A speech recognition approach for an industrial training station

Valentin-Cătălin Govoreanu[1,*], *Adrian-Nicolae Țocu*[1], *Alin-Marius Cruceat*[1], and *Dragoș* Circa[1]

[1]Lucian Blaga University of Sibiu, 10 Victoriei Boulevard, Sibiu, Romania

Abstract. This paper presents a speech recognition service used in the context of commanding and guiding the activities around an industrial training station. The entire concept is built on a decentralized microservice architecture and one of the many hardware and software components is the speech recognition engine. This engine grants users the possibility to interact seamlessly with other components in order to ensure a gradual and productive learning process. By working with different API's for both English and Romanian languages, the presented approach manages to obtain good speech recognition for defining task phrases aiding the training procedure and to reduce the recognition required time by almost 50%.

1 Introduction

Speech recognition (SR) or sometimes referred to as Automatic Speech Recognition (ASR) is one of the most used such forms of alternative interaction methods, others are by using physical controls such as buttons. Speech is the most natural way for humans to interact with each other and an appealing approach for introducing the uninitiated users to interact with machines. It is also an efficient way to interact with computers, especially when the users have their hands occupied while doing a task. The goal was to extend our training station capabilities by creating a recognition system for continuous speech that could understand a predefined set of commands, such as "Open first application", "I need help" or "Stop the training", and execute specific instructions. The system would be able to understand both the Romanian and the English language, and it would also return the gender of the user based on speech.

1.1 Methodology

The purpose of this article was to demonstrate the usability of a speech recognition service and test its usefulness as an interaction method with a training system, for increasing the productivity of the trainee. In this respect, a quantitative method (see Fig. 1) was chosen to approach our thesis. This method was chosen because the experiment participants were workers in the industry domain and this approach can help to improve the experience and skills of the potential workers in the manufacturing industry.

* Corresponding author: valentin.govoreanu@ulbsibiu.ro

The speech recognition system has been designed by using the C# language, which was used for integrating the Google Speech-to-Text API [1]. We chose to use Google Speech-to-Text API because it supports a wide variety of languages and dialects: 120 dialects of 65 languages. Another advantage of using this API is that Google is including their research findings in their consumer products. An example of this would be the WaveNet [2], which is a deep neural network for generating raw audio from text and is currently included in the Google Text-to-Speech API [3]. Also, we have used a Microsoft speech recognition library for additional tasks. Moreover, the Python language was used because it permits a simple implementation of the algorithm chosen for gender recognition. For testing, a separate virtual reality application has been developed in Unreal Engine. The tests were designed for the Romanian and English languages.

As testers, people who have both Romanian and English language knowledge were selected. A group of 15 people with ages between 18 and 30 years old was selected. It took an average of 4-5 minutes for each user to test the application. Each sentence consists of an average of 4 words. For testing, each of them had to speak ten sentences for each language.

The system accuracy was measured by counting the number of unrecognized words and the recognition time for each sentence. The results would be stored in a CSV file format for later data analysis and interpretation.

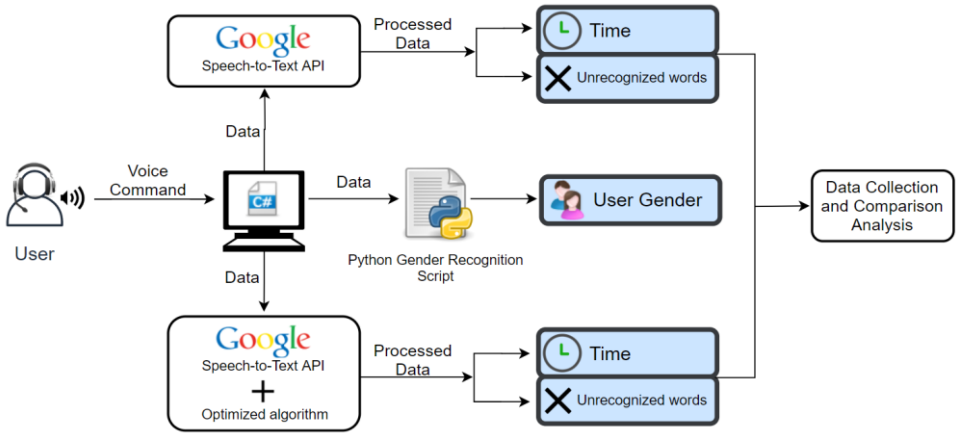

Fig. 1. Overview of the research methodology.

