香　港　大　學

THE UNIVERSITY OF HONG KONG

Progress Report I

Hand Gesture Recognition System

Author: Dhruv Agrawal

UID: 3035344405

Supervisor: Dr. K.K.Y. Wong

Course: CAES 9542: Technical English for Computer Science

Instructor: Choi Yuan Han Mable

Abstract

Acknowledgements

Table of Content

List of Tables

List of Figures

List of Acronyms

Chapter 1: Introduction

In this section, I present a short overwise of this project stating the required contextual details, scope, significance of the project and outline of this project

1.1 Background: Brief overview of Hand Pose Estimation Systems

Hands are our most important devices when it comes to interacting with the world. Even in a quiet library, two people are able to communicate with little constraint on their communicational facilities due to an extensive use of hand gestures that covers for the lack of verbal conversation. Hand Gestures also see applications in presentations and conversing in presence of a large language carrier. Perhaps the most significant application of effective hand gestures is in Sign Language. With clear distinct set of hand gestures, sign languages allow conversing with people with auditory impairment. Thus, we clearly see that humans use hand gestures to convey a large amount of information. This means that in order to achieve Artificial Intelligence (AI) that is capable of seamlessly interacting with humans, it is vital that we are able to encode our inherent ability to convey information with the use of our hands into the AI. A system allowing for such information inference is called a Hand Gesture Recognition System. It will be discussed in more detail in the following section.

Commented [d1]: Find another word.

In order to achieve a marketable Hand Gesture Recognition System, it is essential to first create a system that allows the computer to detect, and regress the position and orientation of a hand from a camera input. Such a system is called a Hand Pose Estimation System. Such a system would be capable of analyzing a video input and regressing the 3D locations of all the palm and finger joints in the video stream. This system has invited a large amount of interest and a proportionate amount of research has been conducted into developing such a system. More technical details of different present state-of-the-art performing systems will be discussed in Chapter 2. However, presently it suffices to mention the salient features of these systems and discuss the challenges faced while estimating hand poses. Most of present day state-of-the-art models are based on deep learning algorithms that employ 2D or 3D Convolutional Neural Networks (CNN). Majority of these applications use either stereo vision or IR

cameras in order to input some form of depth information into the system. Lastly, another popular approach for hand pose estimation is to use video input and use spatial or temporal pooling to maximize the amount of information available to the system.

Next, we look at some of the challenges faced with Hand Pose Estimation. Firstly, note that the fingers are the most "free" parts of the human body. Due to the three joints, Metacarpophalangeal joint, Proximal Interphalangeal joint and Distal Interphalangeal joint, the fingers are capable of fine movements and have a large volume that each finger can potentially occupy. Since large degree of freedom for the fingers results in a large latent output space for a model, this severely limits the possible accuracy of a hand pose estimators. Next, another challenge due to the anatomy of the fingers arises due to the varying finger and palm sizes. The sizes of fingers vary from human to human. However, besides the sizes of particular fingers, the ratio of the lengths of different fingers also varies amongst individuals. Easiest example of this being that many people have their index and ring fingers of almost same length while others have one clearly larger than the other. This results in a large number of possible human hand models. This is less desirable for programming a generic hand model into a system. The system has to account for different variations of human hand and the different latent spaces that arises from these differences. Lastly, one of the most challenging problems faced during Hand Pose Estimation is due to self-occlusion. Due to the large number of joints in such a small region, it is easy for fingers to block each other from the view of the camera. This can result in the system having to estimate position of a finger that is completely occluded from the view of the camera. In summation, due to the aforementioned challenges, the field of Hand pose estimation remains very popular for research and continues to attract many researchers.

1.2 Background: Brief overview of Hand Gesture Estimation Systems

The current research community often uses Hand pose estimation and hand gesture recognition interchangeably. Hand pose estimation is almost often used to refer to regressing the location of hand joints. However, the term hand gesture is sometimes used as a synonym for hand pose and some other times used to refer to the class or label assigned to the particular posture. In this study, we refer to the latter meaning when we say hand gesture. For example, imagine an image depicting a hand with only the index and middle fingers extended. In this study, locating the hand joints in the image while be referred to as hand pose while labeling the image as "2" (since the image depicts the sign 2) as hand gesture.

With this differentiation, one can realize that hand gesture recognition is indeed a harder problem when compared to hand pose estimation. Hand gesture recognition not only involves understanding the configuration of the hand in the image but also to infer meaning from this configuration. Due to the variability of hands caused by the different sizes and shapes, a single gesture can appear very different. Therefore, a hand gesture recognition system has to be invariant of these differences and infer meaning from a generalized model of a hand. When discussing hand gestures, there are again two types. Firstly, the gesture can be static. As the one is the previous example. Such gestures can be depicted through a single image. The other type of hand gestures are dynamic. This means they involve some form of hand motion in order to convey the meaning. For example clapping or waving. Due to the motion involved, a single image is not longer sufficient and we use videos to capture these gestures. Since dynamic gestures add motional and temporal complexities into the system, it makes classification of such gestures even more challenging. Lastly, due to the change in file format, even storage limits become actual concerns. This leads to finding a balance between the dataset disk size and the quality of the videos when constructing the videos.

