**ABSTRACT**

In this proposal we will be discussing our idea for a web app that can recognize various hand signals to produce a working hands-off camera. We will additionally include features that include fun filters to use with the app, such as facial recognition box and grayscale images when shown a specific symbol. These various forms of computational photography combined with the analysis of hand images will come together to make a new way to experience photography.

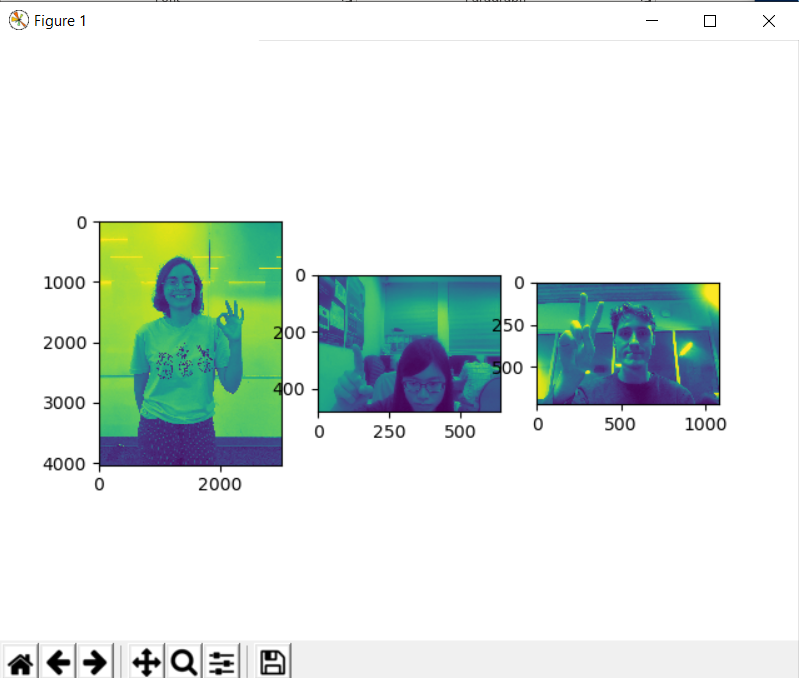
**1. Introduction**

Our primary objective is to build a model that can identify hand gestures using a deep learning network. We want our web app to recognize a peace sign and an open hand as a photo timer for 1-5 seconds, respectively. When the application recognizes a fist, it will stop the program. It will do a variety of photo editing features based on more hand gestures. We plan to achieve this gesture recognition by utilizing deep learning [1]. We researched gesture recognition and began to plan how we will implement that into our project. Once we trained the model, we made the algorithm which will take photos and videos when it recognizes a hand gesture. The process of this project took many different approaches in trial and error, but after changing various aspects of our initial idea we were able to create an algorithm that uses gesture recognition to take photos and in the process learned an invaluable amount of information that we can use in the future. In this report we discuss how exactly our app works, the strenuous process in coming up with out final product, a description of each algorithm and filter we researched (along with whether we used it or not), an overall view of what we learned, and how we could improve the project in the future.

**2. How It Works**

Through a lengthy process of trial and error we finally settled on the following implementation of our algorithm. When the program begins to run it will continuously take photos and process them to check if a gesture is being shown. Each photo will be denoised by using a non local means filter, and then simplified to be tested against the model by using a GMM based subtraction. If a gesture is recognized for three continuous frames (in order to filter out wrong readings) it implements a timer for that amount of time. For example, if you put up five fingers a five second timer will go off. It will set a timer for fingers 1-5, to give the user a variety of options depending on the poses they would like to make or time. Once the timer ends a photo is taken. The photo being taken means the process has ended and it will again continue to check whether a hand gesture is being shown. If an ‘okay’ sign is recognized the images are converted to grayscale. We added

additional features when the application detects a hand gesture done consecutively. For two readings of five fingers, the video feed will show GMM Based Background Subtraction. For two readings of three fingers, the app begins to do facial recognition. When you are done with the application a fist will close the application.



Example of Grayed Out Figures1

**3. Process**

A person looking at the camera

Description automatically generated

Sample Photo of the Bharatanatyam Mudra Dataset [2]2

We utilized the Comprehensive Bharatanatyam Mudra Dataset [2], a large dataset (over 80Gb or 35000 photos) of various hand gestures, which is comprised of rotating hand gestures that also move throughout the frame. There was also dummy data where no hand gestures were shown. We planned to input some photos into the data set with filters so that our algorithm could work through blurred and noisy images. These measures ensure that our model can work in a variety of events. As a way of testing tensorflow 2.0, we have built a fully connected neural network tested with MNIST input data and trained on our larger data set. We also brainstormed several ideas of which algorithm to implement. We are training the last layers of the VGG16 network, that is pre-trained on IMAGENET (transfer-learning). The goal was to get a very good result of identifying non hand gesture type and a good enough (80%-90%) of recognizing the right-hand gesture. Afterwards we planned to make the app which will take an unsaved photo every second, filter it so that it matches out dataset, and process whether it sees a hand gesture. Once we proceeded to complete more of the project, we realized that it would be better to create a web application, so we changed the direction of our project.

