AIR CANVAS USING COMPUTER VISION

1Harshita, 2Muskaan Gupta

1,2Department of Computer Science & Engineering

1,2Shri Shankaracharya Engineering College, Bhilai (C.G.)

1. **Abstract**

Writing in the air has become one of the most interesting and challenging developments in the fields of image processing and pattern design nowadays. It significantly contributes to the advancement of automated processes and can enhance the interface between humans and machines in various application. In several research areas, efforts are focusing on new techniques and methods that help reduce processing time and provide high recognition efficiency and accuracy. Object tracking is considered an essential task in the field of computer vision. It involves initially detecting the object, subsequently tracking its movement from frame to frame, and finally analyzing the behaviour of that object. We will use computer vision to track the path of a finger. This will be a powerful means of communication and an effective method that reduces the need to write.

As we all know, artists create paintings on a canvas. Now imagine if we could paint in the air just by waving our hands. In this project, we are going to design an air canvas using OpenCV and Python.

**Keywords**: computer vision, air writing, hand and real-time gesture control system, object detection.

**2. Introduction**

In the evolving digital world, the traditional art of writing is increasingly being supplanted by digital art. Digital art involves expressing and transmitting art forms digitally, as opposed to traditional art, which encompasses creations from before the digital era. The analysis of these arts can be categorized into visual arts, audio arts, audio-visual arts, and audio-visual fantasy arts. There exists a symbiotic relationship between digital and traditional art.

Currently, digital and traditional arts coexist in a state of symbiosis, highlighting the importance of understanding traditional methods such as using pen and paper, chalk, and blackboard writing. The primary aim of digital art in this context is to develop a hand gesture recognition system for digital writing. Digital writing can be executed through various methods including keyboards, touchscreen surfaces, digital pens, styluses, and electronic gloves. However, in this system, we focus on hand gesture recognition using computer vision and Python programming to create a natural human-machine interaction. With technological advancements, there is a growing need to develop natural human-computer interaction (HCI) systems to replace traditional ones.

Color detection is an image processing technique that allows for the identification of specific colors within a certain range of the HSV color space. Image segmentation, on the other hand, is a process of labeling each pixel in an image such that each pixel is grouped with others that share similar characteristics.

**Hardware Requirements:**

* Processor – Intel I5
* RAM – 4GB
* Storage – 1GB
* Web Camera

**Software Requirements:**

* Windows 10
* Python
* NumPy
* OpenCV
* Mediapipe

**3. Challenges identified**

1. Fingertip detection is a system that only works with fingers, and there are no devices like highlighters or any other related gadgets. Identifying and recognizing an object like a finger from an RGB image without an advanced device like a depth sensor is a great challenge.

2. Lack of pen up and down movement of the system uses an RGB camera to start writing. Since depth sensing is not possible, the pen's up and down movement cannot be tracked. Thus, tracking the entire trajectory of the fingertip results in an image that is meaningless and not recognized by the model. The difference identified between handwritten and air-written 'G' is shown in Figure 2.1 Canvas Based Drawing.

3. Real-time system control: Using real-time hand gestures to transition a system from one state to another requires a lot of code attention. In addition, the user must learn many movements to fully master the system.

**4. Problem Statement**

Who would have thought that raising a finger in the air could help us draw on a real picture? It's wonderful how this aerial web works in Computer Vision Projects.

**5. Problem Solution**

Computer projects help us draw on a screen easily by waving our fingers with an indicated color. OpenCV helps in more advancements in various ways of writing, such as using keyboards, touchscreen surfaces, digital pens, styluses, electronic gloves, and more. But in this system, we use hand gesture recognition with a machine learning algorithm using Python programming, creating a natural human-machine interaction. With advancements in technology, the need to develop natural human-computer interaction (HCI) systems to replace traditional systems is rapidly increasing.

Our project mainly focuses on solving major problems –

**1. Deaf people:** Although we take listening and hearing as easy tasks, communication using sign language is challenging for most of the world without a translator.

**2. Smartphone abuse:** Smartphones cause accidents, depression, distractions, and other illnesses that we humans can still detect. While their portability and ease of use are deeply admired, their downside includes life-threatening events.

**3. Paper waste:** It's not uncommon to waste a lot of paper when doodling, writing, drawing, etc.

1. **Features**

1.Can track any specific individual required color pointer.

2.Users can draw four different colours and even change them whenever required without any difficulties.

3.There is Clear option at the top of the display that helps to scrub the board at a time.

4.Once the program has started No need to involve the contact with the computer.

