

# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

#### JNANA SANGAMA, BELAGAVI - 590018, KARNATAKA

##### A Project Phase-1 Report

**on**

#### CAMERA BASED INTERACTIVE COMPUTER FUNCTIONS USING HAND GESTURES

Submitted in partial fulfilment of the requirements for the VII Semester of a degree of **Bachelor of Engineering in Information Science and Engineering** of Visvesvaraya Technological University, Belagavi

##### Submitted by

**ABHISHEK N NAIRY (1AM20IS004) D N MEGHANA (1AM20IS0323) NAMRATHA P (1AM20IS058) SANCHALI A (1AM20IS083)**

**Dr. R AMUTHA**

**Associate Professor Dept. of ISE, AMCEC**

**Department of Information Science and Engineering**

# AMC ENGINEERING COLLEGE

18 Km, Bannerghatta Road, Bangalore - 560083

# 2023-24



AMC ENGINEERING COLLEGE

18 Km, Bannerghatta Road, Bangalore-560083

**DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING**

**CERTIFICATE**

Certified that the project work entitled: **CAMERA BASED INTERACTIVE COMPUTER FUNCTIONS USING HAND GESTURES** has been successfully completed by **ABHISHEK N NAIRY(1AM20IS004), D N MEGHANA** (**1AM20IS023**)**, NAMRATHA P(1AM20IS058)**

**and SANCHALI P(1AM20IS083)** all bonafide students of **AMC Engineering College, Bengaluru** in partial fulfilment of the requirements for the award of degree in **Bachelor of Engineering in Information Science and Engineering** of **Visvesvaraya Technological University, Belgaum** during the academic year **2023-2024**. The project report has been approved as it satisfies the academic requirements in respect of project work for the said degree.

**Dr. R. Amutha Dr. R. Amutha Dr. K Kumar**

Project Guide Professor and HOD Principal

Associate Professor Department of ISE AMCEC Department of ISE AMCEC

**External Viva**

**Name of the Examiners Signature with Date**

**1. 1.**

**2. 2.**

i.

**DECLARATION**

##### We, ABHISHEK N NAIRY (1AM20IS004), D N MEGHANA(1AM20IS023), NAMRATHA P (1AM20IS058) and SANCHALI A (1AM20IS083) students of VII

Semester**,** B.E. in Information Science and Engineering, AMC Engineering College hereby declare that the Project entitled **CAMERA BASED INTERACTIVE COMPUTER FUNCTIONS USING HAND GESTURES** has been carried out by us and submitted in partial fulfilment of the requirements for the VII Semester of the degree of Bachelor of Engineering in Information Science and Engineering of Visvesvaraya Technological University, Belgaum during the academic year 2023-2024

Place: Bengaluru

Date:

##### ABHISHEK N NAIRY (1AM20IS004) D N MEGHANA (1AM20IS023) NAMRATHA P (1AM20IS058) SANCHALI A (1AM20IS0083)

ii.

## ACKNOWLEDGMENT

At the very onset, we would like to place our gratitude on all those people who helped us in making this project a successful one.

Coming up, this project to be a success was not easy. Apart from the sheer effort, the enlightenment of our very experienced teachers also plays a paramount role because it is they who guided us in the right direction.

First of all, we would like to thank the **Management of AMC Engineering College** for providing such a healthy environment for the successful completion of project work.

In this regard, we express our sincere gratitude to the Director **Dr. K. R Paramahamsa** and the principal **Dr. R. Nagaraja,** for providing us all the facilities in this college.

We are extremely grateful to our Professor and Head of the Department of Information Science and Engineering, **Dr. R. Amutha**, for having accepted to patronize us in the rightdirection with all her wisdom.

We place our heartfelt thanks to **Dr. R. Amutha**, **Professor,** Department of Information Science and Engineering for having guided us for the project, and all the staff members of our department for helping us out at all times.

We thank **Dr. R. Amutha**, Project coordinator, Department of Information Scienceand Engineering. We thank our beloved friends for having supported us with all their strength and might. Last but not the least; we thank our parents for supporting and encouraging us throughout. We made an honest effort in this assignment.

**ABHISHEK N NAIRY (1AM20IS004) D N MEGHANA (1AM20IS023) NAMRATHA P (1AM20IS058) SANCHALI A (1AM20IS0083)**

## ABSTRACT

The recognition of Hand gestures has become a critical point as it is widely used in everyday applications. The challenge in this is to improve the recognition effect and develop a fast recognition method.

Glove and led-based methods involve external devices in detecting and interpreting hand gestures, making human-computer interaction less natural. So, different approaches have been used previously that use purely hand gestures in many systems based on human-computer interaction.

This system provides a more natural human-computer interaction; it must be made efficient processing speed of classifying the test data (images) from among the training data (database stored for gestures recognition).

This speed makes gesture recognition more effective and reliable to use as compared to previously proposed methods. In this research paper, a proposed system based on a camera based interactive wall display using bare hand gestures with efficient processing speed for controlling the speed of the mouse and other functions.

This system has three modules: one uses Genetic Algorithm and Otsu thresholding to identify the query images as the right or wrong gesture and perform the correct action in case of the proper motion, another module controls functions outside of PowerPoint files or Word documents, e.g., to open folders and go through drives, and the third module uses the convexity hull method for finding the number of fingers open in the user’s gesture and operates accordingly.

