Research Article

Personalized and Dynamic Rear-View Mirror Adjustment and Profiling with Voice Signature

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 September 12, 2023 and in final form December 25, 2023)

Reference: Karacalı, H., Donum, N., Cebel, E. Personalized and Dynamic Rear-View Mirror Adjustment and Profiling with Voice Signature. The European Journal of Research and Development, 3(4), 390-413.

Abstract

This paper introduces a novel automated system designed for adjusting automobile rear-view mirrors intelligently by utilizing head pose orientation. In today's increasingly personalized car interiors, ensuring the correct alignment of the rear-view mirror based on the driver's head orientation significantly enhances both safety and driving comfort. Manual adjustment of rear-view mirrors can result in issues such as improper angles and driver distractions. This study aims to automate the rear-view mirror adjustment process by accurately detecting the driver's head position and orientation, thereby mitigating these challenges. The system incorporates a camera within the vehicle's cockpit to track the driver. Raw data captured by the camera undergoes processing using the Perspective-n-Point (PnP) algorithm to determine the driver's head position and orientation. The computed positional information is then employed to precisely align the rear-view mirror, optimizing the field of view and eliminating blind spots. Within a brief timeframe, the system establishes the most suitable mirror settings for any driver. Moreover, it dynamically adapts to changes in the driver's posture during driving, ensuring consistently optimal visibility. Additionally, faster and more comfortable use of this in-car smart system is aimed with voice identification technology. Mirror angle values are kept in memory and can be applied instantly at the driver's request, with driver profiles created with voice identifications. Consequently, this

research proposes an innovative application that contributes to the integration and personalization of smart technologies, potentially becoming part of forthcoming automotive cockpit designs.

Keywords: Rear-view, smart cockpit, face recognition, voice identification

1. Introduction

The automotive industry is undergoing a major transformation with the rapid development of technology. Vehicles, which are now more than just means of transportation, aim to establish an even closer relationship with drivers. Accordingly, driver-based profiling and vehicle configurations have become a rapidly popular trend in the automotive industry in recent years.

Traditional cars often offered the driver limited tuning options or had to adjust many of them manually. Today, however, cars have become smart devices. Drivers can now make their vehicles unique with many different customization options. Driver tuning is an innovation that completely transforms the in-car experience.

The basic idea behind this trend is to allow drivers to interact more with their vehicles. Configuration by driver in automotive puts drivers one step ahead and allows them to shape their vehicles according to their personalities, preferences and needs. This makes the driving experience more enjoyable and satisfying.

Customizable in-vehicle systems encompass certain challenges and drawbacks about the manual configuration of each profile after drivers have saved their profiles. The process of manual profile selection necessitates drivers to manually choose their profile settings every time they enter their vehicles. This requires drivers to invest time and effort in locating and selecting their settings. Particularly in cases where multiple drivers utilize the same vehicle, each driver being obliged to choose their own profile becomes a time-consuming and laborious process. To overcome these challenges, efforts are being made in the automotive domain to develop more user-friendly and automated profile selection systems. Systems developed to automate the process of seat adjustment using facial recognition are examples of such studies [1]. In-vehicle customizable systems can also be controlled by voice recognition technologies in addition to facial recognition technology. Voice processing and voice recognition in the automotive industry is a technology that has been studied for years. Managing in-car features with voice commands has become a topic that some brands are particularly interested in. In the paper published by Victor Ei-Wen Lo and Paul A. Green (2013), they focused on developing a speech interface in the automotive industry as a solution by identifying problems with factors that may

distract the driver, such as navigation and telephone [2]. In recent years, these systems have become more realized in the automotive industry and the development of voice and human machine interface has turned to more specific issues. The study conducted by W. Astuti and his colleagues (2021) focused on the development of Headlights Beam Control with a voice command system. In addition, in this recent study, Mel Frequency Cepstral Coefficient (MFCC) was used as a neural network structure and identification technology similar to our own study [3].

In this study, artificial intelligence assisted face recognition technology, which was used in vehicles before, is looked at from a different point and the rear view mirror is adjusted at the most ideal angle according to the head posture of the driver. This angle, which changes dynamically throughout the ride, not only adds an innovative dimension to tailor made adjustments in the automotive industry, but also improves the driving experience by saving time. Moreover, the study uses technology that helps automate and personalize in the automotive industry, such as voice identification. Thanks to the models created with neural network structures, the rearview mirror angle that the driver previously found and recorded as appropriate is automatically applied by the system by recognizing the driver's voice. A very quick adjustment can be made for each driver by keeping these mirror angle values in the profiles previously created with the sound ID. It is envisaged that the work will not only focus on the rearview mirror angle, but also lead to studies that will enable the adjustment of some structures in the vehicle through voice command and voice identification.

Nowadays, voice recognition technology has made significant progress, enabling accurate and efficient voice interactions between humans and machines. If the issue is looked at from the perspective of establishing a driver-centric structure in in-vehicle use, there have been studies that have highlighted this idea since the recent past. In the automotive industry, managing all systems, from driving experiences to entertainment, step by step with personalized profiles, has come to the fore many times. To give an example, in the study published by Martina Hasenjäger and Heiko Wersing (2017), they prepared a comprehensive paper on personalization with driver preferences in advanced driver assistance systems and autonomous vehicles [4]. Additionally, another study addressing the issue of human-vehicle interaction with the driver centripetal structure was conducted by Jung Min Lee and Da Young Ju (2018). They carried out this study by taking data from many drivers to classify the services that can be applied in the design of the automobile as socio-behavioral [5].

2. Materials

2.1. CPU

Computer systems known as embedded systems are created with a specific objective in mind. They are typically tuned to function effectively with restricted resources. Because they are designed to work well in devices with limited resources, embedded system CPUs are different from general-purpose computer CPUs [6]. These CPUs have features like integrated memory, low power consumption, small size, and hardware integration. Even though their clock speeds may be lower, they place a higher priority on attributes like excellent efficiency, quick response times, and low power consumption. They are able to give high performance for their intended jobs because to this optimization [7].

There are many different CPU architectures available for embedded systems, and each architecture has a big impact on how these systems are designed and function. Based on the unique demands and needs of embedded systems and their applications, the CPU architecture is chosen.

