Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 55-4 (2022) 133–138

Multi-modal communication system for mobile robot

Jan Švec * Petr Neduchal * Marek Hrúz *

* *Faculty of Applied Sciences*
New Technologies for the Information Society
University of West Bohemia
Univerzitní 8, Plzeň, 306 14
(e-mail: honzas@kky.zcu.cz, neduchal@kky.zcu.cz, mhruz@kky.zcu.cz)

Abstract: This paper presents a cloud-based multi-modal human-robot interaction architecture. The architecture consists of parts running on the edge (mobile) platform and cluster services. Software on the mobile platform is represented primarily by a Robot Operating System, an MQTT broker and the client side of the SpeechCloud architecture, which provides the communication with the computing cluster. Its detailed description is a part of the paper. The main contribution of the paper is a design of an architecture based on standard frameworks, protocols, and validated JSON messages. The paper also focuses on the description of a tested mobile platform equipped with an NVIDIA Jetson family computer and information about tasks that the robot will be able to perform using the proposed architecture.

Keywords: Multi-modal interaction, Networked robotic systems, Mobile robots, Perception and sensing, Intelligent robotics

1. INTRODUCTION

Multi-modal communication is an essential part of the human-robot interaction system. It enables a robot to communicate with a human in a human-friendly way. Moreover, it is an important factor of Industry 4.0, where cooperation between robots and humans is one of the key concepts.

The robot capable of multi-modal communication can perform communication based on a fusion of multiple modalities such as speech, gestures, text, face, and action recognition. In more detail, the robot should use the following human-machine interaction technologies: speech recognition, text to speech synthesis, gesture recognition, or sign language recognition. Moreover, face recognition could make the robot more social allowing it to recognize the identity of an user and act more familiar for her/him.

The above mentioned modalities should be divided into two groups based on the direction of communication. The first one contains all approaches for enabling communication from human to robot. The second one contains all parts for communication from robot to human. It is worth mentioning that bi-directional communication is possible only if the robot can perform at least one approach from each group.

A research field focused on developing social robots capable of multi-modal communication has been active for at least the last two decades. The work of Gorostiza et al. (2006) represents the research at the beginning of the mentioned period. The research is focused on developing all aspects of a social robot called Maggie. Maggie is a human-like personal robot equipped with several sensors, including a camera, microphone, tactile sensors, laser scanners, etc. The authors focus on the description of information flux between humans and the robot and on the social skills that the robot should acquire.

The authors of paper Budkov et al. (2010) held similar research. They proposed a human-robot interface for the mobile information robot, which had two parts. The first is a mobile platform, and the second is an information desk that secures communication with humans. A disadvantage of this approach is that the information desk is 1500 mm high and weights 50 kg. As will be described later in the paper, our approach is focused on the mobile platform and cloud computing instead.

The authors of paper Lucignano et al. (2013) proposed a dialog system for multimodal human-robot interaction based on the Partially Observable Markov Decision Process (POMDP). The novel part of the paper is the utilization of POMDP in multimodal human-robot interaction instead of spoken dialog system only. In particular, they used gesture and speech recognition in their system.

Similar research was proposed in paper Yongda et al. (2018). The proposed approach is based on Microsoft Speech SDK and on a Leap motion sensor to obtain audio-visual data. The novelty of the paper is the approach of direction estimation using Interval Kalman Filter to reduce the noise of the leap motion sensor. In the experiments, the authors compare the proposed method with previous research and prove its higher precision in hand tracking and gesture recognition tasks.

The paper of Mizuchi and Inamura (2017) addresses the problem of cloud-based human-robot interaction. For that purpose, they use Robot Operating System (ROS) (Garage (2012)) for robot control.

The goal of this paper is to present and describe our multi-modal human-robot interaction architecture based on the ROS, the MQTT protocol and the SpeechCloud client on the edge robot-side and the SpeechCloud cluster on the powerful server-side. The SpeechCloud is a system for communication between a client – i.e., the mobile robot – and a server cluster. Moreover, it is possible to run multiple actions such as speech or face recognition, as described later in the paper. The main contribution is the architecture that enables the mobile robot to perform human-robot interaction with maximum efficiency in resources usage.

