HAND GESTURE CONTROLLED PRESENTATION

Enrollment No. - 9920102046, 9920102050, 9920102057

Name of Student - Aryan Jain, Adeeb Mohsin, Payodhi Sharma

Name of supervisor - Dr. Vimal Kumar Mishra



Submitted in partial fulfillment of the Degree of

Bachelor of Technology

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA (U.P.)**

**CERTIFICATE**

This is to certify that the major project report entitled, *“***Hand Gesture Controlled Presentation***”* submitted in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in **Electronics and Communication Engineering** of the Jaypee Institute of Information Technology, Noida is an authentic work carried out by them under my supervision and guidance. The matter embodied in this report is original and has not been submitted for the award of any other degree.

**Signature of Supervisor:**

**Name of the Supervisor: Dr. Vimal Kumar Mishra**

**ECE Department,**

**JIIT, Sec - 128,**

**Noida - 201304**

**Dated: 04/12/ 2023**

# **DECLARATION**

We hereby declare that this written submission represents our own ideas in our own words and where other's ideas or words have been included, have been adequately cited, and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission.

Place: **JIIT, NOIDA**

Date: 04/12/2023

Name: **ARYAN JAIN**

Enrollment:**9920102046**

Name: **ADEEB MOHSIN**

Enrollment:**9920102050**

Name: **PAYODHI SHARMA**

Enrollment:**9920102057**

**ACKNOWLEDGEMENT**

We would like to express our special thanks of gratitude to my teacher (**Dr. Vimal Kumar Mishra**) who gave us the golden opportunity to do this wonderful project on the topic (Hand Gesture Controlled Presentation), which also helped us in doing a lot of Research and we came to know about so many new things.

**ABSTRACT**

Presentations are crucial in many aspects of life. The objective is to allow people to control the slideshow using hand gestures. The usage of gestures in human-computer interaction has drastically risen in recent years. The system has tried to govern numerous PowerPoint functionalities using hand movements. In this system, machine learning has been applied to recognize motions with tiny differences and map them using multiple libraries in Python. The rising hurdles to creating the optimal presentation are due to several aspects, including the slides, the keys to changing the slides, and the audience's calmness. An intelligent presentation system employing hand gestures gives a simple method to update or control the slides.

Using hand gestures as the system's input to control presentation, we are constructing a presentation controller in this paper. The OpenCV module is mostly utilized in this implementation to control the gestures. MediaPipe is a machine learning framework with a hand gesture detection technology that is available today. This system primarily employs a web camera to record or capture photos and videos, and this application regulates the system's presentation based on the input. The primary purpose of the system is to change its presentation slides, and also had access to a pointer that allowed to draw on slides, in addition to that erase. To operate a computer's fundamental functions, such as presentation control, we may utilize hand gestures. People won't have to acquire the often-burdensome machine-like abilities as a result. These hand gesture systems offer a modern, inventive, and natural means of nonverbal communication. These systems are used widely in human-computer interaction. This project's purpose is to discuss a presentation control system based on hand gesture detection and hand gesture recognition. A high-resolution camera is used in this system to recognize the user's gestures as input. The main objective of hand gesture recognition is to develop a system that can recognize human hand gestures and use that information to control a presentation.

**TABLE OF CONTENTS**

ACKNOWLEDGEMENT ………………..………………………………..……IV

[ABSTRACT…………………………………………………..…………….……...V](#_TOC_250017)

1. Introduction………………………………………………………………………1
2. Literature Survey…………………………………………………………………3
3. Hardware and Software Description…………………………………………….6
   1. Software………………………………………………………………………..6
      1. PyCharm…………………………………………………………………6
      2. Hand Tracking Module…………………………………………………..6
   2. Hardware……………………………………………………………………….7
      1. Webcam…………………………………………………………………7
4. System Design………………………………………………………………………8
   1. Methodology…………………………………………………………………….8
      1. Image Pre-Processsing…………………………………………………….8
      2. Detection Phase………………………………………………………….9
      3. Tracking Phase………………………………………………………….10
      4. Capturing Phase…………………………………………………………10

4.1.5 System Process………………………………………………………….11

* 1. System Architecture…………………………………………………………...12

1. Results and Discussions………………………………………………………….14

Conclusion and Future Scope……………………………………………….…..17

References……………………………………………………………………….18

Appendices………………………………………………………………………19

**LIST OF FIGURES**

1. **Fig 1.1 (System Flow)..……………………………………………………….2**
2. **Fig 2.1 (Proposed system for the research done)…………………………...3**
3. **Fig 2.2 (Method based on computer vision)………………………….…….4**
4. **Fig 2.3 (System based on gesture technique)……………………………......4**
5. **Fig 2.4 (Interactive Presentation System)………………………………...…5**
6. **Fig 2.5 (Flow diagram of system)……………………………………………5**
7. **Fig 3.1 (Logo of Pycharm)…….……………………………………………..6**
8. **Fig 3.2 (Hand Tracking)……………………...……………………………..6**
9. **Fig 3.3 (Webcam)……………………...………………………………….....7**
10. **Fig 4.1 (Representation of PowerPoint Presentation)…………………….8**
11. **Fig 4.2 (Flow diagram of image Pre-processing)……………………….…9**
12. **Fig 4.3 (Detection Phase)…………………………………………………...9**
13. **Fig 4.4 (Flow diagram of Capturing phase)………………………………10**
14. **Fig 4.5 (Hand Capturing)……………………………………….…...……..11**
15. **Fig 4.6 (System Architecture)………………………………….……….....13**
16. **Fig 5.1 (Forward Movement)………………………………….…………..14**
17. **Fig 5.2 (Backward Movement)………………………………….…..…….15**
18. **Fig 5.3 (Pointer Movement)…………………………………….…………15**
19. **Fig 5.4 (Eraser Movement)……………………………………….……….16**
20. **Fig 6.1 (Future Possibility)……………………………………..…………17**

