Hand Gesture Estimation Using a Convolutional Neural Network and a Single RGB Camera

Student Name: T.M.A. Hui

Supervisor Name: G. Koulieris

Submitted as part of the degree of [MEng Computer Science] to the

Board of Examiners in the Department of Computer Sciences, Durham University

Abstract —

Context/Background

Currently, users are bound to a physical controller when using computers, and techniques that use gesture/pose estimation to control computers are either poorly performing, expensive, or inconvenient.

Aims

The aim is to create an affordable Human Computer Interaction solution without a physical controller, to provide a more immersive, convenient, and hygienic method of interacting with computers. This is done with a single basic RGB camera instead of a more complex sensor, such as an RGB-D camera.

Method

The aims are achieved within the constraints of a singular RGB camera setup, by using a mixture of computer vision techniques to segment a hand from an image, predicting a gesture from the image using a Convolutional Neural Network, and mapping the gestures to controller inputs. Using a regular RGB camera makes this solution accessible to most laptops, and is inexpensive compared to more complex sensors.

Results

The results show consistency and reasonably good usability in certain configurations, specifically where a glove is used, as shown by the high F1 scores for each mouse action. Glove-less usage is difficult and inconsistent. The quality of the CNN predictions are high.

Conclusions

The solution has much room for improvement, in terms of improving the segmentation capabilities for a glove-less configuration. Future endeavours include multi-hand tracking and additional functions through added gestures.

Keywords — Pose, 6-DoF, RGB, YCrCb, AR, VR, Interaction Devices, Human Computer Interaction, Computer Vision, Occlusion, Convolutional Neural Networks

I INTRODUCTION

With the rise of Virtual and Augmented Reality (VR, AR), cheap and reliable Human Computer Interaction (HCI) solutions become sought after to make experiences as immersive, or seamless as possible. Earlier research into gesture control, and more current research into pose estimation, have been explored as a means to provide such an HCI system. However, few such systems have seen success; those that are feature-rich sacrifice affordability, such as implementations from Ultraleap and Oculus, whose hardware comes at a high cost, whereas accessible options are limited in their uses.

This rise in popularity of VR and AR largely stems from it's increasing adoption outside of homes, as well as it's falling price. Workplaces are considering it as an alternative to physical meetings, making them more personable than voice chats and video conferences. Other important use-cases for VR and AR may be found in sectors ranging from service, to military and medical - although research and deployment is still in early stages, results are promising for VR and AR in terms of it's effectiveness in training new recruits/employees. Although this project does not directly propose a solution built for VR or AR applications, the method and techniques are easily transferable.

Traditionally, physical hardware is required to interact with computers. For a standard desktop computer or television display, such methods of interaction may seem natural, whether it be using a remote controller, a hand-held gamepad, or a mouse and keyboard. With the widespread adoption of digital interfaces in public spaces, such as interactive travel guides and shopping mall directories, as well as already present public-use computers, the constant sharing of devices poses a health and hygiene risk; the shared point of contact becomes a concentration point of bacteria, posing a risk to the health of users. Other than increasing immersiveness, hygiene needs to be considered in the pursuit of more suitable interaction methods for computers.

Hand-gesture based HCI is an immature field, and sees application in real-world scenarios very rarely. Such a problem can be partially solved by a convolutional neural network (CNN), but the problem lies in using gestures as an effective way to interact with a device. Such a problem can be simple if the application is also simple, but in the scope of usage for a computer, this quickly becomes complex, comfort and extent of control becoming factors to be considered. As an extension to VR and AR, immersiveness is another factor to be considered; the solution must account for the usage of both hands, which presents the problem of self-occlusion, where one hand is obstructed from view due to the other hand. Current advanced HCI systems, using gesture/pose estimation are poor at coping with such scenarios.

The usage of a single RGB camera in this project increases it's accessibility; most laptops have built-in RGB webcams, and most popular VR/AR headsets contain an array of cameras and sensors. However, the simple 3-channel output of an RGB camera has proven to be difficult to work around, with much of prior research opting to use more complex sensors and configurations.

In this paper, the different approaches to camera-based HCI with the hand are considered, as well as the techniques used in order to make such systems possible. Many Computer Vision techniques are applied, and a CNN is used in gesture recognition. Applications of the system are considered, and an HCI system is implemented, which uses the hand as a complete alternative to the mouse. This system is then evaluated across several criteria, such as intuitiveness, comfort, performance metrics, and accuracy. Much focus is directed towards making an efficient system, capable of running on lower-powered computer systems. This system has been made to support glove-usage, and glove-less usage to an extent. Finally, the paper explores the limitations of the

current implementation, and suggests further improvements.

A Background

A CNN has many uses, ranging from image recognition, to classification and labelling, and can be applied in many different ways. Gesture recognition can be considered as a classification problem, hence this project aims to train a CNN to perform the task. The field of CNNs is thoroughly explored, hence a state-of-the-art model, such as ResNet (He et al. 2016), is selected. However, the performance of even a state-of-the-art CNN is typically not ideal for real-time usage, being limited by the time taken to perform a classification, hence this project explores ways in which the process of gesture recognition can be made more efficient, rather than improvements to the CNN model. Due to the lack of readily available datasets as well as the uniqueness and variability of hand gestures, a dataset of hand images will need to be compiled manually.

Using hand gestures as a means of interaction provides a fair amount of control, as many hand gestures can be formed. The removal of a physical, hand-held controller may improve the immersiveness and convenience of the experience. This is a result of the reduced setup time, the decrease in physical clutter in a workspace, as well as the freedom of being able to use a system without regards to battery levels or misplacing a controller. Without a physical controller, users may also feel like they are interacting with virtual items more directly, adding to the immersive experience.

Outside of VR and AR, however, this system still gives a user more freedom. Users may be away from their desks and still interact with their computers, given that their hands are within the frame captured by a sensor. This can be useful in use-cases ranging from giving presentations on a large stage, to simply changing the channel on a television. Similarly, by removing the point of physical contact, we improve hygiene for public computer systems; by removing a physical controller, there is less likelihood of cross-contamination.

