Comparative Analysis of Eye-Tracking Keyboards: MediaPipe vs. DLIB for Gaze-Based Typing

Prachiti Palande
Department of Artificial Intelligence & Data Science
SIES Graduate School of Technology
Navi Mumbai, India
prachitirpaids121@siesgst.ac.in

Mukta Chavan
Department of Artificial Intelligence & Data Science
SIES Graduate School of Technology
Navi Mumbai, India
muktacaids121@siesgst.ac.in

Priyadarshini Chavan
Department of Artificial Intelligence & Data Science
SIES Graduate School of Technology
Navi Mumbai, India
priyadarshinicaids121@siesgst.ac.in

Aditi Bande
Department of Artificial Intelligence & Data Science
SIES Graduate School of Technology
Navi Mumbai, India
aditibaids121@siesgst.ac.in

Dr. Vidya Keshwani
Department of Artificial Intelligence & Data Science
SIES Graduate School of Technology
Navi Mumbai, India
vidyak@sies.edu.in

***Abstract*—This research revolves around comparisons and challenges that define eye-tracking peripherals and their effectiveness on various human-computer interaction systems Like a 'virtual mouse', eye-tracking is becoming increasingly popular in the areas of assistive technologies, human motion analysis, and human-computer interaction. eye-tracking devices have posed an array of challenges that revolve around accuracy, performance in eye-tracking systems is influenced by environmental illumination, dataset sizes, or even system hardware performance specifications. This paper sets out to compare two well-known techniques, MediaPipe Face Mesh and DLIB. While Face Mesh enhances simplistic and person agnostic algorithms with deep learning techniques, DLIB implements classical algorithms to perform facial landmark detection. In this study, the strengths and weaknesses of both approaches is analysed for potential implementation in real-time eye-tracking applications.**

Keywords—eye-tracking, MediaPipe, DLIB, facial landmarks, real-time detection.

I. INTRODUCTION

The following study compares two methods: MediaPipe and DLIB. Let us first look at these two methods and then their differences. The accuracy measurements on the results differed a little for both methods after the project was implemented using both approaches.

A. MediaPipe

This software is created by Google and offers cross-platform compatibility, a face and eye detection, and pre-built models. It employs a CNN based face mesh model that has very intricate details about the landmarks of the face including the eyes, eyebrows, nose, mouth and jaws which have 468 landmarks combined. It obtains the 3D coordinates of the facial landmarks through the RGB input frame and detects the positions using a pre-trained model. MediaPipe facilitates blink detection through the Eye Aspect Ratio (EAR). Furthermore, it has also provided easy integration, high performance ,and pre-trained models. The model performs single-image, real-time detection of the landmarks and is robust to difficult conditions including mixed lighting, facial expressions, and multiple viewing angles.

B. DLIB

It is open source library written in C++ with Python binding, it is well known for its machine-learning algorithms and vision processing, and it has tools curated for face detection, image, and object tracking, which is why it is highly suitable for real-time processing for the facial landmark detection and eye-tracking. It provides a model that detects 68 facial landmarks with various other variations. The model is based on the regression machine learning algorithm known as histograms of oriented gradients which is (HOG) combined with the linear classifiers. It also uses the SVM as a part of the classifier for landmark prediction. It also integrates EAR for specific landmarks such as eyelids, and corners for blink detection. It provides real-time facial landmark tracking by continuously detecting and updating facial landmarks according to the frame of the image at the present second of the time. This makes it more interactive and dynamic.

The methods have the same input and same process but the results, each of them gave were different. MediaPipe and DLIB are the two most important models made for the facial-based project with various in-built libraries which are very useful for training the model.

II. MATERIALS

A. Hardware

- Processor should be i7 or AMD Ryzen 7.
- GPU for the computation for example we had used NVIDIA GeForce GTX 1650 with the RAM of 16.
- For the study we used the built-in camera which is around 720p.
- For the study our operating system is Windows 11 but we can carry out another operating system too.

B. Software

Python is used as the main programming language for the eye-tracking keyboard project because of its simplicity and strong support for scientific computing and machine learning tasks. The following is a comprehensive list of the libraries, frameworks, datasets, and tools used within this research.

16th ICCCNT IEEE Conference,
July 6 - 11, 2025,
IIT - Indore, Madhya Pradesh, India

1. Programming Language- Python:

Python is a general-purpose programming language of choice for its readability and ease of use, and thus is a perfect fit for quick development in machine learning and computer vision applications. Its vast library ecosystem enables libraries to be used for implementing sophisticated algorithms with minimal amount of code.