In this study, we use the system we build for hand pose estimation in order to aid our hand gesture recognition by providing a normalized version of hand pose to the recognition system. We believe by doing so, we can reduce the variability of hands. This should in turn improve the performance of our classifier since it has to learn lesser amount of information about hand poses.

1.3 Purpose of the project

This project attempts build a Hand Gesture Recognition System with two key features. Frist, the project attempts the minimize the amount of input data required to achieve a functional Hand Gesture Recognition System. Therefore, the project focuses on using a single RBG-D sensor as the input source. The project posits that using a strong Hand Pose Estimation model will allow the complete system to recover from a lack of information available. Secondly, the project aims to find evidence of feasibility of using a hand gesture recognition system in online conditions. This would mean that the total execution time of the algorithm will have to be sufficiently low in order to not introduce a noticeable lag in any hand gesture based application. Since such a system is yet to be practical, this project will attempt to find evidence on feasibility of such a system given the constraints of execution time and executing hardware.

The deliverables of this project will include a Graphic User Interface (GUI) application that allows for user-friendly interaction with the underlying machine learning models. Furthermore, this project will

also deliver a novel machine learning architecture that can perform hand gesture recognition effectively.

1.4 Significance of the project

It is expected that through the development of this project, I will be able to develop to an architecture that can contribute to the global research in the area of Human Computer Interaction. The second objective of this project to find evidence regarding the feasibility of online hand gesture recognition might be able to contribute into deciding the future areas of research under this topic.

Lastly, this project also serves a personal purpose of allowing me to gain insight into the practices and modern techniques influencing postgraduate research.

1.5 Outline of the report:

Chapter 2: Related Works

2.1 Introduction:

In this section we explore the vast amount of research that has already being conducted related to Hand Pose Estimation and Hand Gesture Recognition. We first look at overview of the prevalently applied techniques for the two tasks and then we look at few highly relevant research papers in details that have a large effect on the application of the system in this project.

Khan et. al. [1] provides a thorough literature review of Hand Pose estimation techniques that do not employ Deep Learning Algorithms. [1] divides the process of hand gesture estimation into three key steps: 1) Extraction Method; 2)Feature Extraction; and lastly 3)Classification. Extraction method refers to extracting the hand segment from the image background. This allows the system to isolate the hand and remove the noise introduced from the background textures. Next, feature extraction refers to extracting contextual information such as Centre of Gravity, Hand Contours and Silhouettes, and Fingertips position. Lastly, Classification refers to using the extracted features to make a prediction of the depicted hand gesture in the image. Different classification systems greatly effect the overall accuracy of the complete system. Some of the frequently employed non-deep learning models include Fuzzy C-Means Clustering and Genetic Algorithms [1].

2.2 Hand Pose Estimation

However, with the recent increase in the popularity of the Deep Neural Networks(DNNs), a majority of the Hand Pose Estimators attempt to use CNNs for performing the aforementioned Extraction, Feature Extraction and Classification. This decreases the amount of the data preprocessing carried out by the researchers and also allows for the model to regress its own data processing parameters with low exterior intervention. In such a scenario, researchers primarily focus on the different architectural features of the DNNs. One of the most popular such architecture was introduced by Oberweger [2] called *Deep-Prior*. This architecture has two salient features. First, they use a pinch layer as a second last layer of the network. They argue that since the latent output space is not purely three dimensional, a pinch layer allows the network to learn a lower dimensional embedding intrinsically. This then help increase the accuracy of their prior network. Next, they train several refining networks, one for each joint, called *ORRef*. These networks use overlapping crops, centered at the join location regressed by

the first network, of the original input image. This network refines the regressed joint location and can be applied multiple times in order to achieve a more accurate result.
More recently, Du et al [19] theorized that by decomposing the task into palm pose estimation and finger pose estimation, we can use cross information to improve the performance of the model Therefore *CrossInfoNet*, split the task into the two subtasks and trains a separate subnetwork for each task. Since the two tasks are mutually distinct, [19] argues that the noise in one of the network will act as valuable information in the other network. Therefore, after performing a primary feature extraction they propose the two subnetworks exchanging their noise information and use this noise in further feature extraction.