However, the difficulty lies in building a system which is both efficient, cheap and intuitive. Currently, solutions suffer from many drawbacks, such as being too economically expensive for the general public, being too unreliable for meaningful usage, being too computationally expensive to be usable in realtime, or being too inconvenient to setup for the average consumer. In this project, the issue of expense is accounted for, through the usage of a single RGB camera, which is available in most modern laptops and smartphones. Hence, a large focus of this study is to find a good balance between computational expense and performance.

B Objectives

The research question proposed for this project is *Can a gesture-based Human-Computer Interaction system be used as an alternative to traditional, accessible controller interfaces?*. To answer this question, objectives were set. These objectives are split into minimum, intermediate and advanced categories.

The minimum objective of the project is to develop a system capable of replacing a traditional mouse. This includes development of some method to extract a hand from the frame of an image, generating a set of hand-gesture images to be used as the dataset for the CNN, implementing and training such a CNN, and developing a program that converts the gestures obtained from the CNN into suitable mouse actions. These objectives assume the use of a coloured glove. This objective is considered complete if a system with left-click, right-clicks and scrolling functionality, as well

as cursor movement, is produced.

The intermediate objectives focus on improving the solution's performance. The performance will be measured in frames per second (FPS). Improvements include better usage of available hardware, comparing and discerning effective Computer Vision techniques in order to extract the hand and testing different video stream configurations. The accuracy of the system will also be measured. This will be done by looking at the confusion matrix of each gesture-to-mouse function, and calculating the F1-score.

The advanced objectives focus on expanding the functionality of the system, and removing the need for wearables. This includes the addition of more gestures and corresponding functions, as well as adding support for multiple-hand tracking. In order to remove the reliance on a coloured glove, optimizations to the hand-extraction technique will need to be made to support skin-colour detection.

II RELATED WORK

A Applications

In order to gauge the value of a gesture-/pose-based HCI, possible applications must be explored and put into context. Bhuiyan & Picking (2009) explores a variety of situations and scenarios in which gesture-based applications can be used, such as in entertainment, controlling home appliances, and elderly or disable care, mentioning that most applications are done as to replace a traditional mouse and keyboard. This paper builds upon that concept by attempting to create a truly universal replacement for the mouse and keyboard, using gesture controls.

However, with the advent of VR and AR, the range of applications is further expanded - with immersion in mind, solutions to interact with a virtual environment with 6 or more Degrees of Freedom (DoF) using a user's hands are being researched. Currently, most VR headsets require a physical controller. The Valve Index controllers, being the most state-of-the-art amongst commercially available controllers, provide a true 6-DoF experience in conjunction with a VR headset, but fails to model the hand fully, only providing individual finger extension/retraction. (Han et al. 2019) has developed a pose-estimation solution for the Oculus Quest, capable of recreating a user's hands in VR, but has problems common to the field, and requires specific hardware to use. However, this implementation allows for the full 27-DoF for each hand, as the entire hand is modelled virtually in real-time.

Given an improved version of Facebook's solution, many different sectors can be provided with more realistic training methods, through simulations in VR. These sectors range from the industrial sector, as in (Mujber et al. 2004), to the medical sector, as in (Alaraj et al. 2011). It's usefulness is more obvious in the medical sector, where surgeons can practice training operations within VR. Considering the type of precision often required in surgeries and operations, a reliable and accurate hand-pose-estimation HCI would be very beneficial for training a surgeon's dexterity.

B Hand Detection

A key area of this project is being able to detect the hand within the frame of a camera/sensor. In this section, multiple computer vision (CV) techniques, as well as a deep learning method, are explored.

B.1 Colour Masking and Skin Colour

Colour Masking is a very basic, yet effective CV technique for segmenting an image, and is especially effective in situations where the object being segmented is of a unique colour. One can easily make their hands a unique colour by wearing a glove, as in (Wang & Popovi'c 2009); the glove will trivially segment the hand from the forearm, colour-wise, given that the glove's colour contrasts the user's skin tone. However, backgrounds must be considered, and it cannot be guaranteed that a user owns a pair of gloves whose colour is unique both to their background and skin tone.

Skin colour detection is also explored. (Vezhnevets et al. 2004) surveys the performance of different colour spaces for skin-colour detection, and finds that colour-space selection relies on the dataset being used, as well as the overlap of skin and non-skin colours within the colour space. A key consideration is how skin colour changes with the level of lighting, hence a colour space with a luminance component is presumably desirable; by ignoring luminance in a colour mask, lighting is essentially ignored. However, this does not take into account the light's colour temperature. For example, sunlight may cast a blue hue over the image, and household light bulbs may cast a yellow hue over the image. The warmer colour temperatures may cause issues where the background is predominantly white. Similarly to a gloved approach, there is no guarantee that the background colour(s) is uniquely different to the user's skin colour.

Ultimately, colour masking is not enough alone to segment an image for hand detection, regardless of a gloved or glove-less approach.

B.2 Background Subtraction

Background subtraction is a method of learning a background, and then creating a mask segmenting foreign objects in the image. (Ogihara et al. 2006) makes use of this to extract the hand from the image. One disadvantage to this approach, however, is that it is unable distinguish between the hand and forearm. This becomes significant in situations where only the hand gesture is desired. Since this study aims to look at hand gestures specifically, this must be accounted for. In conjunction with a colour mask and a glove, this issue can be subverted. However, this will not work with skin colour detection, assuming a user's skin colour is uniform across their hand and forearm. Furthermore, any other sections of exposed skin would be detected in the mask, increasing the difficulty of narrowing the colour mask down to just the hand. This issue is somewhat alleviated in the case a user is learnt to be part of the background, however users are likely to move throughout the duration of using the system.

B.3 Optical Flow

Optical flow creates a similar mask to that of background subtraction. However, details, such as direction of movement and magnitude of speed, are identified. Using the speed of motion, the hand can be segmented from the image, under the assumption that the hand is the fastest moving part within the frame at any given time. (Cutler & Turk 1998) explores the use of real-time optical flow for gesture recognition. A disadvantage to this approach is the reduced detail in the shapes within the mask, which is essential for this project. (Cutler & Turk 1998) avoids this problem entirely, as it looks at gestures performed with the entire body. In this study, in conjunction with a colour mask, this issue can be alleviated, at the expense of extra computation.

