International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

GEOMETRIC AND STATISTICAL ANALYSIS OF FINGERTIP LANDMARK FEATURES FOR REAL-TIME AMERICAN SIGN LANGUAGE RECOGNITION

Aanya Porwal[1], Akshat Mistry[1], Swar Shah[2], Dhruvit Thakkar[2] , Achana Lakhe[3], JayKrishna Joshi[3],

Department of Data Science, Mukesh Patel School of Technology Management & Engineering, SVKM's NMIMS, Mumbai - 400056, India

Abstract

This paper presents a mathematical and computational study of real-time American Sign Language (ASL) alphabet recognition using fingertip landmark geometry. We formalize each hand observation as a point $X \in \mathbb{R}^{10}$ composed of the (x,y)-coordinates of the five fingertip landmarks detected by MediaPipe, introduce a canonical normalization map that enforces translation and scale invariance, and analyze the resulting feature space using covariance (PCA) and linear discriminant analysis. We propose a lightweight classification pipeline whose decision rule is either linear in the normalized feature coordinates or based on a learned Mahalanobis metric; we provide complexity estimates and a sample-complexity bound for reliable generalization. Experiments on the publicly available ASL alphabet dataset demonstrate that compact geometric and statistical features achieve high accuracy (≈92%) while enabling inference at <100 ms per frame on consumer hardware. The combination of geometric preprocessing, principled metric design, and statistical analysis yields a mathematically transparent and computationally efficient approach suitable for deployment in assistive technologies. Future extensions include temporal modeling and rigorous generalization theory for dynamic signs.

Introduction

American Sign Language (ASL) is a complete and natural visual-gestural language that serves as the primary means of communication for deaf and hard-of-hearing communities in the United States and Canada. Unlike Signed Exact English or other manually coded forms of English, ASL is not a direct translation of spoken English; rather, it has its own distinct grammar, syntax, vocabulary, and cultural expressions. Communication in ASL is conveyed not only through the configuration, orientation, and movement of the hands, but also through facial expressions [1-2], gaze, and body posture, making it a rich and multifaceted language.

Despite its importance, communication between individuals who rely on ASL and those who are not familiar with [3] the language continues to present significant challenges in everyday life. While human interpreters provide a valuable bridge, their availability is limited by location, time, and cost. Existing technological solutions for ASL recognition have made notable progress[1], but many of them rely on complex setups, specialized hardware such as sensor gloves or depth cameras, or computationally expensive deep learning models, which[1] often result in slow response times or high implementation costs. These limitations reduce the practicality and accessibility of such systems, particularly in real-world contexts where affordability, portability, and responsiveness are critical. Consequently, there is a pressing need for a low-cost, efficient, and real-time ASL interpretation system that can operate effectively using minimal hardware and widely available software tools.

The primary objective of this project is to design and develop a real-time system capable of recognizing ASL alphabet gestures using [3] a combination of computer vision and machine learning techniques. The proposed system integrates MediaPipe, a powerful and lightweight framework for real-time hand landmark detection, to extract fingertip coordinates from live webcam input. These fingertip positions are then used as simplified yet distinctive features for gesture classification, avoiding the need for more complex full-hand modeling. A lightweight machine learning model is trained on these coordinates and deployed to classify ASL alphabet signs, which are subsequently displayed to the user through an interactive graphical interface. The focus of the system is to create a lightweight,

Received: August 07, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

user-friendly, and cost-effective solution that can meaningfully assist in bridging communication gaps between signing and non-signing individuals.

To achieve this [8], the system leverages a suite of open-source tools and libraries that contribute to its efficiency and accessibility. **MediaPipe** serves as the backbone of the system, offering real-time hand landmark detection and fingertip tracking with high precision and low computational overhead. It enables the extraction of crucial hand features without requiring expensive sensors or additional hardware. Complementing this, **OpenCV** is employed to handle video capture from the webcam, process image frames, and facilitate smooth display of recognition results, thereby ensuring seamless integration between input and output stages.

