Intelligent Human Sign Language Translation using Support Vector Machines Classifier

[1] P Krishna Prasad, [2] Akhil P Shibu

[1]Software Engineer, E2E Transport, Nokia Networks, Bangalore, India
[2]Data Analyst, Assurance, EY GDS, Kochi, India

Abstract: Sign is a visual language that uses hand shapes, facial expression, gestures and body language. For many with a profound to severe hearing loss, sign language provides vital access to language and communication. Although the use of sign language is common among personnel with hearing impediments, rest of the society is generally unaware of this. This creates a gulf in communication between them. In this work, we propose a novel methodology that interplays concepts of computer vision and machine learning to create an application that could accurately predict the text equivalent of sign language inputs. We experiment the use of Support Vector Machines(SVM), a supervised machine learning algorithm in classifying sign patterns. We employed Microsoft XBOX 360 Kinect Camera Sensor for capturing input sign language in high resolution, and, increased pixel depths. Providing a set of training examples, each labelled with textual value of the sign, we evaluate how the SVM training algorithm builds a model that classifies new untrained examples. We analyze the efficacy of pattern recognition algorithm, SIFT (Scalar Invariant Feature Transform) and evaluate classification accuracy of SVM.

IndexTerms **- Support Vector Machines, Pattern Recognition, Computer Vision, Machine Learning, Artificial Intelligence, SIFT**

1. INTRODUCTION

Sign language is used by individuals with hearing loss, as the main means for communication [1]. Communication between individuals possessing hearing disabilities and the ones without hearing losses, is based on the latter learning the sign language or alternatively, a third person who understands both oral and sign languages, who could do the necessary translation from one language to the other. A third alternative, is to use a device that act as a translator. Ideally, the device should sense a user performing sign language and then translate it to speech/text. Such a system has been developed here, focusing on the first translation direction: from signs to speech. An automatic sign language translator captures the sign that is performed in front of a camera and translates it into word/speech to be understood by another user. In this project, a Microsoft Kinect XBOX 360TM is used for high quality image capture. A sign input by a user is recorded by its sensors and after processing the raw image, the artificially intelligent translator; powered by SVM, will provide the corresponding word/letter in spoken language as output.

Fig -1: Human Sign Language Translation System

2. PROPOSED SYSTEM

The proposed human sign language translation system involves the following modules:

- Sign Acquisition
- Feature Extraction
- Machine Learning
- Classification

2.1 Sign Acquisition

Sign acquisition phase involves capturing the input sign using Microsoft Xbox 360 Kinect sensor. Kinect consists of Color VGA video camera and a Depth sensor. This video camera aids in facial recognition and detection of hand/finger gestures by detecting three color components: red, green and blue. An infrared projector and a monochrome CMOS (complementary metal-

oxide semiconductor) sensor work together to "see" the room in 3-D regardless of the lighting conditions. The video and depth sensor cameras have a 640 x 480-pixel resolution and run at 30 FPS (frames per second). The Kinect can also stream the view from its IR camera directly (i.e.: before it has been converted into a depth map) as 640x480 video, or 1280x1024 at a lower frame rate. The Kinect sensor has a practical ranging limit of 1.2–3.5 m (3.9–11.5 ft) [2].
For this research, Kinect is connected to a computer that has Kinect adapter/drivers installed. Openkinect's libfreenect; a user-space driver for Microsoft Kinect, allows calibration of RGB and Depth Sensors, Motors, Accelerometer, LED and Audio. The python-based application running in a UNIX environment uses python wrappers provided by libfreenect, to turn on Kinect, control accelerometer, position image sensors and to capture video frames in desired format.
In this research, finger spellings from American Sign Language(ASL) are used. Application calibrates Kinect to obtain frames of RGB images containing signs input by users. Each of these images are then sent to the feature extraction module of the application, which does image processing.

