**Abstract**

Sign language is a sequence of hand gestures that generate a meaningful sentence. Gestures are performed by deaf and dumb community to perform sign language .Here for gesture recognition we are using image processing and computer vision. Gesture recognition enables computer to understand human actions and also acts as an interpreter between computer and human.

**Introduction**

The only mode of offline communication for hearing-impaired persons is through sign language. Learning the sign language is convoluted and has many conventions. The aim of this project is to develop an American Sign Language translator in order to mitigate the aforementioned difficulties. Another aim of this project is to develop a hand gesture recognition system to establish human computer interaction.

Sign Languages are a set of languages that use predefined actions and movements to convey a message. These languages are primarily developed to aid deaf and other verbally challenged people. They use a simultaneous and precise combination of movement of hands, orientation of hands, hand shapes etc. Different regions have different sign languages like American Sign Language, Indian Sign Language etc. We focus on Indian Sign language in this project.

Indian Sign Language (ISL) is a sign language that is predominantly used in South Asian countries. It is sometimes referred to as Indo-Pakistani Sign Language (IPSL). There are many special features present in ISL that distinguish it from other Sign Languages. Features like Number Signs, Family Relationship, use of space etc. are crucial features of ISL. Also, ISL does not have any temporal inflection.

In this project, we aim towards analyzing and recognizing various alphabets from a database of sign images. Database consists of various images with each image clicked in different light condition with different hand orientation. With such a divergent data set, we are able to train our system to good levels and thus obtain good results. We investigate different machine learning techniques like Support Vector Machines (SVM) and a neural network technique Convolution Neural Networks (CNN) for detection of sign language.

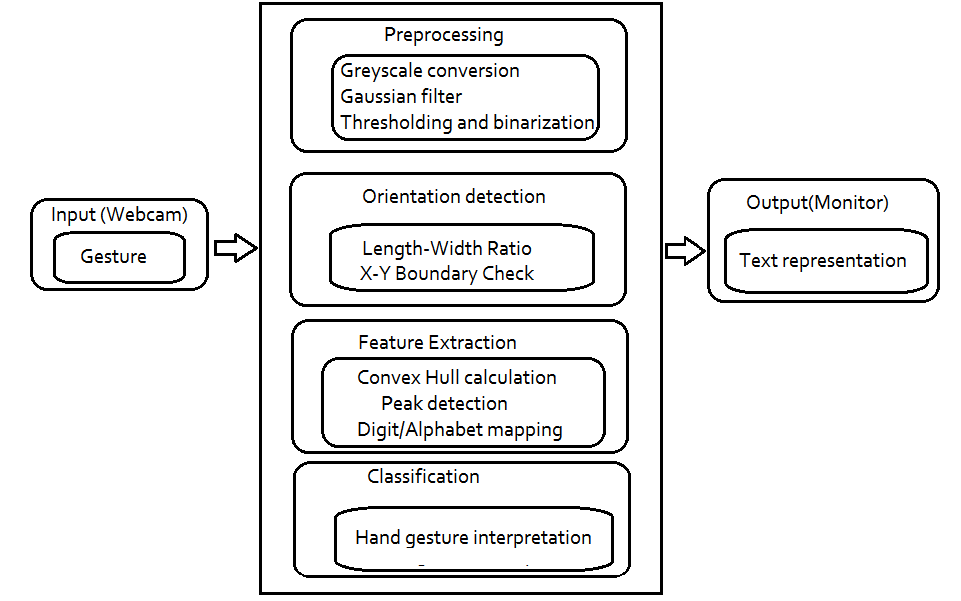
**Our Goal**

Our Goal Construct words for detecting American Sign Language(ASL) Fingerspelling gestures from a video input/Live Webcam.

