SDU Summer School
Human Robot Interaction

August 2021

1 Introduction

For this exercise you have to use the OpenFace application to extract facial features using the camera. These features should then be used to create interactions between you and the "robot". This document shows how to setup the Jetson Nano, start OpenFace, and gives example code to extract the facial features and begin creating interactions.

All the code examples can be found on the github repository:
https://github.com/akollaki/SDUSummerSchool.git

2 Task

Work in groups in the breakout rooms.

Make the "robot" react to establishment of "mutual gaze". When the robot detects a human looking at it, make it ask a question from the human. Then use speech recognition to understand what they are saying, and react to that again. For example:

mutual gaze established, i.e. human looks at robot/camera

Robot: Would you like to sanitize your hands?

Human: Yes.

Robot: Ok, put your hand under the dispenser.

Once a basic interaction is achieved, you can come up with something more advanced. You could start by implementing eyes on the robot itself (face simulated by computer screen). You could use

the `pygame` library for that (https://www.pygame.org/news). Eyes are easily created by drawing circles for the eyes and pupils. Additional ideas for tasks could be as follows:

- Simon says
- Blink murder
- Rock paper scissors using head orientation
- Game of Chicken - who will look away first?
- Object selection with gaze on screen
- Facial expression detection

Once you have implemented a task that you like, invite colleagues from other groups as experiment participants. Best is if they are uninformed about the purpose of the testing. Measure some dependent variable, e.g. reaction time, task completion time, user satisfaction (via questionnaire feedback). For a controlled experiment you can implement two versions of the task, one in which you use gaze itself and another where head pose is used instead. OpenFace detects gaze less precisely compared to head pose, so the latter might work better. If you have enough subjects, you could use the t-test (e.g. in Excel, Data - Data Analysis - t-test) to determine if there is significant difference between your data. You could also implement smiley eyes vs. angry eyes and see if performance changes.

Feel free to implement as much or as little of the tasks as it suits you.

3 Internet Connection

To connect to the internet through WiFi you need to first download the driver for your Edimax dongle. With the ethernet cable and the dongle both plugged in, go to:
`System Settings` → `Software & Updates` → `Additional Drivers`

You should arrive at the window below. Press `Apply Changes`. After it is done, you can unplug the Ethernet cable.

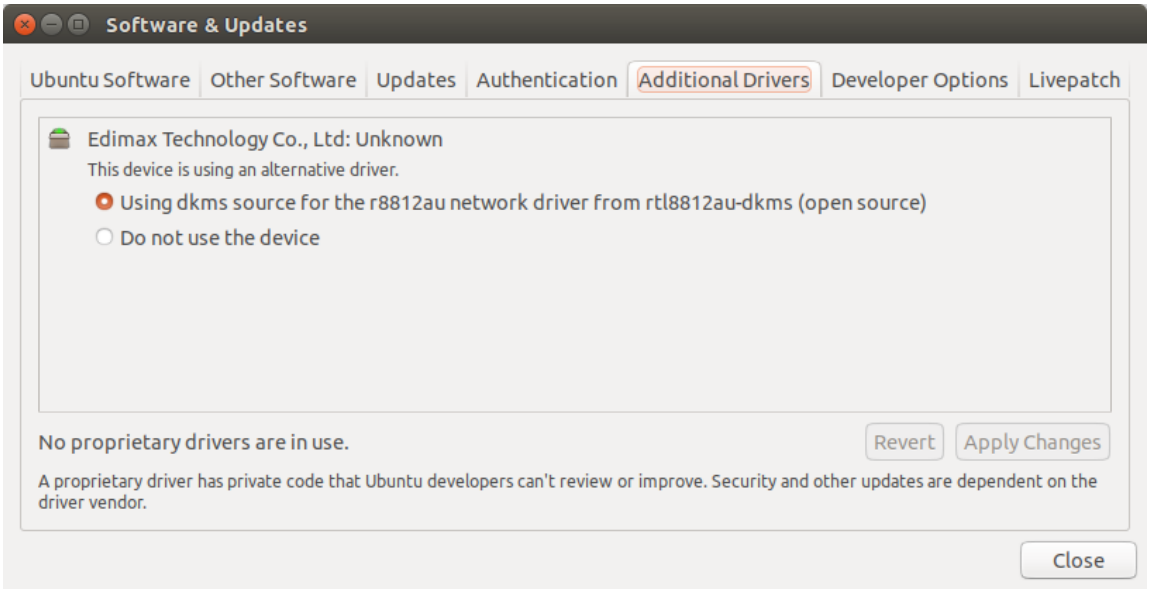

Figure 1: Edimax WiFi Dongle driver - download window

Reboot your Jetson and go to: `System Settings` → `Network` → `Wireless`.

Press `Connect to Hidden Network` and enter the following details:

Network Name: ProjektNet
Wi-Fi Security: WPA & WPA2 Personal
Password: RobotRocks

The network should only be used for project purposes. You can also try to connect to SDU-Visitor and follow the instructions.