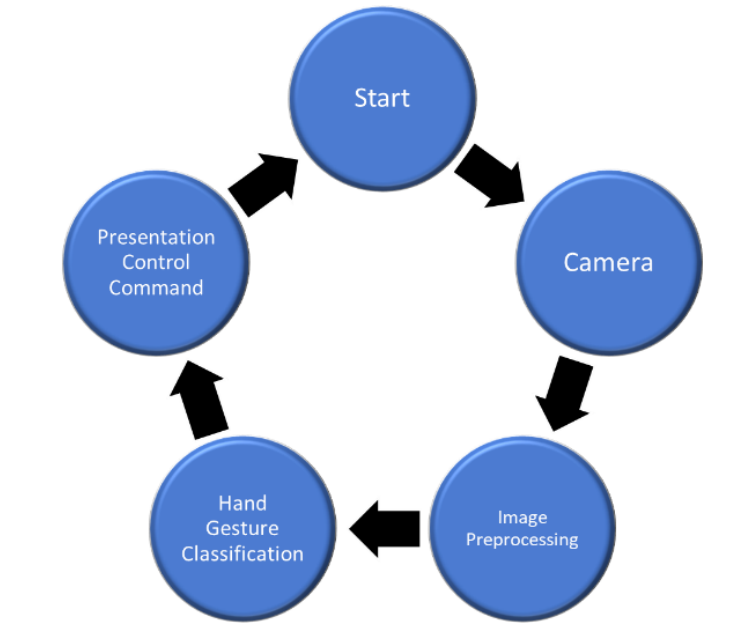
**Chapter - 1**

**Introduction**

In today's digital environment, presenting is an attractive and efficient strategy to assist presenters in convincing the audience and delivering information. Slides can be manipulated using a mouse, keypad, laser pointer, etc. The downside is that controlling the devices needs prior device expertise. A few years ago, gesture recognition became more helpful in operating software like media players, robotics, and games. The hand gesture recognition system facilitates the employment of gloves, markers and other objects. However, using such gloves or markers raises the system's cost. This system's suggested hand gesture detection technique is based on artificial intelligence. Users can edit the slides. The interactive presentation system employs cutting-edge human-computer interaction techniques to develop a more practical and user-friendly interface for controlling presentation displays. The presenting experience is greatly enhanced when utilizing these hand motion options instead of a standard mouse and keyboard control. The use of body movement to express a specific message through gestures is nonverbal or non-vocal communication.

The use of technologies and improvements in human-machine interaction allow people to identify, communicate, and engage with one another using a wide variety of gestures. The gesture is a type of nonverbal communication or non-vocal communication that makes use of the body's movement to express a specific message. The hand or face are the most frequently used portions of the body. Recognizing hand movements using Human-Computer-Interaction (HCI) might aid in achieving the necessary ease and naturalness. Hand gestures serve the purpose of communicating information when engaging with other individuals. encompassing both basic and complicated hand motions. For instance, we can point with our hands towards an item or at individuals, or we can convey basic hand shapes or motions using manual articulations in conjunction with sign languages' well-known syntax and lexicon. Therefore, employing hand gestures as a tool and integrating them with computers might enable more intuitive communication between individuals. To simplify things to anybody thus create Artificial Intelligence (AI) based apps, various frameworks or libraries have been developed for hand gesture detection. MediaPipe is one of them. Few benefits for employing mediapipe framework's showcases include helping programmer concentrate on model and algorithm creation for application and supporting application’s environment via result repeatable allover multiple architecture and gadgets. To conduct different activities, such as seeking ahead and backward through slides, drawing and erasing in a presentation, the project employs hand gestures, often no of raised fingers inside the region of interest.

The system was constructed mainly using the Python framework along with technologies like an open cv, cv zone, NumPy, and media pipe. This method seeks to enhance the efficacy and utility of presentations. Additionally, the system employs movements to write, undo, and get the pointer on different text regions. To enhance the slideshow experience, we wanted to make it possible for users to control it with hand gestures. To optimize and improve the portability of the display, the system minimizes the use of an external interface. Slides may be managed and controlled by various motions, including swiping left, right, thumbing up, and others. This system uses a Hand gesture-based human-machine interface for a conventional presentation flow. The interface has been actively developed throughout the previous several years. We built a quick and straightforward motion image-based technique to identify dynamic hand gestures. Fig.1. illustrates the process flow diagram. This technique allows users to govern the presentations more naturally, rationally, and conveniently.



**Fig.1 System Flow [1]**

This project arises from the desire to bridge the gap between presenters and their audiences, offering a seamless and captivating means of controlling presentations. The Gesture-Controlled Presentation System utilizes advanced gesture recognition technology, enabling presenters to navigate their content and interact with their audience in an entirely new way.

**Chapter - 2**

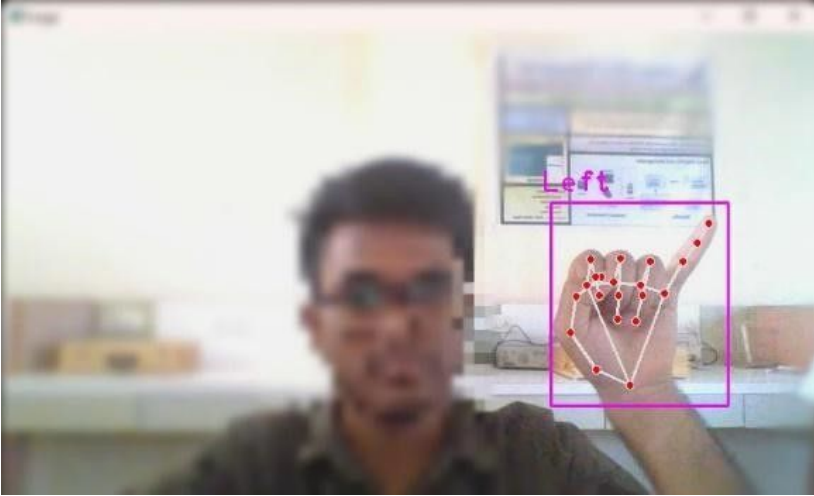
**Literature Survey**

### In the ever-evolving landscape of human-computer interaction, the integration of hand gesture control has emerged as a captivating and innovative approach, particularly in the realm of presentations. This literature survey delves into the rich tapestry of research and advancements surrounding hand gesture-controlled presentations. As technology continues to bridge the gap between humans and machines, the exploration of gesture-based interfaces holds tremendous potential for transforming the way we interact with and convey information during presentations.

