Research Article

AI-Enhanced Automotive Navigation: Enriching Driving Experiences through Intelligent Contextual Information Delivery

Huseyin KARACALI[1*], Nevzat DONUM[2*], Efecan CEBEL[3*]

[1] TTTechAuto Turkey, Software Architect, Orcid ID: https://orcid.org/0000-0002-1433-4285, E-mail: huseyin.karacali@tttech-auto.com
[2] TTTechAuto Turkey, Embedded Software Engineer, Orcid ID: https://orcid.org/0000-0002-8293-8267, E-mail: nevzat.donum@tttech-auto.com
[3] TTTechAuto Turkey, Embedded Software Engineer, Orcid ID: https://orcid.org/0000-0002-2027-0257, E-mail: efecan.cebel@tttech-auto.com
[*] Correspondence: efecan.cebel@tttech-auto.com

(First received January 21, 2024 and in final form May 20, 2024)

Reference: Karacalı, H., Donum., N., Cebel, E. Cryptographic Enhancement of Named Pipes for Secure Process Communication. The European Journal of Research and Development, 4(2), 110-129.

Abstract

Improving in-car features and increasing the driving experience has become one of the main goals in the automotive industry. Driver-specific adjustments, automatic operations, increased functionality of infotainment systems and the development of automotive navigation systems, as discussed in this article, have been factors that improve driving quality. This paper presents a novel approach to navigation system development tailored for automotive use, aiming to enrich the driving experience through the integration of artificial intelligence (AI) technologies. The proposed system operates by leveraging AI algorithms to gather pertinent information about historical destination landmarks with temporal significance once a driver selects it via the navigation interface. Moreover, the system not only provides information about the selected destination, but also provides information about all historically important locations on the route, depending on the location over GPS. Subsequently, this information is intelligently synthesized and conveyed to the driver in a voice-assisted manner, enriching their journey with insightful details and enhancing situational awareness. By seamlessly integrating AI-driven contextual information delivery into the navigation paradigm, this system aims to not only facilitate smoother navigation but also elevate the overall driving experience. Through a synthesis of AI, natural language processing, and location-based services, this innovation represents a significant step towards more intuitive and enriching automotive navigation systems.

Keywords: *Navigation, automotive, in-car features, location-based services*

1. Introduction

The automotive industry is undergoing a transformative phase, where the focus has shifted from conventional transportation to enhancing the overall driving experience. This evolution is driven by a growing demand for in-car features that not only provide comfort and convenience but also integrate seamlessly into the driver's lifestyle. Central to this evolution are advancements such as driver-specific adjustments, automated operations, and the continuous improvement of infotainment systems. Moreover, the development of sophisticated automotive navigation systems has emerged as a critical area for enhancing the quality of driving experiences.

Amidst these advancements, the integration of AI technologies stands out as a pivotal enabler of innovation in the automotive sector. AI-powered navigation systems, in particular, hold immense promise in revolutionizing the way drivers interact with their vehicles and the road. By leveraging AI algorithms, these systems have the potential to offer personalized, contextually relevant guidance and insights, thereby enhancing both the efficiency and enjoyment of driving.

This paper presents a new approach to developing a navigation system specifically designed for automotive use, focusing on enriching the driving experience through the integration of AI technologies. The basis of the proposed system is the use of AI algorithms to collect data on historical destination landmarks of temporal significance. By understanding the historical context of destinations, the navigation system can provide drivers with comprehensive information beyond traditional route guidance.

Upon driver selection of a destination via the navigation interface, the system dynamically retrieves information about historically significant locations along the route, leveraging GPS-based location tracking. As locations of historical importance on the route are approached at a certain distance, the collected data is prepared to be transferred to the driver. Furthermore, the system employs voice-assisted information delivery to ensure seamless integration into the driving experience, enhancing situational awareness and minimizing distractions. The data collected by AI is transferred to the driver with the speech feature. The driver can stop and restart this storytelling, which is under user control, and can switch between detected historical locations. This is the control panel provided to the user by the system, which works completely automatically with AI support in the background.

There are studies in the literature that, although not similar, are close in concept and compatible in purpose. In recent years, various studies on improving driving experience, integrating AI into automotive, and providing online guidance about historical regions are included in this paper. A similar technology was used in Vékony's study in 2016 and a speech recognition mechanism was used in the car navigation industry. The text to speech used in that work by Vékony was also used in the study in this paper [1].

Another related work is the mobile application called 'Piri Guide'. Thanks to ready-made routes created in certain parts of the world, or while the user is actively traveling in specified regions, it conveys information to the user about important points, historical monuments, and structures in the vicinity. However, there is no AI integration in this system. Since it works based on pre-recorded data, it is only valid in certain regions [2].

Throughout this paper, we delve into the methodology employed in developing the AI-driven navigation system, emphasizing its potential to not only facilitate smoother navigation but also elevate the overall driving experience. Through a synthesis of AI, text to speech, and location-based services, this innovation represents a significant step towards more intuitive and enriching automotive navigation systems.

The following sections provide a detailed review of the system's design, implementation, and outcomes, as well as a comprehensive analysis of its functionality and effectiveness. Additionally, the technologies and materials used throughout development are also included. Additionally, the broader implications of the findings are discussed and potential avenues for future research in this emerging field are outlined. Ultimately, this article aims to contribute to the ongoing discourse on advancing automotive technology towards smarter, user-centered solutions.

2. Materials

2.1. CPU

Specialized embedded systems differ from general-purpose computers in their tailored design aimed at efficient operation within limited resources. Embedded CPUs are customized to prioritize compactness, low power consumption, and integrated memory, emphasizing quick response times and energy efficiency rather than high clock speeds. Despite potentially lower clock speeds, these CPUs excel in performance for their designated tasks [3]. The selection of CPU architectures significantly impacts the design and functionality of embedded systems, driven by specific requirements and applications [3].

