Mouse Control Using Hand Gestures

Anush.J[1], Sunil kumar.G [2], M.Boobash Ayyanar [3],
Harish raj.M[4] ,Sai Kishore.R[5]
[1, 2, 3, 4, 5]II CC, SRM Institute of Science and Technology, Ramapuram

Abstract:

This program utilizes computer vision techniques and the PyAutoGUI library to enable mouse control through hand gestures captured by a webcam. It processes live video frames, identifies a yellow-colored object (typically a hand), and interprets specific hand gestures to control the mouse cursor. The system distinguishes between open and closed hand gestures to perform actions such as moving the cursor and simulating mouse clicks.

Introduction:

Mouse control using hand gestures is a practical and intuitive way to interact with a computer without physical input devices. This program combines computer vision, image processing, and input simulation to achieve this functionality.

Methodology:

1.Environment Setup:

- The program initializes the necessary libraries, including OpenCV for video processing, NumPy for array operations, wxPython for screen size retrieval, and PyAutoGUI for mouse control.

2.Webcam Initialization:

- It accesses the computer's webcam using OpenCV, allowing real-time video capture.

3.Color Detection:

- The program uses the HSV color space to isolate the yellow color, typically associated with a user's hand.
- It applies a lower and upper HSV color range (`lb` and `ub`) to create a mask that highlights the yellow object.

4.Noise Reduction:

- Morphological operations (opening and closing) are employed to reduce noise and ensure accurate hand detection.
- Kernels (`kernelOpen` and `kernelClose`) are applied to the mask to remove unwanted artifacts.

5.Contour Detection:

- Contours within the processed mask are identified using OpenCV's findContours function.
- The program selects the largest contour, which is expected to represent the hand.

6.Gesture Recognition:

- The system calculates the coordinates of the contour's extremities (west, east, north, south) and its center.
- It calculates the contour area (`bint`) as a basis for recognizing gestures.

7.Mouse Control:

- If the contour area falls within the range of 8000 to 18000, it is interpreted as an open hand gesture:
- The left mouse button is released (if it was pressed).
- The mouse cursor is moved to the calculated position based on the hand's center.
- If the contour area falls within the range of 2000 to 7000, it is interpreted as a closed hand gesture:
- The left mouse button is pressed.
- The mouse cursor is moved to the calculated position based on the hand's center.

8.Visualization:

- The processed video frames are displayed in a window using OpenCV, showing the detected hand contours and mouse cursor movement.

9.Exit Mechanism:

- The program continues running until the spacebar is pressed.
- Upon exiting, it releases the webcam and closes all OpenCV windows.

Results:

The program successfully recognizes hand gestures and uses them to control the computer mouse. Users can perform open and closed hand gestures to move the cursor and simulate mouse clicks. The system's responsiveness and accuracy depend on the lighting conditions and the user's ability to make distinct gestures.

Discussion:

This program demonstrates a basic implementation of mouse control through hand gestures. While it can be a fun and interactive way to control a computer, there are several limitations and areas for improvement:

1. **Lighting Conditions:**
   - ➢ The program's accuracy heavily relies on consistent and adequate lighting conditions, as it detects colors in the HSV space. Changes in lighting can affect gesture recognition.
2. **Gesture Variability:**
   - ➢ Users may find it challenging to consistently make precise gestures, leading to false positives or misinterpretations.
3. **Gesture Expansion:**
   - ➢ To enhance functionality, the program could be extended to recognize a broader range of gestures for additional actions, such as scrolling or specific keyboard shortcuts.
4. **Robustness:**
   - ➢ More advanced computer vision techniques, such as deep learning-based hand pose estimation, could be employed to improve the robustness of gesture recognition under various conditions.

Conclusion:

The program provides an introductory demonstration of mouse control using hand gestures, offering an engaging way to interact with a computer. While it has room for improvement in terms of accuracy and gesture diversity, it serves as a foundation for more advanced and robust gesture-based interfaces. Further developments could lead to innovative and intuitive computer interaction methods.

Algorithm:

1.Importing Libraries:

- Import necessary libraries, including OpenCV (`cv2`), NumPy (`numpy`), wxPython (`wx`), and PyAutoGUI (`pynput.mouse`).

2. Setting up Variables:

- Create a `Controller` object from `pynput.mouse` to control the mouse.
- Initialize a wxPython application (`wx.App`) to get the display size.
- Define the desired capture size (`capturex` and `capturey`) for the webcam feed.
- Create kernels for morphological operations (`kernelOpen` and `kernelClose`).

3.Open Webcam:

- Initialize the webcam using `cv2.VideoCapture(0).
- Set the capture dimensions to the desired size using `cap.set().

4.Define HSV Color Range.

- Define the lower (`lb`) and upper (`ub`) HSV color range to detect the yellow color.

5.Main Loop:

- Enter a continuous loop to capture frames from the webcam.
- Convert the captured frame to HSV color space (`cv2.cvtColor`).
- Create a mask by thresholding the HSV frame to extract the yellow color.
- Perform morphological operations (open and close) to reduce noise in the mask.

6.Contour Detection:

- Find contours in the processed mask using `cv2.findContours`.
- If contours are found:
- Identify the largest contour (likely representing the hand) based on contour area.

- Calculate the coordinates of the contour's extremities (west, east, north, south) and the center.
- Draw the contour, circles at extremities, and a center point on the frame.
- Calculate the contour area (`bint`) to determine if the hand is open or closed.

7.Mouse Control:

- ➢ If the contour area falls within a certain range (8000-18000), it is considered an open hand:
- ➢ Release the left mouse button (if pressed).
- ➢ Move the mouse cursor to the calculated position based on the hand's center.
- ➢ If the contour area falls within another range (2000-7000), it is considered a closed hand:
- Press and hold the left mouse button.
- Move the mouse cursor to the calculated position based on the hand's center.

8.Display the Video:

- Display the processed video frame with contours and mouse cursor movement.

9.Exit the Program:

- If the spacebar is pressed, break out of the main loop.
- Release the webcam and close all OpenCV windows (`cap.release()` and `cv2.destroyAllWindows().

This program continuously captures frames from the webcam, detects yellow-colored objects (likely a hand), tracks their movement, and controls the mouse cursor based on hand gestures (open or closed). It provides a basic example of using computer vision to interact with the computer.