The NXP iMX8 Quad Max CPU has been selected for this study. The ARM Cortex-A72 and Cortex-A53 cores of the IMX8QM CPU are combined in a heterogeneous multi-core architecture. The ARM architecture is renowned for its excellent performance, efficiency, and low power consumption. The four ARM Cortex-A72 cores handle demanding computing tasks, whilst the four ARM Cortex-A53 cores prioritize energy efficiency and handle lesser workloads. NXP i.MX8QM provides a high performance and powerful platform for facial recognition applications [8]. This processor has advanced image processing capabilities and enables face recognition algorithms to work quickly and effectively. Face recognition is the process of identifying a person by analyzing facial features. This process is quite complex and computationally intensive. However, NXP i.MX8QM performs face recognition algorithms quickly thanks to its high processor speed and parallel processing capabilities. This is a great advantage for real-time face recognition applications.

2.2. Embedded Linux

Embedded Linux is an operating system in which the Linux kernel and related software components are customized to meet the specific needs of embedded systems. The Linux kernel is used in embedded devices to communicate with hardware, manage processes, and control other system resources. It is generally used in embedded systems that have special hardware interfaces and have limited memory and processing power. Embedded

Linux benefits embedded system developers with a number of features [9]. Linux is an open-source operating system. This means that developers can access the source code and customize it according to their needs. It is also supported and constantly improved by a large open-source community. Its open-source nature also reduces licensing costs and allows easy integration into commonly used hardware and software components.

In this study, Embedded Linux was preferred as the operating system. The Embedded Linux operating system offers several advantages in facial recognition applications. Its powerful processing capabilities enable complex mathematical calculations to be performed effectively. Wide source code and tool support provides convenience to developers and enables rapid prototyping, testing and application development.

2.3. Yocto Project

The ability to create customized embedded Linux distributions is provided by the Yocto Project. This project aims to create a fully customized Linux-based operating system by combining components such as applications, drivers, toolsets, and the Linux kernel. Yocto Project enables users to configure and compile these components by taking a set of source files to create a customized distribution [10]. At its core, the Yocto Project is based on a build system generally referred to as OpenEmbedded. OpenEmbedded is a framework that brings software components together to create a complex distribution. This system provides package management and build tools, allowing developers to select and customize desired components [10].

The Yocto Project encompasses a range of tools and components, with the most notable ones being BitBake and OpenEmbedded Core. BitBake is a customized build tool for the Yocto Project. It compiles components using customized files called recipes, ultimately generating a customized distribution. On the other hand, OpenEmbedded Core includes a set of predefined recipes and configuration files, offering ready-made components for initial use [10].

In this study, a customized Embedded Linux distribution was created for the IMX8QM CPU using the Langdale version of the Yocto Project. The distribution was augmented with the Kaldi, TensorFlow, and OpenCV packages for artificial intelligence and speech recognition applications.

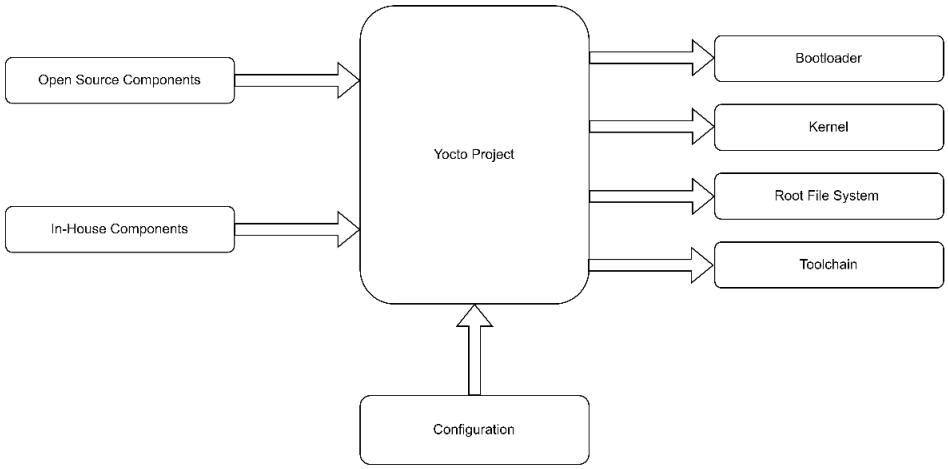

Figure 1: Yocto Project basic working diagram

2.4. OpenCV

OpenCV (Open-Source Computer Vision Library) is a powerful library frequently used in computer vision and machine vision applications. It can be used in harmony with Python, C++, Java, and many other programming languages [11]. OpenCV has functions such as uploading and saving image and video files, processing on image pixels. These functions include image transformations (resize, rotate, crop), image colour and contrast adjustment, image blur removal, edge detection, image segmentation [11]. OpenCV supports a few image processing related functions. It performs operations such as object detection, object tracking, face recognition, facial expression analysis, object recognition, image matching, point matching [11]. In addition, it offers the opportunity to perform pixel-level operations on the image. OpenCV library was used in this study.

2.5. WM8960 Sound Card

The WM8960 is an audio card solution produced by Wolfson Microelectronics (now known as Cirrus Logic). It is an integrated circuit commonly used in mobile devices, tablets, digital audio players, and other portable electronic devices. It offers functional and high-quality audio processing features. The WM8960 can play and record stereo audio with a resolution of 24-bit/192kHz [12]. It has a built-in headphone driver for high-quality audio output and includes an internal speaker amplifier. This allows for direct audio playback through headphones or speakers. Additionally, the WM8960 supports various audio processing features such as noise cancellation and equalization. It possesses important characteristics for portable devices, such as low power consumption and low voltage operation. This audio card solution has a user-friendly serial control interface and provides an extensive range of configuration and control options. It can be configured by software and supports various audio formats [12].