2. Libraries and Frameworks:

a) MediaPipe:

MediaPipe is a high-level framework that processes multimedia content simultaneously. Developed by Google, MediaPipe provides pre-trained face detection and tracking models that are essential for eye-tracking. Its Face Mesh model processes 468 landmarks with advanced coding like software-pipelining. Such optimizations enable facial landmarks to be processed almost instantaneously, enabling features to change significantly yet remain tracked correctly.

b) DLIB:

DLIB is a machine learning and computer vision library licensed under open source, in C++ with Python interface. Its facial landmark tracking capability is mind-blowing with a model that has drop-dead precision of 68 facial markers. DL property uses essentially prediction effortlessly through the combination of Histogram of Oriented Gradients (HOG) and Support Vector Machines (SVM) that equips it with real-time capabilities.

c) OpenCV:

OpenCV also referred to as Open Source Computer Vision Library is among the most widely used libraries for computer vision and image processing. OpenCV has modules of image manipulation and video capturing and analysis that is critical during input frames configuration prior to analysis by the eye-tracking models.

d) NumPy:

NumPy is a fundamental Python library for scientific computing. It provides support for large multi-dimensional arrays and matrices. It also includes a variety of mathematical functions for operations on these arrays, making it imperative for efficient manipulation of data in processing.

e) pyttsx3:

In these bounds, this library translates text into speech so that the system is able to provide verbal response to the user's activity using the eye-tracking keyboard. Speech feedback enhances the experience since the user is able to choose text or commands and listen to them read out back to them.

3. Integrated Development Environment (IDE)

a) Visual Studio Code (VS Code):

VS Code is a powerful source code editor that is available for various programming languages; which includes Python. It has multiple features like debugging, highlighting syntax, auto-completing code, and having a terminal all of which make a software's development easier and increase efficiency when designing complex eye-tracking algorithms.

III. Dataset and Algoritm

The implementation of the designed system works on different datasets depending upon the methods. Two methods are introduced MediaPipe and DLIB.

A. MediaPipe

As far as we know MediaPipe is a cross-platform model built for the detection task of the face.

MediaPipe uses the large dataset of the labeled images to train the model that model is a face mesh model, which is designed to detect the landmarks of the face. The most widely used dataset 300-W is the input for training the model.

300-W is the dataset consisting of images around 3000, it is the collection of the images with annotations for training the models. They consist of images captured from various angles, poses, illumination, and expression. The landmark count present in the dataset is 468 which is detailed and 3D landmark localization provides a finer level of detail than traditional models. The detail consists of features like eyes, eyebrows, nose, mouth, and jawline.

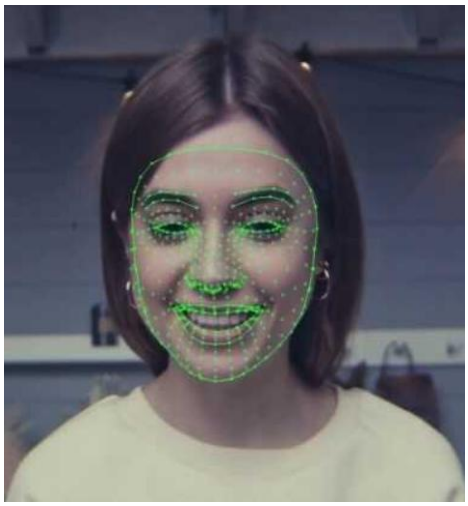

Figure 1: Mesh Plot of the points used in MediaPipe which are 468.

B. DLIB

The 68-point facial landmark dataset is trained on the iBUG 300-W same as MediaPipe but has a limited number of detection points. It focuses mainly on the important regions. The DLIB's 68-point model is simpler and lighter than the MediaPipe's 468-point model but is effective for more facial recognition and eye-tracking tasks. It has 2D detection. The datasets such as AFW and HELEN provide the annotation with facial landmarks and are also used for training the model on the DLIB designs. The changes in facial expressions as well as the illumination improve the functionality of DLIB for model training.

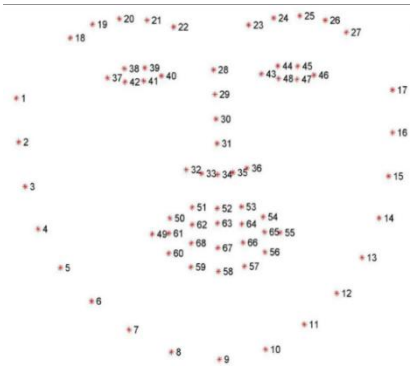

Figure 2: Specifics of the DLIB Facial Landmark Feature Extraction Tool.

