Human-robot Interaction (HRI) with Gesture Recognition

Final Presentation

Supervisors:
Prof. Dipl.-Inf. Ingrid Scholl
Christian Carlhoff

Students:
Sameer Tuteja,
Venkata Gopi Krishna Miriyala,
Florisa Zanier

Agenda

Topic Description

Gesture Recognition

Air Canvas Board

Overview

Functionalities

App Flow

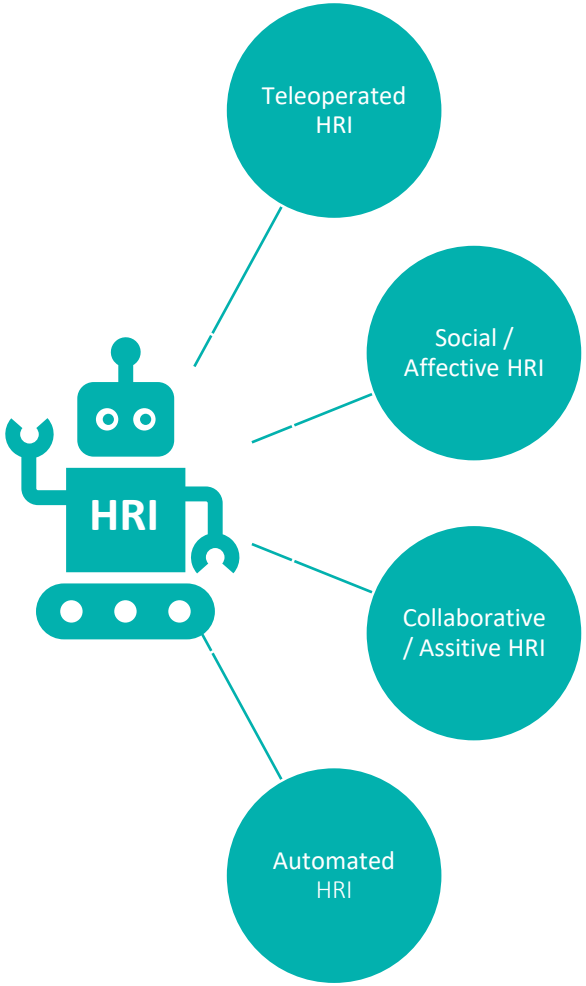
Use Cases

Live-Demo

Topic Definition

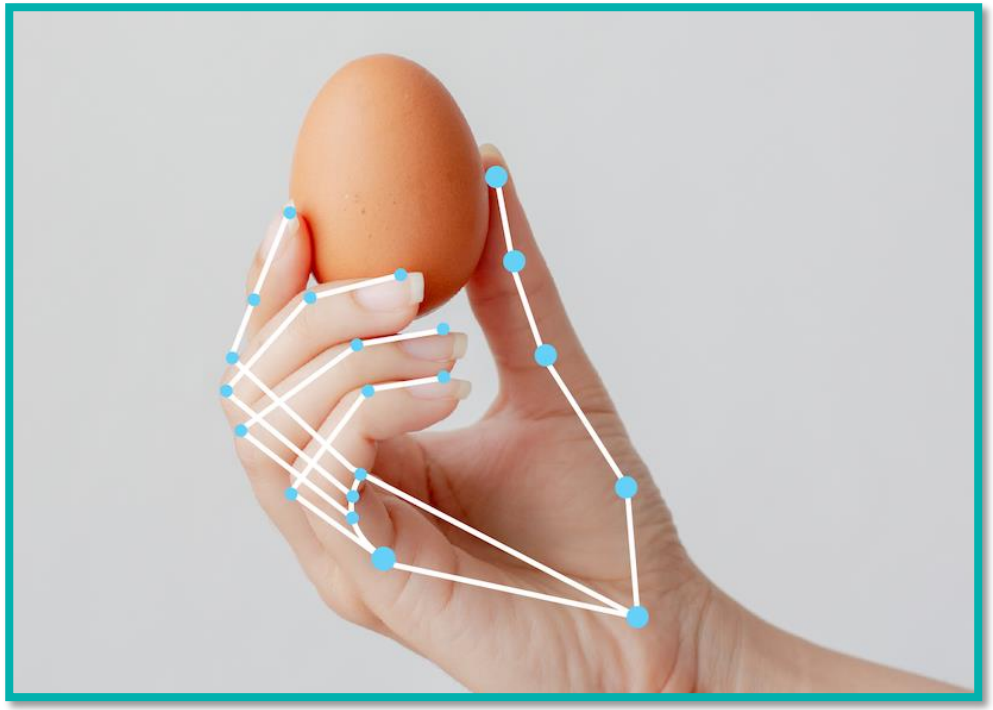
- **Definition of Human-Robot Interaction (HRI):**
  - It is a multidisciplinary field focusing on the dynamics of interaction between humans and robots.

