

Integration of Augmented Reality Vision and Speech Modules into The Cognitive Assistant System

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Sneha Iyer
Spring, 2023

Technical Project Team Members:
Keshara Weerasinghe

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Signature _____ Date _____
Sneha Iyer

Approved _____ Date _____
Homa Alemzadeh, Department of Electrical and Computer Engineering

INTRODUCTION

In the field of emergency medicine health specialists utilize a variety of information available in a situation in order to provide the best care possible to a patient in dangerous circumstances. According to Kim et al. (2021), in emergency medical care processes, the responders collect large amounts of data with different levels of importance and confidence, including the patient's past medical history, their present medical conditions, and interventions performed. While there are many standards and tools for collecting, storing, and distributing emergency medical services (EMS) data (Becknell & Simon, 2016), more attention should be given to reliably translating this variety of information into actionable knowledge for assessing and performing emergency operations. Additionally, recalling this information requires cognitive effort in a crisis situation. If some of the responsibility for aggregating and analyzing information is delegated to assistive technologies, then more cognitive effort can be channeled into improving the speed and precision of pre-hospital care.

Emergency medical technicians (EMTs) collect, filter, and interpret information from real-time sources in order to provide timely and appropriate medical interventions during emergency situations. However, doing so in a high-pressure situation causes a cognitive strain on those performing during a crisis (Lawn et al., 2020). Assistive technologies can help to lessen this pressure on first responders by improving situational awareness and facilitating appropriate decision making (Holthe et al., 2022). Additionally, the integration of machine learning technologies within cognitive assistant systems aims to improve the accuracy and effectiveness of EMTs by utilizing analytical algorithms that collect heterogeneous data streams from the incident scene, aggregate that data with publicly available data, extract valuable information, and provide applicable feedback.

In an emergency, necessary activities in the scene, prehospital, and in hospital setting must be conducted, as precisely and quickly as possible. According to Kalhori (2022), technologies such as AI might be a beneficial support to achieve this crucial aim. Already in emergency departments in some hospitals, AI has been applied for predictive modeling, patient monitoring, and day-to-day running of emergency departments. These intelligent tools support health care providers in reducing waiting times in the emergency department, decreasing errors, and increasing the efficiency of care.

This technical project focuses on working on a part of a Cognitive Assistant system that acts as an artificial intelligence (AI) agent who is assisting the responders observing and processing the data and interacting with responders during response operations to provide them with reminders, feedback and insights to improve their situational awareness and operations outcome.

RELATED WORK & BACKGROUND

Need for automation in emergency response

Currently, in an emergency response situation, which is usually initiated by a 911 call, EMS responders follow a certain procedure that produces a flow of information. According to Kim et al. (2021, p. 3), “In each call, responders are dispatched to the incident location and informed of the “Call Type,” which is the general reason for the incident. On arrival, responders interact with the patient and others to identify the “Chief Complaints,” which are the primary reasons for the EMS call. Then, the responders use the patient’s Chief Complaints, Signs and Symptoms, past medical history, history of the present illness (HPI) or injury, and current presentation to form a set of “Impressions.” From the Impressions, the responders select and

follow the appropriate “EMS Protocol Guidelines” to perform “Interventions” (including “Procedures” and “Medications”), which are a series of treatments to stabilize the patient before transporting to the hospital. Responders document this information flow from Call Type to Chief Complaints to Impressions and, finally, to Interventions, along with other information described as “Narrative” and/or “Medic Notes”, in the EMS incident reports”. It is evident that a plethora of information has to be processed very quickly and with as much accuracy as possible. Having all this information being processed and feedback displayed in real-time would be useful for efficiency and decreasing cognitive strain on the EMT.

AR and VR technology in emergency medical field

With the rapid technological advancements in recent decades, virtual reality (VR) and augmented reality (AR) technologies have been increasingly adopted to address various challenges in medical environments and emergency management. For example, studying emergency management is important, but one main obstacle in this field is that disaster scenes are difficult to construct in real life. Forcing subjects to simulate real hazards would probably not be possible, with many legal and moral challenges. Thus, VR and AR technologies have been used to construct/imitate disaster scenes for emergency research (Zhu & Li, 2021). Another example is that traditional EMS provider training relies on mannequins and verbal descriptions of a scene. However, mannequins do not provide real-time reactions to treatment such as improved breath sounds or skin signs. Thus, AR has been utilized to generate a display of such responses to provide a more realistic training exercise (Titzler & Zuniga-Hernandez, 2023). These examples show instances where AR and VR technology has been utilized for improving the emergency response industry. In this project, we seek to use AR to stream data to our Cognitive Assistant system as well as to display useful feedback to an EMT.

PROBLEM STATEMENT

The CognitiveEMS system in particular focuses on processing this information collected from the responder, which includes stages for converting speech to text, extracting medical and EMS protocol specific concepts, and modeling and execution of an EMS protocol (See Figure 1 for an architectural overview of this system). Specifically, when a first responder speaks to a patient, the CognitiveEMS system converts speech to text using a speech to text library.

Negation detection, value retrieval, and concept mapping to protocols are performed on the converted text for each sentence. Based on the information from the converted text, as well as information from sensors connected to the wearable device and the patient, the system predicts relevant EMS protocols, and provides feedback and suggestions to the first responder. The main technologies used in the system are the Google Speech API for the speech to text conversion, MetaMap software for concept extraction, and a rule engine for enabling the policies and operational decisions to be defined, tested and executed (Preum et al., 2019). The pipeline utilizes a behavior tree framework and machine learning (ML) methods to extract critical information from both audio and visual input and recommend interventions to specialists.

Early identification of necessary protocols and/or interventions by a system such as the Cognitive Assistant in an emergency situation may save patients' lives by helping EMT's to make quicker and more accurate decisions regarding the patients' transfer and care. Additionally, systems such as the Cognitive Assistant allow us to analyze the ways that AI and humans interact and collaborate. A better understanding of this will help grow potential for humans to learn from and collaborate with algorithms in an ethical manner.

