Data Generation

We were trying to build a hand tracking-based Rock Paper Scissors that would play as an opponent for this project. With the help of images classification and object processing methods, researchers are trying to find solutions to such issues by dealing with real-time hand motion detection. During a search on google for "the Rock-Paper-Scissors" game, it brings up plenty of web pages where you may play these games.

Playing versus an AI opponent or different human gamers is possible in several of these games. Afiniti.com is an example of either of such web pages. Distant gamers may now duel against one another in a virtual arena, for each hand sign a graphic is clicked by each participant once they have connected to play. Game results and the opponent's moves are shown to the gamers after each has picked their respective moves for the round. Getting icons to press makes it easy for people to be using the software inside an internet browser with little processing capacity.

As a consequence, this strategy necessitates a whole new manner of gameplay. Using a computer to game Rock Paper Scissors is a very different experience from enjoying the game in reality. We looked at object detection techniques since we wanted gamers to be able to employ their basic bodily motions in conjunction with our effort to classify images. Object detection differs from image classification in that it focuses on detecting distinct component regions together in a picture. Observing the player's wrists & detecting his motion is similar to how a person might compete. Therefore, among the most significant decisions taken throughout this research was indeed the selection of the detecting technique.

———————————————————————————————————————————————————————————————

It is common for current movement identification systems to compute the RGB representation of such hand motion beforehand and use pattern match to detect its motion. Computer vision-based recognition system was shown to have several restrictions upon the internet, which may be summarised as follows:

* Gesture-based systems typically just have a palm picture as an input, but a truly human interface situation presents an image with a great quantity of additional info in the background as well as the movements.
* To fulfil the demands of HCI, where certain motion might send multiple messages depending on how it is performed, current gesture detection and classification only provide classification systems and ignore the geolocation data of the movement.

This work, taking into account the aforementioned constraints, provides a quick and easy data collecting approach that is unrelated to complexion luminance while demonstrating high precision in low-light circumstances, hence increasing the variety of samples and resolving the information shortage.

———————————————————————————————————————————————————————————————

It was chosen to use OpenCV for pattern recognition due to the obvious aforementioned limits on the information and pictures accessible. We recorded hand gestures datasets comprising the user's palm movements all combining static and flickering photos. To prevent the system from mistakenly not recognising the palm, we include flickering classes so to ensure that perhaps the algorithm understands hand motions that aren't even any signals but merely just motion. Our method uses the three motions of Rock, Paper, Scissors to play. There are set no’s of images taken at a specific time interval for every move category. For each picture, the user's palm is rotated throughout all 3 axes and also sidewards in almost all the distinct viewpoints. The dataset collection process makes use of OpenCV as well as Matplotlib.

Mediapipe, which is Google's open-source tech, is used to detect palm. These include palm recognition, human supervision, facial recognition or item detection. It was presented in 2019 as a cutting-edge technological advancement. One must use Mediapipe, which grips the hands and impresses spots on every joint using its libraries. As an open-source programme, OpenCV uses its libraries for certain of its surveillance and image-processing functions.

Neural networks, such as CNN's, are particularly useful for image analysis. The following explanation will emphasize picture processing because this is the sole usage for this feature underneath the MediaPipe structure. For our model, TensorFlow, a deep learning architecture was incorporated, as with all the necessary modules including Keras & TensorFlow. We also utilise the whole matplotlib toolkit to display as well as build static, dynamic, and live visuals, while CV focuses on inputs that help us display but also evaluate the information.

———————————————————————————————————————————————————————————————

To build a dataset of real-time pictures at 30 fps, we use OpenCV and a computer's 720p camera for gathering 2,000 photos, which would be the ideal amount of training images. In total, we have 6000 photos in our database, divided into three categories, each with a total of 2000 pictures. Detecting pictures in complicated backgrounds seems challenging again for the algorithm, thus in order to prevent the framework from incorrectly detecting the gestures, we feed on blurry photos. This loop continues until the user pushes a key to quit or until we attain the desired number of photos for a category. Afterwards, each group is separated by a time interval of 50 seconds.

