Investigation of HCI systems for users with limited mobility

Ruta Kulkarni
Department of Computer Science
Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY 14586
rk6930@rit.edu

***Abstract*—Paralysis is the loss of voluntary muscle control in which a person loses the ability to communicate using his or her limbs. In some cases of paralysis, such as pseudocoma, a person loses his or her abilities to speak also. This project examines issues and approaches for building a human-computer interaction system for such people who can still move their neck. An initial design of a head-controlled system is built from the facial landmark points extracted using Dlib and OpenCV. Several different classifiers are compared, including K-NN, Random Forests, Decision Trees, and Neural Networks. Different feature selection techniques were used, comprising both explicit and automatic. One of these resulted in a 100% accuracy when predicting where a person is looking in the range of a 3*3 grid using 10-fold cross validation on the labeled data. This paper makes a small contribution in the independence of pa'ralyzed people.**

***Index Terms*—Human-Computer Interaction system, Facial landmarks, Autoencoders, Feature extraction**

I. Introduction

Researchers in human-computer interaction have always been trying to explore novel ways of communication between a user and the system. The research covers a wide range of topics including designing a new system, ways of implementing the interface (libraries) and determining if a user is human or computer. These interfaces can be used on devices like tablet or phone but can be difficult to use for some people eg. people with paralysis. Paralyzed people do not have control over most of the muscles except their eyes and some of the face muscles. These muscles can be tracked by either mounting external sensors on the user's body or building contact-less vision-based systems [7]. Contact-less systems offer much more comfort for the users and are researched widely. Interesting work has been proposed such as building a new mouse whose operation is based on the head movements of the user [3], and a robot wheelchair with hand gestures based controller to assist the disabled people. [13].

Eye contact and gaze detection are important cues in communication. Eye tracking refers to tracking the eye movements or the Point Of Gaze (POG) [10]. 'Codeblink' [11] is a system that uses eye blinks as morse code for communication. It can also differentiate between voluntary and involuntary blinks. The movement of cheek muscles was used to build a communication system for Stephen Hawking who was the longest survivor of ALS - Amyotrophic Lateral Sclerosis, also known as Lou Gehrig's disease. These systems are the motivation behind this project.

In this project, facial landmarks are used to accurately predict the direction in which a subject is looking given a 3-by-3 grid. This work is based on the previous work of Daniel, called Intellicursor [1], which solved the problem of losing the cursor when using two monitors. Intellicursor is a system that takes multiple facial landmark points as input and predicts which monitor the user is looking at using binary classification on a multi-layer perceptron. The work is expanded in this paper by accurately predicting the class of the target box within nine boxes. Best possible features are extracted from the facial landmarks data to train several classifiers including Decision Trees, K Nearest Neighbours, Random Forests and Neural Networks. These features are the ratios of Euclidean distances between the point on the nose and the perimeter points. Using ratios as features instead of distances help us compensate for optical magnification. After manually extracting the features, the next thought was to try automated features from neural networks. Two experiments with autoencoders on numeric and image data are performed for the task of extracting features and using them for classification. Results from all the models including neural networks are then compared.

The previous work and background is discussed in the section II. The section III defines a general approach to handle the problem of predicting where a person is looking at. The detailed methodology of the implementation is explained in the section IV. The section V concludes the project.

II. Background

Amyotrophic Lateral Sclerosis(ALS) - Stephen Hawking is the longest survivor of ALS and wrote about 10 books only with the help of his finger [9]. He is an epitome of using augmentative and alternative communication (AAC). To a lot of people, it looks like a mystery how the person is communicating. It includes gestures, communication boards, etc. Some systems are non electric i.e. they just use books and picture and people can point at those things, while others are high-tech, i.e. they make use of the technology available. Stephen Hawking is considered one of the most brilliant theoretical physicists since Einstein and continues to be an

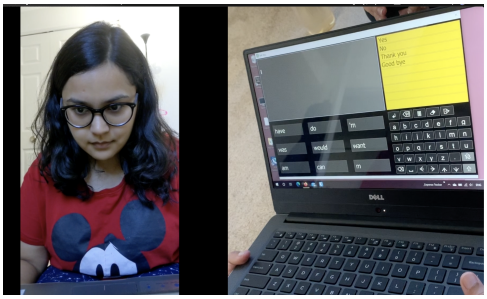

Fig. 1. Trying out the ACAT software

inspiration in spite of all the challenges in life. He used a software developed by Intel, called Assistive Context-Aware Toolkit (ACAT) [4] which was customized according to his needs. Intel continued to build and develop this software for around 20 years and released it in public in 2015.