### Summary of the Papers Studied:

1. **Smart Presentation System Using Hand Gestures. By: Bhairavi Pustode, Vedant Pawar, Varun Pawar, Tejas Pawar, Samiksha Pokale.**

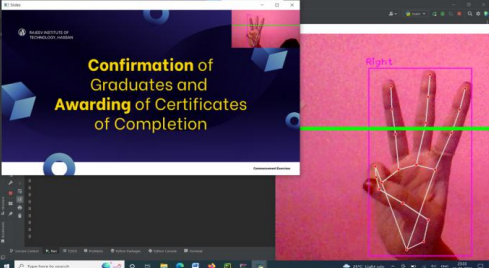
In this paper, we presented a smart presentation control system using hand gestures, implemented using Python and computer vision techniques. The system allows the presenter to control the slides using simple hand movements, providing a convenient and intuitive way for controlling presentations. Through various experiments, we demonstrated the effectiveness and performance of the system. We believe that the proposed system has the potential to improve the overall presentation experience and make presentations more interactive and engaging. Fig. 2.1. shows hand gesture tracking.



**Fig. 2.1 Proposed system for the research done [1]**

1. **Smart Presentation Control by Hand Gestures Using Computer Vision and Google’s Mediapipe. By: Hajeera Khanum, Dr. Pramod H B**

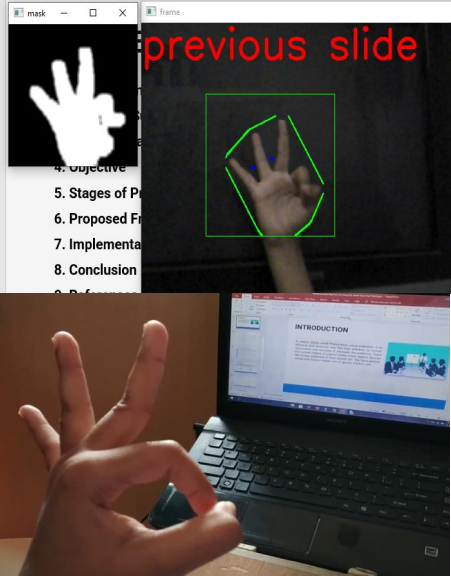
A gesture-based presentation controller doesn't need any special markers, and it can be used in real life on basic PCs with inexpensive cameras since it doesn't need particularly high-quality cameras to recognize or record the hand movements. The method keeps track of the locations of each hand's index finger and counter tips. This kind of system's primary goal is to essentially automate system components so that they are easy to control. As a result, we have employed this method to make the system simpler to control with the aid of these applications in order to make it realistic. Fig. 2.2. illustrates hand tracking through computer vision.



**Fig. 2.2 Method Based on Computer Vision [2]**

### **Automated Digital Presentation Control using Hand Gesture Technique By: Salonee Powar, Shweta Kadam, Sonali Malage, Priyanka Shingane**

It is the implementation of adaptive thresholds for the estimation of the flicker parameters, that is, the values that decide whether the hand is open or closed. To evaluate the flicker detector, each frame processed by the software was analyzed, resulting in an efficiency of 97%. Fig. 2.3 shows the idea of this technique.



**Fig. 2.3 System based on Gesture Technique [3]**

1. **A Hand Gesture Based Interactive Presentation System Utilizing Heterogeneous Cameras By: Bobo Zeng, Guijin Wang, Xinggang Lin**

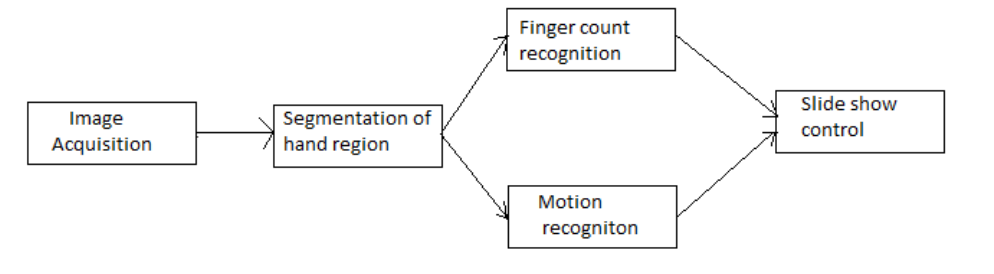
In this paper, a real-time system that utilizes hand gestures to interactively control the presentation is proposed. The system employs a thermal camera for robust human body segmentation to handle the complex background and varying illumination posed by the projector. Hand trajectories are segmented and recognized as gestures for interactions. A dual-step calibration algorithm is utilized to map the interaction regions between the thermal camera and the projected contents by integrating a Web camera.



**Fig. 2.4 Interactive presentation system [4]**

1. **Hand Gesture Recognition System to Control Slide Show Navigation. By: Dnyanada Jadhav1, Prof. L.M.R.J. Lobo2, Volume 3, Issue 1, January 2014**

Both static and dynamic gestures are used to control the slide show. Any finger can be used to denote the gesture. No restriction to use a specific finger for particular gesture as active fingers are counted using distance transform method. This does not require any training phase to identify a hand gesture hence does not require storage of images in database to recognize the hand gestures. Usage of hand gestures can be extended to control real-time.



**Fig. 2.5 Flow diagram of the system [5]**

# Chapter 3

**Technology Overview**

## **(3.1) Software:**

# **(3.1.1) PyCharm:**

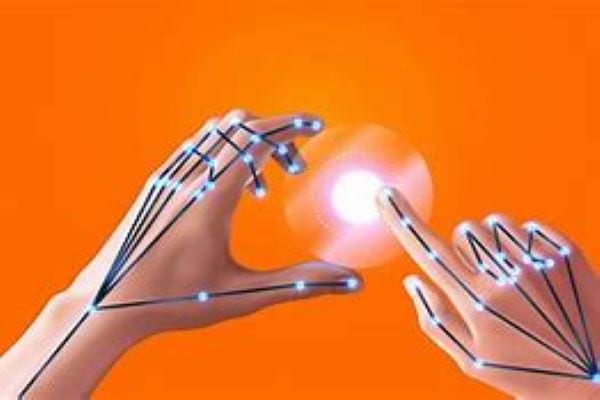
PyCharm is an integrated development environment (IDE) that helps professional Python developers be more productive, be more confident, and write better code. It comes in two versions, PyCharm Pro and PyCharm Community, and supports the full Python workflow out of the box in the latter, including web frameworks, frontend technologies, databases, and scientific tooling. It has full support for Python, including code insight, debugging capabilities, testing, and more. You’re free to use it whenever, and wherever, you like, including at work.