2.2 Feature Extraction

RGB frames, pertaining to a sign input via Kinect is processed by feature extraction module of the application. The purpose of feature extraction module is to identify characteristics of an RGB image, which can be further used to classify the sign. This module of the application implements the logic which processes the image and creates a data matrix which would be unique for each sign of ASL. This data created by the module is later used for training as well as classification. The same sign could be performed by users with different skin tone, in a different lighting and at different distances from image sensor. This demands the image processing algorithm used should be accurate, stable, scalar & rotational invariant. Apart from these qualities, the ability to work with smaller data sets make Scalar Invariant Feature Transform(SIFT) algorithm a good choice. For any object in an image, SIFT extracts 'interesting points' from the object to provide a "feature description" of the object. This description, extracted from a training image, can then be used to identify the object when attempting to locate the object in a test image containing many other objects [3]. SIFT can robustly identify objects even among clutter and under partial occlusion, because the SIFT feature descriptor is invariant to uniform scaling, orientation, illumination changes, and partially invariant to affine distortion [4].

The feature extraction module of application does the following steps:

- Cropping gesture part of the image from unwanted background
- Conversion of cropped RGB image to grayscale
- Perform gaussian blur and perfrom binary thresholding
- Application of SIFT algorithm to transform image into a large collection of feature vectors.

Since in our study, we only consider finger spellings, we only need that portion of the image wherein hand gestures are present. This is done by identifying presence of a hand in the image using convexity defects. A bounding rectangle is drawn around the convex hull. Then the image within the bounding rectangle is cropped out and used for further processing. The cropped image is now converted to grayscale to make sure classification is independent of skin tone and background.

Fig 2: Grayscale image of a sign

Fig-3: Gaussian blur and Thresholding

To reduce the noise and detail in the image, the grey-scaled image is then made to undergo a gaussian blur. We vary the threshold value until we get a clear binary (ie only black and white) image consisting only our ROI (Region of Interest).

SIFT algorithm is then applied to the resulting image to identify key-points. SIFT is implemented in our application using the wrapper API provided by OpenCV. SIFT algorithm generates a scale space and uses the scale space to calculate the Difference of Gaussians. This is further used to calculate Laplacian of Gaussian approximations, which is scalar invariant. Next it iterates through each pixel and marks a pixel as a "key point" if it is the greatest or least of all its 26 neighbours. To generate a unique descriptor for each of these key-points, SIFT creates a 16x16 window around the keypoint. This 16x16 window is broken into sixteen 4x4 windows. Within each 4x4 window, gradient magnitudes and orientations are calculated. These orientations are put into an 8-bin histogram and orientation in the range 0-44 degrees are added to the first bin. 45-89 add to the next bin and so on. The amount added to the bin depends on the magnitude of the gradient. Doing this for all 16 pixels, we would've "compiled" 16 totally random orientations into 8 predetermined bins. We do this for all sixteen 4x4 regions. So, we end up with 4x4x8 = 128 numbers. These 128 numbers form the "feature vector". This keypoint is uniquely identified by this feature vector. The output of this module are

keypoints and a numpy array of shape; Number_of_Keypoints×128. This way, SIFT algorithm transforms an image into a large collection of feature vectors.

Fig 4: Feature points extracted

Each sign from ASL would have a unique set of feature vectors. No two different signs would have a completely matching set of feature vectors. So, feature vectors are used to train SVM classifier. The key-points and descriptors from image are again used in the classification phase to predict the text corresponding to input sign.

2.3 Machine Learning

A straight forward feature vector to sign mapping does not guarantee an accurate classification of image. Although no two signs would have the same set of feature vectors; some signs may share the same feature points and descriptors. This demands a machine learning algorithm that could read patterns from training images and predict class of an input image based on the presence/absence of keypoints.

Support Vector Machines (SVM) is one of the best-known methods in image classification. It is designed to separate a set of training images into different classes, (x1, y1), (x2, y2), ..., (xn, yn) where xi is an element of d-dimensional feature space, and yi is an element of the set {-1,+1}, the class label, being i=1..n . SVM builds the optimal separating hyperplanes based on a kernel function (K). All images, for which feature vector lies on one side of the hyperplane belong to class -1 and the others belong to class+1 [5].