Before this project, there were offline, individual ML modules for speech that were separate from the pipeline. Only audio data was collected via the microphone of the device the pipeline was running on; no vision data was collected, analyzed, or streamed. Additionally, the AR glasses and smartwatch were not being utilized for collecting and streaming data. Thus, my capstone project centered around integrating these smart devices into the current pipeline (which involves creating Android applications to collect and send data, server side scripts to receive and organize that data, and code to cohesively organize different modules in the pipeline), creating a vision module (where image/video data was collected, streamed, and analyzed), managing the various input and output streams, as well as processing and demoing the real-time results on a graphical user interface (GUI) in the Cognitive Assistant system. It also includes managing the pipeline and helping to integrate the offline modules by other students into the pipeline to analyze and interpret the data to help the assistant recommend protocols and interventions to EMS specialists.

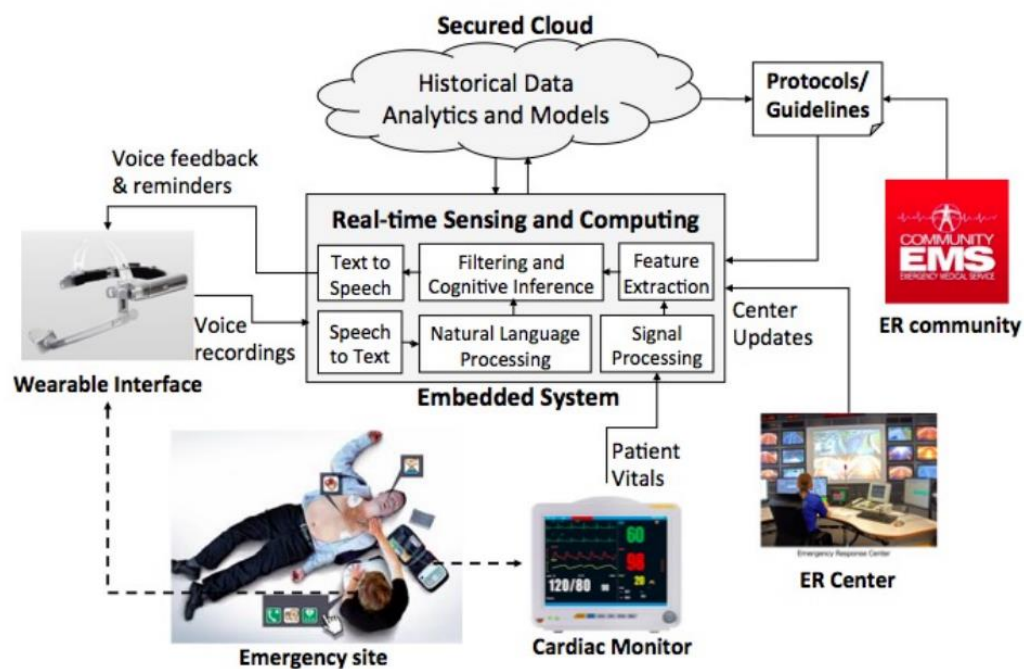


Figure 1. The architecture of the CognitiveEMS system (Preum et al., 2018)

METHODS/RESULTS

For this project, we had a VUZIX M400 Augmented Reality Smart Glasses and a Samsung Smartwatch that were the main hardware components utilized for data collection and analysis.

We aimed to collect audio, video, accelerometer, and gyroscope data. The audio data would be fed to the existing system for speech to text translation using Google Cloud Speech and concept extraction using MetaMap. Then machine learning models would run on that data to identify protocols and interventions. The video data would be displayed on a front-end GUI interface and wrist/hand positions would be annotated by Google Mediapipe. Finally, the accelerometer and gyroscope data would be utilized for determining CPR Rate.

Some of the main concepts/activities involved in streaming data are using network protocols such as Transmission Control Protocol (TCP) and User Datagram Protocol (UDP), Android app development, creating server-side scripts to receive and process data, handling data analysis, as well as returning feedback and displaying the output. When designing an application that involves streaming and networking, it is important to select the appropriate network protocol based on what type of data is being handled/transmitted and the requirements of the system concerning speed, reliability, existing infrastructure, etc. Additionally, we had to plan out how we would design the streaming so the end-to-end system worked properly.

Video Streaming/Vision Module

The first part of this project involved streaming data from a video camera on the AR glasses to the server hosting the CognitiveEMS system (a laptop). The Transmission Control Protocol (TCP) protocol was used for streaming this video data (images). Client-server streaming using the TCP protocol and Python scripts is a well known efficient way to stream data between a server and multiple clients. Specifically, TCP is a connection-oriented protocol that provides reliable and ordered transfer of data packets between two endpoints on a network. TCP is widely used for applications that require error-free transmission of data, such as web browsing, email, and file transfer. We chose to utilize this protocol because image data is very large and we need data reliably.

To actually do this, I utilized an existing client side Android app written by PhD student Lahiru Nuwan (edited by Keshara Weerasinghe) in the lab to send the data and then wrote a server side python script to receive the data. In the script, to receive data, first I used the python socket library to set up a TCP socket and bind it to a specific IP address and port number. Then I used that socket to receive data from the Android app and process it as needed. When the image was received, it was processed using Google MediaPipe, to annotate wrists/hand positions. In particular, their MediaPipe Hand Landmarker solution lets you detect the landmarks of the hands in an image. You can use this library to localize key points of the hands and render visual effects over the hands. It operates on image data with a machine learning (ML) model as static data or a continuous stream and outputs hand landmarks in image coordinates of the detected hands.

Finally, I converted the image with annotations to a pixmap format and scaled it, in order to display it in a PyQt5 graphical user interface (GUI). The conversion was necessary, since the PyQt5 library QImage module takes in a pixmap image as input. An example screenshot of typical output is shown below:



Figure 1: Hand/Wrist detection by Google Mediapipe

I also tried to use the detected hand/wrist positions to calculate CPR Rate. However, the results were very far off from the correct value. The large error was probably due to the hand/wrist annotations not being exactly correct. While Mediapipe is beneficial and chosen by us due to its real-time capabilities, its accuracy does not seem to be the best. Thus, for the demo/experiments we decided to only rely on smartwatch data to calculate CPR rate and leave image analysis for later.

