INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

ISSN2454-9940

www.ijasem.org

Vol 18, Issue 2, 2024

AMERICAN SIGN LANGUAGE ALPHABET RECOGNITION

[1]MR.R.VENKATESH,[2]NAINENI SHINY,[3]CHILUVERI ASHWAJ,[4]J AJAY,[5]M ARUN KUMAR

[1]Assistant Professor, Department of CSE-AI&ML, Malla Reddy College of Engineering,secunderabad , Hyderabad

[2,3,4,5]UG Students,Department of CSE-AI&ML, Malla Reddy College of Engineering,secunderabad , Hyderabad

ABSTRACT

The "Real-Time American Sign Language Recognition" focuses on developing and evaluating a system for accurately identifying the letters and digits of the American Sign Language (ASL). This project addresses the critical need for enhanced communication tools for the deaf and hard-of-hearing community by leveraging advances in machine learning and computer vision. The proposed system employs OpenCV for real-time image capture, a convolutional neural network (CNN) for feature extraction, and a random forest classifier for recognizing ASL hand gestures. A comprehensive dataset of ASL alphabet images and digits (0-9) was collected, pre processed, and used to train the models. The model's performance was assessed through rigorous testing, achieving high accuracy rates in recognizing static hand gestures representing ASL letters. This study demonstrates the potential of integrating AI-driven recognition systems into real-world applications, such as educational tools, assistive technologies, and communication aids, thereby contributing to improved accessibility and inclusivity for individuals relying on ASL. Future work will explore expanding the system to dynamic gesture recognition and incorporating more complex aspects of ASL.

KEYWORDS: American Sign Language (ASL), Machine learning, Computer vision, Convolutional neural network (CNN).Model training, Accuracy rates, Static gestures, Dynamic gestures, Assistive technology, Accessibility, Inclusivity, and AI-driven recognition systems.

I.INTRODUCTION

American Sign Language (ASL) is a vital means of communication for the deaf and
hard-of-hearing communities in the United States and parts of Canada. Unlike spoken
languages, ASL relies on visual-gestural modality, utilizing a complex system of hand
shapes, movements, facial expressions, and body postures to convey meaning. Within
this system, the ASL alphabet, also known as the manual alphabet, plays a crucial role.

It consists of 26 distinct hand gestures, each representing a letter of the English alphabet and digits (0-9) enabling users to spell out words, names, and specialized terms that do not have designated signs. The manual alphabet is not only fundamental for learning ASL but also essential for facilitating nuanced communication, especially when specific or unfamiliar terms need to be conveyed. The development of an automated ASL alphabet recognition system represents a significant advancement in making ASL more accessible to both users and non-users.Such a system would allow for seamless interaction in various contexts, from educational environments where students can learn ASL more effectively, to public and private sectors where ASL users can communicate more freely with those who do not understand the language. Additionally, this technology can be integrated into mobile applications, wearable devices, and other assistive technologies, providing real-time translation and enhancing the inclusivity of ASL users in everyday activities.

II.LITERATURE REVIEW

1. Wadhawan S, Kumar R. A comprehensive survey on hand gesture recognition:
Challenges, methods and applications. Journal of Ambient Intelligence and Humanized Computing, 2020. Wadhawan and Kumar provided an extensive survey of hand gesture recognition techniques, covering traditional and deep learning-based methods. They discussed the unique challenges of gesture recognition, such as occlusion, lighting variations, and inter-user differences. This survey includes a section on sign language recognition, emphasizing the relevance of robust feature extraction and model

ISSN2454-9940

www.ijasem.org

Vol 18, Issue 2, 2024

generalization in ASL alphabet recognition.

➢ 2. Geetha M, Panwar M. Motion and orientation invariant American Sign Language recognition using deep learning framework. Expert Systems with Applications, 2019. Geetha and Panwar developed a deep learning framework for ASL recognition that is invariant to motion and orientation changes. Their work addressed the challenges of recognizing hand gestures from different angles and movements, improving the robustness of ASL alphabet recognition systems.