Camera calibration in ACAT : The facial gestures are detected and the system is calibrated accordingly in this part. There are gestures for eyebrow movements, cheek muscle movements, lip movements. The head must be kept still and only the face muscles must be moved in order to calibrate properly. (For Hawking) Infrared sensors were places on the eyeglasses and cheek movements were detected. OpenCV was also used as shown in the image 1, where some facial points are detected. Each box in the displayed grid is highlighted in sequence and then selected based on the cheek movement. There are some inbuilt phrases as well as the option to convert text to speech.

Eye-tracking is another method that can be used for human-computer interfaces. There are four methods of eye tracking[6][10]:

1) Electro-oculography - A pair of electrodes is placed near the eyes and the movement of eyes creates a potential difference which can be measured.
2) Infrared-oculography - It measures the intensity of the IR light reflected by the sclera. The degrees of movement this method can measure is comparatively small.
3) Video-oculography - Images from multiple cameras are used to determine the exact movement of the eyes. Most widely used method.
4) Scleral Search Coils - Small coils are embedded in the contact lens. The orientation of these coils in the magnetic field is used to track the eye movement.

A system with DIY eye glasses which have PS3 eye camera, 4 IR LEDs, copper wire and a battery was built [6]. The software used was called 'Eyecan', written in C++. Method -

1) Adjust the camera so pupil can be detected.
2) Calibration without blinking of eyes - takes 2 seconds to calibrate s single point. The eyes should then be moved left to right, top to bottom to calibrate.
3) Press F8 to connect with cursor on the screen.
4) Click on virtual keyboard and start writing. Different blinks are assigned meaning : Blink more than 0.5 seconds is a mouse click action, longer blink is a double click action.

The best response was 57 seconds to write the sentence 'Eyewriter is the best invention'.

Another way of communication is the Brain Computer Interfaces (BCI) systems which take brain signals, analyzes them and translates them to carry out the desired action. Also called as neural -control interface, Brain-Machine Interface (BMI), etc. The research about BCI systems began in 1970 at UCLA, to explore merging of brain and computer, but with the motivation of helping paralyzed people. It can be used by chronically injured people who have lost limb, damaged spinal cords and also most useful for the people with LIS or ALS : as their brain works perfectly but the body is paralyzed.

Braingate is one of the most significant systems owned by Braingate co. It consists of a sensor implanted in the brain which is made of hundreds of electrodes to sense the firing of the neurons in the particular area of the brain. These firings are converted to electrical signals and then sent to a decoder which in the end controls an external device like a robotic arm. Braingate's first patient was a quadriplegic who had his spinal cord damaged(and therefore lost control of the limbs) because of a sports injury. With sensors in his mind, he was successfully able to carry out regular tasks like moving his arms for sending emails, operating tv or getting food.

P300 is a wave that occurs in the EEG of the brain when the user is trying to identify the target [14]. Its amplitude varies with the improbability of the target. The magnitude, topography, presence and timing are together used to study the cognitive function in the decision making process. One of the oldest application of P300 is in lie detectors. Recently, these wave are also being used in the BCI applications for paralyzed patients. In a system [5] proposed by the some researchers in Chennai, the 8 channels of scalp EEG are analyzed out of which 5 shows the P300 wave. These 8 channels are recorded using electrodes placed on the scalp of ALS patients. A linear classifier is used on the features like mean and range of these waves, to classify the desired and non-desired stimuli, i.e. the characters user desired to select and the undesired. The rows are randomly highlighted and the users are asked to focus the attention on the desired character, to spell a few predefined words. After collecting the wave data from all the patients for different words, the mean is calculated for each channel and for each epoch (each epoch of 32 samples). Similarly, range is also calculated. The classifier then uses feature matrix of all the range and means for all the channels. The classifier can then be used to improve the accuracy of a speller BCI. The result of this study was an 89.5% accuracy.