Speech Module

Next, we worked on streaming audio data from the microphone on the AR glasses. The User Datagram Protocol (UDP) was used to stream audio data. UDP is a connectionless protocol that does not guarantee the delivery or order of data packets, however it is very fast. UDP is often used for applications that require quick and efficient transmission of data. We chose to use UDP for audio streaming because it reduces latency, is faster at streaming since it does not wait

for acknowledgement, and content is being consumed in real time and humans would be unlikely to notice small drops of data as much as they notice delays in motion caused by retransmission.

To actually do this, I developed an Android application to capture sound bytes from the microphone and send them via UDP to the server hosting the CognitiveEMS system. Then on the server side, we wrote a python script to receive and process that data.

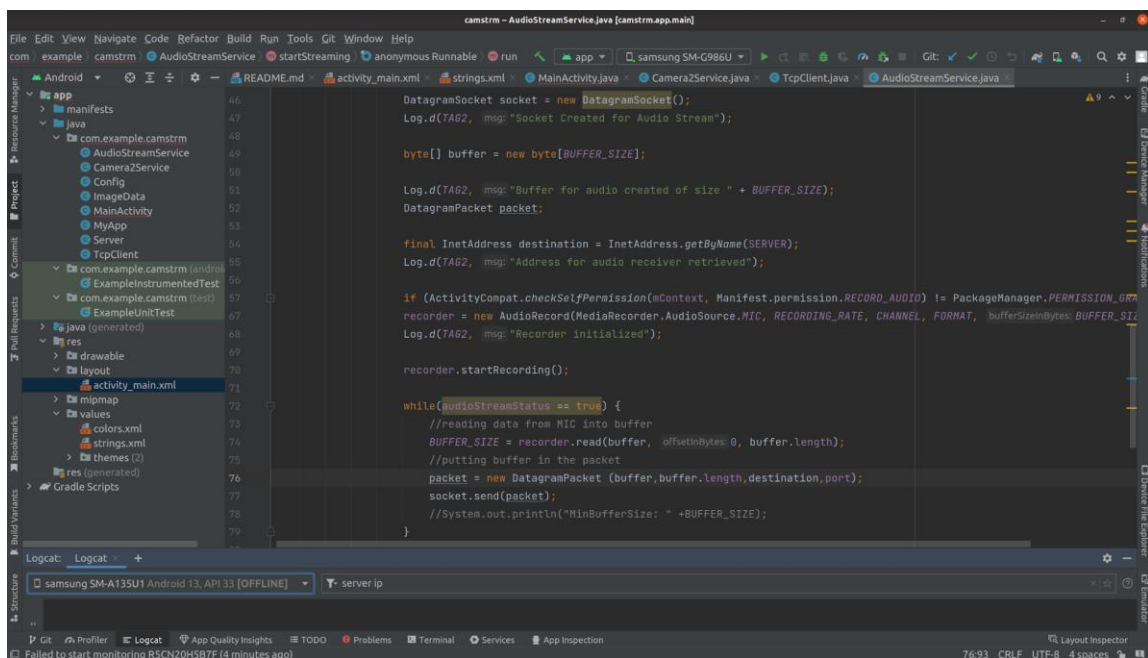
Android app:

First I set up the Android Studio development environment for a Linux operating system. The adb and Android SDK libraries were installed to enable us to do wireless streaming easily. Then I designed a user interface. In order to allow a more efficient setup for demos, I put a text input where the user can specify the server IP address to which the data can be sent wirelessly. As well as buttons to start streaming and close/shutdown the application. Next, I added permission requests. In order for the Android application to record audio data, it needs permission to access the microphone of the device. So to allow the app to access the microphone, I added permission requests to the AndroidManifest.xml file. This will prompt the user of the device to grant permission to the app to record audio.

Then to implement audio recording and encoding, I used the AudioRecord and MediaRecorder classes. This class provides methods for setting the audio source and format, as well as starting and stopping audio recording. Before recording audio, important parameters such as recording rate, number of channels, buffer size etc. need to be specified. After some research, we chose a recording rate of 16000 hertz, since that is a well known rate for human speech recording, 1 channel, and a PCM format (which is an integer representation of audio data, usually signed numbers in two's complement format). Once the audio is recorded, it needs to be

encoded into a format that can be transmitted via UDP. Some common formats for audio transmission include WAV, MP3, and AAC. We chose MP3 and also experimented with WAV.

To send the data I utilized the Android DatagramPacket, InetAddress, and Socket libraries. First, I read data from the microphone into a buffer using the AudioRecord.read() method. Next I put that buffer in a packet using DatagramPacket. Finally the packet is sent using socket.send() after specifying the destination IP address and port. This is done in a while() loop to ensure continuous streaming (until the app is closed).



```
com - AudioStreamService.java [camstrm.app.main]
File Edit View Navigate Code Refactor Build Run Tools Git Window Help
com example camstrm AudioStreamService startStreaming anonymous Runnable run app samsung SM-G986U Git
Resource Manager
app
  manifests
  java
    com.example.camstrm
      AudioStreamService
      Camera2Service
      Config
      ImageData
      MainActivity
      MyApp
      Server
      TcpClient
    com.example.camstrm (android)
      ExampleInstrumentedTest
    com.example.camstrm (test)
      ExampleUnitTest
  java (generated)
  res
    drawable
    layout
  activity_main.xml
  mipmap
  values
    colors.xml
    strings.xml
  themes (2)
  res (generated)
  Gradle Scripts
Build Variants
Logcat: Logcat
samsung SM-A135U1 Android 13, API 33 [OFFLINE] server ip
Failed to start monitoring R5CN20H5B7F (4 minutes ago)
76:93 CRLF UTF-8 4 spaces
```

```
44 DatagramSocket socket = new DatagramSocket();
45 Log.d(TAG2, msg: "Socket Created for Audio Stream");
46
47 byte[] buffer = new byte[BUFFER_SIZE];
48
49 Log.d(TAG2, msg: "Buffer for audio created of size " + BUFFER_SIZE);
50 DatagramPacket packet;
51
52 final InetAddress destination = InetAddress.getByName(SERVER);
53 Log.d(TAG2, msg: "Address for audio receiver retrieved");
54
55 if (ActivityCompat.checkSelfPermission(mContext, Manifest.permission.RECORD_AUDIO) != PackageManager.PERMISSION_GRANTED) {
56 recorder = new AudioRecord(MediaRecorder.AudioSource.MIC, RECORDING_RATE, CHANNEL, FORMAT, bufferSizeInBytes: BUFFER_SIZE);
57 Log.d(TAG2, msg: "Recorder initialized");
58 recorder.startRecording();
59
60 while(audioStreamStatus == true) {
61 //reading data from MIC into buffer
62 BUFFER_SIZE = recorder.read(buffer, offsetInBytes: 0, buffer.length);
63 //putting buffer in the packet
64 packet = new DatagramPacket(buffer, buffer.length, destination, port);
65 socket.send(packet);
66 //System.out.println("MinBufferSize: " + BUFFER_SIZE);
67 }
68 }
```