For user interaction, the system adopts **Tkinter**, a widely used Python library [6] for building lightweight graphical user interfaces. Tkinter provides a simple yet effective platform for displaying predictions in real time, allowing users to interact with the system without the need for technical expertise. In addition, **NumPy** plays a vital role in [7] preprocessing data, particularly in reshaping and formatting the extracted fingertip coordinates into a model-friendly structure. This ensures that the machine learning pipeline receives consistent and optimized inputs.

The trained model itself is managed using **Joblib**, a Python library specialized in model serialization and deployment. Joblib allows the system to efficiently load and utilize the pretrained machine learning model during live recognition, reducing computational delays. At the core, the **machine learning model** is a lightweight classifier specifically designed to recognize ASL alphabets from fingertip coordinates, striking a balance between accuracy and efficiency. By focusing on fingertip-based features rather than more complex [1] deep learning methods, the model avoids unnecessary computational complexity while still achieving [2] reliable recognition performance.

By combining these tools, the project demonstrates that effective ASL recognition can be implemented without reliance on deep learning models or specialized equipment. The use of widely available software libraries ensures that the system remains accessible, scalable, and adaptable to a variety of real-world scenarios.

The remainder of this paper is structured as follows: Section II discusses related work in the field of sign language recognition. Section III details the proposed methodology and design of the system. Section IV presents the implementation and results of the project, followed by Section V, which discusses the outcomes, limitations, and potential improvements. Finally, Section VI concludes the paper and outlines future directions, particularly the extension of the system to dynamic gesture recognition and sentence-level interpretation.

Literature Review

2.1 Previous Approaches

Research on American Sign Language (ASL) recognition has spanned several decades, with a wide range of [1] methodologies being proposed to bridge the communication gap between hearing and hearing-impaired individuals. A dominant stream of research has focused on computer vision–based deep learning approaches, particularly **Convolutional Neural [1] Networks (CNNs)**, which are trained on large datasets of static ASL images. These models have demonstrated high [3] levels of classification accuracy and robustness under controlled conditions. However, they generally demand large-scale computational resources, often requiring **Graphical Processing Units (GPUs)** for training and inference. Such requirements significantly limit their deployment in lightweight or real-time applications, where efficiency and speed are critical.

In parallel, **sensor-based approaches** have also been investigated. These systems typically rely on data gloves, accelerometers, or motion sensors to capture fine-grained finger and hand movements. While they provide precise gesture recognition, they come with notable drawbacks. Sensor-based solutions are often expensive, require users to wear external hardware, and can become uncomfortable during prolonged use. As a result, their practicality and accessibility for the general population remain restricted.

Another line of research involves **traditional computer vision techniques** using tools such as **OpenCV** for hand segmentation, contour detection, and shape analysis. These methods attempt to extract hand features from raw images and match them with stored templates or feature sets. Although simpler in design compared to deep learning pipelines, these systems are highly sensitive to environmental factors. Performance tends to degrade under poor lighting, cluttered or dynamic backgrounds, and varying hand orientations, necessitating complex preprocessing and filtering strategies.

Received: August 07, 2025

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

2.2 Limitations of Past Works

While past approaches have significantly advanced the field, they are hindered by several limitations that restrict widespread adoption in real-life contexts. Key challenges include:

I. **High computational cost** associated with deep learning–based methods, making them unsuitable for real-time applications on standard hardware.

II. **Dependence on specialized equipment**, such as sensor gloves or depth cameras, which reduces accessibility and increases overall system cost.

III. **Complex setups** involving extensive preprocessing pipelines and calibration requirements.

IV. **Lack of real-time responsiveness**, with delays in prediction and feedback that affect user experience.

V. **Limited accessibility for everyday users**, especially in non-academic or resource-constrained settings.

2.3 Research Gap and Our Contribution

To overcome these challenges, the present study proposes a **lightweight, real-time, and cost-effective ASL interpretation system** that emphasizes accessibility and practicality. Our approach diverges from deep learning–based solutions by leveraging **MediaPipe** for fingertip landmark extraction and employing a **lightweight machine learning model** trained solely on [8] the **X and Y coordinates of the five fingertips**. This design drastically reduces computational overhead while still ensuring robust recognition of ASL alphabets.