Neuralink [8] is a startup by Elon Musk in 2017 which aims to create a link between neurons inside the brain and the machine, with an aim of understanding brainly disorders as well as enhancing human capabilities for a better future. It will be useful to operate artificial limbs and treat diseases like Alzheimers and Parkinson's. People are also talking about Musk bringing us to the 'Matrix' world as Neuralink will

also try to recreate 'Neural lace', the fictional method of transferring information from brain to machine.

III. Approach

A. Facial Landmarks detection

Computer vision engineers and researchers have always been trying to understand human faces. In early days, localizing a human face and drawing a bounding box around it was a very hot research topic. After that was successful (thanks to Viola, Jones), the next step was to find particular features on the face accurately. Facial landmarks are used to localize and represent features of the face such as nose, eyes, lips, jawline. Thus the two steps in facial landmarks detection are localization of the face and then finding the key features in the ROI of the face.

Algorithm 1 Real-time facial landmarks detection

1: Initialize the dlib's HoG and SVM based face detector and the facial landmark predictor.
2: **for** each frame **do**
3: convert it to gray-scale.
4: Apply the face detector to localize the faces.
5: **for** each face in the image **do**
6: apply the predictor and then return the (x,y) coordinates.
7: **end for**
8: Draw circles on each of the points in the returned numpy array to visualize the facial landmarks points.
9: **end for**

In order to detect the facial landmarks [12], Dlib is used. It works in four steps as shown in 1:

1) Initialize the dlib's HoG and SVM based face detector and the facial landmark predictor.
2) Grab the frame from the video stream that we have, convert it to gray-scale.
3) Apply the face detector to localize the faces.
4) Loop over each face in the image to apply the predictor and then return the (x,y) coordinates.
5) Draw circles on each of the points in the returned numpy array to visualize the facial landmarks points.

This library provides with 68 different points, each having its (x,y) coordinates. An image of these coordinates is shown in Figure 2.

B. Feature extraction and comparison

After detecting the facial landmarks, we have a dataset that we can use to train any model. This dataset is the facial landmarks points, each having 136 (x,y) coordinates obtained from the dlib's facial landmark detector. These coordinates are a large sequence of numbers and feature selection is important in order to extract meaningful information from it. Feature selection is a process of determining what are most important characteristics of the data that will help us in predicting the class label for the new data. It is critical as we do not need all the raw data every time, some of it is noise and would lead to either 1. Over-fitting our model or 2. Wrong training causing wrong predictions. The extracted features can now be used to train different machine learning models.

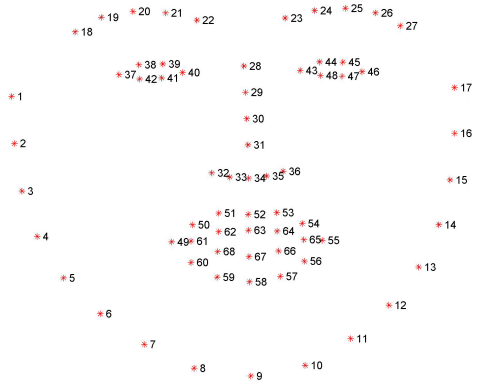

Fig. 2. Landmark Points

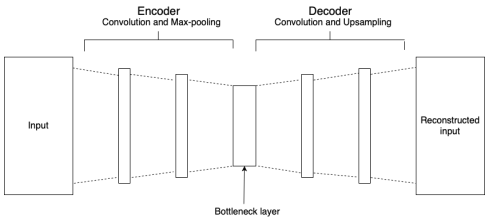

Fig. 3. Convolutional Autoencoder architecture

The next obvious thought after this is what if we use a neural network and let it extract the features for us? An autoencoder is a type of neural network that learns encodings of the given input in a supervised manner by setting target values equal to inputs. This is typically for dimensionality reduction and to teach the network to ignore the noise. The compression and decompression functions are data-specific and learned automatically from the data rather than being engineered by a human [2]. As this algorithm is data-specific, its applications are not useful in image compression as JPEG does a better job. Autoencoders are rather good at de-noising images and dimensionality reduction. It applies back-propagation setting the target values equal to inputs, in other words it is trying to learn the approximation of the identity function.

