

Title- Affective Awareness Agents in Virtual Reality

By- Sweta Das

Student ID- 18080395d

Programme- 61431-FCS

Examiner- Dr. Richard Li

Co-Examiner- Dr. Carne De Canavalet Xavier

Second Co-Examiner- Dr. Kay Chen Tan

Date of submission- 20 April 2022

Table of contents

1. Abstract - pg 5
2. Background - pg 5
 - 2.1 Introduction - pg 5
 - 2.2 Problem Statement - pg 6
 - 2.3 Objectives - pg 6
 - 2.4 Current prototypes developed - pg 7
3. Choice of tools - pg 8
 - 3.1 Choice of Cloud service - pg 8
 - 3.1.1 Google Cloud Platform - pg 9
 - 3.1.2 IBM Watson - pg 9
 - 3.1.3 Microsoft Azure Cloud Services - pg 9
 - 3.1.4 Amazon Web Services - pg 10
 - 3.2 Choice of headset - pg 11
4. System design and architecture - pg 11
 - 4.1 Requirements and initial setup - pg 11
 - 4.2 Program design - pg 13
 - 4.2 Speech To Text and Intent Recognition - pg 15
 - 4.3 Facial expression recognition and and extraction - pg 24
 - 4.4 Response by agent - pg 31
5. Results - pg 35
6. Impact - pg 38
7. Future Improvements - pg 39
8. Discussion - pg 40
9. Conclusion - pg 41
10. References - pg 42

List of tables and figures

Figure 1: Overall setup of the project with hardware and software components to run the program on VR. - pg 13

Figure 2: Start Recognition - pg 14

Figure 3: Stop Recognition - pg 14

Figure 4: Overall program design - pg 15

Figure 5: Start Recognition code - pg 16

Figure 6: Stop Recognition code - pg 17

Figure 7: Emotional recognition from speech overall process. - pg 17

Figure 8: Major steps utilizing cloud service for emotional recognition from speech - pg 18

Figure 9: Emotion words in entities and relevant keywords - pg 19

Figure 10: .json format for importing data. - pg 20

Figure 11: Python program lines for data cleaning - pg 21

Figure 12: Mapping with relevant lexicons on LUIS portal. - pg 21

Figure 13: Creation of the LUIS intent recognition model code. -pg 22

Figure 14: Intent processing code - pg 22

Figure 15: Adding the values in each list. - pg 23

Figure 16: Intent recognised correctly, then show the emotion on screen. - pg 23

Figure 17: Interaction of program with cloud service and data cleaning. - pg 24

Figure 18: List of emotions. - pg 25

Figure 19: Mouth sad left expression. Reference- SDK of SRanipal [12]. - pg 26

Figure 20: Weights for mouth sad left expression. - pg 26

Figure 21: Mouth smile right expression. Reference- SDK of SRanipal [12]. - pg 26

Figure 22: Weights for mouth smile right expression. - pg 27

Figure 23: Enable the detection of emotions when the “Response” button is clicked. - pg 27

Figure 24: Call the function in the Update method if the “Response” button is clicked. - pg 27

Figure 25: Assign the probability weights to emotions. - pg 28

Figure 26: Adding probability weights to variables of emotions only when it is high in range [0.5,1).
- pg 28

Figure 27: Take average of weights - pg 29

Figure 28: Add all emotions and weights in a dictionary and just weights in another list.- pg 29

Figure 29: Finding the score and final emotion. - pg 30

Figure 30: Emotional recognition through face - pg 30

Figure 31: Compare the scores and decide emotion. - pg 31

Figure 32: Final emotion decided from both speech and facial tracking input channels - pg 31

Figure 33: List of responses with respect to each emotion. - pg 32

Figure 34: Code to provide a suitable response as per the corresponding emotion. -pg 32

Figure 35: Text to speech and lip synced response by the agent. - pg 33

Figure 36: Create authenticator. - pg 34

Figure 37: Create TTS service - pg 34

Figure 38: Create authenticator for TTS service. - pg 34

Figure 39: Process response and convert to audio clip. - pg 34

Figure 40: TTS conversion with IBM Watson Cloud and lip sync - pg 35

Figure 41: Results - pg 36

Figure 42: Adding 0.1 to the score of face recognition from 10th result onwards. - pg 37

Figure 43: Assign all weights when the program is running. - pg 38

Figure 44: Update newer weights. The comments state that “weights” are obtained from the Lip module. - pg 38

Figure 45: Region containing a smiling face. - pg 41

1. Abstract

The project aims to create an affective awareness agent in Virtual reality (VR), which can understand the user's emotions from multiple input channels. In this case speech and facial recognition are used to analyze the user's emotions as the scale of the project is small. In return of the inputs by the user, an agent replies by lip synchronizing the response to the user as per the emotion analyzed. Multiple such prototypes have been developed by universities and technology companies. Tools and techniques have been adapted into this project from these prototypes suitable for the project. Lexical analysis for speech recognition and probability weight extraction of facial expressions for facial recognition are used to analyze emotions from these two channels. From the results discussed, some better methods are suggested, like semantic analysis for speech and continuous recognition for face.

2. Background

2.1 Introduction

It is very common to see virtual agents such as chatbots in applications and player objects in games. It becomes much better when such agents become more realistic, and the user feels connected to the agent as a friend. The inspiration is taken from these examples to create one such use case. As mentioned by Ahmed and Harjunen et al, interpersonal touch is very important to connect on a personal level, that the virtual agents in Virtual Reality can help to establish [1]. In a study conducted by Numata et al, it was discovered that the users could feel the connection with agents if

the agent showed positive actions, regardless of their belief that the agent was a computer or a human [2].

As Masahari Mori explains the uncanny valley theory, that a robot can look like a human being only upto certain limit and not beyond that, this needs to be taken care of too. Else, the agent will look like a servant who is only responding to the orders of the user [3]. The outcome will be to create a virtual friendly agent, establishing a relation of friendship, rather than a master-servant relation. The creation of such an agent will help to discover better approaches of improvement in virtual companions, discovering more use cases, features and increasing more possibilities in the field of XR related to affective awareness agents.

2.2 Problem Statement

The aim is to recognise the overall emotions of a user from his or her facial expressions and speech.

2.3 Objectives

The whole project will be discussed in different sections, where the overall process is divided into different problems. Since this field of research and exploration is quite novel, therefore the approach will be to take inspiration from existing technologies to create and combine the features to make it a unique solution. For example, taking inspiration from chatbots, empathetic sentences will be added as a response to the user as a chat feature. Overall, the product will be the combination of some of the pre-existing features and techniques, but in a different arrangement.

While the speech can be captured by the microphone of the headset, the facial expressions can be captured using the VIVE facial tracker attached to the Head Mounted Device (HMD) or the headset.. As per the input, the agent will act towards the user. Hence, the agent created will be intelligent enough to understand the user and behave like an understanding friend.

The first objective is to generate text from the user's speech and analyze the emotions. This text will be used as input to analyze the emotions of the user through some specific words of emotion like “happy” and “sad”.

The second objective is to retrieve the facial expressions of the user and analyze emotions through facial expressions. This is possible with the VIVE facial tracker attached to the HMD.

The third objective is to convert the response into a text and display it in the form of speech by the avatar in the program which will lip sync the audio output. This will help in giving a humane touch, rather than making it appear like a game.

Finally, testing will be done with some use cases to test the effectiveness of the system as an understanding friend to the user.

2.4 Current prototypes developed

Some of the useful and workable prototypes from which inspiration is taken are discussed here. These are from academia to industrial applications.

The first application discussed is by MIT's Media Lab [4]. The affective awareness agent is a virtual 3D character developed in Graphical User Environment. The sensors used for the user input are pressure-sensitive mouse, a Bluetooth wireless skin conductivity sensor, a TekScan pressure sensor on a chair for posture tracking, a stereo head tracking system and IBM Blue eyes infrared camera for tracking eye input. The data retrieved by the sensors are fed into the algorithms designed by the team to analyze the user's emotions and feelings, in short the affective state. The data is received by the program and the agent responds accordingly. The video of the user is also displayed on the screen. This helps in analysis of the user's overall behavior through different forms, as only one channel may partially express the user's intentions. It was also found out that the user felt more attached to the agent when it expresses a similar reaction to the user, as it looks empathetic. While this may not be the best case as some users may want to have a different reaction, for example, a hidden joke in a sad statement of failure which they want to laugh at but the agent reflects a sad behavior, it may not be favorable. In our project, since it is simple, we will take into account the empathetic response generated by the agent. According to the availability of resources, we will take the input from the agent from facial expressions and speech.

A useful example of calculation of final emotion value from multiple channels is presented by the researchers at the University of Maryland [5]. They created an algorithm named EmotiCon, which took input of the user from facial expressions and physiological features. The recognised features are plotted in the 3D graph for facial expressions and physiological features. The idea is to do multi modal analysis, which means that the feature points will be plotted on the graph from all the input channels, and the algorithm EmotiCon will calculate the final emotion using all the feature points.

The final feature will be known from the result that comes from the algorithm. The importance is laid on the fact, as mentioned in their paper citing the statement of Frege, that words should not be taken in isolation, but taken in context. Therefore, other channels of inputs are important too. Motivated from this approach, the aspect of facial recognition is considered and to take the scores of speech recognition and facial recognition is also considered for emotional recognition in our program. The emotion is not recognised in our program as a mixture of facial expression and speech, but the both are taken into consideration to determine the type of emotion exhibited by the user.

A game developed by the researchers at the University of Augsburg [6], designed a game in which the agent is a snail named Tiffany. Tiffany would move as per the emotions of the user. For example, if the user will get angry, Tiffany will show to become scared or hide. If the user is happy, Tiffany will show more cooperative actions. From here, the motivation was taken to provide responses by the agent which will look realistic. Our agent in the game is a female 3D avatar, but the responses were chosen to be empathetic. From the game, the researchers concluded that the users would expect a reflection of their own responses instead of only getting a direct response from the agent in the game. Also, the user was sometimes confused on the type of response from the agent. The researchers have discussed the future improvement to use more obvious animations and sounds. Therefore, in our project, the emotion of the user is displayed on the screen as well as an empathetic response in the form of a 3D avatar lip syncing the response is shown.

3. Choice of tools

3.1 Choice of Cloud Services

As mentioned in the abstract, there are technology companies specially interested in developing tools for affective computing. Some of the big technology giants have developed tools compatible with Unity3D engine, like Speech To Text (STT) and Text To Speech (TTS) services. Multiple STT and TTS services were tried and tested that are compatible with Unity3D engine as it is the main platform for development. In addition to this, features provided by some of these companies like intent recognition from speech if available, and lip synchronizing (lip sync) of avatar with audio were also tested.

Most of the tools provided are a part of the cloud services. Therefore, the most famous cloud services providers were researched into, which are- Google Cloud Platform (GCP), IBM Watson, Microsoft Azure Cloud Services and Amazon Web Services (AWS).

3.1.1 Google Cloud Platform

The API of GCP with Dialogflow is used for the interaction between the program in Unity3D engine and Dialogflow. The speech of the user is accepted by the microphone enabled by the program in Unity, sent to GCP for conversion to text, and then to Dialogflow to be sent as an intent input. Then, the response is generated by Dialogflow in the form of text, further converted to speech by GCP and sent to the program in the Unity3D engine.