***Keywords:*** *Hand Gesture Recognition, Gesture Matching, Gesture Validation, Otsu’s Method, Genetic Algorithm***.**

## TABLE OF CONTENTS

**Sl. No. TITLE Page No.**

Certificate I

Declaration ii

Acknowledgement iii

Abstract iv

**1 Introduction 9**

1.1 Problem Statement 10

1.2 Objective of project 11

1.3 Scope of project 13

**2 Literature Review 14**

**3 Analysis 16**

3.1 Objectives 17

3.2 Purpose 17

3.3 Methodology 18

3.4 System Requirement Specification 20

3.4.1. Software Requirement Specification

3.4.2. Hardware Requirement Specification

3.4.3. Data Requirement

1. System Design 21

4.1 [Detailed Design](#_TOC_250004) 21

4.2 [Flow Chart](#_TOC_250003) 22

1. Implementation 25

5.1 [Hardware Implementation](#_TOC_250002) 25

5.2 [Software Implementation 2](#_TOC_250001)6

1. Modules of the project 29

6.1 Module 1 29

6.2 Module 2 30

6.3 Module 3 31

6.4 Module 4 34

6.5 Module 5 38

6.6 Module 6 44

1. System Testing 45

7.1 Types of testing 45

8 Conclusion and Future work 47

References 48

## LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **FIG. NO.** | **DESCRIPTION** | **PAGE NO** |
| Fig 3.1 | Architecture Design | 19 |
| Fig 4.1 | Detailed Design | 23 |
| Fig 4.2 | Data Flow Diagram | 24 |

**CHAPTER 1**

### INTRODUCTION

|  |
| --- |
| In recent years, the quest for more intuitive and natural interfaces between humans and computers has led to significant advancements in technologies enabling gesture-based interaction. Hand gesture recognition, in particular, has emerged as a critical area of research and development due to its potential to revolutionize the way we interact with digital devices and systems.  The recognition of hand gestures offers several advantages over traditional input methods such as keyboards or touchscreens. It allows for hands-free interaction, making it particularly useful in scenarios where users need to maintain mobility or when direct physical contact with devices is impractical or undesirable. Additionally, gesture-based interfaces have the potential to enhance accessibility for individuals with disabilities, providing alternative means of interaction that cater to diverse user needs.  One of the primary challenges in hand gesture recognition is to achieve high accuracy and efficiency in recognizing and interpreting a wide range of hand movements. This involves developing robust algorithms capable of detecting and classifying gestures accurately, often in real-time, to ensure seamless interaction between users and computers.  Various approaches have been explored to tackle the complexities of hand gesture recognition. These include glove-based systems, LED-based methods, and camera-based techniques. While each approach has its merits, there is a growing emphasis on camera-based systems that leverage computer vision algorithms to detect and interpret hand gestures without the need for wearable devices or external markers. This approach offers greater flexibility and naturalness in interaction, paving the way for more immersive and intuitive user experiences.  In this context, this paper proposes a novel system for hand gesture recognition based on camera input, aimed at enabling natural human-computer interaction. The system leverages deep learning algorithms, such as Convolutional Neural Networks (CNNs), combined with image processing techniques to accurately detect and classify hand gestures in real-time. By employing efficient processing methods and leveraging advances in machine learning, the proposed system seeks to enhance the speed, accuracy, and reliability of gesture recognition, making it suitable for a wide range of applications.  The remainder of this paper will delve into the details of the proposed system, including its architecture, algorithm modules, dataset gathering and preprocessing techniques, model training and validation procedures, real-world testing methodologies, and performance evaluation metrics. Through a comprehensive exploration of these aspects, we aim to demonstrate the potential of hand gesture recognition as a transformative technology for human-computer interaction and lay the foundation for future research and development in this exciting field. |
|  |
| **1.1 Problem Statement**  Despite advancements in hand gesture recognition technology, existing systems face several challenges that limit their effectiveness and usability. One of the primary issues is the lack of robustness in recognition accuracy, especially in diverse environmental conditions and with varying hand poses and orientations. Many systems struggle to achieve real-time processing capabilities, resulting in delays and inefficiencies in user interaction. Moreover, environmental factors such as background noise, clutter, and occlusions pose significant challenges to system robustness, affecting the reliability of gesture recognition. Additionally, existing systems often lack adaptability to different users, hand sizes, and movements, leading to inconsistent performance and usability issues. Integration with existing computer systems and applications also remains a challenge, requiring seamless compatibility and ease of use. Furthermore, security and privacy concerns associated with gesture-based interaction need to be adequately addressed to ensure user trust and data protection. Overall, improving the accuracy, speed, robustness, adaptability, integration, and security of existing hand gesture recognition systems is essential to enhance usability and user experience effectively. |
| **1.2 Objective of Project**  1. Develop Robust Recognition Algorithms: Design and implement efficient algorithms for accurately detecting, classifying, and interpreting hand gestures in real-time, ensuring high reliability and accuracy.  2. Create a Diverse and Annotated Dataset: Gather a comprehensive dataset of hand gesture images or videos, annotated with corresponding gesture labels, to facilitate supervised learning and model training.  3. Optimize Model Performance: Train machine learning models, such as Convolutional Neural Networks (CNNs), using the collected dataset, and fine-tune model parameters to achieve optimal performance in recognizing a wide range of hand gestures.  4. Implement Real-time Gesture Recognition System: Develop software modules for integrating the trained models into interactive applications, enabling real-time interpretation of recognized gestures and triggering appropriate computer actions.  5. Evaluate System Performance: Conduct thorough performance evaluation of the hand gesture recognition system under various environmental conditions and user interactions, assessing metrics such as accuracy, speed, and reliability.  6. Enhance User Experience: Improve the usability and user experience of the system by iteratively refining algorithms, optimizing system responsiveness, and incorporating user feedback to ensure seamless interaction with computer systems.  7. Ensure Compatibility and Scalability: Ensure compatibility of the system with common operating systems and software applications, and design it to be scalable for future expansion and integration with new technologies or functionalities.  8. Document Project Findings and Recommendations: Document the design, implementation, and evaluation processes of the project, and provide clear instructions and guidelines for system deployment, configuration, and usage.  By achieving these objectives, the hand gesture recognition project aims to develop a robust and versatile system capable of enabling natural and intuitive interaction between users and computer systems, thereby enhancing usability and user experience across various domains. |
| **1.3 Scope of Project**  1. Enhanced Human-Computer Interaction: Facilitates natural and intuitive interaction with computer systems through hand gestures, eliminating the need for physical input devices.  2. Improved Accessibility: Increases accessibility for users with mobility impairments or disabilities by providing alternative input methods that cater to diverse needs.  3. Efficient System Control: Enables efficient control of computer functions and applications, such as navigating presentations, controlling multimedia, and interacting with software interfaces.  4. Real-time Recognition: Offers real-time recognition of hand gestures, ensuring swift and responsive interaction with computer systems without perceptible delays.  5. Adaptability to Environmental Conditions: Capable of operating under varying lighting conditions and accommodating different hand sizes and orientations, ensuring robust performance in diverse environments.  6. Seamless Integration: Integrates seamlessly with existing computer systems and software applications, enhancing user experience without requiring significant hardware modifications.  7. Potential for Future Expansion: Provides a foundation for future research and development in gesture recognition technology, enabling the integration of additional gestures and functionalities.  8. Usability and User Experience Enhancement: Enhances overall usability and user experience by offering a more natural and engaging interaction paradigm, leading to increased productivity and satisfaction.  In summary, the hand gesture recognition project aims to revolutionize human-computer interaction by enabling natural, efficient, and accessible control of computer systems through hand gestures, thereby enhancing usability and user experience across various domains. |
|  |

