KINESICS - HAND FREE KEYBOARD CONTROL THROUGH GESTURES

AN INTERIM PROJECT REPORT

Submitted by

JERIN DAVIS (SHR21CS076)
JOEL JOY (SHR21CS080)
JOHN AMAL (SHR21CS081)
KEVIN PAULSON (SHR21CS084)

to

the APJ Abdul Kalam Technological University

in partial fulfillment of the requirements for the award of the Degree

of

Bachelor of Technology
in
Computer Science and Engineering

Department of Computer Science and Engineering

SAHRDAYA COLLEGE OF ENGINEERING AND TECHNOLOGY
KODAKARA, THRISSUR - 680684

NOVEMBER 2024

DECLARATION

We, undersigned, hereby declare that the project report "KINESICS - HAND FREE KEYBOARD CONTROL THROUGH GESTURES", submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under the guidance of Ms.Sruthy Sukumaran, Assistant Professor, Department of Computer Science Engineering. This submission represents our ideas in our own words and where ideas or words of others have been included; We have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to the ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

JERIN DAVIS
JOEL JOY
JOHN AMAL
KEVIN PAULSON

Kodakara
04-11-2024

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SAHRDAYA COLLEGE OF ENGINEERING AND TECHNOLOGY, KODAKARA, THRISSUR

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **KINESICS - HAND FREE KEYBOARD CONTROL THROUGH GESTURES** submitted by **JERIN DAVIS (SHR21CS076)**, **JOEL JOY (SHR21CS080)**, **JOHN AMAL (SHR21CS081)**, **KEVIN PAULSON (SHR21CS084)** to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering is a bonafide record of the project work carried out by them under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any other purpose.

GUIDE
Ms. Sruthy Sukumaran
Assistant Professor

PROJECT COORDINATORS
Mr Anil Antony, Mrs. Anly Antony M
Assistant Professor

HEAD OF THE DEPARTMENT
Dr. Manishankar S
Associate Professor

Kodakara

04-11-2024

ACKNOWLEDGMENT

We would like to express our immense gratitude and profound thanks to all those who helped us to make this project a great success. We express our gratitude to the almighty God for all the blessings endowed on us. We express our sincere thanks to our Executive Director **Rev.Fr.Dr. Anto Chungath** and Principal **Dr. Nixon Kuruvila** for providing us with such a great opportunity. We also convey our gratitude to our Head of the Department **Dr. Manishankar S** for having given us a constant inspiration and suggestion. We extend our deep sense of gratitude to our project coordinator **Mrs. Anly Antony M**, Assistant Professor of Computer Science & Engineering Department for providing enlightening guidance through the project. We can hardly find words to express our deep appreciation for the help and warm encouragement that we have received from our project guide **Ms. Sruthy Sukumaran** , Assistant Professor of Computer Science & Engineering Department for his wholehearted support. It was their encouragement that helped us to complete the project. We can hardly find words to express our deep appreciation of the help and warm encouragement that we received from our parents. We are extremely thankful and indebted to our friends who supported us in all aspects of the project work.

INSTITUTIONAL VISION

Evolve as a leading technology institute to create high caliber leaders and innovators of global standing with strong ethical values to serve the industry and society.

INSTITUTIONAL MISSION

Provide quality technical education that transforms students to be knowledgeable, skilled, innovative and entrepreneurial professionals. Collaborate with academia and industry around the globe, to strengthen the education and research ecosystem. Practice and promote high standards of professional ethics, good discipline, high integrity and social accountability with a passion for holistic excellence.

QUALITY POLICY

We at Sahrdaya are committed to provide Quality Technical Education through continual improvement and by inculcating Moral & Ethical values to mould into Vibrant Engineers with high Professional Standards.

We impart the best education through the support of competent & dedicated faculties, excellent infrastructure and colloboration with industries to create ambience of excellence.

DEPARTMENTAL VISION

To be a nationally recognized centre for quality education and research in diverse areas of computer science engineering with a strong social commitment.

DEPARTMENT MISSION

- Impart relevant technical knowledge,skills and attributes along with values and ethics.
- Enhance creativity and quality in research through project based learning environment.
- Mould Computer Science Engineering Professionals in synchronization with the dynamic industry requirements.
- Inculcate essential leadership qualities coupled with commitment to the society.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOS)

PEO1	Take up challenging careers in suitable corporate, business or educational sectors across the world, in multi-cultural work environment.
PEO2	To develop and design innovative and novel solutions to solve real life problems in the domain of computer science.
PEO3	To be responsible citizens with good team-work skills, competent leadership qualities and holistic values.

PROGRAMME SPECIFIC OBJECTIVES (PSOS)

PSO1	To mould students to understand, analyze and develop computer technologies in the areas such as algorithms, system software, multimedia, web design, big data analytics and networking for efficient design of computer-based systems of varying complexity.
PSO2	To enhance knowledge in the evolutionary changes in computing and apply standard practices and strategies in software project development using open-ended programming environments in order to deliver a quality product for business success and meet the future challenges.
PSO3	To enable students to employ modern computer languages, environments, and platforms for creating innovative career paths to be an entrepreneur with social accountability.

PROGRAMME OUTCOMES (POS)

PO1	Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3	Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4	Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12	Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

COURSE OBJECTIVES

- To apply engineering knowledge in practical problem solving.
- To foster innovation in design of products, processes or systems.
- To develop creative thinking in finding viable solutions to engineering problems.

COURSE OUTCOMES

The student will be able to

CO1	Identify academic documents from the literature which are related to her/his areas of interest.
CO2	Read and apprehend an academic document from the literature which is related to her/ his areas of interest.
CO3	Prepare a presentation about an academic document (Cognitive knowl edge.
CO4	Give a presentation about an academic document.
CO5	Prepare a technical report.

ABSTRACT

This project is a computer vision-based application that enables users to control keyboard functions using hand gestures, providing an alternative, hands-free interaction method. Leveraging OpenCV and Tkinter, the program captures live video from a webcam to recognize and register custom hand gestures. These gestures are stored in a dedicated directory and can be mapped to specific keyboard keys. Using the pyautogui library, recognized gestures automatically trigger corresponding keyboard actions, enabling seamless control over applications or system functions without direct physical contact. The project offers an intuitive graphical user interface, allowing users to register, view, and manage gestures in real time. This hands-free interface is particularly beneficial for accessibility, gaming, or scenarios where keyboard access is limited. By combining gesture recognition with automated keyboard simulation, the project demonstrates a unique approach to human-computer interaction through natural gestures.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ABBREVIATION	DESCRIPTION
HCI	Human Computer Interaction
AI	Artificial Intelligence
GUI	Graphical User Interfaces
RSI	Repetitive Strain Injuries
VR	Virtual Reality
AR	Augmented Reality
ML	Machine Learning
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
HMM	Hidden Markov Model
EM	Expectation Maximization
DTW	Dynamic Time Warping

CHAPTER 1

INTRODUCTION

In today's technology-driven world, the demand for intuitive user interfaces is rapidly increasing. Gesture recognition systems offer a natural way for users to interact with digital devices, transforming physical movements into digital commands. Utilizing advanced libraries such as Mediapipe for hand tracking, OpenCV for video processing, and Tkinter for creating user-friendly graphical interfaces, these systems capture and process hand gestures in real time.[**?**]. Furthermore, convolutional neural networks (CNNs) can enhance the accuracy and responsiveness of gesture detection. This integration of machine learning and computer vision techniques facilitates seamless control over various applications. The evolution of gesture-based technology opens up innovative possibilities across accessibility, gaming, and remote device control, highlighting its potential to create more immersive and engaging user experiences.

1.1 GENERAL BACKGROUND

In today's fast-paced digital landscape, the advancement of gesture recognition technology is transforming how users interact with devices. Hand gesture recognition systems enable intuitive control mechanisms, allowing users to execute commands through natural movements. This technology leverages computer vision and machine learning frameworks, such as Mediapipe and OpenCV, to accurately detect and interpret hand gestures in real time, providing a seamless user experience.

Unlike traditional input methods, gesture recognition eliminates the need for physical controllers, offering a more immersive and engaging interaction model. The integration of deep learning techniques enhances the accuracy and adaptability of these systems, enabling them to learn and recognize a variety of gestures with precision. However, challenges remain, particularly in terms of environmental factors, gesture variability, and real-time processing demands.

The rise of user-friendly interfaces, powered by libraries like Tkinter, facilitates the development of customizable applications that allow users to define their own gestures and associate them with specific commands. This flexibility paves the way for innovative applications across diverse fields, including gaming, virtual reality, and assistive technologies.

As this technology continues to evolve, it underscores the importance of developing robust and responsive gesture recognition systems that can effectively translate physical actions into digital commands, enhancing human-computer interaction and opening new avenues for user engagement.

1.2 PROBLEM DEFINITION

While traditional rehabilitation exercises can help patients recover hand mobility, they often suffer from low engagement and adherence rates due to their repetitive nature. This project aims to develop an engaging gesture-controlled rehabilitation game system with three specialized modes: Classic mode, which follows the traditional maze navigation approach for basic rehabilitation exercises; Drive mode, which simulates vehicle control to practice sustained gesture holds and smooth transitions; and Advanced mode, which combines complex gesture sequences with time-based challenges to promote advanced motor skill development. The system utilizes computer vision and deep learning techniques to recognize hand gestures through a standard webcam, eliminating the need for specialized hardware while making rehabilitation accessible from home. By gamifying rehabilitation exercises across these three progressive difficulty modes, the system seeks to increase patient engagement, provide measurable progress tracking, and improve rehabilitation outcomes while addressing the core challenge of making repetitive therapeutic exercises more enjoyable and effective

1.3 MOTIVATION

In the realm of rehabilitation technology, the integration of gesture recognition systems is redefining how patients engage in recovery exercises. These systems leverage advanced machine learning and computer vision techniques to enable intuitive control through natural hand movements, enhancing user interaction and motivation. For example, a rehabilitation game can track a patient's hand gestures using a standard webcam, translating movements into digital commands that facilitate various therapeutic activities. This real-time interaction not only promotes active participation but also adapts to individual progress, providing personalized feedback

that keeps users engaged.

As patients practice their exercises in a gamified environment, they can experience a sense of achievement and enjoyment, which is often lacking in traditional rehabilitation methods. The use of libraries like Mediapipe, OpenCV, and CNN allows for accurate gesture recognition, ensuring a seamless user experience. Furthermore, customizable settings enable users to modify gestures for different games, catering to varying skill levels and preferences. Such technology has profound implications for enhancing rehabilitation outcomes, making therapy more accessible and effective. By fostering an engaging and interactive recovery process, gesture recognition systems have the potential to improve adherence to rehabilitation protocols and ultimately facilitate faster recovery times, transforming the landscape of patient care.

1.4 OBJECTIVES

The objectives of this work are:

- To Develop an intuitive and user-friendly interface that enhances user interaction and facilitates seamless navigation through the game modes.
- To Analyze user performance data to identify trends in rehabilitation progress, enabling tailored feedback and recommendations for improvement.
- To Assess the effectiveness of gesture recognition accuracy using Mediapipe and OpenCV, ensuring reliable control and interaction within the game..
- To Implement customization features in the Advanced mode that allow users to create personalized gestures, catering to individual rehabilitation goals.
- To Evaluate user satisfaction and engagement through surveys and feedback mechanisms, ensuring the game meets the needs and expectations of its target audience.

CHAPTER 2

LITERATURE SURVEY

2.1 ADVANCEMENTS IN GESTURE-CONTROLLED SYSTEMS

This literature review explores recent advancements in gesture-controlled systems leveraging Machine Learning (ML)[1] It examines the role of ML algorithms in enabling real-time recognition and interpretation of hand gestures, emphasizing the importance of labeled datasets and preprocessing techniques. Various ML models, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), are discussed in the context of gesture recognition. Evaluation metrics such as accuracy, precision, and recall are highlighted for assessing model performance. The review also addresses deployment strategies, integration with hardware and software components, and mechanisms for model updating and adaptation. By synthesizing research findings, this review sheds light on the potential of gesture-controlled systems to offer users a fluid and responsive interface, facilitating natural interaction with digital devices across diverse applications and domains.

