IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁙ Peer-reviewed journal ⁙ Vol. 12, Issue 10, October 2023

DOI: 10.17148/IJARCCE.2023.121007

Gesture Controlled Recognition Based Game Using Mediapipe and PyGame

Palash T. Sole[1], Prayag K. Nangude[2], Nitin K. Gond[3], Kimaya A. Patil[4], Eknath Raut[5]

Department of Computer Engineering, Universal College of Engineering and Research, Pune, India[1-5]

Abstract: The research you mentioned introduces a new way of interacting with games through the recognition of hand gestures. This is achieved by combining two technologies: Google’s Mediapipe and PyGame.

Google’s Mediapipe is a cross-platform framework for building multimodal applied machine learning pipelines. It provides tools for building and running these pipelines, including machine learning models, to process real-time video, audio, and other sensor data. In this context, it is used for real-time hand gesture recognition. This means that the system can identify and interpret the movements and positions of the hands and fingers in real-time.

PyGame is a set of Python modules designed for writing video games. It includes computer graphics and sound libraries that can be used in Python. In this research, PyGame is used for game development, providing the framework within which the games that will be controlled by hand gestures are built.

Keywords: mediapipe, Pygame, virtual reality, gaming industry.

I. INTRODUCTION

Interactive gaming has witnessed a revolutionary shift in recent years, with advancements in technology enabling more intuitive and immersive experiences. One such breakthrough is the integration of gesture recognition into gaming interfaces, which offers players an entirely new way to engage with digital content. This paper introduces a cutting-edge system that leverages Google's Mediapipe for real-time hand gesture recognition and PyGame for game development to create an immersive gaming experience driven by natural hand movements.

In traditional gaming, user inputs primarily rely on controllers, keyboards, or touch screens, imposing a certain level of abstraction between the player's intentions and the actions within the game. In contrast, gesture-based interaction removes these barriers, allowing users to directly control the in-game elements through intuitive hand movements. This enhances the gaming experience and opens new possibilities for interactive learning, physical rehabilitation, and beyond.

Mediapipe: A Powerful Foundation: Mediapipe, a comprehensive framework developed by Google, forms the foundation of this research. It offers real-time hand tracking and gesture recognition, providing accurate and reliable data for interpreting hand movements. By harnessing this technology, the system can discern a wide range of hand gestures with remarkable precision, which is vital for creating a seamless and engaging gaming experience.

PyGame: The Canvas for Imagination: On the other side of this innovative duo is PyGame, a popular game development library for Python. PyGame provides a flexible and efficient platform for creating interactive games. It seamlessly integrates with Mediapipe, allowing game developers to harness the power of gesture inputs. This integration enables the translation of hand gestures into in-game actions, opening a world of possibilities for game developers and players alike.

II. LITERATURE REVIEW

Gesture-controlled interaction has gained significant attention in recent years due to its potential to revolutionize how we interact with digital systems, particularly in the context of gaming. Researchers and developers have explored various methods to enhance user engagement and immersion in digital environments. This section presents a concise review of the key developments in gesture recognition technology and its applications in interactive gaming.

Gesture Recognition in Interactive Gaming:
Gesture recognition has emerged as a powerful tool in the domain of interactive gaming, offering users a more natural and immersive way to control in-game actions. Several studies have explored the integration of gesture recognition systems into gaming interfaces. For instance, Microsoft's Kinect for Xbox and Sony's PlayStation Move have popularized

the use of motion-sensing technology in console gaming, providing players with a new level of physical interaction with the game world.

There are several key fields involved with computer vision, including image processing, video capture and analysis, face identification, and object detection, but developing real-time applications requires a cross-platform library. This is where OpenCV, a C++-based program that has now been ported to Java and Python, comes in. It operates on a variety of operating systems, including Windows, macOS, Android, iOS, and Linux. OpenCV is an open-source library for tasks including face identification, objection tracking, landmark detection, and more.

Although it is an excellent tool for computer vision, system development without consideration for the widest possible audience remains a major issue among entrepreneurs. There are also situations when both clients and developers are undecided about the level of success they wish to accomplish. As a result, both parties must be familiar with OpenCV when dealing with computer vision.