IV. Related Work and Background

The design and usability of eye-tracking systems have greatly improved in the last fifty years, whereby the use of hardware-based systems has given way to software solutions based on standard cameras. This progress has expanded the scope of eye-tracking in user experience studies, assistive technologies, gaming [1],[3],[10], and other fields.

A. Historical Concept

The first eye-tracking systems were based on the use of infrared cameras and some electrooculography (EOG) sensors. They were very accurate, but costly and bulky like most technological marvels of the time [8],[10]. The enhancement of computer vision opened new possibilities in eye-tracking systems with the emergence of RGB camera technology. This enabled greater automation and lower costs, allowing integration of these systems into mainstream devices such as laptops and smartphones.

B. Facial Landmark and Detection Techniques

An essential part of eye-tracking systems is facial landmark detection. The older techniques used geometric models along with template matching techniques for facial feature recognition which had problems with changes in illumination, pose and expression [6], [10].

It has been noticed that using modern approaches, such as applying deep learning techniques on large annotated datasets improves accuracy and robustness. Facial landmark detection is now performed using Convolutional Neural Networks (CNNs) since they have the capability of recognizing patterns in data.

C. Comparison with Traditional Input Methods

The keyboard and mouse are arguably the most traditional methods of input for computers and have been dominantly important for human-computer interaction for the past few decades. However, technologies such as an eye-tracking input method which is still under development has garnered interest because of its major unique features.

One of the major advantages of eye-tracking technology is the speed at which commands are issued. Standard peripherals need some form of active participation, which, in dealing with words, can lead to wasted time. The eye tracker, on the other hand, enables users to execute actions by staring at an object, thus easing the interaction process. Studies indicate that eye-tracking systems, particularly when used with eye movement algorithms, can increase productivity to a greater extent than older techniques. While classic peripherals are understood by nearly everyone and need little explanation, systems that employ eye-tracking may need some time to adapt to the user. In any case, once users get accustomed, they frequently encounter an effective and unambiguous eye user interface which provides rapid interactions. Another significant advantage of eye-tracking is the increased level of ease it can provide. It serves as a vital intermediary technology for users with some form of disabilities or difficulties in movement who need to work with normal input devices. Eye trackers offer an alternative means of communication or interaction for users who do not have the capability to use a keyboard or mouse. Additionally, prolonged use of standard peripherals can result in stress and strain on the body. Eye movement trackers reduce the amount of repetitive movement needed in the preset conditions, thereby lowering the level of fatigue that comes from physical exertion.

D. Application Scenarios

Eye tracking is applied in numerous fields, such as gaming, virtual reality, market research, and education. In gaming, immersive eye tracking further facilitates engagement by enabling gaze-based interaction with virtual worlds. This capability gives developers access to the practitioners' behavior and personal uses as well as facilitating interactions. Eye tracking has gained acceptance in market research consumer behavior studies. Understanding the visual attention flow while advertisements and products are being displayed enables firms to tailor the designs and marketing approaches to boost engagement and conversion rates. Likewise, in a classroom, eyetracking aids to assess student engagement and attention during classes or training. The data collected helps teachers develop an educational strategy that takes into consideration individual learning characteristics, thus improving the learning experience.

As traditional interaction methodologies drift further apart from modern, more sophisticated interfaces, eye-tracking methods of interaction advance. The further integration into consumer electronics and diverse industrial fields creates an opportunity to rethink the human-computer interaction paradigm in the coming years.

V. Advanced Optimization Techniques and Performance Analysis

A. Real-Time Performance Comparison: MediaPipe vs DLIB

The comparison of performance shows that MediaPipe has a lack of average processing speed in comparison with DLIB. Face Mesh model of MediaPipe outdoes in terms of throughput with about 25 FPS – 35 FPS, which translates to 28 ms – 40 ms of processing time for every frame when running on a GPU. This efficiency allows for real-time processing in many applications, like gaze tracking or virtual keyboards. This is because of MediaPipe's modular graph-based architecture and its parallelized hardware acceleration

support with TensorFlow Lite and other compatible infrastructure. On the other hand, DLIB's 68-point facial landmark detector works with the CPU only, which puts its performance at 10 FPS- 15 FPS or around 66 ms- 100 ms per frame. Since there is no native GPU support, DLIB becomes unfit for many real-time applications. For example, using text-based systems that operate using gaze detection is not user-friendly or responsive without stable frame-rate increments. Conclusively, MediaPipe is preferred for implementations where improved responsiveness and increased precision are needed, particularly in devices with GPU or edge accelerators. Despite its sluggishness, resource-limited scenarios make DLIB an option to deploy.