While KB and PM approaches request high designing exertion and profound information about the machine's way of behaving , ML does not, since it gains from machines' way of behaving and can manage high-layered information [8]. Additionally, ML approaches can perform nonlinear relations in information and are to some degree adaptable to exceptions. Another mechanized ML approach for constant shortcoming location and conclusion (RT-FDD) in discrete assembling machines (DMMs) is introduced and approved with two case studies: a Heater and a Pick And Spot recreated machines. The models created by the methodology introduced the most noteworthy mean F1 Scores and the least fluctuations among all 16 classifiers considered in the model determination process. Additional Trees and Arbitrary Woods are the classifiers chosen for the Heater and Pick And Spot Machines, with normal F1 Scores of 100 percent and 85%, individually. A huge improvement of

6% in the mean F1 Score is confirmed when consistent also, discrete factors are consolidated following this study's proposed approach analyzed with a dataset worked with just coordinated defer discrete occasions. The measurable contrast in the circulations is affirmed by applying a speculation test with a 95% certainty stretch. Additionally, in the element significance examination, six constant factors are recorded inside the ten most applicable elements, certifying the positive commitment of the mix of consistent factors and discrete occasions in the dataset. In both contextual investigations, the classifiers carried out with the Dad dataset show a F1 Score higher than 80% when just 20 examples of each issue class are accessible. The F1 Score is upgraded to more than 90% when 70 examples of each defective condition are accessible. These outcomes show the methodology's ability to analyze deficiencies when it is first sent as well as its ability to further develop each time another shortcoming event is identified.

2.2 MACHINE LEARNING TECHNIQUES FOR REAL-TIME GESTURE RECOGNITION

It delves into the realm of machine learning techniques applied within real-time gesture recognition systems. It meticulously explores the intricate processes involved in acquiring and preprocessing labeled datasets, with a keen focus on methodologies aimed at standardizing and extracting pertinent features from hand gestures. A diverse array of machine learning models, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), are rigorously evaluated for their efficacy in accurately recognizing and interpreting gestures.[2] Furthermore, the survey sheds light on the integration of IoT innovations within this domain. It elucidates on the practical manifestations of IoT technology, exemplified through sensor-based washing machines, IoT-powered air conditioning systems, and web-based power regulators. This technological advancement operates synergistically with web infrastructure, distributed computing, and sensor networks, forging a formidable synergy that underpins modern gesture recognition systems.

Distributed computing emerges as a pivotal technological paradigm, affording access to a vast array of computational resources including servers, storage, databases, and a plethora of application services over the internet. Its adoption accelerates innovation, enabling rapid progress compared to conventional technologies.

Within the discourse, cloud computing emerges as a linchpin for the efficient management of device data, encompassing historical fault data, control parameters, and predictive analytics for enhanced productivity. Clients anticipate reliable services coupled with elevated processing speeds, driving the widespread adoption of

this technology across various sectors including education, entertainment, healthcare, and government.

Moreover, the survey underscores the transformative potential of artificial intelligence (AI), particularly in the domains of machine learning (ML) and deep learning (DL). AI-driven automation revolutionizes system responses and processes, with companies increasingly investing in AI technologies, with projected expenditures reaching $57.6 billion by 2021$.

Despite the promising advancements facilitated by AI, the survey acknowledges the inherent challenges, such as reasoning, planning, learning, and perception. Nonetheless, machine learning algorithms, exemplified by Naïve Bayes, demonstrate commendable efficacy in fault prediction, leveraging past fault data to preemptively identify and alert users to potential future faults. This algorithmic approach serves as a practical and effective means of addressing challenges within gesture recognition systems.

2.3 INTEGRATION OF MACHINE LEARNING IN GESTURE RECOGNITION

embarks on a comprehensive exploration of machine learning methodologies integrated into real-time gesture recognition systems. It meticulously scrutinizes the intricate phases of data collection and preprocessing, emphasizing the significance of acquiring meticulously labeled datasets. Methodologies for data normalization and feature extraction from hand gestures are extensively examined, laying the groundwork for subsequent model training.[3]. A diverse spectrum of machine learning models, ranging from traditional approaches to state-of-the-art techniques, is evaluated for their efficacy in recognizing and interpreting gestures. Among these, Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) stand out for their ability to discern complex patterns and temporal dependencies inherent in gesture data.

Moreover, the survey illuminates the intersection of gesture recognition with IoT innovation, showcasing its practical applications in various domains. This includes sensor-based appliances like washing machines, IoT-enabled HVAC systems, and web-connected power regulators. These innovations harness the power of the internet, distributed computing, and sensor networks to enhance the functionality and efficiency of gesture recognition systems.

Distributed computing emerges as a cornerstone technology, providing access to an expansive array of computational resources over the internet. This facilitates rapid innovation and scalability, empowering gesture recognition systems to handle

large volumes of data with ease.

Cloud computing plays a pivotal role in managing and analyzing device data, encompassing historical fault records, control parameters, and predictive analytics. This enables the seamless integration of gesture recognition systems into diverse applications, ranging from education and entertainment to healthcare and government services.

Furthermore, the survey underscores the transformative impact of artificial intelligence (AI) on gesture recognition systems. AI-driven automation streamlines system responses and processes, leading to improved efficiency and accuracy. As companies increasingly invest in AI technologies, the landscape of gesture recognition continues to evolve, with projected expenditures reaching unprecedented levels.

Despite the remarkable advancements facilitated by AI, the survey acknowledges persistent challenges in reasoning, planning, learning, and perception. Nonetheless, machine learning algorithms, exemplified by Naïve Bayes, offer promising solutions for fault prediction, leveraging past data to anticipate and mitigate potential future issues. This pragmatic approach enhances the reliability and effectiveness of gesture recognition systems, paving the way for seamless human-machine interaction.

2.4 ADVANCE IN REAL-TIME GESTURE RECOGNITION

embarks on a thorough investigation of machine learning techniques integrated into real-time gesture recognition systems. It meticulously dissects the intricate processes of data collection and preprocessing, emphasizing the critical role of meticulously labeled datasets. Techniques for data normalization and feature extraction from hand gestures are extensively examined, laying the foundation for subsequent model training.[4] A diverse array of machine learning models, spanning from classical methodologies to cutting-edge approaches, is scrutinized for their efficacy in recognizing and interpreting gestures. Notably, Hidden Markov Models (HMMs) emerge as a prominent choice due to their ability to capture temporal dependencies inherent in gesture sequences.

Moreover, the survey delves into the convergence of gesture recognition with IoT innovations, showcasing practical applications across various domains. This includes sign language recognition systems, where IoT technologies facilitate seamless communication for individuals with hearing impairments. Additionally, IoT-enabled healthcare applications leverage gesture recognition for intuitive patient monitoring and diagnostics.

Distributed computing emerges as a cornerstone technology, providing scalable and efficient computational resources for processing large volumes of gesture data. This enables gesture recognition systems to operate in real-time, even in resource-constrained environments[5].

Cloud computing plays a pivotal role in managing and analyzing device data, facilitating predictive analytics and fault detection in gesture recognition systems. By leveraging cloud infrastructure, these systems can achieve unprecedented levels of scalability, reliability, and performance.

Furthermore, the survey underscores the transformative impact of artificial intelligence (AI) on gesture recognition systems. AI-driven algorithms, such as Deep Learning (DL), enable systems to autonomously learn and adapt to new gesture patterns, leading to improved accuracy and robustness.

Despite the remarkable advancements facilitated by AI, the survey acknowledges persistent challenges in fault detection and prediction. Machine learning algorithms, such as Dynamic Time Warping (DTW), offer promising solutions for identifying anomalies and deviations in gesture sequences, enhancing the overall reliability and effectiveness of gesture recognition systems.

In conclusion, the survey elucidates the multifaceted landscape of machine learning in gesture recognition, highlighting its potential to revolutionize human-computer interaction across various domains. Through ongoing research and development efforts, gesture recognition systems continue to evolve, offering enhanced capabilities and usability for diverse applications.

2.5 RECENT DEVELOPMENTS IN GESTURE RECOGNITION

It meticulously examines the intricate processes of data collection and preprocessing, emphasizing the paramount importance of meticulously labeled datasets. Techniques for data normalization and feature extraction from hand gestures are extensively scrutinized, laying the groundwork for subsequent model training.

A diverse range of machine learning models is evaluated for their efficacy in recognizing and interpreting gestures, with a focus on the application of Skin Colour Segmentation. This approach leverages color-based segmentation techniques to isolate skin regions in images or video frames, facilitating gesture recognition based on the movement and configuration of these segmented regions. [6] It meticulously examines the intricate processes of data collection and preprocessing, emphasizing the paramount importance of meticulously labeled datasets. Techniques for data normalization and feature extraction from hand gestures are extensively scrutinized, laying

the groundwork for subsequent model training.

A diverse range of machine learning models is evaluated for their efficacy in recognizing and interpreting gestures, with a focus on the application of Skin Colour Segmentation. This approach leverages color-based segmentation techniques to isolate skin regions in images or video frames, facilitating gesture recognition based on the movement and configuration of these segmented regions.

Furthermore, the survey underscores the transformative impact of artificial intelligence (AI) on gesture recognition systems. AI-driven algorithms, such as Motion Detection, enable systems to autonomously detect and track movements in real-time, leading to enhanced accuracy and responsiveness[7].

Despite the remarkable advancements facilitated by AI, the survey acknowledges persistent challenges in fault detection and prediction. Machine learning algorithms, such as Dynamic Time Warping (DTW), offer promising solutions for identifying anomalies and deviations in gesture sequences, thereby improving the overall reliability and effectiveness of gesture recognition systems.

In conclusion, the survey elucidates the multifaceted landscape of machine learning in gesture recognition, highlighting its potential to revolutionize human-computer interaction across various domains. Through ongoing research and development efforts, gesture recognition systems continue to evolve, offering enhanced capabilities and usability for diverse applications.

2.6 MACHINE LEARNING APPROACHES FOR GESTURE RECOGNITION SYSTEMS

It delves into the realm of machine learning techniques employed in real-time gesture recognition systems, with a particular focus on Motion Detection. This approach involves analyzing the motion patterns and trajectories of hand gestures to infer user intentions and commands accurately.

At the core of this methodology lies the utilization of computer vision techniques, such as OpenCV and MediaPipe, to capture and process video input from cameras in real-time. By extracting motion-related features from the captured video frames, such as optical flow or motion vectors, the system can discern the dynamic movements associated with different gestures.[8]. Furthermore, the survey scrutinizes the process of training machine learning models to recognize and interpret these motion-based features effectively. Techniques such as supervised learning, where labeled training data comprising motion samples for various gestures are used to train the model, are explored in depth. Additionally, unsupervised or semi-supervised learning approaches may be employed to handle unlabeled or partially

labeled data, further enhancing the system's versatility.

The integration of motion detection techniques with gesture recognition opens up a myriad of applications across diverse domains. In the realm of surveillance and security, motion-based gesture recognition can facilitate the identification of suspicious or unauthorized movements, thereby enhancing situational awareness and security measures[9].

Moreover, in the context of gaming and entertainment, motion detection enables immersive and interactive experiences, allowing users to control virtual environments and characters through natural hand movements. By leveraging the capabilities of OpenCV and MediaPipe, developers can create compelling gaming experiences that blur the lines between reality and virtuality.

Additionally, motion detection finds applications in healthcare, where it can be used for gesture-based control of medical devices or rehabilitation systems.[10],By analyzing the motion patterns of patients, healthcare professionals can monitor progress, assess mobility, and customize treatment plans effectively.

In conclusion, the sixth survey sheds light on the innovative fusion of motion detection techniques with machine learning for real-time gesture recognition. By harnessing the power of computer vision and advanced algorithms, gesture recognition systems can accurately interpret user gestures, opening up a world of possibilities for seamless human-computer interaction across various domains.