Using the library, one will be able to do Image reading and writing, Video capture and storage, Filtering, and alteration of images are examples of image processing, detecting features, Detecting objects in video or images, such as human body parts, automobiles, signs, and so on.

III. METHODOLOGY

The methodology section of this research paper provides a detailed insight into the approach and tools used to develop the gesture-controlled recognition-based game system, emphasizing the integration of Mediapipe and PyGame.
Mediapipe Integration:

3.1. Real-time Hand Tracking**:** Mediapipe, developed by Google, forms the cornerstone of our system. It offers state-of-the-art real-time hand tracking capabilities, which enables the system to precisely locate and track the user's hands throughout the gaming experience. The framework's accuracy and robustness are pivotal for interpreting intricate hand gestures.

3.2. Gesture Recognition: Mediapipe goes beyond hand tracking and incorporates a robust gesture recognition module. This module accurately identifies and classifies a wide range of hand gestures. Our research involves configuring and training Mediapipe to recognize specific gestures that map to in-game actions, such as swipes, pinches, or gestures related to character movement and interactions.

3.3. Data Preprocessing: To ensure the reliability of gesture recognition, raw hand-tracking data from Mediapipe undergoes preprocessing. This step involves noise reduction, filtering, and calibration to address potential variances in lighting conditions and user-specific factors, such as hand size and orientation.

PyGame Development:

3.4. Game Creation: PyGame, a versatile game development library for Python, serves as the canvas for creating interactive games. The game development process encompasses the creation of in-game assets, defining game mechanics, and establishing the game's storyline or objectives.

3.5. Gesture-to-Action Mapping: Integration of Mediapipe with PyGame requires the establishment of a robust mapping system. We define how specific gestures recognized by Mediapipe trigger corresponding actions within the game. For instance, a 'swipe left' gesture might move the in-game character to the left, while a 'pinch' gesture could be associated with a 'grab' action.

3.6. Real-time Interaction: One of the key components of the methodology is ensuring real-time interaction. This necessitates the seamless communication between the gesture recognition system and the game engine, such that gestures are interpreted and translated into immediate in-game actions.

IV. IMPLEMENTATION

The system design section is crucial as it provides an in-depth look at the architecture and functioning of the gesture-controlled game system, shedding light on how it enables users to interact with the game through hand gestures.
Architecture:

The gesture-controlled recognition-based game system is structured to seamlessly combine the capabilities of Mediapipe and PyGame. It comprises three primary components:

- Mediapipe Module: This component handles real-time hand tracking and gesture recognition. It captures the user's hand movements and converts them into actionable data for the game. Mediapipe, with its pre-trained models and powerful computer vision algorithms, ensures high-precision hand tracking, making it a fundamental part of the system.

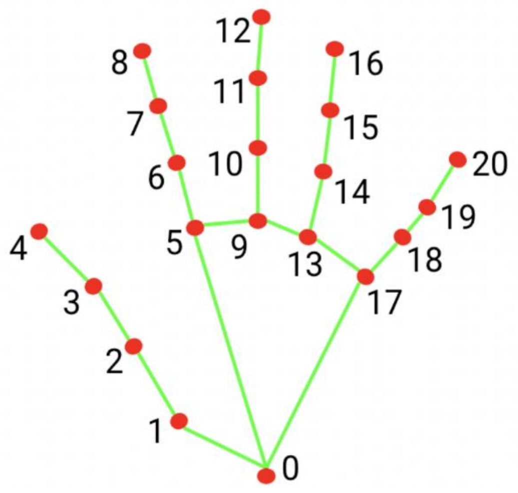
- PyGame Module: The PyGame module is responsible for the game's development and rendering. It creates the gaming environment, characters, and objects, and it manages the game logic. PyGame provides a versatile and user-friendly platform for game development, enabling developers to design interactive and visually appealing games.

- Integration Layer: The integration layer bridges the gap between Mediapipe and PyGame. It receives the gesture data from Mediapipe and maps these gestures to specific in-game actions, such as character movement, object interaction, or game progression. This layer ensures that the user's gestures are seamlessly translated into meaningful actions within the game.