While this is useful for real time speech-to-text-to-speech conversion, however, the responses and questions in Dialogflow need to be input manually. Also, the responses are dependent strictly on the phrases entered as intents. This functionality may be good for a program with a limited set of questions. Also, due to the cost of services and limited documentation and support for Unity3D engine, this was not considered finally.

3.1.2 IBM Watson

The cloud platform provides features for STT service and examples on the GitHub page for the same by Dr. Scott Hwang [7]. The API included in the sample takes the speech input from the user and passes to the cloud for conversion to text. The text response is shown on the screen of the program. Furthermore, the text response to be provided will be converted to speech and shown in the Unity3D program screen. This comes with lip sync features where there is a sample 3D model as an avatar which will synchronize its lips while the text is converted to speech.

The lip sync feature is adapted from an example implemented by Dr. Scott Hwang. The text in the program is converted to speech at runtime using IBM Watson TTS service.

However, for intent recognition, there is not clear enough documentation for Unity3D engine and limited explanation. The STT feature may be good for only showing the text output.

3.1.3 Microsoft Azure Cloud Services

The cloud services provides examples and Software Development Kit (SDK) for cognitive services for running in Unity3D. There are well written documentations for utilizing the STT service in C#, which is the language used in Unity3D for writing program scripts. Microsoft's Cognitive Services is included in the Azure Cloud Services, used especially for STT and TTS services. The API takes the microphone input from the device, when enabled by program in Unity3D. Then, the program passes the input in the cloud, processes into text and returns the text output. There are options for translation features which may be utilized in the future for non-English speaking participants.

The STT conversion also helps in straight forward recognition of intents. Language Understanding (LUIS) is a cloud-based conversational AI service provided by Microsoft used for the project. This uses custom machine-learning intelligence for a user's conversational and natural language text that applies custom machine-learning intelligence to a user's conversational language in the form of input and gives a relevant output. The sentences are intents, which are possible inputs from the user provided in the LUIS portal. The service is referenced using API and SDK. While the possible user responses are provided in form of intents, the entities are the keywords used to determine a certain class of objects like happy or sad emotion in our case. This helps in determining the emotion of the user and the context. While it may not be accurate, but for this stage it is enough. Hence, the STT service is used in the project.

The TTS conversion is also possible with Azure's TTS service enabled by Microsoft's Cognitive Services. This is further enabled by the presence of Visemes (the facial features mapped with certain poses of mouth). However, there is no example of supported implementation of Visemes with real time TTS, therefore this option is not used.

Another reason for using the services by Azure is because of the free account services provided to students, which makes it easier to run the services for a longer period of time.

3.1.4 Amazon Web Services

The objectives of the STT for conversion of speech to text can be achieved through AWS but the cost factor and limited documentation prevent from using it in Unity and understanding. However, the TTS feature is used in this case for an alternate case.

Didimo is a special service used to create custom 3D avatars for lip sync. If integrated with Amazon Polly, the TTS service by AWS, lip sync can be enabled with the converted text. This feature has a

limitation that it cannot be implemented realtime, as the speech files must be downloaded from Amazon Polly's web portal along with pronunciation marks and added into our project. A good feature is that the avatars can be customized, which means that unlike Oculus's SDK, it is much easier to generate 3D avatars in our system just by a 2D image of a human-like face. As the user feels good with a human avatar, therefore this feature is very strong as there are many options to create avatars. With this, we have placed conditional responses, which means that the user will respond in certain ways based on the feeling of the user.

As there is no proven direct way to generate Visemes and Audio clips in runtime, but only avatars can be customized well, therefore as an alternate case where the user wants a personalized agent but not concerned about a more robust TTS system, this can be used.

3.2 Choice of headset

The Game lab of our university has Oculus Quest 2 and HTC VIVE Pro headsets. Out of the two, HTC VIVE Pro was chosen for program development. VIVE has released some interesting products to enhance VR development and experience, like facial tracking features with VIVE Visual tracker. With this, the most compatible headset is HTC VIVE Pro, though VIVE is releasing its support for other headsets too. If other updates are released for the software products released by VIVE, it will be easier to configure on HTC VIVE Pro headset rather than on Oculus Quest 2.

4. System design and architecture

The whole process is divided into three major parts- Speech To Text (STT) and intent recognition, facial expression extraction and recognition, and response by the agent. Before going into these major steps, overall setup and program design are explained for clarity.

4.1 Requirements and initial setup

The hardware devices needed are the HTC VIVE Pro headset along with the additional components for VR experience that come along with the package, and an additional facial tracker. Facial tracker is used for tracking the movements of the lower mouth of the user. The headset HTC VIVE Pro has advanced features like enabled facial tracking of lower mouth region with facial tracker device by VIVE. Hand controllers in the headset are for the purpose of interaction with the program running on

the headset. Base controllers that come along with the headset are provided for the purpose of tracking the location of the headset and the hand controllers. The headset needs to be attached to the computer with all the setups done properly- wires attached properly, hand controllers charged and both the base stations placed in the same line of sight without obstacles in between to ensure smooth tracking.

The essential softwares needed to be installed on the computer are SteamVR application and Super Runtime Animation pal (SRanipal) Runtime application. SteamVR ensures the connection of the headset to the computer and exchange of data between headset and computer, and SRanipal Runtime application ensures the same with the facial tracker. In addition to these softwares being installed on the computer, the Software Development Kits (SDK) associated with these softwares compatible with Unity3D engine, need to be imported in the program as Unity3D engine is used as the platform for development.

The facial tracker and the headset HTC VIVE Pro are connected well. After the connection is detected by the computer, SteamVR application and SDK detects the headset and establishes a connection with the headset for information exchange. Simultaneously, the SRanipal runtime software and SDK detect the headset connection first and then the facial tracker connection. After it is confirmed that the connection is established between the computer and the hardware devices, i.e., facial tracker and headset, the data can be exchanged between the hardware devices and the program on the computer. The screen of the program at runtime can be viewed in the headset. The face of the user can also be tracked by the facial tracker and the results will be visible on the program screen and the headset.

The following figure shows the whole arrangement of the hardware and the computer program briefly.

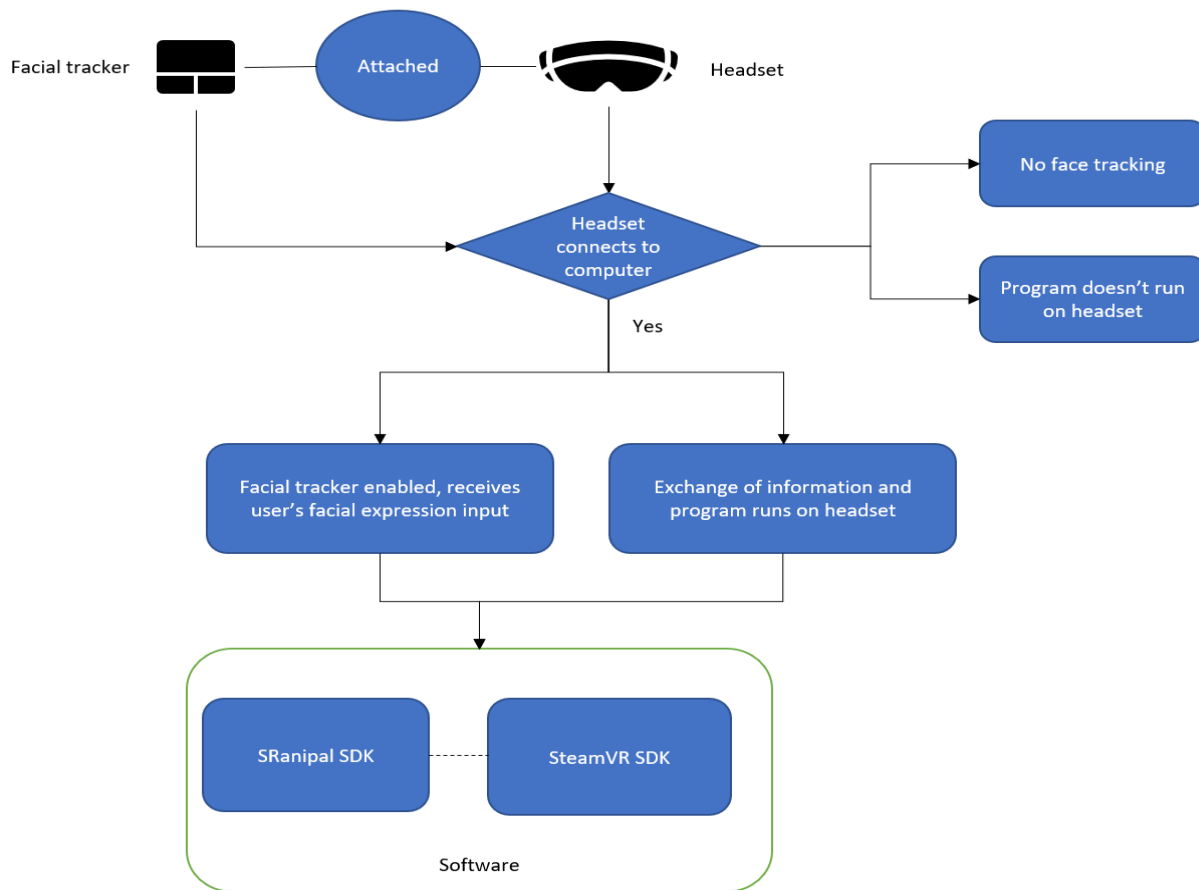


Figure 1: Overall setup of the project with hardware and software components to run the program on VR.

4.2 Program design

When the user clicks on the “Start Recognition” button, speech and facial expression recognition are enabled. When the user speaks, the data goes into the speech input channel. When the user makes some facial expressions in front of the tracker, the data goes into the facial input channel. Until the recognition of speech and facial expressions are enabled, the latest input overrides the previous input in both speech and facial input channels.

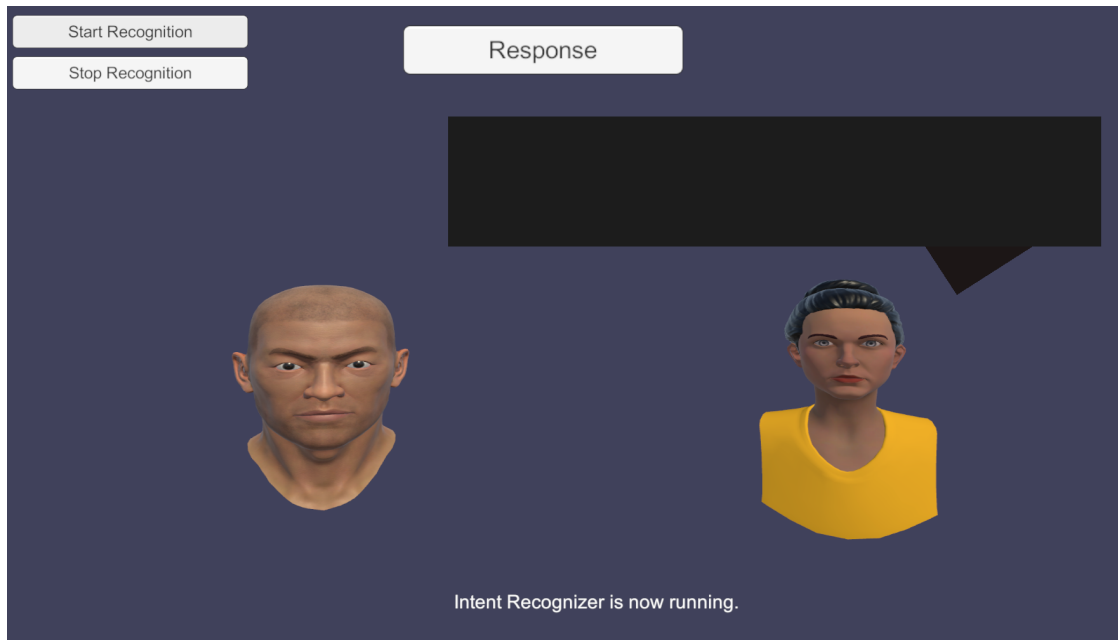


Figure 2: Start Recognition

When the user clicks on the “Stop Recognition” button, the speech and facial recognition stops. The latest recognised speech and facial expression are considered for emotional detection.