- **HRI with gesture recognition:**
  - Developing systems enabling robots to interact in a human-like manner.
  - Enhancing gesture recognition to allow more natural and intuitive interactions.



Gesture Recognition – Hand Landmark Detection

- **MediaPipe's Hand Landmark Overview**
  - Developed by Google
  - Real-time hand tracking and gesture recognition.
  - Utilizes machine learning to identify hand landmarks.
  - Useful in augmented reality, sign language translation, and human-computer interaction.



Gesture Recognition – Hand Landmark I/O

Hand Landmark Model Features

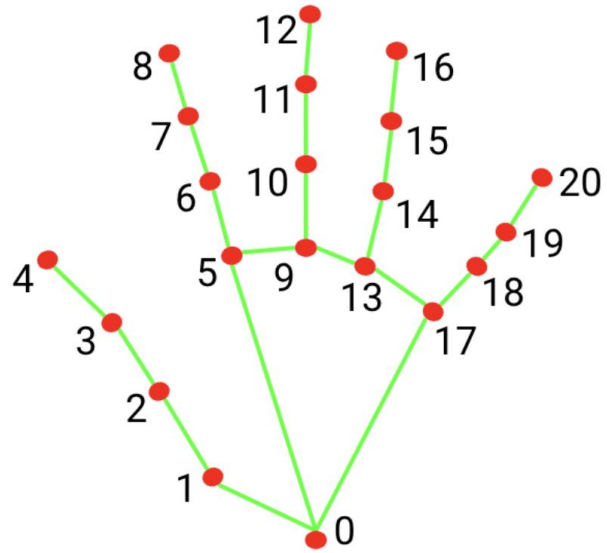
- **Input image processing:**
  - It includes image rotation, resizing, normalization, and color space conversion
- **Score threshold:**
  - Filter results based on prediction scores

Task Inputs	Task Outputs
The Hand Landmarker accepts an input of one of the following data types: ▪ Still images ▪ Decoded video frames ▪ Live video feed	The Hand Landmarker outputs the following results: ▪ Handedness of detected hands ▪ Landmarks of detected hands in image coordinates ▪ Landmarks of detected hands in world coordinates

Gesture Recognition – Hand Landmark Model Bundle

Hand Landmarker Model Bundle Overview

- Utilizes a palm detection model and a hand landmarks detection model.
- Detects keypoint localization of 21 hand-knuckle coordinates within detected hand regions.
- Palm detection model locates hands within the input image.
- Hand landmarks detection model identifies specific hand landmarks on the cropped hand image.
- In video or live stream mode, localizes hand region using bounding box defined by the hand landmarks model.
- Re-triggers palm detection model if hand landmarks model fails to identify or track hands, reducing tiggering times.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Gesture Classification Model