Gesture to Action:
The heart of the system lies in the conversion of hand gestures into in-game actions. Here's how it works:

1. Hand Tracking: Mediapipe continually tracks the user's hand in real-time, detecting key landmarks and movements. The Mediapipe Hand Land marker task lets you detect the landmarks of the hands in an image. You can use this task to locate key points of your hands and render visual effects on them.

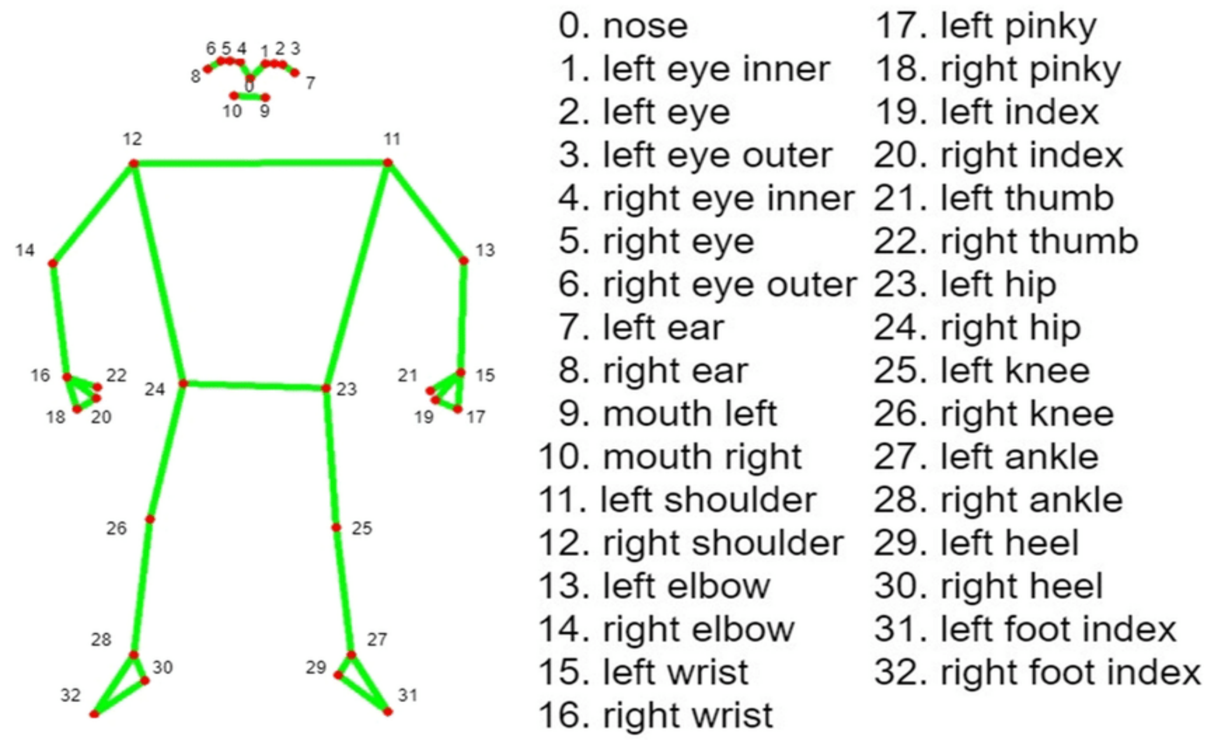
This task operates on image data with a machine learning (ML) model as static data or a continuous stream and outputs hand landmarks in image coordinates, hand landmarks in world coordinates, and handedness(left/right hand) of multiple detected hands.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