B.4 Mask R-CNN

Mask R-CNN is a deep-learning approach to image segmentation, and is typically used for object detection and segmentation for larger unique objects. (Nguyen et al. 2018) trains a mask R-CNN model to do hand segmentation under different viewpoints. Using this method avoids any need for additional CV techniques, as a hand mask is generated in the output, but the method is severely limited by its processing speed - it is capable of processing at 7-10 frames per second (FPS), making it not ideal for real-time usage, where the target FPS should be 25-30FPS at minimum. It should be noted that taking such an approach will require hand-specific datasets, which may not be readily available.

C Camera Placement

For this study, camera placement should be considered. These can be separated into first-person and third-person views.

In the case of choosing a first-person viewpoint, such as those in (Rogez et al. 2014) and (Niehorster et al. 2017), skin detection becomes a very viable method, so long as the backgrounds do not match in colour. Due to the sensor being faced away from the user, the camera is unlikely to be able to capture any parts of the body other than the hands. Amongst first-person camera placements, the camera may be placed on the head, as in (Niehorster et al. 2017). The limitation of this placement is that the user must have their hands in their field of view, in order for them to be detected. This can be slightly alleviated using a camera with a wider field of view.

Another camera position would be on the chest, as in (Rogez et al. 2014). In this case, the camera would be faced in a constant direction, perpendicular to the chest, allowing the user to look away from their hands, but still have the hand be detected. However, due to the range of motion in the arms, the hands may often be in the blind spot of the camera, and a user will be limited to using their hands only in front of them. Similarly, a wider field of view will help to an extent.

Third-person viewpoints are somewhat more limited, as the camera is in a fixed spacial position, hence the space in which a user's hands may be detected is fixed as well. The most common placement for a third-person viewpoint is in front of the user, similar to a webcam on a laptop. This is used in (Gupta & Hebbalaguppe 2018), and has the advantage of being convenient, taking advantage of a user's/manufacturer's tendency to place a webcam in front of the user. It's also accessible in the way that anyone who has a laptop is likely to have a built-in webcam. This has the disadvantage of capturing more than just the user's hands. Another option for third-person camera placement would be in a top-down configuration, where the camera is above the hand, faced downwards, as in (Wang & Popovi'c 2009). Though less convenient, much less of the user's features is captured in this configuration.

Ultimately, all camera placements are limited by the field of view of the camera. However, a first-person viewpoint would require that a user has a separate, external webcam to use.

D Gesture- and Pose-Estimation

Gesture-estimation is the process of associating an image to a gesture, hence it is a simple classification problem that may be solved with a CNN. This technology is already available, evidenced in (Montanaro et al. 2016), (Cutler & Turk 1998) and (Bhuiyan & Picking 2009), but only within their certain specific products/services. Pose-estimation is more complex, in that it

fits a model of the hand to an image, and often requires some depth information. Hence, the corresponding datasets required for training a pose-estimation neural network requires much more details, outside of simple labels for gesture names. This is a very relevant problem in this field. (Gupta & Hebbalaguppe 2018) attempts to make an affordable pose-estimation solution, but is limited to only detecting the fingertips and having 6-DoF. (Oberweger et al. 2015), (Oikonomidis et al. 2010), (Rogez et al. 2014), (Segen & Kumar 1999) and (Simon et al. 2017) produce more complete solutions with greater DoF, but require complex and relatively expensive hardware to use. (Zimmermann & Brox 2017) succeeds in creating a 3D hand-pose-estimation solution with a single RGB camera, but uses multiple neural networks in conjunction in order to achieve these results, impacting performance.

A complete solution to this problem would be to use pose-estimation to produce a 3D model of the hand, but the solution must also be accessible, meaning the majority of consumer systems can make use of this solution.

III SOLUTION

A Overview

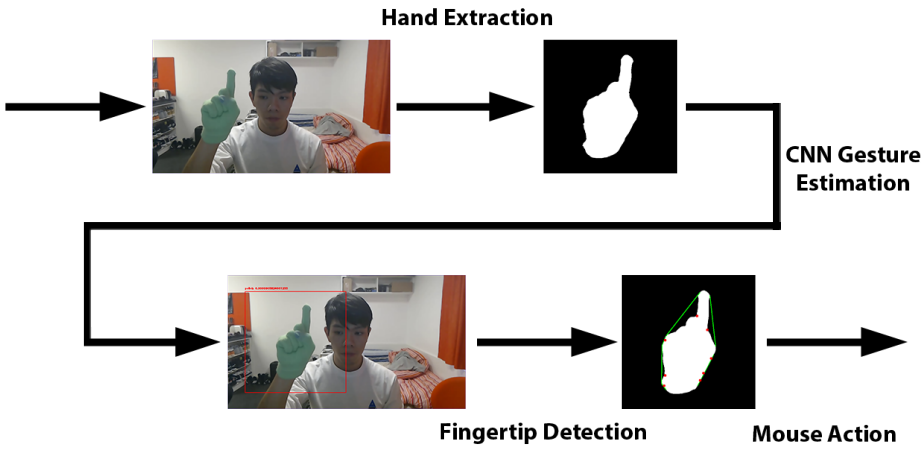

Figure 1: Overview of processing pipeline for the gesture-based HCI system

The main system is implemented in Python with OpenCV, and is capable of processing both real-time footage from a camera, or pre-recorded footage. The system is initially calibrated for background removal and colour masks.

The first part of the system is the hand extraction process, which applies the mask produced by background subtraction to the input image, and uses colour masking within the range found in calibration to extract either the glove or skin colour. Contours are then searched for in the colour mask, and the contour with the greatest area is assumed to be for the hand. A bounding square is then made, encapsulating the hand.

This extracted mask of the hand is passed into a CNN, which receives an image, and outputs a list of confidence values corresponding to a list of learnt gestures.