2.3 Hand Gesture Recognition

Köpüklü[3] focuses on creating a model that is suitable for online applications. It uses an online video stream as input. In order to avoid the problem of running a costly hand gesture recognition system continuously over the video stream, [3] uses a *detector* and *classifier* combination. The *detector* is a lightweight DNN that is executed continuously over the video stream in order to detect a hand gesture. If a hand gesture if detected, it then activates the *classifier* network that takes in a larger section of the video as the input as makes the prediction of the hand gesture depicted in the video stream. The detector network is able to achieve 98+% recall rate of EgoGesture Dataset[3][4][5].

Chang et al. [17] instead attempt to limit the size of the problem by focusing only on the fingertip instead of the complete hand. They perform frame by frame fingertip detection. By detecting the fingertip of the index fingertip in each frame, they create a trajectory of the movement of the fingertip. They then use this trajectory to predict the letter written in air by the use. Markussen, et al [18] build a word-gesture keyboard that can type by detecting word shaped gestures in mid air. They adapt already existing touch based word-gesture algorithms to work in mid- air. They then projects users' movement onto a display and use pinch as a word delimiter.

Chapter 3: Methodology

3.1 Introduction

In this section, we first discuss the details of the datasets used and the architecture of the machine learning model. Next, we discuss the heuristics used in order to assess the performance of the different models. Towards the end of this section, we discuss the implementation details and the proposed project timeline and milestones.

The data available to a machine learning model is the deciding factor in the final performance of the model. Therefore a large amount on research was conducted in deciding which publicly available datasets were used in the implementations of the architecture. The key factors affecting the choice of a dataset include dataset size, hand pose variability and image quality of samples. Based on these factors NYU hand pose dataset [6] and NVIDIA Dynamic Hand Gesture Dataset [7] were selected as the primary datasets of choice. Besides the aforementioned datasets, the architecture was also implemented on a number of other datasets. The details of these other datasets is discussed in the following chapter.

Commented [d2]: Find a better word.

The architecture of the model is the key innovation of this project. This study posits that by transforming the process of hand joint regression into a hierarchical system, a better performance can be achieved with similar model sizes. This assumption is based on the fact that the position of each hand joint is not truly free from the position of other hand joints. For example, the position and orientation of the wrist almost completely decides the position of the finger bases. Finger bases in turn limit the area in which each finger joint can lie in. Therefore, by using a hierarchical approach the position of wrist joint can aid the model regress the position of the finger bases.

The models were implemented on the HKU GPU Farm that provides a NVIDIA GeForce GTX 1080 Ti GPU for the training of the models.

3.2 Datasets Used

Since the hand pose estimation problem has been made extremely popular due to the easy availability of good depth sensors, a large number of hand pose datasets are available. The datasets fall into two main categories: 1) Static images and 2) Dynamic videos. The static images depict static hand gestures such as depicting numbers using fingers. On the other hand, dynamic videos provide hand gestures that require hand movement such as waving hand. While the static images offer the advantage of having a smaller file size and easier to generate, they suffer from a low number of real world

applications. Since the biggest focus of the Hand Gesture Recognition is its implementation in Human Computer interaction, it is essential that the recognition model can handle gestures that involve movement. This makes the dynamic video models more suitable for Hand gesture recognition. However, dynamic videos have three primary drawbacks. Firstly, since the data is in form of videos, the file size of a single sample is much large than that of images. Large file size coupled with limited storage space results in severe down sampling and compression of the videos. This results in poor quality videos which effect the amount of information that the model can extract from the data. Secondly, shooting videos of hand gestures takes a large time. In contrast, static hand poses are composed by shooting a video of a static hand and extracting frames from the video feed. This allows authors of datasets to create 30 samples per second on average. In case of videos, the authors have to repeatedly perform the dynamic gesture. This severely decreases the speed at which a sample is produced. This results in the fact that most of the video datasets have a smaller number of samples when compared to datasets containing static images. Thirdly, as mentioned earlier, videos have a large file size. A machine learning model that is large enough to ingest the complete video file needs to have a very large input layer. This has a ripple effect and the sizes of the other layers is also large. This results in an enormous number of total parameters to be trained, making both the training process and the execution much slower.

Next, we now look at some of the important datasets used in this project:

3.2.1 **NYU Hand Pose dataset [6]:** This dataset contains 8252 test-set and 72757 training-set frames of captured RGBD data with ground-truth hand-pose information. All the images comprise of various hand poses. For each hand pose, Kinect data from three different angles is captured. Finally, this dataset is presently popular among Hand Pose researchers due to the high variability of Hand Poses captured in this dataset.

3.2.2 **ICVL Hand Pose dataset [8]:** This dataset annotates 16 joint locations with (x,y,z). Coordinates available for each image. The x and y coordinates are measured in pixels while z coordinate is measured mm.

3.2.3 **MSRA Hand Pose dataset [9]:** This dataset contains images from 9 subjects' right hands are captured using Intel's Creative Interactive Gesture Camera. Each subject has 17 gestures captured and there are about 500 frames for each gesture.