2.7 COMPARISON TABLE

Table 2.1 Comparison Table

PAPER	ALGORITHM	ACCURACY	APPLICATION
Gesture Controlled Systems using ML	ML	85%	AR and VR
Gesture Controlled System Using Template Matching Algorithm	Template Matching Algorithm	80.5%	Smart Home Automation
Gesture Controlled System Using HMM	HMM	90%	Sign Language Recognition
Gesture Controlled System Using DTW	DTW	89.73%	Healthcare
Gesture Controlled System Using Skin Colour Segmentation	Skin Colour Segmentation	97.60%	Gaming and Entertainment
Gesture Controlled System Using Motion Detection	Motion Detection	92.40%	Surveillance and Security

CHAPTER 3

PROPOSED SYSTEM

The proposed system aims to develop a robust and versatile gesture recognition platform leveraging machine learning techniques, particularly focusing on OpenCV and MediaPipe libraries, to enable intuitive human-computer interaction. This system will revolutionize the way users interact with digital devices by providing a seamless and natural interface based on hand gestures.

At the core of the proposed system lies the integration of computer vision algorithms and machine learning models to capture, process, and interpret hand gestures in real-time. OpenCV, a powerful open-source computer vision library, will be utilized for image and video processing tasks, such as hand detection, tracking, and feature extraction. MediaPipe, a cross-platform framework for building machine learning pipelines, will complement OpenCV by providing pre-trained models and pipelines for hand pose estimation and gesture recognition.

Data Collection and Preprocessing: The system will collect a diverse dataset of hand gestures, covering a wide range of movements and poses. Each gesture sample will be labeled and preprocessed to extract relevant features, such as hand keypoints, contours, and motion trajectories, using OpenCV and MediaPipe functionalities.

Model Training and Optimization: Machine learning models, such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs), will be trained on the preprocessed dataset to learn the patterns and characteristics of different gestures. Transfer learning techniques may be employed to fine-tune pre-trained models for gesture recognition tasks, thereby reducing training time and improving accuracy.

Real-time Gesture Recognition: The trained model will be deployed onto the gesture recognition system to operate in real-time. OpenCV and MediaPipe libraries will be utilized to capture video input from a camera, detect and track hand movements, and feed the processed data into the trained model for gesture recognition. The system will leverage the parallel processing capabilities of modern GPUs to achieve low-latency and high-throughput performance.

Gesture Mapping and Interaction: Upon recognizing a gesture, the system will map it to a specific function or action, enabling users to control digital devices, interact with software applications, or trigger predefined commands. Gesture mapping rules will be defined based on user preferences and application requirements, allowing for customizable and flexible interaction scenarios.

User Interface and Feedback: The system will incorporate a user-friendly interface to provide visual feedback and guidance to users during gesture interaction. Graphical elements, such as gesture overlays, instructional prompts, and feedback indicators, will be displayed on the screen to enhance user experience and facilitate gesture learning and recognition.

Adaptation and Personalization: The system will feature adaptive learning capabilities to continuously improve gesture recognition accuracy and adapt to individual user preferences and behaviors. Reinforcement learning techniques may be employed to adjust gesture mapping rules based on user feedback and performance metrics, ensuring a personalized and adaptive user experience.

In summary, the proposed gesture recognition system offers a comprehensive solution for intuitive human-computer interaction, leveraging the combined strengths of OpenCV and MediaPipe libraries. By harnessing the power of machine learning and computer vision, the system promises to revolutionize the way users interact with digital devices, opening up new possibilities for seamless and natural gesture-based interfaces across a wide range of applications and domains.

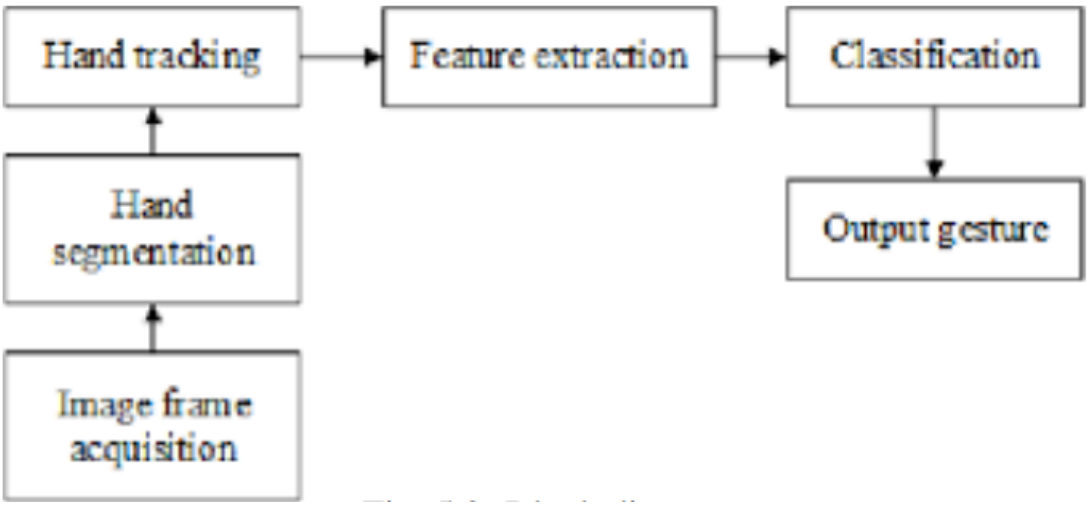

Fig. 3.1: Block diagram

CHAPTER 4

DESIGN PHASE

4.1 ALGORITHM

Algorithm 1 Hand Gesture Recognition

1: Import necessary libraries such as ‘math‘, ‘keyinput‘, ‘cv2‘, ‘mediapipe‘, and ‘tkinter‘.
2: Set up the MediaPipe Hands module for hand tracking with specified detection and tracking confidence levels.
3: Use OpenCV to start capturing video from the webcam.
4: Create a loop to continuously capture and process frames from the webcam.
5: Convert the frame to RGB and use the MediaPipe Hands module to detect hand landmarks.
6: Retrieve the coordinates of detected hand landmarks for further processing.
7: Recognize gestures by calculating distances or angles between landmarks.
8: Simulate keyboard inputs based on recognized gestures..
9: Display visual indicators of gestures on the frame.
10: If only one hand is detected, perform specific actions such as moving backward or stopping.
11: Exit the loop on a specific key press (e.g., ’q’).
12: Release resources and close OpenCV windows.

The algorithm enables gesture control through computer vision. Initially, video input is captured from a webcam using OpenCV. The MediaPipe Hands module detects and tracks hand landmarks in real-time. As frames are processed, landmark coordinates are extracted and analyzed to recognize gestures by calculating distances or angles. Recognized gestures are mapped to keyboard inputs using the ‘keyinput‘ library, allowing control over software or devices. Visual feedback is provided by overlaying gesture indicators on the video feed. The system processes input continuously until a specific key press (e.g., ’q’) ends the session.

CHAPTER 5

DESIGN DIAGRAMS

5.1 ARCHITECTURE DIAGRAM

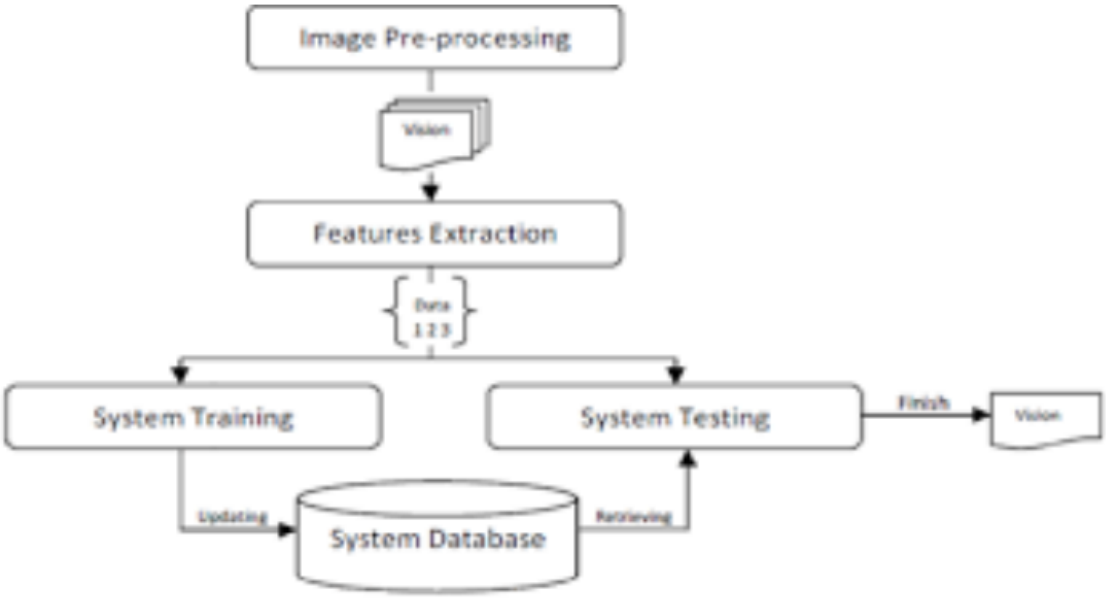

Fig. 5.1: Architecture diagram

Figure 5.1 represents the architecture for the hand gesture recognition project leveraging OpenCV and MediaPipe encompasses several key components. The input source, typically a webcam or camera, captures video feed for processing. The heart of the system lies in the Hand Gesture Recognition Module, where MediaPipe library comes into play for hand detection and landmark tracking. The User Interface component renders real-time video feed and graphical overlays to display detected gestures, allowing users to interact with digital devices or applications. This architecture provides a comprehensive overview of how video input is processed, gestures are recognized, and actions are executed based on the recognized gestures.

5.2 FLOWCHART

Fig. 5.2: Flowchart

Figure 5.2 represents the Flow chart of our proposed system. The flow chart outlines the sequential process of the hand gesture recognition system, starting with capturing video input from a webcam or camera, followed by preprocessing tasks such as color space conversion and noise reduction to enhance image quality. Subsequently, the preprocessed frames undergo hand detection and landmark tracking using the MediaPipe library to identify hand landmarks. Recognized gestures are displayed in real-time through a graphical user interface (GUI), enabling user interaction with digital devices or applications. Overall, the flow chart provides a structured representation of the sequential steps involved in capturing, preprocessing, detecting, classifying, and displaying hand gestures in real-time.

5.3 SEQUENCE DIAGRAM

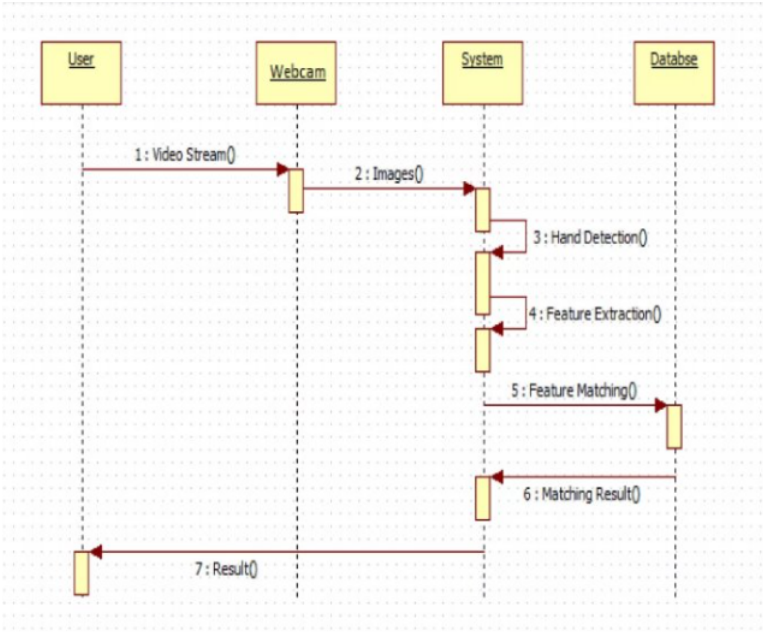

Fig. 5.3: Sequence diagram

Figure 5.4 represents the sequence diagram provides a visual representation of the interactions and communications between different components or objects within the hand gesture recognition system over time. It illustrates the sequence of messages exchanged between these components during the execution of a specific task or scenario, such as recognizing a hand gesture. Typically, the sequence diagram includes lifelines representing various entities involved in the process, such as the user interface, gesture recognition algorithm, input/output devices, and external systems. The arrows between lifelines depict the flow of messages or method calls between these entities, indicating the chronological order of their interactions. By following the sequence of events depicted in the diagram, stakeholders can gain insights into how the system functions and understand the interactions between its different elements. This helps in identifying potential bottlenecks, optimizing system performance, and ensuring that the system behaves as intended during real-world usage.