———————————————————————————————————————————————————————————————

**OBJECT TRACKING, CLASSIFICATION, DETECTION**

Palms gestures are detected using a combination of Object detection, tracking and classification techniques. In object detection, "articles" are clusters of pixels that have been grouped. We want to recognise those pixel groups as things that move not just in the x or y-axis, but then in time as well. Motion detection is very effective when it relies on the temporal link among frames of a video. The procedure of identifying the palms inside the pictures is made easier by identifying the locations within every subsequent frame.

The method whereby the identified items inside the picture are categorised as to what the element is known as "object classification. It boils down to determining whatever the object is. Various factors, such as form, movement, colouration, and surface, could be used to identify objects. An entity's movement, stance, pace, and orientation may all be determined by monitoring it over time, as can its relationship to other objects in the scene. Frame-by-frame monitoring of a recognised item in actual time is a huge and challenging subject, for monitoring to be reliable, it is essential to know and comprehend the movement and change of the item in question.

Consequently, we've put photos into a system that can handle temporal features in our method. We may use our algorithm, which is built on a wide range of palm movements in actual time, to follow the item at a pace of 30 fps instead of concentrating on it throughout the test phase. Using Haar Cascade and Mediapipe, we can place our palms at any level giving optimum perception while enhancing accuracy.

———————————————————————————————————————————————————————————————

**HAAR CASCADE**

There are various favourable and unfavourable interpretations utilised in the construction of a cascading function in this computer-based method of learning. Hair cascading characteristics are used to calculate the concepts of integral image computation. Cascades classifications, as opposed to neural nets, function with a fixed collection of Haar features.

Images may be detected with this technique irrespective of geographic position or size. Real-time object detection is also feasible because of the individual's ability to run simultaneously with video feeds. The system can identify if a certain structure appears inside an image provided a vast quantity of training examples. As part of the feature extraction method, the Haar cascade would be used to detect objects in subsequent images. OpenCV, which had been obtained through cv2, and library matplotlib are required to run this application.

———————————————————————————————————————————————————————————————

**MEDIAPIPE**

Hand motion detection tools and libraries have indeed been developed to simplify things for anybody to construct Intelligence apps, which includes MediaPipe. It is a multimodal framework, which means it applies to a variety of mediums, including sound/video. With MediaPipe, a programmer can fixate upon that method and design features of the proposed, and afterwards assist the software with results that can be reproduced throughout various devices, that also have a few benefits with utilising features just on MediaPipe framework.

To deploy output-ready deep learning, the MediaPipe framework was created. It contains released code that accompanies studies and is used to create technological demonstrations. Information manipulations, multimedia processing framework and functional reasoning models also make up the graph modular components of MediaPipe. PyTorch, Tensor-Flow, PyTorch, OpenCV, etc all employ a graph of processes as an ML tool.

Two methods are operating together in the Mediapipe's hand gesture recognition optimiser, one is Hand Landmark Model and the other is Palm Detection Model. The Palm Detection Method delivers an appropriately clipped palms picture, which is then handed onto the Landmarks System for further processing. There is a lot less usage of data augmentation (such as flips, inverting, resizing) in the ML algorithms because of this procedure. Palm detection and landmark placement are traditionally done by detecting the hands and overlaying that on the consecutive frames. A new method is used in this Hand Analyser to deal with the difficulties of ML pipelines. Recognizing palms is a time-consuming technique that requires picture analysis and thresholding, as well as working across a wide range of palm lengths. There is an easier way of recognising palms besides simply identifying them out of a bounding box: the Palms detectors are learned, which predicts coordinates surrounding immovable items such as the palms and knuckles. This is followed by the employment of a decoder-encoder, which extracts a larger scene background. Hands Feature modelling comes into the equation as soon as hand recognition has been applied to the entire captured image.

