Indian Institute of Technology Kanpur

CS-724 Sensing Communications and Networking for Smart Wireless Devices

Project Report

Gesture Controlled Multimedia System

Submitted by:
Sevak Shekokar (241110065)
Rohan (241110057)
Krishna Kumar Bais (241110038)
Major Rajan Kumar (241110087)
Tsewang Namgail (241110093)
Rishit Kumar (241110056)
Tsewang Chukey (241110092)

Introduction

The project aims to develop a camera control system that operates entirely through hand gestures and sound commands. By using device-free sensing technology, the system allows for controlling various camera operations, such as capturing photos, recording videos, and zooming, without requiring wearable devices. This system is designed to be intuitive and efficient, making it particularly useful in situations where hands-free control is essential, such as in medical, research, security, and military applications.

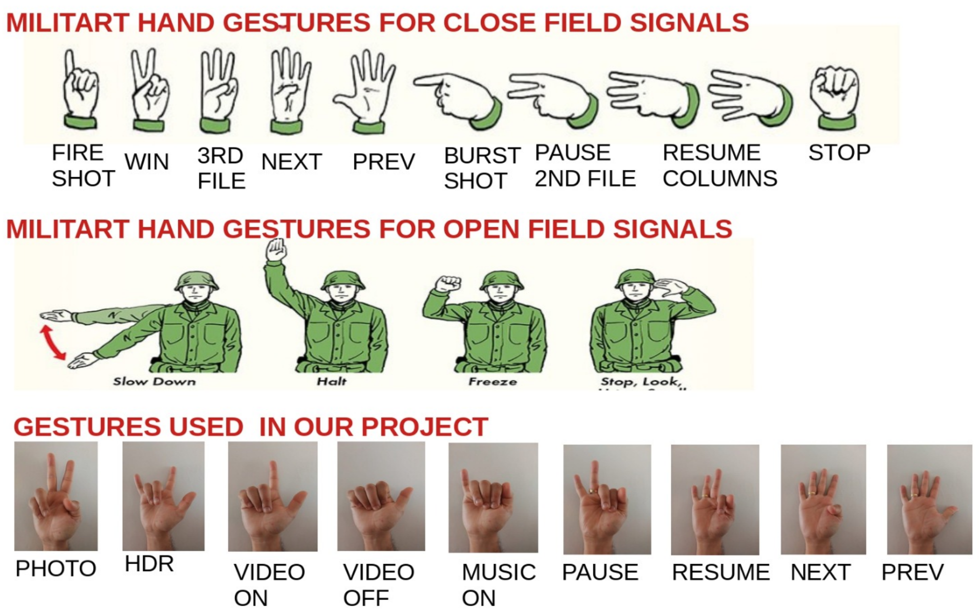
The increasing dependence on cameras across different fields, from professional photography to security surveillance, necessitates an evolution in how we control these devices. Traditional camera operation requires manual handling, which can be impractical in certain environments. In fields such as the medical field, where maintaining sterility is critical, or in situ experiments conducted in hazardous or remote locations, the ability to control a camera without physically interacting with it becomes vital. Similarly, video conferencing setups and security operations demand quick and seamless camera control without disruption. This project addresses these challenges by designing a camera control system that relies solely on hand gestures and voice commands. By using device-free sensing methods, the system can capture and interpret hand movements and sound signals to control the camera, offering an effective solution in scenarios where traditional methods are not feasible.

Project Aim

The goal of this project is to create a camera control system that operates entirely through hand gestures and sound signals. The system uses device-free sensing technology to detect hand movements and audio inputs, enabling camera operations such as capturing photos, recording videos, and zooming—without the need for any wearable devices.

Project Concept

The project takes a cue from the military hand gestures, utilized during close combat opera-tions. These gestures have been formulized and incorporated in security forces since ages. These gestures are being used in armies across the globe with minor modifications. We have incorporated these gestures in our project in order to simulate the real-world use case.



Apart from this we also have used voice commands to enable the camera operation through our voice.

Project Timeline

September: Research and Conceptualization: Conducted background research on camera and voice control systems and studied existing technologies and libraries related to gesture recognition and voice commands.