The WM8960 Audio Card provides an ideal solution for voice recognition processes by offering high-quality microphone input along with advanced audio processing features. This combination enables accurate and clear voice recordings and the utilization of effective voice recognition algorithms. The audio processing features of the WM8960 include noise cancellation, equalization, and other audio editing functions. This allows for the reduction of background noise, balancing of frequencies, and optimization of audio signals. These processes assist in generating more accurate and reliable results from voice recognition algorithms. Additionally, the WM8960 features a high-quality microphone input. The quality of the microphone input enhances the details and nuances of the audio recording while minimizing unwanted background noises, thereby increasing the accuracy of voice recognition processes. In voice recognition applications, the advanced audio processing features of the WM8960 optimize the recorded audio signals, while the microphone input ensures clear and detailed audio recording. This enables more effective and accurate results in areas such as voice recognition, natural language processing, or voice-based control. Due to these features, the WM8960 audio card is used in this study.

2.6. CNN

A Convolutional Neural Network (CNN) is a powerful deep learning technique that specializes in analyzing visual data, such as images or videos. It has become widely utilized in computer vision tasks like recognizing images, detecting objects, and segmenting images. Additionally, CNNs can also be applied to audio data, just like in this particular study where they were used to train a DNN-based model [13].

One of the remarkable capabilities of CNNs is their ability to automatically learn and extract relevant features from raw input data. This is made possible through a series of specialized layers, including convolutional layers, pooling layers, and fully connected layers [14].

Throughout the training process, CNNs acquire the best values for their parameters using a technique called backpropagation [15]. This iterative process involves adjusting the weights and biases of the network to minimize the discrepancy between the predicted output and the actual labels, also known as the ground truth.

In this study, the incoming speech data is converted into a spectrogram which is a format of the data as its frequency over time. After that, convolutional layers and pooling layers utilization are performed to extract significant features according to the spectrogram.

Hereby, the dimensionality is reduced and the essential information from the speech is captured. The extracted features become the base for making predictions on the fully connected layers and the Deep Neural Network based model is trained with these parameters. Finally, to facilitate the training process, backpropagation is utilized to calculate gradients and iteratively update the model based on these gradients. Thus, the DNN-based model is trained with the CNN technique for voice recognition.

2.7. DNN

A deep neural network (DNN) is a type of artificial neural network and an algorithmic method used by computers to perform complex tasks [16]. DNNs are inspired by the functioning principles of neurons in the human brain and have a structure with multiple layers [17]. These layers process input data to generate final outputs, guided by weights and biases that perform the calculations in this process. By employing a data-driven learning approach, DNNs have the ability to self-learn by analyzing large amounts of data. DNNs are commonly trained using deep learning algorithms, which promote data-driven learning and automatically adjust the weights. As a result, DNNs offer an effective solution for various complex tasks such as image and speech recognition, natural language processing, machine translation, and more [18]. The success of DNNs requires a substantial amount of impressive data, high computational power, and appropriate hyperparameter tuning.

In this study, a DNN-based model trained using the convolutional neural networks (CNN) deep learning technique is employed to perform speech recognition. Speech recognition is the process of converting human-generated speech into text and holds significant importance in numerous application domains. In this process, DNN-based models analyze and process speech signals to produce meaningful textual outputs. The speech recognition process consists of several fundamental steps. Firstly, a large dataset of speech is collected, which includes various speech samples from different speakers. Next, the speech signals are represented and feature extraction is performed in the time domain. Features are obtained by converting them into frequency components using mathematical transformations such as spectral features or mel-frequency cepstral coefficients. DNN-based models are trained using these features. During the training process, a large labeled dataset of speech is used, and weight adjustments are made using deep learning algorithms. This process typically requires high computational power and is conducted with large datasets.

2.8. Kaldi

Kaldi is an open-source framework or software package for the development and investigation of automatic speech recognition (ASR) systems [19]. It is the result of a project initiated by Douglas O'Shaughnessy and his colleagues at UCLA in 2011. This ASR framework provides a set of tools and libraries to implement complex speech recognition algorithms. Kaldi is written in the C++ programming language and utilizes various optimization techniques for fast and efficient computation [20]. This framework can be used to achieve high levels of accuracy by applying deep learning techniques using large datasets. Kaldi enables developers and researchers to configure ASR systems flexibly by providing extensive control. It offers a comprehensive guide that facilitates rapid prototyping of speech recognition algorithms. Kaldi supports tasks such as speech transcription, speech synthesis, and speech command recognition. The design of the framework is based on the principle of modularity. Kaldi is a framework capable of processing large amounts of speech data and supporting various modeling techniques. It serves as a comprehensive resource for both industrial applications and academic research. In this study, the Kaldi framework was employed to recognize the driver's voice.

3. Method

3.1. Rear-view Adjustment Based on Head Pose Orientation

Data collection is one of the first and most important steps of our methodology. The primary purpose here is to collect various images of the drivers we intend to recognize. Each of these images is tagged with a unique identifier (ID) corresponding to a particular driver. The images were collected under various lighting conditions and include different facial expressions and head orientations of the drivers.

The diversity in the dataset enables the model to recognize the driver under different scenarios (day, night, direct sunlight, shadows, etc.), even if the facial expression changes.

3.1.1. Image Preprocessing

The next step after data collection is preprocessing. This step is crucial to aligning the images with the successive phases of the project, which includes face detection, face recognition and head posture estimation.

The first stage of preprocessing involves the implementation of a face detection algorithm. This algorithm uses the Haar cascading classifier, which is a component of the OpenCV library. The classifier uses machine learning techniques to detect the presence of a face in an image by recognizing patterns unique to human faces.

Before applying the algorithm, the images were converted to grayscale format. This is because the Haar cascade classifier processes grayscale images more effectively. After the

transformation, the classifier evaluates the grayscale images and determines the coordinates of the bounding boxes where it detects a face.

After detecting faces, the algorithm performed a visibility check. This control is designed to ensure that the face is visible enough for later stages. The control was performed by examining the brightness levels inside the bounding box at which the face was detected. If an image is dimly lit or the face is not fully visible, it is excluded from further processing.

For real-time data capture from a camera recording a driver's face, the steps are the same but executed in a continuous loop. Each video frame is read, converted to grayscale, and then a face is detected. Brightness controls are applied and the area of the face is marked in the frame. The rendered frames are then broadcast in real time. If the brightness of a face is below a predefined threshold, that frame can be discarded or appropriate brightness enhancement techniques can be applied.

At the end of the preprocessing stage, a series of grayscale images or real-time video frames with clear, well-oriented and sufficiently bright faces are obtained. These are now ready for the next stages in the pipeline.