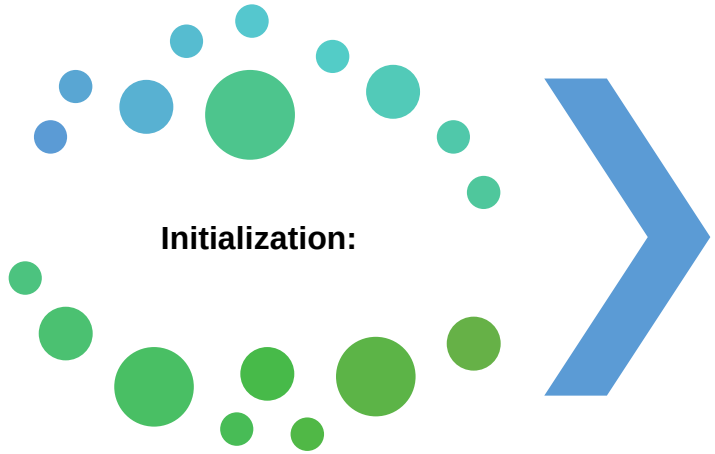
- The Gesture classification model bundle can recognize these common hand gestures:

```
0 - Unrecognized gesture, label: Unknown
1 - Closed fist, label: Closed_Fist
2 - Open palm, label: Open_Palm
3 - Pointing up, label: Pointing_Up
4 - Thumbs down, label: Thumb_Down
5 - Thumbs up, label: Thumb_Up
6 - Victory, label: Victory
7 - Love, label: ILoveYou
```

Classification Process

- Model detects hands but doesn't recognize gestures.
- Returns "None" if model doesn't detect hands.
- Two-step neural network pipeline: gesture embedding model and gesture classification model.
- Encodes image features into a feature vector.
- Classification model is a lightweight gesture classifier.
- Bundle includes canned gestures classifier for 7 common hand gestures.
- Custom gesture classifier can be trained to recognize more gestures.
- Gesture Recognizer prefers custom gesture if both classifiers recognize the same gesture.

Hand Landmarks Detection Process



Initialization:

MediaPipe Hands is initialized with certain parameters: **hands = mp_hands.Hands(min_detection_confidence=0.5, min_tracking_confidence=0.5)**

This creates a "Hands" object with minimum detection and tracking confidence thresholds set to 0.5.

These thresholds dictate how confident the model should be in the detection and tracking process to consider a hand as detected or being tracked.

Processing Frames

In the main loop, frames from the webcam are captured using OpenCV: **ret, frame = cap.read()**

Each frame is converted from BGR (OpenCV's default color format) to RGB because MediaPipe requires RGB format: **frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)**

The converted frame is then processed by MediaPipe to detect hand landmarks: **results = hands.process(frame_rgb)**

The process method of the "Hands" object detects hands in the image and returns the landmark data.

Hand Landmarks Detection Process

The returned results contain the landmark data.

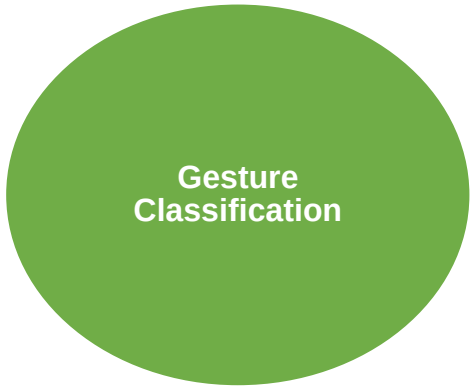
Each hand detected in the image is represented as a list of 21 landmarks, each with x, y, and z coordinates (3D)

The result will be a list of hands detected, with each hand having its own set of landmarks.

Landmark Visualization

In our script, these landmarks are drawn on the frame using the mp_drawing module for visualization:
mp_drawing.draw_landmarks(frame, hand_landmarks, mp_hands.HAND_CONNECTIONS)

This line draws the landmarks and the connections between them on the original frame.