****

Fig-1: Structure of Air Canvas

1. **Methodology**

**1. Hand Gesture Recognition**

Hand gesture recognition and tracking are handled by the MediaPipe framework, while computer vision is handled by the OpenCV library. To track and recognize hand movements and hand tips, the program makes use of machine learning ideas.

**A. MediaPipe:**

MediaPipe is a Google open-source framework that was initially released in 2019. MediaPipe has some built-in computer vision and machine learning capabilities. A machine learning inference pipeline is implemented using MediaPipe. ML inference is the process of running real data points. The MediaPipe framework is used to solve AI challenges that mostly include video and audio streaming. MediaPipe is multimodal and platform independent. As a result, cross-platform apps are created using the framework. Face detection, multi-hand tracking, hair segmentation, object detection, and tracking are just a few of the applications that MediaPipe has to offer. MediaPipe is a framework with a high level of fidelity. Low latency performance is provided through the MediaPipe framework. It's in charge of synchronizing time-series data. The MediaPipe framework has been used to design and analyze systems using graphs, as well as to develop systems for application purposes. In the pipeline configuration, all of the system's steps are carried out. The pipeline that was designed can run on a variety of platforms and can scale across desktops and mobile devices. Performance evaluation, sensor data retrieval, and a collection of components are all part of the MediaPipe framework. Calculators are the parts of the system. The MediaPipe framework uses a single-shot detector model for real-time detection and recognition of a hand or palm. It is first trained for the palm detection model in the hand detection module since palms are easier to train. It designates a hand landmark in the hand region, consisting of 21 joint or knuckle coordinates as shown in the Figure 1. Fig -1: Coordinates or landmarks in the hand

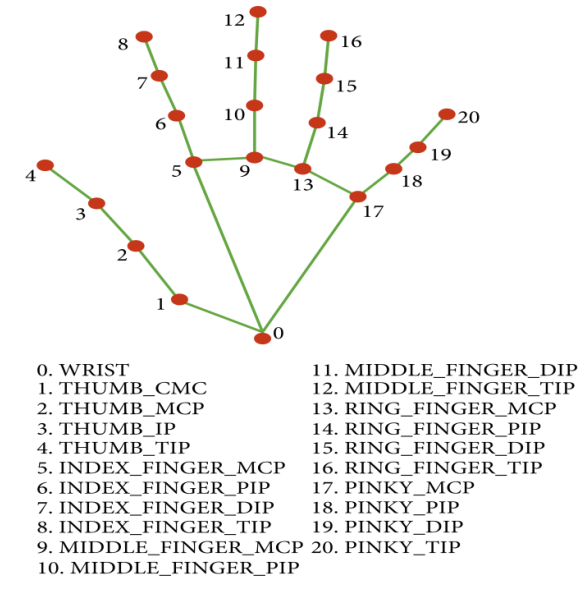


Fig -1: Coordinates or landmarks in the hand

**B. OpenCV:**

The computer vision library OpenCV is a must-have for everyone who works with computers. It includes object detection image-processing methods. OpenCV is a python package for creating real-time computer vision applications. Image and video processing and analysis are handled by the OpenCV library.

**2.Working of the Model**

The system requires a dataset for finger detection modelling. The primary purpose of the fingertip model is to record movement, with this recorded movement representing air characters.



Fig-2: Workflow of Air Canvas

**A. Fingertip sensor model:** Air writing can be done simply by using a single-color stylus or an air pen. However, the system uses fingertips. We believe that people can write in the air without having to wear a stylus. We used deep learning algorithms to detect the tip of the finger in each image, generating a list of coordinates.

Technique for creating finger recognition dataset:

Video to Frames: In this approach, two-second videos of a person's hand movements are captured in different regions. These videos are then divided into 30 individual images, as represented in diagram 3. This totals 2000 images.

This dataset was manually labelled using Labelling. The best model on this dataset yields 99% accuracy. However, since all 30 frames are produced from the same video and region, the dataset lacks diversity. Therefore, the model does not perform well for backgrounds different from those in the dataset.

**B. Take photos in separate backgrounds:** To overcome the lack of diversity in the previous method, we introduced a new dataset. This time, we are aware that we need a few gestures to control the system. So, we collected four different hand poses, shown in Figure 4. The main intention was to make the model effectively recognize the fingertips of the four fingers.

