Voice assistant for an autonomous vehicle aimed at elderly

José Berriales-Mateos, Pablo Pardo-Decimavilla, Santiago Montiel-Marín, Miguel Antunes, Ángel Llamazares, Luis M. Bergasa.

***Abstract*—In an era where autonomous vehicles are becoming increasingly relevant, older age groups often encounter significant technological barriers. This paper outlines the design of a voice assistant tailored for an elderly audience, implemented within an autonomous vehicle in view of the facilities provided by a voice control method for elderly people. The system features a modular structure composed of four key components: audio capture using Voice Activity Detection (VAD) algorithms, audio transcription and processing through Speech To Text (STT) engines, response generation using a text generation engine based on data obtained from the vehicle through Robot Operating System (ROS) and various APIs that provide the car with external information, and finally, a Text-To-Speech (TTS) engine in charge of generating an auditory response. Additionally, a user interface (UI) has been developed to further reduce the generational digital barrier. Furthermore, the research includes a comprehensive comparative review of current text generation and speech synthesis technologies, with the proposed design validated through simulations and tested by real users.**

***Index Terms*—Voice Assistant, VAD, STT, text generation, TTS, ROS, API, autonomous vehicles, user interface.**

I. Introduction

NOWADAYS, the digital revolution has brought with it a number of challenges for elderly. Older people often face difficulties in controlling electronic devices, which can lead to neglect and disinterest in acquiring new technological skills. In many cases, this results in giving up the use of electronic devices and subsequent frustration [1]. In an age where digitisation and automation are transforming all aspects of life at an exponential rate, this disconnection with technology can lead to an isolation of the older generations from the younger ones.

This paper will focus on autonomous vehicles, benchmark of current technologies advancements. The main objective is to make the vehicle's decisions explainable to the occupants, regardless of their age, by simply asking verbal questions. In this way, we aim to bring the elderly closer to this technology, giving them a sense of control and security that would otherwise be difficult to achieve. In addition, our solution would not require any effort or training on the part of the users, which will facilitate their adoption.

On the one hand, we will ensure that vehicle occupants can receive useful information about the vehicle's status, such as speed, range, destination and upcoming decisions. On the other hand, we will enable the occupants to obtain non-vehicle related information, such as weather conditions or road status, among others.

To achieve these goals, we will use a voice recognition system that will convert questions asked verbally into text. We will analyse the questions in search of a key word that indicates the reference to launch the voice assistant. We will generate an answer based on data obtained from the car and various APIs that provide the car with external information. Finally, through a TTS engine, we will generate an audio response with the greatest possible speed and naturalness. The main contributions of this work are:

- Development of a lightweight voice assistant for resource-limited environments with real-time car communication and based on four main modules.
- Development of an operative user interface for car occupants.
- Study and comparison of the main commercial solutions for each module of the proposal in order to identify the optimal ones.
- Testing of the developed voice assistant in a simulated environment with real users obtaining subjective feedback.

II. Related Works

A. Chat Bots

A chatbot is a conversational agent that interacts with users in a given domain or on a given topic with natural language phrases [2].

Recent studies indicate that text-based user interaction is no longer attractive compared to other types of interaction such as voice [3]. Over the last few years, voice chats have evolved dramatically. Many chatbots have been deployed on Internet for the purpose of education, customer service, site guidance, or even entertainment functions. Existing famous chatbot systems include Gemini, ChatGPT, Cleverbot or ALICE [4].

The operation of these systems is mainly based on four stages. The first stage involves voice collection using voice activity detection strategies. The second stage is responsible for transcribing the voice to text. The third stage generates a response in text to the user's answer. Finally, the fourth stage converts the answer text to audio. Next, a review of the state of the art for each of these four stages will be conducted.

This work has been supported by the Spanish PID2021-126623OB-I00 project, funded by MICIN/AEI and FEDER, TED2021-130131A-I00, PDC2022-133470-I00 projects from MICIN/AEI and the European Union NextGenerationEU/PRTR, PLEC2023-010343 project (INARTRANS 4.0) from MCIN/AEI/10.13039/501100011033, and ELLIS Unit Madrid funded by Autonomous Community of Madrid.