2. MOTIVATION

The main motivation is to develop a cloud based system that will be able to run speech processing and computer vision algorithms easily on either the edge device or the powerful server. The design should maximize efficiency in resources utilization on both the edge device and the shared cluster resources and at the same time it should efficiently use the wireless network bandwidth to provide real-time response.

In the following paragraphs, computer vision (Sec. 2.1) and speech technologies (Sec 2.2) that are intended for use in the system will be described. Especially speech technologies will be addressed in detail because the technologies stand as a conversion from speech to semantics and semantics to speech as used in the dialog manager (Sec. 3). Then, the proposed multi-modal architecture (Sec. 4) is described as a super set of the speech only architecture (Sec. 4.1). We also provide the details about implementation and possible future work (Sec. 6). The last section concludes the paper (Sec. 7).

2.1 Computer vision

The main computer vision task – that we address in our current research – is face recognition using a Deep Neural Network (DNN). This task consists of several parts: face detection, face description, and face identification. The most recent works on the topic of face detection include Lin et al. (2017), Zhu et al. (2020), Guo et al. (2021). These systems detect the regions containing faces in the image, which are then geometrically aligned to a default frontal pose and fed into the system of face description. The work of Deng et al. (2019) is a prime example of such a system. Given an aligned face image, the DNN produces a vector (referred to as an embedding) representing the identity of the face. If a pre-trained model is used, the results are very robust, and no fine-tuning of the model is required. The last part consists of comparing the obtained vectors with the ones that are stored in a reference dataset. The comparison is usually carried out as a computation of the distance of the vectors. In the described case, it is the angular distance. The closest reference vector represents the identity of the input vector unless the distance is greater than a threshold. Then, the input vector is considered to be a new, unknown identity. In any case, the embedding vectors can be used to track individual faces in time. Moreover, based on the information about the position and distance of the detected face, it is appropriate that the robot moves to center the face in the acquired image stream and reach a predefined distance. In practice, it is done by transforming the actual position of the face directly into the movement of the robot.

2.2 Speech technologies

Our own speech cluster technology named SpeechCloud provides a simple solution to accessing speech technologies from a wide variety of end devices. The SpeechCloud cluster was designed entirely as the in-house solution for providing a flexible interface for various speech-related engines, such as automatic speech recognition (ASR), speech synthesis (TTS), and spoken language understanding (SLU). Prior to the development of the SpeechCloud platform, we used a single-user dialog implemented as a desktop application or as a VoiceXML-based dialog (Švec and Šmídl (2010)). Both approaches were very limited in the means of flexibility and modularity. However, the experience gained using such a system led us to a design of a distributed and extensible SpeechCloud platform.

During the design phase of the SpeechCloud in 2015, we used the recent technologies enabling the use of speech interfaces in the web browser, which transformed into a set of standards now called the WebRTC. WebRTC is a web browser API allowing the acquisition of an audio recorder (microphone) and audio player (loudspeakers) from the JavaScript code together with the session maintaining protocols like SDP (Session Description Protocol). On the server-side, we used the SIP (Session Initiation Protocol) to connect the web browser with the speech engines and RTP (Real-time Transport Protocol) for the transmission of the audio packets. We used the FreeSwitch software [1] as the telecommunication stack, which interconnects the web-browsers with the allocated speech engines.

To interconnect the WebRTC with the FreeSwitch, we used the SIP.js [2] library, which encapsulates the SIP protocol inside the WebSocket messages. This is due to a well-known limitation of web-browsers, which are unable to communicate on an arbitrary TCP/IP port using an arbitrary protocol. Instead, the standard WebSocket protocol must be used, and FreeSwitch allows the export of a SIP-over-WebSocket interface. After initiation of the SIP session, the SDP messages and RTP packets are sent directly by the browser as part of the WebRTC.

The interaction with the web browser allows using the speech technologies provided by SpeechCloud in the interactive and multi-modal web applications. Such applications could combine the use of speech-based dialog with the human-machine interaction using a keyboard, mouse, touchpad, or touchscreen.