3-dimensional hand-knuckle coordinates are used to accurately locate 21 important spots inside the identified hands areas by regression, which generates the coordinated projection straight from the palm highlight framework in MediaPipe. There are three values for every hand-knuckle marker: two for the picture widths and heights (x, y), and the other one for the hand's knuckle's depth (z). Value decreases when the hand gets nearer to cam. Drawing.utils is used to display the points over the hands and palms, and holistics is used to combine them. The holistic category contains five criteria, all of which have been outlined below:

**Static Image Mode:**This method, which defaults to False, handles the incoming images as though they were part of a video stream. If it detects palms in the initial frames it receives, it will attempt to locate the landmarks.

**Upper Body Only:**Holistic modelling includes facial recognition, eye, posing, and palms among others. For action identification, the parameter is disabled by the standard. When it comes to a holistic approach, hands and posture modelling get a lot in common with each other.

**Smooth Landmarks:**The lag is reduced by setting this value to true which is the default.

**Min Detection Confidence:**There is a range of [0.0 to 1.0]. The standard value is 0.5, which results in the rapid detection of a certain element. In our research, we don't want to identify any randomised item with a probability level lower than 0.5.

**Min tracking confidence:**By default, the value is set to 0.5.

We employ the Right-Hand Features to classify our scenario since the systems approach incorporates palm connections. For example, point 5 represents the beginning of the index finger (Index Finger MCP), and it is linked to point 8 represents the Index Finger TIP, point 9 represents the Middle Finger MCP, having the points 6 and 7 are also linked. In just about every other location on the picture, it displays exactly the same. Since all of the recognition is dependent on dots, there is no need to worry over colouration or reflecting characteristics in our approach. This method recognises palms in photos with only partly visible dots too, due to its outstanding range and extensive hand-detection.

**Preprocessing of Images**

The images are initially captured and evaluated utilizing OpenCV, and afterwards, the analysed outputs are passed onward for further process. After capturing a picture within the window, we use OpenCV to identify the specific palm gestures from the different data kinds by detecting the important areas. The highlighted dots represent these crucial spots, and when meshes are found, we must link them together. As a result, the mediapipe library contains a utility feature called drawing utils, which is used to join the points upon the hands.

In order to proceed, we need two frames, one for obtaining pictures, and the other for plotting the results of those pictures. A black backdrop is required for the second frame. OpenCV utilises a BGR video channel, whereas Matplot, as well as MediaPipe, are using the RGB video channel, thus the colour channel is changed. Since OpenCV is just a one-shot method, no more computations are required once the coloured stream has been converted. Additional data is processed and thus the output is maintained. For the second frame's markers to be plotted, a series of computations are required.

We ought to create a border all around our hand so that we can trim it and save it in a certain directory. Other approaches use a fixed-size rectangular frame to locate the hand's region, which is not the case here. Gaming motion capture doesn't always need accurate placement of the player's hands since it implies an unreasonable increase in the player's effort. Thus, we can see that in earlier research, the precision of a camera's entire visual field has not been addressed.

In this case, if the frame we are receiving is shorter than the one we previously surrounded the hand with, we would substitute it. Finally, we get a frame that's nearest to the points on the hand. This frame is then saved in the directory. Once the rectangular border is plotted around the hand, and when the movements were recognised with the pictures being scaled to the same dimensions, we pass this information to the CNN model for later processing. Flipping the image is done using the cv.flip() function with the image as well as the direction it is inverted, horizontally as well as vertically.

We experimented with a variety of resolution ranges while capturing these images. We started with 28x28 pixels, which was too little, and it detected anything that it wasn't meant to identify. Images couldn't be seen there either. Taking 128\*128 pixels, the computing effort was far more since the image was larger than required.  In the end, we landed on 96x96 pixels since it was the most comprehensive form we could come up with.

The images on the black screen were then stored in a specified directory in our training phase program. All inputs used are set to release, as well as all frames that are deleted when the images were collected and saved.