P. Pardo-Decimavilla, S. Montiel-Marín, M. Antunes, Ángel Llamazares and L.M. Bergasa are with the Electronics Departament, University of Alcalá (UAH), Spain.{pablo.pardod, santiago.montiel, miguel.antunes, angel.llamazares, luism.bergasa} @uah.es

B. Voice Activity Detection (VAD)

A VAD is a key component in the field of speech signal processing. Its primary function is to discern the presence of speech activity within an auditory signal by classifying time segments as speech or silence.

A large number of VAD algorithms have been proposed over the years, among them, we can highlight those based on periodicity [5] or those based on entropy [6], as they are quite simple and efficient in terms of time. As a drawback, these algorithms have to be adjusted for particular situations, resulting in systems that do not adapt well to day-to-day variable noise conditions.

Several systems based on complex statistical approaches have been proposed to solve this problem: VAD using long-term speech information [7], VAD based on a statistical amplitude distribution [16] and low-variance spectrum estimation and adaptive threshold VAD [8], among others. The problem found in these algorithms based on statistical models is that they need to estimate the background noise and the configuration of some parameters, making them a more flexible solution than those mentioned at the beginning, but far from what we would consider an ideal VAD.

In recent years, studies have been published with the aim of solving the problem described in the previous paragraph from the machine learning point of view [9].

C. Speak To Text (STT)

STTs are a key part of today's technological ecosystem, enabling the automated conversion of acoustic signals into written text with increasing accuracy. This process involves a complex amalgam of signal processing, language modelling and machine learning algorithms, which have been the subject of extensive study.

Based on this, several approaches to language comprehension have been given:

- **Template-based**: In this approach a collection of speech patterns stored in a word dictionary [20].
- **Rules-based**: Based on linguistic, phonetic and spectrographic information [10].
- **Based on neural networks**: Superior performance in situations where training data is available and vocabulary is limited is.
- **Based on Dynamic Time Warping (DTW)**: Applied to speech transcription, it is an algorithm that allows finding the optimal match between two given sequences by time warping them to match [11].
- **Based on statistical modelling**: In this approach, variations in speech are modelled statistically (e.g. based on HMM) using training methods [14].

  The main current commercial STT solutions are divided into two categories: local and online. Locally, Whisper [28] (from OpenAI) is a neural network-based solution that provides high-quality transcription and offers five different models. Vosk [26] is an engine designed for embedded systems, while Kaldi is a toolkit for developing such engines. In the cloud, there are solutions from main cloud providers, Deepgram [27], a company focused on developing these technologies, and OpenAI also offers an API for Whisper.

D. Text Generation

The main objective of text generation is to automatically learn an input-output mapping from the data to build a comprehensive solution with minimal human intervention. The first approaches usually make use of statistical language models to model the probability of the words [12].

Problems related to this approach led to the emergence of deep learning techniques, which have dominated the main text generation methods with great success. Despite their success, they suffer from major performance problems, so in recent years foundation pre-trained linguistic models as BERT or GPT [13] have become more and more important. The basic idea of these models is based on pre-training the models in unsupervised environments and then tuning them to fit a particular task.

A key point of these systems is their ability to provide real-time information about the vehicle's status or external information of interest to the occupants. This topic will be addressed in this project using the Robot Operating System (ROS) and various APIs.

E. Text to Speak (TTS)

Speech synthesis consists of converting natural language text into sound by faithfully imitating a human voice. Main methods for speech synthesis are:

- **Concatenative synthesis**: Based on the general combination of pre-recorded audio segments, achieving high quality voices [21].
- **Formant synthesis**: Phonation, frequency and noise are varied over time by this model to generate an artificial voice, resulting in a robotic voice [22].
- **HMM Synthesis**: the frequency spectrum, the fundamental frequency and the speech duration are modelled simultaneously based on HMM [14].
- **Deep learning-based synthesis**: By combining natural language processing techniques and deep neural networks, high quality audios are obtained [15].

Main modern architectures of TTS engines are:

- **WaveNet**: This architecture is based on dilated convolutional layers, which allow capturing long-term temporal relationships in the audio signal by gradually expanding the receptive field without exponentially increasing the number of parameters. It is an autoregressive model, meaning it predicts the next audio sample conditioned on the previously generated samples, allowing it to capture long-term dependencies and generate high-fidelity audio [23].
- **Tacotron**: Consists of two main components: an encoder that converts the input text into a latent context representation and a decoder that generates the audio waveform based on this representation [24].
- **Deep Voice**: This architecture consists of two main stages: text encoding and audio synthesis. In the text

encoding stage, a recurrent neural network is used to convert the input text into a latent representation that encapsulates the semantic and contextual information of the text. Subsequently, this representation is fed into a generative network, such as a convolutional or recurrent network, in the audio synthesis stage, with the aim of generating the audio waveform corresponding to the given text [25].