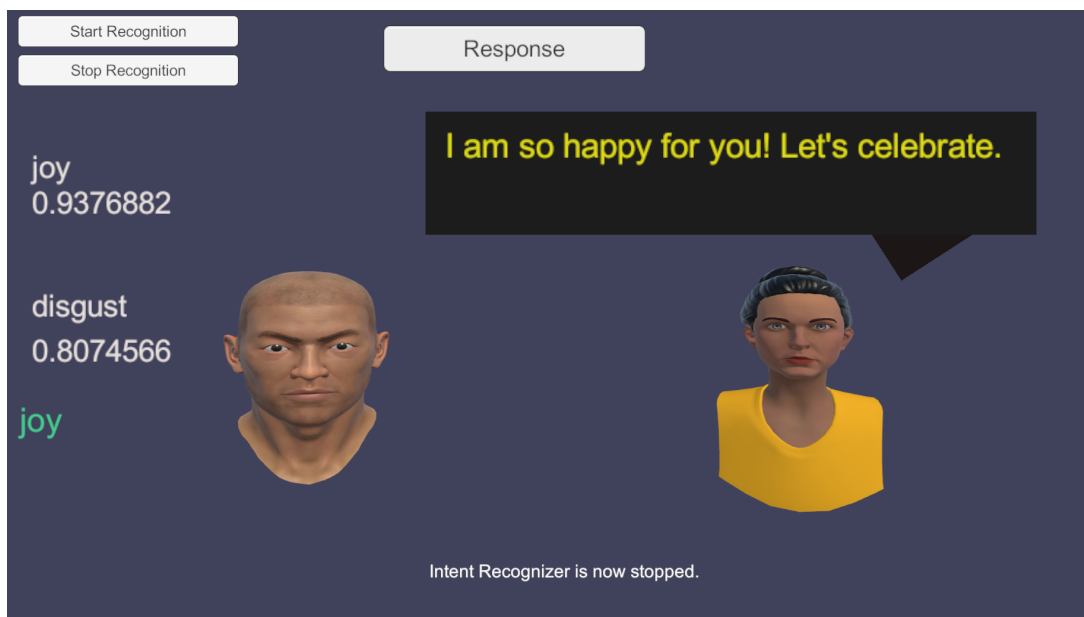


Figure 3: Stop Recognition

The overall predicted emotion is displayed. Figure 4 shows the overall program design.

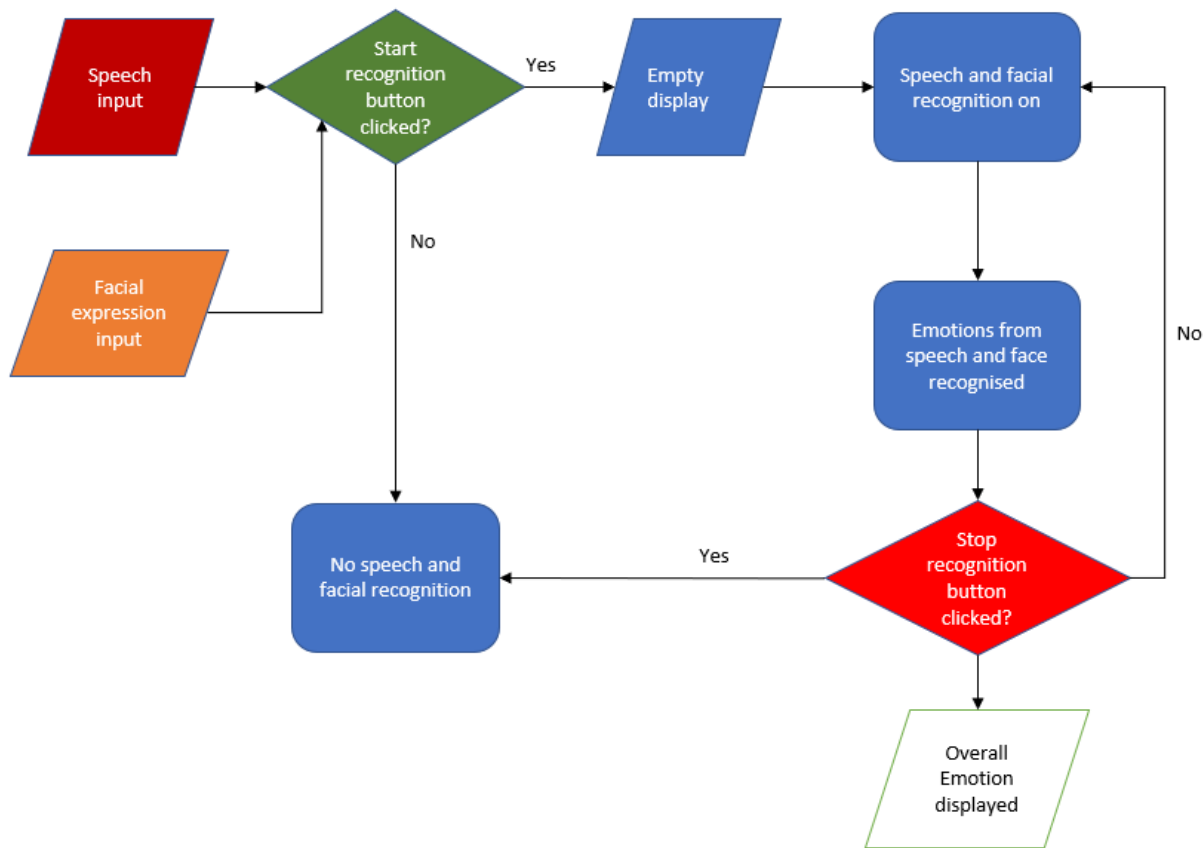


Figure 4: Overall program design

4.3 Speech To Text (STT) and intent recognition

Azure Cloud provides multiple types of services for different purposes. One of them is Azure Cognitive service for speech analysis. Speech service provided by Azure Cognitive service is used for STT conversion. Language Understanding (LUIS), which is included in Azure Cognitive service, is used for intent recognition from sentences. The way the emotional detection is done is, to convert the speech of the user to text first using Speech service, and then using the text for intent recognition using LUIS. The whole procedure is shown in figure 7, and the specific procedures related to data cleaning and training is shown further in figure 17.

The first step is the user's input into the program using speech. When the user clicks the button of "Start Recognition", speech recognition is enabled. The access to the microphone is granted for speech input.

```

public void StartContinuous()
{
    this.GetComponent<FinalEmotion>().emotionFace.text = "";
    this.GetComponent<FinalEmotion>().emotionFinal.text = "";
    this.GetComponent<FinalEmotion>().emotionSpeech.text = "";
    this.GetComponent<FinalEmotion>().responseAgent.text = "";
    this.GetComponent<FinalEmotion>().scoreSpeech.text = "";
    this.GetComponent<FinalEmotion>().scoreFace.text = "";

    if (micPermissionGranted)
    {
        StartContinuousIntentRecognition();
    }
    else
    {
        recognizedString = "This app cannot function without access to the microphone.";
    }
}

```

Figure 5: Start Recognition code

The speech of the user is converted into text using the Speech service as mentioned before. The emotion of the user is detected from the keywords in the converted text. For example, if a word in the sentence is “happy” and it is being recognised as a possible emotion, it will infer an emotion of “joy”, depending on how the model is trained. This type of intent recognition is done with the help of LUIS as mentioned before. The emotion detected and the score of detection accuracy are shown on the program’s screen, along with the speech of the user already converted to text. Every time the user speaks a new sentence, the text converted from the speech, the emotion detected from the text and the score of detection accuracy are displayed.

Once the “Stop Recognition” button is clicked, the recognition of speech is stopped. The current text converted from the speech, the emotion detected from the text and the score of detection accuracy displayed on the screen are considered final. If the user clicks on the “Start Recognition” button again, a new round of emotional detection from speech will begin.


```

1 reference
public async void StopIntentRecognition()
{
    if (intentreco != null)
    {
        await intentreco.StopContinuousRecognitionAsync().ConfigureAwait(false);
        intentreco.Recognizing -= RecognizingHandler;
        intentreco.Recognized -= RecognizedHandler;
        intentreco.SpeechStartDetected -= SpeechStartDetectedHandler;
        intentreco.SpeechEndDetected -= SpeechEndDetectedHandler;
        intentreco.Canceled -= CanceledHandler;
        intentreco.SessionStarted -= SessionStartedHandler;
        intentreco.SessionStopped -= SessionStoppedHandler;
        intentreco.Dispose();
        intentreco = null;
        recognizedString = "Intent Recognizer is now stopped.";
        UnityEngine.Debug.LogFormat("Intent Recognizer is now stopped.");
    }
}