This allows the system to be controlled by the user based on the number of fingers they show. For example, they can now type quickly by showing their index finger, convert that writing motion into electronic text by showing two fingers, add space by showing three fingers, and press the backspace key by showing five fingers. They can also make consecutive predictions by showing four fingers and choose 1st, 2nd, or 3rd predictions by showing 1, 2, or 3 fingers, respectively. To exit prediction mode, they can show five fingers. This dataset includes approximately 1800 images. Using this script, the previous model automatically labelled this dataset. Next, we corrected mislabelled images and introduced another template. The accuracy reached 94%. Unlike before, this model works well in different contexts.

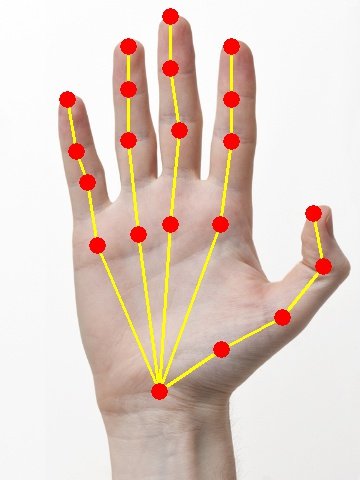


Fig-3: Detection of Finger Tips Using OpenCV

**C. Training the finger recognition model:** Once the dataset is ready and labelled, it is divided into training and development groups (85% to 15%). We used pre-trained faster Single Shot (SSD) and RCNN detector models to train our dataset. Fast RCNN is more accurate than SSD. For more information, we must verify the results. SSD combines two standard detection modules - one that suggests regions and one that classifies them. This speeds up the performance of objects detected in a single attempt and is usually used for real-time object detection. Faster RCNN uses the feature map output from Fast RCNN to compute area recommendations. They are evaluated by a network of regional recommendations and delivered to an area of interest group class. The final result is given as two fully connected classes for bounding box classification and regression. We adapted the final fully connected Fast RCNN layer for fingertip recognition in images.

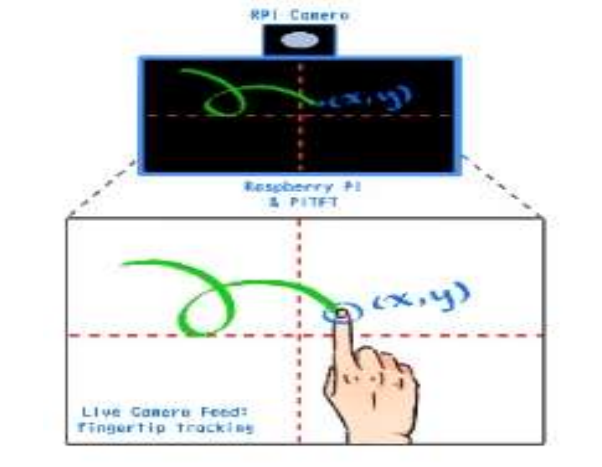
****

Fig-5: Training the Finger Recognition Model

1. **Algorithm**

This is the most interesting part of our system, where features are introduced due to writing. Therefore, the number of gestures used to control the system corresponds to the number of actions involved. The basic features we have included in our system are:

* **Writing Mode:** In this state, the system will draw the fingertip coordinates and store them.
* **Color Mode:** Users can change the text color among different colours available.
* **Erase Mode:** To erase any content, a erase feature is required, introduced through a specific gesture.

**Description Points to Develop an Air Canvas Using OpenCV:**

* Import and install necessary packages and files.
* Capture frames from the webcam.
* Invoke a canvas window for drawing.
* Detect and track the green color.
* Draw on the canvas window.

**9. Air Canvas: Flow Chart**

1. Start image playback and convert the captured image to HSV color space (easy to spot color).

2. Make the canvas and place the corresponding color buttons on it.

3. Adjust the trace bar values to find the color highlight mask.

4. Mask pre-treatment with morphological manipulations (erosion and dilation).

5. Detect the contours, identify the coordinates of the centre of the largest contour, and keep storing them in an array for the next frames (tables for drawing points on the canvas).

6. Finally, draw the points stored in the table.

This detailed process ensures a dynamic and interactive digital art experience using the Air Canvas system.