A simple architecture of a convolutional autoencoder is shown in the fig. 3. The encoder can be written as:

$$z = \sigma(Wx + b)$$

where z is the latent vector. The decoder can be written as:

$$x' = \sigma'(W'z + b')$$

where x' is the reconstructed input.

IV. Methodology

A. Camera Calibration

Camera Calibration in this project is to make the system usable for the current user. When a new user starts using the system, he/she has to calibrate the webcam according to its

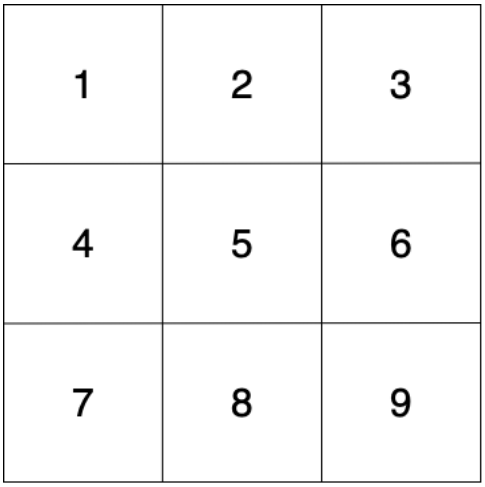

Fig. 4. Grid to display on screen

alignment with the user's face. For this, a grid is displayed on the screen as shown in the fig. 4 using OpenCV.

1) Data Collection: Data is collected in the process of camera calibration, a 3-by-3 grid is displayed and the user is asked to look at the boxes sequentially. At each box, the space key is pressed. As the key is pressed, the current x,y coordinates of all the points on the face are stored. These steps are performed 25 times in total which gives us 25*9 i.e. 225 data points.

2) Data Annotation: Data points are given labels from 1-9 as the numbers on the grid as shown in 4.

B. Calculation of features

There are in total 68 points that define facial features and (x,y) coordinates for each point which means 136 values in total. To make sense of the location of all the points, two features are introduced that are based on these coordinates. The two features are called 'lr-ratio' and 'tb-ratio'. The lr-ratio is the left-to-right ratio of the face and tb-ratio is the top-to-bottom ratio calculated from the points on the face.

From the Fig. 2, the centre point is the point 31, i.e. the tip of the nose. The left distance is the Euclidean distance between point 2 and point 31 while the right distance is the distance between point 16 and point 31. The top distance is between point 28 and point 31 and the bottom distance is between point 10 and point 31. The ratios are then calculated as:

$$\text{lr-ratio} = \text{left distance} / \text{right distance}$$
$$\text{tb-ratio} = \text{top distance} / \text{bottom distance}$$

The ratios were chosen as the features and not directly the distance between points because the ratio will help us normalize the distance between the webcam and the face. The ratios between distances will remain the same irrespective of the distance between the webcam and the face. For example, the lr-ratio in fig.5 is 0.40 and the lr-ratio in fig.6 is 0.41. This clearly shows us that using the ratios as features will help us normalize the values in itself, which means, the values for each box will be in the same range irrespective of the distance between the webcam and the face, in turn compensating for optical magnification.