3.2.4 **Multiview Hand Pose Dataset [10]:** This dataset captures hand pose from different angles. This dataset not only provides the 3D hand joint locations for each image but also provides the bounding boxes for the hands in the images.

3.2.5 **EgoGesture[11][12]:**This dataset contains 2,081 RGB-D videos, 24,161 gesture samples and 2,953,224 frames from 50 distinct subjects. The authors define 83 classes of static or dynamic gestures focused on interaction with wearable devices.

3.2.6 **NVIDIA Dynamic Hand Gesture Dataset [7] :** nvGesture is a dataset of 25 gesture classes, each intended for human-computer interfaces. The dataset has 1532 weakly segmented videos, which are performed by 20 subjects at an indoor car simulator. The dataset is then split into training (70%) and test (30%) sets, resulting in 1050 training and 482 test videos [13].

3.2.7 **BigHand 2.2M Benchmark Hand Dataset [14]:** BigHand dataset is one of the largest hand pose datasets available. It contains 2.2M images captured from 10 different subjects using kinematic 6D electromagnetic sensors[15].

[15] provides a more comprehensive summary of the major datasets related to hand pose and hand gestures including the information such as number of subjects, frames and annotation types.

3.3 Model Architecture: Hand Pose Estimation System

As discussed earlier, the performance of any model depends on the inherent architecture of the model. In this section we propose a novel architecture that uses the underlying hierarchy of the positions of the hand joints in order to aid the joint position regression process. In order to understand the reasoning behind this approach, it is first imperative to understand the geometry of the hand. [16] describes this geometry very thoroughly. They compare the difference between a 3D coordinates representation of the hand joints against a hierarchical model designed by them. In essence, when we use 3D coordinates to define the hand joint locations, we lose all information regarding the relation between the hand joints. Therefore, each hand joint has to be located independently. However, such a situation is not entirely true. Since the joints are connected, there are constraints on the possible locations of one joint given the position of another joint. For example, consider Proximal Interphalangeal joint, as shown in the figure 1 below.

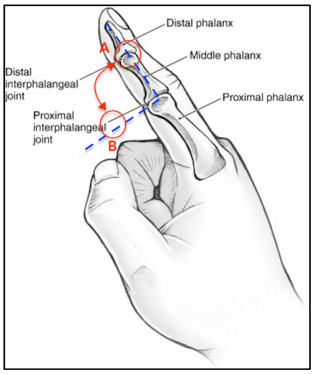

Figure 1. *Insert caption*

Clearly, if we fix the position of that joint, the Distal interphalangeal joint can only take the positions shown by the red arrow in the figure. This means that the position of the Distal interphalangeal joint is bounded by locations A and B in the image since we fixed the position of the Proximal Interphalangeal joint. Given this observation, we concur that knowing the position of the base joints can aid the model to predict the location of the distal joints.

Based on this assumption, we design our model to predict the hand joints in three steps. In the first step, only the Wrist joint and the Metacarpophalangeal joints. In the second step, we predict the Proximal Interphalangeal joints and the Distal Interphalangeal joints. Finally in the third step, we predict the locations of the finger tips. The summary of the network architecture is illustrated in the figure below.

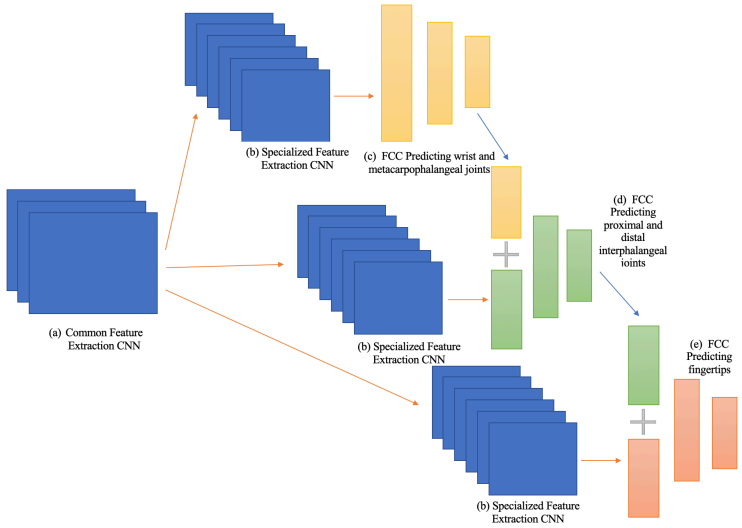

Figure 2. (a) This is a common feature extraction CNN. (b) These secondary CNN networks extract features related to their respective joints. (c) This Fully Connected Component makes prediction of the Wrist joint and the Metacarpophalangeal joints. (d) This Fully Connected Component makes prediction of Proximal Interphalangeal joints and the Distal Interphalangeal joints.(e) This Fully Connected Component makes prediction of the locations of the finger tips.