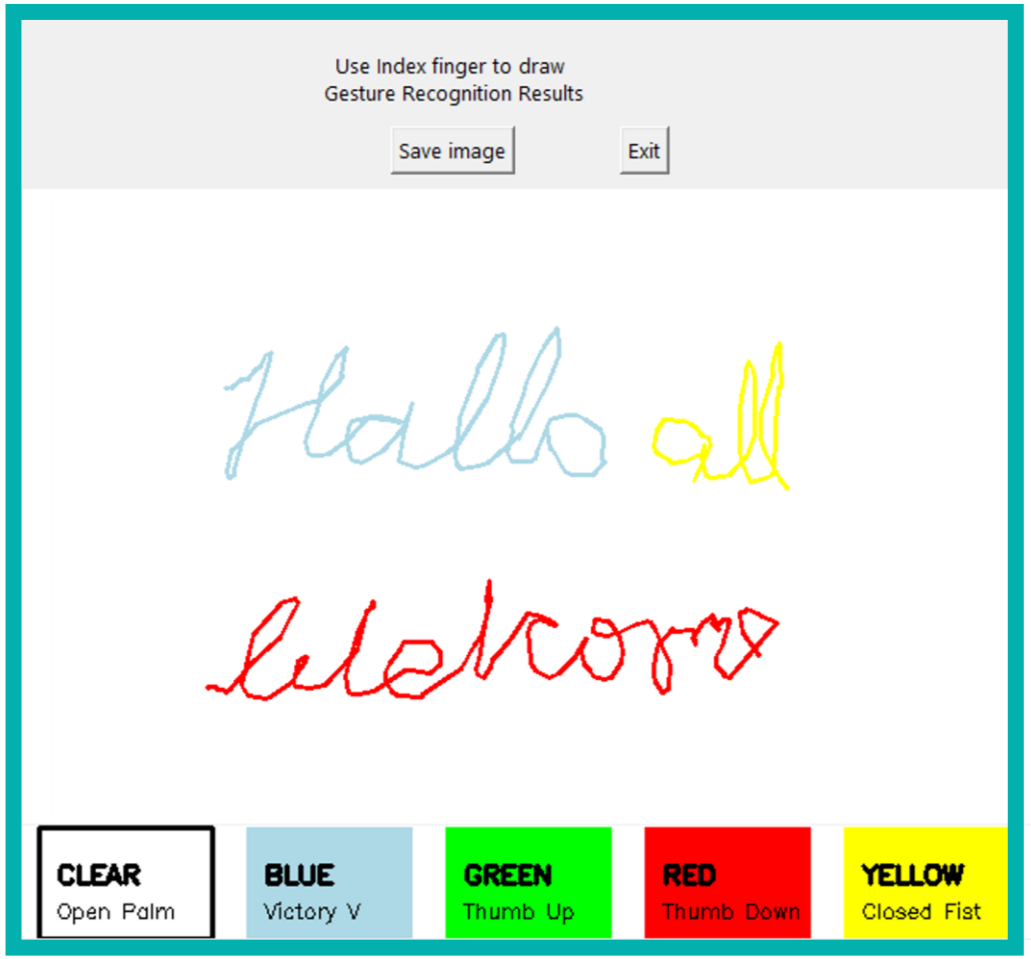
Apart from visualization, these landmarks are used to classify hand gestures.

By calculating angles between different landmarks (fingertips), the type of gesture being made is determined.