**Fig. 3.1. Logo of PyCharm**

### **(3.1.2) Hand Tracking Module:**

Hand tracking is the process in which a computer uses computer vision to detect a hand from an input image and keeps focus on the hand’s movement and orientation. Hand tracking allows us to develop numerous programs that use hand movement and orientation as their input.



**Fig. 3.2 Hand Tracking**

**(3.2) Hardware Used:**

### **(3.2.1) Webcam/Camera:**

A **webcam** is a [video camera](https://en.wikipedia.org/wiki/Video_camera) which is designed to record or stream to a [computer](https://en.wikipedia.org/wiki/Computer) or [computer network](https://en.wikipedia.org/wiki/Computer_network). They are primarily used in [video telephony](https://en.wikipedia.org/wiki/Video_telephony), [live streaming](https://en.wikipedia.org/wiki/Live_streaming) and [social media](https://en.wikipedia.org/wiki/Social_media), and [security](https://en.wikipedia.org/wiki/Closed-circuit_television). Webcams can be built-in computer hardware or [peripheral devices](https://en.wikipedia.org/wiki/Peripheral), and are commonly connected to a device using [USB](https://en.wikipedia.org/wiki/USB) or [wireless protocols](https://en.wikipedia.org/wiki/Internet_protocol_suite). These cameras are built into a monitor and are typically positioned just above the screen and affixed to the monitor’s frame. Internal webcams are on laptop computers. However, you can also find them in stand-alone monitors and all-in-one computers.



**Fig. 3.3 Webcam**

A **webcam** is a [video camera](https://en.wikipedia.org/wiki/Video_camera) which is designed to record or stream to a [computer](https://en.wikipedia.org/wiki/Computer) or [computer network](https://en.wikipedia.org/wiki/Computer_network). They are primarily used in [video telephony](https://en.wikipedia.org/wiki/Video_telephony), [live streaming](https://en.wikipedia.org/wiki/Live_streaming) and [social media](https://en.wikipedia.org/wiki/Social_media), and [security](https://en.wikipedia.org/wiki/Closed-circuit_television). Webcams can be built-in computer hardware or [peripheral devices](https://en.wikipedia.org/wiki/Peripheral), and are commonly connected to a device using [USB](https://en.wikipedia.org/wiki/USB) or [wireless protocols](https://en.wikipedia.org/wiki/Internet_protocol_suite).

Webcams are considered an essential accessory for [remote work](https://en.wikipedia.org/wiki/Remote_work), mainly to compensate for lower quality video processing with the built-in camera of the average laptop. During the [COVID-19 pandemic](https://en.wikipedia.org/wiki/COVID-19_pandemic), there was a shortage of webcams. Most laptops before and during the pandemic were made with cameras capping out at 720p recording quality at best, compared to the industry standard of 1080p seen in smartphones and televisions.

**Chapter 4**

**System Design**

## **(4.1) Methodology**

## There are several different algorithms and methods for eye tracking, and monitoring. Most of them in some way relate to features of the eye (typically reflections from the eye) within a video image of the driver. The original aim of this project was to use the retinal reflection as a means to finding the eyes on the face, and then using the absence of this reflection as a way of detecting when the eyes are closed. Applying this algorithm on consecutive video frames may aid in the calculation of eye closure period. Eye closure period for drowsy drivers is longer than normal blinking. It is also very little longer time could result in severe crash. So, we will warn the driver as soon as closed eye is detected. See Fig. 4.1.

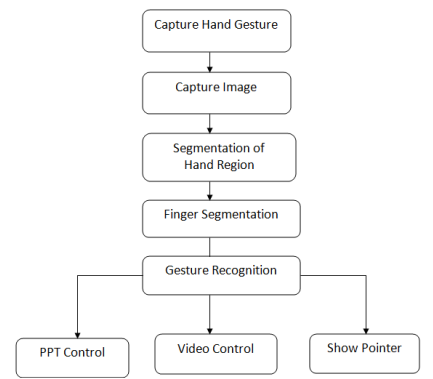
## 

**Fig. 4.1 Representation of PowerPoint Presentation**

**(4.1.1) Image Pre-Processing**: The point of pre-processing is to improve the standard of the image all together that we will examine it in an exceptionally better manner. By pre-processing we will smother undesired distortions and upgrade a few elements which are essential for the real application we are working for. Those features might vary for different applications.

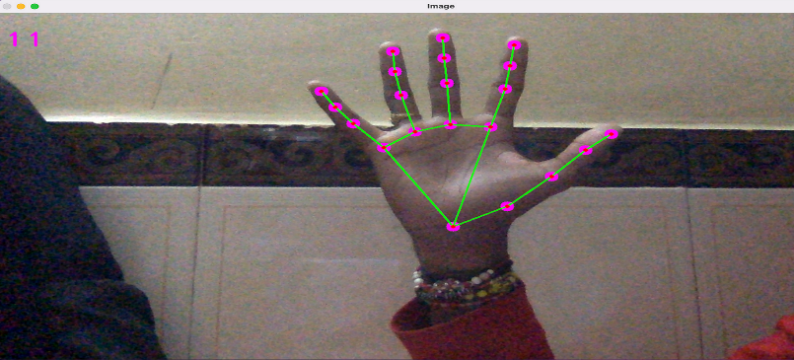
Steps for Image pre-processing:

* Select a boundary of the input image within which we'll scan for the presence of a person's hand.
* Produce a mask by opting only pixels that match a specified color range.
* Blur the mask image so that missing data points can be filled.
* Draw a contour of the hand and use Open CV to identify the fingers.



**Fig 4.2. Flow diagram of Image Pre-Processing [1]**

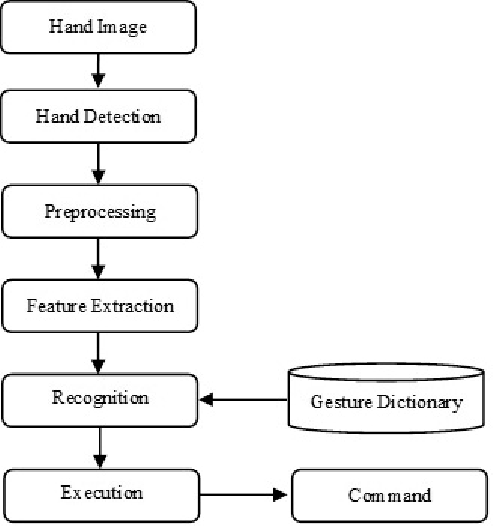
**(4.1.2) Detection Phase:** The analysis of information from the camera are done to deduce the user’s current hand behavior style. The open/closed state of hand is deduced by means of image processing techniques using computer vision. The image processing techniques are performed inside PC.