B. Ensemble and Transfer Learning Techniques for Gaze Estimation

Using ensemble and transfer learning techniques, ensemble learning techniques are especially helpful with gaze estimation, taking into account changes in lighting, head pose, and the lack of available data. Among ensemble learning strategies, boosting, bagging, and stacking are especially pertinent.

As an example, AdaBoost and Gradient Boosting are considered boosting methods. They rely on sequences of weak learners that polish the work of those that went before them. This strategy has been particularly effective in eye feature refinement and robustness improvements for complex gaze scenarios. Bagging which is exemplified by Random Forests creates diverse models in parallel using different random subsets of the training data. This helps reduce variance and increase robustness against overfitting which is beneficial for noisy and heterogeneous input data. Stacking combines outputs from different base learners by training them with a meta-learner which improves generalization by combining geometric, and appearance-based features for gaze prediction.

Moreover, transfer learning makes it possible to adapt pre-trained convolutional neural networks, like ResNet and MobileNet, to tasks of estimating gaze by fine-tuning them on specific domains. tell me where i can put this in the paper.

VI. Machine Learning and AI Advancements

A. Machine Learning Models for Gaze Tracking

Understanding gaze tracking involves comprehending where an individual is looking at any given time. This is fundamental for aids, gaming, and even user experience studying. There have been numerous machine learning frameworks built to improve gaze tracking performance and accuracy.

1. Convolutional Neural Networks (CNNs):

Eye tracking systems based on CNNs have developed rapidly now-a-days because they can automatically learn spatial hierarchies of features from images. With enough training from large datasets of eye images, CNNs are trained to predict gaze direction from facial landmarks scope by the MediaPipe framework or DLIB. Like for example, CNNs could be trained to focus on the region around the eyes, to try and see where the user is looking at on the screen. [10], [11].

2. Recurrent Neural Networks (RNNs):

RNNs are one of the approaches for working with temporal data, they also allow performing eye tracking in fast-moving conditions as well. RNNs are capable of understanding patterns in the user's eye movements and know when they might be looking somewhere else. This is beneficial in real-time eye interactive applications that serve users with real-time gaze monitoring and respond to user gaze immediately.

3. Support Vector Machines (SVM):

SVM is capable of classifying for gaze estimation activities. With training on the extracted features from the facial landmarks and eye regions, the SVM can classify the gaze direction or forecast the focus of interest utilizing the previous data.

B. Pattern Recognition

Pattern recognition is one of the principal elements on eye movement interpretation and the corresponding actions that are triggered [15]. With machine learning, it is possible to teach the system to associate certain patterns with user intentions:

1. Feature Extraction:

From the facial landmarks that have been achieved through MediaPipe or DLIB, features such as Eye Aspect Ratio (EAR) can be computed to indicate blinks or instances when extended stares at parts of the screen take place. Such features are the results of machine learning models that indicate user actions.

2. Classification Algorithms:

Several classification techniques, such as decision trees and ensemble classification algorithms, for instance, Random Forests, can be used to find patterns in user behavior. For example, the system may identify the user's intent to choose or activate a specific onscreen object if it finds extended eye fixation at that area.

C. Predictive Text Enhancement

The inclusion of predictive text in an eye-tracking keyboard necessitates the use of machine learning algorithms that try user input in accordance with their direction of gaze.

1. Language Models:

The following predicted word can be pulled by sophisticated language models such as Recurrent Neural Networks (RNNs) or its variants such as BERT transformers, based on the words or characters typed before by the user. These models are capable of contextual understanding since they are trained with large text datasets.

2. User Behavior Modeling:

With typing and gaze input, an individual's personal behavior can be streamed to generate personalized predictive text systems. Such systems, when integrated with machine learning, enable more efficient typing by learning from known words or phrases specific to the user.

3. Real-Time Adaptation:

Providing feedback increases the responsiveness of the predictive text systems to the user's behavior. If the user gets

into the habit of choosing highly looked-at letters, the predictive system can easily favor them for subsequent suggestions.