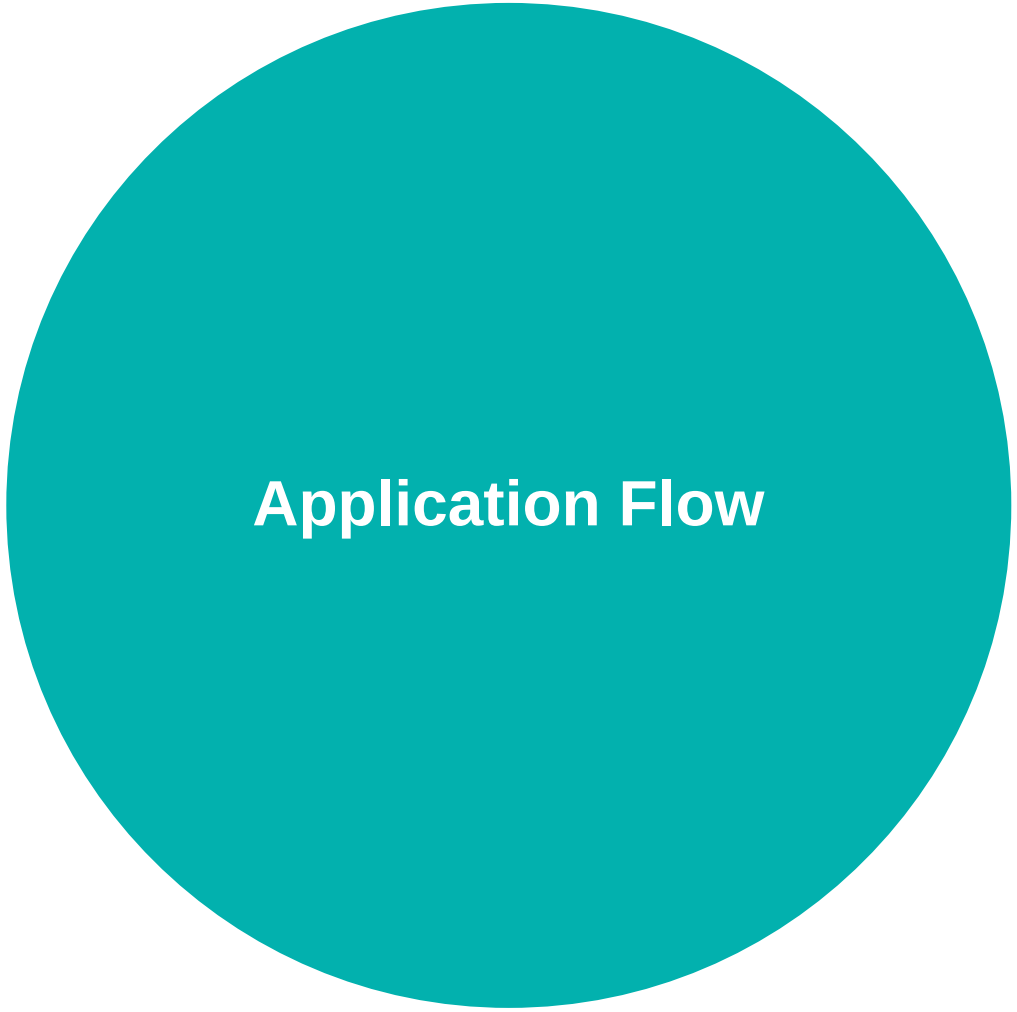
Air Canvas Board – Application Functionality

Functionalities

- **OpenCV and MediaPipe**: Used for real-time hand tracking and gesture recognition. MediaPipe Hands is employed to detect and track hand landmarks in the webcam feed processed by OpenCV.
- **Gesture Recognition and Classification**: The application includes a method to classify hand gestures based on the angles between fingertips.
- **Tkinter GUI**: Provides a graphical interface for the application, displaying the webcam feed and the virtual canvas where users can draw using hand gestures.
- **Drawing Functionality**: Users can draw on the canvas by pointing up with their index finger. Different gestures change the drawing color or clear the canvas.
- **Audio Feedback**: Using pyttsx3, the system gives voice feedback on the selected color when a color-changing gesture is recognized.
- **Saving Functionality**: Users can save the drawn image by clicking a button.



Air Canvas Board – Application Flow



Initialization: Sets up the MediaPipe model, OpenCV for video capture, and initializes the speech engine.

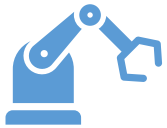
Gesture Recognition: Detects and labels gestures using the webcam and MediaPipe.

User Interface: Displays a window where the user can interact and receive feedback.

Voice Feedback: Speaks out certain actions or gestures recognized.

Image Processing: Processes and saves images based on gestures or commands.

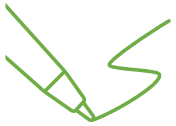
Use Cases



Gesture-Controlled Applications: For controlling software or devices through hand gestures.

Educational Tools: Teaching about computer vision and gesture recognition.

Accessibility Tools: Assisting users with disabilities by enabling gesture-based commands.