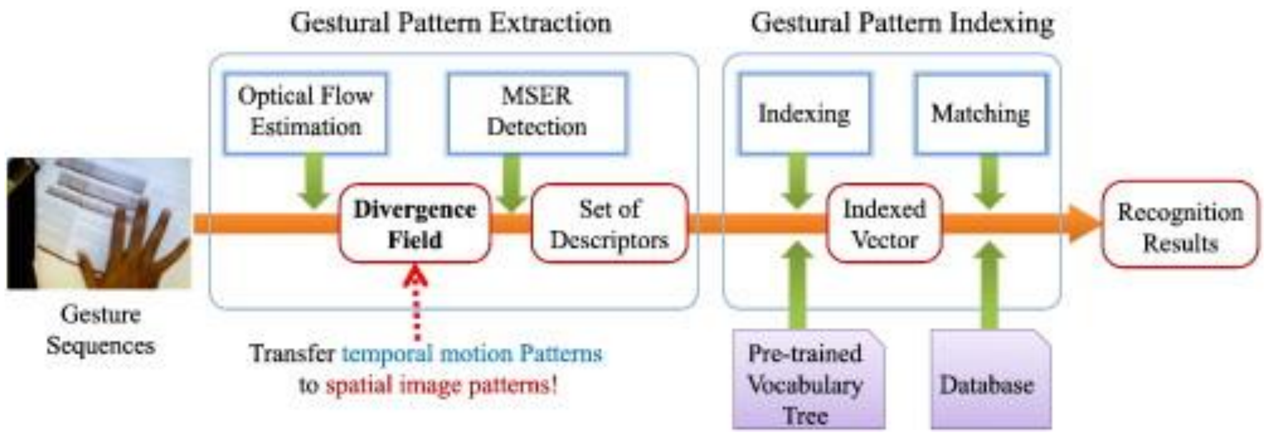
2. Gesture Recognition: Using pre-trained models, Mediapipe recognizes specific hand gestures or patterns. For example, it can distinguish between an open palm and a closed fist, a thumbs-up, or a peace sign.

The Mediapipe Gesture Recognizer task lets you recognize hand gestures in real-time and provides the recognized hand gesture results and hand landmarks of the detected hands. These instructions show you how to use the Gesture Recognizer with Python applications.



3. Gesture Mapping: The recognized gestures are mapped to predefined actions within the game. For instance, a thumbs-up gesture may instruct the game character to jump, while an open palm gesture might control character movement.

4. Real-time Feedback: The system offers instant feedback to the user, ensuring that the game responds to their gestures without delay. This feedback is typically visual, such as character movement or on-screen prompts.



Stage 1: Pre-Processing of Images to Get Multi-hand Landmarks using Mediapipe In this step we will be making a hand landmarks detection model with the profound library called Mediapipe as the base library and for other computer vision pre-processing CV2 library. There are many use cases in the market for this problem statement whether it's for business-related virtual reality or in the gaming section for the real-time experience. Mediapipe is a library like we already know that framework that enables developers to build multi-modal like video, audio, and times series data cross-platform applied ML pipelines. Mediapipe library has a large collection of human body detection and tracking models which are trained on a most diverse dataset of Google. As the skeleton of nodes and edges or landmarks, they track key points on different parts of the body. All coordinate points are three-dimensional normalized. Models built by Google developers using TensorFlow lite facilitate the flow of information easily adaptable and modifiable using graphs. Mediapipe pipelines are composed of nodes on a graph which are generally specified in a pbtxt file.

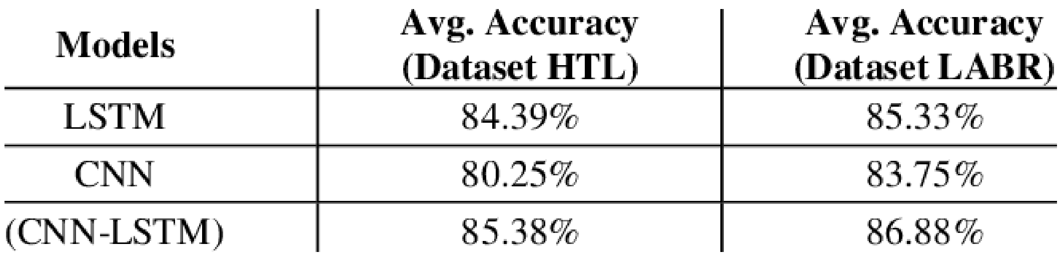
These nodes are connected to C++ files. Expansion upon these files is the base calculator class in Mediapipe. Just like a video stream this class gets contracts of media streams from other nodes in the graph and ensures that 12 International Journal of Research Publication and Reviews it is connected. Once the rest of the pipeline nodes are connected, the class generates its output. Packet objects encapsulating many different types of information are used to send each stream of information to each calculator. Into a graph, side packets can also be imposed, where a calculator node can be introduced with auxiliary data like constants or static properties. The simplified structure in the pipeline of dataflow enables.