Figure 2: Code Snippet for Audio Streaming from Android Device

Server Side Script

On the server side, we used the python socket library to set up a UDP socket and bind it to a specific IP address and port number. In a separate thread, audio data was received and put into a buffer. The data is received continuously in a while loop. The data from the buffer is processed by the Google Speech Cloud Speech to Text API. Google generates requests and

responses based on the text chunks given from the audio data buffer. Then we take the responses, create a transcript, and display the results back on the GUI for the user to view. The sentences from the transcript are also sent to other buffers for concept extraction using MetaMap, intervention suggestion using a behavior tree framework, and protocol suggestion using a machine learning model developed by PhD student Xueren Ge.

Smartwatch Module

The smartwatch streaming functionality was developed by PhD student Keshara Weerasinghe. I helped with integrating it into the CognitiveEMS system. This involved creating an output display box to show CPR rate calculations on the GUI as well as editing the existing code to incorporate Keshara's code. A general overview of Keshara's code: He also utilized Android studio to develop an app to stream accelerometer and gyroscope data from a Google Pixel smartwatch to a specific server device (specified by IP address). He used the UDP protocol to send the data and also wrote a python script to receive this data as well as calculate CPR rate. The specific algorithm involves using xyz coordinates of a hand/wrist, calculating the magnitude and time between peaks, and using that information to calculate CPR rate.

Main Module

While we continued to integrate several data streams, we also had to make sure that the overall system could handle this and also organize the various modules' input and output. For the CogEMS system demos, the main file is GUI.py. In this, we set up several different threads for each data stream. Additionally, beforehand this main file contained code for several different functionalities. This caused the code to be very long and difficult to navigate through. Thus, we

worked on modularizing the code and creating different scripts for each of the main outputs being displayed (i.e. a separate script for video_streaming and display, script for smartwatch, etc.).

Data Collection

In order to show real time results, test the different streaming modules, and provide a demo of comprehensive results, I worked on adding scripts to collect data from several streams and writing that data to files/output automatically for analysis purposes. The specific data streams collected were audio, video, smartwatch, speech to text transcription, concept extraction, intervention, and protocol suggestion data. This data is collected and stored in a folder marked by a timestamp to show at what point the data was collected. Every time the pipeline system is run, the user can specify if they want to collect data and even specify certain streams. Data will be collected until the user exits or closes the system.

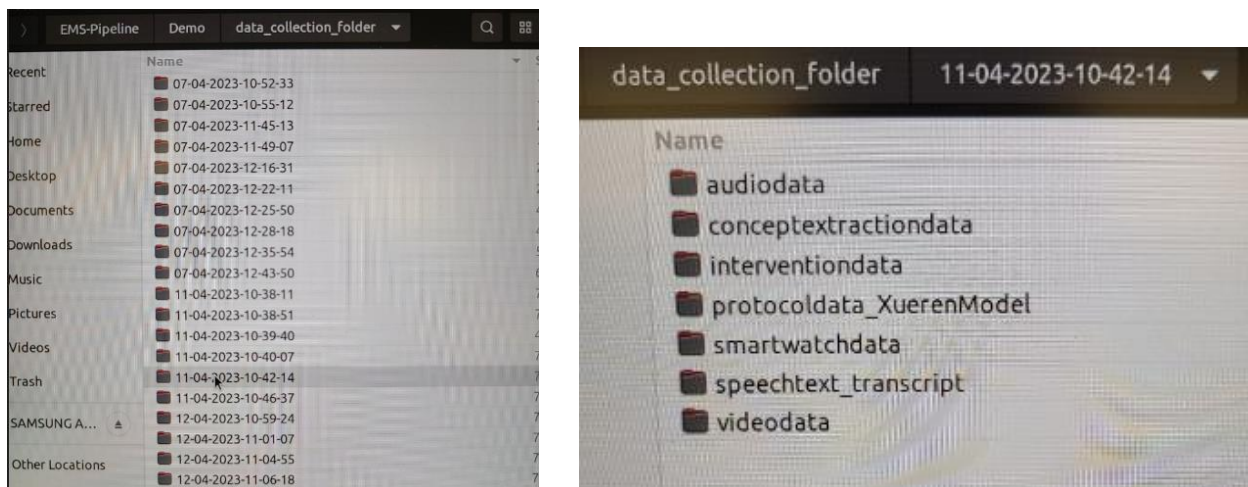
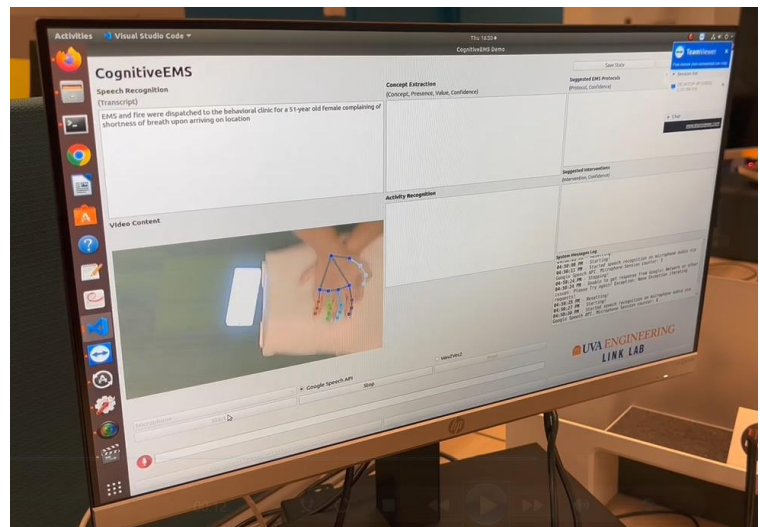
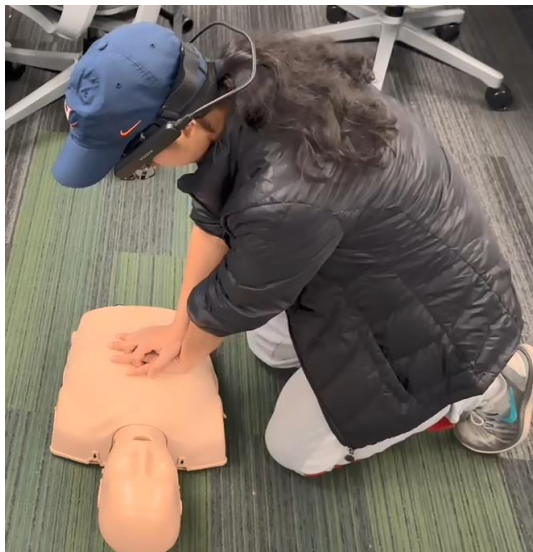


