Mask-Aware Smart Camera Dolly with Real-Time Alerts (MASCARA)

(Arduino X OpenCV X TensorFlow)

The Mask-Aware Smart Camera Dolly with Real-Time Alerts (MASCARA) project aims to develop an intelligent system capable of detecting mask compliance in real-time and providing alerts when individuals are not wearing masks. The system leverages OpenCV, MediaPipe, and machine learning for real-time detection and utilizes a stepper motor for smooth camera movement. The system is designed to autonomously follow individuals while ensuring they wear masks, emitting warnings and transmitting in case of non-compliance.

Project Phases (Previously proposed):

- ✓ Phase 1: Hardware Setup and Basic Functionality (Arduino & Stepper Motor with webcam)
- ✓ Phase 2: Face and Body Tracking with OpenCV and MediaPipe (Arduino & python)
- ✓ Phase 3: Mask Detection Integration with Machine Learning (Arduino & python)

Current Project Flow:

- ✓ Phase 1.1: Face and Body Tracking with OpenCV and MediaPipe
- ✓ Phase 1.2: Mask Detection Integration with Keras TensorFlow
- ✓ Phase 2: Fully integrated Mask Detection system with MediaPipe and TensorFlow
- ✓ Phase 3: Serial data communication with Arduino
- ✓ Phase 4: Ensemble packages of MASCARA

1. **Creating a Mask Detecting Model:** This step involves developing a machine learning model capable of accurately detecting whether a person is wearing a mask or not.
2. **Building a Mask Detection System:** It is integrated into a real-time system capable of analyzing live video feed to perform mask detection on detected faces.
3. **Arduino Integration:** Arduino microcontroller is utilized to control a stepper motor attached to a camera dolly to work on facemask detection. Camera rotates horizontally, following the movement of individuals within the camera's field of view.
4. **Arduino Serial Communication:** The system utilizes serial communication between MediaPipe and Arduino via pyserial to share cartesian coordinates gathered from masked

face movements as well as the general string which states Mask or No-Mask which can later be converted into Boolean for ease of program.

Creating Mask Detecting Model: Used a Python script for training a face mask detection model using TensorFlow and Keras. It utilizes transfer learning with the MobileNetV2 architecture pre-trained on ImageNet.

https://colab.research.google.com/drive/17bwb-mp8rLLZB7y0p7G6qxlIEwuvGgPE?usp=sharing

The code loads face mask images from directories, preprocesses them, and splits them into training and testing sets. It then constructs a convolutional neural network (CNN) model with MobileNetV2 as the base and adds custom fully connected layers on top. The model is compiled and trained using data augmentation techniques. Finally, it evaluates the model's performance and plots training/validation accuracy and loss curves.

1. Import necessary libraries and dependencies.
2. Load face mask images from directories and preprocess them.
3. Define data augmentation techniques.
4. Build the MobileNetV2 base model without the classification head.
5. Add custom fully connected layers on top of the base model.
6. Freeze the layers in the base model and compile the model.
7. Train the model on the training data with data augmentation.

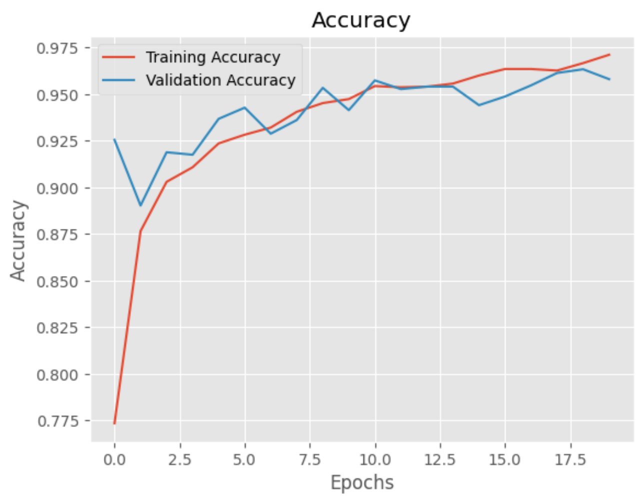

Figure 1.1: CNN Accuracy on detecting Mask.

Accuracy is the ratio of correctly predicted observations to the total observations. The overall accuracy of the model is 0.99, indicating that 99% of the images in the test set were correctly classified.