This gesture is processed on another thread, for conversion into an action corresponding to a mouse function. Currently, gestures include:

- Pointing - moves the cursor on the x and y axis, with it's position corresponding to the user's hand position within the image frame.
- Two fingers pointed up/down (Peace-up/-down) - by changing between these two gestures, the scroll function will be activated.
- Pinching - upon releasing the pinching gesture, a left-click event will be initiated.
- OK-sign - similarly to pinching, releasing this gesture performs a right-click event.

For both left- and right-click gesture controls, if the user goes from the pointing gesture to the pinching / "OK" gesture, they will be able to make fine adjustments to the mouse.

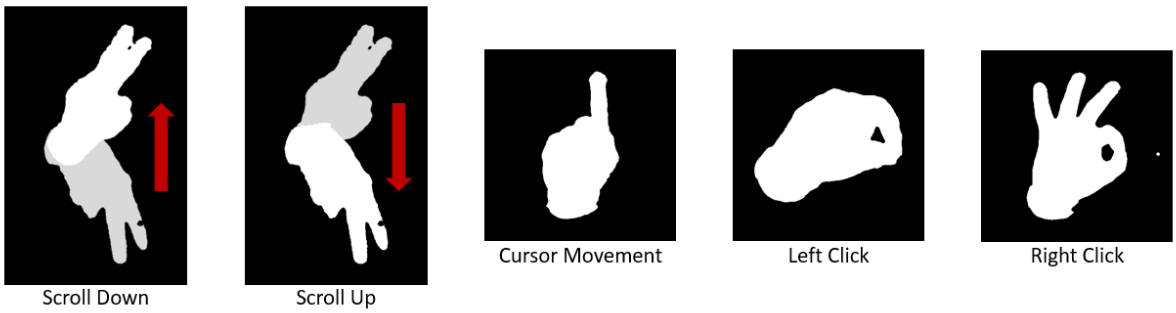

Figure 2: Gestures and their associated actions

B Calibration

B.1 Colour-space Choice

The main candidates for colour-space were LAB, HSV and YCrCb, which all have a luminance-type channel that may be used to account for lighting conditions. As the performance of the 3 colour-spaces were very similar, any one could be used. For this paper, YCrCb was used.

B.2 Real-time

For real-time calibration, the process requires the user stay still in front of the webcam for approximately 5 seconds, to calibrate and train the background subtractor. The background subtraction algorithm used is the Mixture of Gaussian (MoG) 2 method, which is available in OpenCV.

Following the background calibration, the user is prompted to place their hand over a box drawn on the frame. The user is to do this 3 times, for 3 different gestures to ensure full coverage of the hand. For each gesture, the image is cropped to just the box overlapping the hand, and the image is converted into the YCrCb colour-space. The minimum and maximum channel values for the box is updated, discluding the Y luminance channel. The minimum and maximum values for the Y channel is fixed to 15 and 240 respectively, accounting for most levels of lighting, and ignoring blacks/whites.

Finally, a mask is generated using the background subtractor and applied to the original frame, and the Cr and Cb minimums and maximums are readjusted such that the white pixels

of the mask amount to 10% of the frame. To generate the colour mask, the OpenCV function *cv2.inRange()* is used.

B.3 Calibrating for Tests

For test calibration, only the process for finding the appropriate colour mask range is changed - the minimum and maximum values are entered manually. This change was made so as to simplify the process of recording a test video without the use of Python; instead, test subjects can use in-built applications in their operating systems to record such videos.

B.4 HAAR Cascade

OpenCV has a module for facial detection using HAAR cascades. In order to avoid detecting the face during the process of skin detection, facial detection was used for locating any faces within a frame, and the bounding boxes for each face is filled with black. The detection rate was highly inconsistent, and presumably depends on the video resolution.

B.5 Object Tracking

OpenCV also has a module for object tracking, and this was used following initial facial detection. This was done with the intention of reducing the frame processing time, after an initial face detection. However, the object tracker itself was not efficient enough, and also is incapable of dealing with scenarios where the tracked object leaves and returns to the frame. It is also poor at tracking objects moving at higher speeds, hence isn't suitable for hand tracking. The performance was also comparable to using a HAAR cascade.

B.6 Optical Flow Masking

As an alternative to background masking, an optical flow mask was considered. This would provide the advantage of being able to distinguish between objects by the speed at which they move. Specifically, a fast object such as the hand could be distinguished from a slow object such as the user's head and body. However, similarly to object tracking, it was an inefficient method that resulted in noticeably longer processing times per frame, hence it was deemed impractical for real-time usage. Moreover, it requires usage in conjunction with another type of mask, as this technique is only effective on the edges of objects.

B.7 Expanding on Background Subtraction

The background subtractor in OpenCV can be dynamically trained. An attempt to make use of this was made, by setting a binary threshold on the mask, such that if the white-to-black ratio of the mask exceeded a certain amount, the background subtractor should begin training again. This is effective in dealing with slight movements of the head, as well as possible changes in lighting. However, if the hand is in frame during training, there is a possibility that the hand is recognised as the background if it does not move enough. This happened frequently in testing, and as a result this idea isn't used in the final product.

C Hand Extraction from Video Stream

A bilateral filter is applied to the input frame in order to reduce the noise present in the frame. Using the background subtractor (if no glove is used) and minimum/maximum YCrCb values from calibration, a mask is produced for the frame, by applying the background mask to the original frame, and then doing a colour mask.

The contours of the colour mask are found with the OpenCV function *cv2.findContours()*. The contour with the largest area is selected, and then checked to be greater than a certain threshold. This threshold is determined as a percentage of the total area of the frame, hence if the largest contour is below this threshold, it is assumed to be too small to be a hand.

Using the *cv2.boundingRect()* function in OpenCV, which returns the root coordinates and width/height of a bounding box, a square is drawn around the largest contour, using the maximum of the width and height as the square size, and using the root coordinates to find the center of the contour. Padding is added to the box, as a percentage of the width. The mask is then cropped to the generated square, hence leaving a square mask of the hand.