In this investigation, the NXP i.MX 8QuadMax CPU serves as the central processing unit, designed specifically for robust system integration in embedded applications. It utilizes a heterogeneous multicore architecture, combining Arm Cortex-A72 and Arm Cortex-A53 cores to efficiently manage high-performance and energy-efficient tasks [4]. For graphic processing, it integrates a GC7000XSVX GPU capable of handling 4K resolution videos and offering high performance for embedded graphics tasks [4]. The CPU also offers extensive connectivity options, including dual PCIe interfaces, dual Gigabit Ethernet, and various data communication ports, making it adaptable and connectivity-oriented [4]. Its security features comply with industry standards, making it particularly suitable for security-focused embedded applications.

2.2. Android Automotive Operating System

The Android Automotive Operating System (AAOS) is an adapted version of the Android platform tailored explicitly for vehicle use. It offers an intuitive interface for in-car infotainment and navigation systems, facilitating seamless integration of smartphone features and services into the vehicle environment [5]. AAOS encompasses a diverse array of functionalities, including media playback, navigation, communication, and connected services, supporting various connectivity options like Bluetooth, Wi-Fi, and cellular connectivity for effortless interaction with other devices and services [6]. Its flexible architecture allows vehicle manufacturers to customize and tailor the user experience according to their specific requirements. Furthermore, AAOS serves as a platform for developers to design and distribute automotive-centric applications, thereby augmenting the system's capabilities. In summary, the Android Automotive Operating System presents a robust and adaptable solution for automotive infotainment and navigation systems, delivering a rich and immersive experience for both drivers and passengers.

Figure 1 represents the abstract layer architecture of Android Automotive Operating System (AAOS). The top layer is related with the applications that anything the user can interact with such as navigation, music, and messaging. The service layer provides core services that the applications need to run, such as window management, resource management, and security. Hardware Abstraction Layer (HAL) is a separator between the operating system and the hardware. It allows the operating system to work with different types of hardware without having to be rewritten for each type of hardware. The last layer, Board Support Package (BSP) includes the Linux kernel to run Android Automotive OS.

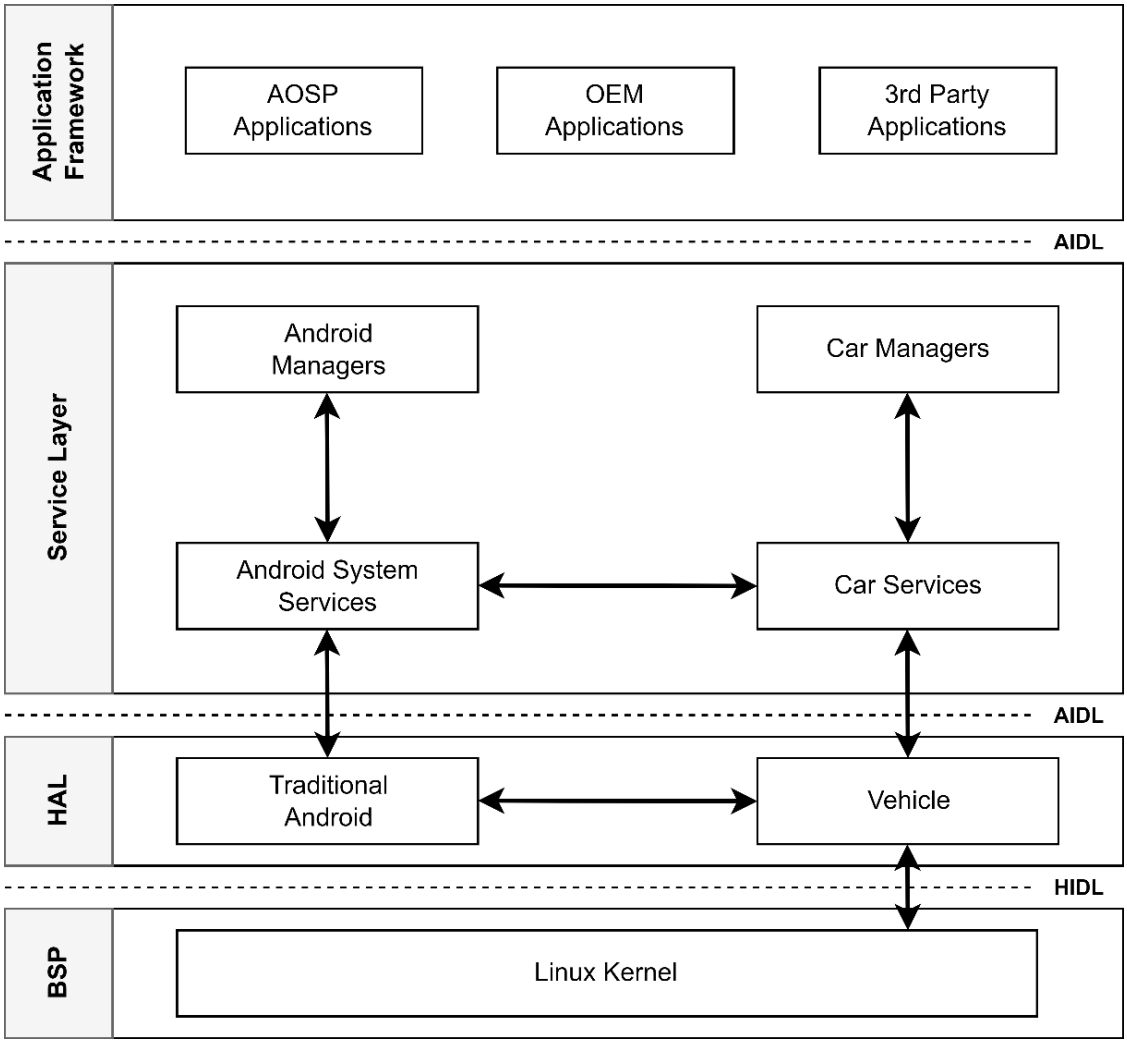

Figure 1: Abstract layer architecture of Android Automotive Operating System [7]

2.3. Organic Maps

Organic Maps is a free and open-source mobile application offering offline navigation services with a focus on privacy and community-driven data [8]. Developed as a fork of the popular 'Maps.me' application, it stands out with its commitment to user data protection and its reliance on crowd-sourced data from OpenStreetMap (OSM) [9].