SVM machine learning module has two stages in its functioning: training and classification. During the training phase, we train the module with approximately 1000 images of each sign. The number of keypoints generated from each image varies greatly and lack of uniformity can affect the training phase negatively. To bring about uniformity, we use Vector Quantization or K-Mean Clustering and build a Bag-of-Words model histogram. The output of feature extraction module is a matrix X of the shape Number_of_Keypoints×128. During training, we horizontally concatenate all the feature vectors Xi (i <= n and n is the number of images in the dataset for all i) into a big matrix V. This matrix acts as input to K-means clustering with a chosen k. The output of K-means is a matrix C with size 128 * k. For our experiment a k value of 100 is used. For each image in the dataset, we create a histogram vector h of size k and initialize it to zeros. For each key point's vector, we find the index of its "best match" vector in the matrix C. The value stored in corresponding index of histogram vector is incremented by 1.

Many elements used in the objective function of learning algorithm (such as the RBF kernel of Support Vector Machines or the L1 and L2 regularizers of linear models) assume that all features are centered around 0 and have variance in the same order. If a feature has a variance that is orders of magnitude larger than others, it might dominate the objective function and make the estimator unable to learn from other features appropriately as expected. So, the histogram vector h is then standardized by removing the mean and scaling to unit variance. The resulting matrix and a numpy array denoting the class label of the data is then passed on for training SVM.

LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True, intercept_scaling=1, loss='squared_hinge', max_iter=1000,multi_class='ovr',penalty='l2',random_state=0, tol=0.0001,verbose=0)

Scikit Learn's SVM wrappers are used to fit the data into a linear support vector classifier having a linear kernel [6]. It decomposes the multi-class classification problems (k > 2) into a series of binary problems such that the standard SVM can be directly applied. Then it models each situation by creating a feature space, which is a finite-dimensional vector space, each dimension of which represents a "feature" of a sign. SVM constructs linear separating hyperplanes in high-dimensional vector spaces. Hyperplanes separate training data perfectly according to their class labels. Training dots on either side of hyperplane signify two different signs. The training model so created is saved as "pkl "file and loaded during the classification. The training phase was computing intensive and took nearly 60mins for training 1000 images of a sign.

2.4 Classification

During the training phase, system is trained with signs pertaining to 15 finger spellings. During the testing phase, untrained finger spellings and trained finger spellings in varying environments are used. The initial stages of classification phase are same as the learning phase. A sign is performed by user in front of Kinect. The feature extraction module provides the keypoint and descriptors of the image. A histogram of the features is then made and is then scaled. The training SVM model stored during the training phase is loaded using function "load()" of joblib python module. The histogram of features is then provided to "predict()" api of scikit learn package [6]. The output is then searched in the training model to obtain textual mappings of the class label. The textual representation of the sign is then printed on screen for the other user to read. Pyttsx package of python is used in providing voice output of the sign.

3. EXPERIMENT AND ANALYSIS

We use OpenCV library for image processing and sklearn.svm. LinearSVC module of Scikit Learn Library for developing Support Vector Machine (SVM) component; the Machine learning module of our application. The SVM component is trained with 1000 images of each of the following finger spelling signs from American Sign Language.