5.4 USE CASE DIAGRAM

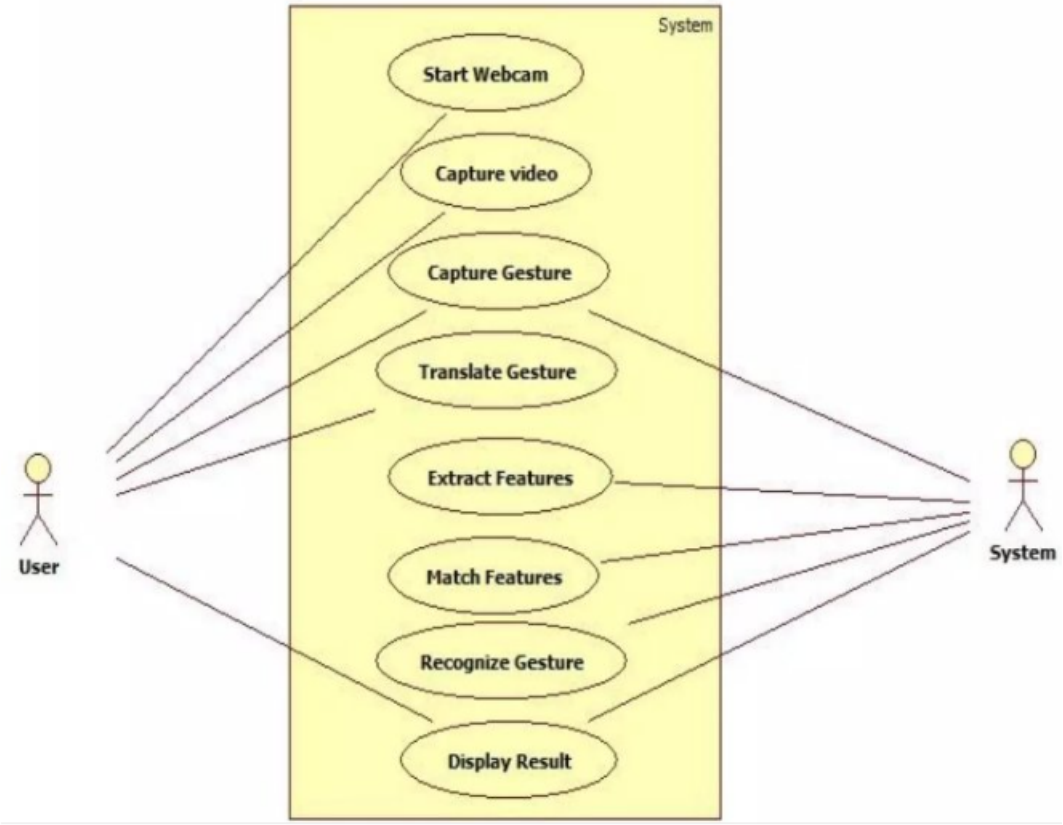

Fig. 5.4: Use case diagram

Figure 5.5 represents the use case diagram provides a high-level overview of the system's functionality by illustrating the various interactions between users (actors) and the system itself. Actors represent entities external to the system, such as users or other systems, while use cases represent the specific functionalities or actions that the system can perform. Each use case describes a particular interaction or scenario in which the system provides value to its users. The use case diagram visually depicts these relationships through actors connected to use cases by lines, indicating the interactions between them. This diagram helps stakeholders understand the scope of the system and its intended functionality by identifying the different ways in which users can interact with it and the corresponding system responses. It serves as a communication tool for eliciting and documenting user requirements, guiding system design, and ensuring that all necessary functionalities are accounted for in the system's development. Additionally, use case diagrams facilitate collaboration between stakeholders and development teams by providing a common understanding of the system's behavior and functionality, ultimately contributing to the successful design and implementation of the system.

5.5 DATA FLOW DIAGRAM

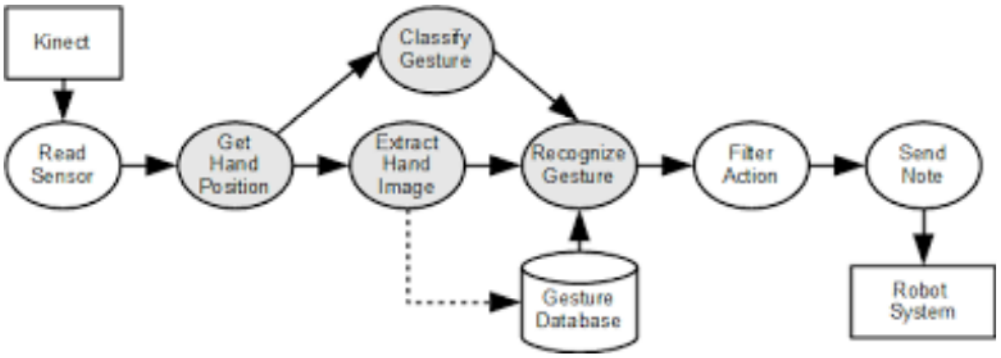

Fig. 5.5: Data Flow diagram

Figure 5.6 represents the Data Flow Diagram of our proposed system. The data flow diagram (DFD) for the project illustrates the flow of data within the system, depicting how information moves between various components and processes. At the center of the diagram is the main process, representing the core functionality of the system, which in this case is the hand gesture recognition module. The diagram shows the input data, which consists of video frames captured by the camera, flowing into the system. These video frames are processed by the gesture recognition algorithm, which identifies and interprets hand gestures in real-time. The output of the recognition process includes information about the detected gestures, which is then used to control the game or perform other relevant actions.

CHAPTER 6

REQUIREMENTS

6.1 HARDWARE REQUIREMNETS

1. Processor: AMD Ryzen 5 / Intel i5 (10th gen)
2. RAM: 8GB and above
3. Hard Disk: 512GB SSD

6.2 SOFTWARE RQUIREMENTS

1. Operating System: 64-bit Windows,Ubuntu or MAC
2. Languages: Python 3.8.8
3. Tools and Frameworks: Visual Studio Code

CHAPTER 7

CONCLUSION

In conclusion, the hand gesture recognition project using OpenCV and MediaPipe presents a significant advancement in human-computer interaction technology. By accurately detecting and interpreting hand gestures in real-time, the system offers a seamless and intuitive user experience, particularly in gaming applications where timing and precision are critical. The integration of machine learning techniques further enhances the system's capabilities, showcasing the potential of artificial intelligence in enhancing user interfaces.

Moreover, the project's versatility extends its applicability beyond gaming, with potential applications in sign language recognition, smart home automation, healthcare, and entertainment. Its user-friendly interface makes it accessible to a wide range of users, highlighting its potential for widespread adoption and impact.

Moving forward, the project lays the groundwork for continued innovation and exploration in gesture recognition technology. By open-sourcing the codebase and documentation, it fosters collaboration and knowledge sharing within the research community, paving the way for future advancements and applications.

Overall, the hand gesture recognition project represents a significant step forward in human-computer interaction technology, with the potential to revolutionize user interaction in various domains. As technology continues to evolve, this project serves as a testament to the power of innovation and collaboration in shaping the future of human-computer interaction.

REFERENCES

[1] K. H. Shibly, S. K. Dey, M. A. Islam, and S. I. Showrav, "Design and development of hand gesture based virtual mouse," in *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*. IEEE, 2019, pp. 1–5.

[2] S. A. Sabab, S. S. Islam, M. Hossain, and M. Shahreen, "Hand swifter: a real-time computer controlling system using hand gestures," in *2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT)*. IEEE, 2018, pp. 9–14.

[3] E. ul Haq, S. J. H. Pirzada, M. W. Baig, and H. Shin, "New hand gesture recognition method for mouse operations," in *2011 IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2011, pp. 1–4.

[4] Y. Shi, R. Taib, and S. Lichman, "Gesturecam: a smart camera for gesture recognition and gesture-controlled web navigation," in *2006 9th International Conference on Control, Automation, Robotics and Vision*. IEEE, 2006, pp. 1–6.

[5] P. Atre, S. Bhagat, N. Pooniwala, and P. Shah, "Efficient and feasible gesture controlled robotic arm," in *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*. IEEE, 2018, pp. 1–6.

[6] J. H. Gourob, S. Raxit, and A. Hasan, "A robotic hand: Controlled with vision based hand gesture recognition system," in *2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI)*. IEEE, 2021, pp. 1–4.

[7] F. Erden and A. E. Cetin, "Hand gesture based remote control system using infrared sensors and a camera," *IEEE Transactions on Consumer Electronics*, vol. 60, no. 4, pp. 675–680, 2014.

[8] Y. V. Parkale, "Gesture based operating system control," in *2012 Second International Conference on Advanced Computing & Communication Technologies*. IEEE, 2012, pp. 318–323.

[9] S. Thakur, R. Mehra, and B. Prakash, "Vision based computer mouse control using hand gestures," in *2015 International Conference on Soft Computing Techniques and Implementations (ICSCTI)*. IEEE, 2015, pp. 85–89.

[10] N. Mohamed, M. B. Mustafa, and N. Jomhari, "A review of the hand gesture recognition system: Current progress and future directions," *ieee access*, vol. 9, pp. 157 422–157 436, 2021.

Appendix A

Base Paper

HAND-REHA: Dynamic Hand Gesture Recognition for Game-based Wrist Rehabilitation

Farnaz Farahanipad*
Harish Ram Nambiappan*
The University of Texas at Arlington
Arlington, Texas, USA
farnaz.farahanipad@mavs.uta.edu
harishram.nambiappan@mavs.uta.edu

Ashish Jaiswal
The University of Texas at Arlington
Arlington, Texas, USA
ashish.jaiswal@mavs.uta.edu

Maria Kyrarini
The University of Texas at Arlington
Arlington, Texas, USA
maria.kyrarini@uta.edu

Fillia Makedon
The University of Texas at Arlington
Arlington, Texas, USA
makedon@uta.edu

ABSTRACT

Hand-gesture recognition systems have recently gained more popularity. Moreover, there is a growing interest in building games for other purposes apart from entertainment, such as education and rehabilitation. This paper focuses on developing a novel game-based system for wrist rehabilitation, called HandReha. The idea is to automatically recognize pre-defined hand gestures using a web-camera, so to control an avatar in a three dimensional maze run game. The pre-defined gestures are picked from a pool of well-defined gestures suitable for wrist rehabilitation. Deep learning techniques were utilized to perform real-time hand gesture recognition from the images. To evaluate the performance of the developed wrist rehabilitation system, a preliminary study with 12 healthy participants was conducted. The results showed that the developed wrist rehabilitation system is intuitive and engages the user, which is crucial for rehabilitation purposes.

CCS CONCEPTS

• **Human-centered computing → Human computer interaction (HCI)**; • **Applied computing → Computer games**; • **Computing methodologies → Image segmentation**.

*Both of the authors contributed equally and considered as the first author

PETRA '20, June 30-July 3, 2020, Corfu, Greece

ACM ISBN 978-1-4503-7773-7/20/06…$15.00
https://doi.org/10.1145/3389189.3392608

KEYWORDS

gesture recognition, convolutional neural networks, hand rehabilitation, gesture-based gaming

ACM Reference Format:
Farnaz Farahanipad, Harish Ram Nambiappan, Ashish Jaiswal, Maria Kyrarini, and Fillia Makedon. 2020. HAND-REHA: Dynamic Hand Gesture Recognition for Game-based Wrist Rehabilitation. In *The 13th PErvasive Technologies Related to Assistive Environments Conference (PETRA '20), June 30-July 3, 2020, Corfu, Greece.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3389189.3392608

1 INTRODUCTION

Rehabilitation is the recovery from the lost ability to control or coordinate a body part of the patient through a repetitive task. Based on worldwide statistics annually, more than 15 million people suffer from stroke, which is one of the main causes for people to lose their control over their body-part such as hands or feet [1]. Though physical rehabilitation can help patients regain the ability to control the affected body-parts for many decades, conventional rehabilitation exercises tend to not engage stroke patients due to tedious and repetitive nature [19]. Moreover, it is important to perform these rehabilitation exercises at home to improve their recovery. Therefore, a new area of research focuses on developing gamified approaches for various purposes other than entertainment or recreation. For instance, in [8] they designed and developed an automated Activate Test for Embodied Cognition (ATEC), which measures cognitive skills through physical activity.