In addition to map display, its key features include transferring real-time traffic data, providing navigation features by providing public transport routes, being open-source, and being available offline [8]. Moreover, the application adheres to strict privacy principles. It avoids user tracking, data collection, and third-party integrations [10].

Figure 2: Organic Map logo [8]

In this study, the integration of artificial intelligence (Gemini AI), which will provide data collected about historical regions, and text to speech algorithms, which will voice this data to inform the driver, has been made into the Organic Maps application. It was chosen as the navigation basis of this study due to its open-source development and wide scope.

2.4. Gemini AI

In the ever-evolving landscape of AI, Google's Gemini AI emerges as a groundbreaking technology, pioneering a new era of "multimodal AI." Unlike traditional models confined to individual modalities like text or images, Gemini seamlessly integrates information across text, images, and audio, unlocking richer context analysis and reasoning capabilities.

Departing from the limitations of unimodal approaches, Gemini's core strength lies in its ability to process and understand information from various modalities simultaneously [11]. This enables the model to handle complex scenarios requiring comprehension of diverse information sources, paving the way for advancements in various fields like visual question answering, multimodal search, and machine translation. Furthermore, the end-to-end training approach optimizes information flow across modalities, fostering comprehensive understanding [12].

In this study, data about historical locations up to the destination are obtained online with Gemini AI. Thanks to the integration of the AI algorithm into the Organic Maps navigation application, locations on the route were detected and these locations were sent

sequentially as input to Gemini AI. It was preferred because of its fast response time, its popularity close to the date of the study, its ease of configuration, and its ease of implementation into projects.

2.5. TextToSpeech

The TextToSpeech (TTS) application programming interface (API) in the Android SDK empowers developers to seamlessly integrate speech synthesis capabilities into their applications. This enables devices to "speak" textual content, enhancing accessibility, user interaction, and information delivery. Text-to-Speech conversion by ensuring proper pronunciation and inflection, fine-tuning speech characteristics, and queue management, offline speech synthesis, and third-party TTS engines are the key features of this API [13].

Integration and development of this API requires several steps. Firstly, the TTS API is added into the project by including necessary permissions and dependencies. Then, an instance for TextToSpeech is created and potential speech data download. The speak method from this API specifies the text to speech by synthesis. Finally, the events should be handled like successful synthesis completion or errors to manage speech playback [14].

2.6. Android Studio

Android Studio serves as the primary integrated development environment (IDE) specifically crafted for Android app development. This sophisticated tool, endorsed by Google, offers an extensive array of features and resources to streamline the app development workflow. Android Studio amalgamates efficiency, flexibility, and robust functionality, making it a preferred choice among developers embarking on Android application projects [15]. Its user interface is intuitive and user-friendly, facilitating seamless navigation and access to various tools, thus promoting productivity within the development workspace.

The IDE incorporates an advanced code editor that supports multiple programming languages, with a primary focus on Java and Kotlin for Android development. These features enrich the coding experience by providing functionalities like code completion, syntax highlighting, and real-time error checking [15].

The development process of the AG application was conducted using Android Studio, the officially endorsed IDE for Android app development. Android Studio was selected owing to its status as the official IDE, offering a comprehensive suite of features and

resources to expedite the development of Android applications [15]. Employing Android Studio ensures smooth integration with the latest Android SDKs, APIs, and platform updates, guaranteeing optimal performance and utilization of the latest Android OS features [15].

The choice of Java as the programming language for the AG application is driven by the necessity for platform independence and the widespread adoption of Java in Android development. Java's principle of "write once, run anywhere" aligns with the objective of ensuring that the AG application operates seamlessly across various devices and operating systems, providing a consistent user experience. The extensive support and libraries available within the Java community simplify the development process of the AG application, enhancing its reliability.

In conclusion, the AG application was developed utilizing Android Studio as the designated IDE, leveraging its official status and robust features tailored for Android development. Adopting Java as the programming language ensures platform independence and benefits from the extensive support within the Android development community. This strategic approach aims to create a potent and user-friendly AG application capable of delivering a consistent experience across diverse platforms.

3. Method

This work was carried out entirely on a previous open-source navigation application, Organic Maps, and therefore, it was first aimed to obtain the source codes of this project and compile it for the Android-based operating system. Current source codes were downloaded from the project's shared repository web page. In order to develop this project, there must be a study in which we will have the source codes, because artificial intelligence, text voice-over, and other remaining modules will be added to these source codes and the study will be completed. The navigation application, whose source codes were downloaded, was first opened as an Android Studio development environment, and compiled for the Android Automotive operating system.

The next stage of the study is to examine these source codes to obtain the necessary information. The purpose of this review is to determine in which method and how the project uses the data, the current location and the latitude and longitude locations of the selected destination, which will be given as a prompt to the AI model in the future and enable it to receive the necessary information. Thanks to the gemmulation method of the LocationHelper class, the current location of the vehicle can be read in terms of latitude

and longitude. These read values are converted from decimal to degree, minute, and second as shown in Figure 3 for artificial intelligence to better understand. Obtaining this data was appointed as the first step of development as it will be used in the prompt that will be given to AI in the future.