Figure 3: Side-by-side screenshots of data collection folder produced by running the system

EVALUATION

To evaluate this project, I captured screenshots of sample output/code and created demos to show progress and get feedback from other members of the group. The figures below show some example screenshots that were captured after some of the milestones achieved. For some milestones screenshots are missing and note that we also showed progress with demo videos taken that cannot be displayed in this report. Regarding getting feedback and evaluating the system outside of the lab, we also demonstrated the system at the SEAS Engineering Open House and are planning to demo the system at other conferences in the future. Additionally, we are planning to take the system to the North Garden Volunteer Fire Company to test it out and get feedback from professionals involved in emergency situations.

Creating Demo's (side-by-side of what is being done behind the scene and what shows up on the output GUI when creating a demo) :



Progress - System at the beginning:

CognitiveEMS

Speech Recognition (Transcript)

55 year old male found unconscious driver side passenger seat of his car. His wife reported that he snorted a line of heroin before I just prior to losing consciousness. Patient originally presented unresponsive. The pall, with shallow, ineffective respirations at a rate of about 5 with heart rate, was 118. The blood pressure was 205 over 119 in blood. Glucose levels, 126 is O2, saturations was nightwear, 94 % choir bag mask ventilation with mountain with the Tosh oxygen. However, after 0.25 mg of naloxone intravenously patient is now awake and breathing normally with improvement, Vital Signs and respiratory status and no longer needs supplemental oxygen.

Concept Extraction (Concept, Presence, Value, Confidence)

(Gender, True, male, 0.81)
 (Age, True, 55, 0.65)
 (Loss of consciousness, True, unconscious, 0.58)
 (Resp, True, 5, 0.77)
 (Bradypnea, True, 5, 0.77)
 (Heart rate, True, 118, 1.0)
 (Blood pressure, True, 205/11, 1.0)
 (Pulse oximetry, True, saturations, 1.0)

Suggested EMS Protocols (Protocol, Confidence)

(Medical - Respiratory Distress/Asthma/COPD/Croup/Reactive Airway, 0.35)
 (Medical - Overdose/Poisoning - Opioid, 0.34)
 (Medical - Altered Mental Status, 0.31)

Suggested Interventions (Intervention, Confidence)

(Transport, 1.0)
 (Cardiac monitor, 0.65)
 (Normal saline, 0.31)
 (Narcan, 0.26)

System Messages Log

Welcome to CognitiveEMS, a cognitive assistant system for Emergency Medical Services (EMS).

CognitiveEMS captures speech by first responders and attempts to

000_190105 Google Speech API DeepSpeech

Start Stop Reset

UVA ENGINEERING LINK LAB

Progress - Integrating Video from smartphone, with Google Mediapipe:

CognitiveEMS

Speech Recognition (Transcript)

Concept Extraction (Concept, Presence, Value, Confidence)

Suggested EMS Protocols (Protocol, Confidence)

Suggested Interventions (Intervention, Confidence)

System Messages Log

Welcome to CognitiveEMS, a cognitive assistant system for Emergency Medical Services (EMS).

CognitiveEMS captures speech by first responders and attempts to find appropriate EMS protocols and interventions. The following protocols are currently supported:

Video Content

Activity Recognition

Microphone Google Speech API Wav2Vec2

Start Stop Reset

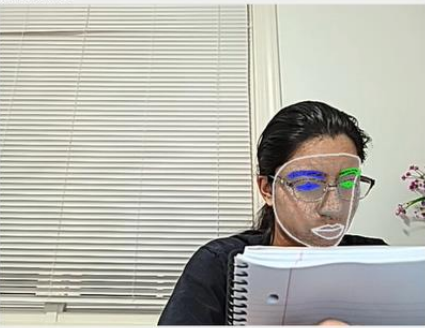
UVA ENGINEERING LINK LAB

Right AI

Progress - Video + Audio from an external smartphone microphone (not from the AR glasses).