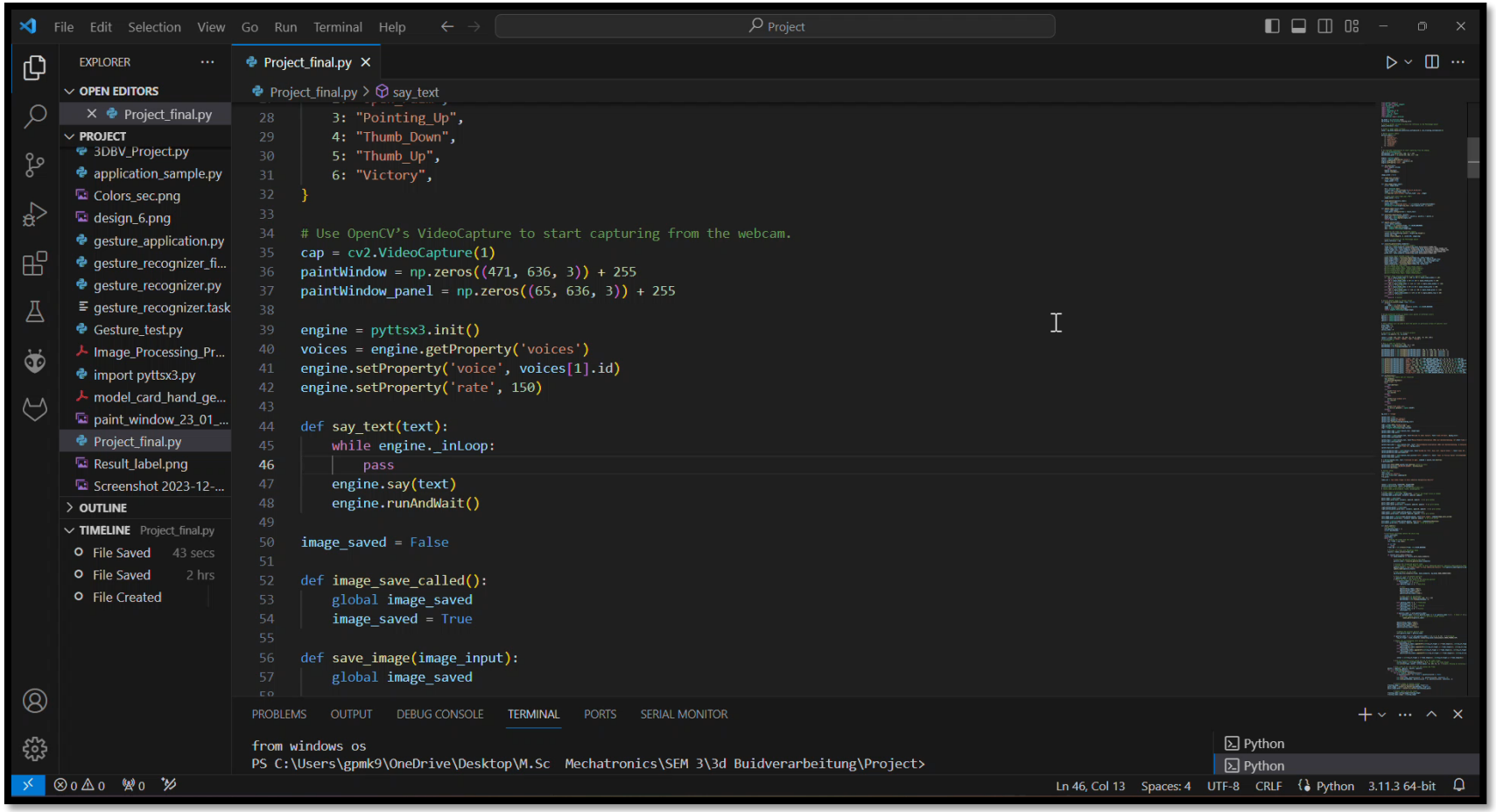
Handwriting detection: Involves the use of technology to recognize and interpret the shapes and patterns of handwritten characters and words.

Live Demo

GitHub

Paper

Sample Video



Thank you