III. Method

A. Problem Formulation

Our voice assistant is part of the HMI(Human Machine Interface) module of an Autonomous Driving (AD) stack architecture developed by the Robesafe research group [19]

Majority of its communications are through ROS, and also has a 4G module in the vehicle providing it with Internet connectivity.

The goal is to develop a car-user voice communication system capable of reading and writing to the relevant ROS nodes, and receiving information from external APIs to provide the answers to the user's questions with as much quality information as possible in the shortest possible time, imitating a natural conversation and using state-of-the-art techniques.

B. Our Approach

The aim of the system is to allow vehicle occupants to make a spoken request to the car at any time and for it to respond appropriately by voice. To make this possible we propose the architecture depicted in Fig.1.

The audio inside the vehicle will be continuously analysed with a machine learning VAD and the audio will be collected only when voice is detected. After the audio is extracted, it will be processed to obtain its associated transcript and validate the existence of a keyword that triggers the rest of the system.

Once a predefined keyword has been detected, the assistant will extract information from the car's sensors, such as speed, steering wheel angle or the status of the decision-making module via ROS (the architecture's main communications protocol). In addition, through various APIs (TomTom, Open Street Map and Open Weather), non-vehicle information such as nearby incidents, road information or weather status is obtained.

All this information is placed at the disposal of a powerful text generation engine that will interpret and generate a natural, coherent and relevant response to the user's question with the information provided.

The initial audio input will be processed by an STT engine to be able to work with it in text format, and the text output from the text generation model will be fed into a TTS model that converts the text into an audio file that can be played by the vehicle's multimedia system.

Given that this is a real-time system that forms part of a vehicle, in the study of possible solutions for each of the modules presented in the next section, the speed, the resources necessary for their operation and the quality/naturalness they offer will be highly valued, these being key factors in providing the user with the best possible experience.

C. prompt Engineering

Prompt engineering is a challenging yet crucial task for optimizing the performance of large language models on customized tasks [16]. The quality of prompts is crucial to maximise the potential of these models, influencing the accuracy, relevance, efficiency and safety of the generated responses.

In our context of a voice assistant for an autonomous vehicle, Prompt Engineering is essential to properly interpret user responses, sensor data and APIs. By using well-designed prompts, the system can better understand the user's instructions, process information from the vehicle's sensors and access external data through APIs to make informed decisions.

In the proposed architecture, this type of practice is of great importance due to the large amount of information that will be fed to the text generation engine. As mentioned above and as will be discussed in later sections, the selected text generation engine will be as close to the state of the art as possible and prompt engineering will be a key factor in the tuning of the system so that it only displays the desired responses. The next section will outline how have been implemented these techniques in a combined way to the selected text generation engine.

IV. Experiments

This section will detail the various experiments to which the different alternatives available for the modules of the architecture detailed above have been subjected.

A. Speak To Text (STT) Engine

For speech capture from inside the vehicle cabin, the parallel execution of a microphone and a VAD has been developed. This means that the speech transcription engine does not have to process audios periodically but will only process audios when speech is detected and for the duration of the speech, drastically reducing the load and allowing us to opt for less powerful engines that do not have the ability to detect speech in the audio fragment.

TABLE I
AVERAGE RESPONSE TIMES IN SECONDS OBTAINED TESTING SELECTED SPEECH TRANSCRIPTION ENGINES AS DESCRIBED.