```

Figure 6: Stop Recognition code

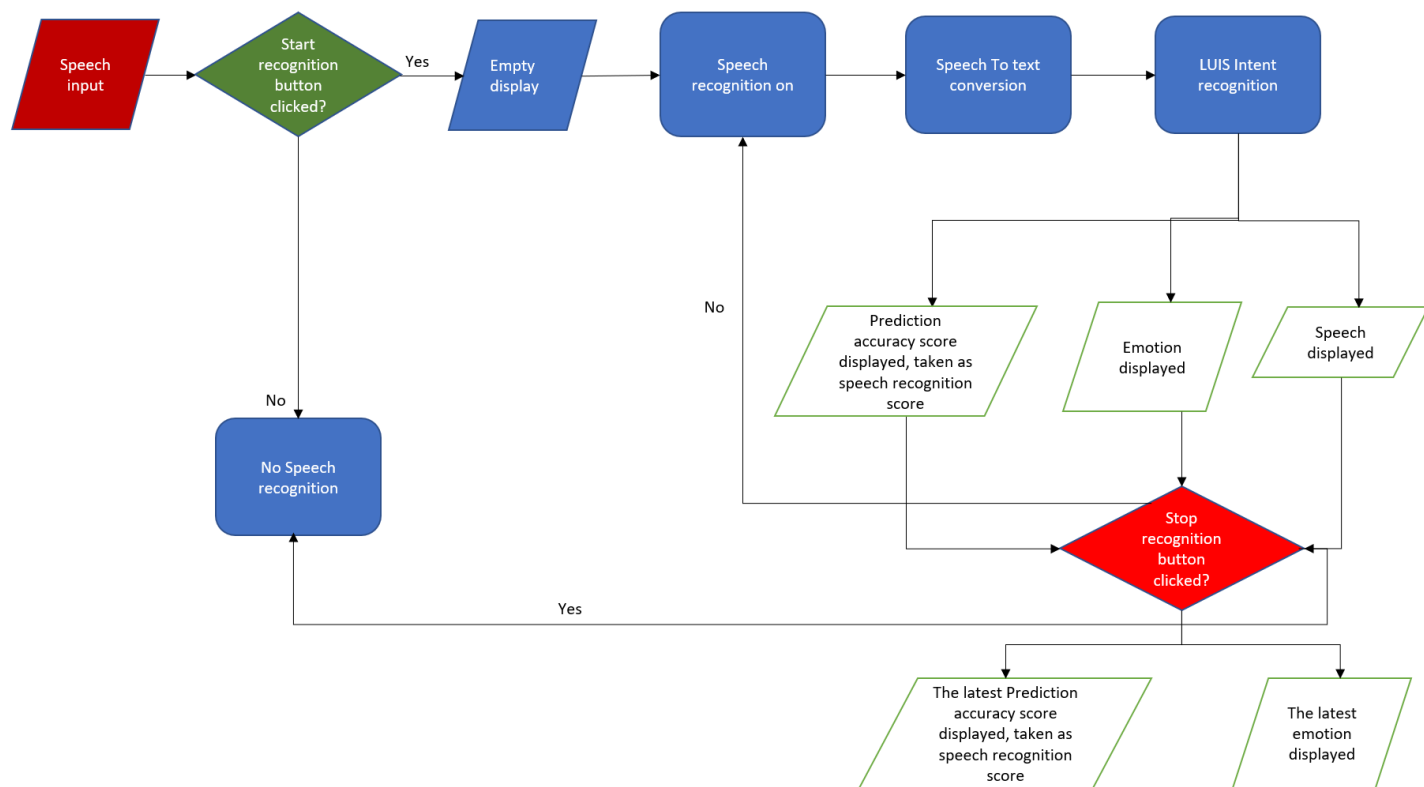


Figure 7: Emotional recognition from speech overall process.

The three steps in the figure 7 above- “Speech recognition on”, “Speech To Text conversion” and “LUIS intent recognition” are the major steps where cloud services are utilized, which are Speech

service and LUIS. Figure 8 shows the steps in different colors, which are shown further in detail in figure 17, with steps of the same color corresponding to the steps in the figure 17.

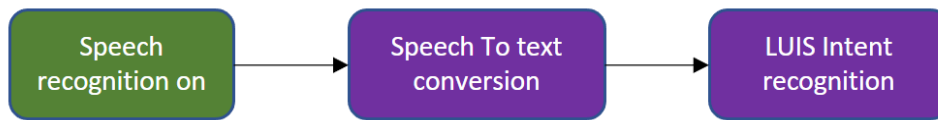


Figure 8: Major steps utilizing cloud service for emotional recognition from speech

Language Understanding (LUIS) is a cloud-based conversational AI service that applies custom machine-learning intelligence to a user's conversational, natural language text to predict overall meaning, and pull out relevant, detailed information [8]. LUIS works in a systematic manner. In the context of this project, the relevant details only are explained. LUIS contains two major components- intents and entities. An intent represents a task or action the user wants to perform. It is a purpose or goal expressed in a user's utterance or sentence [9]. In this project, the intent is “FindEmotion”, the purpose of which is to detect the emotion of the utterances contained in it. An entity is an item or an element relevant to the user's intent. It defines the data that can be extracted from the utterance and is essential to complete a user's required action [10]. In this project, the entity is “emotion”, which is of the type list. This list contains a set of words with their respective synonyms, or expressing the essence, if not the exact synonym. The eight basic emotions defined for this project are- joy, sadness, anger, anticipation, trust, surprise, disgust and fear. LUIS contains a database of these eight emotions, with each emotion containing a large number of lexicons which are either synonyms of the emotion or used as an alternative to describe the emotion. Figure 9 shows the entity “emotion” containing the words expressing the 8 distinct emotions, with each one having a set of words expressing the same meanings.

emotion

List

List items Examples Roles

List entities represent a fixed, closed set of related words along with their synonyms. List entities are extracted by an exact text m

 Import values  Delete

☐ Normalized values 

Synonyms

Type in a list item ...

☐ anger

agitated X roar X revving X backgroundcheck X hormonal X hungry
satins X shitberry X price X makenemad X unreliable X theworld X
brat X rent X raging X dumbasses X ranting X wrath X angryb

☐ anticipation

one X mycolour X vigilance X vigilant X hmmm X aroused X
hmm X savings X 23rd X ambition X highly X jame X intrigue
generate X prominent X eagerly X births X mistakes X springsummer

☐ disgust

enwwwwww X exatam X cruelty X betis X condoning X tampongirl X
fakes X molesting X grossest X sleazy X tranny X scrublife X sca
flabby X staystrongustin X pyongyang X disgusted X hatred X disgust

☐ fear

avetbank X apprehension X su4mh X aaaaaah X ied X coldswat
bombing X sweatyourfear X agoraphobia X 269 X bigday X waitingg
doom X wahlberg X israeli X panicattack X panicattacks X cbt X

☐ joy

namaste X firelynov11 X rejoice X inna X carefree X dollhouse X
aromatherapy X never X listening X lovinlife X contented X tranquil
xlliegoulding X christmaspirit X zen X uplifting X peaceofmind X pr

☐ sadness

gray X stillbirth X dreary X trund X bereavement X depress X ja
fallacy X depicts X mcready X summertime X eyore X grieving X
breakup X emo X prayfornewtown X regretful X 30rock X fringe X

☐ surprise

yada X preoccupied X jaden X easilyamused X needtofocus X amaze
scio13 X hypnotized X pleaseknow X cantgetenough X part2 X dian
nofocus X cuties X unfocused X glows X tryingtostudy X sidetracked

Type in value and press enter...

☐ trust

berminat X inclusion X admiring X socontagious X basicglitter X 3275
contagiously X hemat X 25font X theirfusedlife X newellie X trusty X
letter X on X pale X favs X end X 47k X bbw X n

Figure 9: Emotion words in entities and relevant keywords

Before importing the dataset in raw format, data cleaning is necessary. This includes converting the raw data into a suitable format for importing into LUIS. In this case, the keywords for each emotion were imported from a raw dataset taken from the National Research Council, Canada (NRC), which was contributed by Dr. Saif Mohammed [11]. Since the LUIS entity is of the type list, only the database in suitable format in .json can be imported. The following figure shows the suitable .json format for an emotion and its lexicon.

```
[
  {
    "canonicalForm": "anticipation",
    "list": [
      "crae",
      "mycolour",
      "vigilance",
      "vigilant",
      "hmmmmm",
      "aroused",
      "than",
      "dmt",
      "fines",
      "alphabetical",
      "wasnt",
      "inwood",
      "expecting",
    ]
  }
]
```

Figure 10: .json format for importing data.

Therefore, the original dataset in .txt format was converted into a suitable .json format using a python code written out of the main program. This code exports the first 1000 keywords of each emotion from the NRC dataset in the required .json format, and removes some unnecessary symbols like “#”, “/” so that words can be consumed in literal meaning by LUIS. Only the first 100 keywords were taken from the NRC dataset and imported into LUIS because LUIS has a current system limitation of not being able to upload a large number of keywords beyond a certain number.

```

a=[]
i=0
d={}

#Add all 8 emotions as keys from the dataset
for item in dict1:
    d.setdefault(item['canonicalForm'], []).append(item)

for val in d.keys():
    d1={}
    d1["canonicalForm"]=val
    d1["list"]=[]

    k=0
    for i in d[val]:
        #Remove unnecessary characters
        p=(i["list"]).replace("#", "").replace("/", "")

        # add the first 100 keywords for the particular emotion
        d1["list"].append(p)
        k=k+1
        if(k>100):
            break
    a.append(d1)

#save into a .json file
out_file = open("test3.json", "w")
json.dump(a, out_file, indent=4)
out_file.close()

```

Figure 11: Python program lines for data cleaning

This completes the data cleaning outside of the program. Figure 17 shows this process.

As each emotion contains a set of keywords used to recognise that particular emotion, this approach becomes lexical analysis. The set of keywords for a particular emotion interpreting the same meaning is the lexicon for that keyword. The keywords are the lexemes. After the lexicons for the emotions are imported into the dataset, the words in the utterances inside the intents are assigned the entities by the programmer initially.

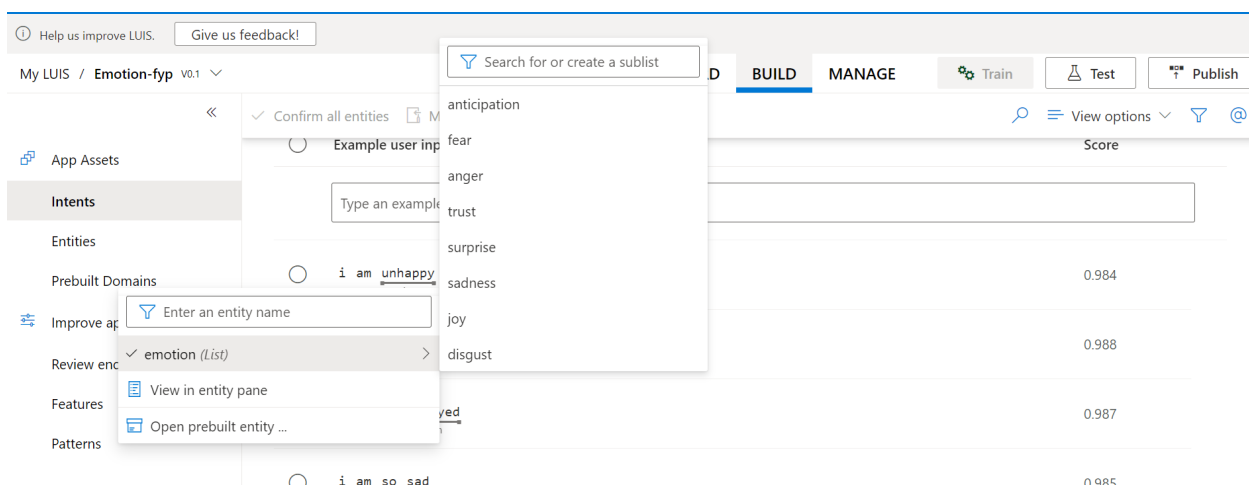


Figure 12: Mapping with relevant lexicons on LUIS portal.

Then, the whole model on the LUIS system is trained, tested to see if some lexemes may go unrecognized, if so newer lexemes would be added to the entities and some additional utterances may also be added. After that, the model is published. Figure 17 shows the training of the model on LUIS.

Inside the program, when the program runs and the recognition is enabled for speech, a “SpeechConfig” object is created. “IntentRecognizer” takes this as an input and creates an intent recognizer object. A model for LUIS intent recognition is created unique to the LUIS portal used in the project. The intent recognizer object takes the model, name of intent and entity as inputs. By default, microphone input is enabled for receiving speech from the user. When speech is received, it is converted to text and displayed on the screen of the program. This text is received by the LUIS instance in the program.

```
// Creates an instance of a speech config with specified subscription key
// and service region. Note that in contrast to other services supported by
// the Cognitive Services Speech SDK, the Language Understanding service
// requires a specific subscription key from https://www.luis.ai/.
// The Language Understanding service calls the required key 'endpoint key'.
var config = SpeechConfig.FromSubscription(LUISAppKey, LUISRegion);
// Creates an intent recognizer using microphone as audio input.
intentreco = new IntentRecognizer(config);

// Creates a Language Understanding model using the app id, and adds specific intents from your model
var model = LanguageUnderstandingModel.FromAppId(LUISAppId);
intentreco.AddIntent(model, "FindEmotion", "emotion");
```

Figure 13: Creation of the LUIS intent recognition model code.