3.1.2. Driver's Face Recognition

After the preprocessing phase is complete, the next critical step is to identify the driver's face from the preprocessed images. In this study, we use the Local Binary Patterns Histograms (LBPH) technique to recognize faces. LBPH is a practical and efficient texture operator that captures the texture and spatial structure of an image. Calculation of local binary models (LBP) is the first step involves calculating the LBP representation of the grayscale image of the driver's face. The LBP feature serves as a texture identifier that includes local texture information, which is the basis of the LBPH face recognition process. The procedure compares each pixel in the image with its neighbors, using intensity values to create a new image that captures the structural patterns of the original image. The following step is splitting image into grids. After calculating the LBP image, it is divided into non-overlapping cells or grids. Generally, each cell is about 16x16 pixels. A histogram of the LBP codes within each cell was then calculated. These histograms provide a spatially coded representation of the driver's facial image. All histograms from all cells are combined to form a single spatially enhanced feature vector. This feature vector defines the original face image and is used in the face recognition task. At the end, driver's face recognition can be done by the feature vector of an unknown face (in this case the driver's) is compared with the feature vectors of known faces stored in the database. Comparison is made using various distance measures such as Euclidean distance or chi-square distance. The face recognition system identifies the unknown face as belonging to the person with the most similar feature vector in the database.

In the estimation step, the estimation function provides the label of the most similar face (or closest match) in the database. It also gives a confidence score showing the similarity between the face of the invisible driver and the known face in the database. The lower the confidence score, the higher the similarity between the faces.

3.1.3. Estimating Driver's Head Pose

After successful recognition of the driver's face, the next critical component of our system is head posture estimation. This step is important to adjust the rear-view mirror to the sitting position and natural head position of each driver.

The purpose of head posture estimation is to understand the orientation of the driver's face in real time, which gives us the direction the driver is facing. To achieve this, the Perspective-n-Point (PnP) problem-solving approach, a standard procedure in computer vision and photogrammetry, was used.

Landmark Detection: The first step involves detecting prominent facial landmarks. Corners of the eyes, tip of the nose, etc. such landmarks are identified using a pre-trained face landmark detector such as LBFModel from OpenCV.In the 3D model creation phase, a simplified 3D model of the head was created using these landmarks. This model is a representation of a typical human head that captures the basic geometric features. After modelling, the PnP Problem is solved with the 3D model of the head and the corresponding 2D points in the image, a PnP solver has been applied to predict the head posture. This solver gives us the relative orientation and position of the head by calculating the rotation and translation vectors that map the 3D points to their 2D counterparts.As last, Euler Angles are calculated by the rotation vector derived from the PnP solver is then used to calculate the Euler angles (yaw, pitch and roll). These angles ensured precise orientation of the head in 3D space.

However, once a driver is recognized from previous examples and the mirror adjustment preference is saved, the system skips the head pose estimation step. It directly adjusts the rear-view mirror according to the saved preferences corresponding to the recognized driver.

This combination of initial head posture estimation for new drivers and reuse of stored preferences for recognized drivers provides a tailored, comfortable and safe driving experience for all users.

3.2. Voice Identification for the Rear-view Adjustment System

The voice recognition process utilizes a DNN (Deep Neural Network) based model trained using Convolutional Neural Networks, a deep learning technique. This advanced method provides a system that can uniquely identify the driver by analyzing their voice

characteristics and automatically apply previously stored rear-view angle. An open-source voice recognition framework called Kaldi is used to recognize the driver's voice. This section provides a detailed description of the steps involved in data collection, data preprocessing, deep learning model training, and driver recognition. The collection and preparation of data, model training, real-time driver recognition, and automatic application of previously stored rear-view angle constitute the fundamental steps for this innovative system.

3.2.1. Data Collecting

The data collection phase is an important step in which the dataset is created that will be used to recognize the driver's voice. It is a key attribute for a successful voice recognition system. The data collection process consists of the following stages.

3.2.1.1. Sound Recording

Driver audio data to be used for training the Kaldi speech recognition model has been recorded. Real-time voice recordings of drivers were captured using a WM8960 sound card and an external microphone. During the recording of the audio, the sound data was captured in continuous real-time Pulse Code Modulation (PCM) format. PCM is a process that converts the audio signal from analog to digital. Consequently, the sound data was digitally recorded with high sampling rates. The sampling rate represents the number of samples per second in the audio and ensures accurate and detailed recording of the sound data.

Accurate and clear acquisition of the audio recording is of utmost importance for Kaldi. Kaldi aims to provide high accuracy and precision for speech recognition systems. Therefore, the quality of the audio recordings is crucial during the data collection process. The accurate and recorded sound data forms the foundation for creating a better speech recognition model. High-quality audio recording enables Kaldi to process the sound data more accurately.

3.2.2. Data Preprocessing

The audio data preprocessing stage includes several steps to convert the audio data into a suitable format for the Kaldi framework used to recognize the driver's voice and improve its quality.

3.2.2.1. Format Conversion of Audio Files

Audio data is represented in Waveform Audio File (WAV) format. Kaldi framework uses a special data format. Therefore, audio files need to be converted to a suitable format for Kaldi to process them. In this step, various operations are carried out to convert audio files to format and make them suitable for Kaldi.

The audio file in WAV format is read and loaded into memory. This is the basic step required to process audio data.

3.2.2.2. Sample Rate and Bit Depth Conversion

Sample rate conversion was performed to make the audio data compatible with Kaldi. The sample rate conversion enables the sampling rate of the audio data to be changed to the desired value.

The sampling rate has been considered for the development of a better speech recognition model. Higher sampling rates allow for more detailed capture of the sound and enable more precise analysis of the sound data. Thus, a sampling frequency of 16 kHz has been set. A 16 kHz sampling frequency indicates that 16,000 samples are taken from the audio signal per second. This frequency captures the most significant frequency components of human speech while omitting higher frequency components. In voice recognition systems, capturing the frequency range that is crucial for communication and intelligibility of human speech is important, and 16 kHz generally serves this purpose. Accurately recorded driver audio data has provided improved speech recognition performance for Kaldi.