Model	**Variant**	**Results** Clean audios	**Results** Noisy audios
Whisper	Tiny	0.257	0.255
	Base	0.291	0.299
	Small	0.310	0.293
	Medium	0.515	0.520
	Large	0.728	0.714
Google Cloud STT	API	1,199	1,313
Deepgram	API	1,554	1,444
Vosk	US Model	0.603	0,749

On this basis, VOSK, Whisper (in all its variants), Google Cloud STT and Deepgram APIs have been tested for this purpose. They have been subjected to 100 samples of clean

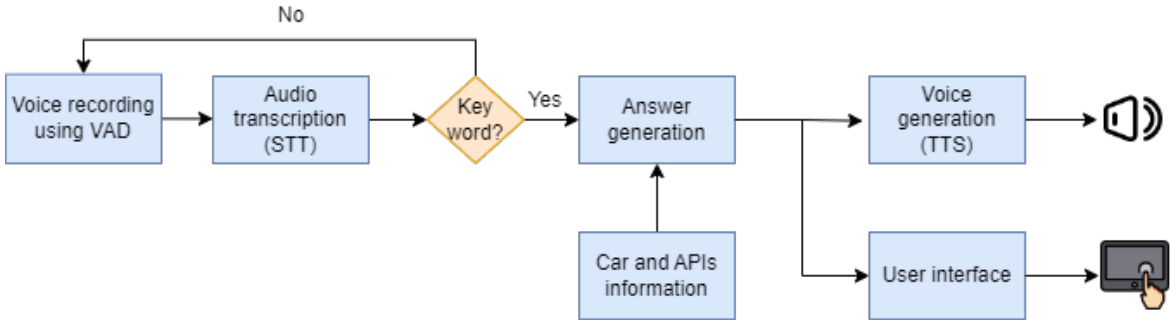

Fig. 1. Proposed architecture

audio and another 100 samples of the same audio but with white noise simulating the possible noise in the passenger compartment of the moving vehicle. In both cases, the response time and the resources required for each of them have been measured.

The local engines (Vosk and Whisper) have been tested on two different setups, in order to see the variation of the response time depending on the available computational power. A first setup consisting of an AMD Ryzen 7 5800X3D CPU, 32GB of RAM and a Nvidia RTX 4070 GPU with 12GB of VRAM and a second, more basic setup with an Intel I7-10500U CPU, 32GB of RAM and a MX330 GPU with 2GB of VRAM.

In local tests, it has been observed that Whisper's resource consumption is very high in terms of VRAM, to the point that, in the larger models, running a process in parallel is unfeasible. It is impossible to run Whisper medium and large on the setup with the MX330. Meanwhile, VOSK, being oriented towards embedded devices, offers slightly worse performance at the cost of minimal resource consumption exclusively on the CPU. Given that the audio transcription process will be repeated multiple times during the execution of the program, it has been considered that it would be optimal to incorporate an engine with a low response time and low resource consumption. In addition, the local execution of the programme has been positively valued in order to provide answers and warn the user even when there is no internet connection. With the results obtained in Table I, it has been decided to implement VOSK as it offers a very high transcription quality at a good response speed and with much less resources than Whisper [29].

B. Text generation

In order to be able to generate complex answers to any type of user question, the implementation of large LLMs is being studied. To find the optimal implementation, two approaches have been studied: locally or through an API of a cloud service.

For the local models, Falcon 7b, Wizard 7B, 13B snoozy and mistral 7B have been tested. For the cloud-hosted models, all the models supplied by OpenAI have been tested. Unfortunately, they could not be compared with Gemini as it is not yet available in the region where this study was carried out.

In a similar way as described in the previous section (200 continuous questions), each of the solutions proposed above has been tested to get a clear view of the resource consumption of each model and the average response time.

Table II shows the results obtained with the setup consisting of the Ryzen 7 and the RTX 4070. It should be noted that when making a large number of requests to the free OpenAI API, this imposes a delay between requests due to a maximum number of tokens available; to overcome this, a premium tier of the API has been used to obtain the real result without restrictions.

TABLE II
AVERAGE RESPONSE TIMES IN SECONDS OBTAINED TESTING SELECTED TEXT GENERATION ENGINES AS DESCRIBED.
L: LOCAL **C**: CLOUD API

Model	Type	Results
Gpt 4o	**C**	2.46
Gpt 4 Turbo Preview	**C**	11.60
Gpt 4 Turbo	**C**	8.32
Gpt 3.5 Tubo	**C**	2.17
Mistral7b	**L**	13.89
Falcon 7b	**L**	16.97
Snoozy 13b	**L**	25.04
Wizard 7b	**L**	29.46

As can be seen from the results, the response time of the local systems is too slow compared to the APIs, showing a significantly lower response maturity. Based on this, an API has been implemented and to overcome the possible problem of lost connectivity and not leaving the user without feedback, a series of predefined messages will be created to warn the user of the status of the vehicle while trying to recover the connection.