➢ 3. Ahsan MM, Siddique Z, Hossain MS, Das M. A robust system for real-time American Sign Language alphabet recognition using k-nearest neighbors.IEEE International Conference on Electro Information Technology (EIT), 2019. Ahsan et al. proposed a real-time ASL alphabet recognition system using the k-nearest neighbors (k-NN) algorithm. Their approach emphasized simplicity and robustness, achieving notable accuracy in real-time scenarios.

III.EXISTING SYSTEM

1.Data Collection: The first step in the project is to collect a dataset of images or videos of ASL fingerspelling gestures. The dataset should be diverse, with varying lighting conditions, hand shapes, and orientations. The dataset is collected using a webcam or camera, and each image or video is labeled with the corresponding alphabetical character.

2. Data Preprocessing: Once the dataset is collected, the next step is to preprocess the data. Preprocessing involves converting the images to RGB format, detecting hand landmarks, and storing the landmark coordinates in an array. The hand landmarks are detected using the Media Pipe Hand Landmark model, which provides 21 3D landmarks for each hand.

3. Feature Extraction: The preprocessed data is then used to extract features that can be used to train the machine learning model. The features extracted include the x and y coordinates of the hand landmarks, which are normalized by subtracting the minimum value of x and y coordinates.

Model Training: The extracted features are then used to train the machine learning model. The model can be a Random Forest classifier, a Support Vector Machine (SVM) classifier, or a Convolutional Neural Network (CNN) classifier. The model is trained using the training dataset, and the hyperparameters are tuned to achieve high accuracy.

4. Model Evaluation: Once the model is trained, it is evaluated using the testing dataset. The accuracy of the model is computed by comparing the predicted values with the actual values. The accuracy score is used to evaluate the performance of the model.

5. Model Testing: The trained model is then used to test new data points. The model takes an image or video of an ASL finger spelling gesture as input and predicts the
corresponding alphabetical character.

Disadvantages of Using SVM:

• **Computational Complexity**

SVMs can be computationally expensive, especially when dealing with large datasets.
The training process involves solving a quadratic programming problem, which can be time-consuming. This can be a significant disadvantage in real-time ASL recognition systems, where fast processing is crucial.

• **Overfitting**

SVMs can suffer from overfitting, especially when the dataset is small or noisy. Overfitting occurs when the model is too complex and fits the training data too closely, resulting in poor generalization to new, unseen data. In ASL recognition, overfitting can lead to poor performance on new signs or variations of signs.

• **Choice of Kernel**

SVMs rely on kernel functions to transform the input data into a higher-dimensional
space, where the data can be linearly separated. However, the choice of kernel function can significantly affect the performance of the SVM. In ASL recognition, the choice of kernel function can be critical, and a poor choice can lead to poor performance.

• **Sensitivity to Hyperparameters**

SVMs have several hyperparameters, such as the regularization parameter (C) and the
kernel parameter (γ), that need to be tuned for optimal performance. However, the
tuning process can be time-consuming and require significant expertise. In ASL

recognition, the hyperparameters may need to be adjusted for each sign or gesture,
which can be challenging.

• **Limited Interpretability**

13SVMs are often considered black-box models, meaning that it can be difficult to
interpret the results or understand why a particular classification decision was made. In ASL recognition, interpretability can be important, as it can help identify errors or
improve the system.

• **Not Suitable for Multiclass Problems**

SVMs are typically designed for binary classification problems, where there are only
two classes. In ASL recognition, there are 26 classes (one for each letter of the alphabet), which can make SVMs less suitable. While SVMs can be extended to
multiclass problems using techniques like one-vs-all or one-vs-one, these approaches
can be computationally expensive and may not perform as well as other multiclass
classification algorithms.

• **Not Robust to Noisy Data**

SVMs can be sensitive to noisy data, which can affect their performance. In ASL
recognition, noisy data can arise from various sources, such as variations in lighting,
camera angle, or hand shape.

• **Not Suitable for Real-Time Systems**

SVMs can be computationally expensive, which can make them less suitable for real
time ASL recognition systems. Real-time systems require fast processing and low
latency, which can be challenging to achieve with SVMs.

III.PROPOSED SYSTEM

Data Collection:

Data collection is the foundation of any machine learning project. For ASL alphabet
recognition, a diverse dataset of images or videos capturing hand gestures representing each letter of the alphabet (A-Z) and Digits (0-9) is essential.