Bit depth refers to the number of bits at which each sampling point of the audio data is represented. In this study, a 16-bit bit depth is used for the Kaldi framework. This means that each sampling point is represented by a 16-bit number. Different methods are used for bit-depth conversion. This conversion process is performed using an algorithm that minimizes data loss. In the study, the Linear Quantization algorithm was used for the Kaldi framework. Linear Quantization is a method that provides the best possible accuracy and quality when converting a signal to a lower bit depth. This conversion algorithm from Kaldi ensures the best possible quality and accuracy while representing the audio data with the correct bit depth. Thus, voice data can be processed effectively by Kaldi and driver's voice recognition can produce more accurate results.

3.2.2.3. Audio Data Conversion

In order to make the audio data compatible with Kaldi, the conversion step of the audio files was carried out by converting them to PCM (Pulse Code Modulation) format. PCM is a method for converting audio signals into digital form. This conversion provides the conversion of audio data from analog to digital.

PCM is a system in which audio signals are represented by sampling over time. The audio signal is sampled, and each sampling point is converted to a digital value. These digital values are stored as an array representing consecutive sampling points. In order to convert the audio signal to digital form, parameters such as sampling rate, bit depth, and number of channels are used in PCM format. The sampling rate indicates how many

times the audio signal is sampled. Bit depth refers to the number of bits in which each sampling point is represented. The number of channels indicates the number of simultaneously recorded audio channels. During PCM conversion, the audio file is decompressed and a PCM format is selected with properties such as sample rate, bit depth, and number of channels. Then, each sampling point of the audio file was converted to PCM format and assigned a digital value. This process is important so that the audio data can be processed correctly by Kaldi.

3.2.3. Feature Extraction from Data

Data feature extraction is a process performed to convert audio data into a more suitable representation. At this stage, various feature vectors are extracted from the audio samples. Feature extraction methods are used to capture the frequency and temporal components of sound and represent important features. In this study, the Mel-Frequency Cepstral Coefficients (MFCC) feature extraction method was used. In this section, the process of extracting data features for voice recognition using Kaldi is examined.

3.2.3.1. Mel-Frequency Cepstral Coefficients

The MFCC feature extraction process begins with the step in which the audio data is first sampled at an appropriate sampling rate and then passed through a filtering process called pre-emphasis. The driver audio data sampled at 16kHz was subjected to a filtering process called pre-emphasis. Pre-emphasis is used to emphasize the low-frequency components of the sound and improve signal quality.

The MFCC feature extraction process begins with the step in which the audio data is first sampled at an appropriate sampling rate and then passed through a filtering process called pre-emphasis. The driver audio data sampled at 16kHz was subjected to a filtering process called pre-emphasis. Pre-emphasis is used to emphasize the low-frequency components of the sound and to improve the signal quality. It is implemented as a first-order Finite Impulse Response (FIR) filter. The pre-emphasis process was performed with the output signal obtained by subtracting some of the previous samples of the input signal. Mathematically, the pre-emphasis process is expressed as shown in Equation-1.

$$\boldsymbol{y(n) = x(n) - \alpha * x(n-1)} \quad (1)$$

In Equation 1, x(n) refers to the input signal, y(n) refers to the pre-emphasis applied output audio signal, α refers to the pre-emphasis coefficient and n refers to the sampling index.

In the pre-emphasis process, the output signal is calculated by subtracting an amount from the previous sample of the input signal. This filtering process does not change the ratio of the frequency components but helps to highlight the low-frequency components

of the signal. High-frequency components in audio data usually carry more information and can be separated better. Thus, the pre-emphasis process is aimed to better emphasize the important features of the audio signal and to achieve a better recognition performance.

After the pre-emphasis processing, the audio data with better accentuated high-frequency components were subjected to windowing. Windowing is a method for temporally dividing and breaking audio data into smaller parts. It was implemented using a window size of 25 milliseconds. With this process, the audio signal is continuously analyzed over time and its changes over time are captured more precisely.

The windowing process is expressed mathematically as shown in Equation 2.

$$\boldsymbol{w(n) = h(n) * x(n)} \tag{2}$$

In Equation 2, x(n) denotes the sampled audio signal, w(n) the windowed audio signal, and h(n) the window function.

The windowing process takes a part of the sampled audio signal at a certain time and analyzes the frequency content of this part and its change over time. The window function (h(n)) is a weight function that determines how the windowing is applied. This function is used to emphasize or decrease a certain duration of the audio signal. The Hanning windowing function is used in this study. The Hanning window function has a cosinusoidal shape that is symmetrical and smooth. This function has a pulse-shaped peak and low values to reduce edge effects. During the windowing process, the Hanning function is applied to the signal sample that has low values relative to the window size. In this way, the edge effects of the signal are reduced, and better results are obtained in the analysis process. In the feature extraction step of the Kaldi framework, frequency analysis was performed on the parts obtained by using the Hanning window function. This process has been used to represent the frequency and temporal characteristics of sound in greater detail.

As a result, the windowing process temporally divides the audio data into smaller parts. Fast Fourier Transform (FFT) is used to perform the frequency analysis on these parts.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} \qquad k = 0, \ldots, N-1, \tag{3}$$

As shown in Equation 3, X(k) represents the spectrum of frequency components. Another parameter of the equation, w(n), represents the windowed audio signal. N represents the FFT dimension.

The FFT algorithm was used for the Kaldi framework to efficiently calculate the frequency content of each windowed segment. The FFT algorithm transforms the time domain representation of the windowed signal into the frequency domain and provides amplitude and phase information of the frequency components. By applying the FFT to the windowing process, the spectral information required for Kaldi is extracted and used for further processing and feature extraction. The resulting frequency spectrum represents the distribution of frequency components within each window and provides a more detailed analysis and characterization of the audio signal.

The frequency spectrum obtained after the FFT process is subjected to Mel-Scale conversion, and the frequency components are represented more harmoniously with the human hearing system. This transformation aims to express frequency components more closely to human perception and relates physical frequencies to perceived frequencies. This process converts the linear frequency scale to the Mel scale and provides a better representation of human perception. The Mel scale is designed to be more sensitive to lower frequencies and less sensitive to higher frequencies, reflecting the non-linear characteristics of human hearing.

The Mel-Scale transform is expressed mathematically as in Equation 4.