The input is first passed into a common feature extraction layer. This layer is depicted by (a) in Figure 2. This layer is responsible for extracting features that are common for the next three steps. The main function of this layer is to reduce the time complexity and number of trainable parameters in the network. By making the starting extraction layer common for the three steps, the basic features are extracted only once. If instead a completely independent network was trained for each step, the total size of the network will be closer to thrice the present size. This adds a significant time penalty in my testing. Hence, we instead use a common extraction layer first.

Next, we perform a branching in our network. The rest of the network is divided into three sub networks with similar architecture. This is illustrated as (b) in the figure. These networks extract the features that are more relevant to their respective sub-networks. Lastly, we have sections (c), (d) and (e) of the network. They regress the locations of (i) Wrist joint and the Metacarpophalangeal joints, (ii) Proximal Interphalangeal joints and the Distal Interphalangeal joints and, (iii) the locations of the finger tips respectively.

The earlier discussed idea of using the locations of base joints to predict the locations of distal joints is applied here. As illustrated in the figure, part (c) first predicts locations of Wrist joint and the Metacarpophalangeal joints and next passes these locations to part (d). Part (d) uses this information in addition to the information received from its special feature extraction network to in turn predict the locations of Proximal Interphalangeal joints and the Distal Interphalangeal joints. It then passes the locations of all joints that have been located to the last part , part (e). Part (e) then performs a function congruent to Part (d) to predict the locations of finger tips. This results in the regression of all the joints.

3.4 Measurement Heuristics

After discussing the architecture and workflow of the model, we next look at the performance measurement heuristics used throughout is project.

The project has been divided into 2 separate phases, being Hand Pose Estimation System development and Hand Gesture Recognition System development. The two phases differ in the type of problem they attempt to solve. Hand pose estimation is a regression type problem. This means that the locations of each joint can take any value between a smooth interval. On the other hand, hand gesture recognition is a classification type problem. It attempts to label each gesture with an appropriate name from the set of classes already provided by the dataset. Due to this major difference, the measurement heuristics differ for the two phases.

In the first phase, since the problem is of regression nature, we use a L2-distance based error measurement. The L2-distance can be expressed in the following equation:

$$error = \sum_{i=0}^{all\ samples} (y_i - \hat{y}_i)^2 \quad (1)$$

In the above equation y_i is a vector containing the actual locations of all the joints for a particular sample and $\hat{y}_i$ is a vector containing the prediction made by our model. By taking the difference of the two errors, we can get the error made by our network. Next we square the difference in order to make the loss function invariant of the sign (+/-) of the error. Finally, we calculate the above defined loss function for each sample in our dataset and sum all the resulting values to get the total error made by our model. Using such a error function enforces the training method to try to minimize the distance between the actual position and the predicted position by our model.

For the second phase, we have a classification problem. Therefore we use the accuracy heuristic. The accuracy heuristic is simply the number of mislabeled samples by our model. We compare the actual label and predicted label for each sample. If the labels do not match, we add 1 to our total error. In addition to accuracy, recall and precision heuristics might also be considered in the future. Since the project is still Phase 1 development, these different heuristics are still yet to be completely explored. More research will be conducted on the appropriate performance heuristics at the start of the Phase 2

3.5 Development Details : Division of Work into two phases

The project has been primarily spilt into two phases. The first half of the project focuses on Hand Pose Estimation and the second half of the project focuses on Hand Gesture recognition. Currently, the project is in Phase I. The details of the two phases are as follows:

3.5.1 Phase 1: Hand Pose Estimation Problem

For this phase, I am attempting to create a lightweight hand pose estimating model. To achieve this, I have spent a majority of the time during the summer internship reading research papers regarding the same problem. After gaining suitable knowledge about the state of the art performance on the subject, I came up with my own insight into the problem. This being, the movements of the hand joints is primarily hierarchical and we can use this fact to refine the estimations made by the model.
After this I designed my first model and I have been refining its design ever since. The implementation details of these models are as follows:

3.5.1.1 Implementation Details: Using branched architecture

This is an implementation of the architecture discussed in section 3.3. The implementation slightly differs from the earlier described network due to allow for ease of execution. Here I first discuss the major aspects of the implementation. The architecture consists of three branches. The first branch is responsible for regressing the position of the palm joint as well as the Metacarpophalangeal joints. The second branch then uses the positions predicted by the first branch in addition to its own feature extraction layer in order to detect the Proximal and Distal Interphalangeal joints. Note that unlike the architecture discussed in section 3.3, this layer also acts as a refining layer for the joints detected in the first stage. This is achieved by adding the set of joints predicted by the first layer to set of joints predicted by the second layer. Lastly, the third layer computes the position of the fingers tips using the output of the second layer in addition to its own feature extraction layer. Also note that similar to the second layer, the third layer also acts as a refining layer for the joints predicted by the preceding two layers. Hence, the output of the third layer is the complete set of joints in the hand.

