Group - 11

Soumava Paul 16EE10056
Yerramsetty Rohit 16EE10055

Keyboard Control using Gesture Recognition

Aim:

Implementing remote keyboard control using gesture recognition through Ultrasonic Sensors

Apparatus Required:

● Arduino Uno Board
● PC with Arduino IDE, Python, pyserial and pyautogui installed
● Ultrasonic Sensor x2
● Jumpers

Description and Procedure:

In this project, we have implemented an Arduino-based hand gesture recognition method for remote control of a computer keyboard. As examples, we demonstrate some basic functionalities like switching between applications, switching between browser tabs, scrolling up/down in a web page or document, playing or pausing a video, changing volume and taking screenshots. Python is used for serial communication between Arduino serial port and the computer using the PySerial library. The keyboard functionalities are implemented using the PyAutoGUI library.

Using a single ultrasonic sensor, we can implement a maximum of three hand gestures using the following conditions-
1) Placing our hand in front of a sensor and moving it towards the sensor.
2) Placing our hand in front of a sensor and moving it away from the sensor.
3) Swiping our hand in front of the sensor.

Using two ultrasonic sensors, it is possible to implement a total of eight gestures, three for each of the sensors and two more for swiping the hand across both sensors (from Sensor1 to Sensor2 and the other from Sensor2 to Sensor1).

After sensing the gesture, a particular data or command for that gesture is sent to the Python program running in the background.

- Gesture 1: Place the hand in front of sensor 1 and move towards the sensor. This Scrolls the webpage down and also decreases the volume when in VLC media player.
- Gesture 2: Place the hand in front of sensor 1 and move away from the sensor. This Scrolls the webpage up and also increases the volume when in VLC media player.
- Gesture 3: Swipe the hand in front of sensor 1. This navigates the user from current tab to previous tab in a web browser.
- Gesture 4: Swipe the hand in front of sensor 2. This navigates the user from current tab to next tab in a web browser.

- Gesture5: Swipe the hand in front of sensor 1 to Sensor 2 immediately. This navigates from current application to next application.
- Gesture6: Swipe the hand in front of sensor 2 to Sensor 1 immediately. This command takes the screenshot.
- Gesture7: Place the hand in front of sensor 2 and move towards the sensor. This command plays or pauses a video in VLC.
- Gesture8: Place the hand in front of sensor 2 and move away from the sensor. This command mutes the volume in a video in VLC.

Codes:

C Code on Arduino IDE

```
const int trigPin1 = 11;

const int echoPin1 = 10;

const int trigPin2 = 6;

const int echoPin2 = 5;

long duration;

int distance1, distance2;

float r;

unsigned long temp=0;

unsigned long temp1=0;

int l=0;


void find_distance (void);


void setup(){

  Serial.begin(9600);

  pinMode(trigPin1, OUTPUT);

  pinMode(echoPin1, INPUT);

  pinMode(trigPin2, OUTPUT);

  pinMode(echoPin2, INPUT);
  delay(100);
}

void find_distance (void){
  digitalWrite(trigPin1, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin1, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin1, LOW);
  duration = pulseIn(echoPin1, HIGH, 5000);
  r = 3.4 * duration / 2;
  distance1 = r / 100.00;
  digitalWrite(trigPin2, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin2, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin2, LOW);
  duration = pulseIn(echoPin2, HIGH, 5000);
  r = 3.4 * duration / 2;
  distance2 = r / 100.00;
  delay(100);
}

void loop(){
  find_distance();
  if(distance2<=35 && distance2>=15){
   temp=millis();
   while(millis()<=(temp+300)){
     find_distance();
   }
   if(distance2<=35 && distance2>=15){
     while(distance2<=50 && distance2>=10){
       temp=distance2;
       find_distance();
       if((temp+3)<distance2){
         Serial.println("up2");
         }
       else if((temp-3)>distance2){
         Serial.println("down2");
       }
     }
   }
 else{
   Serial.println("next2");
 }
}
 if(distance1<=35 && distance1>=15){
   temp1=millis();
   while(millis()<=(temp1+300)){
     find_distance();
     }
   if(distance1<=35 && distance1>=15){
      while(distance1<=50 && distance1>=10){
        temp1=distance1;
        find_distance();
        if((temp1+3)<distance1){
          Serial.println("up1");
          }
        else if((temp1-3)>distance1){
          Serial.println("down1");
        }
      }
    }
  else{
    Serial.println("next1");
  }
  }
}
```

Python Code for Serial Communication and Remote Keyboard Control

```
import serial
import pyautogui

Arduino_Serial = serial.Serial('/dev/ttyACM1',9600)
X='a'

while 1:

            incoming_data = str(Arduino_Serial.readline())
```

```
print(incoming_data)
if 'next2' in incoming_data :
        pyautogui.hotkey('ctrl', 'pgdn')
if 'next1' in incoming_data :
        pyautogui.hotkey('ctrl', 'pgup')
if 'up2' in incoming_data:
            pyautogui.press('volumemute')
if 'down1' in incoming_data:
        pyautogui.scroll(-100)
if 'up1' in incoming_data:
        pyautogui.scroll(100)
if 'down2' in incoming_data :
        pyautogui.press('space')
if ('next2' in incoming_data) & ('next1' in X):
        pyautogui.keyDown('alt')
        pyautogui.press('tab')
        pyautogui.keyUp('alt')
if ('next1' in incoming_data) & ('next2' in X):
        pyautogui.keyDown('win')
        pyautogui.press('prtsc')
        pyautogui.keyUp('win')
X=incoming_data
incoming_data = "";
```

<u>Links</u>

The video demonstration and the arduino and python code files are available in the following links:

https://drive.google.com/open?id=1TZ1757snap9b3eke4JI1UzGl9czaQN-z

https://drive.google.com/open?id=1ma5D17bz1ubTvvfHcfSO0xsEezA1IfVe

https://drive.google.com/open?id=1QmlBweA3VFtUD0PWxAarmo9ftpLHjOqk

https://drive.google.com/open?id=1zSfhlVZSXYulzVbcYbrxl53yr9IXC0xg

<u>Conclusion</u>

The algorithm described here needs a little bit of finetuning for some of the gestures. We have demonstrated five of the eight gestures mentioned in the 2 videos shared above. This project can be easily extended to some more complex keyboard functionalities using the PyAutoGUI library. Also, introducing some more ultrasonic sensors would increase the number of gestures recognized successfully by the system, but may introduce some complexities from the user point of view.