Define Data Requirements:

• Specify the desired characteristics of the dataset, including variations in lighting conditions, hand shapes, orientations, and backgrounds.

• Ensure sufficient representation of each ASL alphabet letter to avoid class imbalance issues during training.

Source or Generate Data:

• Collect existing datasets available in the public domain, such as the ASL Alphabet dataset from Kaggle or academic repositories.

• Augment the dataset by capturing additional images or videos using cameras or depth sensors, ensuring a diverse range of hand gestures.

Data Annotation:

• Annotate each image or video frame with the corresponding ASL alphabet letter label.

• Maintain consistency and accuracy in annotations to facilitate model training and evaluation.

Data Quality Control:

• Perform quality checks to ensure the dataset meets the defined criteria.

• Eliminate low-quality images or videos that may introduce noise or bias into the training process.

IV.IMPLEMENTATION

Data Preprocessing

Data preprocessing involves transforming raw data into a format suitable for feature extraction and model training. In the context of ASL alphabet recognition, preprocessing steps aim to standardize image or video frames and enhance their quality.

Image or Video Resizing:

• Resize all images or video frames to a consistent size to ensure uniformity in the dataset.

• Choose an appropriate resolution that balances computational efficiency with preservation of detail.

Normalization:

• Normalize pixel values to a common scale (e.g., [0, 1]) to facilitate convergence during model training.

• Standardize the intensity levels across images or video frames to mitigate variations in lighting conditions.

Noise Reduction:

• Apply filters, such as Gaussian blur or median filtering, to reduce noise and smooth out irregularities in the images or video frames.

• Preserve essential features while suppressing irrelevant details that may interfere with recognition.

Contrast Enhancement:

• Adjust the contrast of images or video frames to enhance visibility and improve feature discriminability.

• Use techniques like histogram equalization or adaptive contrast stretching to amplify image details.

ISSN2454-9940
www.ijasem.org
Vol 18, Issue 2, 2024

Feature Extraction:

Feature extraction involves identifying and extracting relevant information from preprocessed images or video frames that can discriminate between different ASL alphabet letters. Various visual cues, such as hand shape, orientation, and movement

patterns, serve as discriminative features for recognition.

Hand Segmentation:

• Use techniques like color thresholding, skin tone detection, or background subtraction to isolate the hand region from the background in each image or video frame.

• Segmenting the hand facilitates focused analysis and extraction of handrelated features.

➢ **Feature Representation:**

• Extract descriptive features that capture key characteristics of the hand gestures,such as:

• Hand shape and contour: Represented by geometric properties like area, perimeter, and centroid.

• Finger positions and movements: Captured through finger detection or tracking algorithms.

• Skin tone and texture: Characterized by color histograms, texture descriptors CNN based feature maps.

• Choose features that are invariant to variations in hand orientation, scale, and lighting conditions to ensure robust recognition.

Feature Selection:

- Evaluate the relevance and discriminative power of extracted features using techniques like mutual information, correlation analysis, or feature importance scores.

- Select a subset of informative features that contribute most to distinguishing between different ASL alphabet letters.

➢ **Model Training:**

Dataset Splitting:

Splitting the dataset into training and validation sets is crucial to assess the performance of the trained model accurately. Here's a more detailed explanation:

Cross-Validation:

• Utilize techniques like k-fold cross-validation to ensure robustness in model evaluation.

• Divide the dataset into k subsets (folds), train the model on k-1 folds, and evaluate it on the remaining fold.

• Repeat this process k times, rotating the validation fold each time, and average the performance metrics across all folds to obtain a comprehensive evaluation.