If the words in the text sentence match any of the lexemes of any of the lexicons of an emotion in the LUIS database, the detected emotion is displayed on the screen. The code is explained in figure 16.

```
void ProcessIntent(IntentResult intent)
{
    string attribute = "";

    // Extract the entities if any, here it is emotions
    if (intent.entities.Count > 0)
    {
        recognizedString += $"{Environment.NewLine}Entities=";
        for (int i = 0; i < intent.entities.Count; i++)
        {
            if (intent.entities[i].type.ToLower() == "emotion")
            {
                attribute = intent.entities[i].entity;
            }
        }
    }
}
```

Figure 14: Intent processing code

```

List<string> sadness = new List<string>();
List<string> surprise = new List<string>();
List<string> trust = new List<string>();

string path = Application.dataPath;
string kk = File.ReadAllText(path+"/test3.json");
Console.WriteLine(kk);
JSONArray jArray2 = JSONArray.Parse(kk);

foreach (JObject jobject in jArray2)
{
    if ((string)jobject["canonicalForm"] == "anger")
    {
        foreach (var k in jobject["list"])
        {
            anger.Add((string)k);
        }
    }
    if ((string)jobject["canonicalForm"] == "fear")
    {
        foreach (var k in jobject["list"])
        {
            fear.Add((string)k);
        }
    }
}

```

Figure 15: Adding the values in each list.

```

if (anger.Contains(attribute))
{
    Emotions.text = "anger";
}
else if (disgust.Contains(attribute))
{
    Emotions.text = "disgust";
}
else if (anticipation.Contains(attribute))
{
    Emotions.text = "anticipation";
}
else if (fear.Contains(attribute))
{
    Emotions.text = "fear";
}

```

Figure 16: Intent recognised correctly, then show the emotion on screen.

Along with this, the confidence score for detection accuracy is taken as the score for the speech recognition. Figure 17 shows the part of interaction of the program with the cloud service. As shown in figure 8 about the major steps in short, these are explained in detail in figure 17.

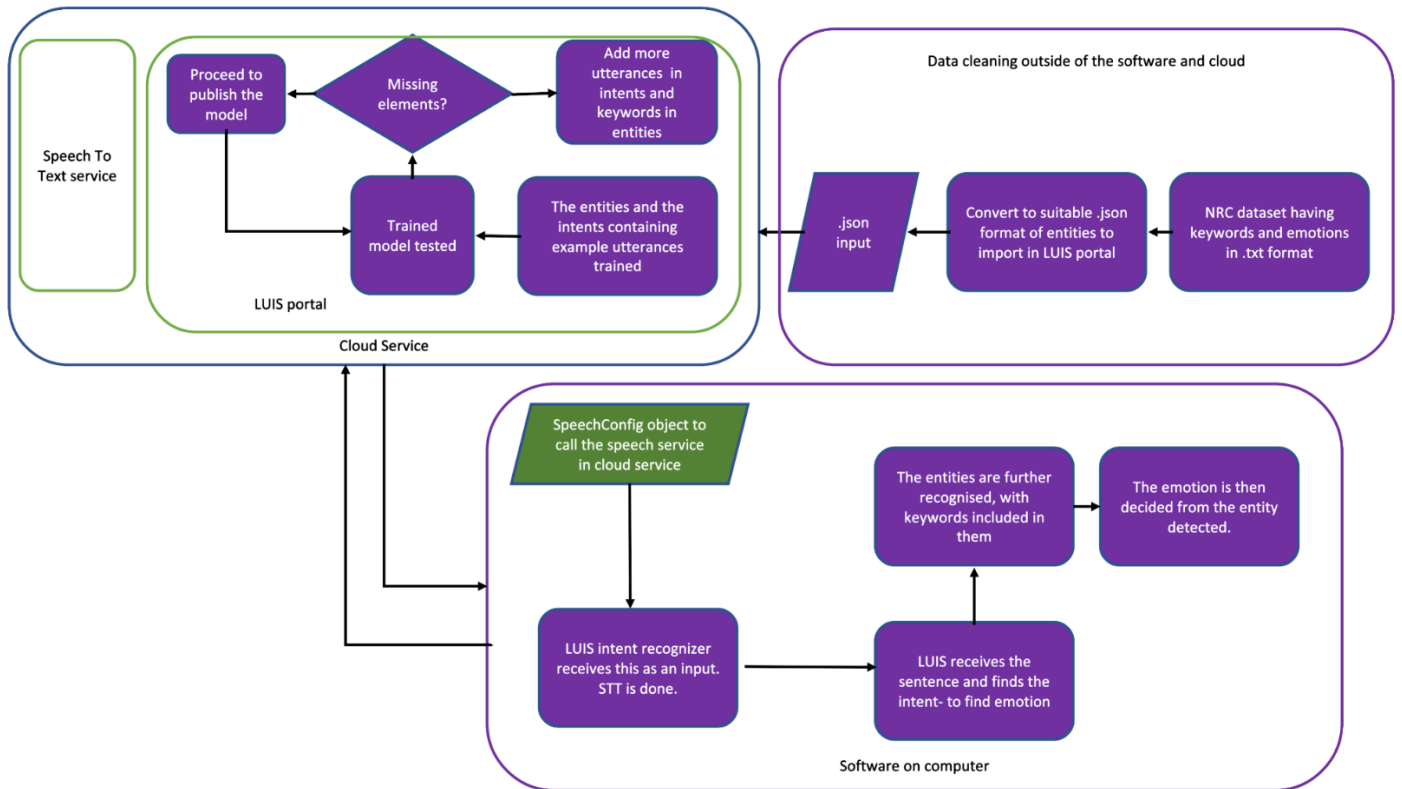


Figure 17: Interaction of program with cloud service and data cleaning.

4.4 Facial expression recognition and extraction

The facial tracker attached to the HTC VIVE Pro headset detects the facial expressions of the user from the lower mouth only, that is, the region of the face from below the nose to the end of the chin. Therefore, only the data of the face of the lower mouth will be saved. The lower mouth movements of the user are tracked by the facial tracker and replicated by the 3D face model in the program, having a human-like face. This interaction is possible due to the API of SRanipal, which connects the facial tracker and the Unity3D program, as shown in Figure 1 previously.

A list of 27 facial expressions are already provided by SRanipal SDK. As discussed before, there are 8 basic emotions to be detected from speech, similarly each of these 8 emotions are to be detected from the face too. Therefore each of the emotions is mapped with an expression from the

given list of facial expressions. Figure 18 shows the 8 basic emotions with mapped facial expressions.

	A	B
1	Emotion	Lip Shape
2	joy	Mouth_Smile_Right
3	sadness	Mouth_Sad_Left
4	anger	Cheek_Puff_Left
5	disgust	Mouth_UpperRight_Up
6	surprise	Mouth_O_Shape
7	fear	Mouth_Lower_Inside
8	trust	Cheek_Suck
9	anticipation	Mouth_Lower_Overlay

Figure 18: List of emotions.

When the user clicks on the “Start Recognition” button, facial recognition is enabled. The facial expressions being tracked by the facial tracker are replicated by the 3D face model. The facial data transferred from the facial tracker to the program is used to calculate the probability weight of the 27 basic facial expressions provided for the basic version of SRanipal. As the weights are mentioned in the code but not enough information is provided as to what the weights are, but these show behavior of increasing value when similar face expressions are being made, and always lie between 0 and 1. Thus, it is assumed that the “weight” is probability weight. Probability weight denotes the probability of the facial expression made, matching to the original one. If the facial expression made is more similar to the “mouth smile right” expression, then the probability weight of this expression from the given list of expressions increases, lies in the range [0.5,1). If the facial expression made is more similar to the “mouth sad left” expression then the probability weight of the “mouth sad left” from the given list increases, in the range [0.5, 1). For the expression made similar to “mouth sad left”, the probability weight of the expression “mouth smile right” is low, lies in the range (0,0.5). Just like each facial expression of a human being is unique to an expressed emotion, therefore the list of 27 expressions is used to determine the emotion of the user. Larger the probability weight, stronger the emotion is. Figure 20 and 22 show how the values for each facial expression change. Here, “joy” refers to “mouth smile right” and “sadness” refers to “mouth sad left” in the console.

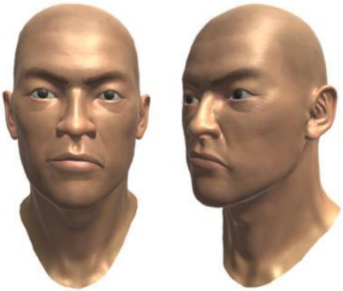
Mouth_Sad_Left	
	Description
	This blendShape lowers the left side of the mouth further with a higher value.

Figure 19: Mouth sad left expression. Reference- SDK of SRanipal [12].

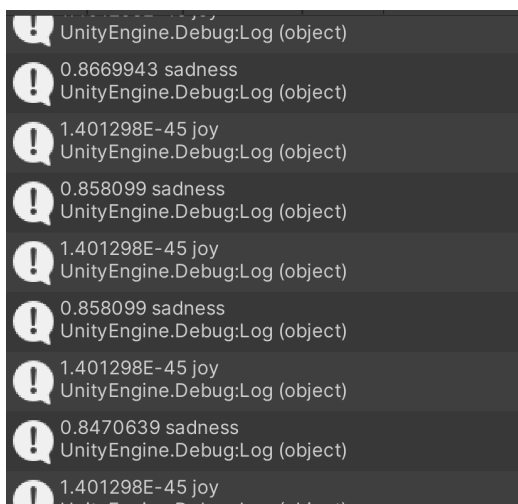


Figure 20: Weights for mouth sad left expression.

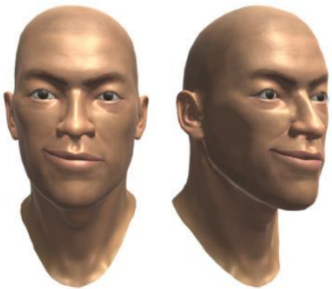
Mouth_Smile_Right	
	Description
	This blendShape raises the right side of the mouth further with a higher value.

Figure 21: Mouth smile right expression. Reference- SDK of SRanipal [12].

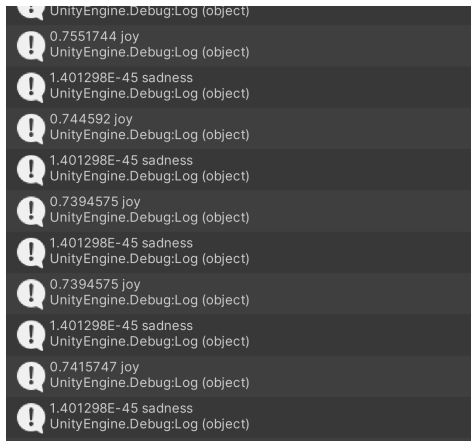


Figure 22: Weights for mouth smile right expression.