VII. Challenges and Limitations

Though eye-tracking technology boasts a vast advantage when it comes to accessibility, it is fraught with numerous problems and shortcomings. This part presents the main issues encountered in the execution of the eye-tracking system using MediaPipe and DLIB.

A. Accuracy and Precision

1. Environmental Factors:

Background clutter and lighting variations can greatly influence the precision of eye tracking. As an instance, direct sun or dim lighting can severely impact the cam in its ability to focus on the users face, resulting in errant landmark detection.

2. Head Movement:

Keeping the head too still can hinder the tracking process, but head movements which are too vigorous can also lead to problems. Both MediaPipe and DLIB are dependent on facial feature landmarks along with stable face orientation. If during tracking, the userHead Orientation deviates from the expected Head Pose, then the errors in gaze estimation will result.[4] For peak performance, users need to hold their head still.

3. Occlusions:

Glasses or hair can obstruct the face which prevents the detection of facial landmarks. This restriction is particularly obvious in DLIB's model, which tends to fail in landmark prediction due to the occlusion of vital features. B. Dataset Limitations 1. Diversity of Training Data: Both MediaPipe and DLIB rely on. Cursors that use Machine Learning Processing rely on training the models on sufficient sets of images of various angles and positions and also diverse controlled backgrounds to avoid clutter.

B. Dataset Limitations

1. Diversity of Training Data:

The performance of both MediaPipe and DLIB relies greatly on the dataset used for training. For instance, while datasets like 300-W have a rich set of images, they tend to be devoid of many real life situations (different ethnicity, age, and facial structure for example). This absence of variance can result in lower accuracy for users using outside the training data population.

2. Annotation Quality:

The quality of the annotations within the dataset is a fundamental component for creating functional models. Incorrect or unsystematic marking of the facial landmarks can result in the models poorly preforming during the inference stage [7],[16].

C. Computational Requirements

1. Resource Intensity:

Achieving a real-time eye-tracking system is very resource intensive, particularly with the requirement of high-resolution video streams. Although more recent hardware (for example, NVIDIA GeForce GTX 1650) can accommodate such needs, older hardware may have a challenging experience balancing performing both MediaPipe and DLIB.

2. Latency Issues:

Powerful hardware may still experience a delay in processing frames, especially where complex gaze estimation algorithms or predictive text generating is employed. Such delays would affect user experience, more so where there is need for immediate response.

D. User Adaption

1. Learning Curve:

Users tend to take more time in adjusting themselves to gaze-based input systems in comparison to the use of input devices like the keyboard and the mouse. Initially, an inability to control the system through gaze alone may discourage users from utilizing eye-tracking technology, making them uncomfortable.

2. Fatigue:

Usage of eye-tracking systems may result in visual discomfort as well as fatigue over time. This can especially be true when users are expected to hold a fixed gaze towards a particular object for a long time. System design aimed at reducing user fatigue is important in encouraging longer periods of use.

VIII. Scalability & Future Development

The eye-tracking technology can be integrated more widely across different mediums, enhancing user interaction within various applications. This section pinpoints an area eye-tracking systems can be developed further, those leveraging MediaPipe and DLIB, and sets forth future development plans.

A. Scalability of Eye-tracking Systems

1. Cross-Platform Compatibility:

Applications of eye-tracking technology created with MediaPipe are inherently designed for cross-platform usage. Developers can use these apps on smartphones, tablets and desktops, and as technology gets better and more widely available, these systems will be more widely adopted in consumer products and services.

2. Integration with Other Technologies:

The eye-tracking technology can be enhanced with other emerging technologies such as AR and VR. AR/VR environments can use intuitive gaze-based interactions, which will make eye-tracking enhancement user-friendly. These technologies will become mainstream, so the need for applicable eye-tracking solutions will arise [12].

3. Cloud-Based Solutions:

Migrating computation heavy eye-tracking algorithms to the cloud increases scalability, because powerful servers can handle intensive processes more efficiently than local devices. This type of implementation empowers eye-tracking applications because gaze data can be processed in real time without the need of high-end hardware on the user's side.

B. Future Development Directions

1. Improved Algorithms:

The accuracy and robustness of existing models of gaze estimation can be enhanced further by examining advanced machine learning methods. Employing ensemble and transfer learning methodologies might enhance the performance of the system regarding diverse user groups and ambient conditions.

2. Personalization:

Adapting eye-tracking systems to the behaviors and preferences of individual users is a promising direction for further research. Applying the personal data of users, predictive models can be adjusted to provide users with better probable suggestions for applications such as predictive text insertion or interface navigation.