1.2 System Overview

The training station [4] consists of a table with adjustable height, provided with depth cameras and sensors for continuously tracking an assisted assembling operation. The table automatically adjusts its height which reduces the number of scenarios where the user will hold a harmful posture during his training, resulting in efficiency from both ergonomic but also an operational point of view.

The architecture of the training (see Fig. 2) consists of a series of decentralized microservices [5] communicating between themselves, a structure that allows the enabling or disabling of different units and without affecting the whole application. These microservices range from simple notifications microservices to more complex microservices that allow the user to complete the same training in the virtual reality environment.

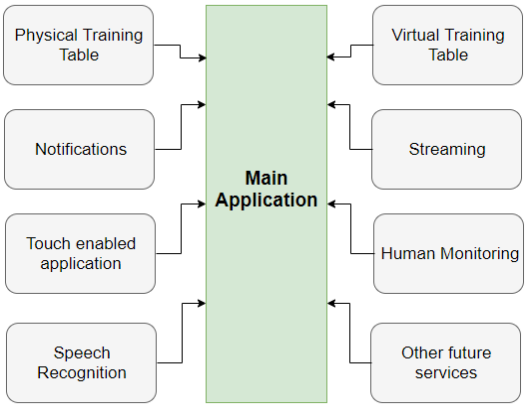

Fig. 2. Service-Oriented Architecture - Training Station.

The training station facilitates step-by-step user assistance procedures with simple instructions for assembling a modular tablet (current experiment). This tablet is customizable, meaning that the user can choose between three different modules (flashlight, power bank, Bluetooth speaker) in order to fit his needs. More details about the modular tablet can be found in a previous article [6]. Incorporating the SR module in the training station, not only is a method that enables a user-friendly approach for the interaction between the training station and the trainee but also represents a step forward in creating an Intelligent Manufacturing System (IMS) [7]. In the assembling process, the user's hands are occupied most of the time. By enabling the SR capability, the user could interact with the training station even with the hands occupied. Based on the assembling process the user could ask for help and get details about specific components in audio or non-audio ways, thus providing valuable guidance for the user and improving the training efficiency.

Industrial training is an important strategy for improving trainee performance. When learning a new process, the trainee's greatest need is to understand the instructions and if the user meets a language barrier limitation, his training process will be slowed. By approaching an SR system for not only English but also Romanian, the language barrier can be overcome, at the same time leaving the addition of other languages as a possibility for the future.

2 Development of Speech Recognition Microservice

SR has been researched for almost 50 years [8] and it is one of the most difficult areas in computer science because it requires very complex linguistics, mathematics, and tremendous computing power. Speech recognition has improved a lot over the years, due to the great advances in computer technology, and right now there are many great recognition systems available for use, either open-source or closed-source. A good speech recognition system is characterized by two major factors, speed and accuracy and open-source recognition systems tend to excel at both of them [9]. Unfortunately, few of these systems are dedicated to the Romanian language.

The SR microservice was implemented in C# using mainly the Google Speech-to-Text API, but also by using the System.Speech.Recognition library [10] provided by Microsoft.

The library provided by Microsoft doesn't support the Romanian language, but it is able to distinguish between a set of predefined words and it was used for the language selection feature only. On startup, the microservice waits for either of the following commands to be spoken by the user: "English" or "Română". Depending on the recognized command the microservice switches to that language and the Google Speech-to-Text API will not recognize commands in other languages for its lifetime other than that. The ability to support more than 120 languages and variants, including the Romanian language was one of the main reasons for choosing Google Speech-to-Text API. Also, this system proved its superiority when compared to other popular SR systems such as Microsoft API (SAPI) and CMU Sphynx, because of its low word error rate [11].

After the microservice starts, and the language has been selected, the main SR application (see Fig. 3) looks for a microphone, and whenever a specified peak in volume is detected, we start recording audio and set a timer to call us back after one second so we can stop recording. The audio recorded in the specified interval of one second is added to an internal wave buffer for later use. When there is enough audio in the internal wave buffer, a request with the required configuration needs to be written before sending the audio to Google Speech-to-Text API. The configuration is set based on the language selected and has multiple speech recognition models, so you can choose the optimal model for your use case. For example, if the audio comes from a phone call there is a *phone_call* model available. For the training station, the *command_and_search* model was selected, because its use it's intended for short queries such as voice commands. Also, based on the language selected we set a list of possible words because the configuration system supports customization in the form of providing the list of possible words to be recognized which is useful in scenarios where the list of possible words is limited.