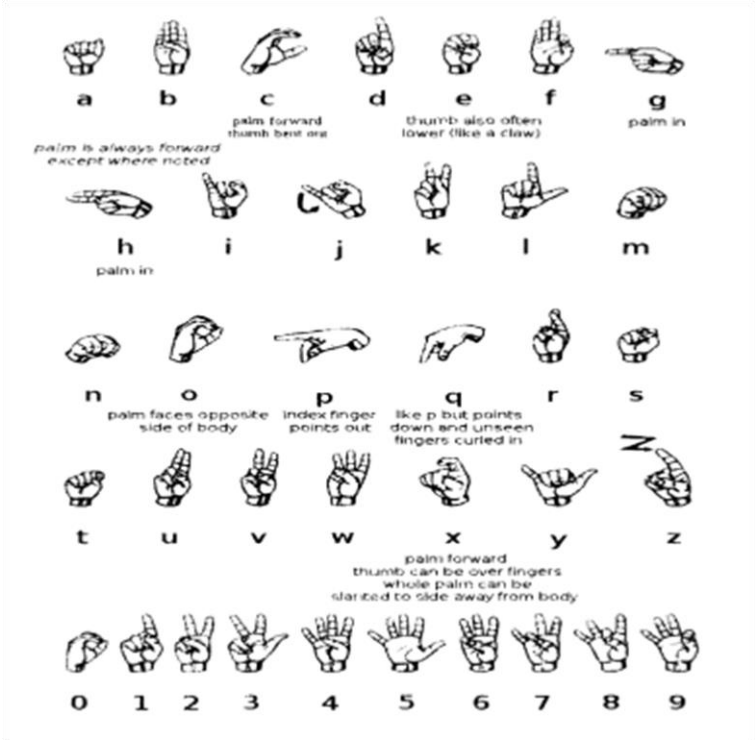

Fig -5: Signs used in training

The images used in training are of the size of 30-40kb, with very less noise and good illumination. Images like below ones are used in training the SVM model. Once the training is complete, we test the model for classification.

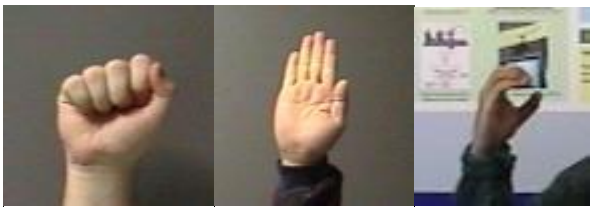

Fig -6: Training image of signs A, B and C (from left)

Contrary to well illuminated images in the dataset, we provide gesture inputs to the application in noisy and less illuminated surroundings. The application provides a UI for users in the classification phase which includes a live feedback from the image sensors. As the user performs a gesture, the part of the image containing the gesture/sign is highlighted in the feedback on the screen.

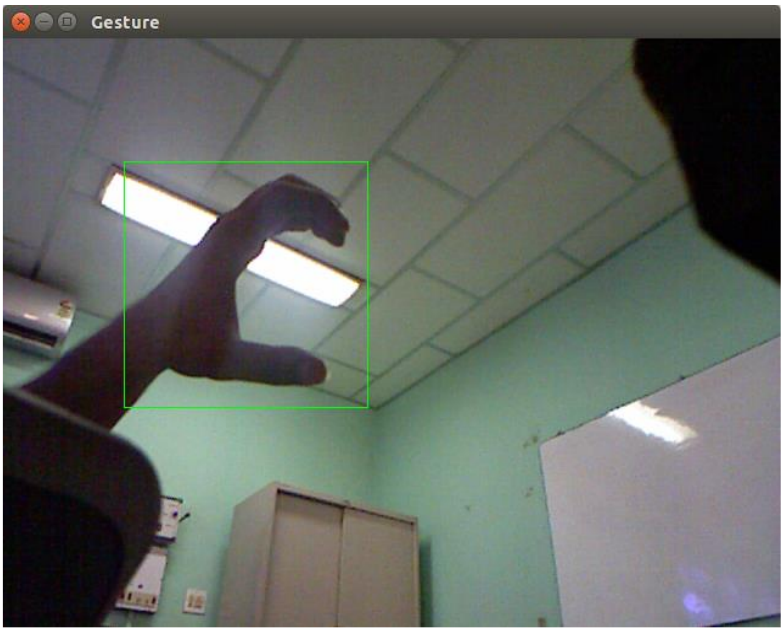

Fig -7: Input user sign and highlight of the gesture

In the above example user performs a gesture corresponding to the alphabet ‘C’. The gesture part of the image is highlighted, and background is subtracted. HSV threshold is applied to the image and displayed for debug purposes.

Fig -8: HSV-Threshold of the captured sign

Post processing, the histogram vector of the image is given to SVM model for classification. The output of the classification is displayed on screen and played as speech through speakers. The below screenshot is the UI that displays classification result. In the example, we could see that the noise and absence of illumination has rendered the thresholding less perfect. The number of keypoints obtained from the image would thus be greatly reduced and would not be as elaborate as the training image. Still the system was able to identify the sign to be ‘C’ based on the presence/absence of the few identified keypoints.