TABLE I
LR RATIO VALUES

0.51757	1.09369	1.76321
0.63124	0.94722	1.30838
0.46699	0.85457	1.34865

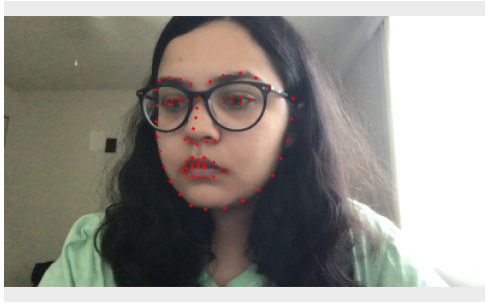

Fig. 5. Looking at point 4 and closer to webcam

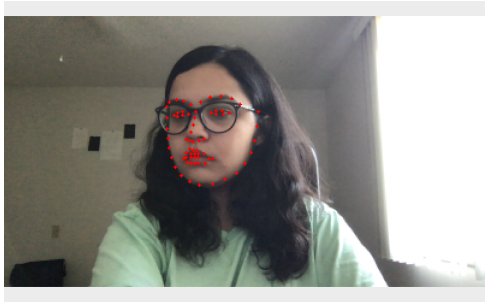

Fig. 6. Looking at point 4 and farther to webcam

C. Training and Validation

The 225 points are trained on three different models:

- K Nearest Neighbours: It is a simple algorithm that saves all the data and classifies new data based on similarity measure. Approach:
  1) Load the data and initialize 'k' as the chosen number of neighbours which is specific for the data. Here, k=10 worked the best.
  2) For the query input, calculate its distance from each saved data point and add it in an ordered collection.
  3) Sort this collection in the order of smallest to largest and return the mode of the first k entries.
- Decision Trees:
  1) Split the data on the feature that has highest information gain from the root. We have used the information gain as the criterion but another option is to use gini impurity instead.

TABLE II
CONFUSION MATRIX FOR 3*3 GRID ON THE RANDOM FOREST CLASSIFIER

2	0	0	0	0	0	0	0	0
0	2	0	0	0	0	0	0	0
0	0	2	0	0	0	0	0	0
0	0	0	2	0	0	0	0	0
0	0	0	0	2	0	0	0	0
0	0	0	0	0	3	0	0	0
0	0	0	0	0	0	3	0	0
0	0	0	0	0	0	0	3	0
0	0	0	0	0	0	0	0	3

2) Repeat the splitting process for each child in the tree until all leaves are pure i.e. all the samples at each leaf node has the same class label.

- Random Forest: Random Forest is an ensemble of uncorrelated trees. Each of the trees vote for a class prediction and the class with highest number of votes is our predicted label. The concept is simple - a large number of unrelated components forming a committee will outperform the individual constituents. After multiple trials, 50 number of trees worked the best in this case.

K-fold cross-validation: Cross-validation is an evaluation method to estimate the skills of a machine learning model. This method has one parameter 'k' which determines the number of groups the data should be split into. The k used here is 10, after different trials. It works in following steps:

1) Shuffle the dataset randomly and then split it into k groups.
2) For each group:
   a) Hold this group out as the test data.
   b) Take the remaining data as the training data.
   c) Train the machine learning model on the training data and evaluate on the test data.
   d) Save the accuracy score and discard the model.
3) Calculate the mean of all the accuracies are the overall accuracy of the model.

It can be seen from the fig. **??** that the Random Forests achieved

From the confusion matrix in the table II, we can see that the results are 100% perfect. All the numbers are in the diagonal which means they are correctly classified.

D. Autoencoder

After manually finding the features for the machine learning models, the next step was to try the obvious, the neural networks. There would be no need to manually find out the best features and we can feed all the data and let the neural network learn the best features for us. Two different experiments were done with the autoencoders:

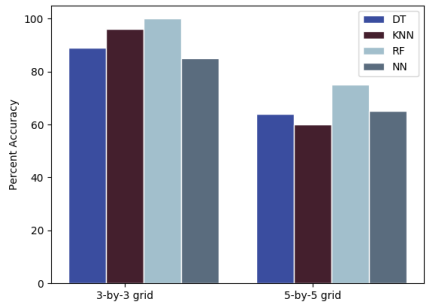

Fig. 7. Accuracy graph

Algorithm 2 Convolutional Autoencoder

```
for i ← 1, 3 do
    Conv2D with activation relu
    Max-pooling
end for
encoded layer last layer in the loop
for i ← 1, 3 do
    Conv2D with activation relu
    Upsampling
end for
Compile with binary cross-entropy loss and adadelta optimizer
for i ← 1, 200 do
    Train the model
end for
encoded layer.predict() to get the encoded features
```