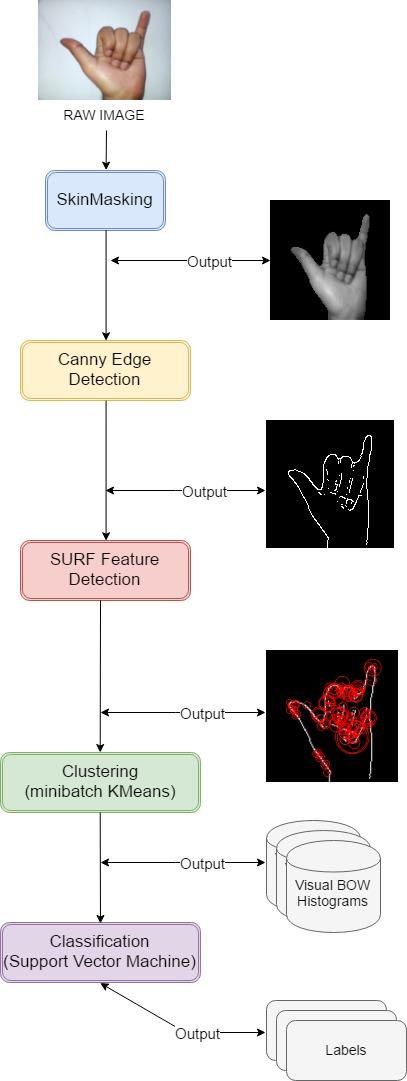
**Expected Outcome of the project :**

The deaf people and people who cannot speak can make use of this system, and communicate with the outside world with the help of sign language. Our main motivation in this research is to bridge the gap between human computer interaction between physically disabled people. Conventionally, mute people needed to type in order to interact with machines but now they can use their sign language and this will make tasks very easy and simple for them. The desired sign language messages is converted into a text message, which is further displayed on the screen.

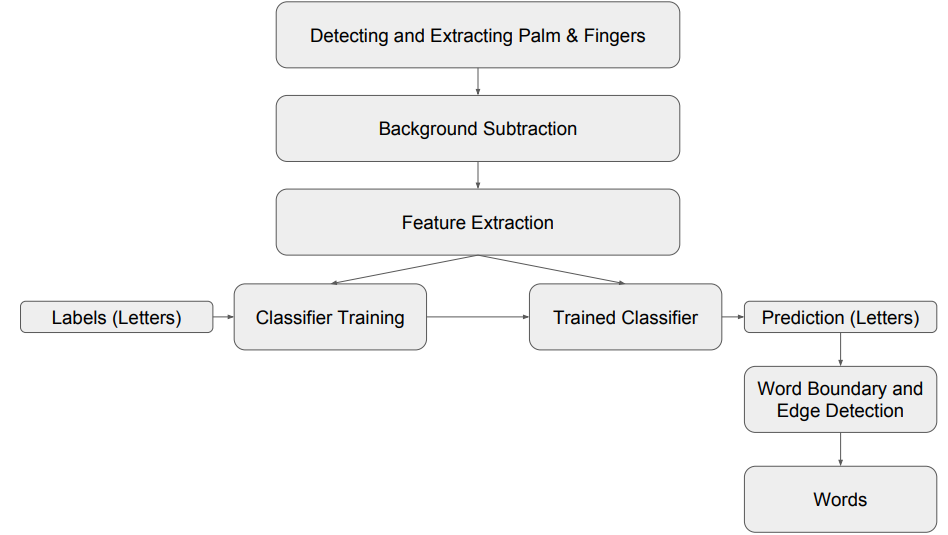
**Architecture Diagram**



**WorkFlow Diagram**



**FlowChart**



**Methdology**

Steps involved in the projects are

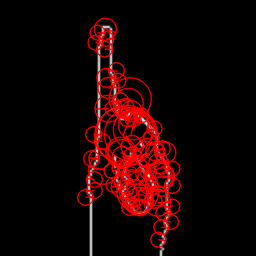
**1) Segmentation:**

The main objective of the segmentation phase is to remove the background and noises, leaving only the Region of Interest (ROI), which is the only useful information in the image. This is achieved via Skin Masking defining the threshold on RGB schema and then converting RGB colour space to grey scale image. Finally Canny Edge technique is employed to identify and detect the presence of sharp discontinuities in an image, thereby detecting the edges of the figure in focus.

    
BGR to HSV           Masked           Canny Edge

**2) Feature Extraction:**

The Speeded Up Robust Feature (SURF) technique is used to extract descriptors from the segmented hand gesture images. SURF is a novel feature extraction method which is robust against rotation, scaling, occlusion and variation in viewpoint.



**3) Classification**

The SURF descriptors extracted from each image are different in number with the same dimension (64). However, a multiclass SVM requires uniform dimensions of feature vector as its input. Bag of Features (BoF) is therefore implemented to represent the features in histogram of visual vocabulary rather than the features as proposed. The descriptors extracted are first quantized into 150 clusters using K-means clustering. Given a set of descriptors, where K-means clustering categorizes numbers of descriptors into K numbers of cluster center.

