, se_code] = next_code

    return s_table
```

Listing 4.1: *The algorithm for packing the 3×3 outer-totalistic binary CA successor rule into a $16 \times 16 \times 16 \times 16$ 4 bit lookup table, running an equivalent, notionally 16-state 2×2 CA.*

5 Evaluation

How good is your solution? How well did you solve the general problem, and what evidence do you have to support that?

5.1 Movement Lag

5.2 Movement Accuracy

5.3

5.4 Guidance

- Ask specific questions that address the general problem.
- Answer them with precise evidence (graphs, numbers, statistical analysis, qualitative analysis).
- Be fair and be scientific.
- The key thing is to show that you know how to evaluate your work, not that your work is the most amazing product ever.

5.5 Evidence

Make sure you present your evidence well. Use appropriate visualisations, reporting techniques and statistical analysis, as appropriate.

If you visualise, follow the basic rules, as illustrated in Figure 5.1:

- Label everything correctly (axis, title, units).
- Caption thoroughly.
- Reference in text.
- **Include appropriate display of uncertainty (e.g. error bars, Box plot)**
- Minimize clutter.

See the file `guide_to_visualising.pdf` for further information and guidance.

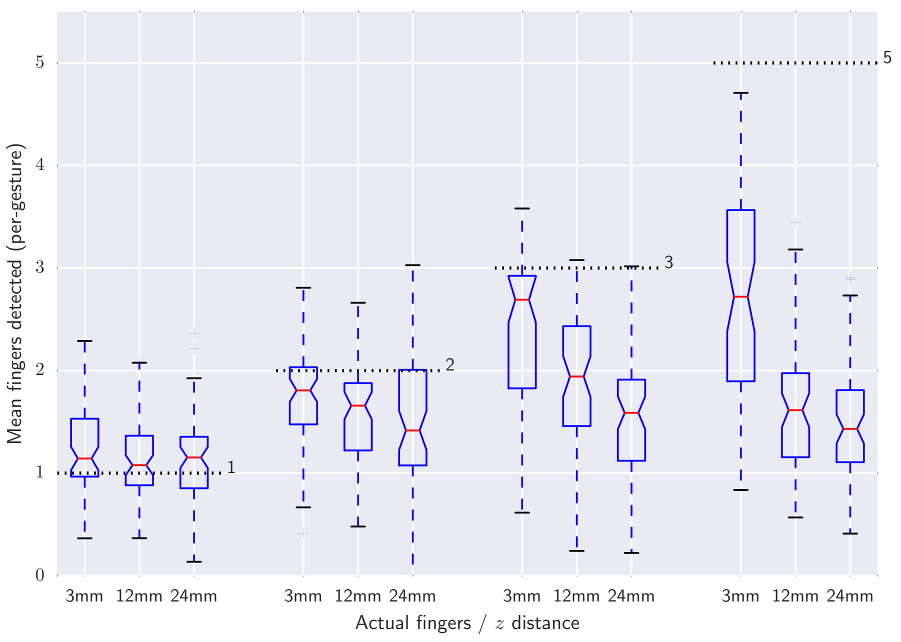


Figure 5.1: *Average number of fingers detected by the touch sensor at different heights above the surface, averaged over all gestures. Dashed lines indicate the true number of fingers present. The Box plots include bootstrapped uncertainty notches for the median. It is clear that the device is biased toward undercounting fingers, particularly at higher z distances.*

6 Conclusion

Summarise the whole project for a lazy reader who didn't read the rest (e.g. a prize-awarding committee).

6.1 Guidance

- Summarise briefly and fairly.
- You should be addressing the general problem you introduced in the Introduction.
- Include summary of concrete results ("the new compiler ran 2x faster")
- Indicate what future work could be done, but remember: **you won't get credit for things you haven't done**.

A Appendices

Typical inclusions in the appendices are:

- Copies of ethics approvals (required if obtained)
- Copies of questionnaires etc. used to gather data from subjects.
- Extensive tables or figures that are too bulky to fit in the main body of the report, particularly ones that are repetitive and summarised in the body.
- Outline of the source code (e.g. directory structure), or other architecture documentation like class diagrams.
- User manuals, and any guides to starting/running the software.

Don't include your source code in the appendices. It will be submitted separately.

6 Bibliography

C. C. Bracken and P. Skalski. *Immersed in media: Telepresence in everyday life.* Routeledge, 2010.

H.-H. Hsu, Y. Chiou, Y.-R. Chen, and T. Shih. Using kinect to develop a smart meeting room. pages 410–415, 09 2013. ISBN 978-1-4799-2509-4. doi: 10.1109/NBiS.2013.66.

J. I. Lipton, A. J. Fay, and D. Rus. Baxter's homunculus: Virtual reality spaces for teleoperation in manufacturing. *IEEE Robotics and Automation Letters*, 3(1):179–186, Jan 2018. ISSN 2377-3766. doi: 10.1109/LRA.2017.2737046.

D. C. Niehorster, L. Li, and M. Lappe. The accuracy and precision of position and orientation tracking in the htc vive virtual reality system for scientific research. *i-Perception*, 8(3): 2041669517708205, 2017. doi: 10.1177/2041669517708205. URL `https://doi.org/10.1177/2041669517708205`.

J. Siebert and D. M. Shafer. Control mapping in virtual reality: effects on spatial presence and controller naturalness. 2017. URL `https://link.springer.com/article/10.1007/s10055-017-0316-1`.

P. Vanik. Teleoperation of the baxter robot using oculus touch and zed camera. 2017.

N. G. Vinson, J.-F. Lapointe, A. Parush, and S. Roberts. Cybersickness induced by desktop virtual reality. In *Proceedings of Graphics Interface 2012*, GI '12, pages 69–75, Toronto, Ont., Canada, Canada, 2012. Canadian Information Processing Society. ISBN 978-1-4503-1420-6. URL `http://dl.acm.org/citation.cfm?id=2305276.2305288`.

D. Weibel and B. Wissmath. Immersion in computer games: The role of spatial presence and flow. *Int. J. Comput. Games Technol.*, 2011:6:6–6:6, Jan. 2011. ISSN 1687-7047. doi: 10.1155/2011/282345. URL `http://dx.doi.org/10.1155/2011/282345`.

Y. Xu, C. Yang, P. Liang, L. Zhao, and Z. Li. Development of a hybrid motion capture method using myo armband with application to teleoperation. pages 1179–1184, 08 2016. doi: 10.1109/ICMA.2016.7558729.