With Mask Precision: 0.98

Without Mask Precision: 1.00

Building a Mask Detection System: The script integrates the MediaPipe library for face detection and pose estimation, and a pre-trained mask detection model based on MobileNetV2.The Python script utilizes computer vision and machine learning techniques to detect faces in a live video stream and determine whether the detected faces are wearing masks or not. It reads frames from a webcam (Can change by defining src ={0,1,2...} based on the devices connected to the notebook), detects faces using MediaPipe, predicts mask presence using the loaded model, and displays the results on the video stream. Additionally, it sends face coordinates and mask detection status to an Arduino device for further processing.

1. Imports and Library Setup: Import necessary libraries and setup configurations for MediaPipe, OpenCV, TensorFlow, and serial communication.
2. Model and Video Stream Initialization: Load the face and pose detection models, mask detection model, and initialize the video stream from the webcam.
3. Main Loop: Loop over frames from the video stream.
4. Face Detection and Mask Prediction: Use MediaPipe to detect faces, predict mask presence using the loaded model from MobileNetV2, and annotate the frame with the prediction results.
5. Display and Communication: Display the annotated frame with face coordinates and mask detection status and send this information to Arduino for further processing.
6. Cleanup: Close all OpenCV windows and release video stream resources after the loop ends.

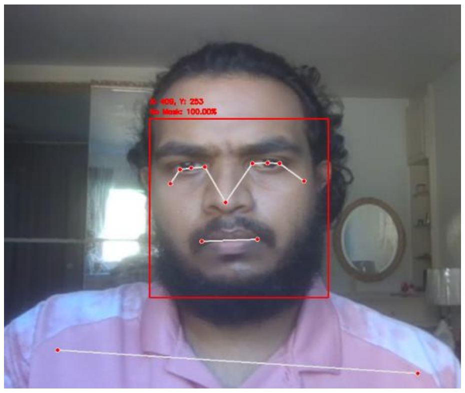

Figure 2.1: No Mask Face Detection

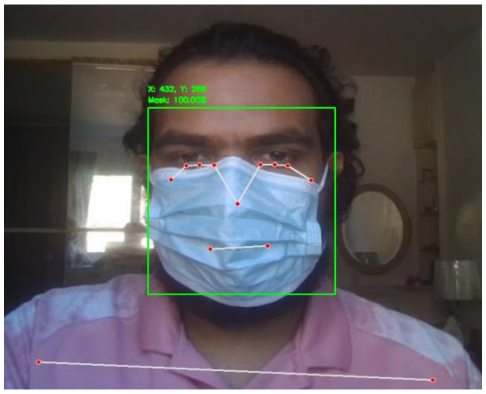

Figure 2.2: Mask Detected Face Snap

Arduino Integration: Arduino sketch controls a stepper motor based on input received from a serial connection via previous python script to follow the head movements. It also displays mask detection status on an LCD and controls an LED accordingly.

1. Library and Pin Definitions: The sketch includes necessary libraries for stepper motor control and LCD and defines pin assignments for the stepper motor coils and LED.
2. Stepper Motor Setup: An instance of the AccelStepper class is created for controlling the stepper motor in half-step mode. Maximum speed and acceleration parameters are set for smooth motor movement.
3. Main Loop:
    a. *Serial Input Processing:* Once python script sends data to serial port, it reads the input string until a newline character is encountered.
    b. *Coordinate and Mask Status Extraction:* The input string is parsed to extract the X coordinate and mask detection status (either "M" for mask detected or "N" for no mask detected).
    c. *Coordinate to Angle Conversion:* The X coordinate is mapped to an angle suitable for the stepper motor movement.
    d. *Stepper Motor Control:* The stepper motor is moved to the specified angle using the moveTo() function to follow up the face movements. Left or Right head movements of detected face sends X coordinates in positive or negative manner to rotate the stepper motor both ways.
    e. *Mask Detection Display:* The LCD displays the mask detection status message based on the received status.
    f. *LED Control:* The LED is turned on or off based on the mask detection status (No mask glows Red).
4. Stepper Motor Movement: Continuously update the stepper motor movement.