**Fig 4.3 Detection Phase**

**(4.1.3) Tracking Phase:** The Media Pipe frame-work is utilized for hand motion reputation or tracking, whilst the Open CV library was used for pc vision. To detect and recognize hand actions and hand suggestions, the programmed employs gadget getting to know thoughts. 3.4.1 MediaPipe: The ability to recognize the form a motion of hand might be crucial in enhancing the consumer experience along a broad spectrum of technological domains domain names or structures. It could, for instance, serve as the foundation for gesture recognition technologies as well as palm and motion control, in augmented reality, digital materials and information are superimposed over top of physical surroundings. Since hands entrap oneself or others on a frequent basis (e.g., thumb one may and holding hands) and lack strong similarity pattern, strong real-time palm notion was a seriously challenging pc imaginative and prescient challenge in humans.

**(4.1.4) Capturing Phase:** This is a home desktop package that automates picture processing. And artificial intelligence activities. Its foundation is based on the Open CV and Media pipe libraries. Open CV seems to be a huge open source, deep learning, as well as image analysis library. Open CV works with several different programming translations, featuring to Python, Java, and C++. Using pictures and mp4, this could understand things, humans, or even handwriting of humans. In conjunction with other libraries, that is NumPy, a surprisingly effective numerical library an operation, your arsenal expands; for example, certain to operations that are simple in NumPy may be merged with Open CV.



**Fig. 4.4. Flow diagram of Capturing Phase [2]**

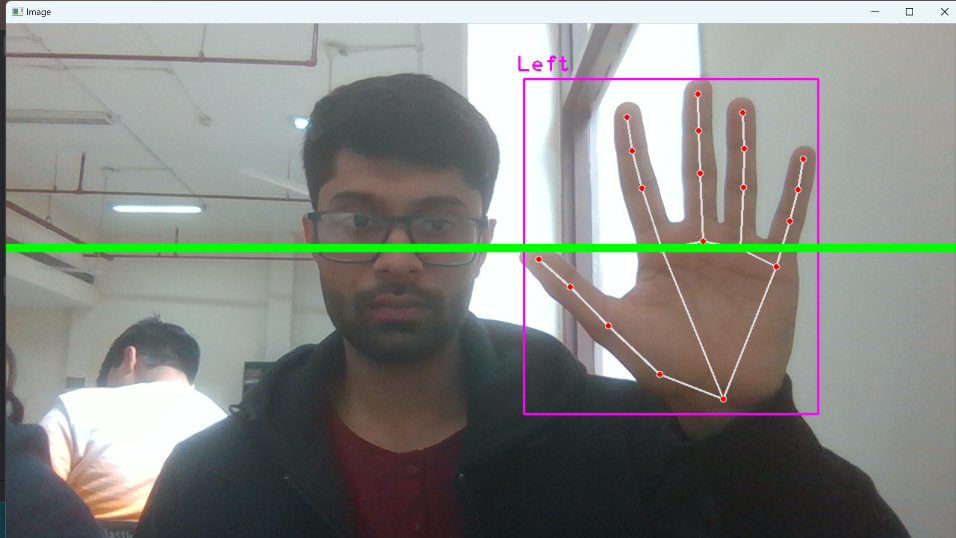
### **(4.1.4) System Process:**

**Hand detection function:** Once a hand is detected, we use a Hand Module to detect and classify the hand gestures. Data is trained using a vector set of hand gestures, which includes examples of various gestures such as changing slides, next slide, previous slide, pointing, and highlight points.

1. A hand detector model processes the captured image and turns the image with an oriented bounding box of the hand.

2. A hand landmark model processes on cropped bounding box image and returns 3D hand key points on hand.

3. A gesture recognizer that classifies 3D hand key points then configuration them into a discrete set of gestures.



**Fig. 4.4 Hand Capturing**

Hand gesture recognition is done using Python programming language and OpenCV as library. Python programming language produces simple and easy system code to understand. Also, Python package used here is NumPy.

The image that is captured using web camera will be processed in a region called as Region of Interest (ROI) where act as a region of wanted area while ignoring the outside region, called background. See Fig. 4.4 for reference.

**Testing Gesture**: 1: to next slide [0,0,0,0,1] Only the little finger is open in this gesture, and the other four fingers are all closed.

2: to Previous slide [1,0,0,0,0] Only the thumb is open in this gesture, whereas the other fingers are all closed.

3: to obtain the pointer [0,1,1,0,0] In this gesture, the forefinger and middle finger are both open, while the remaining fingers are all closed. Gesture

4: to write on the slides [0,1,0,0,0] In this gesture, just the forefinger is open; the other four are all closed.

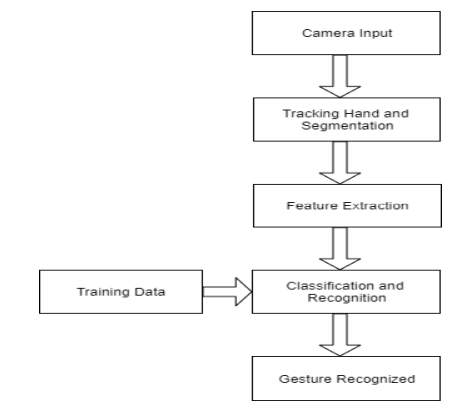
5: Undo [0,1,1,1,0] In this gesture, just the forefinger, middle finger, and ring finger are open; all the other fingers are closed. The most recent written part is to be erased with this gestures.

**(4.2) System Architecture:**

## The system architecture for a hand gesture-controlled presentation involves a multi-faceted integration of hardware and software components. The process begins with a Gesture Input Device, such as a depth-sensing camera or motion sensor, capturing and processing user hand movements in real-time. The Data Processing Module interprets this information through gesture recognition algorithms, transmitting actionable commands via a Communication Interface to the Presentation Software. This software, equipped with slide navigation and control features, collaborates with the User Interface to provide feedback on gesture recognition and system status. A Feedback Mechanism enhances the user experience with haptic or audio cues, while User Settings and Calibration enable customization for individual preferences. Security measures in the Security and Reliability Module ensure protected access, and Hardware Integration ensures compatibility with presentation hardware. The architecture also incorporates a Power Management system for efficient energy use, collectively creating an immersive and interactive presentation experience through intuitive hand gestures.