The WebRTC-enabled web-browsers are only a subset of supported client types. The remaining client types rely on the use of SIP directly. Such clients include the telephone-only clients which use only the speech-based dialogs –

[1] `https://freeswitch.com/`
[2] `https://sipjs.com/`

e.g. a customer-care service dialog which is a part of the customer-care call-center.

Another types of clients include the stand-alone devices and robots, which are also capable to communicate with the SpeechCloud cluster using the SIP protocol. Such devices have their own audio hardware (microphone, loudspeakers), and the user is interacting with them. For example, the Google VoiceKit[3] development kit contains the audio hardware, and we provide an OS image that uses the SpeechCloud cluster instead of the standard Google services.

SpeechCloud architecture The overall architecture of SpeechCloud is based on Docker images and allows the multi-node deployment to exploit the computation power of a large computer cluster. The SpeechCloud client interacts with the SpeechCloud API server and starts a session using a specific URL that bears the specification of the set of required speech technologies. The SpeechCloud API server then allocates a specific SpeechCloud worker allocated for the session, and the audio is routed by FreeSwitch between the client and the associated worker. Alongside the audio, the control messages from client to worker and vice-versa are transferred using the WebSocket connection as JSON-encoded messages. The API server validates such messages against the session schema and also logs the messages for further debugging and statistics collection.

SpeechCloud workers The workers typically instantiate a speech recognizer and a speech synthesizer. We use our in-house large vocabulary continuous speech recognition system, Pražák et al. (2021) and also the in-house speech synthesis, Tihelka et al. (2018). We typically use 2 CPU cores for each worker, but the nature of spoken dialog allows us to overprovision the workers because the humans are not talking the whole time during the dialog and the CPU load interleaves across simultaneous sessions.

The workers also have a set of post-processors allowing additional processing of the speech recognition outputs. One kind of post-processor allows generating n-best lists of hypotheses. Another post-processor provides the Semantic Entity Detection (SED) algorithm. Such an algorithm uses a set of pre-defined grammars in the Speech Recognition Grammar Specification (SRGS) format[4] to describe the entities which are further detected in the recognition output (Švec et al. (2013)).

SpeechCloud infrastructure In addition to the real-time processes (API server and workers), we also have specific tooling for maintaining the models. The tooling allows to automatically interpolate small domain-specific language models with a large generic (background) language model. This way, we can design the speech recognition models for specific dialogs, including domain-specific words and commands.

Applications of SpeechCloud We used the SpeechCloud cluster in many dialog systems, including the *An Automatic Training Tool for Air Traffic Control Training*, Stanislav et al. (2016), *A Multimodal Dialogue System for Air Traffic Control Trainees*, Šmídl et al. (2016) or *A Multimodal Dialog with the MALACH Audiovisual Archive*, Chýlek et al. (2019). The SpeechCloud cluster is also a backend for the HTTP-based speech recognition service *UWebASR*, Švec et al. (2018).

3. DIALOG MANAGER

In the unimodal architecture (speech-only), the dialog manager observes the state of a user using ASR and SLU modules, and on the opposite side, it changes the state of the user using natural language generation (NLG) and speech synthesis (TTS). The ASR, SLU, and TTS modules are provided as a part of the SpeechCloud platform. Therefore the dialog manager only maintains its internal state based on events from ASR/SLU and generates output texts which are synthesized using TTS methods.

The dialog manager is implemented as a WebSocket server. We developed a small Python library that simplifies the implementation using an asynchronous programming paradigm (async/await primitives and coroutines). This way, many parallel sessions to a single WebSocket server could be established simultaneously without the use of multiple processes/threads. The dialog manager consists of asynchronous calls to methods provided by the SpeechCloud platform and also of await commands for SpeechCloud events.