CognitiveEMS

Speech Recognition (Transcript)
 EMS and fire were dispatched to the behavioral clinic for a 51-year old female complaining of shortness of breath upon arriving on location we found the female with the Richmond fire department the patient stated that she had not feeling well since this morning and had shortness of breath

Video Content


Microphone: Google Speech API (selected) | Wav2Vec2

Buttons: Start, Stop, Reset

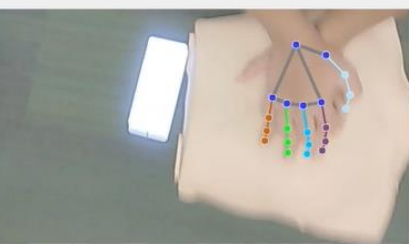
System Messages Log:
 08:11:10 PM - Ready to start speech recognition!
 08:11:20 PM - Starting!
 08:11:22 PM - Started speech recognition on microphone audio via Google Speech API. Microphone Session counter: 1

UVA ENGINEERING LINK LAB

Progress - Video from AR glasses:

CognitiveEMS

Speech Recognition (Transcript)
 EMS and fire were dispatched to the behavioral clinic for a 51-year old female complaining of shortness of breath upon arriving on location we found a female with the Richmond fire department and the patient stated that she had not been feeling well since the morning

Video Content


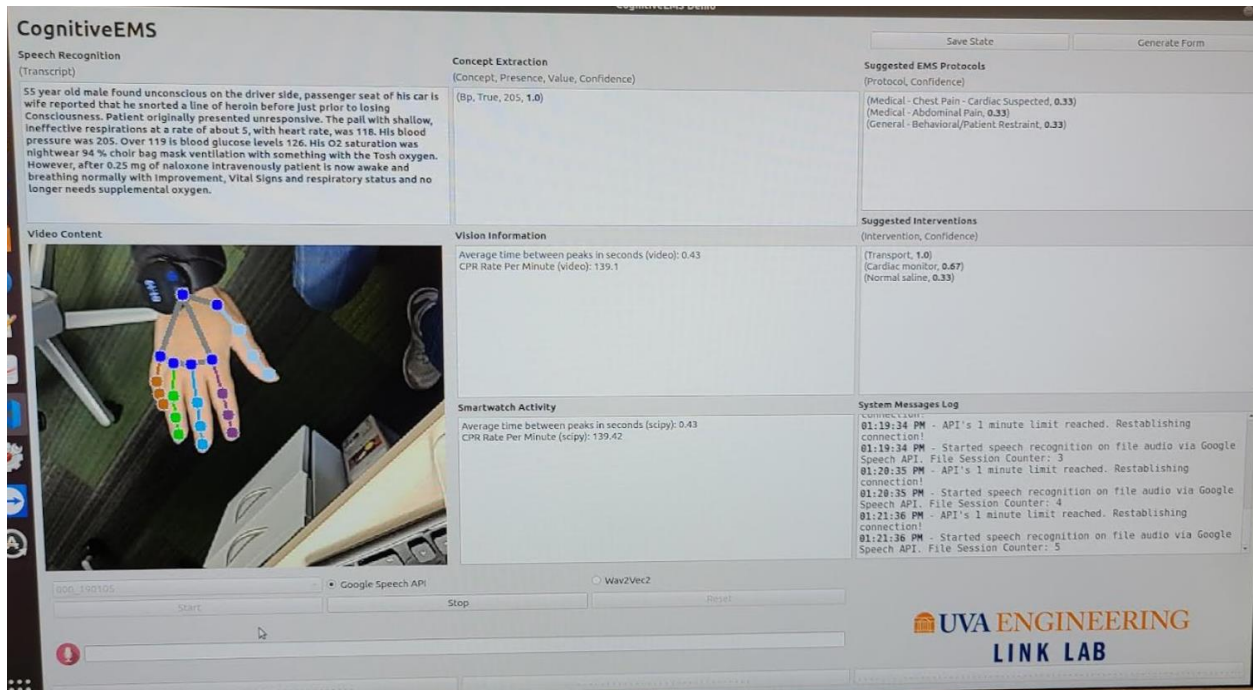
Microphone: Google Speech API (selected) | Wav2Vec2

Buttons: Start, Stop, Reset

System Messages Log:
 04:50:08 PM - Starting!
 04:50:11 PM - Started speech recognition on microphone audio via Google Speech API. Microphone Session counter: 5
 04:50:24 PM - Stopping!
 04:50:24 PM - Unable to get response from Google! Network or other issues. Please Try again! Exception: None Exception iterating requests!
 04:50:25 PM - Resetting!
 04:50:27 PM - Starting!
 04:50:30 PM - Started speech recognition on microphone audio via Google Speech API. Microphone Session counter: 6

UVA ENGINEERING LINK LAB

Progress - Video from AR glasses and Smartwatch Stream integrated; Audio is still from file stream



There are a couple of missing screenshots for milestones of integrating Audio, fixing the Concept extraction, integrating the Smartwatch module, and integrating Xueren's ML model, but the screenshot below shows everything together.

Video, Audio from Glasses and Smartwatch Stream with Concept Extraction fixed and Xueren's ML Protocol Suggestion model (Current model):

CognitiveEMS

Save State Generate Form

Speech Recognition
(Transcript)

dispatched for cardiac arrest . arrived on scene with RFD and EMS to find a 25-year-old male unresponsive and being ventilated by a home ventilator RFD arrived later . Last known at approximately midnight family found the patient unresponsive and called EMS at 02:40 family denies the patient having cold or flu-like symptoms the patient has a history of muscular dystrophy and asthma the patient has had a tracheostomy tube since 2013 unresponsive . pulseless with a GCS of 3 the patient is being ventilated on a home ventilator via a tracheostomy tube . Good breath sounds with equal chest rise and fall pupils are 4 mm and nonreactive Skin is pale . warm and dry lower extremities contracted PEG and tracheostomy tube in place

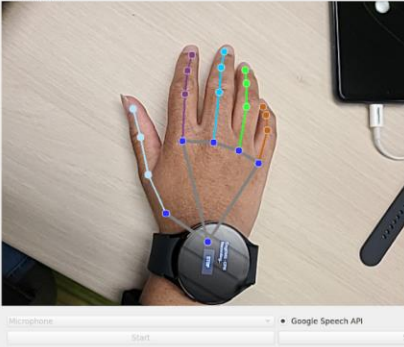
Concept Extraction
(Concept, Presence, Value, Confidence)

(Gender, True, male, 0.81)
(Age, True, 25, 0.65)
(Pulse, True, 3, 0.75)
(Bradycardia, True, 3, 0.75)
(Wheezing, True, breath sounds, 0.76)
(Pain, True, pain, 0.69)

Suggested EMS Protocols
(Protocol, Confidence)

(Kueren Model: general - cardiac arrest : 0.919859)

Video Content



Smartwatch Activity

Average time between peaks in seconds (scipy): 0.45
CPR Rate Per Minute (scipy): 132.74

Vision Information

Hand Detected! Wrist Position Identified.