Moreover, based on the initial testing in [27], they claimed that the interactivity aspect of the game had the potential which cause more enjoyment by the user and it can be used to engage a diverse population and clinical setting. Recently, Ramirez at al. [29] proved that how the design of an augmented reality (AR) increase the engagement in independent stroke rehabilitation. In 2018, Heng-Yi chen et al. designed

a smart suit, Heath Posture Protector (HP2), and 4 serious games for rehabilitation of office workers who usually suffer from neck and back pain [5]. Furthermore, there is a growing interest in research to utilize gesture recognition as a viable interface between humans and computers [10] [15]. Gesture-based interfaces can be an alternative to several physical peripherals that we use with our personal computers [2]. In [6], Chiang Wei Tan et al. introduced a human computer interaction system by using hand gesture recognition to apply into game-based rehabilitation application. However, in most of these studies, the focus was on design and development of 2D games or performing a limited variety of tasks such as going up or down or performing ordered-actions repeatedly.

In this paper, a game-based wrist rehabilitation system was developed, called HandReha. It recognizes pre-defined hand gestures performed in front of a web camera. The recognized gestures are used to control the actions of an avatar in a three-dimensional maze run game. The system runs on a personal computer equipped with a web-camera, which enables patients to perform their hand rehabilitation exercises at the comfort of their home. Additionally, as the hand recognition is vision-based, no additional sensors, such as Armbands [24], special suits [5], gloves, motion sensors or depth-cameras [16][27] are needed. The system uses images from the web-camera to recognize the gestures. The raw images are first pre-processed using image processing techniques followed by a background subtraction for hand segmentation. Subsequently, the segmented hand in the processed image is sent as an input to a three-layered convolutional neural network that classifies the type of gesture performed. Based on the classified gesture type, the avatar in the designed maze run game performs actions such as moving, rotating and shooting hostile drones in the 3D environment. Five different gestures were selected from a well-researched pool of medically-approved gestures suitable for hand therapies[28].

Despite the works in [6] and [32], where they restricted their focus on using gestures for simulating mouse and certain keyboard events for system interaction, we focused on using gestures to directly interact with the system and navigate the avatar in the 3D game. Besides, in many studies, they try to collect hand gestures through a variety of sensors, including gloves, electromagnetic or optical position and orientation sensors for the wrist. Yet wearing gloves or trackers, as well as other mentioned sensors, is uncomfortable and time-consuming approaches due to setup time or calibration process. Finally, due to our computer-vision-based interfaces, there are several notable advantages such as providing a non-intrusive approach where the hardware is commercially available at a lower cost compared to other approaches. Additionally, by applying the background subtraction method to segment hand we were able to achieve a robust gesture classifier model that can classify the performed gesture in a real-time manner with the highest accuracy of 98.8, which is in the same range of the state of the art methods.

The following sections of the paper are organized as follows. Section 2 describes the background and related work is done in gesture recognition. Section 3 explains the methodology and architecture of the entire system. Section 4 describes the gesture classifier,the data-set obtained for this system, followed by the achieved result in detail. Section 5 describes the game design and development. Section 6 explains the experimental user study and the devices used for experimentation and finally the results obtained through pre and post-survey results for our experiment. Section 7 discusses the conclusion and future work to be done in this system respectively.

2 BACKGROUND AND RELATED WORK

In recent years, as the percentage of older adults increased, the need for medical and rehabilitation dramatically raised. Furthermore, motivating game-based training improved therapy for people with physical impairments. As a result, research works that are performed in the area of developing games for rehabilitation purposes became extremely popular. Moreover, some research has been done in integrating gestures with a human-computer interaction system. [32] focused on integrating human gesture recognition in a human-computer interaction system by using gestures for certain mouse events such as mouse hold, mouse drag and mouse click along with certain keyboard events such as up and left keyboard press and used those gestures for certain scenarios such as playing angry birds game and working in Robot Operating System (ROS). In [4], Yassine Bouteraa et al. developed a customized augmented reality system for stroke rehabilitation.

We were inspired by the work done in [12] that implemented the use of gesture recognition for therapy. There has been compelling research on hand gestures that help in increasing a joint's range of motion or lengthen the muscle and tendons via stretching. Some of the popular motions like wrist ulnar and radial deviation gestures that are effective for improvements in hand motion are included as part of gestures in our system. This study has a focus on assisting patients to carry out rehabilitation with hand gesture recognition [27]. The process is supported by the patient playing a game on their computer to make the process interesting and make it as a rehabilitation session.

As noted in the introduction, the proposed model has two main parts for the vision-based hand gesture recognition; background subtraction along with hand detection and gesture classification. For the image processing part, we apply a background removal technique by taking a continuous average of the background at the start of the game, which then acts as a threshold throughout the game. Any object

or obstacle that forms a contour in the image after the background is subtracted is detected by the system. This detected contour is passed to our classifier which predicts one of the five possible gestures.

We utilized some naive approaches explained in [20][3] to apply background subtraction and extract relevant information about the hand. By doing so, we were able to remove unnecessary noise from the background making it easier for the neural network to perform the classification. However, hand detection and background removal using only an RGB camera can be relatively tough depending on the lighting conditions and objects present in the background. So, we used a fixed bounded box that would enclose the person's hand. Therefore, the recognition task is done faster, processing the smaller area, which is a key factor to have a real-time hand gesture recognition.

In deep learning, a convolution neural network (CNN, or ConvNet) is mostly applied to analyze visual imagery[22]. Ever since CNNs gained popularity back in 2012 after the revolutionary success of AlexNet on the public dataset ImageNet, they have come into the hype and been implemented in a lot of research areas. We wanted to apply similar methods to the datasets we collected after applying the processing techniques. Since CNNs are mostly applicable for image data, we first started with a simple 3-layer convolutional neural network in PyTorch [26] (a popular open-source deep-learning library). Besides, We implemented transfer learning with pre-trained models like ResNet50 and VGG16 [17] [30]. On comparing the results, the CNNs trained from scratch performed better than other methods for our use-case.

3 METHODOLOGY

In this paper, we design and develop a unique Maze run game such that the user can perform specific hand movements (according to their therapy treatment) in front of the web-camera and control the character in the game through the maze. As a result, doing the therapy through the game can motivate and interest the patient to continue their treatment without getting bored, all the user needs is a personal computer with web-camera. As soon as the user starts performing the predefined gestures in front of the web-camera, the avatar in the game will perform several actions such as starting the game, moving forward, rotating right/left and shooting hostile drones in the maze. The conceptual model for our proposed method is shown in figure 1.

In summary, the network consists of two stages, one for hand detection and the second stage for gesture classification. Once the gesture is shown by the candidate the hand is detected and then the gesture classification process is performed on the hand image. The gesture classification is explained in detail in the next section. Once the gesture classification is done, the classified gesture is sent to the interface module that interconnects the gesture classifier and the game. Based on the gesture, the corresponding action is performed by the avatar in the game.

4 BUILDING THE GESTURE CLASSIFIER

To build a real-time gesture recognition model, we propose a two-stage model. In the first stage, the user's hand region is segmented from the static background and then, in the second stage, these segmented hand gestures will be used as input to the Convolutional Neural Network (CNN), to be classified.

4.1. Hand detection and tracking

One of the main challenges and essential processes for information extraction in many computer vision applications is the detection and tracking of the moving objects in video streams or image sequences [14]. Generally, there are three approaches for this moving object detection task; Background subtraction, temporal referencing and optical flow [7]. Among these, we chose a background subtraction method because it is one of the most popular ones in motion object detection and it takes less computational time and space. In this method, the foreground mask for every frame is generated by subtracting the background frame from the current frame. In other words, it has two major steps. First, constructing a good statistical representation for the background which is robust to noise; second, building a statistical model for foreground object which represents the changes that take place in the current frame [25].

To build a background frame which is less affected by noise, we applied some pre-processing steps. First, all the captured frames were resized to 128 by 128, then flipped to avoid mirroring problem. Afterward, to simplify the process and decrease the processing time, we cropped the images from the pre-defined Region Of Interest (ROI window) around the hand in the original frames. Figure 2 displays the output images for this step.

Then, the cropped images are converted to grayscale to avoid the long processing time for color image analyses. The converted images are then followed by Gaussian blurring filter[13] to remove noise. Afterward, to create a smooth background image, we calculate the average frame for the first k stationary background frames where k equals 30 and it is selected empirically(clean plates).

$$AvBg = 1/n \sum_{i=1}^{n} BG(i) \tag{1}$$

where n is total number of frames, *AvBg* represents the smooth-average background image(clean plate) and *BG(i)* is i-th background frame. As a result, noise is suppressed as

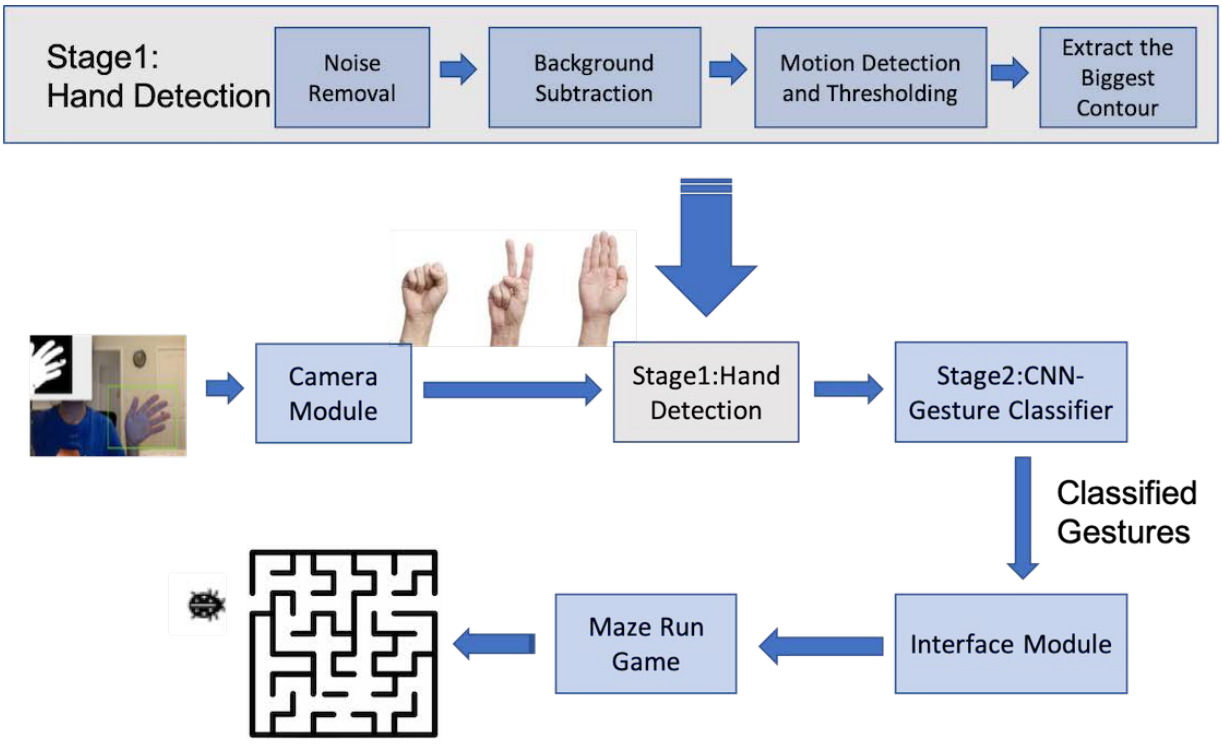

Figure 1: Overview of HandReha System

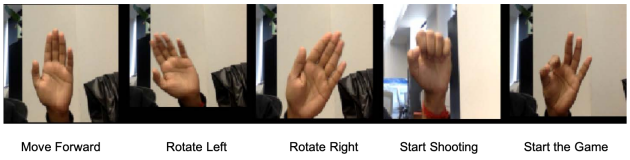

Figure 2: Input frames after applying resize, flip and crop operations

much as possible and the model will become more robust to changes in lightening. Second, to build a proper representation of the foreground object, we need to subtract the background and segment the image such that it has two components; hand as foreground segment and stationary background segment. To eliminate background there are two approaches: one with a known background called a clean plate and the other one is without known background[23]. In this study, we consider the first approach as we already created a clean plate in the first step.