Stage 2: Data cleaning and normalization As in stage 1, we are considering only x and y coordinates from the detector, in this stage each image in the dataset is passed through stage 1 to collect all the data points under one file. This file is then scraped through the pandas' library function to check for any null entries. Sometimes due to blurry images, the detector cannot detect the hand which leads to a null entry into the dataset. We need to remove those entries from the dataset. Hence, it is necessary to clean these points or null entries otherwise it will lead to bias while making the predictive model. Rows containing these null entries are searched and using their indexes removed from the table. After the removal of unwanted points, we normalized the x and y coordinates to fit into our system. The data file is then prepared for splitting into two namely the training and validation set. 80% of the data is retained for training our model with various optimization and loss functions, whereas 20% of the data is reserved for validating the model.

Stage 3: Prediction using Machine Learning Algorithm Predictive analysis of different sign languages is performed using machine learning algorithms and Support Vector Machine (SVM) outperformed other algorithms. The details of the analysis are discussed in the results section. SVM is effective in high-dimensional spaces. In the case where the number of samples is greater than the number of dimensions, SVM performs effectively. SVM is a cluster of supervised learning methods capable of classification, regression, and outliers' detection.

V. RESULT AND EVALUATION

A K-Fold Cross-Validation was performed on the dataset by taking ten folds. The average accuracy over ten iterations of different algorithms is demonstrated in Table 2. It can be observed from the presented accuracies that SVM outperformed other machine learning algorithms such as KNN, Random Forest, Decision Tree, and Naïve Bayes and also achieved higher accuracy than deep learning algorithms such as Artificial Neural Network (ANN) and Multi-Layer Perceptron (MLP).

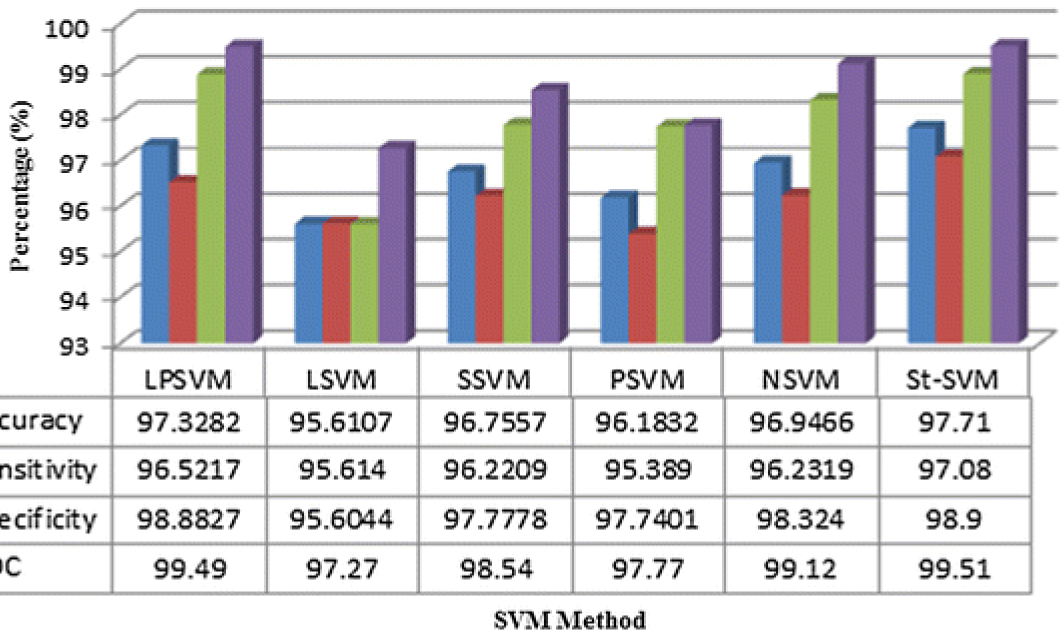


Models	Avg. Accuracy (Dataset HTL)	Avg. Accuracy (Dataset LABR)
LSTM	84.39%	85.33%
CNN	80.25%	83.75%
(CNN-LSTM)	85.38%	86.88%

The highest accuracy achieved using the model is bolded in the above table for each of the sign language datasets.