ISSN2454-9940
www.ijasem.org
Vol 18, Issue 2, 2024

Random Forest Configuration:

Configuring the Random Forest classifier involves selecting appropriate hyperparameters to optimize its performance. Let's delve deeper into this process:

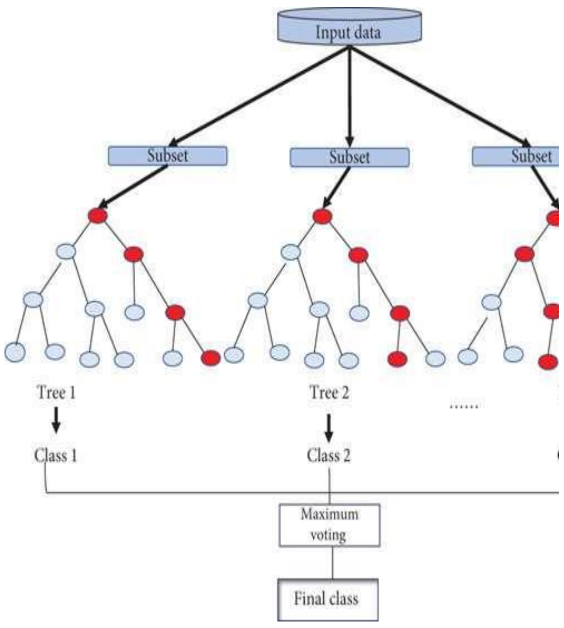


Fig : Random Forest Classification.

Hyperparameter Selection:

• Number of Trees (Estimators):

• Determine the number of decision trees in the forest, balancing computational cost with model complexity.

• Experiment with different values and evaluate their impact on classification accuracy.

Maximum Depth:

• Limit the depth of each decision tree to control overfitting and improve generalization.

• Choose an optimal depth by monitoring model performance on the validation set.

• Minimum Samples per Leaf:

• Specify the minimum number of samples required to split a node further.

• Adjust this parameter to prevent the model from creating nodes with too few samples, which may lead to overfitting.

Hyperparameter Tuning:

• Employ techniques like grid search or randomized search to systematically explore the hyperparameter space.

• Conduct a grid search over predefined ranges of hyperparameter values, evaluating each combination's performance using cross-validation.

• Select the hyperparameter configuration that yields the best validation performance for the final model.

➢ **Model Fitting:**

Training the Random Forest classifier involves fitting decision trees to the training data and aggregating their predictions.

Bootstrapped Sampling:

• Randomly sample subsets of the training data with replacement to create divers training sets for each decision tree.

• Bootstrap sampling introduces variability into the training process, enhancing model robustness and reducing overfitting.

Decision Tree Construction:

• Grow decision trees recursively by selecting the best feature to split on at each node.

• Use metrics like Gini impurity or entropy to evaluate split quality and determine feature importance.

Ensemble Learning:

• Combine predictions from multiple decision trees to make the final classification.

• In classification tasks, employ a majority voting scheme, where the class with the most votes across all trees is chosen as the predicted class.

➢ **Model Evaluation:**

After training the Random Forest classifier, it's essential to evaluate its performance on the validation set.

Performance Metrics:

• Calculate various performance metrics to assess the classifier's effectiveness, including accuracy, precision, recall, F1-score, and confusion matrix.

• Accuracy measures the proportion of correctly classified instances, while precision and recall quantify the classifier's ability to correctly identify positive instances and retrieve all relevant instances, respectively.

Architecture:

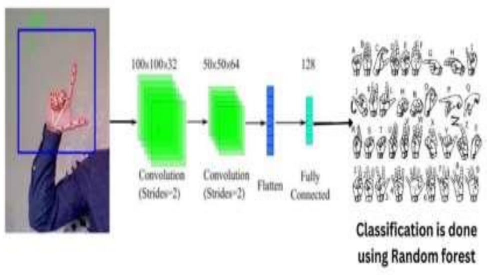


Fig. : Architecture of ASL Model

How Random Forest Works For Classification:

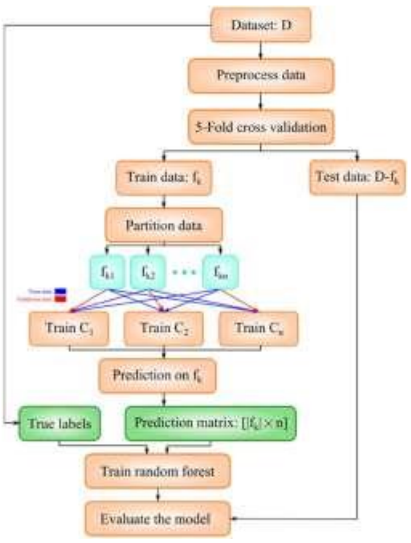


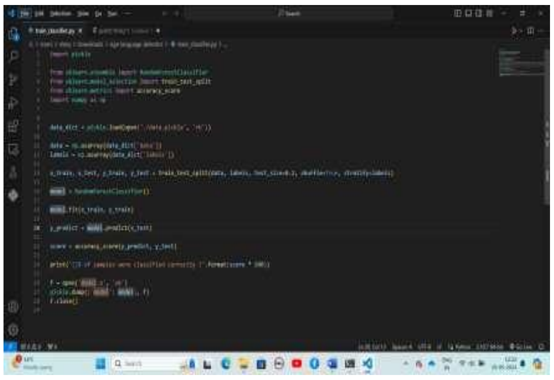
Fig : Recognition and Classification Using Random Forest

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

ISSN2454-9940

www.ijasem.org

Vol 18, Issue 2, 2024



```
import pickle

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import numpy as np


data_dict = pickle.load(open('./data.pickle', 'rb'))

data = np.asarray(data_dict['data'])
labels = np.asarray(data_dict['labels'])

x_train, x_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, shuffle=True, stratify=labels)

model = RandomForestClassifier()

model.fit(x_train, y_train)

y_predict = model.predict(x_test)

score = accuracy_score(y_predict, y_test)

print('{}% of samples were classified correctly !'.format(score * 100))

f = open('model.p', 'wb')
pickle.dump({'model': model}, f)
f.close()
```

Fig :Training the model.

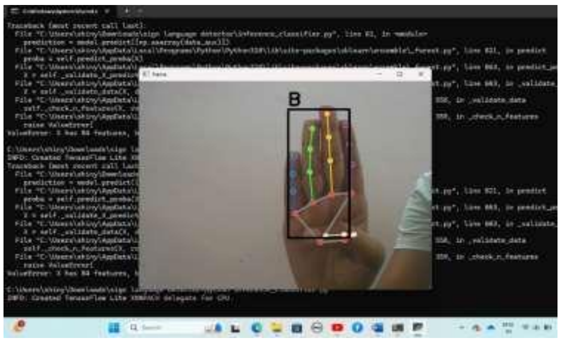


Fig : Real Time Alphabet ‘B’detection.

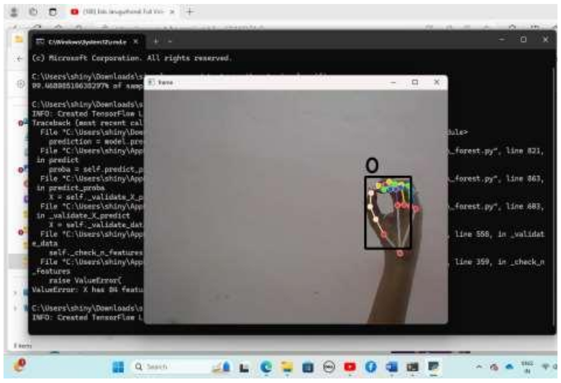


Fig : Alphabet ‘O’ Detection.

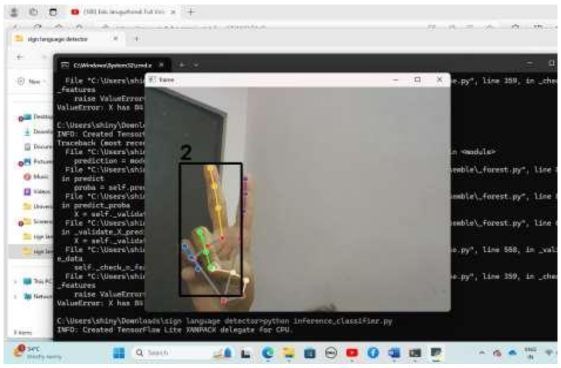


Figure :Real Time Digit Detection.