- **1–10 October:** Procurement of Parts: Procured the necessary hardware components including the camera, microphone, ultrasonic sensor, and Raspberry Pi. We also tested their basic functionality in this phase
- **11 October–3 November:** Development, Training, and Testing: Focused on integration of the hardware and its associated software components, training the gesture recognition model, and testing the system's functionality.
- **4–10 November:** Code Optimization: Refined the code for better performance, ensuring real-time response for hand gestures and voice command recognition.
- **11–20 November:** Deployment, Finalization, and Report Making: Final Integration of all the system components, performed final testing, and prepared the project report.

System Design and Methodology

The system architecture integrates various components to detect and process hand gestures and sound signals, providing seamless camera control.

Components Used

- **Microphone Module (Lenovo 300 FHD Webcam):** The system architecture integrates various components to detect and process hand gestures and sound signals, providing seamless camera control.
- **Camera Module (HP Webcam w300):** The primary device for capturing images and videos based on gesture and voice input commands.
- **Speaker System (XSound GO):** Provides audio feedback to the user for confirming voice command actions and alerts.
- **Raspberry Pi Model 4B:**Acts as the central processing unit for controlling the entire system and running the gesture and voice recognition algorithms.
- **Ultrasonic Sensor (HC-SR04):** Helps in detecting the proximity of the user's hand for gesture recognition.
- **Power modules and other connectors:** Used for providing power to the components and ensuring proper connectivity.
- **Breadboard for Integration:** Used to assemble and connect the various components in a stable manner for prototyping.

System Workflow

1. **Hand Gesture Detection:** The system utilizes the Raspberry Pi and ultrasonic sensor to detect hand movements in the camera's field of view. The Mediapipe library, used for hand detection, processes the ges-tures.
2. **Voice Command Recognition:** The microphone captures voice commands, which are interpreted through speech recognition software to trigger actions such as capturing photos or videos.
3. **Camera Operations:** Upon receiving inputs from either the hand gesture or voice command, the system sends sig-nals to the camera module (HP webcam) to perform actions such as capturing images, zoom-ing in, or recording video
4. **Code:**Attached as appendix

Team Contribution

Name	Role	Roll Number
Rishit	Code integration logic, research and code optimization	241110056
Krishna	Hand detection code using Mediapipe, testing	241110038
Chukey	Code for photo capture and related functions, documentation	241110092
Namgail	Music code, testing	241110093
Major Rajan Kumar	Voice commands code, documentation	241110087
Sewak	Circuit design and code optimization	241110065
Rohan	SOS feature coding, GitHub file upload and testing	241110057

Results and Discussion

The system underwent various tests to evaluate its performance across multiple environments, including different lighting conditions and background noise levels. Key performance metrics such as response time, accuracy of gesture detection, and voice command recognition were assessed and suitably corrected.
Challenges included tuning the system for varying hand gestures and improving the accuracy of voice command recognition. These issues were resolved through optimization of algorithms and testing in diverse environments.

Conclusion and Future Scope

This project successfully developed a camera control system based on hand gestures and voice commands, offering an intuitive and hands-free solution for camera operations. The system's versatility makes it ideal for environments where traditional manual control is impractical or impossible.
Future enhancements could include the use of machine learning to further adapt the system to a wider variety of gestures and voice commands.