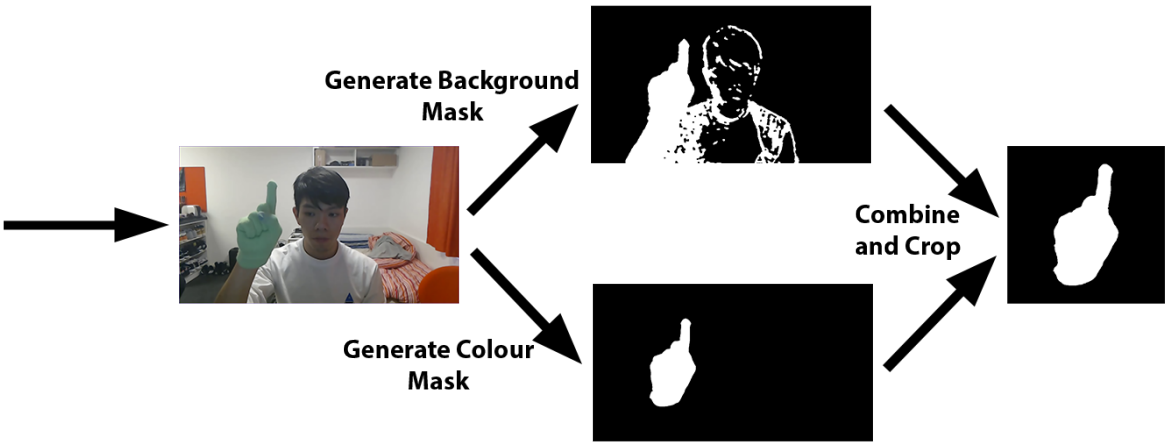

Figure 3: Flowchart showing hand-extraction process

C.1 Fiducial Markers

Initially, fiducial markers, in the form of small spheres, were intended as a means of distinguishing the fingers, as well as a reference for obtaining depth information using the focal length of the camera. However, it was concluded that the markers were too bulky to be used comfortably.

Making them smaller would cause issues, depending on how they are attached to the hand. The most convenient way was to cut a fingertip-sized hole in the marker to slot the finger into. This would require that the sphere is sufficiently large enough such that the difference in width from the insert-hole to the opposite end of the sphere and actual diameter of the sphere is negligible.

Another option was to attach spheres at the tip of the fingertip. However, this would not accurately represent the locations of the fingertips, and may also become a nuisance when a gesture requires that two fingertips touch.

D Gesture-Recognition Network

D.1 Configuration

For gesture-recognition, a CNN is implemented using Pytorch. Pytorch has many pre-configured network models, hence it was a matter of selecting the most suitable model for this use-case. Ultimately, a wide ResNet model was selected, with 50 layers. Key considerations when selecting a model included the accuracy of predictions, as well as the performance and efficiency of the network.

This model takes in images with minimum dimension sizes of 224 and 3 channels. In this study, images are resized to 256x256 pixels. Since the generated masks are greyscale, the masks are converted into 3-channel images using OpenCV's *cv2.cvtColor()* function.

The output of this network consists of a list of confidence values associated with every possible classification as learned by the network. In this case, there are 5 gestures learnt by the network, hence the output is a list of 5 confidence values. The class with the greatest confidence value is the most likely gesture made in the input.

D.2 Training

The training process is as follows:

1. For every class/gesture, make the gesture and:
    - Move the hand throughout the frame, to capture as many angles of the gesture as possible
    - Collect hand masks using the specified hand extraction process
    - Resize the masks to 256x256 images
    - Save the hand masks, using *cv2.imwrite()*, to a folder containing masks of the same gesture
    - Continue until 500 masks are collected
2. Ensure each folder of gestures is contained in the same folder, known as the training folder
3. Using Pytorch's *datasets.ImageFolder()* function, load the training folder into Python
4. Allow the training loop to run

For training, a batch size of 4 was used. To calculate the loss, cross-entropy loss was chosen, and stochastic gradient descent was used as the optimization algorithm. The network is trained over 25 epochs, and the dataset is shuffled with each epoch.

D.3 Grayscale vs. Binary Mask

Initially, grayscale masks were used, such that the binary mask is overlayed on the original frame and converted to grayscale. This allowed for retention of more detail in the hand, but resulted in poor prediction accuracy and high mean-squared error loss when training the model. Hence, a binary mask was used instead. This issue could be circumvented if a larger dataset containing more varieties of hands under different lighting conditions were accessible.

E Conversion from Gesture to Controller Input

E.1 Fingertip Detection

For certain gestures, the fingertip is used as a reference for mouse positioning. This includes the pointing, ”OK” and pinching gestures.

For the pointing gesture, the index fingertip is detected using convexity defects, and done through OpenCV's *cv2.convexHull()* and *cv2.convexityDefects()* functions with the hand contour. The convex hull function finds the smallest set of points that encloses the entire set of points making up the hand contour, and the convexity defects function then finds the spaces within the hull, that do not belong to the object, and returns a set of points. In this case, such points would typically be found between the fingers, or near the wrist. A point on the hull, in between two convexity defects, is likely to be a finger. In this specific gesture, the point with the smallest y-coordinate, hence the point closest to the top of the frame, is most likely to be the tip of the index finger, hence this point is taken.

For the ”OK” and pinching gestures, instead of the point closest to the top of the frame, the point furthest to the right of the frame is selected instead, assuming the user is right handed. From certain angles, defects may not be found, hence the software swaps to a more naive approach, simply taking the point on the contour that is furthest right.

E.2 Mouse Actions through Python

All mouse actions are done through the **pynput** library.

Cursor movement is done through fingertip tracking, hence the position of the cursor is relative to the position of the user's fingertip within the frame. The original frame is scaled such that it is greater-than or equal to the display resolution, and overlapped such that they are aligned centrally. The mouse cursor is set to the position overlapping the fingertip. The sensitivity of the mouse can be adjusted by changing the scale factor between the frame size and display resolution. The cursor is moved with the pointing gesture, as well as the ”OK” and pinching gestures, given that the previous gesture was the pointing gesture.