The selected OpenAI API model is GPT-4o for its incredible message understanding and naturalness of response despite being slightly slower than its predecessor GPT-3.5-turbo.

When using an engine of considerable power, the prompt engeening becomes very important. A prompt has been designed that will divide the input of our text generation system into three different stages and configure two types of outputs enabling the engine to execute pre-programmed functions.

Input:

- **CAR_DATA**: Contains in json format all the information related to the vehicle sensors.

- **EXT_DATA**: Contains information external to the vehicle, obtained through APIs.
- **USR_QST**: Contains the transcription of the request made by the user.

Output:

- **ANS**: The answer generated by the text generation engine is received.
- **./**: Similar to a bash command, a pre-programmed function execution pattern is established. It makes sense in the interaction with the vehicle in the user-to-car sense (e.g.: ./start_route).

C. Text To Speech (TTS) Engine

Two solutions have been considered for voice synthesis, the inclusion of a local TTS engine or the use of an API in the cloud. Based on the studies carried out in the previous sections, the solutions that are really implementable in the vehicle must necessarily be very light in terms of requirements for their execution. The only possible local solution resides in Festival [17]. In case of APIs, any of them can work perfectly well over a 3G/4G/5G connection due to their low resource usage. However, these implementations mean that in situations where the connection is unreliable the system can be severely affected.

Based on the decision taken with respect to the text generation engine, in times of disconnection of this engine, the inputs will not be received, so no response will be generated, making there is no text to synthesise. To solve this problem, it has been proposed to use WaveNet hosted in the Google Cloud to synthesise the voice, providing the assistant with great realism. In disconnection situations, as the STT runs on the system's hardware, it is able to detect that the user has asked a question, and a standard response indicating the lack of connection will be played by Festival without cutting off communication with the user.

Festival offers a far from ideal experience, which is why it will only be used in emergency situations when the connection is lost.

D. User interface

In addition, a user interface inspired by instant messaging applications has been developed to give the user the ability to have a conversation history at a glance.

As shown in Fig. 2, the assistant is able to acquire values from both the vehicle's sensors and external sources through APIs, responding to the user in a coherent and relevant manner.

E. Data recollection and system implementation

As described before, the communication of the vehicle modules is done through ROS. For the communication our system will have a process in charge of subscribing to the different ROS topics and refreshing some shared memory slots which will be used by a function that generates the json described above.

The sending of messages, in a similar way, is done at the request of the speech generation engine. When it generates the specific command, a code function is executed that publishes the data specified by the text generation engine. In this way, real-time communication with all sensors in the vehicle is achieved.

Fig. 2. User interface

The implementation of the AD architecture developed by the RobeSafe group is based on containers and virtualised networks. Each module runs in a separate container interconnected by a virtualised network. This allows the voice assistant as such to run in a separate container connected to the virtualised ROS network, making it easy to activate and deactivate without affecting the rest of the system.

The container in turn will have a multiprocess structure, one process will be dedicated to the collection of data from the car sensors at (10Hz) with the rospy library. Given the high frequency at which the sensor variables will be refreshed, it has been decided to raise the voice assistant code in a different process. The communication between both is done through shared memory slots generated with the multiprocessing library. Furthermore, the architecture (see Fig. 3) has been designed in this way to be able to perform as many tasks in parallel as possible in order to optimise the response time.

The process in charge of the voice assistant is divided into several threads as it is depicted in Fig. 3. The first thread is in charge of recording and storing the voice data. This thread stores in a temporary folder the audio results that are processed by the STT engine in order of arrival. A second thread contains the STT engine, which remains loaded but always inactive as long as a semaphore does not enable it. This second semaphore is enabled by a third thread that thanks to the watchdog library monitors the temporary folder activating the semaphore when there are files to be processed. In this way, no resources are wasted maintaining a constant polling on the folder and the maximum possible speed is practically maintained. This second thread, once it has correctly processed the audio, generates a transcript that is directly forwarded to the text generator engine. The text generator engine, although not defined within a thread, has been defined within an environment of the decorator timeout library limited to 4 seconds. In this way it executes the function inside a thread that when exceeding 4 seconds generates a timeout exception. Since in Python the main thread is the only one that can receive signals, this second thread has been