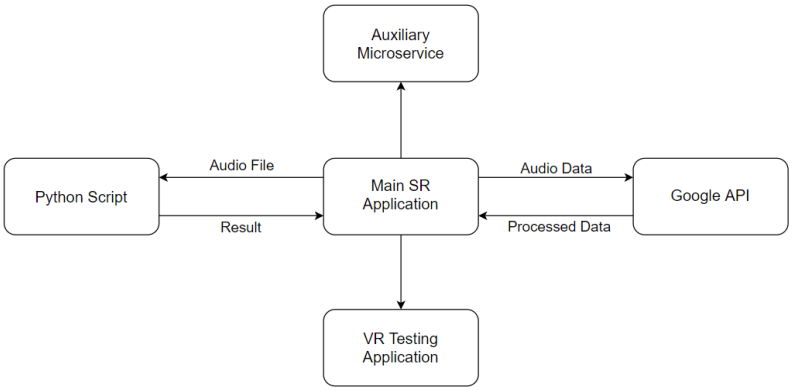

Fig. 3. General application architecture for speech recognition and gender identification.

When the required configuration is set and there is enough audio data stored, a streaming request is sent to Google API and the responses from the server are processed right away by the main SR application. However, the responses from the API were not gratifying, especially for the Romanian language. For solving this problem, we took advantage of the limited number of speech commands, and we guided the system to look for particular keywords in the responses in order to recognize the entire commands.

2.1 Registering Speech Commands and their Actions

While the main SR application records and understands the speech commands there is a need to actually use the recognized commands, for example triggering an event on a separate

microservice. Also, all modules of the training must have a way to see what speech commands have been executed and to register to only certain commands.

These demands are met by using a separate/auxiliary microservice (see Fig. 3) that checks the recognized command against the SQLite database to find if the command is valid. In the case of a match, it logs the command and then calls any other microservices and applications that are associated with the valid command.

When a microservice registers for a command its IP, port, endpoint, and requested command are stored in a separate table within the database. When a command is recognized the table is queried for any registered microservices and are being called one by one.

To add new commands the "addCommand" API call must be used with valid parameters. The following table lists all the API calls available for the service and their uses:

Table 1. Auxiliary microservice API.

HTTP Method	API Call	Description
POST	/addCommand/:command/:description	Registers a new valid command. If it exists it returns an error
POST	/phraseRecognized/:command	Notifies the service that a command has been recognized
PATCH	/phraseRecognized/:command	Notifies the service that a command has been recognized incorrectly (similar sounding words)
GET	/commands	Get a list of all the valid commands and their respective description
GET	/commands/:id	Get the command phrase and the description of the command with passed id
PUT	/registerCallback/:command/:endpoint	Register a callback for the microservice that called this method. IP and port will be taken from the actual request
DELETE	/unregisterCallback/:command	Unregister from being called when the command is recognized
GET	/getLastCommands	Get the last recognized command
GET	/getLastCommands/:n	Get last "n" recognized commands

2.2 Gender Identification in Speech Recognition

The acoustic properties of the voice and speech can be used to identify a speaker's gender [12], and there are many possible ways [13] for detecting gender based on speech. One way is to analyze the user's pitch. Pitch is a very important feature that can be obtained from a segment of speech. It is the property of a sound, determined by the frequency of the waves producing it. Based on pitch, we can classify the genders, due to the fact that the average

fundamental frequency/pitch period for men is typically in the range of 85-185 Hz, whereas for women it is 165-200 Hz, and these intervals could be adjusted based on your needs [14]. For gender recognition, a Python (PI) script was used, where we implemented a pitch detection algorithm (PDA). A PDA is an algorithm designed to estimate the pitch or fundamental frequency. Also, there are several methods for pitch detection, but we used the Harmonic Product Spectrum algorithm (HPS) (See Fig. 4) because of its successful success ratio [15].