Then, as the candidate starts performing desired gestures the absolute value of frames and clean plates are calculated and saved as different images.

$$diff(i) = |I(i) - AvBg| \tag{2}$$

Where i represents the current frame with and $AvBg$ as define in equation (1) is the clean plate image as a background frame. In other words, $diff$ is the absolute difference between these two frames. Afterward, we applied a threshold value to get the foreground object (hand) for the difference image.

$$segmented = \begin{cases} foreground, & \text{if } diff(x,y)(i) > \tau \\ background, & \text{otherwise} \end{cases} \tag{3}$$

τ is the threshold value that was selected empirically as 25. It means that all the pixels in $diff$(i-th) frame which has a value larger than τ will be assigned a value equal to one and the remaining pixels belong to the background segment with a value equal to zero. Although the extracted hand in our case, will be segmented from a black background after applying the threshold, the final segmented hand might have some missing pixels. This can happen due to many reasons such as using an average of the stationary frames for the background. However, in our case, this does not affect the segmented images. The resulting image is a black and white segmented image with a hand segment as white and background as black. Finally, these segmented images are resized into a fixed size keeping the same aspect of ratio and ready to be used as input to our CNN classifier in the second stage.

Figure 3: Output of the Hand Detection Stage

In figure 3 the final processed frames are depicted. Besides, we get the contours in these segmented images and consider the contour with maximum length as hand's contours. These contours are then displayed around the user's hands in each frame to evaluate the performance of the hand detection system.

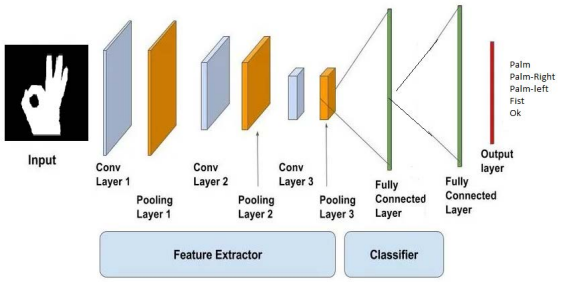

Figure 4: Overview of the CNN classifier

4.2. Hand gesture classifier model

In the second stage, to build the gesture classifier model we propose a 2D Convolution Neural Network (CNN) as a gesture recognition model. The architecture of the model is depicted in figure 4. The CNN feeds on binary images so that the color features do not affect the classifier. All the frames are first pre-processed, as explained in section 4.2. The pre-processing steps include resizing, converting to grayscale and applying Gaussian blurring, background subtraction, thresholding, and contour extraction. The model includes three 2D-convolutional layers for feature extraction, each followed by a max-pooling layer, two fully connected layers and a softmax layer for predicting gestures probabilities. The model classifies the gesture as the one with the highest probability. The input to this classifier is the processed frames from the hand detection stage which have the size of 128 by 128. Compared to other architecture such as RESNET50 [17]and VGG16 [30], the proposed CNN model only has 3 layers which makes the model much faster compared to above architectures.

4.3. HandReha dataset

After researching on which types of gesture have been used in hand therapy, we create our dataset of five gestures; "Fist", "Ok", "Open Palm", "PalmRight" and "PalmLeft" that are widely used in hand therapy procedure especially for wrist hand injuries[28]. A total of 7405 images are taken from three persons (two males, one female). We ask the participants to perform the gestures at a distance of 40 to 50 centimeters from the web-camera. Each class has around 1400 images as we tried to create a balanced data set. While collecting data all the mentioned pre-processing steps are applied to the captured images and finally, the processed images are saved into the separated folders. In our designed game, we assigned different tasks to each of these gesture classes; for instance, the character in the game should go straight if the model recognizes the "Palm" gesture or it should start shooting after recognizing "Fist" gesture.

4.4. Training the CNN gesture classifier

For the training process, we used a HAND-REHA dataset which includes 7405 images. We will explain the reason why we choose these special gestures in the Data collecting section. For each gesture, we used 0.1 percent of each class as the validation set and the remaining as the training set. The model uses input images of size 128 by 128. The loss function set as MSE and ADAM[21] as our optimizer. We trained the model for 5 epochs with processed images with various values of the learning rate, α, and realized α= 0.001 and image size as 128*128 provides the best accuracy for the classification task. Figure 5 and figure 6 show the loss and accuracy on training and validation set. The model achieves 100 and 98.8 percent accuracy for training and test set respectively.

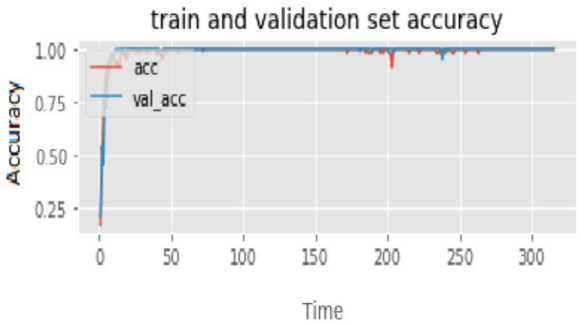

Figure 5: Train and validation accuracy

5 GAME DESIGN AND DEVELOPMENT

We developed a 3D maze run game where an avatar navigates inside the maze surrounded by hostile drones that shoot once the avatar is near to its location. The design for the main avatar, drones and the surrounding environment was done using Blender 2.8[11]. The game engine used for developing the game is Godot 3.1[9]. Godot is a free and open-source

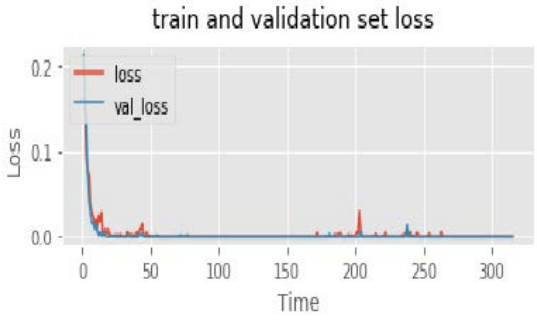

Figure 6: Train and validation loss

Figure 7: Navigation of avatar in the Game

game engine under the MIT License. The avatar, drones and several other aspects of the game designed in Blender were then imported into the Godot game engine. The stage for the game along with the avatar and drone navigation was implemented in the Godot game engine.

We built two levels in the game. The first level is a normal maze without any drones and the avatar only navigates around the maze. This level is built for the purpose of familiarizing the users with hand gestures by using them to navigate the avatar around the maze. The next level is the main level where the avatar navigates around the maze while drones are present in it. The avatar has to navigate around the maze and shoot the drones when they encounter them.

Figure 7 and 8 shows the navigation and shooting performed by the avatar in the game. Figure 9 shows the game played with gestures.

A video of the gesture-based game can be found in the following link: https://www.youtube.com/watch?v=V7X4CCbExmc.

Figure 8: Shooting in the Game

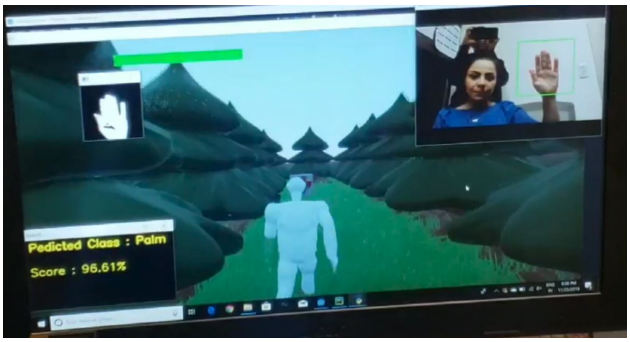

Figure 9: Game played with gestures

6 EVALUATION OF THE HAND-REHA SYSTEM

Twelve healthy participants from the Computer Science and Engineering department at the University of Texas, Arlington participated in the user study to evaluate the Hand-Reha system. . Of those twelve participants, six were male and six were female participants. Eight participants were aged between 25 and 30 years. Two participants were aged between 31 and 50 years. Two participants were aged between 18 and 24 years.

6.1 Hardware

For the experimentation, we used an Acer NITRO 5 Laptop with a Windows 10 64-bit Operating system. The laptop has an 8GB RAM along with an NVIDIA Geforce GTX 1050 Ti GPU. The laptop has a built-in camera which was used for gesture recognition. The Game runs on Godot Game Engine and the gesture recognition model runs on PyCharm IDE. Both the game and the model run on the same laptop.

6.2 Results of vision-based gesture classification

As mentioned in the previous section the accuracy for the validation set after 5 epochs reaches 98 percent. To test and evaluate the model we tried both testings offline and in real-time prediction. Although the model is not 100 percent accurate and has some flaws detecting gestures performed at far distance compared to training data, or in some cases the size

of the hand matters(it defines how far the hand is located from the camera), our model still shows promising results while doing real-time predictions compared to other studies. Considering that we only use a shallow CNN model(only 3 layers) compared to other states of the art models and the fact that we only have around 1400 images per class will support our efficiency of the proposed model. Besides, the other issue was how perfect the participants perform the gestures, as some of them were focused on the game so they did not perform the gestures well, especially in the first couple of minutes starting to play the game. Yet considering all the mentioned reasons, we proved that our gesture classification model has acceptable performance in terms of accuracy and processing time.

6.3 User study methodology

Before the study begins, each participant filled a pre-study survey form. The form contained questions that asked whether the participant had any previous experiences of hand pain or difficulty in hand movement and their preferences in the kind of therapy if they had any such pain.

After filling the pre-study survey form, we explained the user study process of our system in detail. Afterward, the participants played the first level of the game, a plain maze where the avatar can only move around, for making the participants to get familiarized with the gestures. Then they played the main game that included the avatar and the drones along with shooting capability for both drones and the avatar.

After 8 minutes of playing the game, the participants filled the post-study survey form. In one of the questions, participants are asked to compare the difficulty and feeling of pain playing the proposed 3D Maze game through gestures with playing similar Maze game with controllers. The form was used to get feedback from the participants regarding the efficiency of the system along with their opinion about using gestures for gaming and suggestions for future work.

6.4 Pre-study survey responses

In the pre-study survey form, apart from the name, age, and gender of participants we also asked a few questions regarding whether they faced any problem with hand movements and their preferences with treatment if they had any such problems. According to the participants' responses, no one faced any problem with respect to hand movement in the past. With respect to the type of therapy, in case of any pain or difficulty in hand movement, we gave two options in the survey questionnaire: Individual therapy and Group therapy. Everyone chose Individual therapy as their preferred type of therapy.

Table 1 shows the response for the survey question regarding the preferred place for receiving treatment in case of any pain. For this question, 5 participants prefer treatment either

Table 1: Table focusing on response regarding preferred place for receiving treatment

In case of pain where do you prefer to receive treatment?	
Options	Number of responses
Clinic	2
Home/Private Assistant	5
Hospital	4
Rehabilitation Center	0
Any Place is fine	3

at home or through a private assistant, 4 of the participants prefer to receive treatment in hospitals, 2 of the participants prefer clinic treatment and 3 of the participants were fine to receive treatment in any place.

6.5 Post-study survey responses

After conducting the experimentation we had a post-study survey that had a set of questions regarding comfort, difficulty, effectiveness, and excitement while playing the game with hand gestures. The answers are based on the 5-point Likert scale rating with a range from 1 ("least") to 5 ("most"). As shown in Figure 10 and 11 most of the participants gave positive responses in each categories.