The clustered features then form the visual vocabulary where each feature corresponds to an individual sign language gesture. With the visual vocabulary, each image is represented by the frequency of occurrence of all clustered features. BoF represents each image as a histogram of features, in this case the histogram of 24 classes of sign languages gestures.

**4) Bag of Features model**

Following Steps are followed to achieve this:

* The descriptors extracted are first clustered into 150 clusters using K-Means clustering.
* K-means clustering technique categorizes m numbers of descriptors into x number of cluster centre.
* The clustered features form the basis for histogram i-e each image is represented by frequency of occurrence of all clustered features.
* BoF represents each image as a histogram of features, in our case the histogram of 24 classes of sign language is generated.

**Classifiers**

After obtaining the baf of features model, we are set to predict results for new raw images to test our model. Following classifiers are used :

* Convolution Neural Network

**Library and Classification Model to be Used**

**Tensorflow Library**

The core open source library to help you develop and train ML models. Get started quickly by running Colab notebooks directly in your browser.

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

* Easy Model Building

Build and train ML models easily using intuitive high-level APIs like Keras with eager execution, which makes for immediate model iteration and easy debugging.

* Robust ML production anywhere

Easily train and deploy models in the cloud, on-prem, in the browser, or on-device no matter what language you use.

* Powerful experimentation for research

A simple and flexible architecture to take new ideas from concept to code, to state-of-the-art models, and to publication faster.

**MediaPipe Hands Pose Model**

The ability to perceive the shape and motion of hands can be a vital component in improving the user experience across a variety of technological domains and platforms. For example, it can form the basis for sign language understanding and hand gesture control, and can also enable the overlay of digital content and information on top of the physical world in augmented reality. While coming naturally to people, robust real-time hand perception is a decidedly challenging computer vision task, as hands often occlude themselves or each other (e.g. finger/palm occlusions and hand shakes) and lack high contrast patterns.

MediaPipe Hands is a high-fidelity hand and finger tracking solution. It employs machine learning (ML) to infer 21 3D landmarks of a hand from just a single frame. Whereas current state-of-the-art approaches rely primarily on powerful desktop environments for inference, our method achieves real-time performance on a mobile phone, and even scales to multiple hands. We hope that providing this hand perception functionality to the wider research and development community will result in an emergence of creative use cases, stimulating new applications and new research avenues.

**ML Pipeline**

MediaPipe Hands utilizes an ML pipeline consisting of multiple models working together: A palm detection model that operates on the full image and returns an oriented hand bounding box. A hand landmark model that operates on the cropped image region defined by the palm detector and returns high-fidelity 3D hand keypoints. This strategy is similar to that employed in our MediaPipe Face Mesh solution, which uses a face detector together with a face landmark model.

Providing the accurately cropped hand image to the hand landmark model drastically reduces the need for data augmentation (e.g. rotations, translation and scale) and instead allows the network to dedicate most of its capacity towards coordinate prediction accuracy. In addition, in our pipeline the crops can also be generated based on the hand landmarks identified in the previous frame, and only when the landmark model could no longer identify hand presence is palm detection invoked to relocalize the hand.

The pipeline is implemented as a MediaPipe graph that uses a hand landmark tracking subgraph from the hand landmark module, and renders using a dedicated hand renderer subgraph. The hand landmark tracking subgraph internally uses a hand landmark subgraph from the same module and a palm detection subgraph from the palm detection module.

Note: To visualize a graph, copy the graph and paste it into MediaPipe Visualizer. For more information on how to visualize its associated subgraphs, please see visualizer documentation.

**Models**

**Palm Detection Model**

To detect initial hand locations, we designed a single-shot detector model optimized for mobile real-time uses in a manner similar to the face detection model in MediaPipe Face Mesh. Detecting hands is a decidedly complex task: our model has to work across a variety of hand sizes with a large scale span (~20x) relative to the image frame and be able to detect occluded and self-occluded hands. Whereas faces have high contrast patterns, e.g., in the eye and mouth region, the lack of such features in hands makes it comparatively difficult to detect them reliably from their visual features alone. Instead, providing additional context, like arm, body, or person features, aids accurate hand localization.