One unforeseen circumstance of utilizing such a large dataset is that our computers take a significant amount of time processing and training the model. To shorten the time it takes to train the model, we divided up the training sections between our three computers, primarily with the second algorithm that was described in the algorithms section. We tested the model we had and used the mean shift algorithm on photos we take ourselves to see if our trained network can recognize the images.

While working with the mean shift algorithm we realized that the algorithm was not as efficient as we would like. We decided to instead use a GMM based background subtractor in order to use multiple photos and decide which parts are the background. To make this algorithm more precise, we created a data set in the room that we will potentially demonstrate so that it can recognize the background we will use during live testing. Part of this data set also switched between left and right hands, to try and ensure that there is no bias towards one hand versus the other. One issue that we had with this data set was converting from the iPhone photos file HEIC to one that python will have an easier time reading. We took around 300 of these photos, which in retrospect was not enough compared to how large the original data set we trained with was.

A person standing in front of a mirror posing for the camera

Description automatically generated

Example of ‘Okay’ in Data Set Created by Tania3

With each data set we defined around 9 different symbols: each hand gesture for 1-5, an open hand signal, a peace sign, an okay symbol, and a fist. We connected the peace sign with the two, and the open hand with a five where the fingers are put together so that there was less confusion in training the model. We ended up training with three different data sets, which ensure that our app not only recognizes the gestures we need, but it can recognize most of our group members and the background that we demonstrate in.

A picture containing person, man, indoor, wall

Description automatically generated

Example of ‘Peace’ in Data Set Created by Jacob4

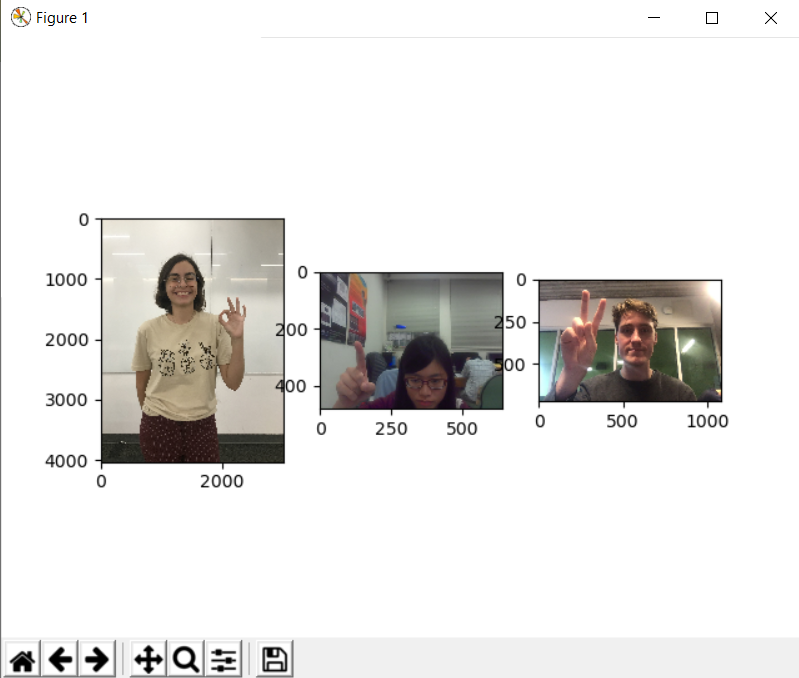
Once we worked through training the model we decided that instead of an app we will work through python. By working through Python we are able to access the model through any application as long as it includes a camera, which makes it more accessible than an iPhone app. In order to do this, we needed to make sure we were able to denoise the photos successfully because computers are not known for having quality cameras.

**4. Algorithms and Filters**

**4.1. Mean Shift Algorithm**

The Mean Shift Algorithm distinguishes different objects and groups together similar colored objects. It will then output an image where all the similar objects are grouped together in distinct colors. It works by using clusters of each color and separating them into different groups. Since the data we wish to recognize is a hand, we tried to use it because we assumed it would be able to isolate the hand in real time from the white background and then decipher which hand gesture is being shown.

In the end we decided to not pursue this algorithm because it was not accurately able to separate the persons arm from the hand. A more efficient way to isolate the hand was to use a GMM based subtraction with a dataset we created to isolate the hand alone from the background. Despite not using this algorithm, we learned about a way to isolate different objects in a photo through color. It would be useful if we were doing a full body isolation against a white background, which has potential for a different project.