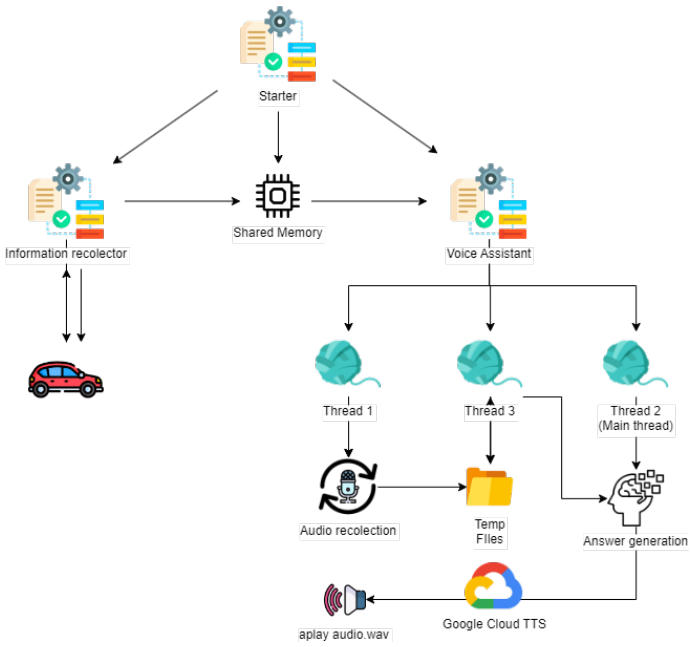

Fig. 3. Implementation diagram.

established as the main thread. Finally, when the response from the text generation engine has been obtained, a request is made to the Google Cloud TTS API, which returns the audio with the synthesized response. To play the audio, a subprocess is started that executes the command aplay audio.wav so that the audio can be played through ALSA, a framework of drivers and software utilities for managing audio on Linux systems, and the user can continue listening to it at the same time.

F. Simulation results

The CARLA (Car Learning to Act) simulator [18] is an open source simulation platform for research and development in the field of autonomous vehicles.

The complete AD architecture developed by the RobeSafe group is perfectly integrable with this simulator and with the Rviz software to show results. To test the developed voice assistant, it has been incorporated into the simulator as a module of the architecture. Through this, the destination of the route to be taken by the vehicle can be established. Figure 5 shows a view of Rviz, where the ”way points” (blue points) generated by the planner between the vehicle’s current location and the final position provided by the voice assistant are displayed.

In order to test the voice assistant in a holistic way, 10 users, aged between 50 and 80 years, had the opportunity to use it by establishing a route and asking questions while the vehicle moved autonomously in the simulator. After the test, each of them answered a questionnaire to evaluate the subjective performance of the tool. Results are shown in Table III.

The results are really good, obtaining an overall satisfaction value of 7.8 and some really high scores when it comes to assessing the naturalness and ease of adaptability. The results are also remarkably good when it comes to evaluating the quality of the feedback information.

TABLE III
AVERAGE SCORES FROM USER FEEDBACK ON VOICE ASSISTANT PERFORMANCE.

Question	Average Score
How would you rate the voice assistant’s comprehension?	6.8
How would you rate the answers to the formulated questions?	8.0
In terms of speed, how much does it resemble a conversation between two people?	4.2
How natural does the voice of the assistant sound?	9.4
If you have tried to set a destination, has the destination been set correctly?	Yes (100%)
How complex was your process of adapting to the voice assistant?	2.7
Would you implement a voice assistant like this in your own vehicle?	Yes (90%)
How do you rate the quality of the non-vehicle-related information provided by the voice assistant?	7.25
How do you rate the quality of the vehicle-related information provided by the voice assistant?	7.6
Do you consider a voice assistant necessary in autonomous vehicles?	Yes (100%)
Overall satisfaction with the voice assistant	7.8

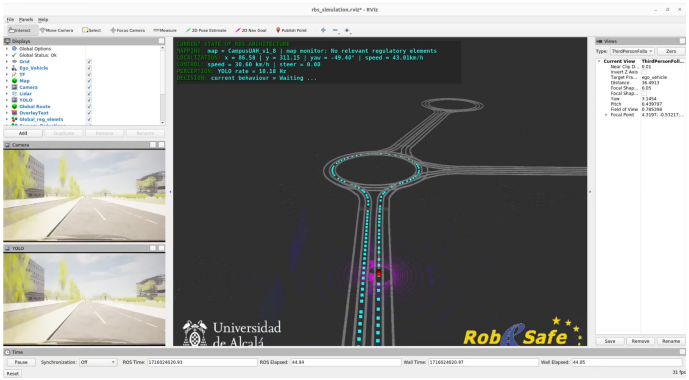

Fig. 4. Rviz Software

The lowest result is obtained when the user is asked to compare the speed of the assistant with that of a normal conversation, making it clear that this is an area where further progress can be made.