1) Autoencoder with the numeric data A vanilla autoencoder was trained on the same dataset that was used to train the other classifiers. The autoencoder was build using the Keras library, Tensorflow on the backend. The procedure to build an autoencoder is:
   a) Build the model: In the first step, we initialize a model using Keras functions. We initialize the size of the encoded features. A fully connected neural network is built for encoder and decoder where the 'encoded' layer is the encoded representation of the input and the 'decoded' layer is the lossy reconstruction of the input.
   b) After building the model, we configure it using the binary cross-entropy loss and the adadelta optimizer.
   c) We then train the model using autoencoder.fit() with x=y=x_train i.e. the labels for data in this case are the input data points itself. This is because we want the autoencoders to learn the reconstruction of its input. We can see the learned encoded features for numerical data in the fig. 9.

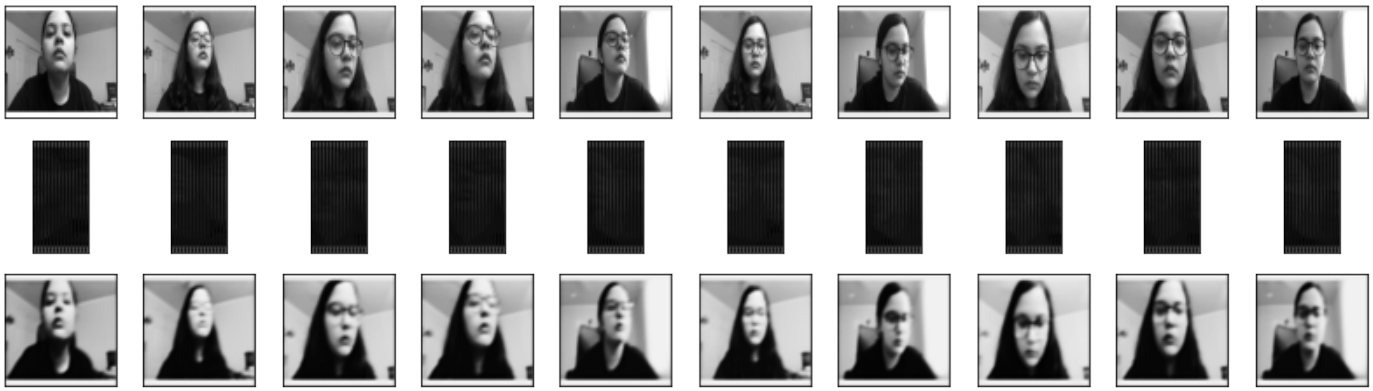

Fig. 8. Top to bottom: The resized input images, Encoded features, Reconstructed input

Fig. 9. numerical features

2) Convolutional Autoencoder with image data
Convolutional autoencoders have a same basic structure like a convolutional neural network with same basic elements like convolutional filters and pooling layers.
A deep convolutional autoencoder was trained on a dataset containing 144 images which are screenshots from the real-time facial landmark detection system. These screenshots have the person's face looking in 9 different directions on the screen, as well as the 68 points describing the facial landmarks marked on the face. The screenshots were taken in diverse settings such as glasses/no-glasses of the person, hair open/hair tied back and also changes in the distance between the webcam and the person. These diverse settings were made to ensure that the algorithm learns only from the direction of the face and not the features of the face itself. We can see the encoded representation of the input and the reconstructed input in the fig. 8.

V. Conclusion

An application was written to create a 3-by-3 grid, and collect head pose data using OpenCV. Here facial landmark data was used with several classifiers to predict where a subject was looking on a 3-by-3 grid, giving good results.

We compared three different classifiers: 10-NN, a Decision Tree, and a Random Forest. Here the features used were the explicit ratio of the nose to the four perimeter points. These features were the left-right ratio and the top-bottom ratio of the distances between facial landmark points. Using these ratios as features, the distance between the webcam and the face was normalized, compensating for optical magnification. For this task, the Random Forests resulted in a 100% accuracy for the test subject.