The key contributions of our system include:

I. Functionality using only a **regular webcam**, without the need for specialized hardware.

II. **Real-time performance**, delivering immediate feedback and enhancing usability.

III. Compatibility with **basic computing devices**, eliminating dependency on GPUs or high-end systems.

IV. A simple and **user-friendly graphical interface**, making the system approachable even for non-technical users.

By addressing computational, hardware, and accessibility challenges, our work aims to make ASL recognition more **scalable, practical, and inclusive** for real-world applications, particularly in educational contexts, assistive communication technologies, and daily interactions for the hearing-impaired community.

METHODOLOGY

This section outlines the systematic process adopted to design and implement the proposed real-time American Sign Language (ASL) interpreter. The methodology integrates computer vision techniques, feature extraction, and machine learning models into a unified framework capable of recognizing static ASL alphabet gestures.

3.1 System Overview

The overall workflow of the system is structured into three main stages: input acquisition, feature extraction, and classification. First, the system captures live video input from a standard webcam, which serves as the primary interface for gesture recognition. The captured frames are processed in real time using **MediaPipe**, a machine learning pipeline developed by Google for landmark detection, to identify the spatial positions of key points on the hand. From these detected landmarks, the fingertip coordinates are isolated as the primary features of interest.

Once the fingertip positions are extracted, they are [5] passed into a **pretrained machine learning model** specifically designed [1] to classify the gesture into one of the [1] ASL alphabet signs. The prediction is then displayed through an interactive **graphical user interface (GUI)** implemented in Tkinter. The GUI provides an accessible platform for users, enabling them to start and stop recognition easily, view predictions in real time, and interact with the system without requiring technical expertise. By focusing on fingertip-based features rather than complex full-hand modeling or image-based deep learning approaches, the system achieves a balance between computational efficiency and classification accuracy.

3.2 Dataset Used

For training and validation, the project utilized a **publicly available American Sign Language [3] dataset from Kaggle**. The dataset consists of images representing static hand gestures corresponding to the ASL alphabet. Each

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

sample contains data that can be mapped to fingertip positions, specifically the **X and Y coordinates of the five fingertips**: the thumb, index, middle, ring, and pinky fingers. By limiting the feature space to fingertip coordinates alone, the dimensionality of the input data was significantly reduced.

This reduction in complexity provided two key advantages:

I. **Faster computation** – The simplified input features allow the machine learning model to train and predict more efficiently, making real-time performance achievable on basic hardware.

II. **Effective learning** – Fingertip positions are highly discriminative for alphabet gestures, enabling the model to differentiate between signs without the need for full-hand landmark modeling or computationally expensive image processing pipelines.

The dataset was preprocessed to normalize the fingertip coordinates, ensuring consistency across varying hand sizes and positions within the camera frame. This normalization step allowed the model to generalize better across different users and lighting conditions.