Currently, I am on the sixth iteration of my design. For this iteration, I am combining the initial feature extraction stage for the three subnetworks. This is comparable to the common feature extraction layer discussed in the architecture in section 3.3. This design allows me to significantly reduce the number of parameters that need to be trained. This also as an effect of significantly lowering the execution time of the model. Other than the sixth iteration, I also have design ideas for future iterations. One of the main ideas is to embed a generic hand model into the architecture itself. This change in design allows to restrict the output latent space from purely 3D to 2.5D. I posit that this reduction in the output latent space should have a positive effect to the performance of this model.

Further details of the implementations will be covered in section 4, when we discuss the results measured using different models created during the iterations.

3.5.2 Phase 2: Hand Gesture Recognition Problem

After completing the hand pose estimation problem satisfactorily, I plan to build upon that model for hand gesture recognition. I am currently searching for state of the art research material regarding the subject and different architectures employed for the task. My current plan is to use a RNN based model that would use my model from phase 1 as a data preprocessor. This phase will be elaborated once phase 1 is complete and I can start working on the corresponding problem for this phase.

As mentioned earlier, the project is currently in phase 1. I should be able to complete phase 1 by mid October. After that, I will initially focus on elaborating the details of the second phase and then move in development of the hand Gesture recognition system.

3.6 Project Schedule and Milestones

In this section, I present a proposed timeline for the execution of the project.

3.6.1 Tentative Project Schedule:

This table summarizes the different periods in the project and the main objectives of each period. Each period is in between 4 and 6 weeks in length.

Table 1: Tentative project schedule

1 Sep 2019 – 20 Oct 2019	Complete Hand Pose Estimation models.
21 Oct 2019 – 01 Dec 2019	Working on first prototype for the Hand gesture recognition system. Also work on input stream and output stream implementations.
24 Dec 2019 – 15 Jan 2020	Work on the second prototype of the Hand gesture recognition system.
16 Jan 2020 – 14 Feb 2020	Work on the final implementation of the Hand gesture recognition system.
15 Feb 2020 – 31 Mar 2020	Testing and Debugging
1 Apr 2020 – 18 Apr 2020	Real world testing and debugging

3.6.2 Milestones and Tentative Completion Dates:

Listed below are the major milestones for this project.

Table 2: Major project milestones.

29 Sep 2019	Submission of Inception Deliverables.
20 Oct 2019	Phase 1 complete. A working hand pose estimator ready.
30 Nov 2019	First hand gesture recognition prototype ready.
15 Jan 2020	Second hand gesture recognition prototype ready.
2 Feb 2020	Submission of Elaboration Deliverables.
14 Feb 2020	Third hand gesture recognition model ready.
19 Apr 2020	Submission of Final Deliverables.

Chapter 4: Results

4.1 Introduction

In this chapter we discuss some of the interesting results observed from the currently implemented models. We first shortly discuss the choices for the internal architecture of the different subnetworks discussed during section 3.5.1.1. The we discuss the differences in architecture over the different iterations of the model. We then compare the results measured from the models.

4.2 Implemented Architecture

In section 3.5.1.1, we briefly discussed the difference between the implemented models and the proposed model by this study. In this section, we detail the changes implemented to the models over a number of iterations. In section 3.5.1.1, we also discussed a key property of our network, that being the proposed architecture is independent of the actual details of each subnetwork. Therefore, during our implementations, it is required to select a design for the architecture of each subnetwork. In this study we used the model in [2] called *Deep-prior* as a base model. We split that model and use the segmented subnetworks in place of different subnetworks in our proposed model. We refer to the resulting model as *HandNet*. By doing so, we believe that we can measure the difference in performance and time penalty between *Deep-prior* and *HandNet*.

Currently, I am still retrieving the results of *Deep-Prior*. Therefore, in this section we discuss an interesting observation made when comparing results from two different iterations of *HandNet*. We refer to these two iterations as *HandNet1* and *HandNet2* respectively. The summary of the implementation details of *HandNet1* can be found in the figure 3

Notice that the architecture in figure 3 is very similar to the architecture of the originally proposed model. The difference lies in the first feature extraction layer for each branch. *HandNet1* features an independent primary feature extraction layer. The rationale behind this variation was to confirm the earlier discussed assumption that the basic features would be similar across the three branches of the network. By comparing the performance of *HandNet1* and *HandNet2*, we can conclusively decide the effect of the primary feature extraction. In order to conduct this experiment, the implementation of *HandNet2* was kept completely identical to the original model.