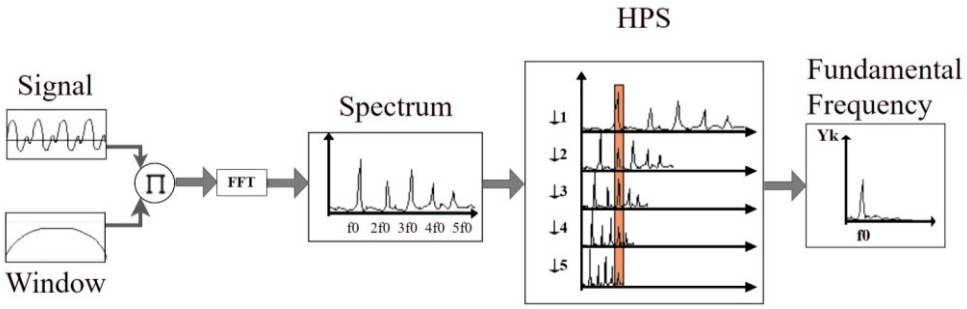

Fig. 4. Overview of the HPS algorithm.

The HPS algorithm is based on the fact that each speech signal consists of a series of spectrum peaks, and the harmonics are being multiples of the fundamental frequency (f0). The Harmonic Product Spectrum is defined such as:

$$HPS(k) = \left(\prod_{n-1}^{N} Y(n\,k) \right)^{\frac{1}{N}} \tag{1}$$

In equation (1), N corresponds to the number of harmonics that are considered, Y represents the magnitude spectrum of the frequency and k is the k-th indexed frequency bin.

The PDA algorithm would receive an audio file generated by the main SR application. In order to apply the HPS method on the data received, the input signal is divided by applying a Hanning window and the Fast Fourier Transform is utilized for converting the input signal from the time domain to frequency domain. Once the signal is in the frequency domain, the spectrum is compressed or downsampled a number of times, this causes the upper harmonics to line up with the fundamental frequency. When the various downsampled spectrums are multiplied together, the pitch can be extracted from the resulting spectrum by finding its maximum. After the Python script processes the received data, the result is sent back to the main SR application.

For testing the accuracy of the algorithm, we tested it on 32 speakers, 16 female and 16 male. The algorithm correctly recognized the speaker's gender with an accuracy rate of 87.5 percent calculated with the following equation:

$$Recognition\ Accuracy\ (\%) = \frac{No.of\ accurately\ recognized\ gender\ x\ 100}{No.of\ correct\ gender\ expected} \tag{2}$$

3 Testing the Microservice

In order to get more accurate data and a clear level of improvement, a comparison was made for each language before and after optimization. A separate testing application that communicates with the SR microservice has been developed in Unreal Engine [16] and for a more natural interaction, we implemented it to work in Virtual Reality (see Fig. 5). To get

the data, a group of 15 people with ages ranging from 18 to 30 years old was selected, and each of them had to speak 10 sentences/commands in both languages. All of them were native Romanian speakers with different levels of English language proficiency. When gathering data, we took into account the time for each command to be recognized, and the number of words unrecognized per sentence.

The results were recorded in a simple comma-delimited-values (CSV) file which was used to generate graphs.

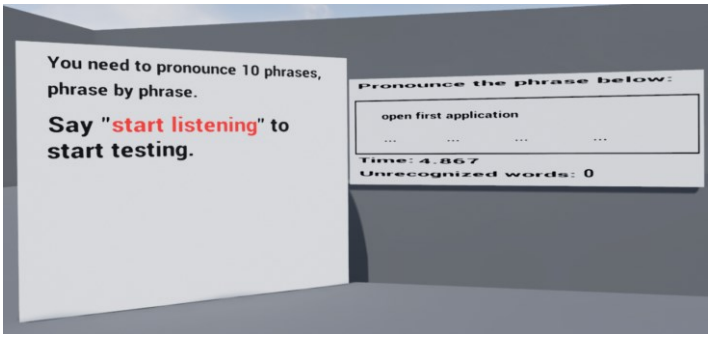

Fig. 5. The testing environment in VR.

4 Results

As seen in Fig. 6, the recognition times for Romanian Language were almost two times faster after optimization, and the average number of unrecognized words has significantly been reduced in both languages (see Fig. 7 and Fig. 9). As for the English language, the recognition times were not substantially improved (see Fig. 8). This is due to the fact that the recognition times were already quite reasonable before optimization, since the English language is one of the most researched and vetted languages in machine learning. Overall, these results are good considering that the Romanian language is categorized as an under-resourced language and thus there are very few available systems that support the Romanian language [17].