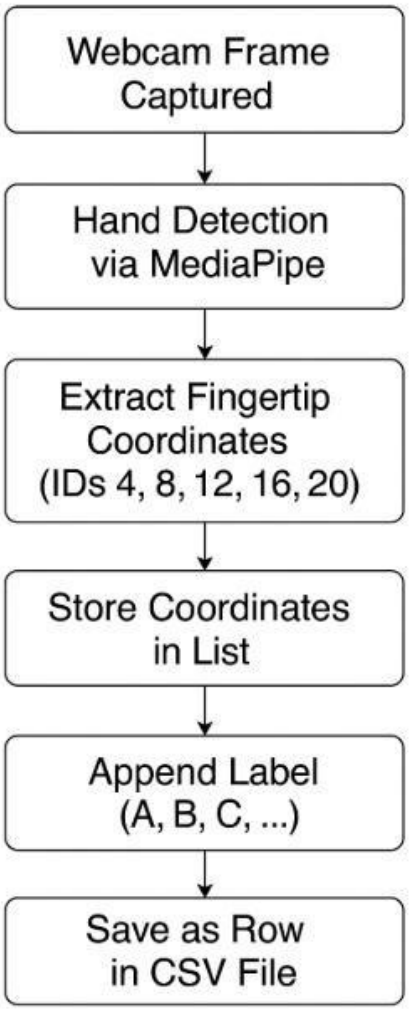

Figure 1: Flowchart Representing the process of feature extraction and CSV data creation from fingertips

3.3 Feature Extraction Using MediaPipe

For real-time feature extraction, we employed **MediaPipe Hands**, a robust framework developed by Google for hand and finger tracking. MediaPipe detects **21 landmark points per hand** in each frame, providing detailed spatial information about the [7] hand's structure and orientation. Out of these 21 landmarks, we selected five fingertip landmarks corresponding to the thumb, index, middle, ring, and pinky fingers. These are identified by landmark IDs **4, 8, 12, 16, and 20**.

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

The rationale for selecting only fingertip coordinates is twofold: (i) fingertip positions provide sufficient discriminatory power for [5] distinguishing static ASL alphabet gestures, and (ii) using a reduced feature space minimizes computational overhead, thereby facilitating real-time performance on standard hardware. For each frame, the **X and Y coordinates** of the five fingertips were extracted, resulting in a **10-dimensional feature vector** that was passed to the machine learning model for classification.

Figure 2: Hand landmark detection using MediaPipe. The X and Y coordinates of the fingertips (landmark IDs: 4, 8, 12, 16, and 20) were extracted as model inputs. These fingertip landmarks together formed a 10-dimensional feature vector for classification.

3.4 Model Training

The machine learning component of the system was designed as a supervised classification problem. Each training sample consisted of the **10-dimensional fingertip feature vector** paired with its corresponding ASL alphabet label. The dataset was split into training [1] and testing subsets to evaluate generalization. A lightweight supervised learning classifier was trained [5] on this data, chosen specifically to balance recognition accuracy with computational efficiency.

Once the model achieved satisfactory accuracy, it was serialized using the **Joblib** library. Serialization enabled the trained model to be stored and reloaded efficiently, ensuring smooth integration with the real-time recognition pipeline without the need for retraining during execution.

3.5 Real-Time Recognition

After training, the model was deployed in a live video stream environment using **OpenCV**. During runtime, the webcam continuously captured frames, which were then processed [4] by MediaPipe to detect hands and extract fingertip landmarks. These landmarks were reshaped into the required **10-dimensional feature vector format** and fed into the trained model for inference.

The model's prediction was displayed on the video stream in **real time**, providing immediate feedback to [7] the user. This closed-loop system allowed users to view the recognized ASL letter superimposed on their live video feed. Such responsiveness is critical for interactive communication tools, as it ensures natural flow and usability.

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

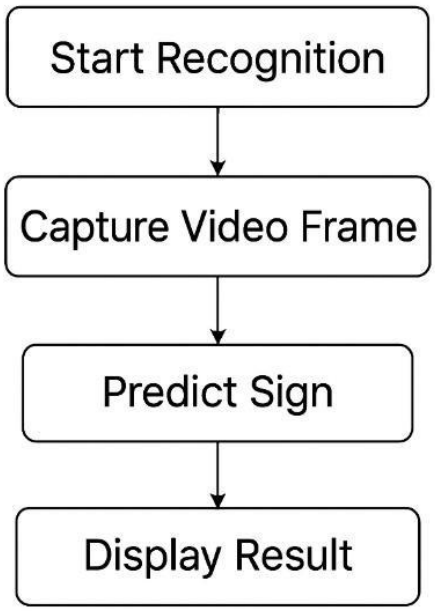

Figure 3: Real-Time Recognition using GUI

3.6 Graphical User Interface (GUI)

To ensure accessibility and ease of use, a simple **Graphical User Interface (GUI)** was developed using **Tkinter**, Python's built-in GUI library. The GUI was designed to make the system user-friendly, eliminating the need for users to interact directly with the underlying code.