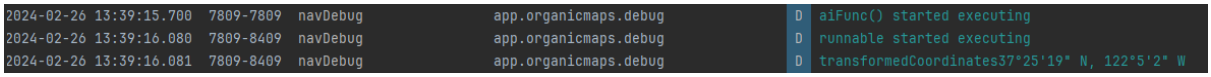

Figure 3: Log messages for read current location

The next and always repeated step in the study is based on the system exchanging data with AI and presenting the coordinates to the AI in a certain structure and receiving information from it. Therefore, the first technical process for this is to apply the AI integration to the project. Gemini AI's application programming interface (API), which has up-to-date data and has a much higher response time than other AI robots, was used to be integrated into the project. Transactions made using AI cannot be used without starting navigation. Since running this system before a route is created for navigation would be unnecessary workload, the connection with AI is activated after a route is planned. To create the route, a destination must be selected on the application and click on the 'Route To' button. The application creates a driving route from the current location (obtained in the previous step) to the selected destination, as traditional navigation applications do. With this operation performed with the onNavigationStarted function, several interface operations are also performed.

Creating the route and starting navigation also enables the integrated AI system to be activated. The function that starts the artificial intelligence is also called within the function that starts this navigation. This function is the part where the system exchanges data with all AI, starting from the integration of AI. The first part of this data exchange is the current location of the vehicle (in degree, minute and second), which we previously obtained from the application, is sent to the AI with a prompt like this: “"list the historical places near these coordinates:" + transformedCoordinates + ". write every place side by side with '*' between them. do not explain them. do not start your response with '*'"”. As seen in the prompt, it is desired to list historical points close to the given coordinate. It is very important that the received message be in a certain format so that it can be parsed and listed by the system, so these historical locations are required to be separated from each other with the * sign. It is also specifically mentioned to the AI side that it does not give any unnecessary information and does not make any statements as it is only intended to share the names of historical landmarks. In this section, it is aimed to obtain

and list the names of historical points currently nearby the vehicle. Figure 7 represents the response of the AI for the prompt given above. The near historical places around the current position (named as transformedCoordinates in the project) are listed and separated with the * sign.

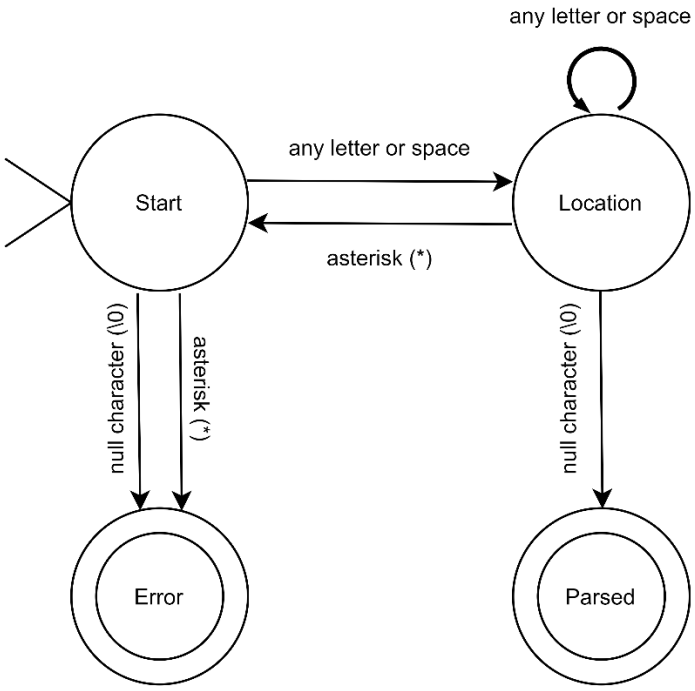

Figure 4: Location name parser finite automaton

Thanks to the developed algorithm, the response shown in Figure 7 is separated item by item using a regular expression and assigned to a string array by using a parser that designed as shown in Figure 4. Designed to comply with the given prompt, this parser can terminate at an error (or empty) or successful parsing node. If any character other than a letter is encountered in the start node, an abnormal situation is detected and the parser returns an error to the relevant functions. In the parsing process, which proceeds in its normal flow, a new location will appear where the asterisk (*) sign is seen, so it returns to the start node. The same location data continues to be received until the asterisk sign appears. If a null character (\0) comes after a letter, the parser has completed its operation successfully and the historical locations are appropriately transferred into the string array. Thus, a list of historical places close to the given location is obtained. Then, the data that will be asked to be sent to us by AI is the distance. It is of great importance how far the mentioned historical locations are from our current location, because in order for the system to transfer these historical locations to the driver, they must be closer than

a certain threshold value. This list of historical locations, kept in a string array, is sent to the AI via a different prompt. This prompt is as follows: "give me coordinates of these locations:" + Arrays.toString(locationList) + " in decimal format. write every coordinate side by side without space. write '*' after every coordinate pair. give the coordinates in decimal form. write ',' between every coordinate pair. do not leave any space. example output: '37.22,122.3*37.12,122.12'". As can be seen in this prompt, decimal type coordinates of the locations whose names are given in the string array are requested. The main purpose of requesting these latitude and longitude values arises from the need to measure the distance between the instantaneous vehicle position and these historical regions. These historical region names, kept under the name locationList, are sent to AI. Just like these names, a parsing process will be applied, so no spaces are required.