When the “Start Recognition ” button is clicked, the detection of emotions is enabled in the program, i.e., “DetectEmotions()” method is called. This helps to call the function “executeInWithFixedTimingss()” in “Update()” method of Unity3D, where data is received in each frame when the program runs. This means that the facial expressions data is received in each frame.

```
public void DetectEmotions()
{
    clicked = true;
}
```

Figure 23: Enable the detection of emotions when the “Response” button is clicked.

```
if (clicked == true)
{
    executeInWithFixedTimingss();
}
```

Figure 24: Call the function in the Update method if the “Response” button is clicked.

The function “executeInWithFixedTimingss()” maps the lip shapes with the emotions to receive the probability weights of the facial expressions made for each emotion value. Each emotion has a list, where all the weights measured are saved, only if the probability weight captured from the facial tracker is high enough, i.e., in the range $[0.5, 0)$. This is important because the facial expression related to the emotion is accurately made if the value of probability weight is high.

1 reference

```
private void executeInWithFixedTimingss()  
{  
    //Probability weights of the emotions  
  
    float happy = LipWeightings[LipShape.Mouth_Smile_Right];  
    float sad = LipWeightings[LipShape.Mouth_Sad_Left];  
    float anger = LipWeightings[LipShape.Cheek_Puff_Left];  
    float disgust= LipWeightings[LipShape.Mouth_UpperRight_Up];  
}
```

Figure 25: Assign the probability weights to emotions.

```
//Add probability weights to their respective lists only if high  
if (happy >= 0.5 && happy < 1)  
{  
    Debug.Log(happy+ " happy");  
    happyWeights.Add(happy);  
}  
  
if (sad >= 0.5 && sad < 1)  
{  
    Debug.Log(sad + " sad");  
    sadWeights.Add(sad);  
}
```

Figure 26: Adding probability weights to variables of emotions only when it is high in range [0.5,1).

Until now, the facial expressions are only being tracked and saved in the program but no final or overall emotion has been detected. Once the user clicks on the “Stop recognition” button, the tracking of facial expressions is stopped. An average is taken for the saved probability weights with respect to each facial expression, only if the list of weights for a particular emotion is not empty. A dictionary is created with the emotion as the key, and the average weight as its value. For example, if the emotion is “fear” and the associated list “fearWeights” is non-empty having some tracked probability weights of facial expressions related to “fear”, then the key is “fear” with the value being the average of all values in “fearWeights”. The average weight values of all emotions are added to the list “allKeys”.

```

//Take average of weights when recognition is stopped
0 references
public void AvgWeights()
{
    clicked = false;

    if (happyWeights.Count != 0)
    {
        float happyWeightsAvg = happyWeights.Average(); addIn(happyWeightsAvg, "joy");
    }
    if (sadWeights.Count != 0)
    {
        float sadWeightsAvg = sadWeights.Average(); addIn(sadWeightsAvg, "sadness");
    }
    if (angerWeights.Count != 0)
    {
        float angerWeightsAvg = angerWeights.Average(); addIn(angerWeightsAvg, "anger");
    }
}

```

Figure 27: Take average of weights

```

6 references
private void addIn(float weight, string emotion)
{
    if (weight!= 0)
    {
        allWeights.Add(weight, emotion);
        allKeys.Add(weight);
    }
}

```

Figure 28: Add all emotions and wrights in a dictionary and just weights in another list.

The expression that has the maximum average value in the list “allKeys”, the emotion mapped with that particular emotion is considered as the final emotion detected from the face. This score is saved as the score for facial recognition. For example, if the “mouth smile right” expression has the highest average value, then the emotion mapped with this facial expression, i.e. “joy” is considered as the final emotion detected from the face and this average weight is considered as the score of facial recognition.

```

//Find the maximum average weight of all the expressions
float maxEmotion = allKeys.Max();
//Final emotion corresponds to the expression having the maximum average weight
string finallyemotion = allWeights[maxEmotion];
Debug.Log("overall "+finallyemotion);
emotionOfFace.text = finallyemotion;
//The maximum average weight is the score as well
faceScore = maxEmotion;

```

Figure 29: Finding the score and final emotion.

Figure 30 shows the brief flow of facial emotional recognition.

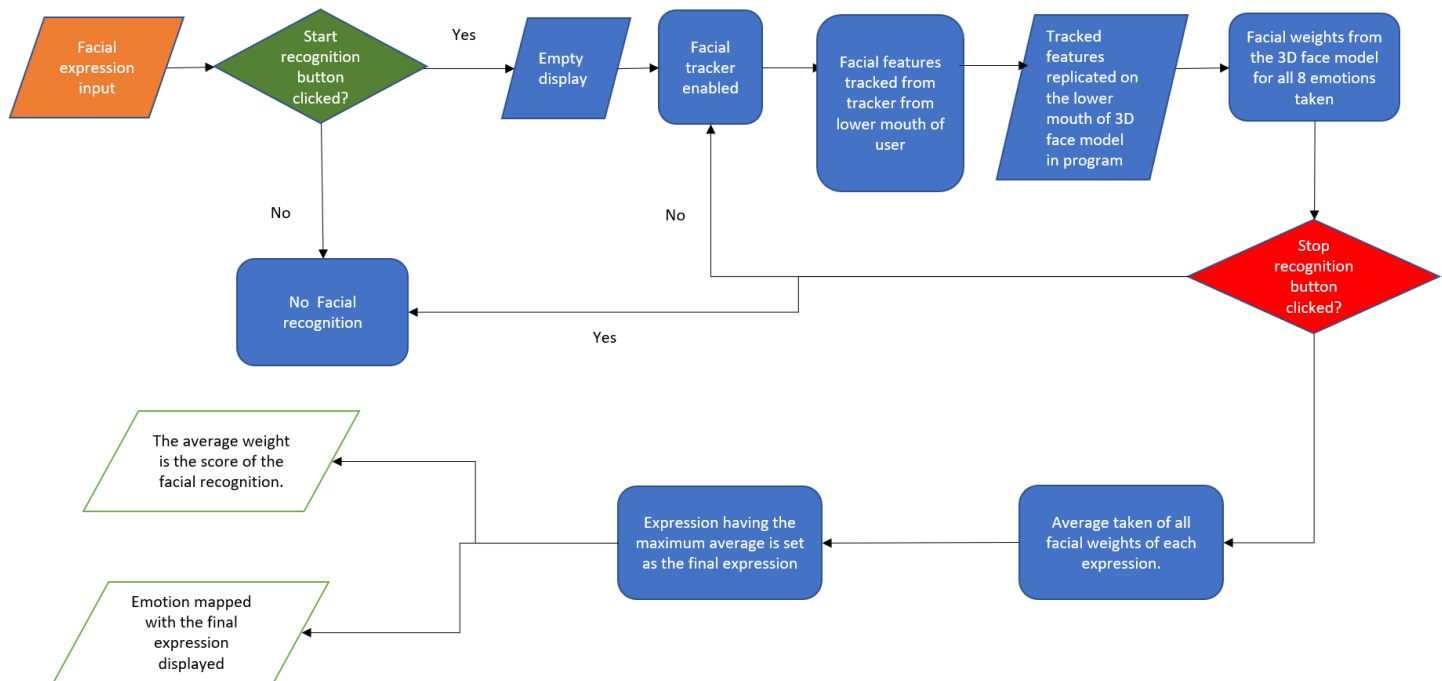


Figure 30: Emotional recognition through face

4.4 Response by agent

This is the final stage where the final emotion is decided after getting the inputs from both speech and facial recognition. The speech score and facial recognition score are compared. If the scores are different, then the score with a higher value determines the final emotion of the user. Figure 32 shows the decision of final emotion.

```
public void ShowFinal()
{
    //Get scores of the speech and facial recognition
    speech = this.GetComponent<DummyOne>().score;
    face = avatarFace.GetComponent<SRanipal_AvatarLipSample>().faceScore;
    scoreSpeech.text = speech.ToString();
    scoreFace.text = face.ToString();

    if (speech > face)
    {
        emotionFinal.text = emotionSpeech.text;
    }
    else if (speech < face)
    {
        emotionFinal.text = emotionFace.text;
    }
    else
    {
        emotionFinal.text = emotionFace.text;
    }
}
```

Figure 31: Compare the scores and decide emotion.

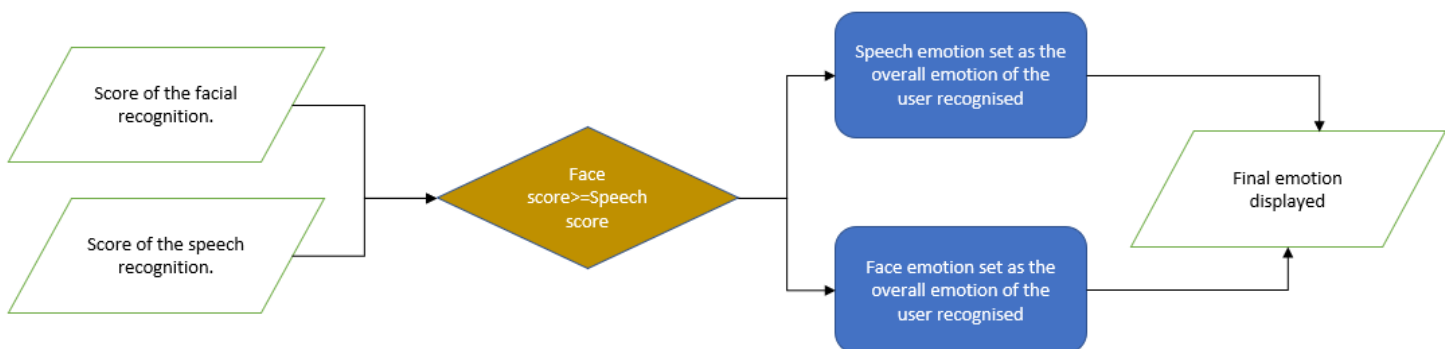


Figure 32: Final emotion decided from both speech and facial tracking input channels

Thus, the determined emotion further produces the output by the agent in the program. A list of responses to be delivered for each emotion is already saved in the program. According to the finally detected emotion, the response is decided. This response is in the form of text, which is converted to speech. Figure 33 shows the list of responses for each emotion.

	A	B
1	Emotion detected	Response
2	anger	Oh please calm down. Anger leads to destruction.
3	disgust	Oh yes I understand how disgustign that can be huh!
4	anticipation	Seems you are expecting this to happen really.
5	fear	Come on calm down, don't be afraid.
6	joy	I am so happy for you! Let's celebrate.
7	sadness	Oh so sorry, that's bad. Karma hits back, don't worry.
8	surprise	Well, some unexpected incidents happen certainly.
9	trust	Trust is very important, I can see that from your words.

Figure 33: List of responses with respect to each emotion.

```
// Show the response by the agent
0 references
public void Response()
{
    string finalEmotion = emotionFinal.text;

    if (finalEmotion == "anger")
    {
        responseAgent.text = "Oh please calm down. Anger leads to destruction.";
    }
    else if (finalEmotion == "disgust")
    {
        responseAgent.text = "Oh yes I understand how disgustign that can be huh!";
    }
    else if (finalEmotion == "anticipation")
    {
        responseAgent.text = "Seems you are expecting this to happen really.";
    }
    else if (finalEmotion == "fear")
    {
        responseAgent.text = "Come on calm down, don't be afraid.";
    }
}
```

Figure 34: Code to provide a suitable response as per the corresponding emotion.