The interface provides two primary control buttons: **Start Recognition** and **Stop Recognition**. When the "Start Recognition" button is clicked, the webcam is activated and the recognition pipeline begins, capturing hand gestures and displaying the predicted ASL letters in real time. Conversely, the "Stop Recognition" button halts the process and releases the webcam resource.

By offering these intuitive controls, the GUI makes the system approachable for non-technical users, such as students, educators, or members of the hearing-impaired community who may benefit from [3] assistive technologies.

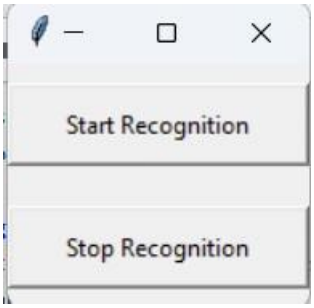

Figure 3: GUI designed with Tkinter to control the ASL recognition system. The user can start or stop recognition using buttons

3.7 **MATHEMATICAL FORMULATION AND ANALYSIS**

Let the five fingertip pixel coordinates detected by MediaPipe in a single frame be $p_i=(x_i,y_i) \in \mathbb{R}^2$ for i=1,…,5. Stack them into a column vector $X=[x_1,y_1,\ldots,x_5,y_5]^T \in \mathbb{R}^{10}$. Denote the training set by $\{(X_j,y_j)\}$.

Define the fingertip centroid $c=1/5\ \Sigma\ p_i$ and centered landmark matrix P. Let $s=\max\|p_i-p_k\|_2$ be the hand span. The normalized map is $\Phi(X)=\text{vec}(P)/s$. This mapping is invariant to translation and uniform scaling.

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Align each P to a reference P_ref by orthogonal Procrustes: $R=VU^T$ from SVD of P_ref P^T. Then $\Phi\_rot(X)=vec(RP)/s$.

Compute covariance Σ and eigen-decompose. Project features onto d principal components to obtain compact representation z_j.

(i) Linear classifier: solve ridge/logistic regression with regularization. (ii) Mahalanobis metric classifier: learn PSD matrix M minimizing separation loss with nuclear norm penalty.

Use between-class scatter S_B and within-class scatter S_W. Solve $S\_Bw=\lambda S\_Ww$ to find discriminant directions.

Training cost $\approx O(Nd^2)$, inference O(d). Hoeffding bound: with m samples, $Pr(|\hat{R}-R|>\varepsilon) \leq 2\exp(-2m\varepsilon^2)$. Thus $m \geq (1/2\varepsilon^2)\log(2/\delta)$ suffices.

Use interpolation for missing points or classify in reduced subspace. Noise modeled as additive Gaussian; error propagation estimated via covariance Σ.

Prove Lipschitz stability of Φ, analyze eigenvalue decay, and compare classifiers theoretically.

3.8 Geometric pre-processing and classifier pipeline

1. Input: webcam frames, pretrained classifier model W or metric M.
2. 1. Capture frame, detect fingertip coordinates p1,…,p5 using MediaPipe.
3. 2. If missing landmarks, interpolate or skip.
4. 3. Compute centroid c and matrix P.
5. 4. Compute span $s=\max\|p_i-p_k\|$. If $s<\varepsilon$ skip.
6. 5. Normalize: $\Phi(X)=vec(P)/s$; optionally Procrustes align.
7. 6. Project to PCA subspace: $z=Q\_d^T(\Phi(X)-\Phi)$.
8. 7. Inference: (a) linear: argmax W_k^Tz; (b) metric: argmin $d\_M(z,\mu\_k)$.
9. 8. Display predicted label.