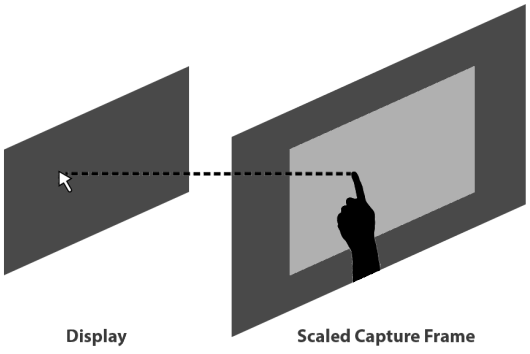

Figure 4: Graphical representation of conversion from fingertip position to cursor position

The left- and right-click operations are allocated to the pinch and ”OK” gestures respectively. Upon making the gesture, if the previous gesture is the pointing gesture, the user may hold the current gesture and make adjustments to the cursor position. Upon releasing the gesture, a left-/right-click is triggered.

Scrolling is triggered when the user changes from the two-fingers-up gesture to the two-fingers-down gesture, or vice versa. The former triggers an upwards scroll, and the latter a downwards scroll.

F Optimizations

F.1 GPU Acceleration

Pytorch has native support for CUDA, hence this was used as a means of optimizing the program's performance where possible. This optimization is only available to systems with a CUDA-enabled NVIDIA GPU.

On the other hand, OpenCV for Python does not natively support CUDA, if installed through package managers such as Conda or PIP. In order to enable CUDA support, OpenCV must be built from source. This process was made trivial, as (Bowley 2019 (accessed April 15, 2020)) provides pre-built python libraries in specific configurations. In this project, OpenCV 4.1.0 and CUDA 10.1 are used.

OpenCV for Python supports GPU acceleration through OpenCL. This could be useful in systems where CUDA is unavailable, including older systems, systems with Intel Integrated Graphics, or systems with AMD GPUs. However, in early testing, the results were found to be inconsistent and often slower than CPU processing.

F.2 Multiprocessing on the CPU

Python processes typically execute on a single thread. Using the multiprocessing Python library, the load can be separated across multiple processes, hence running on multiple threads. However, this will not help in accelerating most functions, namely the OpenCV image processing functions, as these are written to take advantage of only a single thread. Furthermore, much of the processing pipeline is linear, hence it would be redundant to allocate jobs to multiple threads. On the other hand, this allows for the separation of the gesture-to-mouse program and the real-time gesture-prediction program.

IV RESULTS

This section explains the experimental process, and presents the results gathered. These results will portray the performance and accuracy of the system in different configurations, as well as the accuracy of the CNN. The performance is measured in FPS, and accuracy is measured using the precision value, recall value and F1-scores.

Due to the Covid-19 outbreak, interactive experiments were not available to test subjects. Hence, a post-experiment survey could not be performed, and certain qualitative criteria could not be evaluated, such as intuitiveness and comfort. However, brief observations by the experimenter will be given, in regards to these criteria. These observations will be biased, being the thoughts of one person only.

A Experimental Setup

Participants are asked to record themselves performing a fixed sequence of gestures without a glove. At the availability of the participant, they may also be requested to submit a second

recording where the participant is wearing a glove.

Each video is fed into the Python program, and the experimenter actively records the number of incorrectly recognised gestures. These results are separated into the unique gestures, as well as whether or not a glove is used. The Python program automatically records the FPS metric.

B Performance

Using the quantitative measure of frames per second (FPS), the performance of the system was measured across 4 different system configurations, on a desktop containing an AMD Ryzen 7 3700X and an NVIDIA GTX 1080 desktop GPU, and a laptop containing an Intel i7 7700HQ and an NVIDIA GTX 1050ti mobile GPU.

System Configuration	**Frames per Second**		
	1920x1080	1280x720	640x480
R7 3700X + GTX1080	10.244	19.389	26.757
R7 3700X	0.685	0.827	0.674
i7 7700HQ + GTX1050ti	5.259	10.129	14.011
i7 7700HQ	1.013	1.260	1.237

Table 1: FPS of system for different configurations and input resolutions

The results in 1 show that the program struggles significantly in a CPU-only configuration, with unsatisfactory results for real-time usage. However, with the inclusion of a GPU, the system maintains good usability at resolutions of 1280x720, with even more improvements in FPS at lower resolutions. Given most included laptop webcams are capable of 30FPS capture at most, the results at 640x480 are ideal.

C Accuracy

To test the accuracy of the gesture controls, the precision (P) (Eqn. 1), recall (R) (Eqn. 2) and F1-score (F1) (Eqn. 3) of all gestures will be observed. Specifically for cursor movement, a comparison between different source frame-rates will be done. The accuracy of the hand-masks are observed qualitatively by comparing ideal masks, generated with a glove, to masks generated without a glove.

$$P = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{1}$$

$$R = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2}$$

$$F1 = 2 * \frac{P * R}{P + R} \tag{3}$$

As in 2, a glove-based approach shows significant improvements in accuracy as opposed to a glove-less approach. Another observation is that using a camera with a higher recording FPS does not necessarily improve the accuracy of the system, implying that it may not improve the gesture estimation accuracy. However, this requires more study, as not all participants had a camera supporting 60FPS capture.

Gesture Control		No Glove			Glove		
		P	**R**	**F1**	**P**	**R**	**F1**
Cursor	**30FPS**	0.818	0.825	0.821	0.929	0.972	0.950
	60FPS	0.933	0.702	0.801	0.942	0.744	0.831
Left-click		0.332	0.174	0.228	0.291	0.769	0.422
Right-click		0.120	0.908	0.212	0.429	0.891	0.579
Scroll-up		0.482	0.087	0.147	0.855	0.729	0.787
Scroll-down		0.736	0.261	0.385	0.910	0.866	0.887

Table 2: Accuracy results for each gesture control in different configurations

(a) Glove mask in YCrCb colour space (b) Skin-colour mask in YCrCb colour space (c) Skin-colour mask in LAB colour space (d) Skin-colour mask in HSV colour space

Figure 5: Comparison of masks from a gloved approach (a) and skin-colour approach (b,c,d)

Amongst all gestures controls, the cursor seems to be most accurate, with the highest F1-score. In a no-glove configuration, all four other gesture controls are considerably inaccurate, whereas in a gloved configuration, there is significant improvement in the scrolling controls. However, the left- and right-click controls have much room for improvement.