When the response button is clicked, the final emotion is taken as the input and the appropriate response is searched in the program from the list of responses. For ease of understanding, the text response is shown on the screen. This response is converted into an audio file using IBM Watson Cloud's Text-To-Speech (TTS) service. A 3D agent avatar in the program receives this audio file for processing. The 3D agent avatar then lip synchronizes (lip sync) with the speech content in the audio file, while it is playing. Hence, the 3D agent avatar lip sync the response to the user. This 3D agent avatar for lip sync is provided by Oculus and examples of the lip sync integration with IBM Watson Cloud have been contributed by Dr. Scott Hwang as mentioned earlier in section 3.1.2, from where the inspiration is taken. Figure 35 shows the process of generating the text response and lip sync in brief.

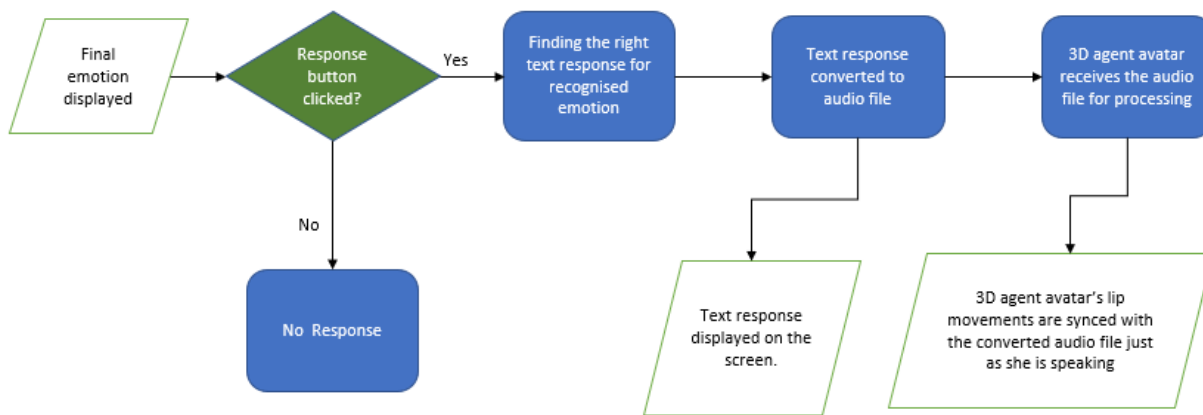


Figure 35: Text to speech and lip synced response by the agent.

Further explained are the details of interaction of the program with IBM Watson Cloud and some details of lip sync.

For TTS conversion, the program interacts with IBM Watson Cloud. An authenticator is created in the program which is further used as an input to create TTS service. This TTS service is used to add texts in a queue so that texts can be converted to speech one by one.

```
// Create credential and instantiate service
tts authenticator = new IamAuthenticator(apikey: settings.tts apikey);
```

Figure 36: Create authenticator.

```
tts_service = new TextToSpeechService(tts_authenticator);
```

Figure 37: Create TTS service

Hence, the text is added to the queue first. The text is processed into audio bytes. After conversion of text into audio bytes, the audio bytes are converted into an audio file. If any TTS converted audio file is playing from before, then the text is queued for being processed into audio bytes until the current audio file finishes playing.

```
nextText = textQueue.Dequeue();
Debug.Log(nextText);
```

Figure 38: Create authenticator for TTS service.

```
synthesizeResponse = response.Result;
clip = WaveFile.ParseWAV("myClip", synthesizeResponse);
```

Figure 39: Process response and convert to audio clip.

The TTS converted audio file is processed by the 3D agent avatar to generate phonemes. These phonemes are then further processed to render the lip shapes of the 3D avatar agent as per the respective pronunciations. Figure 40 shows the interaction of the program with IBM Watson Cloud and lip sync of the avatar.

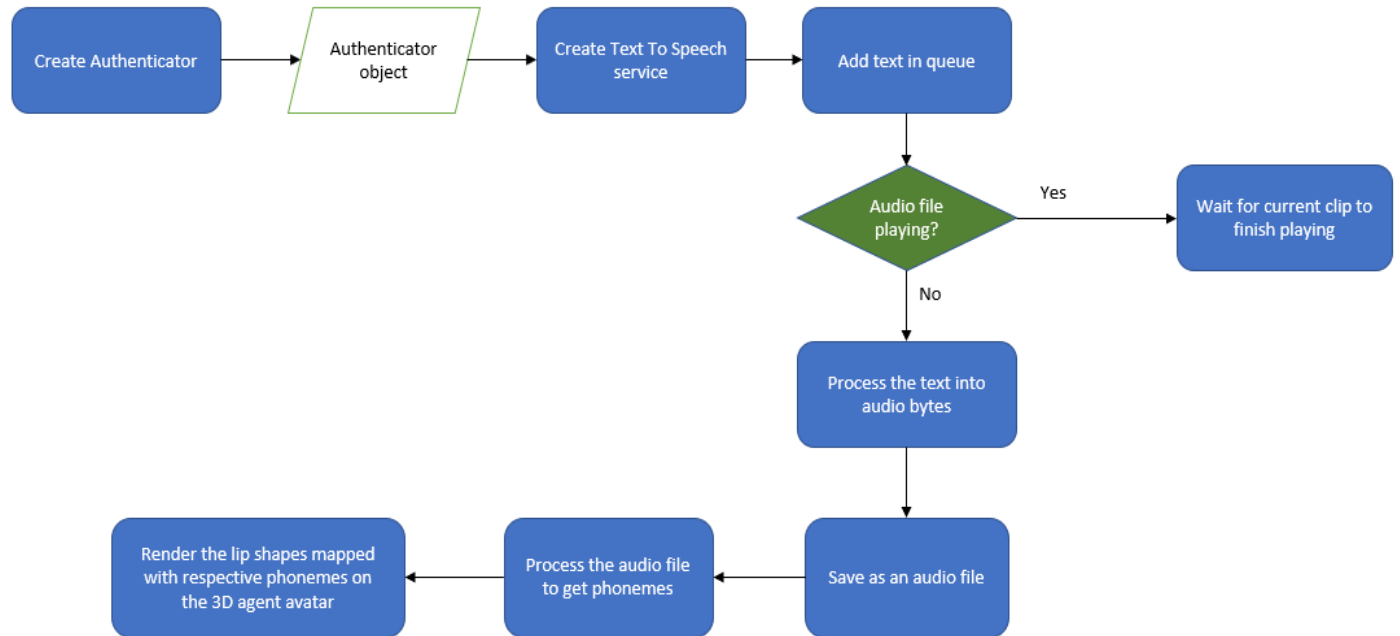


Figure 40: TTS conversion with IBM Watson Cloud and lip sync

5. Results

28 observations were taken while testing the application. Scores obtained from emotional recognition from speech and facial expressions are noted down along with the sentences. The lexemes which are contained in the sentences and related to a certain emotion are highlighted by a strike through. The time for how long the facial expressions are made are also noted down, to see if the score can be influenced by getting more number of weights as the recognition runs for a longer time. It is also mentioned if there were mixed expressions during the period of time these were tracked. The lexemes for each emotion that are not synonymous were deliberately spoken to test the intent recognition accuracy. The facial expressions were made different than the spoken words, to see how effective are they, as compared to spoken words.

	A	B	C	D	E	F	G	H
1	Emotion from words	Score of speech recognition	Emotion from face	Score of facial recognition	Sentence	Seconds of face	Mixed faces?	Results
2	joy	0.9600479	joy	0.8937796	I am happy	30	Yes	joy
3	disgust	0.84636116	disgust	0.8114765	I am disgusted	30	Yes	disgust
4	anger	0.9624286	disgust	0.7834183	I am angry	20	Yes	anger
5	surprise	0.94713885	disgust	0.7885766	I am so surprised	20	Yes	surprise
6	fear	0.8488961	disgust	0.6644828	I am fearful	10	Yes	fear
7	fear	0.8830139	fear	0.8249102	I am fearful	10	No	fear
8	sadness	0.9853371	sadness	0.7644198	I am so sad	10	No	sadness
9	None		joy	0.867041	I am pleased	10	No	None
10	joy	0.94149417	sadness	0.8577735	I am joyful	30	No	joyful
11	joy	0.84636116	surprise	0.9341128	I am joyful	30	No	surprise
12	None		anger	0.8587316	I am so afraid	20	No	None
13	sadness	0.9853371	joy	0.9606678	I am so sad	30	Yes	sadness
14	disgust	0.87616676	sadness	0.905396	I am so disgusted	30	No	sadness
15	None		joy	0.9317733	I am upset	10	No	None
16	None		disgust	0.8985522	Ewww-that is so pathetic (Here, "Eww" is read as "You by program")	10	No	None
17	fear	0.8506737	joy	0.8848188	The subject is psychological	10	Yes	joy
18	trust	0.8753927	sadness	0.8939628	I stand up for equality	10	Yes	sadness
19	None		joy	0.7938358	write your names in alphabetical order	10	Yes	None
20	anticipation	0.8520406	joy	0.9134619	what is your hobby	10	No	joy
21	disgust	0.7408477	anger	0.903406	I strongly recommend to study	10	No	anger
22	joy	0.9537616	disgust	0.8063838	I am interested to watch movies about avengers	10	Yes	joy
23	sadness	0.7210067	sadness	0.9431221	Mirror reflects your face	20	No	sadness
24	surprise	0.8520406	surprise	0.8964818	The work is flawless	20	No	surprise
25	anger	0.8548859	anger	0.9748615	what is the price of this item	20	No	anger
26	joy	0.6618941	joy	0.918696	there is stillness in water	20	No	joy
27	disgust	0.722674	disgust	0.9073125	the world is applauding the genius	10	No	disgust
28	fear	0.8540822	fear	0.8579254	take your medicines -from pharmacy	30	No	fear

Figure 41: Results

Until the 9th observation, the face score is always lesser than the speech score. This might be because the words detected are very obviously related to the emotion, as these are lexemes included in the emotion of a lexicon and only a search operation must be needed to find one lexeme. Even if the time span for making homogeneous facial expressions is higher, the score is still less than speech recognition. Also, the words spoken were synonymous or very commonly used words from which it is obvious what emotion is detected. When some words were spoken out of the lexicon, the emotion could not be detected, hence the emotion is written as “None” and the final emotion was also written as “None” because there could be no comparison with emotion detected by the face.

To test for a fair comparison, 0.1f was added to face score from the 10th result onwards. 0.1f was added because in many of the previous cases and the cases in this test as well, the difference of 0.1 was always there between the score for face and speech. Therefore, 0.1f is decided to be added. The value 0.1f is only added if the score does not become greater than or equal to 1.

```
//The maximum average weight is the score as well
if (maxEmotion < 0.9f)
{
    faceScore = maxEmotion + 0.01f;
}
else
{
    faceScore = maxEmotion;
}
```

Figure 42: Adding 0.1 to the score of face recognition from 10th result onwards.

From the 10th result onwards, in most of the tests, even after making mixed faces for a shorter time span, the score of facial recognition was higher than speech recognition. Some words in the lexicon of the emotion but not synonymous were deliberately spoken from 10th observation onwards, some of which were not well detected. For example, in the 19th row, the word “alphabetical” in the sentence spoken should have been detected as “anticipation” as it is included in the lexicon. However, it went undetected. In row 28th, it is interesting to note that the word “medicine” is included in the lexicon of “fear” which has a high detection score, though the word is purely an object but associated with an emotion. Similarly, in row 27th, the word “applauding” is included in the lexicon of “disgust”, though it is a verb and does not in real life explain any emotion of disgust. In fact, “applauding” is usually associated with events of celebration and happiness. Hence, it is observed that if the words are included inside a lexicon, no matter if it is a synonym or totally unrelated to the emotion, when spoken will be detected as a part of that emotion.