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{4}$$

In the formula, m represents the frequency value obtained by Mel-Scale conversion, and f is the frequency value. This formula converts the linear frequency value to the Mel scale with a logarithmic transformation.

Mel-scale conversion in Kaldi transforms the frequency spectrum from linear scale to Mel scale, making it compatible with human hearing characteristics. This conversion provides a better representation of frequency components and improves the performance of subsequent processing steps in speech recognition applications.

After the Mel-Scala transformation, the obtained frequency components are combined with temporal components with the cepstral transformation, providing a better representation of both the frequency and temporal characteristics of the sound.

In Kaldi, the cepstral transform is used to represent the temporal properties of the frequency components. This transform applies the Discrete Fourier Transform (DFT) process to get the amplitude spectrum of the frequency components and takes the logarithm of the resulting spectrum. Then, this logarithmic spectrum is transformed into temporal components by subjecting it to Inverse Discrete Fourier Transform (IDFT). Cepstral transform provides a more appropriate representation by combining the frequency and temporal properties of the sound. This helps the CNN deep learning

model used in Kaldi to achieve better voice recognition performance. Cepstral coefficients represent audio data compactly and capture important features of the audio associated with the speech content.

3.2.4. Model Training

Model Training is a phase in which a model is trained using a Convolutional Neural Networks (CNN)-based deep learning model, which can make a class prediction that describes the driver's voice by processing the audio data.

3.2.4.1. Deep Neural Network Based Model

The driver's voice recognition process was performed in Kaldi using a DNN-based model using the CNN deep learning technique. This model works on the tools and libraries that Kaldi provides. The model is designed to effectively analyze and classify temporal and frequency features in audio data. The model takes audio feature vectors as input. Audio data was preprocessed and converted into MFCC feature vectors using Kaldi's MFCC feature extraction interface. These feature vectors contain temporal and frequency representations of the audio signal. The DNN architecture represents the neural network architecture on which the model is configured. The model starts from the input layer and passes through the sequentially connected hidden layers. Each hidden layer processes features and learns high-level representations. The last layer generates the class predictions and defines the driver's voice. Model training was carried out using the training interface provided by Kaldi. The training process was carried out on the labeled training dataset. Initially, the weights of the model were randomly initialized and then the weights were updated using gradient-based optimization algorithms. This process was repeated to improve the performance of the model and to optimize the class predictions. All these processes are done using the Kaldi framework.

3.2.4.2. Training Data Subset

A subset of the original dataset is selected for training the model. This subset represents samples containing the driver's voice and allows the model to learn the correct classification. Kaldi offers users various tools to manage and customize this process. Selection of the training data subset is done through data lists of data files in Kaldi. Data lists are used to group data files according to criteria specified by the user. Figure 2 represents the voice identity recording flow of the study.

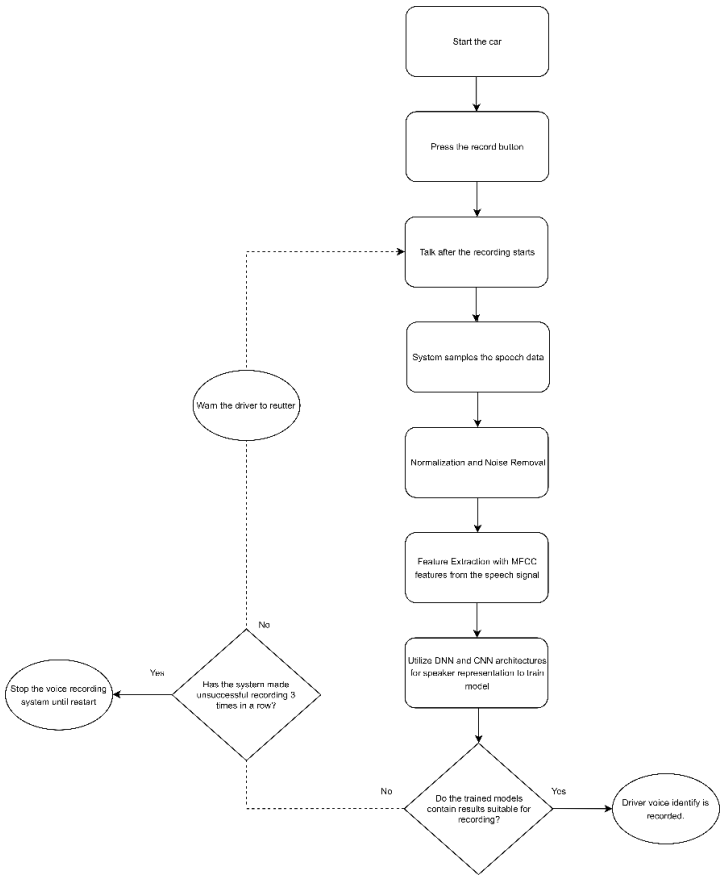

Figure 2: Voice recording and model training flowchart

3.2.4.3. Initialization of Model Weights

The weights of the model are randomly initialized before training. This allows the model to fit the data without any bias or predetermined pattern. Randomly initialized weights are then optimized and developed throughout the training process. In Kaldi, the weights of the model are usually initialized with random values drawn from a particular distribution. This random initialization prevents the model from getting stuck in suboptimal solutions or converging to the same output for different inputs. Throughout the training process, the model continues to learn from the data and adjusts its weights by minimizing its objective function and improving prediction performance. Stochastic gradient descent (SGD) algorithms are used in Kaldi for weight updates.

3.2.4.4. Gradient-Based Optimization Algorithm

Gradient-based optimization algorithms are used to update the weights of the model. These algorithms use gradient calculations to minimize the loss function and increase the performance of the model. Kaldi offers various tools and methods to manage the training of the model using gradient-based optimization algorithms.

In this study, the Stochastic Gradient Descent (SGD) optimization algorithm was used in Kaldi. This algorithm tries to minimize the loss function by updating the weight for each training sample. SGD selects a training sample at each step, calculates the gradient, and updates the weights. This process is repeated to bring the weights closer to the optimum values.