Note the discrepancy between the precision and recall values for the right-click gesture in a glove-less configuration. The 0.120 precision implies a large number of false positives, relative to the number of true positives. Furthermore, the 0.908 recall implies a large number of true positives, relative to the number of false negatives. This shows the number of false positives is very high. Also, the right-click gesture is the only gesture for which the recall is greater than the precision when using a glove-less approach.

Figure 5a shows a clean, ideal mask as obtained with a glove, classified as a pointing gesture by the CNN, whereas 5b shows a noisy mask where parts of the hand are masked off, and parts of the face can be seen in the mask, miss-classified as an ”OK” gesture. With the ”OK” gesture being used to trigger a right-click, this shows that the ResNet CNN is improperly classifying frames similar to 5b very often. As a result of the CNN being trained on ideal masks generated with a glove, it is not trained to cope with situations such as in 5b.

Furthermore, the skin-colour detection’s performance isn’t perfect, as parts of the hand are not detected as part of the colour mask. It should also be mentioned that the participant in 5b has contrasting hair and skin colours. However, 5c and 5d show similar results in different colour spaces, implying that colour space is not the issue.

C.1 CNN Accuracy

To measure the accuracy of the CNN, a test dataset, generated in the same manner as the training dataset, is sent through the CNN, and the recall accuracy is calculated. Figure 3 shows high

	Recall (%)
Point	91
Pinch	99
OK	82
Peace Up	88
Peace Down	99
Overall	92

Table 3: Prediction accuracy of gesture CNN over a test dataset of 1000 images

capability in predicted all gestures, with the ”OK” gesture having the lowest recall percentage at 82%.

At the end of training, the mean-squared error loss was 0.029.

C.2 Comfort, Intuitiveness, Usability

In terms of comfort, there is some concern in regards to arm soreness. Without a surface to rest the arm on periodically, the arm will begin to feel sore within a short period of time. Also, users may feel discomfort/strain when trying to interact with items located far on their non-dominant side, requiring that they reach their dominant arm across their body.

Intuitiveness as a whole is generally good. Scrolling up and down, and moving the cursor feels very natural. Pinching to left-click and using the ”OK” gesture to right-click is also easy to remember. However, having to release the gesture to initiate a click is not ideal.

Usability is not perfect. Scrolling and moving the cursor works very well, but left-clicks and right-clicks are difficult; when changing gestures, the cursor moves off the intended target. The ability to readjust the cursor whilst in the pinching and ”OK” gesture helps slightly, but this function does not work consistently for the right-click function, and when releasing the gesture, the cursor also moves, resulting in the target not being clicked sometimes. This could be a bigger issue if the target is very small.

V EVALUATION

In this section, the strengths and limitations of the proposed solution are looked at, with regards to the original research question *Can a gesture-based Human-Computer Interaction system be used as an alternative to traditional, accessible controller interfaces?*.

A Solution Strengths

The solution demonstrates reasonable success in it’s ability to segment a hand from an image, predict a gesture, and convert gestures into some mouse input, given the usage of a glove. The high level of hand segmentation is demonstrated by 5a, showing a noise-less mask with smooth edges. The ability to predict a gesture, given masks of a high quality, is referenced in C.1, with

a very low mean-squared error loss at the end of training, as well as high prediction accuracy on the test data. The conversion to mouse input is accurate to a usable extent, with the cursor and scrolling inputs being recognised consistently, as implied by their F1-scores, all of which are greater than 0.75.

As far as real-time usage goes, the solution is usable, given certain computer and video-source configurations. The most common source resolution in this study is 1280x720, for which the system runs at a minimum of 10FPS (1) on a laptop with a mid-tier GPU. At this frame rate, the performance will feel choppy, but still usable. More performance could be gained by reducing the source resolution, as shown in 1.

B Solution Limitations

Although usable with a glove, the range of recognisable gestures is limited due to the nature of the training data. Using binary masks is very effective on a dataset with few, unique gestures. However, as more gestures are introduced, many overlaps occur, and the CNN begins to struggle differentiating between certain gestures. As a result, this solution was limited to only 5 unique gestures.

Performance wise, the use of a CNN has resulted in a heavy dependency on parallel computing on a GPU, using NVIDIA's CUDA specifically. As a result, this reduces it's accessibility, especially to laptop users, where a large majority of laptops do not have a dedicated NVIDIA GPU. As shown in 1, CPU-only performance is at most 1.26FPS, which is unusable for a real-time scenario. Moreover, it's lackluster performance at higher source resolutions limits the distance at which the user can be from the camera sensor.

With hand segmentation, the method used with a glove does not transfer well for a gloveless configuration. 5b, 5c and 5d are clear examples of the reduced quality in mask generation, as well as the limitations of colour thresholding with regards to skin colour, with many false-positive detections in the background. Background subtraction does help with the reduction of false positives, but is highly susceptible to very slight lighting changes, reducing the consistency of the solution's performance.

As a result of the poor hand segmentation, the accuracy of the gesture controls become significantly poorer. It is clear to see in 2, that the right-click input is being triggered far too often, hence the CNN must be incorrectly predicting the corresponding "OK" gesture too often. This decrease in accuracy is enough to render the system unusable without a glove.

C Approach

The approach taken for this project was to initially build a system assuming the use of a glove, beginning with a hand-segmentation technique based around colour segmentation, and using said technique to gather the training data for the CNN. After training the CNN, this should be integrated with the hand-segmentation technique. The gestures would then be programmed to trigger certain mouse inputs. Throughout this process, programming was done with CUDA in mind, which slightly increased the initial learning/preparation time, but accelerated the process of mid-development testing of the system. With Pytorch, CUDA integration is trivial, requiring very little prior knowledge. However, CUDA integration for OpenCV is significantly was less straightforward, and much time was required in either building OpenCV from source, or finding a pre-built library with CUDA support.