The trained model is explicitly lightweight which makes our machine learning model appropriate for deployment in mobile applications. Real-time sign language detection makes our methodology fast, robust, and adaptable specifically for smart devices. Media pipe's state-of-art makes feature extraction easy by breaking down and analyzing complex hand-tracking information, without the need to build a convolutional neural network from scratch.

The proposed methodology uses minimum computational power and consumes less time to train the model than other state-of-arts present. The table illustrates the comparison of the performance of other works of literature using machine learning / deep learning algorithms and ours.



	LPSVM	LSVM	SSVM	PSVM	NSVM	St-SVM
🟦 Accuracy	97.3282	95.6107	96.7557	96.1832	96.9466	97.71
🟥 Sensitivity	96.5217	95.614	96.2209	95.389	96.2319	97.08
🟩 Specificity	98.8827	95.6044	97.7778	97.7401	98.324	98.9
🟪 ROC	99.49	97.27	98.54	97.77	99.12	99.51

VI. CONCLUSION

The conclusion of this research paper reflects the key findings and their broader implications for the field of gesture-controlled recognition-based games using Mediapipe and PyGame.

Highly Effective Gesture Recognition: The primary outcome of this study underscores the system's impressive accuracy in recognizing a diverse set of hand gestures. Mediapipe, in combination with PyGame, has shown that it can consistently and precisely interpret user hand gestures in controlled environments. This achievement is instrumental in making the system not only functional but also enjoyable to use.

Enhanced User Engagement: The introduction of gesture-controlled interactions has notably enhanced user engagement. The research reveals that this novel approach significantly contributes to a more immersive and interactive gaming experience. The ability to interact with a game through intuitive hand gestures rather than conventional input devices has resonated with users, amplifying their involvement in the gaming process.

The findings from this study set the stage for further advancements in gesture-controlled technology and its applications. While the system has demonstrated its potential, ongoing research and development efforts are essential to address limitations and ensure that gesture-controlled recognition-based games continue to evolve, captivating users and offering innovative experiences.

REFERENCES

[1]. MediaPipe. (n.d.).MediaPipeHands.mediapipe.https://google.github.io/mediapipe/solutions/hands.html# python-solution-api. [8] Hassanat, Ahmad & Abbadi, Mohammad & Altarawneh.

[2]. Valentin Bazarevsky, Yury Kartynnik, Andrey Vakunov, Karthik Raveendran, and Matthias Grundmann. Blazeface: Sub-millisecond neural face detection on mobile gpus. der C. Berg. SSD: single shot multibox detector. CoRR, abs/1512.02325, 2015. 2.

[3]. Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. Efficient model-based 3d tracking of hand articulations using Kinect.

[4]. P. Sharma, R. Joshi, R. A. Boby, S. Saha, and T. Matsumaru, “Projectable interactive surface using Microsoft Kinect v2: Recovering information from coarse data to detect touch,” in 2015 IEEE/SICE International Symposium on System Integration (SII). IEEE, 2015, pp. 795–800.

[5]. https://arxiv.org/abs/2006.10214 Mediapipe Hands: On-device Real-time Hand Tracking

[6]. https://ijrpr.com/uploads/V2ISSUE5/IJRPR462.pdf Real-time vernacular sign language recognition using mediapipe and machine learning.

[7]. https://www.jstage.jst.go.jp/article/nolta/13/2/13_288/_article/-char/ja/

[8]. Japanese fingerspelling identification by using Mediapipe.
[9]. https://ijrpr.com/uploads/V2ISSUE5/IJRPR462.pdfReal-time vernacular sign language recognition using mediapipe and machine learning.
[10]. https://www.atlantis-press.com/article/125962696.pdf e Cite All 5 versions [11] [PDF] Applying Hand Gesture Recognition for User Guide Application Using Mediapipe