# CHAPTER 2

**LITERATURE REVIEW**

Literature survey is mainly carried out in order to analyze the background of the current project which helps to find out flaws in the existing system & guides on which unsolved problems we can work out. So, the following topics not only illustrate the background of the project but also uncover the problems and flaws which motivated to propose solutions and work on this project. Some of the distinguished ones, which are relevant and carry basic information for this paper have been highlighted briefly. Following section explores different references that discuss about several topics related to hole detection and associated fastener identification.

**Rida Zahra a et. al., [1]** proposes a camera-based interactive wall display for natural hand gesture recognition, eliminating the need for external devices like gloves or LEDs.The system, employing Genetic Algorithm, Otsu thresholding, and convexity hull method in three modules, enhances processing speed, making gesture recognition more effective and reliable than previous methods.

##### Advantages:

* + 1. **Elimination of External Devices:** One significant advantage is the elimination of external devices such as gloves or LEDs. By solely relying on camera-based recognition, the system simplifies user interaction by not requiring users to wear any specialized gear. This enhances user comfort and convenience.
    2. **Enhanced Processing Speed and Accuracy**: The utilization of Genetic Algorithms, Otsu thresholding, and the convexity hull method contributes to improving processing speed and accuracy. Genetic Algorithms optimize the recognition process by refining solutions, while Otsu thresholding efficiently segments the hand from the background. Additionally, the convexity hull method helps in recognizing hand gestures accurately. These techniques collectively enhance the system's effectiveness and reliability compared to previous methods, ensuring more precise and faster recognition of natural hand gestures.

**Deval G. Patel et. al., [2]** proposes a robust system for real-time recognition of 36 static hand gestures in American Sign Language (ASL), using a novel pattern recognition method based on the SIFT algorithm and comparing its performance with PCA and Template Matching.

##### Advantages:

1. **Robustness to Variations**: SIFT (Scale-Invariant Feature Transform) is known for its robustness to variations in scale, rotation, illumination, and noise. In ASL, hand gestures might have variations in how they are formed, the angle at which they are presented, or lighting conditions. SIFT's ability to detect and match distinctive features despite these variations could make it more effective than PCA or Template Matching.
2. **Distinctive Feature Representation:** SIFT algorithm extracts and represents key features in an image regardless of its orientation or scale. This feature representation is based on local keypoints and their descriptors, enabling a more detailed and nuanced understanding of the hand gesture's characteristics. In contrast, PCA might struggle with capturing these fine details, and Template Matching might be limited by rigid templates that don't adapt well to variations.

**Viraj Shinde et. al., [3]** A proposes low-complexity algorithm and gesture recognition scheme aim to enhance human-machine interaction for real-time computer systems in the realm of gesture-controlled electronic products.

##### Advantages:

1. **Efficiency in Real-Time Processing**: Low complexity algorithms often translate to reduced computational demands. By implementing a low-complexity algorithm, this system might offer faster real-time processing. This efficiency is particularly beneficial in gesture-controlled electronic products, where responsiveness is crucial for a seamless user experience. Users would likely appreciate the system's quick and smooth response to their gestures without significant delays or lags.
2. **Improved User Experience:** Simplified algorithms can lead to easier implementation and potentially fewer errors. A gesture recognition scheme with lower complexity might be more robust against variations in gestures or environmental conditions (such as changes in lighting or hand positioning). This robustness can contribute to a more user-friendly experience, making the system more accessible and reliable for users interacting with electronic products through gestures.

# CHAPTER 3

**ANALYSIS**

#### OBJECTIVES

The goal is to introduce and highlight the significance of "camera-based interactive computer functions using hand gestures." This technology employs computer vision algorithms and cameras to capture and interpret hand movements, enabling users to control various applications seamlessly. The objective is to showcase its transformative impact on user experiences, emphasizing its applications in gaming, virtual reality, and smart devices.

The discussion also aims to underscore the broader applications of this technology, envisioning its integration into daily activities and industries. By eliminating the need for traditional input devices, it offers a more immersive and accessible computing experience. The primary objective is to convey the versatility of camera-based hand gesture technology and its potential to reshape human-computer interaction.

Ultimately, the narrative seeks to emphasize the significance of this technology in shaping the future of computing. As advancements refine gesture recognition algorithms, the objective is to convey the exciting possibilities for a computing landscape where users interact with digital devices in a natural and fluid manner, marking a significant evolution in the way we engage with technology.

#### PURPOSE

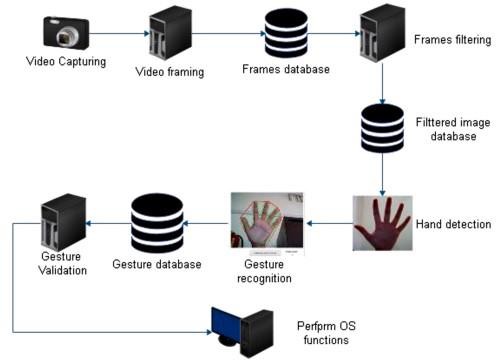
The purpose of discussing "camera-based interactive computer functions using hand gestures" is to provide a concise and informative overview of this innovative technology. The aim is to introduce readers to the principles and applications of the technology, emphasizing its transformative impact on human-computer interaction. By highlighting its ability to interpret hand gestures through computer vision algorithms and cameras, the purpose is to underscore its potential to redefine user experiences in various domains, including gaming, virtual reality,

and smart devices.