3.2.5. Model Optimization and Validation

Upon completion of the training process, the validation set is used to evaluate and optimize the model's performance. The validation dataset is a separate dataset to compare the model's predictions with the actual labels. This dataset is used to evaluate the model's performance metrics such as accuracy, precision, recall, and F1 score. Performance metrics are used to measure the classification capabilities and accuracy of the model. Model optimization is accomplished by adjusting the hyperparameters of the model based on performance metrics and experimenting with different model variations. Kaldi provides various tools and optimization algorithms for hyperparameter tuning. In this way, the performance and classification capabilities of the model are optimized. The validation dataset and performance metrics allow us to objectively evaluate the performance of the model and provide a roadmap for model improvement. In this process, the tools and metrics offered by Kaldi help users analyze and improve the performance of the model.

The DNN-based model trained with the optimized CNN deep learning technique used and the voice recognition process performed with the Kaldi sound processing library provides high accuracy and reliability. The driver's voice is recognized and accurately labeled by feature extraction and model classification. This method offers a powerful voice recognition solution that can be used in real-time applications.

After the processes described in this section are completed, the sound of the driver is recognized. The driver's voice is compared with the sound in the system and it conveys to the customizable rear-view system that the driver has been successfully recognized. Systems. Figure 3 presents the flow of the voice identity recognition and verification part of the study. Also, the basic block diagram of the system is shown in Figure 4.

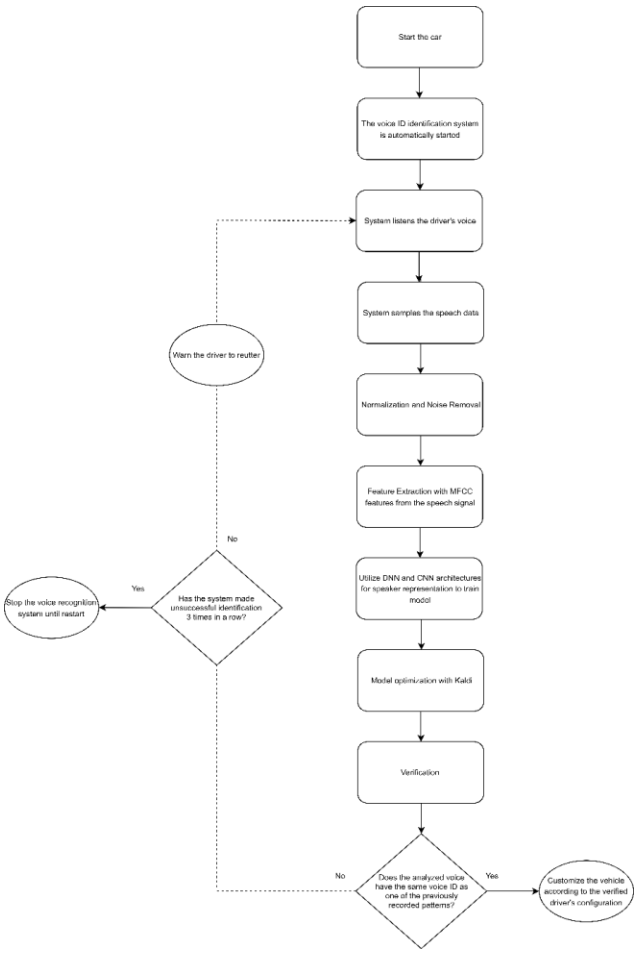

Figure 3: Voice ID recognition flowchart

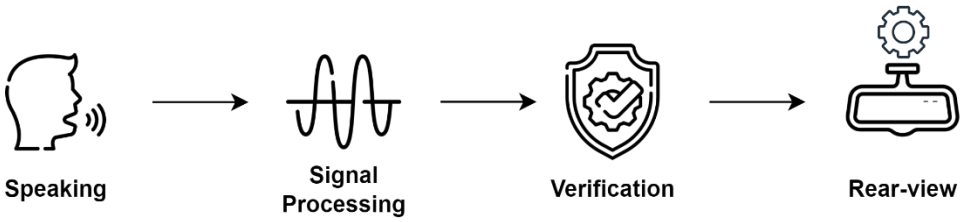

Figure 4: Voice recognition system basic block diagram

4. Result

Facial recognition systems are highly used and advanced in today's more personalized car cockpits. On the other hand, this study provided an innovative perspective on the concept of face recognition in automobiles by identifying the driver's face and making an adjustment.

Thanks to a camera integrated into the vehicle cockpit, the face of the driver, which was controlled throughout the trip, was periodically checked for position and orientation.

These values calculated using the Perspective-n-Point (PnP) algorithm were processed, and the mentioned rear-view mirror was instantly turned to the most correct angle.

Below are some sample images obtained from different angles, in which the driver's face is modeled with the help of artificial intelligence, by courtesy of the Perspective-n-Point algorithm:

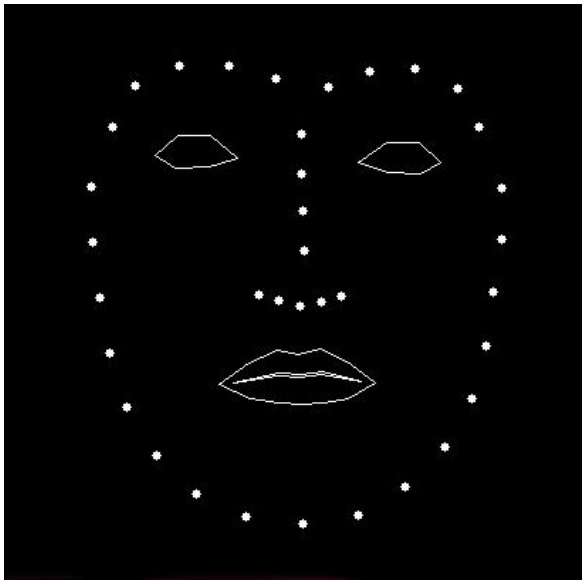

Figure 5: Example face modelling with PnP algorithm

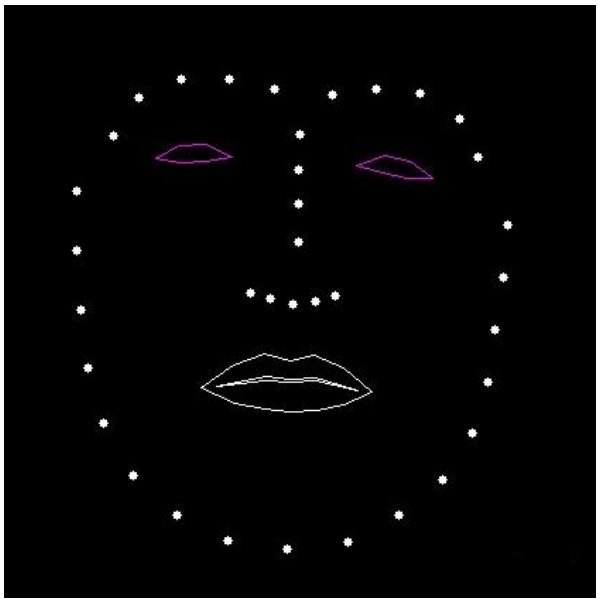

Figure 6: Example face modeling with PnP algorithm with closed eyes

5. Discussion and Conclusion

As a conclusion of this study, a new progress has been made to the increasing personalization and comfort improvement efforts in cars and a rear-view mirror structure that is dynamically adjusted according to the head posture direction has been designed. Adjustment based on this head posture direction has made driving safety more effective as it provides the most correct viewing angle for that driver. In addition, the fact that different drivers driving the same vehicle, or the same driver does not need any manual correction after the initial position changes while driving, aims to increase comfort by reducing time loss.

Besides face recognition, voice control systems in automobiles are a system that has been studied for years and is widely used in modern vehicles. Voice control systems, which have become a feature used by many automobile manufacturers, allow drivers to perform operations such as navigation, phone calls, and playing media content in general. Research by the Capgemini Research Institute predicted that 73% of drivers will use an in-car voice assistant in 2022, and this rate could increase to 85% in the next 3 years [21]. This artificial intelligence and neural network-supported DNN-based voice recognition system has contributed to the development of these voice control systems by analyzing the identity of the driver and providing the vehicle configurations preferred by the driver before and during the ride. Although there are many studies in the literature such as headlight beam control with voice command, and better perception of the driver's voice in the vehicle, this study, which makes the in-car configuration by analyzing the driver's voice, provided an innovative perspective on these systems [22][23].

6. Acknowledge

We would like to thank Software Architect Huseyin Karacali for his mentoring and motivating influence. We would also like to thank TTTech Auto Turkey for their valuable assistance throughout the development of the project.

References

[1] M. Vamsi, K. P. Soman and K. Guruvayurappan, "Automatic Seat Adjustment using Face Recognition," 2020 International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 2020, pp. 449-453, doi: 10.1109/ICICT48043.2020.9112538.

[2] Lo, Ei-Wen (Victor) & Green, Paul. (2013). Development and Evaluation of Automotive Speech Interfaces: Useful Information from the Human Factors and the Related Literature. International Journal of Vehicular Technology. 2013. 10.1155/2013/924170.

[3] W. Astuti, S. Tan, M. Solihin, R. Vincent, and B. Michael, “Automatic Voice-Based Recognition For Automotive Headlights Beam Control”, Int. J. Automot. Mech. Eng., vol. 18, no. 1, pp. 8454 –, Mar. 2021.

[4] M. Hasenjäger and H. Wersing, "Personalization in advanced driver assistance systems and autonomous vehicles: A review," 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), Yokohama, Japan, 2017, pp. 1-7, doi: 10.1109/ITSC.2017.8317803.

[5] Lee, J. M., & Ju, D. Y. (2018). Classification of Human-Vehicle Interaction: User Perspectives on Design. In Social Behavior and Personality: an international journal (Vol. 46, Issue 7, pp. 1057–1070). Scientific Journal Publishers Ltd. https://doi.org/10.2224/sbp.6242

[6] J. Martindale, "What is a CPU? here's everything you need to know," Digital Trends, https://www.digitaltrends.com/computing/what-is-a-cpu.

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

[8] "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.

[9] A. Klinger, "Embedded linux – kernel, Aufbau, toolchain," Embedded Software Engineering - Fachwissen, https://www.embedded-software-engineering.de/embedded-linux-kernel-aufbau-toolchain-a-99d15279522f4d1fcd8b2d852a8f771b/.

[10] "Software," Yocto Project, https://www.yoctoproject.org/software-overview/.

[11] "About - OpenCV." https://opencv.org/about/.

[12] "WM8960." [Online]. Available: http://www.sunnyqi.com/upLoadproduct/month_1306/ WM8960.pdf.

[13] "A comprehensive guide to Convolutional Neural Networks - the eli5 way," Saturn Cloud Blog, https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/.

[14] "What are convolutional neural networks?," IBM, https://www.ibm.com/topics/ convolutional-neural-networks.

[15] "What Is a Convolutional Neural Network? - MATLAB & Simulink." https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html.

[16] "Deep Neural Network - an overview | ScienceDirect Topics." https://www.sciencedirect.com/topics/computer-science/deep-neural-network.

[17] "Deep Neural Network: What is it and how is it working?" https://datascientest.com/en/deep-neural-network-what-is-it-and-how-is-it-working.

[18] "What is Deep Learning? | IBM." https://www.ibm.com/topics/deep-learning.

[19] "About the Kaldi project - Kaldi." https://kaldi-asr.org/doc/about.html.

[20] "Kaldi Tutorial." http://eleanorchodroff.com/tutorial/kaldi/.

[21] Markus Winkler, Rainer Mehl, Jerome Buvat, Ramya Krishma Puttur, Gaurav Aggarwal, Hiral Shah, *Voice on the go*. Capgemini Research Institute, 2019

[22] W. Astuti, S. Tan, M. Solihin, R. Vincent, and B. Michael, "Automatic Voice-Based Recognition For Automotive Headlights Beam Control", Int. J. Automot. Mech. Eng., vol. 18, no. 1, pp. 8454 –, Mar. 2021.

[23] Whittington, Jim, Ye, Hua, Kamalakannan, K, Vu, Ngoc-Vinh, Mason, Michael, Kleinschmidt, Tristan, & Sridharan, Sridha (2010) Low-cost hardware speech enhancement for improved speech

recognition in automotive environments. In Doyle, N (Ed.) The 24th ARRB Conference Proceedings. ARRB Group Ltd., CD Rom, pp. 1-17.