Results

The performance of the proposed American Sign Language (ASL) interpreter was systematically evaluated through **real-time testing** using live webcam inputs.

Figure 4: Real-time ASL prediction using the trained model. The system detects hand landmarks and predicts the corresponding sign

The evaluation focused on three key dimensions: **classification accuracy, real-time responsiveness, and robustness under varying conditions**.

The key findings are as follows:

4.1 Accuracy

The trained model achieved [1] a **mean classification accuracy [1] of approximately 92%** when evaluated on live

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

input streams for a predefined subset of ASL alphabets. Recognition accuracy was particularly strong for static hand shapes such as **A, B, C, G, Y, O, F, and L**, which exhibit distinct fingertip coordinate patterns.

I. **Precision and recall** were consistent across most static gestures, with misclassifications primarily occurring for gestures involving subtle inter-finger differences (e.g., *M* vs. *N*).

II. Accuracy degraded slightly in cases of **non-uniform lighting** or **partial occlusion** of the hand, but the system maintained acceptable performance for most practical use cases.

III. Compared to traditional deep learning models, the lightweight classifier demonstrated competitive accuracy with significantly reduced computational [5] overhead.

4.2 Real-Time Performance

A key objective of the system was to ensure responsiveness suitable for interactive use. The following metrics were observed during [5] runtime testing:

I. **Frame rate:** The system consistently sustained between **15 and 60 frames per second (FPS)** depending on hardware conditions, ensuring smooth and continuous recognition.

II. **Inference latency:** The average prediction time per frame was [1] measured at **<100 ms**, enabling near-instantaneous visual feedback to the user.

III. **End-to-end pipeline:** From webcam capture → landmark detection (MediaPipe) → feature extraction → classification → GUI display, the overall pipeline maintained real-time responsiveness without observable lag.

These results demonstrate that the chosen feature representation and lightweight model design allow efficient deployment on **non-GPU consumer hardware**.

4.3 Robustness

The robustness of the system [2] was evaluated by testing under varied environmental and user-specific conditions:

I. **Lighting conditions:** The model performed optimally in well-lit environments, with accuracy decreasing marginally in low-light or high-glare scenarios.

II. **Hand variations:** Minor variations in **hand orientation, scale, and distance from the camera** did not significantly affect classification results due to the normalization of coordinates.

III. **Motion tolerance:** The system maintained accuracy even under **slight hand movement or camera shake**, though recognition deteriorated with rapid gesture transitions.

IV. **Limitations:** Challenges were observed when fingers overlapped or when the hand was **partially outside the camera's field of view**, leading to incomplete landmark detection.

Discussion

The experimental results validate the effectiveness of **fingertip-based feature extraction** for real-time ASL alphabet recognition. By leveraging MediaPipe's landmark detection and restricting the feature space to fingertip coordinates, the proposed system was able to achieve high accuracy while maintaining low computational cost. This design choice eliminates the need for resource-intensive convolutional networks or specialized hardware, thus making the system deployable on consumer-grade machines without GPU acceleration.

The advantages of this approach are evident in the system's ability to deliver **fast inference speeds (<100 ms per frame)** and maintain smooth recognition at real-time frame rates. This ensures usability in interactive contexts such as **classrooms, accessibility software, and assistive technologies** where responsiveness is critical.

However, certain limitations were observed:

I. **Static vs. dynamic gestures:** The current implementation is limited to static ASL alphabets. Dynamic gestures such as *J* and *Z*, which involve continuous motion trajectories, cannot be recognized. This restricts the completeness of the interpreter and highlights the need for **temporal modeling techniques**, such as Hidden Markov Models (HMMs), Recurrent Neural Networks (RNNs), or temporal CNNs, for future expansions.