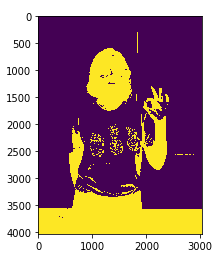


Example of previous photos going through Non Local Means Filter5

**4.2. Non Local Means Filter**

The Non Local Means algorithm was used to denoise images. We needed to denoise images in order to have a more accurate reading of each image. The algorithm replaces every pixel’s luminance as a weighted average of all pixels’ luminance’s. The key idea is that weight is bigger when two pixels have similar luminance and the noise substance can be eliminated by averaging them together according to central limit theorem. This was necessary because it will get rid of the outliers and prevent our model from fitting to the noise.

Part of why this algorithm is so important is that we turned our program into a web application, and we needed to denoise the images that came from computers. A denoising filter also helps overall, because noise can affect any camera.



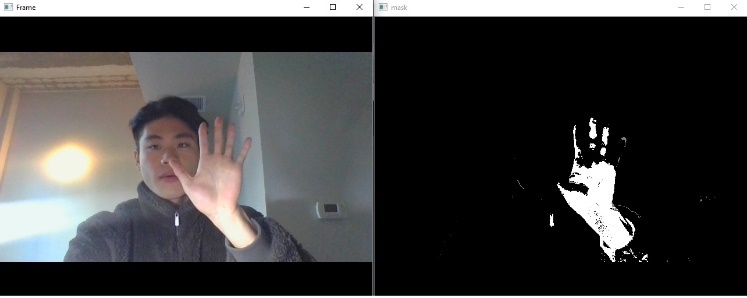
Example Watershed Algorithm6

**4.3. Watershed Algorithm**

This algorithm was not used in our final product, but it helped us understand how the following algorithm (GMM Based Background Subtraction) works. We did not use this algorithm because although it is able to erase the background, it was not able to differentiate between the hand and the rest of the arm. Unlike GMM Based Background Subtraction, this algorithm is able to analyze between background and foreground using one image versus a series of images.

**4.4. GMM Based Background Subtraction**

The Gaussian Mixture Model (GMM) Based Background Subtraction was used to separate the background from the hand so that our predictions could be accurate. Since we are dealing with real time processing, we need the computations done quickly. This is the biggest reason we implemented this algorithm. Separating foreground and background can significantly reduce computational efforts. GMM background subtraction models every pixel as a Gaussian Mixture and compares with their previous values. If a gaussian distribution exceeds threshold, it will be considered as background. We also could convert the results to one bit grayscale images and feed these to our model which would have reduced computations even more [3]. When comparing each image as the program continually runs, it can successfully and continuously isolate the hand. This algorithm made our jobs exponentially easier, as an isolated hand can be read by the model much more effectively. A bonus of this algorithm is that because this works continuously, it can work in real time with our camera. Working in real time is essential because ideally a person would not want to wait a long time to have a timer begin counting down, it defeats the entire point of having specific times for the timer. The example photo below shows how the background subtraction works during a video, and you can see how clearly the hand is able to be isolated.



Example GMM Based Background Subtraction7

**4.5. CNN(R-CNN) + VGG16**

Instead of building a model from scratch, we thought it would be better if we load a pre-trained model and adapt to our taste. We picked VGG-16 because it was the most prevalent and

accurate one. Then we built CNN layers with trainable parameters. Our model could produce accurate results in a short period of time. R-CNN does selective search on the image which gives us higher accuracy. From the previously discussed Comprehensive Bharatanatyam Mudra Dataset [2], is a way to use two convolutional networks to recognize hand signals. We should be able to achieve around 94% of accuracy on the dataset, when pre-processing with CNN and training the last 3 layers of VGG16 (comprised of 2 fully connected and 1 Batch normalization layer). We already figured out how to efficiently unpack all our data and size it to put it in the algorithm. We have also found a nice example, which we follow, that also trains the VGG16 net [2].

Since our application deals with real time signal processing from camera, computation should be fast to minimize the delay. We have found an algorithm that is computationally efficient, but also great in detecting objects in images.

**4.6. One Layer Deep Net**

We started with the well-known MNIST database, which has a great example on simple neural nets on the Kaggle platform and tested it out. On the MNIST database, the algorithm manages to achieve an accuracy of 98%. However, the database consists of gray colored pictures of handwritten letters. Objectively speaking, this task is easier to solve than ours since we are working with color and several hand gestures. We still wanted to train the net because of its features of 1 convolutional hidden layer, trained with dropouts and VGG16 transfer learning as activation function. We are training this algorithm on our database to see what it does. Part of why we are training multiple algorithms is because of the MNIST algorithm is simple and has interesting features such as plotting hand gesture features in its result. In the end we decided to move forward with VGG16 transfer learning versus the MNIST because VGG16 transfer learning was able to produce results with 92% accuracy.