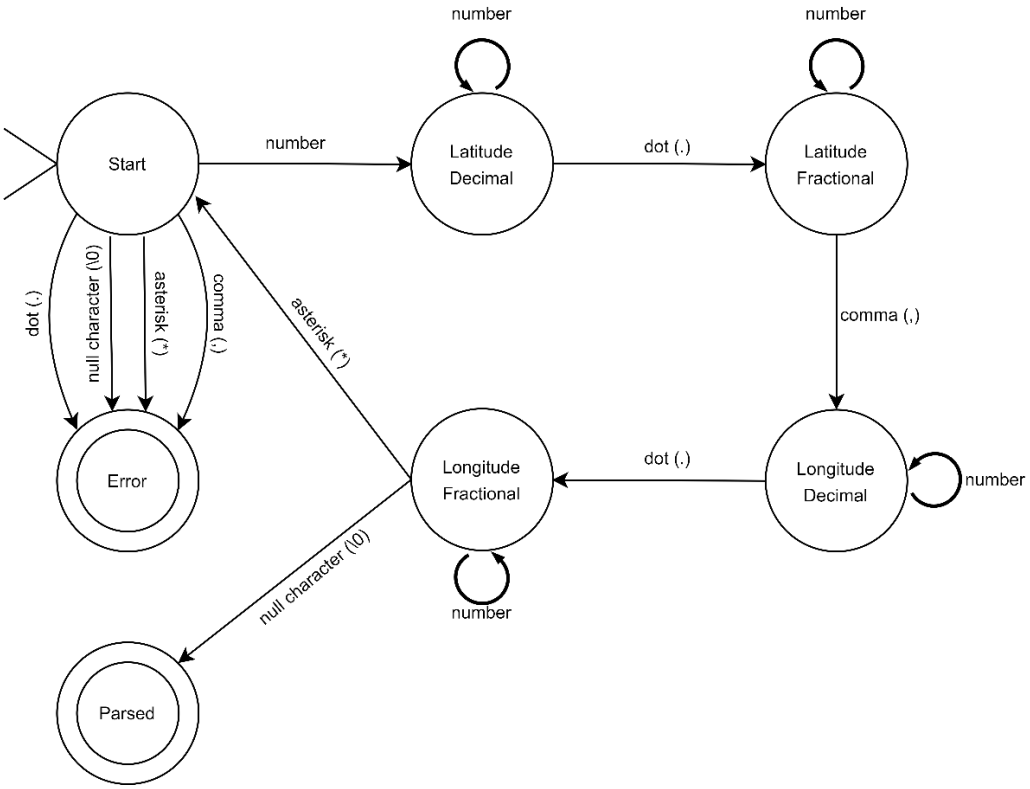

Figure 5: Location coordinate parser finite automaton

In addition, it is requested that there be a "," sign between latitude and longitude, and a * sign is also required between coordinates. Similar to the previous parsing process, a parsing is applied to the coordinates received by the AI and these coordinates are assigned to a string array as shown in Figure 5. The parser algorithm prepared for coordinates is similar to the previous one, but slightly more complex. There is a comma

separating both latitude and longitude, and a dot separating the decimal part for both values. Also, as in the previous parser, the asterisk is used to separate coordinates. The parser, which begins at the start node, returns the function as an error or empty list unless it encounters a numerical value. In normal operation, the decimal part of the latitude value is read until the dot sign is seen, and the fractional part of the latitude is read after the dot. Since there will be a comma between latitude and longitude, when the parser detects the comma, the same operation for latitude is applied to longitude. With the asterisk sign, the coordinate for a location is read successfully and the automata returns to the start node to read the coordinate of the new location. When there is a null character instead of an asterisk, the parser successfully completes the coordinate reading process. As a result of this process, both the historical locations in the vicinity and the latitude and longitude of those locations are kept in separate string arrays. The AI's response to this prompt is shown in Figure 8 below.

As shown in Figure 8, the received coordinate data is stored in a string array. These coordinates are used to calculate how far the current vehicle location is from historical locations specified by the AI. Using the coordinate distance finding function, the distance to each historical place is measured and these distance values are recorded in an array.

At this stage of the study, sharing of coordinates and historical or cultural location data between the AI integrated with the navigation application is terminated for a while. In this part, it is expected that the study will provide constant control of the data it receives and transfer data about locations to the driver when appropriate conditions occur. It includes distance control, collecting summary information about the relevant location by the AI engine, and transferring this collected text to the driver with the help of Text-to-Speech. The function that performed the AI operations up to this point is exited and a different function is called. Location names kept in the form of an array, distance data to the locations kept in the form of an array, the integrated AI model, and the executer object that enables the task to run are given as arguments to this function. Distance measurements kept in an array are examined sequentially by this called function. Starting from the first element of the array, it is checked whether the vehicle is closer than 5 kilometers to its current location. In this case, if the distance is more than 5 kilometers, the next element of the distance array is checked without any action. All values are examined iteratively unless there is a distance of less than 5 kilometers.

On the other hand, if one of the important places specified by the AI is closer than the specified threshold value distance, that is, if any element of the distance array is less than

5 kilometers, the function that will exchange data with the AI is called again with the prompt given below and a summary description text about the relevant location name is requested: “"tell me about " + locationList[locationCounter]”. Since the names array of historically important locations and the distance array are in the same order, the name of the nearby point can be found with the same index of the element detected in the distance array. Thus, thanks to this prompt, general summary information about the selected location is requested by the AI. Figure 9 shows an example of the AI's answer to this prompt.