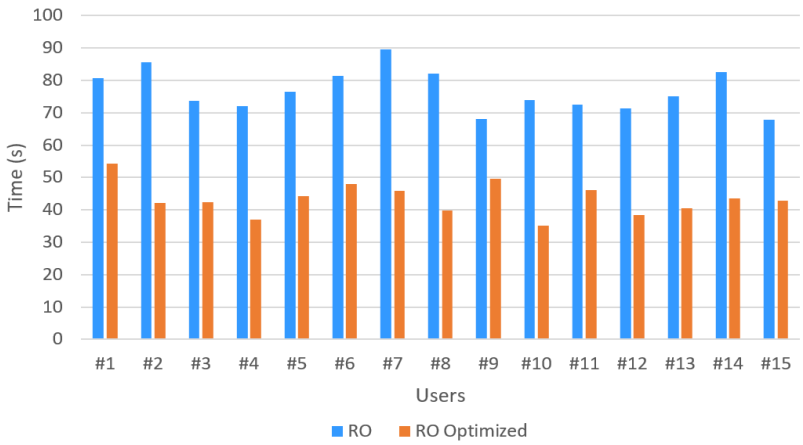

Fig. 6. Recognition times in seconds for the Romanian language, before and after optimization.

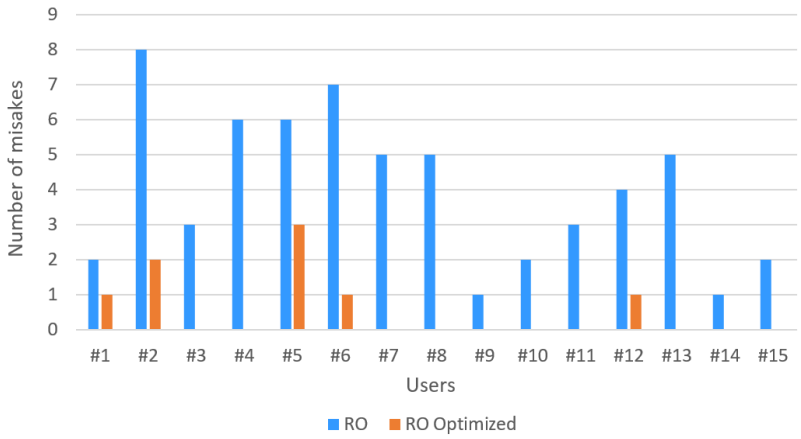

Fig. 7. The number of unrecognized words for the Romanian language, before and after optimization.

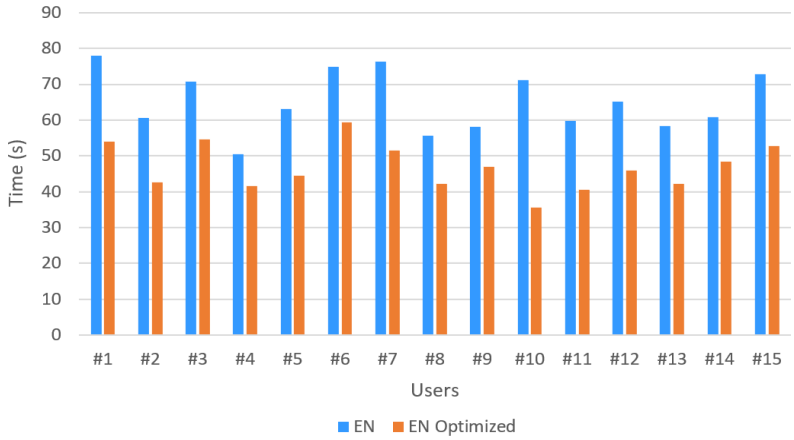

Fig. 8. Recognition times in seconds for the English language, before and after optimization.