Suggested Interventions
(Intervention, Confidence)

(Transport, 1.0)
(Cardiac monitor, 0.37)

System Messages Log

4. Medical - Respiratory Distress/Asthma/COPD/Croup/Reactive Airway
5. General - Behavioral/Patient Restraint
6. Medical - Overdose/Poisoning - Opioid
7. Medical - Diabetic - Hypoglycemia
8. Medical - Chest Pain - Cardiac Suspected

For more information, visit:
<https://github.com/UVA-OSU/EMS-pipeline>

12:02:15 PM - Ready to start speech recognition!
12:02:25 PM - Starting!
12:02:34 PM - Started speech recognition on microphone audio via Google Speech API.
Microphone Session counter: 1

UVA ENGINEERING
LINK LAB

CHALLENGES

In this project, an EMT responder wears “smart” devices such as augmented reality glasses and a smartwatch. Data collected via camera/microphone on the glasses and accelerometer/gyroscope on the smartwatch is recorded and streamed to a pipeline where it is received and analyzed in order to recognize the actions being made by the EMT as well as make recommendations. Integrating the AR glasses and smartwatch (i.e. streaming, receiving, and displaying data collected by these devices) posed certain challenges:

- **Android Studio:** We had to build two different Android applications – one for the glasses and one for the smartwatch. The setup of Android Studio itself proved to be challenging. Especially with a Linux based operating system, there was a lot of configuration involved. Another challenge with Android Studio is ensuring that the development version matches or satisfies the device specifications. There are several different versions of Android depending on the device being used. We ensured that we used a version that

would satisfy both the AR glasses and the smartwatch. Finally, one last challenge was actually coding with Android Studio. Several libraries were deprecated and some libraries did not even work properly. Building Android apps requires a lot of knowledge about the configuration files, which can be hard to understand for someone new at coding with Android Studio. Moreover, as described in the results section, each time a change was made the entire app had to be reinstalled on the device for testing. Overall, it sometimes became a tedious process to write app code. But it was a great learning experience and I came out of this project knowing a lot more about app development and streaming from Android apps to another device.

- Combining new code with preexisting code: One common challenge that developers face, and that we faced in this project, was combining new code with preexisting code and managing different versions. Additionally, sometimes code that was provided by other students did not work properly or did not satisfy all the requirements we wanted. We often had to edit the new code to fit into the system or edit old code and make environment changes to adapt to the new code. Additionally, we had to go through old/existing code and spend time trying to understand what was going on and how it worked. What is described in the results section is the final version that is currently in the pipeline, but to get to that point required a lot of development and adjustments made. Doing integration overall required a lot of editing, testing, and debugging. Github helped with managing version control.
- Realtime streaming: In order to present the pipeline system, we had to get the data streaming and analyze it in real time. Doing this in realtime is difficult because there is a lot of data being handled at once and each stream needs to be efficient. We used threads

to help with performance and edited parameters relating to display or rate to process data faster. Another way we tried to be efficient was we loaded or utilized pre-trained machine learning models that did not require much post processing (GoogleSpeech, Mediapipe hand detection, as well as Xueren's protocol suggestion model).

- **Size/Space/Memory limits:** Initially, I started coding for this project using a virtual machine on my personal laptop. This proved to be difficult since my virtual machine ran very slowly due to memory constraints. Afterward, I moved to using a Desktop machine in the Link Lab. I went to the lab three days per week, and was able to work there. However, even that desktop had limitations. Especially, when we worked on integrating a protocol suggestion machine learning model from PhD student Xueren Ge, the desktop's GPU proved to be not enough and the existing python configuration was causing problems. In order to integrate larger machine learning models and run the demo seamlessly and quickly, we bought an Alienware laptop with a Linux operating system. Then we set up the environments and system from scratch. This definitely helped when it came to running the demo and we were able to clean up the pipeline system (getting rid of unnecessary/problematic dependencies, properly configuring the python environment, etc). We also updated the github readme and put more comments into the code to help future students who get involved in this project.
- **Data synchronization:** This project involved several streams of data. When running the demo, we had to make sure we had a comprehensive system where all the data streams worked properly enough to collect data for analysis and display it all simultaneously on the GUI. If one stream failed we had to make sure the others were not impacted. Modularizing the code and running each stream on its own thread helped with this. The

code is currently not exactly synchronized by timestamp, we just start all the streams at the same time. More work can be done to synchronize the data in the future.

DISCUSSION/CONCLUSION

This project focused on integrating these smart devices into the current pipeline, creating a vision module, managing the various input and output streams, processing and demoing the real-time results on a GUI, as well as managing the pipeline and helping to integrate the offline modules by other students into the pipeline to analyze and interpret the data to help the assistant recommend protocols and interventions to EMS specialists. Throughout this project I was exposed to and learned more about several different concepts such as networking protocols, streaming data, developing applications using Android Studio, utilizing threads for efficiency, and integrating code in modular ways. I also learned better coding practices such as managing version control using Github, documenting code changes, setting up environments from scratch, and working with/configuring systems with various dependencies.