V. CONCLUSION

A modular architecture of a voice assistant has been developed for interaction with the occupants of an autonomous vehicle, targeting an elderly audience in order to reduce the acceptance barrier of this technology. The interaction method has been established as purely oral, making it close to a natural conversation between humans making it significantly easier to interact with the vehicle for the target audience.

The developed system consists of 4 distinct stages that have been carefully designed to provide the best user experience.

Initially, all the audio produced inside the vehicle is processed. Specifically, thanks to a VAD algorithm, the audio coming from the voice of the vehicle occupants can be discerned from other sounds. The next stage carries out the transcription of the audio and its processing. As this stage is performed for each audio sample that is recorded, it has been chosen to implement a fast and lightweight solution that runs locally to save processing time. The chosen solution is the STT VOSK engine for its speed, accuracy and low requirements. Once the audio transcript is obtained, it is analysed for a keyword indicating that the user is conversing with the vehicle's HMI.

After processing the audio, a stage of data collection and generation of a response to the user input is launched. Through ROS the system is able to communicate with all the vehicle's sensors and by making use of certain APIs it is able to obtain information external to the vehicle through the Internet via a mobile phone connection. The generation of the response is handled by a text generation engine. After an in-depth study of the available alternatives, it was decided to use the GPT 4o model of the OpenAI API due to its great naturalness, maturity of the responses and speed. This model, when properly configured, allows the generation of very precise text responses and even interacts with the code by executing previously developed functions that allow actions such as the establishment of routes, etc., to be executed.

Finally, in the same stage, the text response is processed in a TTS engine generating an audio reproducible through the vehicle's multimedia system. For this module, several alternatives have been considered and several algorithms have been studied, but taking advantage of the fact that our text generation engine is in the cloud, we have finally opted for a TTS based on wave-net that offers a voice indistinguishable from a real one through an API in Google Cloud.

The proposed assistant is a robust, accurate and fast system that interacts with the user orally in times very close to those of a normal conversation. In addition, it allows access to vehicle information and even external information of interest to vehicle occupants.

It has been configured so that routes can be established completely orally. In addition, speed and resource consumption have been prioritised to make it lightweight and provide the best possible experience. In this way, the aim is to improve the acceptance of the technology by the general public and the elderly in particular.

The system has been validated in a simulator while the vehicle drives autonomously in an urban environment. The opinion of several users on the functionality of the tool has been collected by means of a questionnaire and the results obtained have been provided, achieving a 7.8 out of 10 in the degree of satisfaction with the voice assistant in general.

REFERENCES

[1] D. Segarra *Las nuevas tecnologías, ¿nuevo factor de exclusión social?* Cuenta y razón del pensamiento actual, ISSN 0211-1381, Nº 135, 2004, pags. 33–37

[2] Huang, Jizhou and Zhou, Ming and Yang, Dan *"Extracting Chatbot Knowledge from Online Discussion Forums."* Ijcai, 2007, pags. 423–428

[3] A. A. Kurniawan, W. E. Fachri, A. Elevanita, Suryadi and R. D. Agushinta, *"Design of chatbot with 3D avatar, voice interface, and facial expression,"* 2015 International Conference on Science in Information Technology (ICSITech), Yogyakarta, Indonesia, 2015, pp. 326-330, doi: 10.1109/ICSITech.2015.7407826. ,

[4] Wanda Dann, Stephen Cooper, and Donald Slater. 2013. *"Alice 3.1 (abstract only)"*. In Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE '13). Association for Computing Machinery, New York, NY, USA, 757.

[5] Tucker, R. (1992). *"Voice activity detection using a periodicity measure."* IEE Proceedings I (Communications, Speech and Vision), 139(4), 377-380.