Additionally, the discussion seeks to convey the broader applications of this technology, envisioning its integration into daily activities and industries. The purpose is to showcase how the elimination of traditional input devices, coupled with a hands-free and immersive computing experience, opens up new possibilities. The overarching goal is to communicate the versatility and significance of camera-based hand gesture technology, pointing towards its role in meeting the growing demand for seamless and efficient interactions between humans and computers.

Ultimately, the purpose extends to fostering an understanding of the technology's potential trajectory, as advancements in gesture recognition algorithms continue. By providing insight into how users might interact with digital devices in the future, the discussion aims to illuminate the evolving landscape of interactive computing.

#### METHODOLOGY



##### Fig.3.1 System Architecture

The intricate process of camera-based hand gesture recognition unfolds as the system meticulously captures and records hand gestures in the form of single frames, each frame subjected to a meticulous preprocessing journey involving smoothing and resizing. Upon this preparatory phase, the hand detection module comes into play, receiving the now-refined image and deftly executing hand extraction. This crucial step precisely isolates the hand region, a key precursor to the subsequent stages of the system's operation. The image is meticulously groomed for the forthcoming gesture validation, a sophisticated process where the test image's gesture undergoes a meticulous comparison with a repository of dataset images, all meticulously represented as binary images. Upon a harmonious match being discerned, the system seamlessly executes the corresponding operating system operation, marking the fruition of a seamless human-computer interaction.

Facilitating this intricate choreography is a camera designed with precision and adaptability. Compliant with the USB 2.0 Interface and MS Windows XP Service Pack 3, this technological marvel boasts features such as autofocus, up to 2-megapixel resolution, and video capture capabilities extending up to 1600 × 1200 pixels, ensuring a high-definition quality experience. The camera's prowess extends to its handling of diverse lighting conditions, deftly employing autofocus to achieve optimal results. In navigating the nuances of lighting, it becomes apparent that dimly lit environments may introduce undesirable elements such as noise and blur, underscoring the importance of an adaptive system.

Furthermore, the system acknowledges the preferential treatment of bright light over direct sunlight, showcasing an awareness of the interplay between environmental factors and technological functionality. The system's architectural prowess unfolds into two major modules – Gesture Validation and Cursor Control – each playing a pivotal role in orchestrating the seamless dance between user gestures and system responsiveness. The intricate flowchart delineating the steps of Gesture Validation serves as a testament to the systematic and thoughtful design that underpins the entire framework, elevating user interaction to a realm of precision and finesse.

* 1. **SYSTEM REQUIREMENT SPECIFICATION**

**HARDWARE REQUIREMENTS**

* + 1. Processor : Core i3/i5/i7
    2. RAM : 2-4 GB
    3. Hard Disk : 500 GB
    4. Input device : Standard Keyboard and Mouse
    5. Output device : High Resolution Monitor

### SOFTWARE REQUIREMENTS

* + - * Platform: Windows XP/7/8/10
      * Coding Language: Python

# CHAPTER 4

#### 4.1. DETAILED DESIGN

TRAINING DATASETS

RAW DATA

**SYSTEM DESIGN**

TESTING DATASETS

DATA PREPROCESSING

DATA PREPROCESSING

ML ALGORITHMS

BEST MODEL

##### Fig: 4.1 Detailed Design



In designing a hand gesture detection system for computer interactive functions, the process begins with a meticulous analysis of the specific interactive requirements. Subsequently, a diverse and annotated dataset is collected, undergoing preprocessing steps like cleaning, normalization, and augmentation to enhance its quality. The data representation is then determined, choosing between image frames, depth maps, or keypoint coordinates. The model selection involves choosing a deep learning architecture, with the option of leveraging transfer learning for improved performance. Following model training, a comprehensive gesture mapping is established, linking recognized gestures to distinct computer interactive functions. The integration of real-time detection into the interactive application is implemented, and a separate validation dataset is utilized to fine-tune the model and assess its accuracy. Continuous user testing and feedback collection inform adjustments to parameters like probability thresholds, ensuring a balance between precision and recall. The adaptability of the model to diverse users and its capacity to handle variations in hand sizes and shapes are crucial

considerations. The implementation of a feedback mechanism, security measures to prevent unintended actions, and comprehensive documentation for developers and users contribute to a well-rounded system. Continuous improvement through periodic updates and scalability considerations further enhance the robustness of the hand gesture detection system, providing a seamless and user-friendly interactive experience.

**4.2 FLOWCHART**

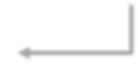
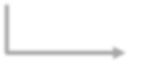
VIDEO CAPTURING

FRAMES

FRAME PROCESSING

HAND DETECTION

HAND EXTRACTION



EXIT

RIGHT CLICK

DOUBLE CLICK

SINGLE CLICK

##### Fig 4.2 Data Flow Diagram



GESTURE VALIDATION



**Video Capturing and Frames Formation**

A real-time capturing device like a webcam or external USB-connected camera is used for recording the video. Video is converted into frames. Each frame is captured by an interval of 2 seconds. Each frame is captured and processed side by side.

##### Frames Preprocessing

Frames are pre-processed before applying recognition techniques. The frame is resized to the size of the dataset images. Filters are applied to remove noise from the image. And finally, a clear image is obtained

##### Hand Detection

For hand detection, firstly the contrast function is applied to enhance skin colour, the skin detection algorithm is then executed. From the captured frame, skin is detected for a specific YGBR value. The system is then trained to detect lighter to darker skin colours. After skin detection, the entire portion other than the skin areas is removed from the image.

##### Hand Extraction

The hand extraction is performed based on the connected component algorithm. The connected component algorithm takes the largest connect part from the image which is always the hand region. The hand region is cropped out of the image in this manner, and the rest of the image is eliminated. Now the obtained image is ready for validation.

**Hand Gesture Validation**

After extracting the hand area from the image, it undergoes a comparison with all dataset images using the Euclidean distance method.

Otsu's method is employed to convert the user's hand image into a binary image, ensuring a uniform background

All dataset images are stored in binary form, and the Euclidean distance between the user's hand gesture image and dataset images is calculated by subtracting corresponding pixels.