Appendix A

Code

The code for the project is included below:

```
[ ]: import cv2
     import time
     import asyncio
     import requests
     import base64
     import mediapipe as mp
     import RPi.GPIO as GPIO
     import pygame
     from gtts import gTTS
     import time
     import os
     import pyttsx3
     import os
     import subprocess
     from datetime import datetime
     from pocketsphinx import LiveSpeech
     import speech_recognition as sr



     GPIO.setmode(GPIO.BCM)
     GPIO_TRIGGER = 18
     GPIO_ECHO = 24
     GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
     GPIO.setup(GPIO_ECHO, GPIO.IN)
     pygame.mixer.init()
     engine = pyttsx3.init()

     def distance():
         GPIO.output(GPIO_TRIGGER, True)
         time.sleep(0.01)
         GPIO.output(GPIO_TRIGGER, False)
```

```python
    StartTime = time.time()
    StopTime = time.time()
    while GPIO.input(GPIO_ECHO) == 0:
        StartTime = time.time()
    while GPIO.input(GPIO_ECHO) == 1:
        StopTime = time.time()
    TimeElapsed = StopTime - StartTime
    distance = (TimeElapsed * 34300) / 2
    #print("Distance:", distance, "cm")
    return distance

def func_voice_warning(text):
    engine.say(text)
    engine.runAndWait()

def load_song(index):
    global current_song_index
    current_song_index = index % len(playlist)
    pygame.mixer.music.load(playlist[current_song_index])


def func_take_photo():
    global last_capture_time
    global frame
    global cam_busy
    global capture_interval
    current_time = time.time()
    if current_time -  last_capture_time >= capture_interval:
        func_voice_warning("Capturing photo")
        timestamp = time.strftime("%Y%m%d_%H%M%S")
        image_filename = f"captured_{timestamp}.png"
        cv2.imwrite(image_filename, frame)
        print(f"Captured photo: {image_filename}")
        last_capture_time = current_time

def func_take_photo_hdr():
    global last_capture_time
    global frame
    global cam_busy
    global capture_interval
    current_time = time.time()
    if current_time -  last_capture_time >= capture_interval:
        func_voice_warning("Capturing hdr photo")
        timestamp = time.strftime("%Y%m%d_%H%M%S")
        image_filename = f"captured_{timestamp}.png"
        hdr_image = cv2.detailEnhance(frame, sigma_s=12, sigma_r=0.15)
        image_filename = "hdr"+image_filename
        cv2.imwrite(image_filename, hdr_image)
        print(f"Captured photo: {image_filename}")
        last_capture_time = current_time

def func_start_video_capture():
    global video_writer
    global is_recording
    if not is_recording :
        func_voice_warning("Video recording on")
        timestamp = time.strftime("%Y%m%d_%H%M%S")
        video_filename = f"video_{timestamp}.avi"
        video_writer = cv2.VideoWriter(video_filename, cv2.
 ↪VideoWriter_fourcc(*'XVID'), 20.0, (640, 480))
        print(f"Recording started: {video_filename}")
        is_recording = True

def func_end_video_capture():
    global video_writer
    global is_recording
    if is_recording :
        func_voice_warning("Video recording off")
        print("Recording stopped. All fingers are down.")
        is_recording = False
        video_writer.release()  # Stop recording
        video_writer = None

def func_music_play():
    global is_playing
    if not pygame.mixer.music.get_busy() and not is_playing:
        func_voice_warning("Music play")
        load_song(current_song_index)
        pygame.mixer.music.play()
        is_playing=1

def func_music_simple_pause():
    global is_playing
    if pygame.mixer.music.get_busy() and is_playing:
        pygame.mixer.music.pause()
        is_playing=0

def func_music_pause():
    if pygame.mixer.music.get_busy() and is_playing:
        func_music_simple_pause()
        func_voice_warning("Music paused")

def func_music_resume():
    global is_playing
    if not pygame.mixer.music.get_busy():
        func_voice_warning("Music resumed")
        pygame.mixer.music.unpause()
        is_playing=1

def func_music_next():
    global current_song_index
    global is_playing
    current_song_index = (current_song_index + 1) % len(playlist)
    func_music_simple_pause()
    func_voice_warning("Playing next music")
    load_song(current_song_index)
    pygame.mixer.music.play()
    is_playing=1

def func_music_prev():
    global current_song_index
    global is_playing
    current_song_index = (current_song_index - 1) % len(playlist)
    func_music_simple_pause()
    func_voice_warning("Playing previous music")
    load_song(current_song_index)
    pygame.mixer.music.play()
    is_playing=1

def func_capture_on_audio():
    global is_recording, video_writer
    global is_audio_capture_on
    is_audio_capture_on = 1
    global camera_1_index
    if not is_recording and video_writer == None:
        cap = cv2.VideoCapture(camera_1_index)
        ret, frame = cap.read()
        global last_capture_time
        global cam_busy
        global capture_interval
        current_time = time.time()
        if current_time -  last_capture_time >= capture_interval and␣
 ↪ret:
            func_voice_warning("Capturing photo")
            timestamp = time.strftime("%Y%m%d_%H%M%S")
            image_filename = f"captured_{timestamp}.png"
            cv2.imwrite(image_filename, frame)