The overall architecture promotes modularity, enabling updates and enhancements to individual layers without disrupting the entire system. The system's responsiveness is optimized by minimizing latency between gesture recognition and presentation control. Additionally, security measures are implemented to safeguard user data and ensure the integrity of the presentation environment. The Presentation Control Layer is responsible for interfacing with the presentation software and translating the recognized gestures into executable commands.

## It includes the user interface (UI) components that allow presenters to customize gestures, providing a user-friendly and intuitive experience.



**Fig. 4.5 System Architecture [2]**

The system prioritizes real-time processing to minimize latency and provide users with instantaneous feedback, ensuring a smooth and natural interaction. A database stores predefined gestures and their corresponding actions. The system continually refines its gesture database through machine learning, adapting to user variations and expanding the repertoire of recognized gestures over time. A calibration process allows users to personalize the system by adapting it to their unique hand movements. This involves mapping individual gestures to specific commands and adjusting recognition parameters based on user preferences. The Presentation Control Layer translates recognized gestures into executable commands, triggering actions such as advancing slides, changing multimedia content, or annotating presentations in real time. The layer ensures synchronization with the presentation software, maintaining a cohesive user experience.

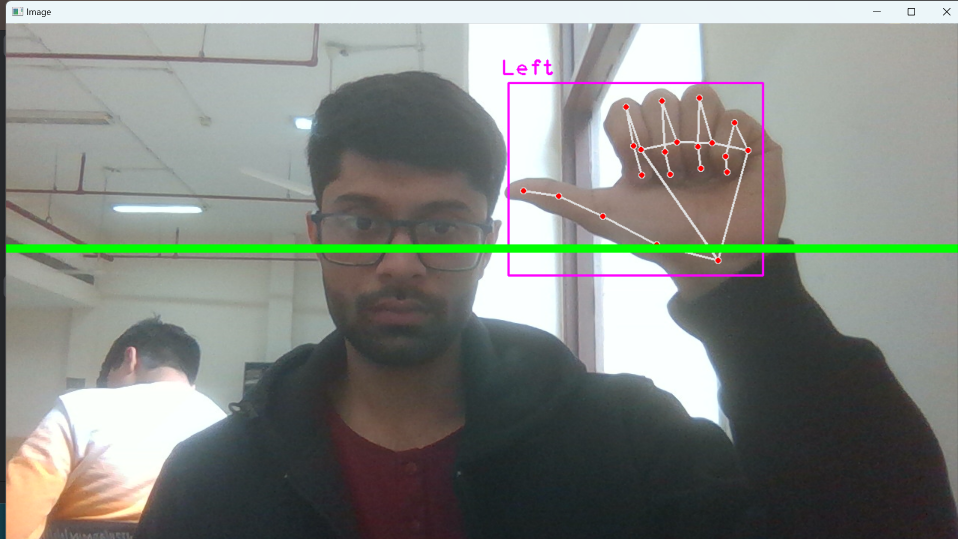
In summary, the Gesture-Controlled Presentation System's architecture is a well-orchestrated integration of hardware, gesture recognition algorithms, and presentation control software. This architecture aims to provide a reliable, adaptable, and user-friendly platform that redefines the way presentations are delivered and experienced.

# **Chapter 5**

# **Results**

### **Backward movement of Slide:**

Below result represent the forward movement of our slide. Fig. 5.1. represent [1,0,0,0,0] matrix which moves to the next slide. This gesture is identified when only the thumb is open, while the other fingers are closed. The system interprets this gesture as a command to navigate to the previous slide in the presentation. It provides an intuitive and easily recognizable gesture for seamless backward navigation.



**Fig. 5.1 Backward Movement**

**Forward movement of Slide:**

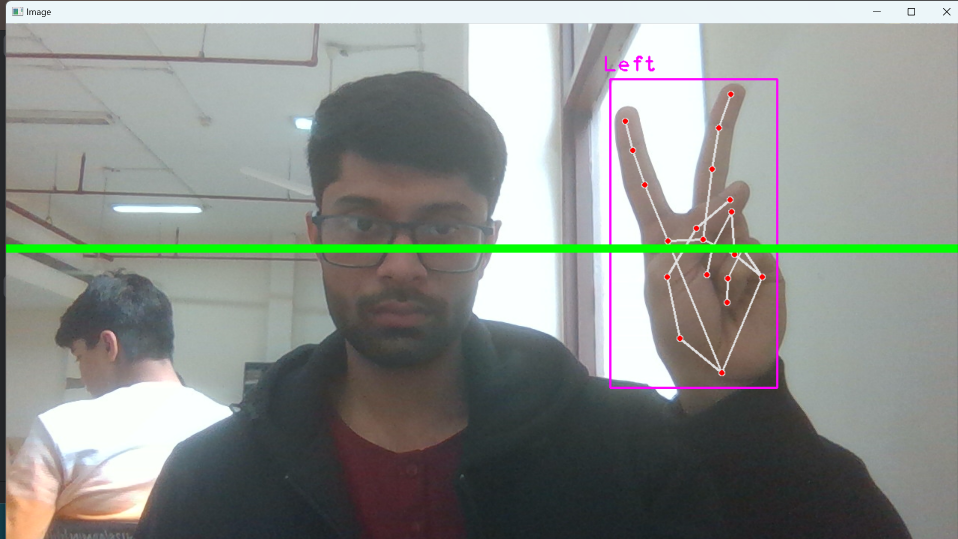
Below result represent the backward movement of the slide. Fig. 5.2. represent [0,0,0,0,1] matrix which moves to the previous slide. This gesture is recognized when only the little finger is open, and the other four fingers are closed. The system interprets this as a command to advance to the next slide in the presentation. It offers a clear and distinct gesture, enhancing user control during presentations. This distinct hand configuration minimizes the likelihood of misinterpretation, providing a seamless and precise way for presenters to navigate through slides effortlessly