Our method addresses the above challenges using different strategies. First, we train a palm detector instead of a hand detector, since estimating bounding boxes of rigid objects like palms and fists is significantly simpler than detecting hands with articulated fingers. In addition, as palms are smaller objects, the non-maximum suppression algorithm works well even for two-hand self-occlusion cases, like handshakes. Moreover, palms can be modelled using square bounding boxes (anchors in ML terminology) ignoring other aspect ratios, and therefore reducing the number of anchors by a factor of 3-5. Second, an encoder-decoder feature extractor is used for bigger scene context awareness even for small objects (similar to the RetinaNet approach). Lastly, we minimize the focal loss during training to support a large amount of anchors resulting from the high scale variance.

With the above techniques, we achieve an average precision of 95.7% in palm detection. Using a regular cross entropy loss and no decoder gives a baseline of just 86.22%.

**Hand Landmark Model**

After the palm detection over the whole image our subsequent hand landmark model performs precise keypoint localization of 21 3D hand-knuckle coordinates inside the detected hand regions via regression, that is direct coordinate prediction. The model learns a consistent internal hand pose representation and is robust even to partially visible hands and self-occlusions.

To obtain ground truth data, we have manually annotated ~30K real-world images with 21 3D coordinates, as shown below (we take Z-value from image depth map, if it exists per corresponding coordinate). To better cover the possible hand poses and provide additional supervision on the nature of hand geometry, we also render a high-quality synthetic hand model over various backgrounds and map it to the corresponding 3D coordinates.

| **hand_landmarks.png** |
| --- |
|  |
| **The Overview of the Method**  The overview of the hand gesture recognition is described in Figure [1](https://www.hindawi.com/journals/tswj/2014/267872/fig1/). First, the hand is detected using the background subtraction method and the result of hand detection is transformed to a binary image. Then, the fingers and palm are segmented so as to facilitate the finger recognition. Moreover, the fingers are detected and recognized. Last, hand gestures are recognized using a simple rule classifier.  https://static-01.hindawi.com/articles/tswj/volume-2014/267872/figures/267872.fig.001.jpg |

**Finger pose Classifier**

Once the Hand Pose is detected we will use finger pose classifier for hand landmarks detected by TensorFlow.js' hand pose model. It can detect hand gestures like "Victory” or "Thumbs Up" inside a webcam source picture. You can define additional hand gestures using gesture descriptions.

**How it works**

Gesture detection works in three steps:

1. Detect the hand landmarks inside the video picture
2. Estimating the direction and curl of each individual finger
3. Comparing the result to a set of gesture descriptions

Step (1) is performed by Tensor Flow’s "hand pose", Step (2) and (3) are handled by this library.

**Steps Involved in the Project:**

Step 1: User takes a picture of the hand to be tested either through the cell phone camera or from the Internet.

Step 2: The image is converted into gray scale and smoothed using a Gaussian kernel.

Step 3: Convert the gray scale image into a binary image. Set a threshold so that the pixels that are above a certain intensity are set to white and those below are set to black.

Step 4: Find contours, then remove noise and smooth the edges to smooth bigcontours and melt numerous small contours.

Step 5: The largest contour is selected as a target.

Step 6: The angles of inclination of the contours and also the location of the center of the contour with respect to the center of the image are obtained through the bounding box information around the contour.

Step 7: The hand contours inside the bounding boxes are extracted and rotated in such a way that the bounding boxes are made upright (inclination angle is 0) so that matching becomes easy.

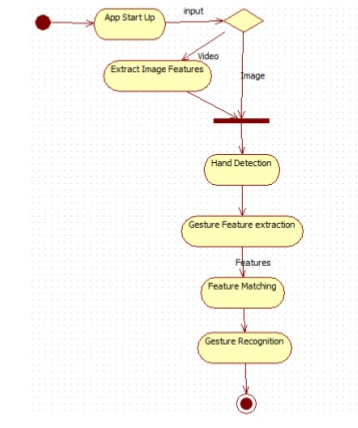
Step 8: Both the images are scaled so that their widths are set to the greater of the two widths and their heights are set to the greater of the two heights. This is done so that the images are the same size.

Step 9: The Trained model is loaded at the early stages of the project is instantiated.