Euclidean distance is computed using the formula:

##### Euclidean = (float)(Math.Sqrt((clr − clr1) \* (clr − clr1)))

where clr is the captured frame pixel, and clr1 is the dataset image pixel. The dataset image with the smallest Euclidean distance to the user's hand gesture becomes the final match, triggering the corresponding operation .

The dataset includes similar gesture images from various angles for optimal real-time matching

# CHAPTER 5

## IMPLEMENTATION

### Hardware Implementation

Hardware implementation for a hand gesture recognition system typically involves the following components:

1. Camera: An appropriate camera is essential for capturing hand gestures. Depending on the application and environment, you may choose a webcam, depth-sensing camera (such as Kinect), or specialized cameras designed for computer vision applications.

2. Processor: A powerful processor is required to handle the image processing and machine learning tasks involved in gesture recognition. An Intel Core i5 or higher processor is recommended for efficient processing.

3. Memory (RAM): Sufficient RAM is necessary to store and manipulate image data, especially during real-time processing. A minimum of 16GB RAM is recommended to ensure smooth operation.

4. Storage: Adequate storage space is required for storing datasets, models, and application code. A hard disk with at least 500GB capacity is recommended.

5. Graphics Processing Unit (GPU) (optional): While not mandatory, a GPU can significantly accelerate the training and inference processes, especially when using deep learning models like Convolutional Neural Networks (CNNs). If GPU acceleration is desired, a dedicated GPU with CUDA support from NVIDIA would be suitable.

6. Microcontroller or Single Board Computer (optional): Depending on the application, you may need a microcontroller or single board computer (such as Arduino or Raspberry Pi) to interface with external hardware or control actuators based on recognized gestures.

7. Connectivity: Ensure that the hardware components are properly connected and can communicate effectively. This may include USB connections for cameras, HDMI for displays, and network connections for data transfer or remote control.

8. Power Supply: Provide a stable power supply to all components to ensure uninterrupted operation.

9. Enclosure: Depending on the deployment environment, you may need to house the hardware components in a protective enclosure to prevent damage and ensure longevity.

10. Additional Sensor:(optional): Depending on the specific application requirements, additional sensors such as accelerometers or gyroscopes may be integrated to complement gesture recognition capabilities or enhance system functionality.

11. Display and User Interface (optional): If the system includes a user interface for feedback or interaction, a display (such as a monitor or touchscreen) and input devices (such as keyboard, mouse, or touchscreen) may be necessary.

12. Networking Components (optional): For networked applications or remote control capabilities, networking components such as Ethernet adapters or Wi-Fi modules may be required.

Once you have assembled the necessary hardware components, you can proceed with installing the required software, developing the gesture recognition algorithms, and integrating the hardware and software components to create a functional hand gesture recognition system. Testing and iterative refinement will help ensure that the system meets the desired performance and usability criteria.

### Software Implementation

For implementing the hand gesture recognition system described in the abstract, you would need a combination of software tools and libraries. Here's a high-level overview of the software components required for each stage of the implementation:

1. Data Collection and Preprocessing:

- Python: Use Python along with libraries like OpenCV for capturing images or videos of hand gestures and performing preprocessing tasks such as resizing, normalization, and data augmentation.

- OpenCV: OpenCV provides extensive functionalities for image and video processing, making it suitable for tasks like gesture annotation, background subtraction, and image filtering.

- NumPy: NumPy can be used for numerical computations and array manipulation, which are common in data preprocessing tasks.

2. Model Building and Training:

- TensorFlow or PyTorch: These deep learning frameworks offer tools for building and training neural network models, including CNNs for image classification tasks like hand gesture recognition. You can choose either TensorFlow or PyTorch based on your preference and familiarity.

- Keras (if using TensorFlow): Keras is a high-level neural networks API that runs on top of TensorFlow, providing a user-friendly interface for building and training deep learning models.

- Scikit-learn: Scikit-learn can be used for tasks such as splitting the dataset into training and testing sets, encoding labels, and evaluating model performance using metrics like accuracy and F1-score.

3. Real-Time Prediction and Integration:

- Python: Use Python for real-time prediction of hand gestures using the trained CNN model. You can integrate this functionality with other applications or systems using appropriate communication protocols or APIs.

- OpenCV: OpenCV can be used for real-time video processing and gesture recognition from webcam feeds or video streams.

- GUI Frameworks (e.g., Tkinter, PyQt): If you plan to develop a graphical user interface (GUI) for your application, you can use GUI frameworks like Tkinter or PyQt to create interactive interfaces for users to interact with the system.

4. Additional Libraries and Tools:

- Matplotlib or Seaborn: These libraries can be used for data visualization and analysis, which may be helpful during the model development and evaluation stages.

- Jupyter Notebook: Jupyter Notebook provides an interactive environment for developing and testing code, making it useful for prototyping and experimenting with different approaches.

5. Deployment and Optimization:

- Optimization Libraries (e.g., TensorFlow Lite, ONNX Runtime): If you plan to deploy the model on resource-constrained devices or platforms, optimization libraries like TensorFlow Lite or ONNX Runtime can help optimize model inference speed and memory footprint.

- Containerization Tools (e.g., Docker): Containerization tools like Docker can be used to package the application and its dependencies into lightweight, portable containers, facilitating deployment across different environments.

By leveraging these software components and tools, you can implement the hand gesture recognition system described in the abstract and customize it according to your specific requirements and use cases.

# CHAPTER 6

# MODULES OF THE PROJECT

# MODULE 1

# #............................................Automatic Capture the images...............................................#

# import os

# import time

# import cv2

# import numpy as np

# vc = cv2.VideoCapture(0)

# pic\_no = 0

# total\_pic = 1000

# flag\_capturing = False

# path = r'Dataset\up'

# while(vc.isOpened()):

# rval, frame = vc.read()

# frame = cv2.flip(frame, 1)

# cv2.rectangle(frame, (300,300), (100,100), (0,255,0),0)

# #cv2.rectangle(frame, (600,300), (300,65), (0,255,0),0)

# 

# cv2.imshow("image", frame)

# 

# crop\_img = frame[100:300, 100:300]

# #crop\_img = frame[300:600, 65:300]

# 

# 

# if flag\_capturing:

# 

# pic\_no += 1

# 

# save\_img = cv2.resize( crop\_img, (50,50) )

# save\_img = np.array(save\_img)

# cv2.imwrite(path + "/" + str(pic\_no) + ".jpg", save\_img)

# 

# 

# keypress = cv2.waitKey(1)

# 

# if pic\_no == total\_pic:

# flag\_capturing = False

# break

# 

# if keypress == ord('q'):

# break

# elif keypress == ord('c'):

# print('c pressed')

# flag\_capturing = True

# vc.release()

# cv2.destroyAllWindows()

# cv2.waitKey(1)

# MODULE 2

# #..................................................Preprocessing of images............................................#