**Fig. 5.2 Forward Movement**

**Pointer Movement on slide:**

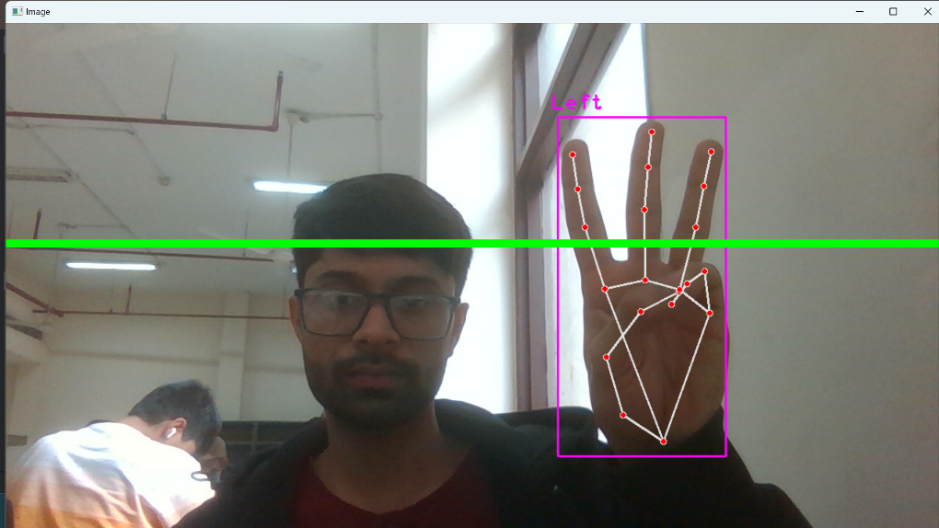
Here, we can draw on slide using the pointer and this is how we can activate the pointer. Fig. 5.3. represent [0,1,1,0,0] matrix which opens the pointer. This gesture is detected when both the forefinger and middle finger are open, and the remaining fingers are closed. The system interprets this as a command to obtain a pointer, allowing the user to interact with the presentation interface. It facilitates a natural and responsive interaction for specific actions.



**Fig. 5.3 Pointer Movement**

**Eraser Movement on slide:**

Anything written using pointer can be erased using the eraser. And this is how the eraser can be activated. Fig. 5.4. represent [0,1,1,0,0] matrix activates the eraser. This gesture is identified when the forefinger, middle finger, and ring finger are open, while all the other fingers are closed. The system interprets this as a command to undo the most recent action, specifically erasing the most recently written part on the slides. It provides a convenient and intuitive way to correct or backtrack during interactive writing sessions.



**Fig 5.4 Erase Movement**

These detailed explanations showcase how each gesture is uniquely defined by the configuration of open and closed fingers. The system's gesture recognition algorithms analyze these patterns in real-time, allowing users to execute specific commands effortlessly during a presentation. The gestures are designed to be distinct, minimizing the likelihood of false positives and ensuring an accurate and responsive user experience.

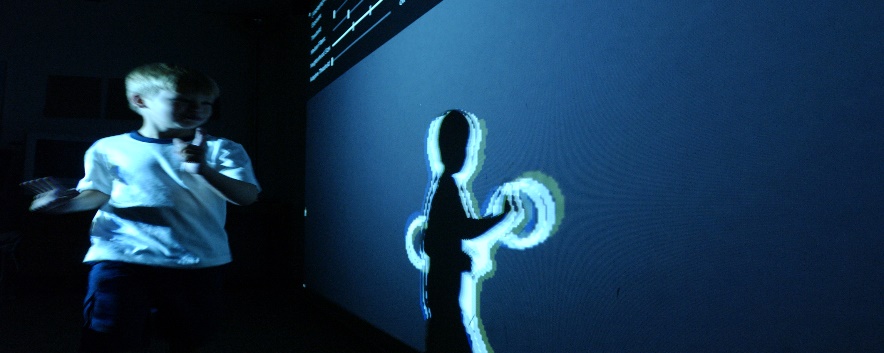
**Conclusion and Future Scope**

In this paper, we presented a smart presentation control system using hand gestures, implemented using Python and computer vision techniques. The system allows the presenter to control the slides using simple hand movements, providing a convenient and intuitive way for controlling presentations. Through various experiments, we demonstrated the effectiveness and performance of the system. We believe that the proposed system has the potential to improve the overall presentation experience and make presentations more interactive and engaging. This kind of system's primary goal is to essentially automate system components so that they are easy to control. Presenter will be able to change slides without using any external device. This will be useful in corporate or institutions where presentation is part of work.

### Global Scope:

Gestures are utilized in numerous disciplines and have considerable value. Gestures are the future of real- time interactions. The technology aids pupils with human-computer interaction when gestures come into play. By adding additional movements, we can control computer programs like cut, copy, paste, etc. We can expand our system to manage the PowerPoint application as well. With potential advances in gesture recognition, augmented reality and artificial intelligence, the future of arts management looks promising. Integration with new technologies can expand their applications in areas ranging from education to business presentations.

Its global applicability is highlighted by its universal language of gestures, transcending linguistic and cultural barriers and fostering cross-cultural communication.



**Fig. FUTURE Possibility [6]**

**References**

[1] D. Jadhav, Prof. L.M.R.J. Lobo, Hand Gesture Recognition System To Control Slide Show Navigation IJAIEM, Vol. 3, No. 4 (2014)

[2] M. Harika, A. Setijadi P, H. Hindersah, FingerPointing Gesture Analysis for Slide Presentation, Bong-Kee Sin Journal Of Korea Multimedia Society, Vol. 19, No. 8, August (2016)

[3] Md.F. Wahid, R. Tafreshi, M. Al-Sowaidi, R. Langari, An Efficient Approach to Recognize Hand Gestures Using Machine-Learning Algorithms, IEEE 4th MECBME, (2018)

[4] D.O. Lawrence, and Dr. M.J. Ashleigh, Impact Of Human-Computer Interaction (HCI) on Users in Higher Educational System: Southampton University As A Case Study, Vol.6, No 3, pp. 1-12, September (2019)

[5] I. Dhall, S. Vashisth, G. Aggarwal, Automated Hand Gesture Recognition using a Deep Convolutional Neural Network, 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), (2020)

[6] Ren, Zhou, et al. Robust part-based hand gesture recognition using Kinect sensor, IEEE Transactions on multimedia 15.5, pp.1110-1120, (2013)

[7] Ajay Talele, Aseem Patil, Bhushan Barse on the Detection of Real Time Objects Using TensorFlow and OpenCV, Asian Journal of Convergence in Technology, Vol 5, (2019)

[8] Ahmed Kadem Hamed AlSaedi, Abbas H. Hassin Al Asadi, A New Hand Gestures Recognition System, Indonesian Journal of Electrical Engineering and Computer Science, Vol 18, (2020)

**Appendices**

## **Libraries Specifications:**

**Libraries imported for Hand Image Processing using Python:**

**NUMPY:** NumPy is a Python library used for **working with arrays**. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open-source project and you can use it freely.