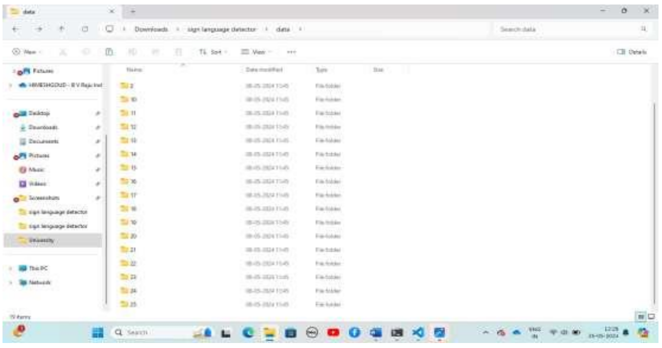


Figure :Data Used for Training the model.

V.CONCLUSION

This study introduces a user-friendly graphical interface designed for the recognition of American Sign Language (ASL) alphabets and digits using OpenCV for real-time
image capture, a convolutional neural network (CNN) for feature extraction, and a random forest classifier for recognition. The application simplifies the recognition process by integrating these technologies, eliminating the need for extensive model
training from scratch. Users can effortlessly capture real-time images of ASL signs,
which are processed, resized, and then input into the CNN for feature extraction. The
extracted features are then classified by the random forest model for swift and accurate identification. The application

displays the model's predictions immediately, aiding in effective learning and communication of ASL. Additionally, the interface includes visualizations of the model's training performance via accuracy and loss graphs, providing insights into the model's reliability. The user-friendly design ensures accessibility for both ASL learners and educators, offering a straightforward and efficient experience.

VI.REFERENCES

[1] H. Koller, C. Hager, M. Huck, and S. D. Nguyen, "Real-time American Sign Language recognition using deep learning," Int. Conf. Pattern Recognit. Mach. Intell., pp. 340–347, 2021.

[2] N. Kumar, R. Kumar, and V. Singh, "ASL alphabet recognition using deep convolutional neural networks," Int. J. Comput. Appl., vol. 178, no. 39, pp. 20–24, 2019.

[3] Y. Li, S. Li, W. Chen, and X. Qi, "Gesture recognition based on deep learning for sign language," Int. J. Autom. Comput., vol. 16, no. 1, pp. 63–72, 2019.

[4] S. Sharma, "A survey on supervised machine learning algorithms," IEEE Int. Conf. Electron. Comput. Commun. Technol., pp. 1–6, 2017.

[5] P. Molchanov, S. Gupta, K. Kim, and J. Kautz, "Hand gesture recognition with 3D convolutional neural networks," IEEE Conf. Comput. Vis. Pattern Recognit. Workshops, pp. 1–7, 2015.

[6] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep structured output learning for unconstrained text recognition," Int. Conf. Learn. Represent., pp. 1–9, 2015.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for largescale image recognition," Int. Conf. Learn. Represent., pp. 1–14, 2015.

[8] C. Zhang, Y. Tian, and J. Liu, "Multimodal deep learning for sign language recognition," J. Vis. Commun. Image Represent., vol. 25, no. 6, pp. 1273–1280, 2014.

[9] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," J. Mach. Learn. Res., vol. 15, no. 1, pp. 1929–1958, 2014.

[10] S. Chuan, B. Regina, and L. G. See, "Hand gesture recognition using Leap Motion with OpenCV," Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops, pp. 1–5, 2014.

[11] T. Theodoridis, "Introduction to pattern recognition: A MATLAB

approach," Pattern Recognit., vol. 40, no. 8, pp. 2121–2131, 2008.

[12] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," Neural Netw., vol. 32, pp. 323–332, 2012.

[13] G. Bradski, "The OpenCV Library," Dr. Dobb's J. Softw. Tools, vol. 25, pp. 120– 125, 2000.

[14] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," Mach. Learn., vol. 46, no. 1, pp. 389– 422, 2002.

[15] D. F. Specht, "Probabilistic neural networks," Neural Netw., vol. 3, no. 1, pp. 109–118, 1990.

[16] H. W. Sorenson, "Least-squares estimation: from Gauss to Kalman," Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, 1970.

[17] T. Starner and A. Pentland, "Real-time American Sign Language recognition from video using hidden Markov models," Proc. Int. Symp. Comput. Vis., pp. 265– 270, 1995.