4. PROPOSED ARCHITECTURE

While designing the architecture of the system, our goal was to reuse the already existing SpeechCloud platform. This platform has many advantages which we wanted to further exploit in the proposed multi-modal architecture:

- Cluster-based architecture – the communication is centered around the *SpeechCloud API server*, which interconnects a *client* and an associated *worker*.
- Worker parallelism – the cluster can run multiple workers each performing speech recognition, synthesis and other services for exactly one client.
- External control of interaction – the dialog with the user is controlled by an external process called *dialog manager*, which uses the services of SpeechCloud cluster to interact with the user using a speech-based interface.
- Use of standard protocols – the components of the SpeechCloud architecture are interconnected using standardized internet protocols including the WebSocket protocol to interchange control messages and SIP/RTP protocols to implement voice-over-IP.
- Message validation – in the SpeechCloud, each session has a pre-defined scheme of communication. Each message exchanged in the session is serialized as JSON and the JSON messages are validated against the JSON Schema associated with the session.

4.1 Speech-only SpeechCloud architecture

In the speech-only mode, the SpeechCloud architecture consists of a *SpeechCloud client* with an audio interface (see Fig. 1). The client interacts with a human user of the dialog system, converts the speech into a stream of

[3] `https://aiyprojects.withgoogle.com/voice/`
[4] `https://www.w3.org/TR/speech-grammar/`

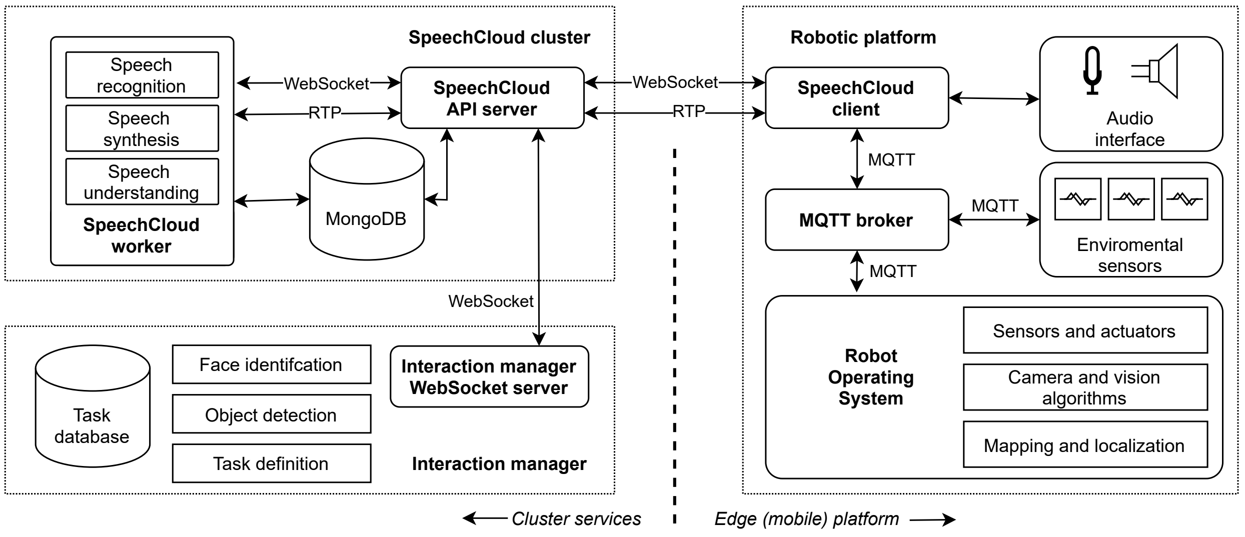

Fig. 1. Proposed architecture of the system. The data-center-based cluster components are on the left part, the edge hardware components on the right part of the figure.

RTP packets and sends them to speech processing in the SpeechCloud cluster.

The *API server* redirects the RTP stream to the *worker* assigned to the current session, and the worker ASR recognizes the speech of the user. The recognition result is emitted as an *event*. An *interaction manager* can listen for such an event (receive a JSON message) and perform a specific scenario based on the recognition result. For example, the interaction needs to synthesise some prompts using TTS. Therefore the interaction manager needs to call a *method* (send a JSON message), the worker receives the messages, invokes the speech synthesis, and the resulting speech is streamed using the RTP protocol back to the client, which transforms it into the audible speech perceived by the user.

For handling the relationship between the clients, workers, and sessions, the SpeechCloud platform uses a MongoDB NoSQL document database. The database also stores all events emitted and methods called during a session. The database also holds all audio records processed and generated during the session.