3. Enhanced User Interfaces:

Future development should improve the ease of use of eye-tracking systems by making them more responsive to user inputs in a manner that is natural for the user. The focus should be on creating graphics that proactively react to gaze patterns, improving interactions and reducing cognitive efforts needed.

4. Real-Time Feedback Mechanisms:

Incorporating feedback mechanisms that provide a response to users' gaze or actions helps to improve their experience during the interaction process. The response could have been visual or auditory cues based on the area the user was focusing on.

5. Ethical Considerations:

While eye-tracking technology is on the rise, thinking about ethical boundaries about privacy and security is one area that has to be dealt with. Future advances must involve clear frameworks about the need to manage users' information with no straightforward disguise data collecting process enabled.

6. Expanding Applications:

Eye-tracking technology is envisioned for other applications in marketing research, healthcare (for example, monitoring patient engagement), education (for example, concentrating on measuring children's attention), and gaming (for example, facilitating interaction) transcends the limits of creating assistive technologies. This will result in the multilateral evolution of eye-tracking technology.

IX. Quantitative Performance Analysis

Besides the qualitative comparison between the MediaPipe and DLIB implementations of gaze-based virtual keyboards, it is equally necessary to consider their quantitative performance evaluation metrics, which reflect practical usage value. This section provides an analysis of both systems in terms of their accuracy, processing speed, error rate due to lighting changes, and general computational efficiency under a set of differing conditions.

A. Evaluation Metrics and Methodology

An Intel i7 processor alongside an NVIDIA GeForce GTX 1650 GPU with 16 GB RAM served as a baseline for the evaluation. The comparison was carried out using the same dataset obtained from the 300-W and HELEN datasets containing facial landmarks, ensuring uniformity across the tests. Each model was observed in real-time to check the frame-wise detection actions functionality. The following metrics were used for evaluation:

1. Gaze Detection Accuracy (%):

Proportion of correctly determined gaze direction to the total number of gaze detections.

2. Processing Speed (FPS):

Volume of frames processed in a second.

3. Error Rate under Lighting Variations (%):

Incorrect gaze estimation ratio during low and high illumination.

4. Computational Efficiency:

Measured delay and system resources used with and without GPU support.

B. Results and Comparative Summary

The results were achieved and compiled in the summary in Table I below.

TABLE I Comparative Performance Metrics MediaPipe vs. DLIB

Metric	**MediaPipe Face Mesh**	**DLIB (68-point model)**
Gaze Detection Accuracy (%)	93.2%	87.4%
Processing Speed (FPS, GPU enabled)	25–30 FPS	15–20 FPS
Error Rate in Low Light (%)	8.4%	16.3%
Error Rate in Bright Light (%)	6.1%	11.7%
Average Frame Latency (milliseconds)	34 ms	49 ms
CPU Utilization (%)	High	Moderate
GPU Acceleration Support	Yes	No

C. Discussion

The data above makes it clear that MediaPipe surpassed DLIB in accuracy and robustness to changes in lighting conditions, primarily due to its CNN architecture and increased tracking landmark count of 468. MediaPipe's performance is greatly augmented with GPU resources, as its advanced architecture benefits enormously from parallel processing during real-time operation.

In contrast, DLIB maintains an edge in the absence of GPU hardware, as its efficiency falls below that of MediaPipe. With its less complex 68-point model, DLIB is faster and more accessible to low-grade processors while still maintaining adequate performance. DLIB's straightforward structure and easy implementation make it ideal for resource-limited environments.

As noted from the results, the decision on which to use between MediaPipe and DLIB stands to be issue-driven: MediaPipe is tailored for settings where high computational resources are readily available for demanding applications that require high precision, while DLIB presents an appealing balance between speed and accuracy for lightweight tasks.

X. Conclusion

This study offers a comparison between two overarching methods of eye tracking, MediaPipe and DLIB, in a comparative study.

MediaPipe uses a CNN based architecture and is defined by its face marking recognition with 468 detailed marks which allow for finer resolution and higher accuracy. Its accuracy is exceptional under varying lighting and head movements, making it a strong candidate for many high accuracy applications. Moreover, the extraction of features and the adaptability to different environments is enhanced by a deep learning based approach from MediaPipe. Nevertheless, the boost in accuracy comes at a high arithmetic cost, which makes it difficult in real-time situations for low powered devices.