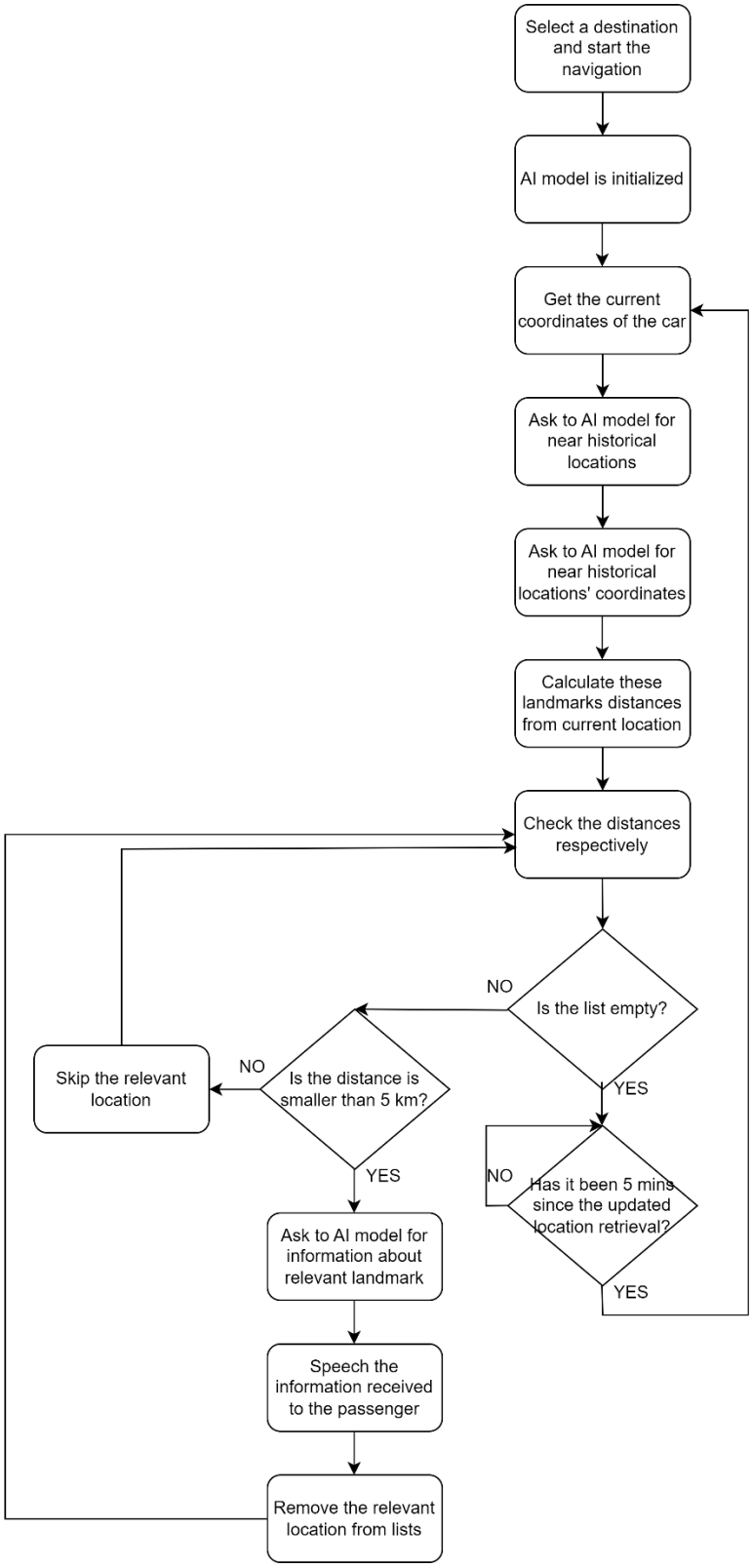
Select a destination and start the navigation
AI model is initialized
Get the current coordinates of the car
Ask to AI model for near historical locations
Ask to AI model for near historical locations' coordinates
Calculate these landmarks distances from current location
Check the distances respectively
Is the list empty?
NO
YES
Is the distance is smaller than 5 km?
NO
Skip the relevant location
YES
Ask to AI model for information about relevant landmark
Has it been 5 mins since the updated location retrieval?
NO
YES
Speech the information received to the passenger
Remove the relevant location from lists

Figure 6: Flowchart of the storytelling system

The next phase in the study is to inform the driver and other passengers in the vehicle about the important location, which is close enough and whose summary information is received in text form. This entire text is transmitted to the integrated Text-to-Speech module and all information is transmitted to the driver at the specified speed and in English. The length and scope of this process may vary depending on the depth of information provided by the AI model. When the Text-to-Speech process is completed, the relevant location and distance data of the location are removed from the arrays. Additionally, the function that checks the distance values is called recursively to decide whether to read subsequent historical landmarks. This recursive call continues until there is no unchecked location in the list.

As the last component of the system, since the vehicle may be moving, the current coordinates of the vehicle are taken periodically every 5 minutes and all operations are performed from the beginning, the lists are renewed and the process is repeated.

Figure 6 represents the general flow of the whole system for intelligent storytelling development on Organic Maps navigation application for the Android Automotive OS. The operations and transitions of the entire system according to the conditions can be observed through this figure.

The last development made within the scope of the project is to enable the user to control this working system on the narrative side. Three new buttons have been integrated into the default interface of the navigation application chosen as the base. These buttons, respectively, allow information to be transferred about the previous historical location in the list, to stop the narration with text to speech or to start if it has been stopped, and to transfer information about the next historical location in the list. The transition between stop and start buttons has been transferred to the UI side in a convenient way for the user.

Historical locations provided by AI and named by parser are kept in a list called narration. AI-generated text content about these locations is kept in exactly the same order as in this list. Thanks to a common index value, the transitions, stop, and start functions provided in this latest development can be used. When moving to the previous or next one, this index value is manipulated and moved to the next in the list. If the end of the list is reached, a warning system has been prepared to inform the user.