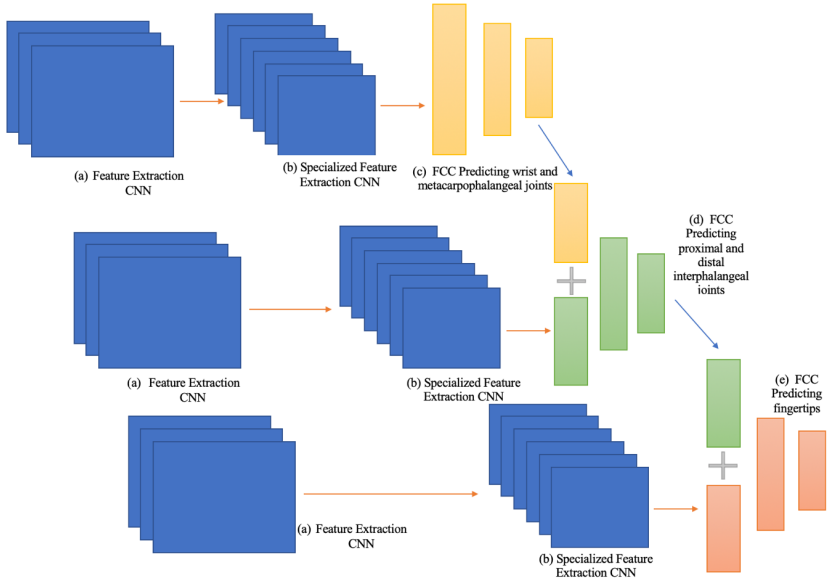

Figure 3. Implemented architecture of *HandNet1*. Architecture of *HandNet1* omits the common feature extraction layer discussed in the original layer and hence has approximately thrice the number of trainable parameters

The two models were trained on NYU Hand pose dataset discussed in section 3.2. The training was conducted on the training sub dataset using Adam optimizer and the min squared error loss (L2 loss) discussed earlier. The generalized performance was measured on the test sub dataset of NYU dataset. We notice that the average loss for HandNet1 is 120.7 with standard deviation of 9.12 while for HandNet2 it is 127.23 with standard deviation of 10.99. The difference between the two models is of 5.8%. This difference is within the margin of variation associated with being due to the randomness with which the input is presented to the models during training. The complete statistics of the comparison of the results of the two models is presented in Table 3 below.

Table 3. Results of *HandNet1* and *HandNet2*.

	Average	Total	Standard Deviation	Minimum Error	Maximum error
HandNet1					
HandNet2			10.9903	95.4217	164.3338

Since the difference in error is within the expected margins, we can conclude that using a common feature layer indeed has little effect on the overall performance of the model. This also confirms that the primary feature extraction layers in *HandNet1* must be extracting similar features from the input. Based on these results, we can conclusively argue that given the decrease in the total number of trainable parameters and the low execution times, using a common feature extraction layer should be preferred over independent feature extractors for the three branches. This finding helps this study focus on the second objective of building a lightweight deep learning model capable of similar performance as more complex models. This study now posits that by applying such refinements to a model architecture, it might indeed by possible to build a model achieving the abovementioned objective.

Chapter 5: Conclusion

Hand Gesture recognition is an exciting field for research in the computer vision. A successful hand gesture recognition system will enable a number of currently unexplored applications. Chief amongst these applications is its applicability in Human-Computer Interaction. Allowing computers to infer meaning from hand and finger movements will open a new form of interaction with the computer that may also increase its accessibility to a larger number of users. Current implementations of this system suffer from two central limitations. Firstly, they require an array of sensors to make a reliable regression and secondly, they have large execution time delays. These limitations make the implementations unportable and unsuitable for real time applications respectively. It is to be noted that research is being conducted in order to mitigate these effects [16][6].

This project directly tackles both these limitations. The key objective of this project is to use a single RGB-D image to regress the locations of the hand joints. Secondly, the project also hopes to investigate the feasibility of using a lightweight model to make the system suitable for real time applications. Hence, the project looks to find an ideal balance between the performance and the speed of the system. Clearly, in order to find this balance, it is key to get a new insight into the hand pose estimation problem.

I posit that this insight can be that the positions of hand joints is not truly independent. Therefore, by using the locations of base hand joints, we can regress the position of the distal hand joints. Building on this insight, I designed and implemented a novel architecture. The salient feature of the architecture is the branching system that occurs after the common feature extraction stage. These branches independently regress the positions of their assigned hand joints supplementing their data processing with the output from other branches. Another fact to note is that the general architecture is not dependent on any particular subnetwork. Any existing popular deep learning networks can be plugged in at the different parts of the main architecture.