**OpenCV:** OpenCV is a Python library that **allows you to perform image processing and computer vision tasks**. It provides a wide range of features, including object detection, face recognition, and tracking.

**Mediapipe:** MediaPipe is an open-source framework for building pipelines to perform computer vision inference over arbitrary sensory data such as video or audio. Using MediaPipe, such a perception pipeline can be built as a graph of modular components.

**OS:** This module provides a portable way of using operating system dependent functionality. If you just want to read or write a file see [open()](https://docs.python.org/3/library/functions.html#open), if you want to manipulate paths, see the [os.path](https://docs.python.org/3/library/os.path.html" \l "module-os.path" \o "os.path: Operations on pathnames.) module, and if you want to read all the lines in all the files on the command line see the [fileinput](https://docs.python.org/3/library/fileinput.html" \l "module-fileinput" \o "fileinput: Loop over standard input or a list of files.) module. For creating temporary files and directories see the [tempfile](https://docs.python.org/3/library/tempfile.html" \l "module-tempfile" \o "tempfile: Generate temporary files and directories.) module, and for high-level file and directory handling see the [shutil](https://docs.python.org/3/library/shutil.html" \l "module-shutil" \o "shutil: High-level file operations, including copying.) module.

**Hand Tracking Module:** Hand tracking is the process in which a computer uses computer vision to detect a hand from an input image and keeps focus on the hand's movement and orientation. Hand tracking allows us to develop numerous programs that use hand movement and orientation as their input. The Hand Tracking Module aims for high precision in tracking hand movements, ensuring that the system can interpret subtle gestures accurately. This precision is crucial for providing users with a natural and responsive interaction experience.

**CODE:**

from cvzone.HandTrackingModule import HandDetector  
import cv2  
import os  
import numpy as np  
  
# Parameters  
width, height = 1280, 720  
gestureThreshold = 300  
folderPath = "Presentation"  
  
# Camera Setup  
cap = cv2.VideoCapture(0)  
cap.set(3, width)  
cap.set(4, height)  
  
# Hand Detector  
detectorHand = HandDetector(detectionCon=0.8, maxHands=1)  
  
# Variables  
imgList = []  
delay = 30  
buttonPressed = False  
counter = 0  
drawMode = False  
imgNumber = 0  
delayCounter = 0  
annotations = [[]]  
annotationNumber = -1  
annotationStart = False  
hs, ws = int(120 \* 1), int(213 \* 1) # width and height of small image  
  
# Get list of presentation images  
pathImages = sorted(os.listdir(folderPath), key=len)  
print(pathImages)  
  
while True:  
 # Get image frame  
 success, img = cap.read()  
 img = cv2.flip(img, 1)  
 pathFullImage = os.path.join(folderPath, pathImages[imgNumber])  
 imgCurrent = cv2.imread(pathFullImage)  
  
 # Find the hand and its landmarks  
 hands, img = detectorHand.findHands(img) # with draw  
 # Draw Gesture Threshold line  
 cv2.line(img, (0, gestureThreshold), (width, gestureThreshold), (0, 255, 0), 10)  
  
 if hands and buttonPressed is False: # If hand is detected  
  
 hand = hands[0]  
 cx, cy = hand["center"]  
 lmList = hand["lmList"] # List of 21 Landmark points  
 fingers = detectorHand.fingersUp(hand) # List of which fingers are up  
  
 # Constrain values for easier drawing  
 xVal = int(np.interp(lmList[8][0], [width // 2, width], [0, width]))  
 yVal = int(np.interp(lmList[8][1], [150, height-150], [0, height]))  
 indexFinger = xVal, yVal  
  
 if cy <= gestureThreshold: # If hand is at the height of the face  
 if fingers == [1, 0, 0, 0, 0]:  
 print("Left")  
 buttonPressed = True  
 if imgNumber > 0:  
 imgNumber -= 1  
 annotations = [[]]  
 annotationNumber = -1  
 annotationStart = False  
 if fingers == [0, 0, 0, 0, 1]:  
 print("Right")  
 buttonPressed = True  
 if imgNumber < len(pathImages) - 1:  
 imgNumber += 1  
 annotations = [[]]  
 annotationNumber = -1  
 annotationStart = False  
  
 if fingers == [0, 1, 1, 0, 0]:  
 cv2.circle(imgCurrent, indexFinger, 12, (0, 0, 255), cv2.FILLED)  
  
 if fingers == [0, 1, 0, 0, 0]:  
 if annotationStart is False:  
 annotationStart = True  
 annotationNumber += 1  
 annotations.append([])  
 print(annotationNumber)  
 annotations[annotationNumber].append(indexFinger)  
 cv2.circle(imgCurrent, indexFinger, 12, (0, 0, 255), cv2.FILLED)  
  
 else:  
 annotationStart = False  
  
 if fingers == [0, 1, 1, 1, 0]:  
 if annotations:  
 annotations.pop(-1)  
 annotationNumber -= 1  
 buttonPressed = True  
  
 else:  
 annotationStart = False  
  
 if buttonPressed:  
 counter += 1  
 if counter > delay:  
 counter = 0  
 buttonPressed = False  
  
 for i, annotation in enumerate(annotations):  
 for j in range(len(annotation)):  
 if j != 0:  
 cv2.line(imgCurrent, annotation[j - 1], annotation[j], (0, 0, 200), 12)  
  
 imgSmall = cv2.resize(img, (ws, hs))  
 h, w, \_ = imgCurrent.shape  
 imgCurrent[0:hs, w - ws: w] = imgSmall  
  
 cv2.imshow("Slides", imgCurrent)  
 cv2.imshow("Image", img)  
  
 key = cv2.waitKey(1)  
 if key == ord('q'):  
 break