Explaining Air Canvas Application Code

June 25, 2024

Introduction

This document explains the implementation details of an interactive Air Canvas application using Streamlit, OpenCV, and Mediapipe.

Python Code Explanation

1. **Streamlit Setup:**
   - Streamlit is configured to create a user interface for the Air Canvas application. This includes setting the page title, icon, and layout using `st.set_page_config`.
2. **Developer and Project Information:**
   - The sidebar (`st.sidebar`) displays buttons linking to developer profiles on LinkedIn (`developers` dictionary). Users can click these buttons to learn more about the developers.
   - Another button in the sidebar provides information about the Air Canvas project itself (`st.sidebar.button("Air Canvas Project")`).
3. **Instructions and Usage Tips:**
   - Instructions (`st.subheader("How to Use Air Canvas:")` and `st.write(...)`) guide users on setting up their webcam, interacting with the canvas using hand gestures, and utilizing various controls for color selection, clearing the canvas, and saving artwork.
4. **Mediapipe Initialization:**
   - `mpHands` and `hands` objects from Mediapipe are initialized for hand detection and tracking. `mpDraw` is used for drawing landmarks and connections on the captured video frames.
5. **Canvas Setup:**
   - The canvas (`paintWindow`) is initialized as a blank image (`np.zeros((471, 636, 3), dtype=np.uint8) + 255`) with predefined color selection buttons and text (`cv2.rectangle`, `cv2.putText`).
6. **Webcam Capture:**

- `cap` initializes the webcam (`cv2.VideoCapture(0)`), capturing frames (`ret, frame = cap.read()`) which are then flipped horizontally (`cv2.flip(frame, 1)`) and converted to RGB (`cv2.cvtColor(frame, cv2.COLOR`$_BGR2RGB$)).

7. **Hand Detection and Drawing:**
   - The captured frames (`framergb`) are processed by Mediapipe to detect hand landmarks (`result = hands.process(framergb)`). Landmarks are extracted (`landmarks`) and drawn on the frame (`mpDraw.draw_landmarks`).
   - The position of the index finger (`fore_finger`) and thumb (`thumb`) is tracked to determine the current drawing position (`center`) and brush size (`brush_size`) based on their distance.
8. **User Controls:**

- Sidebar controls (`st.sidebar`) allow users to select different colors (`colorIndex`), clear the canvas (`clear_button`), save artwork (`save_button`), and stop the application (`stop_button`).

9. **Drawing Functionality:**
   - Depending on the selected color (`colorIndex`), points are appended to respective deques (`bpoints`, `gpoints`, `rpoints`, `ypoints`) to track drawing movements. These points are then used to draw lines on both the frame and the canvas (`cv2.line`).
10. **Loop and Interaction:**
    - The application runs in a loop (`while cap.isOpened()`) continuously capturing frames from the webcam and updating the canvas based on hand movements and user interactions.
11. **Termination:**
    - The application terminates webcam capture and closes all windows (`cap.release()`, `cv2.destroyAllWindows(`) when the `stop_button` is clicked or the user exits the application.

References

- **OpenCV Documentation:** Explore OpenCV functionalities and learn how to manipulate images and video streams.
  - Website: https://opencv.org/documentation/
- **Mediapipe Documentation:** Understand Mediapipe's capabilities for hand detection, tracking, and other AI applications.
  - Website: https://mediapipe.dev/
- **Air Canvas Project:** Learn more about the Air Canvas project and its development details.
  - Website: https://github.com/your-air-canvas-repo