4.2 Multi-modal extension of a SpeechCloud architecture

Our goal was also to design the architecture in such a way, that will allow the *interaction manager* to directly control the low-level robot control primitives running on the edge device (robot) such as SLAM, environmental sensors, motion planning, etc. We addressed these tasks in papers Neduchal and Flídr (2016), Neduchal and Železný (2021) and Neduchal et al. (2020) For this control, we also reused the WebSocket channel and the SpeechCloud concept of methods/events.

To add the multi-modal capabilities to the existing architecture, we extended the functionality of the SpeechCloud client. The client in the multi-modal scenario acts as a bridge between the edge robotic platform and the rest of the architecture. We suppose that the edge platform runs a dedicated local *MQTT broker* (see Fig. 1) – a lightweight service which provides services based on the publish/subscribe paradigm. The MQTT protocol is widely supported, and it can also act as a standard interface to the ROS.

Therefore, the SpeechCloud client connects to the local MQTT broker and translates WebSocket events into published MQTT messages. It also subscribes for MQTT topics representing the SpeechCloud methods, and if the MQTT message is published in such a topic, the corresponding SpeechCloud method is invoked. This way, the bidirectional bridge between the SpeechCloud session and the local MQTT is established.

On the other hand, the ROS messages are directly mapped to MQTT messages, and therefore the ROS processes have direct access to the SpeechCloud events and methods.

The MQTT broker could be used not only to interconnect the SpeechCloud session and ROS but also to integrate other software/hardware features of the edge platform, for example, environmental sensors or third-party services.

5. INTERACTION MANAGER

To provide the task-dependent implementation of the interaction, we reused the concept of dialog manager from pure spoken dialog systems. This concept is generalized to the multi-modal distributed interaction manager, where the computer vision algorithms such as face-feature extraction could run on the edge device, and the corresponding counterpart performing face identification runs on the server sharing its database of known faces. The interaction between the two parts of the computer vision algorithms is mediated using the WebSocket bidirectional channel. This way, the interaction manager could work as a dialog manager for the spoken dialog and also control the interactions between different modalities.

The interaction manager is implemented the same way as a dialog manager (see Sec. 3). The Python library automatically defines a set of events and methods based on the schema of a session. Therefore the multi-modal extensions of the speech-only dialog are directly available to the programmer. The multi-modal events and methods are defined in the asynchronous programming framework (async/await paradigm). Therefore, could be naturally parallelized, and cross-modal implementation of events and methods interactions is possible.

The following listing shows a simple Python code snippet that could be used as a simple interaction manager. The snippet is simplified and tries to illustrate a simple multi-modal interaction: The robot first introduces itself using the TTS (method **tts_synthesize** and event **tts_done**). Then, it waits until the face is detected on the edge device (method **cv_detect_face**) and based on the feature vector of the detected face the person's identity is determined locally (method **identify_face** without the await command). Then the robot greets the user using the TTS by his/her name, starts the ASR (method **asr_recognize**) and awaits the recognition result (event **asr_results**):

```
async def main(self):
    await self.tts_synthesize("Hello, I'm robot")
    await self.tts_done()
    features = await self.cv_detect_face()
    identity = self.identify_face(features)
    await self.tts_synthesize("Hi "+identity.name)
    await self.tts_synthesize("Say your command")
    await self.tts_done()
    await self.asr_recognize()
    command = await self.asr_result()
```

6. IMPLEMENTATION & FUTURE WORK

The system is implemented based on the proposed architecture shown in Fig. 1. Thus, it is visible from the architecture that it is a hybrid system. By hybrid, we mean the possibility of using computing resources both on the side of the mobile platform and on the side of the cluster services. In detail, low-level tasks such as data acquisition and actuator controlling run on the mobile device as well as localization and mapping and computer vision approach. On the other hand, it is possible to transfer vision computation to cluster as well. Thus, the goal is to run a system that can work with the resources of both sides in order to ensure its speed and stability.