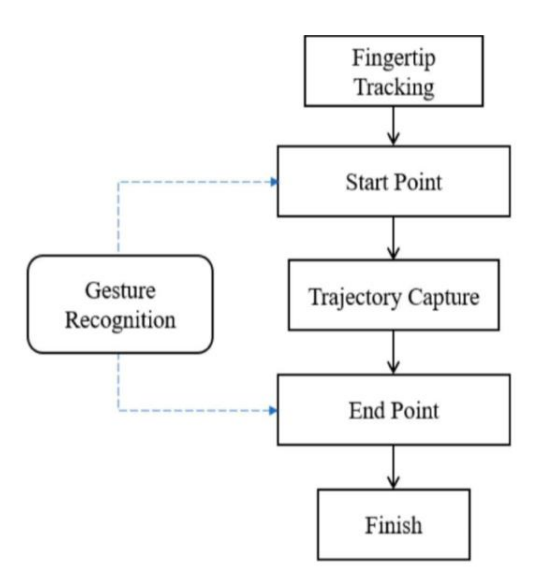
****

Fig-1: Flow Chart of Air Canvas

**10. Result and Discussion**

Here are the results and output for our project. We utilized the OpenCV module to execute our code. The code was run in VS Code, where we first installed the necessary packages and then executed our code to generate the output. Two windows are displayed: one window serves as our camera page where our pen is tracked using the coordinates specified in our code, and the second window is used for drawing the recorded coordinates. In this window, we can use functions like 'clear' or 'change the color'.

1. **Code output:** This is the output of the code which runs in VS Code, where we first installed the necessary packages and then executed our code to get the output.

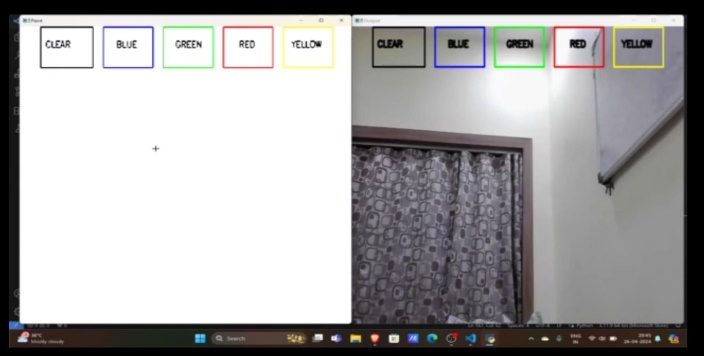


Fig-1: Code Output

1. **Detecting the Mask Contour of the Coloured Object:** After detecting the mask in the Air Canvas, it's time to identify its centre to draw the line. Below, we perform some morphological manipulation on the mask to eliminate impurities and make it easier to detect contours.

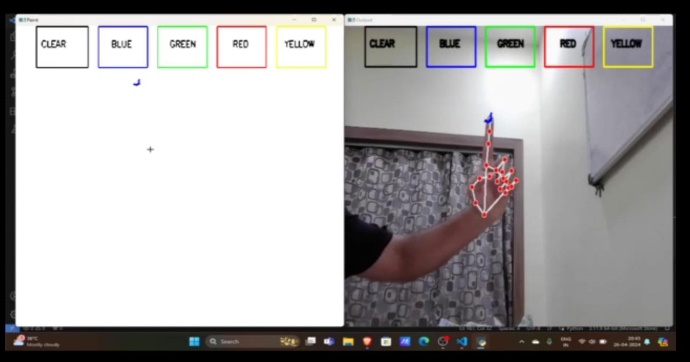


Fig-2: Detecting the Mask

1. **Selecting the colour using contour position:** We are selecting the colours using contour position to draw in the colour canvas.

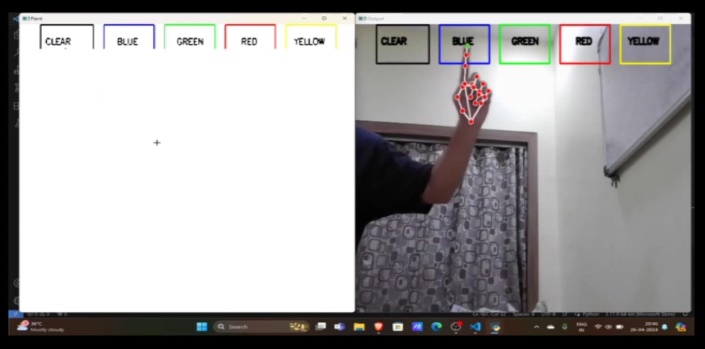


Fig-3: Selecting Colours

1. **Drawing the Line Using the Contour Position:** The real logic involved in this computer vision project; we will perform a Python deque operation. The deque will store the position of the contour from each frame, and we will use these points to draw lines using OpenCV functions. The position of the contour helps us decide whether to click a button or draw on the canvas.