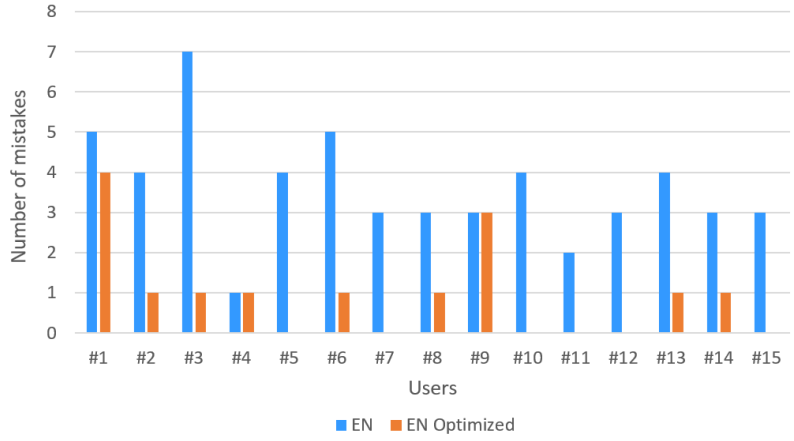

Fig. 9. The number of unrecognized words for the English language, before and after optimization.

5 Conclusion

Industry 4.0 represents the next stage in industry manufacturing. Intelligent manufacturing systems are playing a significant role in this new promising industry. Based on the results it can be concluded that SR is an effective interaction method and a speech recognition microservice using mainly the Google Cloud Speech-to-Text API can be successfully integrated into a manufacturing system where the number of commands is limited. Last, but not least, we have gender identification as a bonus feature which brings an improvement to the working environment by its high recognition accuracy. Gender identification can be used as a parameter for the adaptivity of the training table. For example, the voice instructions can be male or female or the environment can change its appearance, thus making the training more appealing to the user.

Initial tests had a great impact on developing this application because without optimization the recognition times were almost 50% lower. This issue could cost reaction time, efficiency and it could affect the user experience.

Based on the performed experiments users seemed to have liked the idea of a speech-controlled system because it's the most natural way of interaction with a machine and by providing such a feature the user's engagement can be increased. Most of the users prefer to say a command rather than pressing a button or making another action. One important thing is that the users prefer to stay focused on what happens in front of their eyes. Hence, by using voice recognition, their concentration can be relatively constant. This in turn increases productivity.

This work is supported through the DiFiCIL project (contract no. 69/08.09.2016, ID P_37_771, web: http://dificil.grants.ulbsibiu.ro), co-funded by ERDF through the Competitiveness Operational Programme 2014-2020.

References

1. Google Speech-to-Text API Homepage, https://cloud.google.com/speech-to-text, last accessed 2019/07/25.
2. A. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A.Senior, K. Kvukcuoglu, SSW, (2016).
3. Google Text-to-Speech API Homepage, https://cloud.google.com/text-to-speech/, last accessed 2019/10/15.
4. A. M. Cruceat, A. Matei, B. C. Pîrvu and A. Butean, International Journal of User-System Interaction, **12(1),** 54–66 (2019).
5. S. Newman, O'Reilly Media, **1,** (2015).
6. S. Stanciu, R. Petruse, B. Pîrvu, ACTA, **70(1)**, 36–42 (2018).
7. M. Gregor, T. Michulek, Applied Computer Science Book, 55–68 (2009).
8. S. Furui, Journal of The Acoustical Society of America, **116(4)**, 64–74 (2004).
9. R. Matarneh, S. Maksymova, V.V. Lyaschenko, N.V. Belova, IOSR J. Comput. Eng., **19(5)**, 71–79 (2017).
10. Microsoft System.Speech.Recognition Homepage, https://docs.microsoft.com/en-us/dotnet/api/system.speech.recognition?view=netframewor k-4.8, last accessed 2019/07/05.
11. V. Këpuska, G. Bohouta, International Journal of Engineering Research and Application, **7(3)**, 20–24 (2017).
12. M. Buyukyilmaz, A.O. Cibikdiken, MSOTA2016, 409–411 (2016).
13. A. Raahul, R. Sapthagiri, K. Pankaj, V. Vijayarajan, IOP Conf. Series: Materials Science and Engineering, **263(4)**, (2017).

14. D. Kaushik, N. Jain, A. Majumdar, 8th International conference on advanced computing and communication technologies, (2014).
15. C. Kadi, S. Gökhüseyin, Y.D.U. Ariöz, 23rd Signal Processing and Communications Applications Conference (SIU), 132–135 (2015).
16. Unreal Engine Homepage, https://www.unrealengine.com/en-US, last accessed 2019/04/12.
17. B. Iancu, Informatica Economica, **23(1)**, (2019).