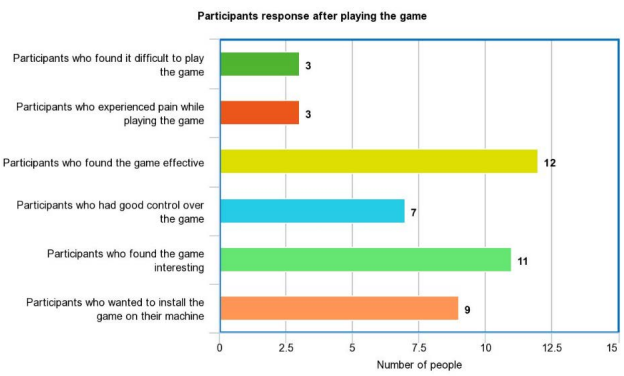

Figure 10: Graph representing participants feedback on HandReha

6.6 Discussion of the user study results

Overall, there was a good response from the participants where most of them reported that the hand gesture-based game was effective, exciting and comfortable. Most of the participants reported that they felt less pain and difficulty in controlling the game with gestures compared to traditional method such as using a controller. However, some participants reported that they had moderate pain and difficulty while playing the game. Three main reasons may have caused such an issue for some participants. First reason is that in the

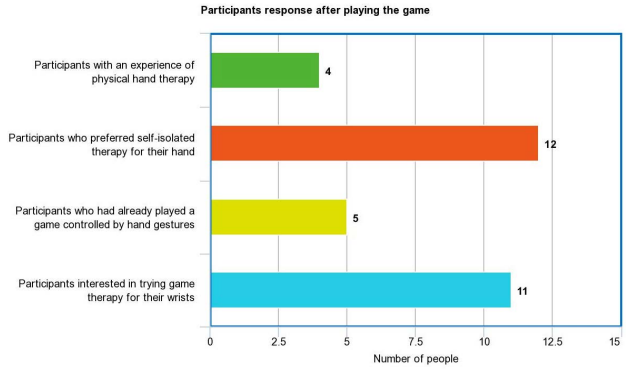

Figure 11: Graph indicating responses from the participants before using HandReha

classifier, for some participants, there was some difficulty in differentiating between "Open Palm" and "Ok" gestures, at certain situations, since both the gestures are shown at the same angle and after final processing "Open palm" and "Ok" gestures seem to be a bit similar. The second reason is that gesture recognition worked with better accuracy for participants with smaller hands when compared to participants with bigger hands which might be because of a shortage of dataset containing gestures from bigger hands. These two issues cause some difficulties for participants while navigating the avatar in the game. The third reason is that there was a time lag between gesture recognition and character movement in the game which is probably because of lower RAM capacity (8 GB) on the laptop. The third reason has been the cause of moderate pain for some participants as the time lag (a few seconds) between gesture recognition and character movement must have made the participants hold a particular gesture for an additional period at certain points in the game.

7 CONCLUSION AND FUTURE WORK

We have designed a game-based wrist rehabilitation system, which enables the user to control an avatar in a three-dimensional maze-run game using hand gestures. This is a unique and novel approach because the gestures are selected from a set of human gestures suitable for wrist rehabilitation and implemented to control a game built in a 3D environment as compared to previous works where most of the games designed for rehabilitation purposes are built in a 2D environment. Moreover, the game is built with an avatar performing more than one action and the gestures are assigned for every action performed by the avatar in the game. The gestures are implemented in such a way that they directly interact with the system. This is a different approach than the previous works where the game is built with an avatar performing only a single action and the gestures are implemented to simulate mouse or keyboard events through which the interaction with the system takes place. A user study was conducted to evaluate the developed system where the participants played the game with gestures. For now, the participants surveyed were from a healthy group of people with no problem with their hands and wrists. In future, we plan to include actual patients to test the efficiency of the system and to assess its feasibility in people with real injuries. This would be an extension of our current research. The results from the user study showed a good and favorable outcome where almost all the participants provided moderate to high ratings in terms of effectiveness and interest in playing the game with gestures.

Regardless, future research could continue to explore in improving the overall efficiency, accuracy, functionality, and usability. We plan to extend the HandReha system to be compatible with other everyday devices such as smartphones and tablets using Mobilenetes(a family of mobile-first computer vision models for TensorFlow) [18] or Inception-V3 [31]. Furthermore, the complexity of the game could be improved and better incentives for the players can be incorporated in the game rather than shooting a drone. Moreover, we plan to extend the HandReha dataset by including additional gestures and capturing data from a larger number of people who have problems in moving their hands and wrists to improve accuracy during gesture recognition and investigate the possible benefits of home rehabilitation.

REFERENCES

[1] 2008. The Internet Stroke Center. http://www.strokecenter.org/patients/about-stroke/stroke-statistics/

[2] Ashwin Ramesh Babu, Mohammad Zakizadeh, James Robert Brady, Diane Calderon, and Fillia Makedon. 2019. An Intelligent Action Recognition System to assess Cognitive Behavior for Executive Function Disorder. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*. IEEE, 164–169.

[3] Khawla Benabderrahim and Med Bouhlel. 2014. Detecting and tracking the hand to create an augmented reality system. *IOSR Journal of Computer Engineering* 16 (01 2014), 31–35. https://doi.org/10.9790/0661-16173135

[4] Yassine Bouteraa, Ismail Ben Abdallah, and Ahmed M Elmogy. 2019. Training of hand rehabilitation using low cost exoskeleton and vision-based game interface. *Journal of Intelligent & Robotic Systems* 96, 1 (2019), 31–47.

[5] Heng-Yi Chen, Tse-Yu Lin, Li-Yang Huang, An-Chun Chen, Yu-Chen Zheng, Hsing-Mang Wang, Shi-Yao Wei, and Yin-Yu Chou. 2018. HP2: Using Machine Learning Model to Play Serious Game with IMU Smart Suit. In *Proceedings of the 17th International Conference on Mobile and Ubiquitous Multimedia (MUM 2018)*. Association for Computing Machinery, New York, NY, USA, 397–402. https://doi.org/10.1145/3282894.3289731

[6] Chiang Wei Tan, Siew Wen Chin, and Wai Xiang Lim. 2013. Game-based human computer interaction using gesture recognition for rehabilitation. In *2013 IEEE International Conference on Control System, Computing and Engineering*. 344–349. https://doi.org/10.1109/ICCSCE.

2013.6719987

[7] Rita Cucchiara, Costantino Grana, Massimo Piccardi, and Andrea Prati. 2003. Detecting moving objects, ghosts, and shadows in video streams. *IEEE transactions on pattern analysis and machine intelligence* 25, 10 (2003), 1337–1342.

[8] Alex Dillhoff, Konstantinos Tsiakas, Ashwin Ramesh Babu, Mohammad Zakizadehghariehali, Benjamin Buchanan, Morris Bell, Vassilis Athitsos, and Fillia Makedon. 2019. An automated assessment system for embodied cognition in children: from motion data to executive functioning. In *Proceedings of the 6th international Workshop on Sensor-based Activity Recognition and Interaction*. 1–6.

[9] Godot Engine. 2019. Free and open source 2D and 3D game engine. https://godotengine.org/

[10] Sergio Escalera, Vassilis Athitsos, and Isabelle Guyon. 2017. Challenges in multi-modal gesture recognition. In *Gesture Recognition*. Springer, 1–60.

[11] Blender Foundation. 2019. Home of the Blender project - Free and Open 3D Creation Software. https://www.blender.org/

[12] Alana Da Gama, Thiago Menezes Chaves, Lucas Silva Figueiredo, Adriana Baltar, Meng Ma, Nassir Navab, Veronica Teichrieb, and Pascal Fallavollita. 2016. MirrARbilitation: A clinically-related gesture recognition interactive tool for an AR rehabilitation system. *Computer methods and programs in biomedicine* 135 (2016), 105–14.

[13] E. S. Gedraite and M. Hadad. 2011. Investigation on the effect of a Gaussian Blur in image filtering and segmentation. In *Proceedings ELMAR-2011*. 393–396.

[14] Sakher Ghanem, Ashiq Imran, and Vassilis Athitsos. 2019. Analysis of hand segmentation on challenging hand over face scenario. In *Proceedings of the 12th ACM International Conference on PErvasive Technologies Related to Assistive Environments*. 236–242.

[15] Shawn Gieser, Angie Boisselle, and Fillia Makedon. 2015. Real-Time Static Gesture Recognition for Upper Extremity Rehabilitation Using the Leap Motion. 144–154. https://doi.org/10.1007/978-3-319-21070-4_15

[16] R. Han, Z. Feng, T. Xu, C. Ai, W. Xie, K. Zhang, and J. Li. 2017. Multi-sensors Based 3D Gesture Recognition and Interaction in Virtual Block Game. In *2017 International Conference on Virtual Reality and Visualization (ICVRV)*. 391–392. https://doi.org/10.1109/ICVRV.2017.00091

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385 (2015). arXiv:1512.03385 http://arxiv.org/abs/1512.03385

[18] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[19] Ya-Xuan Hung, Pei-Chen Huang, Kuan-Ta Chen, and Woei-Chyn Chu. 2016. What do stroke patients look for in game-based rehabilitation: a survey study. *Medicine* 95, 11 (2016).

[20] SN Karishma and V Lathasree. 2014. Fusion of skin color detection and background subtraction for hand gesture segmentation. *International Journal of Engineering Research and Technology* 3, 2 (2014), 1835–1839.

[21] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. http://arxiv.org/abs/1412.6980 cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[22] Alex Krizhevsky. 2012. *AlexNet: Convolutional Neural Networks*. https://en.wikipedia.org/wiki/AlexNet

[23] Hsiang-Yueh Lai, Hao-Yuan Ke, and Yu-Chun Hsu. 2018. Real-time Hand Gesture Recognition System and Application. *Sensors and Materials* 30, 4 (2018), 869–884.

[24] Sih-Pin Lai, Cheng-An Hsieh, Teepob Harutaipree, Shih-Chin Lin, Yi-Hao Peng, Lung-Pan Cheng, and Mike Y. Chen. 2019. FitBird: Improving Free-Weight Training Experience Using Wearable Sensors for Game Control. In *Extended Abstracts of the Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts (CHI PLAY '19 Extended Abstracts)*. Association for Computing Machinery, New York, NY, USA, 475–481. https://doi.org/10.1145/3341215.3356258

[25] Mahmoud A Mofaddel and Walaa M Abd-Elhafiez. 2011. Fast and accurate approaches for image and moving object segmentation. In *The 2011 International Conference on Computer Engineering & Systems*. IEEE, 252–259.

[26] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. (2017).

[27] Rachel Proffitt, Marisa Sevick, Chien-Yen Chang, and Belinda Lange. 2015. User-centered design of a controller-free game for hand rehabilitation. *Games for health journal* 4, 4 (2015), 259–264.

[28] Harvard Health Publishing. 2019. 5 exercises to improve hand mobility. https://www.health.harvard.edu/pain/5-exercises-to-improve-hand-mobility-and-reduce-pain

[29] Edgar Rodríguez Ramírez, Regan Petrie, Kah Chan, and Nada Signal. 2018. A tangible interface and augmented reality game for facilitating sit-to-stand exercises for stroke rehabilitation. In *Proceedings of the 8th International Conference on the Internet of Things*. 1–4.

[30] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[31] Xiaoling Xia, Cui Xu, and Bing Nan. 2017. Inception-v3 for flower classification. In *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*. IEEE, 783–787.