4 SSH Connection

To control the "robot" from your computer you can establish an SSH connection. With both the computer and the robot connected to the same WiFi network, open a terminal window on the robot. Run the command `ifconfig`. Find the information on the WiFi connection (starting with wlp). Read the inet address e.g. 10.126.128.139.

On your computer, open a terminal and run the command ssh user@inet_address, for example:

```
ssh dlinano@10.126.128.139
```

More information on SSH https://www.ssh.com/academy/ssh

5 OpenFace

OpenFace is used to extract gaze vectors, head pose and facial expressions from images. It is already installed on the Jetson Nano and can be run with the following commands:

```
cd OpenFace/build
sudo ./bin/FaceLandmarkVidMulti -device 0 -of output -gaze
# -device to use webcam, -of to set output filename, -gaze to only save gaze parameters
```

When running OpenFace the extracted face parameters are saved to a csv file with the name specified by the argument following -of. The csv file can be read while the program is running and a Python example of this can be seen in the Code section.

More information on OpenFace and additional command line arguments:
https://github.com/TadasBaltrusaitis/OpenFace/wiki/Command-line-arguments

6 Install requirements

In order to run the example functions from the Code section (for Speech Recognition), the below requirements have to be installed first:

- Python Libraries:
  - pydub
  - speechrecognition
  - gtts
- System dependencies:
  - portaudio19-dev
  - python3-pyaudio
  - ffmpeg
  - frei0r-plugins
  - flac

Install with:

```
pip3 install --user pydub speechrecognition gtts
sudo apt-get install portaudio19-dev python3-pyaudio ffmpeg frei0r-plugins flac
```

7 Code

These Python functions can be used to read the face parameters from the csv file given by OpenFace. The two parameters specifying the eye gaze angles are extracted to determine if a person is looking at the camera. The thresholds have to be calibrated.
The files with these functions can be found in the github repository.
Run it using the following command in a terminal:

```
python3 eye_gaze.py
```

```
# determine if a person is looking at camera
def eye_contact(gaze_vector, lower, upper):
    if all(lower < ele < upper for ele in gaze_vector):
        return True
    return False

# continuously read last line of csv file and return if gaze vectors are within bounds
lastRead = 0
def read_csv(filename, lower, upper):
    global lastRead
    path = f"/home/dlinano/OpenFace/build/processed/{filename}.csv"
    while True:
        with open(path) as file:
            data = file.readlines()
            if data:
                lastRow = data[-1].split(',')
                if lastRow[0] != lastRead and lastRow[4] != '0':
                    lastRead = lastRow[0]
                    try:
                        if eye_contact([float(lastRow[11]), float(lastRow[12])], lower, upper):
                            return
                    except IndexError:
                        pass

if __name__ == '__main__':
    lowerBound = -0.3
    upperBound = 0.3
    try:
        while True:
            read_csv("output", lowerBound, upperBound)
            # reach here if eye vectors are within lower and upper bounds
            print('looking at camera')
    except KeyboardInterrupt:
        print("Program closed")
```

The functions below can be used to transcribe audio input from the microphone and search for keywords in the transcription. Additionally, a function performing text to speech is given, but note that the Jetson Nano kits do not come with speakers.

```
from gtts import gTTS
from io import BytesIO
from pydub import AudioSegment
from pydub.playback import play
import speech_recognition as sr
import re

# Google speech recognition
def google_in(lang):
    r = sr.Recognizer()
    r.pause_threshold = 0.5
    with sr.Microphone() as source:
        r.adjust_for_ambient_noise(source, duration=0.5)
        print("Say something!")
        try:
            audio = r.listen(source, timeout=5, phrase_time_limit=3)
        except sr.WaitTimeoutError:
            return "Timeout"
        else:
            try:
                out = r.recognize_google(audio, language=lang)
            except sr.UnknownValueError:
                print("Google Speech Recognition could not understand audio")
            except sr.RequestError as e:
                print("Could not request results from Google Speech Recognition service; {0}".
 format(e))
            else:
                return out
        return "Error"

# can be used to search for words in string from speech recognition
def find_whole_word(w):
    return re.compile(r'\b({0})\b'.format(w), flags=re.IGNORECASE).search

# text-to-speech
def speak(text):
    with BytesIO() as f: # open buffer f to temporarily store audio
        tts = gTTS(text=text, lang="en") # google text-to-speech
        tts.write_to_fp(f)  # write speech to f
        f.seek(0)  # seek to zero after writing
        audio = AudioSegment.from_file(f, format="mp3") # create audio from buffer
        play(audio) # play audio


# Example for speech recognition
speech = google_in('en-GB')

# Example for keyword detection
keyword = 'hello'
if find_whole_word(keyword)(speech):
    print(f'I heard {keyword}')

# Example for text to speech
speak(keyword)
```