# import os

# import time

# import cv2

# import numpy as np

# import matplotlib.pyplot as plt

# path = 'Dataset/'

# path2 = 'Preprocessed/Train/'

# gestures = os.listdir(path)

# print(gestures)

# for ix in gestures:

# print(ix)

# images = os.listdir(path + ix)

# os.mkdir(path2 + ix)

# for cx in images:

# print(cx)

# img\_path = path + ix +'/' + cx

# img = cv2.imread(img\_path)

# grey = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

# thresh = cv2.threshold(grey, 127, 255, cv2.THRESH\_BINARY\_INV+cv2.THRESH\_OTSU)[1]

# save\_img = cv2.resize(thresh, (50,50))

# cv2.imwrite(path2 + ix + '/' + cx, save\_img)

# MODULE 3

# #............................................Training & Testing using deep learning......................................#

# import os

# import cv2

# import time

# import numpy as np

# from keras.layers import Conv2D

# from keras.layers import Dense

# from keras.layers import Flatten

# from keras.layers import Dropout

# from keras.layers import MaxPooling2D

# from keras.models import Sequential, save\_model

# #from keras.utils import np\_utils

# from keras.utils import to\_categorical

# #from tensorflow.keras.utils import to\_categorical

# #from keras.src.utils.np\_utils import to\_categorical

# from keras.callbacks import ModelCheckpoint

# from keras.models import load\_model

# from sklearn.model\_selection import train\_test\_split

# from sklearn.utils import shuffle

# from sklearn.metrics import accuracy\_score

# from sklearn.metrics import classification\_report, confusion\_matrix

# import warnings

# warnings.filterwarnings(action = 'ignore')

# import seaborn as sns

# import matplotlib.pyplot as plt

# %matplotlib inline

# path = 'Preprocessed/Train/'

# gestures = os.listdir(path)

# gestures

# dict\_labels = {

# 'click':1,

# 'dclick': 2,

# 'down': 3,

# 'left':4,

# 'none':5,

# 'rclick':6,

# 'right':7,

# 'up':8

# }

# print(list(dict\_labels.keys()))

# x, y = [], []

# for ix in gestures:

# images = os.listdir(path + ix)

# for cx in images:

# img\_path = path + ix + '/' + cx

# img = cv2.imread(img\_path, 0)

# img = img.reshape((50,50,1))

# img = img/255.0

# x.append(img)

# y.append(dict\_labels[ix])

# X = np.array(x)

# Y = np.array(y)

# Y = to\_categorical(Y)

# print(type(Y),len(Y))

# Y.shape

# plt.figure(figsize = (18,8))

# sns.countplot(x=list(dict\_labels.keys()))

# Y.shape

# categories = Y.shape[1]

# X, Y = shuffle(X, Y, random\_state=0)

# X.shape

# X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.3)

# print(X\_train.shape, X\_test.shape)

# print(Y\_train.shape, Y\_test.shape)

# model = Sequential()

# model.add(Conv2D(64, kernel\_size=(3,3), activation = 'relu', input\_shape=(50,50 ,1) ))

# model.add(MaxPooling2D(pool\_size = (2, 2)))

# model.add(Conv2D(64, kernel\_size = (3, 3), activation = 'relu'))

# model.add(MaxPooling2D(pool\_size = (2, 2)))

# model.add(Conv2D(64, kernel\_size = (3, 3), activation = 'relu'))

# model.add(MaxPooling2D(pool\_size = (2, 2)))

# model.add(Flatten())

# model.add(Dense(128, activation = 'relu'))

# model.add(Dropout(0.20))

# model.add(Dense(categories, activation = 'softmax'))

# model.summary()

# model.compile(optimizer='Adam', metrics=['accuracy'], loss='categorical\_crossentropy')

# history = model.fit(X\_train, Y\_train, batch\_size=128, epochs=10, validation\_data=[X\_test, Y\_test])

# plt.plot(history.history['accuracy'])

# plt.plot(history.history['val\_accuracy'])

# plt.title("Accuracy")

# plt.xlabel('epoch')

# plt.ylabel('accuracy')

# plt.legend(['train','test'])

# plt.show()

# model.save('CNN\_model.h5')

# m = load\_model('CNN\_model.h5')

# test\_data = os.listdir('Test/')

# dict\_labels

# for ix in test\_data:

# print(ix)

# for ix in test\_data:

# images = os.listdir('Test/' + ix)

# for cx in range(1,10):

# img\_path = 'Test/' + ix + '/' + str(cx) + '.jpg'

# print(img\_path)

# img = cv2.imread(img\_path, 0)

# img = img.reshape((50,50,1))

# img = img/255.0

# x.append(img)

# y.append(dict\_labels[ix])

# X\_t = np.array(x)

# y\_t = np.array(y)

# Y\_t = to\_categorical(y\_t)

# X\_t.shape

# y\_pred = m.predict(X\_t)

# acc = accuracy\_score(Y\_t, y\_pred.round())

# print('Accuracy:', acc)

# print(classification\_report(y\_pred.round(), Y\_t))

# MODULE 4

# import os

# import cv2

# import time

# import numpy as np

# from keras.models import load\_model

# import pyautogui

# import time

# import warnings

# warnings.filterwarnings(action = 'ignore')

# model = load\_model('CNN\_model.h5')

# gestures = {

# 1:'click',

# 2:'dclick',

# 3:'down',

# 4:'left',

# 5:'none',

# 6:'rclick',

# 7:'right',

# 8:'up'

# }

# def predict(gesture):

# img = cv2.resize(gesture, (50,50))

# img = img.reshape(1,50,50,1)

# img = img/255.0

# prd = model.predict(img)

# index = prd.argmax()

# return gestures[index]

# vc = cv2.VideoCapture(0)

# rval, frame = vc.read()

# old\_text = ''

# pred\_text = ''

# count\_frames = 0

# total\_str = ''

# flag = True

# speed = 0.5

# status='1'

# while True:

# #time.sleep(10)

# 

# if frame is not None:

# 

# frame = cv2.flip(frame, 1)

# frame = cv2.resize( frame, (600,600) )

# 

# cv2.rectangle(frame, (400,400), (50,50), (0,255,0), 2)

# 

# crop\_img = frame[100:300, 100:300]

# grey = cv2.cvtColor(crop\_img, cv2.COLOR\_BGR2GRAY)

# 

# thresh = cv2.threshold(grey,210,255,cv2.THRESH\_BINARY\_INV+cv2.THRESH\_OTSU)[1]

# 

# blackboard = np.zeros(frame.shape, dtype=np.uint8)