Due to the clear separation in the processes involved in the solution, adapting and experimenting with the hand-segmentation technique was simple. However, another key consideration throughout the process was it's real-world usability, hence when transitioning to development on a skin-colour-based approach, the accuracy of the system was sacrificed.

Initially, experimentation was scheduled for a much earlier date than what ultimately happened, and all experiments would occur with a glove. However, due to the Covid-19 outbreak, health became a large concern, and the entire experimentation process had to be redesigned. As a result, the experiments occur remotely, and results in regards to the user experience and real-world usability could not be gathered. Although real-world usability could be inferred from the accuracy of the system and FPS metrics, the user experience, which takes into consideration comfort, fluidity of gestures and mouse inputs, and intuitiveness, could not be observed.

VI CONCLUSIONS

This project has provided a new means of interacting with a computer, albeit with a physical wearable in the form of a glove. Although a glove is not specifically a controller, we were unable to create a usable HCI system with no dependency on a physical object carried by the user.

However, with a glove, we were able to segment the hand from an image, and predict a gesture using a CNN. Furthermore, using these gestures, interaction with a computer was possible, with the hand acting as a replacement for the mouse. Real-time usage is also possible. Hand segmentation was achieved through a combination of background subtraction and colour masking. Training data for the neural network is also simple to obtain, using the same hand-segmentation technique. The pynput library was used to trigger mouse inputs using gestures.

An attempt to remove the glove, using skin-based colour detection, was made. However, success was limited, and a usable system could not be achieved.

A Further Work

Before making improvements further developing the solution, an interactive experiment should be conducted, to gather a wider range of opinions on intuitiveness, usability and usefulness of the system. Future developments for this project will primarily be around the improvement of the glove-less approach to hand segmentation. Another possibility is to do more experimentation with the CNN, in regards to increasing the detail available in the training data. This means replacing the current training dataset, consisting of binary masks, with an alternative. This may lead to an increase in the number of gestures the CNN can accurately discern, and if the backgrounds are kept in the training data with no detriment to the accuracy of the CNN, this may result in less emphasis on quality of the hand segmentation technique. Further additions will include multiple-hand tracking.

References

Alaraj, A., Lemole, M., Finkle, J., Yudkowsky, R., Wallace, A., Luciano, C., Banerjee, P., Rizzi, S. & Charbel, F. (2011), 'Virtual reality training in neurosurgery: Review of current status and future applications', *Surgical neurology international* **2**, 52.

Bhuiyan, M. & Picking, R. (2009), 'Gesture-controlled user interfaces, what have we done and what's next?', *5th Collaborative Research Symposium on Security, E-Learning, Internet and Networking* .

Bowley, J. (2019 (accessed April 15, 2020)), *Parallel Vision*.
URL: *https://jamesbowley.co.uk/*

Cutler, R. & Turk, M. (1998), View-based interpretation of real-time optical flow for gesture recognition, pp. 416 – 421.

Gupta, O. & Hebbalaguppe, R. (2018), Fingertipcubes: an inexpensive d.i.y wearable for 6-dof per fingertip pose estimation using a single rgb camera, *in* 'SA '18'.

Han, S., Liu, B., Yu, T. H., Cabezas, R., Zhang, P., Vajda, P., Isaac, E. & Wang, R. (2019), *Using deep neural networks for accurate hand-tracking on Oculus Quest*.
URL: *https://ai.facebook.com/blog/hand-tracking-deep-neural-networks/*

He, K., Zhang, X., Ren, S. & Sun, J. (2016), Deep residual learning for image recognition, pp. 770–778.

Montanaro, L., Sernani, P., Calvaresi, D. & Dragoni, A. F. (2016), A touchless human-machine interface for the control of an elevator.

Mujber, T., Szecsi, T. & Hashmi, M. (2004), 'Virtual reality applications in manufacturing process simulation', *Journal of Materials Processing Technology* **155**, 1834–1838.

Nguyen, D.-H., Le, T. H., Tran, T.-H., Hai, V., Le, T. & Doan, H.-G. (2018), Hand segmentation under different viewpoints by combination of mask r-cnn with tracking, pp. 14–20.

Niehorster, D. C., Li, L. & Lappe, M. (2017), 'The accuracy and precision of position and orientation tracking in the htc vive virtual reality system for scientific research', *i-Perception* **8**.

Oberweger, M., Wohlhart, P. & Lepetit, V. (2015), 'Hands deep in deep learning for hand pose estimation'.

Ogihara, A., Matsumoto, H. & Shiozaki, A. (2006), 'Hand region extraction by background subtraction with renewable background for hand gesture recognition'.

Oikonomidis, I., Kyriazis, N. & Argyros, A. A. (2010), Markerless and efficient 26-dof hand pose recovery, *in* 'ACCV'.

Rogez, G., Khademi, M., Iii, J., Montiel, J. & Ramanan, D. (2014), 3d hand pose detection in egocentric rgb-d images, Vol. 8925.

Segen, J. & Kumar, S. (1999), Shadow gestures: 3d hand pose estimation using a single camera, Vol. 1, p. 485 Vol. 1.

Simon, T., Joo, H., Matthews, I. & Sheikh, Y. (2017), Hand keypoint detection in single images using multiview bootstrapping, pp. 4645–4653.

Vezhnevets, V., Sazonov, V. & Andreeva, A. (2004), 'A survey on pixel-based skin color detection techniques'.

Wang, R. & Popovi'c, J. (2009), Real-time hand-tracking with a color glove, Vol. 28.

Zimmermann, C. & Brox, T. (2017), Learning to estimate 3d hand pose from single rgb images, *in* 'IEEE International Conference on Computer Vision (ICCV)'. https://arxiv.org/abs/1705.01389.
URL: *https://lmb.informatik.uni-freiburg.de/projects/hand3d/*