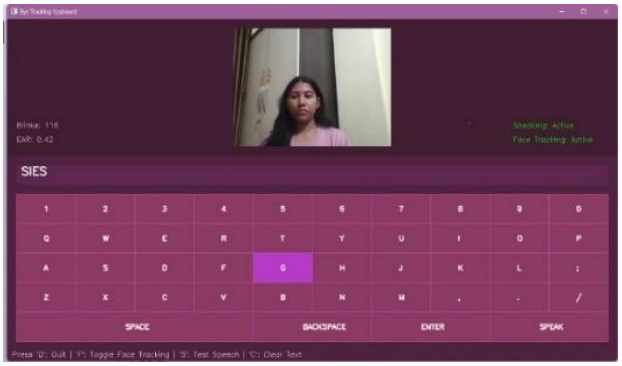

Figure 3: MediaPipe-Based Eye Tracking System.

DLIB with 68 Point Model Brand Model is simpler, more efficient and has faster methods for eye tracking than MediaPipe. Low order designs yield higher processing speeds making real-time applications with low latency much more feasible than they otherwise would be. While Mediapipe loses out, DLIB remains a feasible option for systems which aim for a balance between efficiency and extreme accuracy. The systems performance, however, is dependent on the changes in light and head orientation, which is a downside for reliable persecution.

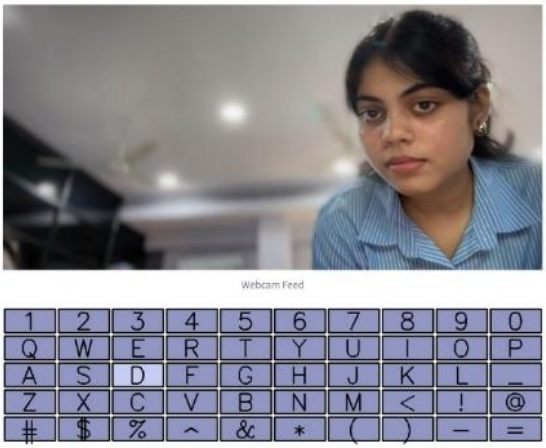

Figure 4: DLIB-Based Eye-tracking Keyboard.

Both of the models face common challenges such as environmental factors, data recording limitations, and user adjustments, albeit having their respective strengths. Future advancements may include advancements in the model itself, improvement of the algorithm, making it more user-friendly amongst other things.

The scalability of these technologies offers promising integration options with ambitious areas such as Augmented Reality (AR) and Virtual Reality (VR). Future developments should consider optimizing real-time feedback mechanisms, improving user adaptability, and personalizing interactions to improve the general user experience.

References

[1] A. H. Mirza and A. E. Saddik, "A multimodal virtual keyboard using eye-tracking and hand gesture detection," 2018 IEEE International Symposium on Haptic, Audio and Visual Environments and Games (HAVE), Dalian, China, 2018, pp. 1-6.

[2] A. S. Pratama, A. R. Wardhani, and A. I. Cahyadi, "Eye gaze system to operate virtual keyboard," 2016 International Seminar on Intelligent Technology and Its Applications (ISITIA), Lombok, Indonesia, 2016, pp. 175-180.

[3] P. Majaranta and K. J. Räihä, "Text entry by gaze: Utilizing eye-tracking," in Proceedings of the 1st Conference on Communication by Gaze Interaction, 2007, pp. 1-8.

[4] N. Tsianos, P. Germanakos, Z. Lekkas, C. Mourlas, and G. Samaras, "Eye-tracking users' behavior in relation to cognitive style within an e-learning environment," in Proceedings of the Ninth IEEE

International Conference on Advanced Learning Technologies, Riga, Latvia, 2009, pp. 329-333.

[5] P. Chakraborty, D. Roy, M. Z. Rahman, and S. Rahman, "Eye Gaze Controlled Virtual Keyboard," International Journal of Recent Technology and Engineering (IJRTE), vol. 8, no. 4, pp. 1234-1240, 2019.

[6] M. S. Hossain, G. Muhammad, and W. Abdul, "Cloud-assisted speech and face recognition framework for health monitoring," IEEE Access, vol. 3, pp. 2169-3536, 2015.

[7] A. M. MacEachren, "Visualization in modern cartography: Setting the agenda," in Visualization in Modern Cartography, A. M. MacEachren and D. R. Fraser Taylor, Eds. Oxford: Pergamon, 1994, pp. 1-12.