[32] Pei Xu. 2017. A Real-time Hand Gesture Recognition and Human-Computer Interaction System. *CoRR* abs/1704.07296 (2017). arXiv:1704.07296 http://arxiv.org/abs/1704.07296

Appendix B

code

B.1 MAIN.PY

```
import tkinter as tk
import subprocess
import webbrowser
import os

def open_classic_mode():
    subprocess.Popen(['python',
    'classic/classic_mode.py'])
    minimize_after_delay()

def open_driving_mode():
    subprocess.Popen(['python',
    'driving/driving_mode.py'])
    minimize_after_delay()

def open_advanced_mode():
    subprocess.Popen(['python',
    'advanced/advanced_mode.py'])
    minimize_after_delay()

def minimize_after_delay():
    root.after(5000, root.iconify)
    # Minimize the window after 5 seconds

# Functions to open HTML files
def open_about():
```

```
    webbrowser.open('file://' +
    os.path.abspath("about_us.html"))

def open_contact():
    webbrowser.open('file://' +
    os.path.abspath("contact_us.html"))

def open_help():
    webbrowser.open('file://' +
    os.path.abspath("help.html"))

# Create the main window
root = tk.Tk()
root.title("Mode Selector")
root.attributes('-fullscreen',
True)  # Set to full screen
root.configure(bg="#9B7EBD")  #
Set background color

# Set font and button styles
heading_font = ("Helvetica", 24,
"bold")  # Font for the heading
button_font = ("Helvetica", 14, "bold")
link_font = ("Helvetica", 10,
"underline")  # Underlined font for links
button_bg = "#3B1E54"  # Green color
button_fg = "#EEEEEE"   # White text
button_active_bg = "#D4BEE4"

# Create heading at the top
heading_label = tk.Label(root,
text="Kinesics", font=heading_font,
fg="#ffffff", bg="#2e3f4f")
heading_label.pack(pady=20)

# Create buttons for each mode
classic_button = tk.Button(root,
text="Classic Mode",
command=open_classic_mode,
```

```
    font=button_font,
    bg=button_bg, fg=button_fg,
    activebackground=
     button_active_bg, width=20, bd=0, pady=5)
driving_button = tk.Button(root,
text="Driving Mode", command=open_driving_mode,
    font=button_font,
    bg=button_bg, fg=button_fg,
    activebackground=button_active_bg,
    width=20, bd=0, pady=5)
advanced_button = tk.Button(root,
text="Advanced Mode",
command=open_advanced_mode,
    font=button_font,
    bg=button_bg,
    fg=button_fg,
    activebackground=button_active_bg,
    width=20, bd=0, pady=5)

# Create the Exit button below the Advanced Mode button
exit_button = tk.Button(root,
text="Exit", command=root.quit,
font=button_font, bg="#d9534f",
fg="#ffffff",
activebackground="#c9302c",
width=20, bd=0, pady=5)

# Pack the buttons onto the window with spacing
classic_button.pack(pady=10)
driving_button.pack(pady=10)
advanced_button.pack(pady=10)
exit_button.pack(pady=10)

# Create a frame for
hyperlinks in the bottom-right corner
link_frame = tk.Frame(root,
bg="#2e3f4f")
link_frame.place(x=root
.winfo_screenwidth() - 220,
```

```
y=root.winfo_screenheight() - 30)

# Add hyperlinks to the frame
about_link = tk.Label(link_frame,
text="About Us", font=link_font,
fg="#00b0ff",
bg="#2e3f4f", cursor="hand2")
about_link.bind("<Button-1>",
lambda e: open_about())
about_link.pack(side="left",
padx=5)

contact_link = tk.Label(
link_frame, text="Contact Us",
font=link_font, fg="#00b0ff",
bg="#2e3f4f", cursor="hand2")
contact_link.bind("<Button-1>",
lambda e: open_contact())
contact_link.pack(side="left",
padx=5)

help_link = tk.Label(link_frame,
text="Help", font=link_font,
fg="#00b0ff", bg="#2e3f4f", cursor="hand2")
help_link.bind("<Button-1>",
lambda e: open_help())
help_link.pack(side="left",
padx=5)

# Run the application
root.mainloop()
```

B.2 STEERING CONTROL

```
import math

import keyinput
import cv2
import mediapipe as mp
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
mp_hands = mp.solutions.hands
font = cv2.FONT_HERSHEY_SIMPLEX
# 0 For webcam input:
cap = cv2.VideoCapture(1)
with mp_hands.Hands(
    model_complexity=0,
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5) as hands:
  while cap.isOpened():
    success, image = cap.read()
    if not success:
      print("Ignoring empty camera frame.")
      # If loading a video, use 'break' instead of 'continue'.
      continue
    # To improve performance,
    optionally mark the image as not writeable to
    image.flags.writeable = False
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    results = hands.process(image)
    imageHeight, imageWidth, _ = image.shape
    # Draw the hand annotations on the image.
    image.flags.writeable = True
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
    co=[]
    if results.multi_hand_landmarks:
      for hand_landmarks in results.multi_hand_landmarks:
        mp_drawing.draw_landmarks(
            image,
            hand_landmarks,
            mp_hands.HAND_CONNECTIONS,
```

```
            mp_drawing_styles.get_default_hand_landmarks_style(),
          mp_drawing_styles.get_default_hand_connections_style())
          for point in mp_hands.HandLandmark:
             if str(point) == "HandLandmark.WRIST":
                normalizedLandmark = hand_landmarks.landmark[point]
                pixelCoordinatesLandmark =
mp_drawing._normalized_to_pixel_coordinates
(normalizedLandmark.x,
                normalizedLandmark.y,imageWidth, imageHeight)
                try:
                  co.append(list(pixelCoordinatesLandmark))
                except:
                    continue
     if len(co) == 2:
          xm, ym = (co[0][0] + co[1][0]) / 2, (co[0][1] + co[1][1])
          radius = 150
          try:
              m=(co[1][1]-co[0][1])/(co[1][0]-co[0][0])
          except:
              continue
          a = 1 + m ** 2
          b = -2 * xm - 2 * co[0][0] * (m ** 2) + 2 * m *
              co[0][1] - 2 * m * ym
          c = xm ** 2 + (m ** 2) * (co[0][0] ** 2) +
co[0][1] ** 2 + ym ** 2 - 2 * co[0][1] * ym - 2 *
          co[0][1] * co[0][
              0] * m + 2 * m * ym * co[0][0] - 22500
          xa = (-b + (b ** 2 - 4 * a * c) ** 0.5) / (2 * a)
          xb = (-b - (b ** 2 - 4 * a * c) ** 0.5) / (2 * a)
          ya = m * (xa - co[0][0]) + co[0][1]
          yb = m * (xb - co[0][0]) + co[0][1]
          if m!=0:
            ap = 1 + ((-1/m) ** 2)
            bp = -2 * xm - 2 * xm * ((-1/m) ** 2) +
2 * (-1/m) * ym - 2 * (-1/m) * ym
            cp = xm ** 2 + ((-1/m) ** 2) * (xm ** 2) +
ym ** 2 + ym ** 2 - 2 * ym * ym - 2 *
ym * xm * (-1/m) + 2 * (-1/m) * ym * xm - 22500
            try:
```

```
            xap = (-bp + (bp **
            2 -4 * ap * cp) ** 0.5) / (2 * ap)
            xbp = (-bp -
            (bp ** 2 - 4 * ap * cp) ** 0.5) / (2 * ap)
            yap = (-1 / m)
            * (xap - xm) + ym
            ybp = (-1 / m)
            * (xbp - xm) + ym
          except:
              continue
        cv2.circle(img=image, center=(int(xm), int(ym)),
radius=radius, color=(195, 255, 62), thickness=15)
        l = (int(math.sqrt((co[0][0] -
        co[1][0]) ** 2 * (co[0][1]
        - co[1][1]) ** 2)) - 150) // 2
        cv2.line(image, (int(xa), int(ya)),
        (int(xb), int(yb)), (195, 255, 62), 20)
        if co[0][0] > co[1][0] and
        co[0][1]>co[1][1] and co[0][1] - co[1][1] > 65:
            print("Turn left.")
            keyinput.release_key('s')
            keyinput.release_key('d')
            keyinput.press_key('a')
            cv2.putText(image, "Turn left",
 (50, 50), font, 0.8, (0, 255, 0), 2, cv2.LINE_AA)
            cv2.line(image, (int(xbp), int(ybp)),
 (int(xm), int(ym)),
 (195, 255, 62), 20)
        elif co[1][0] > co[0][0] and
        co[1][1]> co[0][1] and co[1][1] - co[0][1] > 65:
            print("Turn left.")
            keyinput.release_key('s')
            keyinput.release_key('d')
            keyinput.press_key('a')
            cv2.putText(image, "Turn left",
(50, 50), font, 0.8, (0, 255, 0), 2, cv2.LINE_AA)
            cv2.line(image, (int(xbp), int(ybp)),
(int(xm), int(ym)), (195, 255, 62), 20)
        elif co[0][0] > co[1][0] and
```

```
        co[1][1]> co[0][1] and co[1][1] - co[0][1] > 65:
            print("Turn right.")
            keyinput.release_key('s')
            keyinput.release_key('a')
            keyinput.press_key('d')
            cv2.putText(image, "Turn right",
(50, 50), font, 0.8, (0, 255, 0), 2, cv2.LINE_AA)
            cv2.line(image, (int(xap),
            int(yap)), (int(xm), int(ym)), (195, 255, 62), 20)
        elif co[1][0] > co[0][0] and
        co[0][1]> co[1][1] and co[0][1] - co[1][1] > 65:
            print("Turn right.")
            keyinput.release_key('s')
            keyinput.release_key('a')
            keyinput.press_key('d')
            cv2.putText(image, "Turn right",
(50, 50), font, 0.8, (0, 255, 0), 2, cv2.LINE_AA)
            cv2.line(image, (int(xap),
int(yap)), (int(xm), int(ym)), (195, 255, 62), 20)
        else:
            print("keeping straight")
            keyinput.release_key('s')
            keyinput.release_key('a')
            keyinput.release_key('d')
            keyinput.press_key('w')
            cv2.putText(image, "keep straight",
(50, 50), font, 0.8, (0, 255, 0), 2, cv2.LINE_AA)
            if ybp>yap:
                cv2.line(image,
(int(xbp), int(ybp)), (int(xm), int(ym)), (195, 255, 62), 20)
            else:
                cv2.line(image,
(int(xap), int(yap)), (int(xm), int(ym)), (195, 255, 62), 20)
    if len(co)==1:
        print("keeping back")
        keyinput.release_key('a')
        keyinput.release_key('d')
        keyinput.release_key('w')
        keyinput.press_key('s')
```

```
        cv2.putText(image, "keeping back",
 (50, 50), font, 1.0, (0, 255, 0), 2, cv2.LINE_AA)
    cv2.imshow('MediaPipe Hands', cv2.flip(image, 1))
# Flip the image horizontally for a selfie-view display.
    if cv2.waitKey(5) & 0xFF == ord('q'):
      break
cap.release()
```

B.3 KEY INPUT CONTROL

```
import ctypes
keys = {
    "w":0x11,
    "a":0x1E,
    "s":0x1F,
    "d":0x20,
}
PUL = ctypes.POINTER(ctypes.c_ulong)
class KeyBdInput(ctypes.Structure):
    _fields_ = [("wVk", ctypes.c_ushort),
                ("wScan", ctypes.c_ushort),
                ("dwFlags", ctypes.c_ulong),
                ("time", ctypes.c_ulong),
                ("dwExtraInfo", PUL)]
class HardwareInput(ctypes.Structure):
    _fields_ = [("uMsg", ctypes.c_ulong),
                ("wParamL", ctypes.c_short),
                ("wParamH", ctypes.c_ushort)]
class MouseInput(ctypes.Structure):
    _fields_ = [("dx", ctypes.c_long),
                ("dy", ctypes.c_long),
                ("mouseData", ctypes.c_ulong),
                ("dwFlags", ctypes.c_ulong),
                ("time",ctypes.c_ulong),
                ("dwExtraInfo", PUL)]
class Input_I(ctypes.Union):
    _fields_ = [("ki", KeyBdInput),
                 ("mi", MouseInput),
                 ("hi", HardwareInput)]
class Input(ctypes.Structure):
    _fields_ = [("type", ctypes.c_ulong),
                ("ii", Input_I)]
def press_key(key):
    extra = ctypes.c_ulong(0)
    ii_ = Input_I()
    ii_.ki = KeyBdInput( 0, keys[key],
    0x0008, 0, ctypes.pointer(extra) )
```

```
    x = Input( ctypes.c_ulong(1), ii_ )
    ctypes.windll.user32.SendInput(1,
    ctypes.pointer(x), ctypes.sizeof(x))
def release_key(key):
    extra = ctypes.c_ulong(0)
    ii_ = Input_I()
    ii_.ki = KeyBdInput( 0, keys[key],
    0x0008 | 0x0002, 0, ctypes.pointer(extra) )
    x = Input( ctypes.c_ulong(1), ii_ )
    ctypes.windll.user32.SendInput(
    1, ctypes.pointer(x), ctypes.sizeof(x))
```