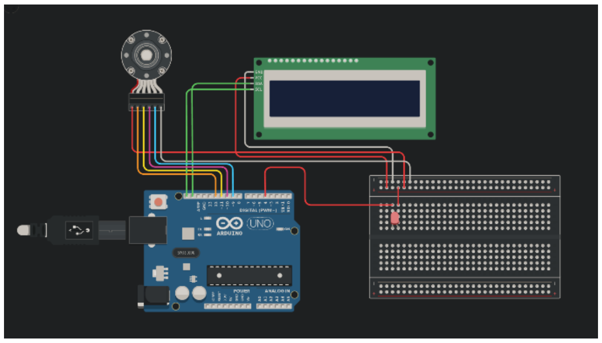

Figure 3.1: Schematic Circuit Diagram of Arduino design which gives output of the integrated system.

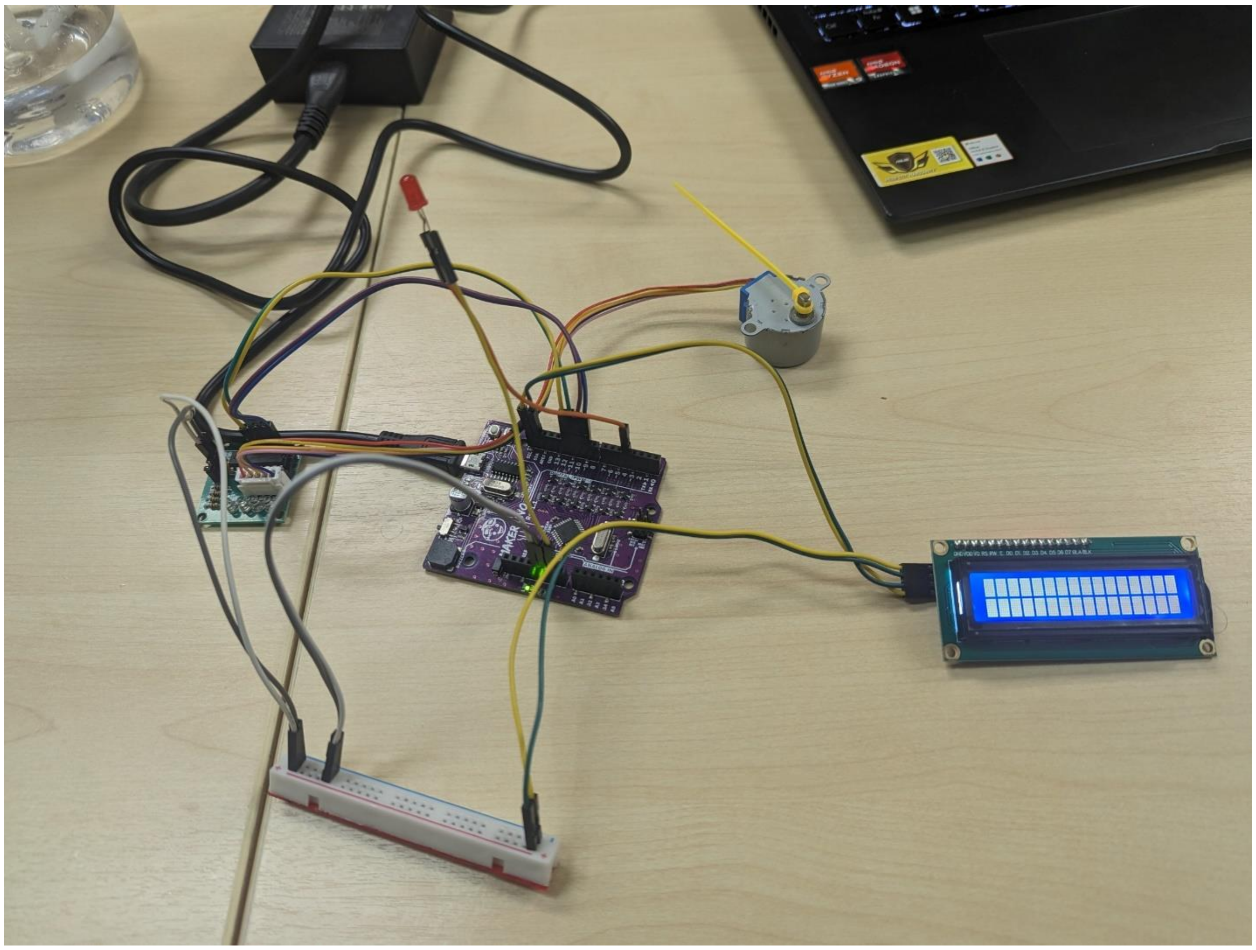
Figure 3.2: Arduino Circuit Diagram of Real Integrated System collaborated with MediaPipe