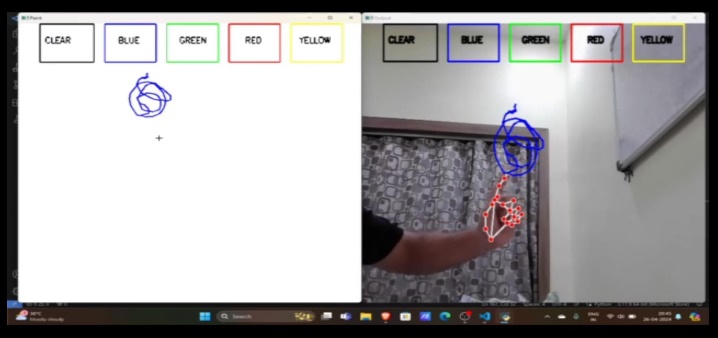


Fig-4: Drawing the Lines

1. **Erase the content:** This Figure is depicting the erasing mode of the content over the canvas. Index finger is hovered over clear to erase the content.

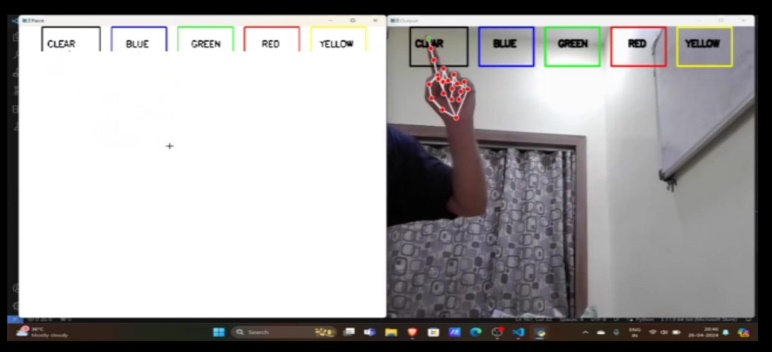


Fig-5: Erase the Content

1. **Future Scope**

This system will facilitate effective communication between people, reducing the reliance on laptops and mobile phones by eliminating the need for traditional writing methods. Its primary scope lies in the teaching field, especially for online teaching or screen-based instruction. With no requirement for a mouse or markers, it can be seamlessly implemented on screens. Additionally, it will find extensive use in design, enabling the creation of immersive and interactive designs.

**12. Conclusion**

The system has the potential to challenge traditional writing methods. It eliminates the need to carry a cell phone for note-taking, offering a convenient way to record information on the go. Moreover, it significantly aids individuals with special disabilities, enabling easier communication. The elderly and those who struggle with keyboards will find the system particularly user-friendly. With expanded functionality, the system can also swiftly control IoT devices. Users are able to draw in the air, making the system a valuable tool for smart devices, facilitating easier understanding and interaction with the digital world. Augmented reality can further enhance text, making it more dynamic.

Future improvements could address some current limitations of the system. First, adopting handwriting recognition instead of character recognition could enable users to write words seamlessly, thereby speeding up the writing process. Second, incorporating hand gestures with pauses for real-time control, as referenced in other studies, could offer a more intuitive alternative to using fingertips. Third, the system currently recognizes fingertips in their default state and modifies them. It is crucial that the air writing system responds solely to the owner's gestures and is not misled by those merely glancing at it. Moreover, the use of the EMNIST dataset, which is not ideally suited for air character recognition, could be reconsidered. Advanced object detection algorithms like YOLO v3 are expected to improve the accuracy and speed of fingertip recognition. Ultimately, future advances in artificial intelligence are likely to significantly enhance the efficiency of aerial writing.

**13. Reference**

[1] Air Canvas Application Using OpenCV, Mediapipe and Python

[https://www.irjet.net/archives/V8/i8/IRJET-V8I8258.pdf 2](https://www.irjet.net/archives/V8/i8/IRJET-V8I8258.pdf%202)

[2] Air Canvas using OpenCV, Mediapipe

https://www.irjmets.com/uploadedfiles/paper//issue\_5\_may\_2022/22247/final/fin\_irjmets165183606 8.pdf

[3] Virtual Paint Application using hand gesture

https://www.irjet.net/archives/V9/i4/IRJET-V9I4399.pdf

[4] Air Canvas: Draw in Air

[https://www.irjet.net/archives/V9/i2/IRJET- V9I2183.pdf](https://www.irjet.net/archives/V9/i2/IRJET-%20%20%20V9I2183.pdf)

[5] Virtual Painter Using Artificial Intelligence and OpenCV

https://www.irjmets.com/uploadedfiles/paper/issue\_6\_june\_2022/26805/final/fin\_irjmets1656128306.pdf

[6] Finger recognition and gesture based augmented keyboard

https://www.ijariit.com/manuscripts/v4i5/V4I5-1394.pd