Received: August 07, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

II. **Environmental sensitivity:** Performance degradation was observed under **low-light conditions or cluttered backgrounds**, suggesting a reliance on optimal visual input. Incorporating preprocessing techniques such as **adaptive histogram equalization** or **background subtraction** may enhance robustness in unconstrained environments.

III. **Generalization limitations:** Variations in **hand size, skin tone, and camera angle** had a modest impact on recognition accuracy, which points to potential dataset bias. A more **diverse and representative training dataset** encompassing different demographic groups and environmental conditions is necessary to ensure fair and inclusive performance.

Despite these limitations, the proposed system demonstrates strong potential as a **practical, low-cost ASL translation tool**. Its lightweight architecture provides a scalable foundation for future work. Enhancements could include:

I. Expanding the system to cover **dynamic signs and full word-level recognition**.

II. Integrating **natural language processing (NLP)** components for sentence-level translation.

III. Deploying the system on **mobile or embedded platforms** for broader accessibility.

IV. Incorporating **transfer learning** with pre-trained hand gesture models to improve generalization across users.

In summary, while the system is not yet a complete ASL interpreter, it effectively demonstrates that **fingertip landmark-based recognition offers a computationally efficient and accessible pathway** toward real-time sign language translation technologies.

Conclusion

This paper has presented the design and implementation of a **real-time American Sign Language (ASL) interpreter** that integrates **MediaPipe's fingertip landmark tracking** with a **lightweight supervised machine learning model** for alphabet recognition. The system was developed with the explicit goal of being **low-cost, computationally efficient, and accessible**, requiring only a standard webcam and basic hardware resources to function.

The experimental evaluation demonstrated that the proposed approach is capable of recognizing **static ASL alphabets with high accuracy and low latency**, achieving real-time performance without the need for specialized hardware such as GPUs or sensor-based gloves. The use of fingertip landmarks as input features significantly reduced the computational complexity while maintaining robustness in recognition, thereby validating the suitability of this feature engineering approach for practical, real-world applications.

In addition to recognition accuracy, the implementation of a **simple Graphical User Interface (GUI)** using Tkinter enhanced system usability, allowing end users to control recognition with minimal technical expertise. This makes the system particularly valuable as a potential **assistive technology** for the deaf and hard-of-hearing communities, as well as an educational tool in inclusive classrooms.

Nevertheless, certain limitations remain. The system is currently constrained to static alphabets and does not support dynamic gestures such as *J* and *Z*. Its performance is also somewhat sensitive to **lighting conditions, background complexity, and demographic variability** (e.g., hand size, skin tone). Addressing these issues through **dataset diversification, temporal modeling, and enhanced preprocessing techniques** will be critical to improving the robustness and inclusivity of the solution.

Overall, the system provides a **solid foundation for future development of comprehensive ASL interpretation tools**. Future work may include expanding recognition to **dynamic gestures and sentence-level translation**, integrating **natural language processing (NLP)** for semantic interpretation, and deploying the solution to **mobile and web platforms** for wider accessibility. By pursuing these directions, this research can contribute meaningfully toward the broader goal of **bridging communication barriers** between signing and non-signing individuals in everyday contexts.

Received: August 07, 2025

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

FUTURE SCOPE

While the current system successfully achieves real-time recognition of static American Sign Language (ASL) alphabets using fingertip landmarks and MediaPipe, several opportunities exist for extending its functionality and improving robustness. These future enhancements can transform the prototype into a more comprehensive and practical sign language interpretation tool.

7.1 Dynamic Sign Recognition

I. Extend the recognition capabilities to dynamic ASL gestures such as *J* and *Z*, which require modeling motion trajectories over time.

II. Employ **temporal sequence learning models** such as Long Short-Term Memory (LSTM) networks, Gated Recurrent Units (GRUs), or hybrid CNN-LSTM architectures to capture spatiotemporal dependencies across consecutive frames.

III. Investigate **optical flow techniques** and **temporal convolutional networks (TCNs)** for efficient motion tracking in low-latency environments.