**4.7. Facial Recognition**

The algorithm generates simple features from face images and uses boosted tree classifier. It breaks image down into pieces by a window and checks if it contains certain structures. The results of the window are fed into a weak tree classifier. The weak classifier will be augmented by another weak classifier, hence decreasing the variance of the classifier.

**5. Packages We Used**

Along with the common packages that are generally used in python (matplotlib, numpy, etc), we learned how to navigate and utilize through TensorFlow 2.0, Keras, and opencv. These packages were invaluable to our project because they provided a good number of processes that made our program possible to implement. Tensorflow 2.0 and Keras were utilized to train our neural network.OpenCV, or more specifically cv2, was helpful to us to process our videos and initialize both the GMM based background subtraction and the facial recognition.

**6. Results**

When tested, our algorithm ended up predicting photos with a 92% accuracy. In the reality our algorithm can be a bit of a hit and miss. Part of this is likely due to a bias that formed with our data set. The majority of our data set came from Bharatanatyam Mudra, in which the data set was done by the same person with the same angle and background in each photo. Now the algorithm does well when compared to photos that are done in a similar setting, but if we were able to train the model for longer (compared to the 30+ hours we already committed) we would have done a much larger variety of photo. There are several resources for hand gesture data sets online, but we only found a few that had the exact hand gestures we needed. Whenever we made our data sets we only included around 500 photos of our own, which in hindsight is nothing compared to the thousands that the model was already trained off of. Despite having a biased model, we are still able to get accurate readings in the right conditions. Because of this we were not able to do a live demonstration during our presentation, but we were able to make a video in the right conditions that show accurate readings being made.

**7. What We Learned**

Throughout this project we learned a great deal about deep learning and the real-world applications of it. We learned about how much of the design process is going through trial and error. We initially tested two different ways to train the model, through VGG16 transfer learning and through MNIST and decided that we would proceed with VGG16 transfer learning because it gave us more accurate results. Afterwards we tried to implement different ways of sweeping and isolating the hand in order to get accurate readings. While none of the approaches were wrong necessarily, we learned how to gauge what we were capable of and what algorithms worked best with our specific parameters. We needed a program that could work in real time so that people only had to wait the desired timer they chose.

Besides the trial and error design process, we learned about how important training a model is. We were lucky to find a relevant data set with hand gestures, but we also had to make our own to have more accurate readings and it was a long process. We needed consistent backgrounds and sufficient data to be read, so even with original data set we used we ended up making over 500 photos of our own to test it further. This was insignificant compared to how large the original data set was, which resulted in bias in our final product.

To go along with that, we learned how to handle immense amounts of data. We had to deal with 145,000 photos size of 80gb so we came up with a method that can efficiently load and handle the data. We had to learn how we could overcome the problem of the limited number of files our computers can open, the limited memory size and limited CPU power. The problem of the limited files we can open was solved by using with/open statement in python. We open each file separately and store the contents in memory. The problem of limited memory was solved by training in batches which helped us implement bagging. Since we could only store 8,000 images in memory, we chose our batch size to be 8,000. The limited power of CPU was solved by limiting our data

and training epochs.

Although most of our group was well versed in python, this project was still valuable for the rest of the group in learning how to navigate between different studios (Jupiter Notebook versus Spyder) and using PowerShell at some points because some computers would not install the models we needed otherwise. In terms of working as a group, we learned about how to work with each other’s coding styles as we split up the code to assemble it together in the end.

**8. Moving Forward**

If given the time and resources, we would move forward with this application by designing a user-friendly interface so that it can be convenient to be used by anyone. We would look into fixing the delay that can occur when the photos begin to back up, and weigh the pros and cons of having the gesture be accurate three times in a row versus less for a shorter runtime but potentially less accurate in the long run. We would add in our original idea of having video capability in the application, so that it can be used for every form of camera needs a person could have. If we were able to train the model again, we would have more variety between our data sets, because it works significantly better in specific cases and lightings as of now. Due to this bias we only have accurate results in specific conditions, and if we could do it over, we would have found much more variety in the data set. Overall, we are incredibly proud of the work we were able to accomplish.

**9. References**

[1] Oyedotun, O.K. & Khashman, A. Neural Comput & Applic (2017) 28: 3941.

[2] Parameshwaran, Anuja P., et al. “Transfer Learning for Classifying Single Hand Gestures on Comprehensive Bharatanatyam Mudra Dataset .” IEEE Xplore.

[3] X. Lu and C. Xu, "Novel Gaussian mixture model background subtraction method for detecting moving objects," 2018 IEEE International Conference of Safety Produce Informatization (IICSPI), Chongqing, China, 2018, pp. 6-10.