4. Results

The study and development which are designed for an in-car entertainment-information system that is integrated with artificial intelligence, only outputs the speech of the text produced by the AI model about the location that meets the criteria and has been selected by the algorithm at that moment. Therefore, the results that can actually be stated in this paper are the answers produced by the AI in the system until it reaches that speech to text.

```
San Jose Museum of Art*
Rosicrucian Egyptian Museum*
The Tech Interactive*
Children's Discovery Museum of San Jose*
Japanese American Museum of San Jose*
San Jose Museum of Quilts & Textiles*
San Jose History Museum*
Kelley Park*
History Park at Kelley Park*
San Jose Civic Auditorium
```

Figure 7: AI's response on nearby historical locations' name prompt

After receiving the current location of the vehicle, the first output of the system can be shown as the first output of the system to return the nearby historically important locations in a format with a prompt to the AI model as shown in the Figure 7. The process of listing nearby locations according to current coordinates is repeated periodically until the navigation route is completed or finished by the driver.

The second output is the latitude and longitude coordinates of the locations listed in Figure 7 by the AI model. These outputs presented in Figure 8 are important since the distances between the vehicle and its current location will be measured using these coordinates. These outputs are also returned by the AI in a certain format so that it can be parsed.

```
37.333333,-121.888889*
37.322222,-121.903611*
37.338611,-121.893056*
37.298611,-121.899167*
37.335278,-121.890278*
37.333056,-121.890278*
37.335278,-121.886944*
37.3225,-121.888056*
37.3225,-121.888056*
37.338056,-121.886111
```

Figure 8: AI's response on nearby historical locations' coordinates prompt

As the final output of the data exchange between the AI model and the navigation application, AI summary information is transferred about one of the locations in the list that is determined to have a certain level of proximity. This text is the text that will be conveyed to the driver and other passengers with speech. An example return of AI model for San Jose Museum of Art is shown in Figure 9.

```
The San Jose Museum of Art (SJMA) is a modern and contemporary art museum located in downtown San Jose, California.
Founded in 1969, the museum is dedicated to collecting, preserving, and exhibiting works of art from the 20th and 21st centuries.

The SJMA's collection includes over 2,500 works of art, including paintings, sculptures, drawings, prints, photographs, and new media.
The museum's collection is particularly strong in California art, with works by artists such as Richard Diebenkorn, Wayne Thiebaud, and Joan Brown.
The museum also has a significant collection of Latin American art, as well as works by international artists such as Pablo Picasso, Henri Matisse, and Andy Warhol.

The SJMA's exhibitions program features a mix of solo and group shows, as well as thematic exhibitions that explore different aspects of modern and contemporary art.
The museum also hosts a variety of educational programs, including lectures, workshops, and tours.

The SJMA is housed in a striking building designed by architect Cesar Pelli. The building features a series of interconnected pavilions, each of which is dedicated to a different aspect of the museum's collection.
The museum also has a sculpture garden, which features works by artists such as Henry Moore, Barbara Hepworth, and Isamu Noguchi.

The SJMA is a major cultural institution in San Jose and the surrounding area. The museum is a popular destination for tourists and locals alike, and it plays an important role in the city's arts community.

Here are some additional facts about the San Jose Museum of Art:

 * The museum is accredited by the American Alliance of Museums.
 * The museum is a member of the North American Reciprocal Museum Association (NARM).
 * The museum offers free admission on the first Friday of every month.
 * The museum has a café and a gift shop.
 * The museum is accessible to visitors with disabilities.
```

Figure 9: AI's response on nearby historical locations' description prompt

As a final result, a snapshot can be presented where information is transferred thanks to AI, accompanied by an actively working road route. The buttons seen at the top right are the panel that performs transition and stopping operations and can be controlled by the user. Additionally, the application works in both vertical and horizontal frames.