[6] Renevey, P., & Drygajlo, A. (2001, September). *"Entropy based voice activity detection in very noisy conditions."* In INTERSPEECH (pp. 1887-1890).

[7] Ramırez, J., Segura, J. C., Benıtez, C., De La Torre, A., & Rubio, A. (2004). *"Efficient voice activity detection algorithms using long-term speech information."* Speech communication, 42(3-4), 271-287.

[8] Tanyer, S. G., & Ozer, H. (2000). *"Voice activity detection in nonstationary noise"*. IEEE Transactions on speech and audio processing, 8(4), 478-482.

[9] Zhang, X. L., & Wu, J. (2012). *"Deep belief networks based voice activity detection."* IEEE Transactions on Audio, Speech, and Language Processing, 21(4), 697-710.

[10] Al-Issa, Suhad & Alshboul, Mohammad & Al-Ayyoub, Mahmoud. (2023). *"Enhanced Neural Speech Recognizer for Quranic Recitations."* 62-66. 10.1109/MCNA59361.2023.10185668.

[11] Gaikwad, Santosh & Bharti, W.Gawali & Yannawar, Pravin. (2010). *"A Review on Speech Recognition Technique."* International Journal of Computer Applications. 10. 10.5120/1462-1976.

[12] Brown, P. F., Cocke, J., Della Pietra, S. A., Della Pietra, V. J., Jelinek, F., Lafferty, J., ... & Roossin, P. S. (1990). *"A statistical approach to machine translation."* Computational linguistics, 16(2), 79-85.

[13] Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., & Huang, X. (2020). *"Pre-trained models for natural language processing: A survey."* Science China technological sciences, 63(10), 1872-1897.

[14] Kireev, N., & Ilyushin, E. (2020). *"Review of existing text-to-speech algorithms."* International Journal of Open Information Technologies, 8(7), 84-90.

[15] Kumar, Y., Koul, A., & Singh, C. (2023). *"A deep learning approaches in text-to-speech system: a systematic review and recent research perspective."* Multimedia Tools and Applications, 82(10), 15171-15197.

[16] Ye, Q., Axmed, M., Pryzant, R., & Khani, F. (2023). *"Prompt engineering a prompt engineer."* arXiv preprint arXiv:2311.05661.

[17] *Festival Official Website:* https://www.cstr.ed.ac.uk/projects/festival/

[18] *CARLA Official Website:* https://carla.org//

[19] Gómez-Huélamo, Carlos and Diaz-Diaz, Alejandro and Araluce, Javier and Ortiz, Miguel E and Gutiérrez, Rodrigo and Arango, Felipe and Llamazares, Ángel and Bergasa, Luis M. (2022) *"How to build and validate a safe and reliable Autonomous Driving stack? A ROS based software modular architecture baseline."* 2022 IEEE Intelligent Vehicles Symposium (IV), 1282–1289.

[20] , Naima Zerari , Bilal Yousfi and Samir Abdelhamid. (2016) *"Automatic Speech Recognition: A Review "*. 63-68.

[21] Khan, R. A., & Chitode, J. S. (2016). "Concatenative speech synthesis: A review." International Journal of Computer Applications, 136(3), 1-6.

[22] Lukose, S., & Upadhya, S. S. (2017, January). Text to speech synthesizer-formant synthesis. In 2017 International Conference on Nascent Technologies in Engineering (ICNTE) (pp. 1-4). IEEE.

[23] A. van den Oord, Y. Li, I. Babuschkin et al., Parallel WaveNet: Fast High-Fidelity Speech Synthesis, 2017. arXiv: 1711.10433

[24] Y. Wang, R. Skerry-Ryan, D. Stanton et al., Tacotron: Towards End-to-End Speech Synthesis, 2017. arXiv: 1703.10135

[25] W. Ping, K. Peng, A. Gibiansky et al., Deep Voice 3: Scaling Text-to-Speech with Convolutional Sequence Learning, 2018.

[26] Vosk official Website, https://alphacephei.com/vosk/.

[27] Deepgram official website, https://deepgram.com/.

[28] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey e I. Sutskever, Robust Speech Recognition via Large-Scale Weak Supervision, 2022.

[29] Rostislav Kolobov and Olga Okhapkina and Olga Omelchishina and Andrey Platunov. "MediaSpeech: Multilanguage ASR Benchmark and Dataset". 2021