            print(f"Captured photo: {image_filename}")
            last_capture_time = current_time
        cap.release()
        cap = None
    is_audio_capture_on = 0

def func_capture_hdr_on_audio():
    global is_recording, video_writer
    global is_audio_capture_on
    is_audio_capture_on = 1
    global camera_1_index
    if not is_recording and video_writer == None:
        cap = cv2.VideoCapture(camera_1_index)
        ret, frame = cap.read()
        global last_capture_time
        global cam_busy
        global capture_interval
        current_time = time.time()
        if current_time -  last_capture_time >= capture_interval and␣
↪ret:
            func_voice_warning("Capturing hdr photo")
            timestamp = time.strftime("%Y%m%d_%H%M%S")
            image_filename = f"captured_{timestamp}.png"
            hdr_image = cv2.detailEnhance(frame, sigma_s=12, sigma_r=0.
↪15)
            image_filename = "hdr"+image_filename
            cv2.imwrite(image_filename, hdr_image)
            print(f"Captured photo: {image_filename}")
            last_capture_time = current_time
        cap.release()
        cap = None
    is_audio_capture_on = 0

def func_start_video_capture_on_audio():
    global video_writer
    global is_recording
    global is_audio_capture_on
    is_audio_capture_on = 1
    camera_1_index
    if not is_recording and video_writer == None:
        cap = cv2.VideoCapture(camera_1_index)
        ret, frame = cap.read()
        func_voice_warning("Video recording on")
        timestamp = time.strftime("%Y%m%d_%H%M%S")
        video_filename = f"video_{timestamp}.avi"
        video_writer = cv2.VideoWriter(video_filename, cv2.
↪VideoWriter_fourcc(*'XVID'), 20.0, (640, 480))
        print(f"Recording started: {video_filename}")
        is_recording = True
        stime = time.time()
        while True:
            ret, frame = cap.read()
            video_writer.write(frame)
            if time.time() - stime > 5 :
                is_recording = False
                video_writer.release()  # Stop recording
                video_writer = None
                break
        func_voice_warning("Video recording stopped")
        cap.release()
        cap = None
    is_audio_capture_on = 0

recognizer = sr.Recognizer()

def recognize_speech():
    global camera_2_index
    mic = sr.Microphone(device_index=camera_2_index)
    with mic as source:
        print("Please say something...")
        recognizer.adjust_for_ambient_noise(source)  # Adjust for␣
 ↪ambient noise
        audio = recognizer.listen(source)

        try:
            # Recognize speech using Google Web Speech API
            text = recognizer.recognize_google(audio)
            print(f"You said: {text}")
            if 'capture photo' in text:
                print("Capture photo command recognized.")
                func_capture_on_audio()
            elif 'HDR' in text:
                print("capture hdr photo")
                func_capture_hdr_on_audio()
            elif 'record video' in text:
                print("Record video command recognized.")
                func_start_video_capture_on_audio()
        except sr.UnknownValueError:
            print("Sorry, I could not understand the audio.")
        except sr.RequestError:
            print("Could not request results from Google Speech␣
 ↪Recognition service.")



'''

def func_take_photo_hdr()
def func_start_video_capture()

def func_end_video_capture()



def func_light_off()

def func_voice_warning()
'''

def function_map(condition):
    if condition == '01100' :
        func_take_photo()
    elif condition == '11001' :
        func_take_photo_hdr()
    elif condition == '11000' :
        func_start_video_capture()
    elif condition == '10000' :
        func_end_video_capture()
    elif condition == '00001' :
        func_music_play()
    elif condition == '00011' :
        func_music_pause()
    elif condition == '00111' :
        func_music_resume()
    elif condition == '01111' :
        func_music_next()
    elif condition == '11111' :
        func_music_prev()


mp_hands = mp.solutions.hands
hands = mp_hands.Hands(max_num_hands=1)
mp_drawing = mp.solutions.drawing_utils


last_capture_time = 0
capture_interval = 3
cam_busy = 0
last_capture_time = 0
capture_interval = 3
is_recording = False
video_writer = None
frame = None
current_song_index = 0
```

```
is_playing = 0
is_audio_capture_on = 0
camera_1_index = 0
camera_2_index = 2

playlist = ["/home/gesture/project0/SoundHelix-Song-1.mp3",
            "/home/gesture/project0/song2.mp3",
            "/home/gesture/project0/song3.mp3",
            "/home/gesture/project0/song4.mp3",
            "/home/gesture/project0/song5.mp3",
            "/home/gesture/project0/Mere Dholna.mp3",
            ]

speech = LiveSpeech(
    lm='7740.lm',          # Path to your language model
    dic='7740.dic',        # Path to your dictionary
    kws_threshold=1e-20   # Adjust for sensitivity
)


def main():
    global last_capture_time
    global function_map
    global frame
    global speech, camera_1_index
    frame_count=0
    c=0
    final_id=[0,0,0,0,0]
    prev_condition = "#####"
    ultra_distance = 80
    lastTime = time.time()
    while True:
        dist = distance()
        time.sleep(0.05)
        #print("Hello")
        if dist < ultra_distance and is_audio_capture_on == 0:
            cap = cv2.VideoCapture(camera_1_index) # 0 for middle␣
 ↪bottom usb port and 1 for left bottom
            print("Camera activated")
            c=0
            final_id=[0,0,0,0,0]
            while dist<ultra_distance or is_recording:
                ret, frame = cap.read()
                frame_count+=1
                #if not ret:
                #   break
                if is_recording and video_writer is not None:  #for␣
↪recording video according to condition
                    video_writer.write(frame)
                if(frame_count % 5 != 0) :
                    continue
                img_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                results = hands.process(img_rgb)
                if results.multi_hand_landmarks:

                    for hand_landmarks in results.multi_hand_landmarks:
                        #mp_drawing.draw_landmarks(frame,␣
↪hand_landmarks, mp_hands.HAND_CONNECTIONS)
                        fingertip_ids = [4, 8, 12, 16, 20]


                        id = []
                        if ((hand_landmarks.landmark[9].x <␣
↪hand_landmarks.landmark[2].x and hand_landmarks.landmark[2].x <␣
↪hand_landmarks.landmark[4].x) or (hand_landmarks.landmark[9].x >␣
↪hand_landmarks.landmark[2].x and hand_landmarks.landmark[2].x >␣
↪hand_landmarks.landmark[4].x)) :
                            #print("Thumb is up ")
                            id.append(1);
                        else :
                            #print("Thumb is down ")
                            id.append(0);
                        if  hand_landmarks.landmark[8].y <␣
↪hand_landmarks.landmark[6].y:
                            #print("Index is up ")
                            id.append(1);
                        else :
                            #print("Index is down ")
                            id.append(0);
                        if  hand_landmarks.landmark[12].y <␣
↪hand_landmarks.landmark[10].y:
                            #print("Middle is up ")
                            id.append(1);
                        else :
                            #print("Middle is down ")
                            id.append(0);
                        if  hand_landmarks.landmark[16].y <␣
↪hand_landmarks.landmark[14].y:
                            #print("Ring is up ")
                            id.append(1);
                        else :
                            #print("Ring is down ")
                            id.append(0);
```

```
                      if  hand_landmarks.landmark[20].y <␣
↪hand_landmarks.landmark[18].y:
                          #print("Little is up ")
                          id.append(1);
                      else :
                          #print("Little is down ")
                          id.append(0);
                  c=c+1
                  #print(final_id)
                  print("....")
                  for i in range(5):
                      final_id[i] = final_id[i] + id[i]
                  if c==5: # for getting fast response with camera
                      for i in range(5):
                          final_id[i] = final_id[i]/c
                      for i in range(5):
                          if final_id[i] >= 0.5:
                              final_id[i] = 1
                          else :
                              final_id[i] = 0
                      print(final_id)
                      condition = "".join(map(str, final_id))
                      currentTime = time.time()
                      #print(lastTime - currentTime)
                      if prev_condition != condition or (currentTime␣
↪- lastTime > 3):
                          if dist < ultra_distance:
                              function_map(condition)
                          prev_condition = condition
                          lastTime = time.time()
                      #function call here
                      print(final_id)
                      final_id = [0,0,0,0,0]
                      c=0
                  #print("\n")
                  #print(id)
                  #print(c)
              dist = distance()
              time.sleep(0.05)
          if cap is not None:
              cap.release()
              cap = None
              print("Camera deactivated")

      else :
          print("Distance is greater than 50 cm. Activating␣
↪microphone...")
```

```
            recognize_speech()
            '''
            print("Listening for commands...")
            stime = time.time()
            for phrase in speech:
                print(f"Detected: {phrase.hypothesis()}")
                if 'capture photo' in phrase.hypothesis():
                    print("Capture photo command recognized.")
                    func_capture_on_audio()
                    break
                elif 'capture high quality photo' in phrase.
 ↪hypothesis():
                    print("Record video command recognized.")
                    func_capture_hdr_on_audio()
                    break
                elif 'capture video' in phrase.hypothesis():
                    print("Record video command recognized.")
                    func_start_video_capture_on_audio()
                    break
                if time.time() - stime > 2:
                    break
                    '''


if __name__ == "__main__":
    main()
```