Based on the new changes and additions to the pipeline, as well as taking into account future plans, below is a revised figure of the architecture of the CognitiveEMS Pipeline that I created:

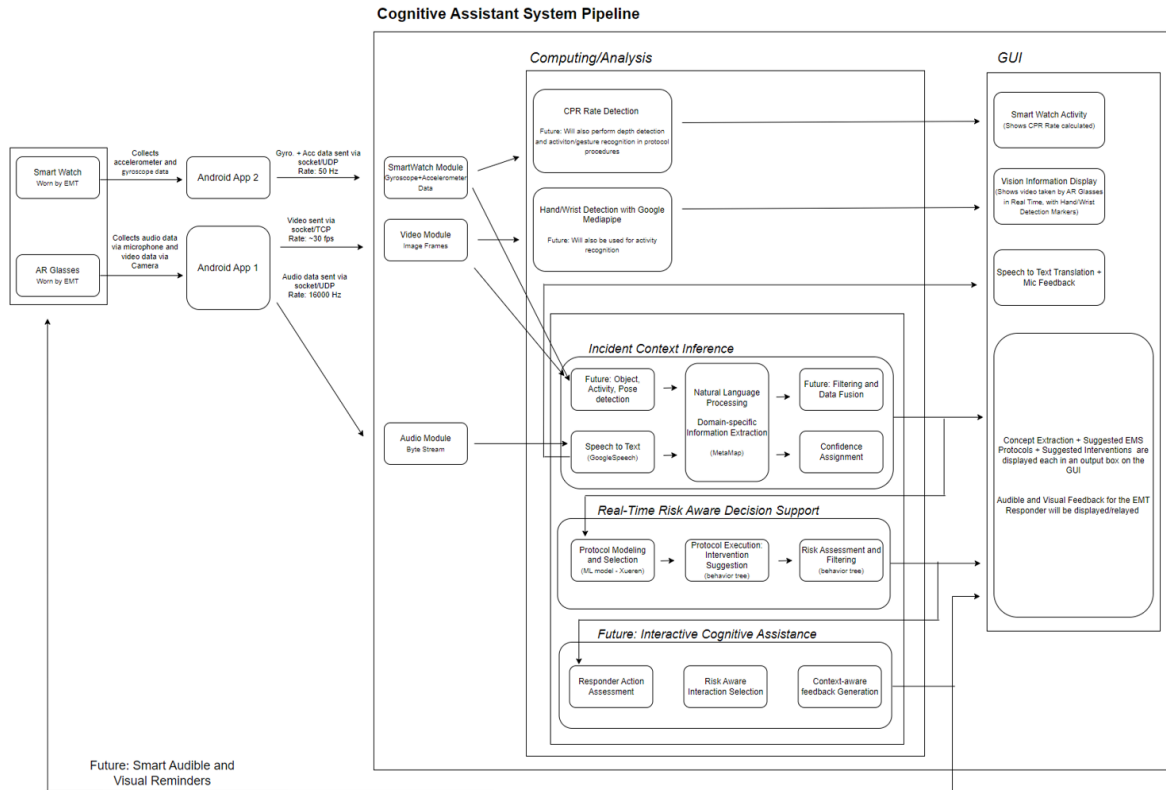


Figure 4: Revised diagram of architecture of CogEMS system

Additionally, I plan to continue working on the system. Currently, we are working on closing the loop, which involves sending the feedback from the data analysis back to the AR glasses for the EMT to see. Below is a figure showing an idea of how the EMT would view information.

Example image of displaying feedback back to the AR glasses:



Figure 5: Example of feedback display on AR glasses

ACKNOWLEDGEMENTS

- PhD Student Keshara Weerasinghe for mentoring and working with me throughout this semester and the previous semester. We worked together on many of the aspects mentioned in this report.
- Advisor Dr. Homa Alemzadeh for her help throughout my time conducting research in her lab, as well as for specific feedback on every aspect of this project.
- Other students in the group: Saahith Janapati and Xueren Ge for their feedback during group meetings and their collaborations. Also PhD student Lahiru Nuwan from Professor Stankovic's group for his feedback and insights.

REFERENCES

- Becknell, J., Simon, L. (2016). Beyond EMS data collection: Envisioning an information-driven future for Emergency Medical Services (Report No. DOT HS 812 361). Washington, DC: National Highway Traffic Safety Administration. Retrieved from https://www.ems.gov/pdf/Beyond_EMS_Data_Collection.pdf
- Committee on Guidance for Establishing Crisis Standards of Care for Use in Disaster Situations; Institute of Medicine. Crisis Standards of Care: A Systems Framework for Catastrophic Disaster Response. Washington (DC): National Academies Press (US); 2012 Mar 21. 6, Prehospital Care Emergency Medical Services (EMS) Retrieved from <https://www.ncbi.nlm.nih.gov/books/NBK201058/>
- Holthe, T., Halvorsrud, L., & Lund, A. (2022). Digital Assistive Technology to Support Everyday Living in Community-Dwelling Older Adults with Mild Cognitive Impairment and Dementia. *Clinical interventions in aging*, 17, 519–544. <https://doi.org/10.2147/CIA.S357860>
- Kalhari S. (2022). Towards the Application of Machine Learning in Emergency Informatics. *Stud Health Technol Inform*, 291, 3-16. <https://doi.org/10.3233/SHTI220003>
- Kim, S., Guo, W., Williams, R., Stankovic J. and Alemzadeh, H. (2021). Information Extraction from Patient Care Reports for Intelligent Emergency Medical Services. *IEEE/ACM Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*, 58-69, <https://doi.org/10.1109/CHASE52844.2021.00014>
- Lawn, S., Roberts, L., Willis, E., Couzner, L., Mohammadi, L., & Goble, E. (2020). The effects

of emergency medical service work on the psychological, physical, and social well-being of ambulance personnel: a systematic review of qualitative research. *BMC psychiatry*, 20(1), 348. <https://doi.org/10.1186/s12888-020-02752-4>

Preum, S.M., Shu, S., Hotaki, M., Williams, R.D., Stankovic, J.A., & Alemzadeh, H. (2019). CognitiveEMS: a cognitive assistant system for emergency medical services. *SIGBED Rev.*, 16, 51-60. <https://doi.org/10.1145/3357495.3357502>

Titzler, J., Zuniga-Hernandez, M. (2023, March 28). Using Augmented Reality Simulation for Emergency Medical Services Training. *JEMS: EMS, Emergency Medical Services - Training, Paramedic, EMT News*. <https://www.jems.com/training/using-augmented-reality-simulation-for-emergency-medical-services-training/>

Zhu, Y., & Li, N. (2020). Virtual and Augmented Reality Technologies for Emergency Management in the Built Environments: A State-of-the-Art Review. *Journal of Safety Science and Resilience*. <https://doi.org/10.1016/j.jnlssr.2020.11.004>