During the development of this project, I am currently facing two severely limitations. Firstly, for a significant number of datasets available, the image and video resolution is very low. This results in a large amount of loss of information. This loss of information in turn effects the performance of any

model. Secondly, due to the limited hardware available, each iteration of my models requires a significant amount of time to complete training. This acutely limits the number of the models, I can train and test. I have tried to mitigate the effect of these limitations by using validation techniques while training and keeping my models as lightweight as feasible. By using validation techniques, I am able to monitor the performance of my model while it is training. This allows me to decide when the model has converged and approximate its real world performance while it is still training.

Through this project, I expect that I can successfully build an architecture that has been previously under researched. By assessing the performance of my models, I should be able to show that the earlier discussed observation can indeed provide valuable information that can be coded into the model. For completing this project, I have split the development into two main phases. Currently, I am working on phase 1 which focuses on building a hand pose estimator. I will next directly work on a hand gesture classifier. Currently, I plan on starting work on hand gesture classification towards the second half of October. I will first focus on researching the current state of the art performers in this sphere and search of common features of these models. After that I will design and implement my own hand gesture classifier.

In conclusion, through this project I posit that I will be able to deliver a Hand Gesture Classifier that can reflect the usefulness of performing Hand pose estimation in a hierarchical manner and the feasibility of using a single input feed to perform classification of hand gestures.

References

1. Khan, R. Z., & Ibraheem, N. A. (2012). Hand gesture recognition: a literature review. International journal of artificial Intelligence & Applications, 3(4), 161.
2. Oberweger, M., Wohlhart, P., & Lepetit, V. (2015). Hands deep in deep learning for hand pose estimation. arXiv preprint arXiv:1502.06807
3. Köpüklü, O., Gunduz, A., Kose, N., & Rigoll, G. (2019). Real-time hand gesture detection and classification using convolutional neural networks. arXiv preprint arXiv:1901.10323.
4. Zhang,Y.,Cao,C.,Cheng,J.,&Lu,H.(2018). Egogesture:anewdatasetand benchmark for egocentric hand gesture recognition. IEEE Transactions on Multimedia, 20(5), 1038-1050.
5. Cao, C., Zhang, Y., Wu, Y., Lu, H., & Cheng, J. (2017). Egocentric gesture recognition using recurrent 3d convolutional neural networks with spatiotemporal transformer modules. In Proceedings of the IEEE International Conference on Computer Vision (pp. 3763-3771).
6. Tompson, J., Stein, M., Lecun, Y., & Perlin, K. (2014). Real-time continuous pose recovery of human hands using convolutional networks. ACM Transactions on Graphics (ToG), 33(5), 169.
7. Gupta, P. M. X. Y. S., & Kautz, K. K. S. T. J. (2016). Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural networks. CVPR.
8. https://labicvl.github.io/hand.html
9. Sun, X., Wei, Y., Liang, S., Tang, X., & Sun, J. (2015). Cascaded hand pose regression. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 824-832).
10. Gomez-Donoso, F., Orts-Escolano, S., & Cazorla, M. (2019). Large-scale multiview 3D hand pose dataset. Image and Vision Computing, 81, 25-33.
11. Zhang,Y.,Cao,C.,Cheng,J.,&Lu,H.(2018). Egogesture:a new dataset and benchmark for egocentric hand gesture recognition. IEEE Transactions on Multimedia, 20(5), 1038-1050.
12. Cao, C., Zhang, Y., Wu, Y., Lu, H., & Cheng, J. (2017). Egocentric gesture recognition using recurrent 3d convolutional neural networks with spatiotemporal transformer modules. In Proceedings of the IEEE International Conference on Computer Vision (pp. 3763-3771).
13. Kopuklu, O., Kose, N., & Rigoll, G. (2018). Motion fused frames: Data level fusion strategy for hand gesture recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 2103-2111).
14. Yuan, S., Ye, Q., Stenger, B., Jain, S., & Kim, T. K. (2017). Bighand2. 2m benchmark: Hand pose dataset and state of the art analysis. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4866-4874).
15. Doosti, B. (2019). Hand Pose Estimation: A Survey. arXiv preprint arXiv:1903.01013.
16. Xu, C., & Cheng, L. (2013). Efficient hand pose estimation from a single depth image. In *Proceedings of the IEEE international conference on computer vision* (pp. 3456-3462).
17. Chang, H. J., Garcia-Hernando, G., Tang, D., & Kim, T. K. (2016). Spatio-temporal hough forest for efficient detection–localisation–recognition of fingerwriting in egocentric camera. *Computer Vision and Image Understanding*, *148*, 87-96.
18. Markussen, A., Jakobsen, M. R., & Hornbæk, K. (2014, April). Vulture: a mid-air word-gesture keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1073-1082). ACM.
19. Du, K., Lin, X., Sun, Y., & Ma, X. (2019). CrossInfoNet: Multi-Task Information Sharing Based Hand Pose Estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 9896-9905).