To push the method to discover the limitations, the grid size was expanded to include a 5-by-5 grid. Expanding the trial to use a 5-by-5 grid caused degraded results because the size of the regions was smaller, and the centroids were too close for correct classification. The Random Forests were still the best, but gave only 75% accuracy which is insufficient if we want to accurately predict the direction in which the user looks.

Additionally, an artificial neural network was used in two experiments with autoencoders. Raw data was passed and automated features were used from the bottleneck layer of autoencoder instead of giving explicit feature data. Using both numerical and image data revealed that the autoencoders were able to extract latent vectors, but the results were not human-readable, and the classification results did not improve.

Future work should expand this approach to include many more subjects and seating configurations. The proposed set of features worked very well with the 3-by-3 grid, but failed for the 5-by-5 grid as the ratios were too close. Perhaps a higher resolution camera, or more controlled environment would help improve future attempts.

A key result from this project is how a simple algorithm with appropriate features beats a neural network when provided with inappropriate and uncleaned data. Because the neural networks are becoming extremely popular, an important thing to keep in mind is to properly prepare the data and think if we can use the basics before straightaway moving to neural networks.

Overall, an initial design of a head-controlled human-computer interface is successfully built.

Acknowledgment

I would like to thank my advisor, Dr. Thomas Kinsman, for guiding me by going above and beyond his necessary duties.

References

[1] D. R. Barman". "intelli-cursor: A webcam assisted multi-monitor intelligent cursor". 2020.
[2] T. K. blog. Building autoencoder in keras. "https://blog.keras.io/building-autoencoders-in-keras.html", 2016.

[3] T. Gupta, H. Verma, G. Verma, and L. Sahoo". "a portable & cost effective human computer interface device for disabled". *International Conference on Communication Systems and Network Technologies*, 2015.

[4] Intel. Assistive context-aware toolkit (acat). "https://01.org/acat/", 2015.

[5] S. Iqbal, B. A. Rizvi, P. P. M. Shanir, Y. U. Khan, and O. Farooq". "detecting p300 potential for speller bci". *International Conference on Communication and Signal Processing (ICCSP), Chennai*, pages 0295–0298, 2017.

[6] R. Kaushik, T. Arora, Sukanya, and R. Tripathi". "design of eyewriter for als patients througheyecan". *International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), Greater Noida (UP), India*, pages 991–995, 2018.

[7] A. Królak". "use of haar-like features in vision-based human-computer interaction systems". *Institute of Electronics Technical University of Łódź Łódź, Poland*, September 2012.

[8] A. Kulshreshth, A. Anand, and A. Lakanpal". "neuralink- an elon musk start-up achieve symbiosis with artificial intelligence". *International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India*, pages 105–109, 2019.

[9] E. of Stephen Hawking. Stephen hawking biography. "https://www.hawking.org.uk/biography".

[10] M. P. and B. A.". "eye tracking and eye-based human–computer interaction". *Advances in Physiological Computing*, page 39–65, March 2014.

[11] F. Rosado, I. Rumbo, H. M. F. Daza, and D. Mier". "morse code-based communication system focused on amyotrophic lateral sclerosis patients". *Symposium on Signal Processing, Images and Artificial Vision (STSIVA), Bucaramanga*, pages 1–6, 2016.

[12] A. Rosebrock. Real-time facial landmark detection with opencv, python, and dlib. "https://www.pyimagesearch.com/2017/04/17/real-time-facial-landmark-detection-opencv-python-dlib/", April 2017.

[13] B. Sabuj, M. J. Islam, and M. A. Rahaman". "human robot interaction using sensor based hand gestures for assisting disable people". *International Conference on Sustainable Technologies for Industry*, December 2019.

[14] J. J. Shih, D. J. Krusienski, and J. R. Wolpaw". "brain computer interfaces in medicine". *Mayo Clinic Proceedings*, pages 266–279, March 2012.