# cv2.putText(blackboard, "Control Mouse Cursor:- ", (30, 40), cv2.FONT\_HERSHEY\_TRIPLEX, 1, (255, 0, 255))

# if count\_frames > 20 and pred\_text != "":

# total\_str += pred\_text

# count\_frames = 0

# 

# 

# if flag == True and status=='1':

# 

# pred\_text = predict(thresh)

# print(pred\_text)

# cv2.putText(blackboard, " "+str(pred\_text), (30, 100), cv2.FONT\_HERSHEY\_TRIPLEX, 1, (255, 0, 255))

# old\_text = pred\_text

# #f = open('output.txt','w')

# #f.write(str(pred\_text))

# #f.close()

# 

# if str(pred\_text)=='click':

# res = 'click'

# print(res)

# 

# 

# 

# elif str(pred\_text)=='rclick':

# res = 'rclick'

# print(res)

# 

# 

# 

# elif str(pred\_text)=='dclick':

# res = 'dclick'

# print(res)

# 

# 

# elif str(pred\_text)=='down':

# res = 'down'

# print(res)

# 

# 

# elif str(pred\_text)=='left':

# res = 'left'

# print(res)

# 

# elif str(pred\_text)=='none':

# res = 'none'

# print(res)

# 

# elif str(pred\_text)=='right':

# res = 'right'

# print(res)

# 

# 

# 

# 

# elif str(pred\_text)=='up':

# res = 'up'

# print(res)

# 

# 

# 

# 

# 

# 

# #cv2.putText(blackboard, res, (30, 70), cv2.FONT\_HERSHEY\_TRIPLEX, 1, (255, 0, 255))

# 

# f = open('result.txt','w')

# f.write(str(res))

# f.close()

# if old\_text == pred\_text:

# count\_frames += 1

# else:

# count\_frames = 0

# 

# 

# #cv2.putText(blackboard, total\_str, (30, 80), cv2.FONT\_HERSHEY\_TRIPLEX, 1, (0, 255, 0))

# res = np.hstack((frame, blackboard))

# 

# cv2.imshow("image", res)

# #cv2.imshow("hand", thresh)

# #time.sleep(1)

# rval, frame = vc.read()

# keypress = cv2.waitKey(1)

# flag=False

# if keypress == ord('c'):

# flag = True

# if keypress == ord('q'):

# break

# vc.release()

# cv2.destroyAllWindows()

# cv2.waitKey(1)

# vc.release()

# MODULE 5

# import os

# import cv2

# import time

# import numpy as np

# from keras.models import load\_model

# import pyautogui

# import time

# import warnings

# warnings.filterwarnings(action = 'ignore')

# import random

# import pygame

# import webbrowser

# path='D:/final/code/code/music/files/'

# def play\_music(file\_path):

# pygame.mixer.init()

# pygame.mixer.music.load(file\_path)

# pygame.mixer.music.play()

# def play\_music(file\_path):

# pygame.mixer.init()

# pygame.mixer.music.load(file\_path)

# pygame.mixer.music.play()

# def pause\_music():

# pygame.mixer.music.pause()

# def unpause\_music():

# pygame.mixer.music.unpause()

# def stop\_music():

# pygame.mixer.music.stop()

# def increase\_volume(factor):

# current\_volume = pygame.mixer.music.get\_volume()

# new\_volume = min(current\_volume + factor, 1.0) # Ensure volume does not exceed 1.0

# pygame.mixer.music.set\_volume(new\_volume)

# print("Volume increased to:", new\_volume)

# def decrease\_volume(step=0.1):

# current\_volume = pygame.mixer.music.get\_volume()

# new\_volume = max(current\_volume - step, 0.0)

# pygame.mixer.music.set\_volume(new\_volume)

# def load\_files\_in\_folder(folder\_path):

# files = []

# for file\_name in os.listdir(folder\_path):

# if os.path.isfile(os.path.join(folder\_path, file\_name)):

# files.append(os.path.join(folder\_path, file\_name))

# return files

# def process\_music():

# 

# files = load\_files\_in\_folder(path)

# print("Files in folder:")

# for file in files:

# print(file)

# random\_integer = random.randint(0, len(files)-1)

# print(random\_integer)

# fname=path+str(random\_integer)+'.mp3'

# print(fname)

# play\_music(fname)

# pygame.time.wait(5000)

# pause\_music()

# # Resume after 2 seconds

# pygame.time.wait(2000)

# unpause\_music()

# 

# for \_ in range(5):

# pygame.time.wait(5000)

# increase\_volume(0.2)

# # Stop after 10 seconds

# pygame.time.wait(10000)

# stop\_music()

# files = load\_files\_in\_folder(path)

# print("Files in folder:")

# for file in files:

# print(file)

# print(file)

# count=0

# c=0

# flag=False

# while True:

# f=open('task.txt','r')

# task=f.read()

# f.close()

# if task=='1':

# f=open('result.txt','r')

# result=f.read()

# f.close()

# if result=='click':

# pyautogui.click()

# elif result=="rclick":

# pyautogui.click(button='right')

# elif result=="dclick":

# pyautogui.doubleClick()

# elif result=="down":

# pyautogui.moveRel(0, 10, duration=speed)

# time.sleep(0.5)

# elif result=="left":

# pyautogui.move(-100, -100, duration=0)

# elif result=="right":

# pyautogui.move(100, 0, duration=1)

# elif result=='none':

# pass

# elif str(result)=='up':

# pyautogui.moveRel(0, -10, duration=speed)

# time.sleep(0.5)

# f = open('result.txt','w')

# f.write('')

# f.close()

# elif task=='2':

# f=open('result.txt','r')

# result=f.read()

# f.close()

# if result=='click':

# flag=True

# random\_integer = random.randint(0, len(files)-2)

# print('',random\_integer)

# count=random\_integer

# fname=path+str(random\_integer)+'.mp3'

# print(fname)

# play\_music(fname)

# if result=='right' and flag==True:

# stop\_music()

# if count==len(files):

# count=0

# else:

# count=count+1

# fname=path+str(count)+'.mp3'

# print(fname)

# play\_music(fname)

# if result=='left' and flag==True:

# stop\_music()

# if count==0:

# count=len(files)

# else:

# count=count-1

# fname=path+str(count)+'.mp3'

# print(fname)

# play\_music(fname)

# elif result=='stop' and flag==True:

# stop\_music()

# elif result=='rclick' and flag==True:

# pause\_music()

# elif result=='dclick' and flag==True:

# unpause\_music()

# elif result=='up' and flag==True:

# for \_ in range(5):

# pygame.time.wait(3000)

# increase\_volume(0.2)

# elif result=='down' and flag==True:

# for \_ in range(5):

# pygame.time.wait(3000)

# decrease\_volume(0.2)

# f = open('result.txt','w')

# f.write('')

# f.close()

# elif task=='3':

# if c==0:

# url = "http://localhost:8084/Control\_Laptop\_Mouse\_Music\_Presentation/index.jsp"

# webbrowser.open(url)

# c=c+1

# MODULE 6

# import tkinter as tk

# def button1\_clicked():

# print("Mouse")

# f=open('task.txt','w')

# f.write('1')

# f.close()

# def button2\_clicked():

# print("Music")

# f=open('task.txt','w')

# f.write('2')

# f.close()

# def button3\_clicked():

# print("Presentation")

# f=open('task.txt','w')

# f.write('3')

# f.close()

# # Create the main window

# root = tk.Tk()

# root.title("Select the task")

# # Create buttons

# button1 = tk.Button(root, text="Control Mouse", command=button1\_clicked)

# button2 = tk.Button(root, text="Music", command=button2\_clicked)

# button3 = tk.Button(root, text="Presentation", command=button3\_clicked)

# # Place buttons in the window

# button1.pack(pady=10)

# button2.pack(pady=10)

# button3.pack(pady=10)

# # Start the Tkinter event loop

# root.mainloop()

# CHAPTER 7

## SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

### 7.1 TYPES OF TESTING

##### Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce validoutputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process

##### Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shownby successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

##### System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

##### White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at leastits purpose. It is purpose. It is used to test areas that cannot be reached from a blackbox level.

##### Black Box Testing

Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, asmost other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

##### Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirement.

# CHAPTER 8

## CONCLUSION

The use of cameras for interactive computer functions through hand gestures represents a significant technological advancement, transforming how users engage with computers by enabling natural and intuitive hand movements. This innovation, powered by sophisticated computer vision algorithms, allows cameras not only to capture but also interpret a wide range of gestures. These interpreted gestures are then translated into commands, influencing applications across gaming, virtual reality, and smart device interfaces. Beyond convenience, this technology holds immense potential in various industries, offering a hands-free and immersive interface that redefines user experiences. Whether navigating virtual landscapes in gaming, enhancing virtual reality interactions, or streamlining commands on smart devices, the hands-free nature of this technology provides a more seamless and accessible computing experience.

### FUTURE WORK

The future of camera-based interactive computer functions using hand gestures holds immense potential for innovation. Researchers and developers are poised to enhance user experiences by refining gesture recognition algorithms for seamless interaction.

Anticipated advancements include expanded gesture vocabulary, enabling more nuanced commands and improved accuracy. Integration with augmented reality (AR) and virtual reality (VR) environments will elevate immersive experiences, while machine learning algorithms will adapt to individual user preferences, enhancing personalized interactions. Collaborations with other emerging technologies, such as 5G connectivity and edge computing, promise faster response times and reduced latency.

As this field evolves, it is likely to find applications in diverse domains, from gaming and healthcare to education and smart homes, ushering in a new era of intuitive and natural human-computer interaction.

## REFERENCES

* + 1. Chen-Chiung Hsieh ,Dung-Hua Liou and David Lee “A real time hand gesture recognition system using motion history image” Proc.IEEE, 2010, pp. V2-394-V2-398 .
    2. Abhik Banerjee , Abhirup Ghosh , Koustuvmoni Bharadwaj , Hemanta Saikia “Mouse Control using a Web Camera based on Colour Detection”IJCTT ,March 2014, volume 9 number 1, ISSN: 2231-2803
    3. S. Thakur, R. Mehra and B. Prakash, "Vision based computer mouse control using hand gestures," International Conference on Soft Computing Techniques and Implementations (ICSCTI), 2015, pp. 85-89.
    4. Pooja Kumari ,Ghaziabad Saurabh Singh, Ghaziabad Vinay and Kr. Pasi “Cursor Control using Hand Gestures” International Journal of Computer Applications (0975 – 8887),2016.
    5. Alisha Pradhan , B.B.V.L. Deepak “Design of Intangible Interface for Mouseless Computer Handling using Hand Gestures” ICCCV(International Conference on Communication, Computing and Virtualization), 2016, oi: 10.1016/j.procs.2016.03.037.
    6. Aashni Hariaa , Archanasri Subramaniana , Nivedhitha Asokkumara , Shristi Poddara and Jyothi S Nayak “Hand Gesture Recognition for Human Computer Interaction” Proc.ICACC(International Conference on Advances in Computing & Communications), 2017 August,pp367–374
    7. S.Shriram, B.Nagaraj, J Jaya, S Shankar and P Ajay “Deep Learning-Based Real-Time AI Virtual Mouse System Using Computer Vision to avoid Covid-19 Spread” Journal of Health Care Engineering, Volume:2021, Article ID:8133076.