The two pipelines of recognition, which are speech and facial recognition, are different. For speech recognition, the score obtained is the confidence of prediction between 0 to 1, indicating lowest to highest value of confidence. However, no such explicit information is mentioned for the facial tracker returning the weights of the facial expressions. In the documentation it is only mentioned that the value turns higher for the facial expression that is made. In the code provided by the SDK of facial tracker, it is only mentioned that weights are returned. These weights are not explained enough, except for the fact that these are obtained from the prediction data.

```

for (int i = 0; i < WeightingCount; ++i) {
    Weightings[(LipShape)i] = LipData.prediction_data.blend_shape_weight[i];
}

```

Figure 43: Assign all weights when the program is running.

```

/// <summary>
/// Gets weighting values from anipal's Lip module.
/// </summary>
/// <param name="shapes">Weighting values obtained from anipal's Lip module.</param>
/// <returns>Indicates whether the values received are new.</returns>
1 reference
public static bool GetLipWeightings(out Dictionary<LipShape, float> shapes)
{
    bool update = UpdateData();
    shapes = Weightings;
    return update;
}

```

Figure 44: Update newer weights. The comments state that “weights” are obtained from the Lip module.

During development as mentioned in system design and architecture, the facial expression, if made correctly, bears a higher value in the range [0.5,1). This assumption was made that it might be the probability of the weighting of the facial expression made, how well the facial expression in front of the tracker is made corresponding to the given facial expressions in the list. Since the two pipelines are different for output values for facial recognition and speech, therefore the comparison may not be fully correct as there is lack of information about the face tracking.

6. Impact

As affective awareness agents in different forms have been developed by various academic institutes and companies for research and development purposes, the main objective lies in how to understand and respond to a user in a realistic and empathetic manner. We aim to create an intelligent friend, instead of developing a master-servant relationship. The ultimate purpose is to understand the user’s behavior through different channels. As the prototypes have already been developed for affective awareness agents, bringing this into VR is an advantage. As VR is an immersive environment, it makes the user feel more involved in the process of talking to the agent in an alternate environment where he or she can feel more belonged. This is researched upon and supported by Wu et al [13].

Though there are various chatbots created for the user to play with, which would provide a suitable response, and some universities and companies have already built 3D chatbots having a human-like avatar. How this project is different is because of two major points. The first one being that it is a VR environment, providing the user an alternative environment for self expression and understanding by the virtual agent. Second, the user's feelings are calculated by the facial expressions of the user and the speech. This ensures a more accurate analysis of the feelings instead of relying on only one channel. Here, facial expressions factor hold more weightage because a person's face can seldom lie about the feelings.

The success of this project will provide a new solution and direction in the ongoing research of affective awareness agents, not just in VR but in other forms too.

7. Future improvements

The project was created with the intention to utilize some accessible and proven techniques for emotional recognition of the user, from speech and facial expressions, being the primary aim. The secondary aim was to provide an appropriate response by the program to the user.

For the emotional recognition from speech using LUIS by Azure Cognitive service, there is a system limitation. The system was not able to accept all the keywords from the NRC dataset, a possible reason is the memory size. Therefore, only 1000 keywords were uploaded in the LUIS portal from the NRC dataset. To tackle this issue, the issue is already submitted to the relevant forum and the team may bring improvements soon.

As the project follows a lexical approach to analyze speech, there may be some obvious words missing from the dataset when imported in LUIS. We should add the words from our side if the words are not detected by the program due to absence in LUIS. Also, lexical analysis uses the relevant keywords to determine the emotions, instead of considering the wholesome meaning of the sentence. For example, if there is a sentence uttered by the user "I am not sad", then the lexical analysis will detect the emotion to be sadness as the keyword "sad" exists in the sentence. To improve on this aspect of overall understanding, sentiment analysis can be used. Currently, the support for Python libraries for machine learning is supported by Unity3D but still under

development. A model can be trained to determine the emotions from the sentence and imported in Unity3D for better and wholesome emotional recognition.

The facial expressions are detected in such a way that the expression having the highest weighted average will be considered as the user's expression. This is a drawback as it may ignore the mixed feelings of the user. For example, if the user smiles upwards most of the time but makes a smile down face for a small time, then the upward smiling expression will be considered with higher weightage and the downward smiling face will be disregarded totally. This is inaccurate in a way as the upward smiling expression will only give an expression that the user is only happy and not sad. The boundary when the user changes expression is clear

Tone analysis may be imparted as to determine emotions of a user, in order to have one more factor for detection.

Chatbot features may be used in the program so that the responses may not be typed in the program, instead the responses are generated dynamically. The project will become like a virtual 3D affective chatbot if chatbot should be included.

The project may be built as a game instead of a chatbot. More actions like opening a new scene for the user to discover when a particular emotion is expressed, can be included. This can be a direction to create fantasy games in VR supported by emotional recognition from facial expressions and speech.

8. Discussion

For the facial recognition feature, more sophisticated algorithms may be developed for advanced levels of facial recognition. As discussed with a professor in our department, an interesting research method can be taking the raw data tracked by the facial tracker and plotting all the features from the data as points in the 3D feature space. Then, a threshold will be defined for the features of each emotion. By defining the thresholds, regions are created in the feature space, belonging to each of the emotions. If the feature point lies inside the region of a certain emotion, then the feature point will represent that particular emotion.

For example, in the diagram below, the threshold of "smiling face" is set in the range $[0.5, 0.7]$ on the Y-axis. The region consumed by the planes of $Y=0.5$ and $Y=0.7$ contain the emotion of "smiling

face”, which means that any feature point lying in this region or very near to the region will be categorized into the emotion.

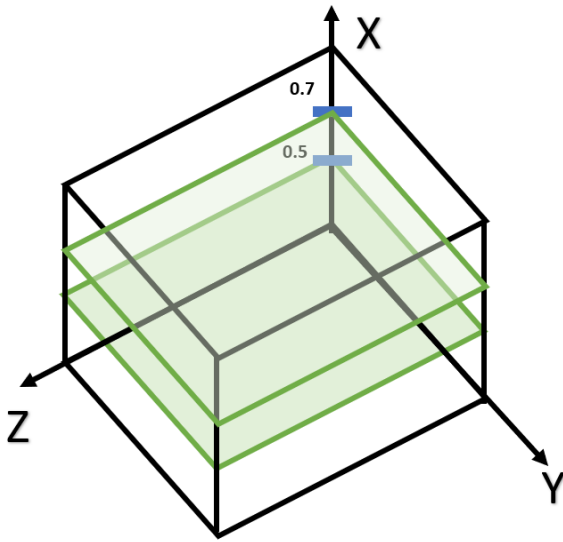


Figure 45: Region containing a smiling face.

Multimodal analysis methods may be adapted in future, which will plot not only the points of the speech in the feature space but also facial expressions. As all the features may be plotted in the feature space from speech and facial recognition channels, then these features may be input in the classifier to detect the final emotion, just from the feature space.

9. Conclusion

This is a motivating project to pursue as there are certain developments in the area of affective awareness, which gives rise to newer ideas and methods. Therefore, the techniques are taken from these projects which have already been established well and adapted in the project. Due to time and resource constraints, only speech and facial expressions were taken into account for emotional recognition. Facial recognition and intent recognition from text have been implemented in different ways, other than what is available online, which hopes to bring an improvement in ease of recognition, by using the already existing methods.

While developments are being made by various cloud services and softwares for implementation of machine learning techniques which are used in emotional recognition, there are multiple ways to utilize them to bring out an effective application made by integration of these tools.

Overall, the basic principles of emotional recognition by speech, facial expressions and other methods like tone are important when designing affective awareness agents. There may exist different combinations of designing the algorithms to take the data from different channels from the user and later analyze the emotion, which depends on the researcher's study.

10. References

- [1] Ahmed, I. and Harjunen, V., 2021. *Touching virtual humans: Haptic responses reveal the emotional impact of affective agents*. [online] ieeexplore.ieee.org. Available at: <<https://ieeexplore.ieee.org/document/9258960/authors#authors>> [Accessed 20 October 2021].
- [2] Numata, T., Sato, H., Asa, Y., Koike, T., Miyata, K., Nakagawa, E., Sumiya, M. and Sadato, N., 2020. Achieving affective human–virtual agent communication by enabling virtual agents to imitate positive expressions. *Scientific Reports*, 10(1).
- [3] IEEE Spectrum. 2021. *What Is the Uncanny Valley?*. [online] Available at: <<https://spectrum.ieee.org/what-is-the-uncanny-valley>> [Accessed 20 October 2021].
- [4] Citeseerx.ist.psu.edu. 2022. [online] Available at: <<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.113.1363&rep=rep1&type=pdf>> [Accessed 20 April 2022].
- [5] Mittal, T., Guhan, P., Bhattacharya, U., Chandra, R., Bera, A. and Manocha, D., 2022. *EmotiCon: Context-Aware Multimodal Emotion Recognition using Frege's Principle*. [online] [arXiv.org](https://arxiv.org). Available at: <<https://arxiv.org/abs/2003.06692>> [Accessed 20 April 2022].
- [6] Kim, J., Bee, N., Wagner, J. and Andr e, E., 2022. *Emote to win: Affective interactions with a computer game agent*. [online] [DL.gi.de](https://dl.gi.de). Available at: <<https://dl.gi.de/handle/20.500.12116/28838>> [Accessed 20 April 2022].
- [7] Hwang, S., 2022. *GitHub - snhwang/Unity-Watson-STT-Assistant-TTS-Oculus-Lipsync: 3D Chatbot using Unity, IBM Watson, and Oculus Lipsync*. [online] [GitHub](https://github.com). Available at: <<https://github.com/snhwang/Unity-Watson-STT-Assistant-TTS-Oculus-Lipsync>> [Accessed 20 April 2022].
- [8] Docs.microsoft.com. 2022. *Language Understanding (LUIS) Overview - Azure Cognitive Services*. [online] Available at: <<https://docs.microsoft.com/en-us/azure/cognitive-services/luis/what-is-luis>> [Accessed 20 April 2022].

- [9] Docs.microsoft.com. 2022. *What are intents in LUIS - Azure Cognitive Services*. [online] Available at: <<https://docs.microsoft.com/en-us/azure/cognitive-services/luis/concepts/intents>> [Accessed 20 April 2022].
- [10] Docs.microsoft.com. 2022. *Entities - Azure Cognitive Services*. [online] Available at: <<https://docs.microsoft.com/en-us/azure/cognitive-services/luis/concepts/entities>> [Accessed 20 April 2022].
- [11] Saifmohammad.com. 2022. *NRC Affect Intensity Lexicon*. [online] Available at: <<http://saifmohammad.com/WebPages/AffectIntensity.htm>> [Accessed 20 April 2022].
- [12] 2022. [online] Available at: <<https://developer.vive.com/resources/vive-sense/eye-and-facial-tracking-sdk/documentation/>> [Accessed 20 April 2022]. (Please download the SRanipal Unity SDK for Lip Documentation, inside there is the document)
- [13] 2022. [online] Available at: <<https://bera-journals.onlinelibrary.wiley.com/doi/full/10.1111/bjet.13023>> [Accessed 20 April 2022].