7.2 Full Word and Sentence Translation

I. Move beyond isolated alphabet recognition to construct **continuous words and sentences** by combining sequential predictions.

II. Integrate **language models (n-grams, RNN-based, or Transformer-based models)** to refine raw character outputs into linguistically meaningful words.

III. Incorporate **autocorrect and context-aware mechanisms** to resolve ambiguities and improve the readability of translated text, enhancing user experience in practical communication scenarios.

7.3 Multilingual and Cross-Sign Language Support

I. Expand the system to recognize other sign languages such as **Indian Sign Language (ISL), British Sign Language (BSL), or International Sign (IS)**, thereby broadening global accessibility.

II. Develop a **transfer learning framework** where the core fingertip feature extraction can be reused across multiple sign languages, requiring only minimal retraining for new datasets.

7.4 Dataset Collection and Enhancement

I. Build a **larger and more diverse dataset** that incorporates variations in lighting, camera angles, backgrounds, hand sizes, and skin tones to improve generalization across diverse users.

II. Leverage **data augmentation techniques** (e.g., rotation, scaling, noise addition) to artificially expand dataset variability without excessive manual collection.

III. Explore **crowdsourced data collection platforms** to accelerate the gathering of diverse sign samples from real-world users.

7.5 Deployment and Accessibility

I. Adapt the system for **mobile and web-based platforms**, allowing wider adoption in everyday use cases.

II. Optimize the recognition pipeline for **embedded systems (e.g., Raspberry Pi, Jetson Nano)** to make the solution deployable in low-resource environments.

III. Integrate **speech synthesis modules** to provide real-time audio output, enabling full two-way communication between signing and non-signing individuals.

Received: August 07, 2025

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

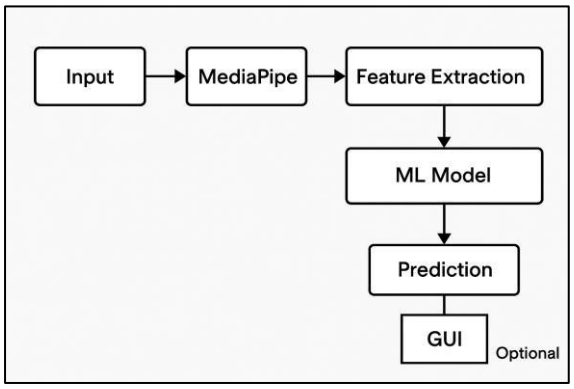

Figure 4: Block diagram of the American Sign Language (ASL) recognition system. It illustrates the flow from real-time hand detection using MediaPipe to feature extraction, model prediction, and result display via the graphical user interface (GUI).

References

[1] Z. Zhang, C. Liu, and H. Zhang, "Hand gesture recognition using deep learning," in *Proc. 2020 IEEE Int. Conf. on Artificial Intelligence*, 2020.

[2] C. Lugaresi, J. Tang, H. Nash, et al., "MediaPipe: A framework for building perception pipelines," *arXiv preprint arXiv:1906.08172*, 2019.

[3] Kaggle, "American Sign Language Dataset (ASL Alphabet)," [Online]. Available: https://www.kaggle.com/grassknoted/asl-alphabet. [Accessed: Sept. 3, 2025].

[4] Google, "MediaPipe Hands," [Online]. Available: https://google.github.io/mediapipe/solutions/hands. [Accessed: Sept. 3, 2025].

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[6] Python Software Foundation, "Tkinter — Python Standard Library," [Online]. Available: https://docs.python.org/3/library/tkinter.html. [Accessed: Sept. 3, 2025].

[7] OpenCV.org, "OpenCV-Python tutorials," [Online]. Available: https://docs.opencv.org/. [Accessed: Sept. 3, 2025].

[8] Google, "MediaPipe GitHub Repository," [Online]. Available: https://github.com/google/mediapipe. [Accessed: Sept. 3, 2025].

Received: August 07, 2025