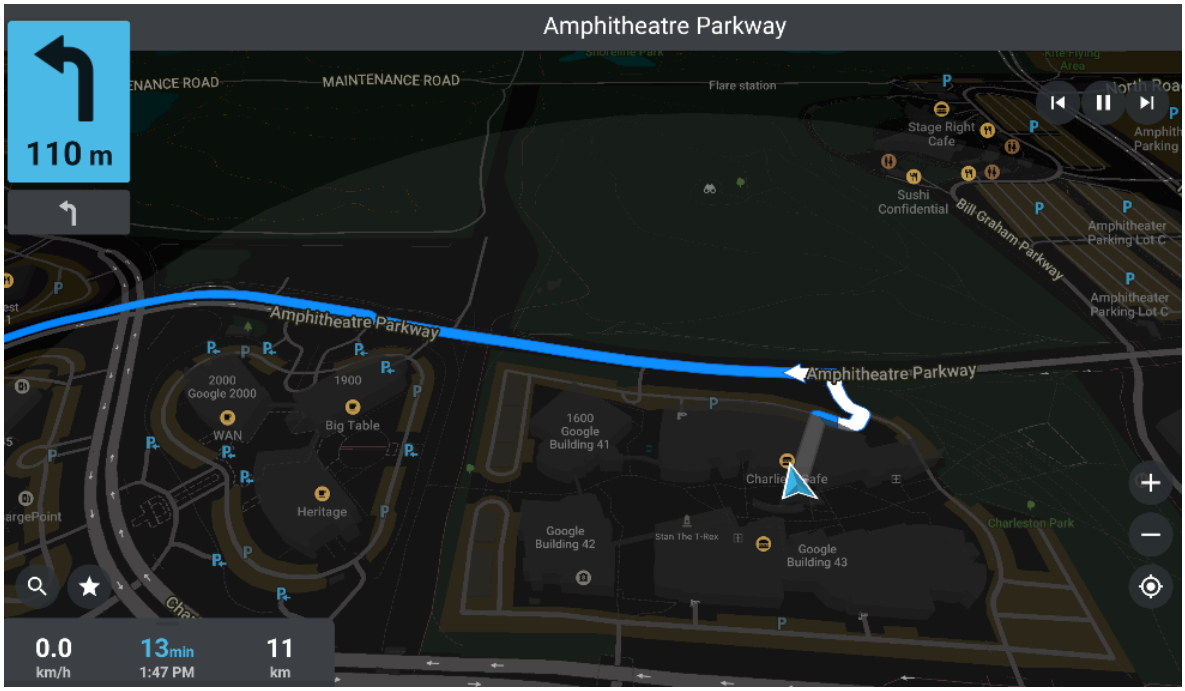

Figure 10: A snapshot from navigation when storytelling is active

5. Discussion and Conclusion

In conclusion, this paper presents a novel approach to enhancing the driving experience through the integration of AI technologies into the development of an automotive navigation system. The proposed system, implemented on the Organic Maps navigation application for the Android Automotive Operating System, leverages AI algorithms to provide drivers with insightful information about historically significant landmarks along their route. The integration of Gemini AI, a multimodal AI technology, along with Text-to-Speech capabilities, enriches the journey by delivering historical context and details about important locations.

The developed system successfully achieves its objectives, seamlessly integrating AI-driven contextual information delivery into the navigation paradigm. By periodically collecting and processing data about nearby historical locations, the system enhances situational awareness and provides an engaging and informative driving experience. The integration of the Android Automotive Operating System, NXP i.MX 8QuadMax CPU,

and Organic Maps application forms a robust foundation for the implementation of this innovative navigation system. The Android Automotive Operating System provides a flexible and customizable platform for in-car infotainment and navigation systems, allowing seamless integration of diverse functionalities. Organic Maps, chosen for its open-source development and privacy-focused approach, serves as a suitable base for incorporating AI-driven features.

Moreover, the integration of Gemini AI introduces a multimodal aspect, enabling the system to process information across text, images, and audio simultaneously. This not only enhances the richness of information provided to the driver but also sets the stage for potential future advancements in other multimodal applications.

The system's functionality, as outlined in the methodology section, successfully captures the essence of providing historical information during the driving experience. The user interface modifications, including the addition of control buttons for narration, contribute to user interaction and control. The periodic updates of the current vehicle coordinates, coupled with the AI-driven storytelling mechanism, create a dynamic and engaging experience for the driver. The ability to control the narration and obtain information about historical landmarks adds a layer of personalization to the navigation system.

Despite the achievements, there are certain limitations to the current implementation. The system relies on the periodic retrieval of historical information, and the level of detail provided by AI may vary. Future iterations could explore real-time data streaming for a more dynamic experience. Additionally, the system's reliance on GPS coordinates may face challenges in dense urban environments or areas with limited historical data. Future research could investigate alternative methods, such as incorporating sensor fusion technologies, to enhance location accuracy. The user interface and control buttons introduced in this development are designed for practicality, but further user testing and feedback could refine the user experience. Incorporating user preferences and expanding language support could contribute to a more inclusive and personalized system.

6. Acknowledge

We want to express our appreciation to Huseyin Karacali, the Software Architect, for his adept guidance and inspiring leadership. Furthermore, we recognize the precious support offered by TTTech Auto Turkey during the project's development phases.

References

[1] Vékony, A. (2016). Speech Recognition Challenges in the Car Navigation Industry. In: Ronzhin, A., Potapova, R., Németh, G. (eds) Speech and Computer. SPECOM 2016. Lecture Notes in Computer Science(), vol 9811. Springer, Cham. https://doi.org/10.1007/978-3-319-43958-7_3

[2] Piriguide, "Piri Guide – Travel planner," App Store, https://apps.apple.com/tr/app/piri-guide-travel-planner/id1095698831.

[3] "Central Processing Unit," Central Processing Unit - an overview | ScienceDirect Topics, https://www.sciencedirect.com/topics/engineering/central-processing-unit.

[4] "I.MX 8 Family applications processor: ARM cortex-A53/A72/M4," i.MX 8 Family Applications Processor | Arm Cortex-A53/A72/M4 | NXP Semiconductors, https://www.nxp.com/products/processors-and-microcontrollers/arm-processors/i-mx-applications-processors/i-mx-8-applications-processors/i-mx-8-family-arm-cortex-a53-cortex-a72-virtualization-vision-3d-graphics-4k-video:i.MX8.

[5] "What is Android Automotive? :   Android Open Source Project," Android Open Source Project. [Online]. Available: https://source.android.com/docs/devices/automotive/start/what_automotive. [Accessed: 01-Feb-2023].

[6] "Design for driving | google developers," Google. [Online]. Available: https://developers.google.com/cars/design/automotive-os?hl=tr. [Accessed: 01-Feb-2023].

[7] H. Hofmann et al., "Android Automotive OS Whitepaper: Android Automotive OS Book," Android Automotive OS Book | Build your infotainment system based on Android Automotive OS, https://www.androidautomotivebook.com/android-automotive-embedded-os-whitepaper/.

[8] Organic Maps. (2023, Jan 1). Organic Maps: Open-source, offline maps: https://organicmaps.app/.

[9] Organic Maps: Open-source, offline maps on GitHub: https://github.com/organicmaps/organicmaps.

[10] OpenStreetMap. (n.d.). OpenStreetMap. Retrieved from https://www.openstreetmap.org/.

[11] LaMDA, et al. (2023). Gemini - Google DeepMind. https://deepmind.google/technologies/gemini/.

[12] Baptista, J., et al. (2023). Introducing Gemini: Google's most capable AI model yet. Google AI Blog. https://developers.googleblog.com/2023/12/how-its-made-gemini-multimodal-prompting.html.

[13] Android Developers Documentation: https://developer.android.com/reference/android/speech/tts/TextToSpeech.

[14] Implementing Text-to-Speech in Android: https://www.tutorialspoint.com/android/android_text_to_speech.htm.

[15] "Android Developers: Android Mobile App Developer Tools." https://developer.android.com/.