The test was similarly done for all letter signs. During the testing phase it was found classification of alphabets like ‘A’, ‘K’, ‘V’, ‘G’, ‘E’, ‘H’, ‘I’, ‘J’, ‘K’’ give a high prediction accuracy. But often the model is confused between ‘u’ and ‘d’, ‘z’ and ‘r’, ‘m’ and ‘’n’. For example, a sign that stands for ‘U’ is sometimes misjudged and classified to be ‘D’. Also, when the image possesses a background containing human body parts like hands, the image processing algorithm occasionally identifies the human body parts in background as a gesture.

Fig -9: Classification output

4. CONCLUSION

The accuracy of prediction varied depending on the sign performed and environment. Some of the signs like 'c' and 'u', 'd' and 'z, 'r' and 'n', 'm' have similar hand gestures. This would result in the signs having a lot more similar keypoints in its thresholded image. For those pairs of alphabets, the accuracy of prediction fell to 20-30 %. When only signs with dissimilar hand gestures are trained, accuracy of prediction went high up to 80%. Even when SVM is trained with all the alphabet signs, certain alphabets like 'A', 'K', 'V', 'G', 'E', 'H', 'I', 'J', 'K'' gave a prediction accuracy of about 80%. Although a cluttered environment with objects in the background did not greatly affect the accuracy of classification, an environment in which other humans and human body parts are present tend to affect the efficacy of image processing module. The confusion that SVM model faces in predicting signs which are minorly different from each-other arises from usage of thresholded image for training. The difference between two signs maybe as smalls one finger being kept over the other. These subtle differences get removed when thresholded. Using images without HSV thresholding can help improve the accuracy in this regard.

According to the results, an overall classification accuracy between 60 to 65 % was obtained by the system. SIFT, being the core of image processing module, proved itself to be a computing intensive algorithm but was efficient in finding out keypoints from images. An improvement that can be made in this regard is to research if unthresholded image can improve classification accuracy. Support Vector Machines model had a higher time complexity involved during the training phase. But it could accurately classify signs performed even in dim-lit and cluttered environments.

ACKNOWLEDGEMENTS

This research was supported by Government Model Engineering College, Kochi. We thank Dr Priya S, Professor, Government Model Engineering College, for her guidance and Uma Narayanan, Associate Professor, Government Model Engineering College for comments that greatly improved the manuscript.

REFERENCES

[1] NIDCD Fact Sheet | Hearing and Balance, American Sign Language, National Institute on Deafness and other Communication Disorders

[2] Luo Q. (2014) Study on Three Dimensions Body Reconstruction and Measurement by Using Kinect. In: Duffy V.G. (eds) Digital Human Modeling. Applications in Health, Safety, Ergonomics and Risk Management. DHM 2014. Lecture Notes in Computer Science, vol 8529. Springer, Cham Tuan D Pham, Object Recognition by Performance of Ratios Based Fusion and Gaussian Bayes Decision, Knowledge-Based and Intelligent Information and Engineering Systems: 13th International Conference, Chili, September 2009

[3] H. Sebai and A. Kourgli, "Improving high resolution satellite images," Image Analysis and Processing — ICIAP 2015: 18th International Conference, Genoa, Italy

[4] B. Karimi and A. Krzyzak, "Computer Aided System for Suspicious Lesions in Breast Ultrasound Images", Artificial Intelligence and Soft Computing: 13th International Conference, Zakopane, Poland, 2014

[5] Chih-Chung Chang and Chih-Jen Lin, LIBSVM: A Library for Support Vector Machines, National Taiwan University, Taipei, Taiwan

[6] Yujun Yang, Jianping Li, Yimei Yang, "The research of the fast SVM classifier method", 12th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 2015

[7] Angelos Tzotsos, Support Vector Machine Approach for Object Based Image Analysis, Laboratory of Remote Sensing, Department of Surveying, School of Rural and Surveying Engineering, National Technical University of Athens, Greece, 2014

[8] David G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", David G. Lowe, Computer Science Department University of British Columbia,2004

[9] Shuiwang Ji; Wei Xu; Ming Yang; Kai Yu, "3D Convolutional Neural Networks for Human Action Recognition", IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume: 35, Issue: 1, Jan. 2013)

[10] Pablo Arbelaez; Michael Maire; Charless Fowlkes; Jitendra Malik, "Contour Detection and Hierarchical Image Segmentation", IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume: 33, Issue: 5, May 2011)