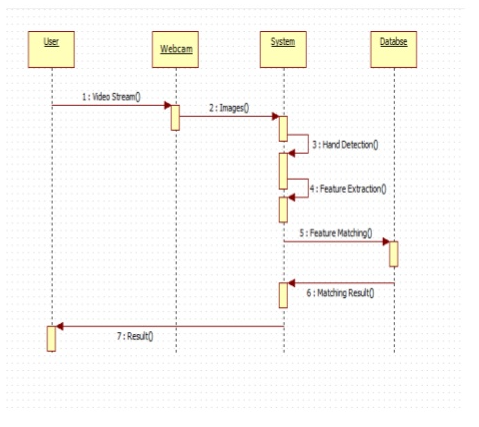
Step 10: The candidate image processed above is sent to the trained model for prediction.

Step 11: Predicted label is then matched to respected text and the text is displayed.

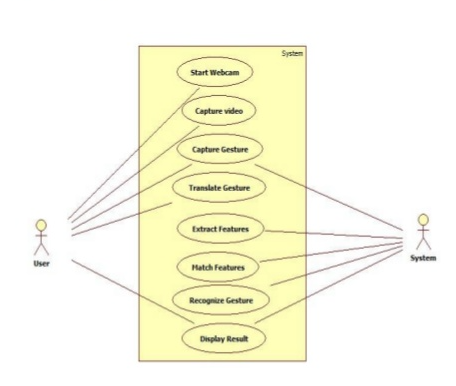
**ACTIVITY DIAGRAM**



**Sequence Diagram**



**Usecase Diagram**



**Conclusion**

In this project we have demonstrated an application of Hand-Gesture recognition in ASL fingerspelling by detecting 9 letters of the English alphabet. Future work will primarily comprise of expanding the dataset to include more letters.

**Bibliography**

[1] Md. Hasanuzzaman, T. Zhang, V. Ampornaramveth, H. Gotoda, Y. Shirai, and H. Ueno, “Adaptive visual gesture recognition for human-robot interaction using a knowledge-based software platform”, Robotics and Autonomous Systems, vol. 55, Issue 8, 31 August 2007, pp. 643-657. http://dx.doi.org.ezproxy.massey.ac.nz/10.1016/j.robot.2007.03.002

[2] R.C. Gonzales and R.E. Woods, Digital Image Processing, Prentice Hall, 2008. [3] M. Sonka, V. Hlavac, and R. Boyle, Image Processing, Analysis, and Machine Vision, PWS Publishing, 1999.

[4] R. Szeliski, Computer Vision: Algorithms and Applications, Springer, 2011.

[5] S.E. Umbaugh, Digital Image Processing and Analysis, CRC Press, 2011.

[6] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 2010.

[7] OpenCv, http://www.willowgarage.com/pages/software/opencv, accessed 11 May 2011.

[8] A.L.C. Barczak, Computer Vision: Study Guide, IIMS, Massey University, 2010.

[9] G. Bradsky and A. Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, O'Reilly, 2008.

[10] J.R. Parker, Algorithms for Image Processing and Computer Vision, Wiley Publishing, 2011

[11] R. Mukundan and K.R. Ramakrishnan, Moment Functions in Image Analysis, World Scientific, 1998.

[12] M.K. Hu, “Visual pattern recognition by moment invariants,” IRE Transactions on Information Theory, vol. 8, pp. 179-187, 1962.

[13] Y. Freund and R.E. Schapire, “A short introduction to boosting,” Journal of Japanese Society Real-Time Hand Gesture Detection and Recognition Using Simple Heuristic Rules Page 51 of 57 for Artificial Intelligence, vol. 14, no. 5, pp. 771-780, 1999.

[14] “IIMS Robotics Lab: Omni-Robot Manual and References,” IIMS, Massey University, 2010.

[15] S.K. Mathanker, P.R. Weckler, T.J. Bowser, N. Wang and N.O. Maness, “AdaBoost classifiers for pecan defect classification,” Computers and Electronics in Agriculture, vol. 77, pp. 60-68, 2011.

[16] Chieh-Chih Wang and Ko-Chih Wang, “Hand posture recognition using Adaboost with SIFT for human robot interaction,” Recent Progress in Robotics: Viable Robotic Service to Human, Lecture Notes in Control and Information Sciences, vol. 370, pp. 317-329, 2008.