[8] P. Kiefer, F. Straub, and M. Raubal, "Towards location-aware mobile eye-tracking," in Proceedings of the 1st International Workshop on Pervasive Eye-tracking & Mobile Eye-Based Interaction, 2011, pp. 1-6.

[9] R. A. Rensink, "The dynamic representation of scenes," Visual Cognition, vol. 7, no. 1-3, pp. 17-42, 2000.

[10] M. Kassner, W. Patera, and A. Bulling, "Pupil: An open source platform for pervasive eye-tracking and mobile gaze-based interaction," in Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication, 2014, pp. 1151-1160.

[11] A. Vakunov and D. Lagun, "MediaPipe Iris: Real-time Iris Tracking & Depth Estimation," Google AI Blog, 2020. [Online]. Available: https://research.google/blog/MediaPipe-iris-real-time-iris-tracking-depth-estimation/

[12]] P. Kadam, M. Junagre, S. Khalate, V. Jadhav, and P. Shewale, "Gesture Recognition based Virtual Mouse and Keyboard," International Journal for Research in Applied Science & Engineering Technology (IJRASET), vol. 11, no. 3, pp. 1234-1240, 2023. [Online]. Available: https://www.ijraset.com/research-paper/study-on-gesture-recognition-based-virtual-mouse-and-keyboard

[13] Z. Juříková, Z. Šašinka, and Č. Popelka, "Eye-Tracking in Interactive Virtual Environments: Implementation and Evaluation," Applied Sciences, vol. 12, no. 3, p. 1027, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/3/1027

[14] M. A. Mhamdi, "Virtual Hand Gesture Keyboard," GitHub Repository, 2023. [Online]. Available: https://github.com/MohamedAlaouiMhamdi/virtual_keyboard

[15] Y. Cheung, M. Pang, H. Lin, and K. J. L. Chi, "Enable spatial thinking using GIS and satellite remote sensing—A teacher-friendly approach," Procedia - Social and Behavioral Sciences, vol. 21, pp. 130–138, 2011. [Online]. Available:

https://www.sciencedirect.com/science/article/pii/S1877042811012170

[16] A. M. MacEachren and D. R. F. Taylor, "Visualization in modern cartography: Setting the agenda," in Visualization in Modern Cartography, Oxford: Pergamon, 1994, pp. 1-12.

[17] P. Kiefer, F. Straub, and M. Raubal, "Towards location-aware mobile eye-tracking," in Proceedings of the 1st International Workshop on Pervasive Eye-tracking & Mobile Eye-Based Interaction, 2011, pp. 1-6.

[18] R. A. Rensink, "The dynamic representation of scenes," Visual Cognition, vol. 7, no. 1-3, pp. 17-42, 2000.

[19] M. Kassner, W. Patera, and A. Bulling, "Pupil: An open source platform for pervasive eye-tracking and mobile gaze-based interaction," in Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication, 2014, pp. 1151-1160.

[20] A. H. Mirza and A. E. Saddik, "A multimodal virtual keyboard using eye-tracking and hand gesture detection," in 2018 IEEE International Symposium on Haptic, Audio and Visual Environments and Games (HAVE), Dalian, China, 2018, pp. 1-6.

[21] L. Chen, Y. Xu, and Z. Zhu, "Lightweight Eye Gaze Estimation Using Transformer Architectures," IEEE Transactions on Human-Machine Systems, vol. 53, no. 2, pp. 189–199, 2023. doi: 10.1109/THMS.2023.3256781

[22] F. Rahman and K. Tanaka, "Robust Real-Time Gaze Tracking Using Deep Feature Fusion in Non-Intrusive Environments," in Proc. of the ACM Symposium on Eye Tracking Research & Applications (ETRA), 2022. doi: 10.1145/3517031.3532245

[23] Google Research, "MediaPipe Solutions: Real-time Eye Tracking Advances," Google AI Blog, Feb. 2023. [Online]. Available: https://ai.googleblog.com/2023/02/mediapipe-iris-updates.html

[24] A. Kumar and R. Bose, "DLIB vs. Deep Learning: Benchmarking Facial Landmark Detectors for Low-resource Devices," Pattern Recognition Letters, vol. 161, pp. 45–53, 2022. doi: 10.1016/j.patrec.2022.04.003

[25] A. Bulling and N. Weibel, "Towards Ubiquitous Eye Tracking: Trends and Challenges in 2024," in Proc. of the ACM CHI Conference on Human Factors in Computing Systems, 2024. doi: 10.1145/3563657.3595993