The system is developed in Python with additional packages and technologies such as the MQTT client library paho.mqtt, Python-based asynchronous web server named Tornado and ROS as a framework for controlling important parts of the mobile robot. The first two mentioned technologies are necessary for communication between the robot and the SpeechCloud cluster. We chose ROS because it is a well-known set of software libraries and tools designed for robotic applications with which we have experience, Neduchal and Flídr (2016). In particular, we used the ROS Melodic version on Ubuntu 18.04. This version of Ubuntu Linux was picked because it is impossible to easily run a newer version on an embedded computer that we used for our tests.

The use of Ubuntu 18.04 also has one disadvantage. The main Python in this particular operating system is 2.7 accompanied with Python 3.6. On the other hand, to run the SpeechCloud client, it is necessary to use at least Python 3.7. Thus, we decide to use a Docker container with a newer Python version to run the SpeechCloud client and MQTT broker.

The message from the robot then travels from ROS into the MQTT broker running inside Docker. Then it is sent to the SpeechCloud client and finally to the SpeechCloud cluster. The message from the cluster is sent through the same parts of the system but in reverse order. The SpeechCloud cluster distributes the messages to the workers associated to the session as well as to the interaction manager, which performs the domain-dependent tasks.

The proposed architecture was tested on the mobile robot platform shown in Fig. 2. It is based on 6WD Pololu Wild Thumper chassis. The main computer is Nvidia Jetson Xavier with an additional 512GB SSD. Images are grabbed by Intel Realsense D435 RGBD camera. Moreover, RPLiDAR A3 and IMU are also mounted to our platform to ensure that the robot would be capable of doing precise localization and mapping.

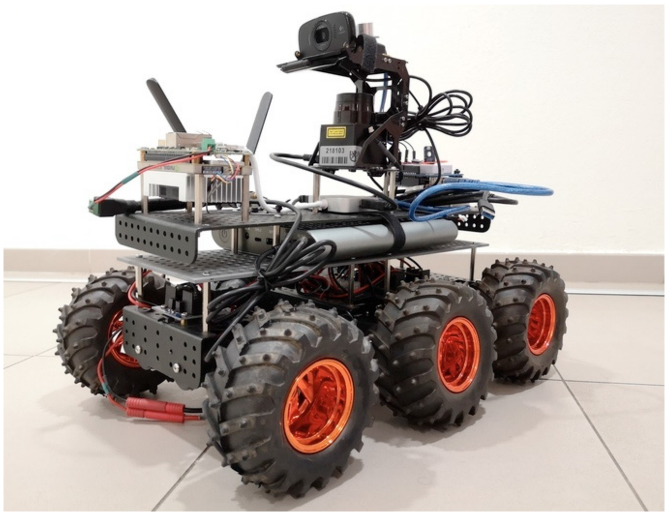
Fig. 2. Illustration photo of used mobile robot platform.

In the future work, we want to address two main areas. The first one is the area of Simultaneous Localization and Mapping. Although we tested the function of both LiDAR and Visual SLAM algorithms, we want to enhance it by our environment classification system proposed in Neduchal and Železný (2021) and Neduchal et al. (2020).

Another area that we want to address is object detection and semantic segmentation that can enable the mobile robot to operate with objects recognized from the input data. The resulting mobile robot should be able to perform tasks given by a human. Moreover, it can be possible to operate in a multi-environment manner – e.g., indoor and outdoor.

7. CONCLUSION

In this paper, the multi-modal human-robot interaction architecture based on a speech cluster technology named SpeechCloud was presented. Its main novelty is in using of an existing cluster-based speech-only architecture. The architecture supports worker parallelism which increases the efficiency of the whole system. The advantage of the system is the use of standard frameworks, protocols, and message validation. By frameworks and protocols, it primarily means Robot Operating System, MQTT broker, and WebSocket protocol. Message validation in the SpeechCloud is done by validating JSON messages used in the communication with a predefined JSON Schema associated with the session.

The SpeechCloud was focused on speech processing only in its early development phase. In this paper, we propose a multi-modal extension of the architecture. It is done in a task-dependent manner by implementing an interaction manager responsible for running actions on the server-side based on the interactions that have occurred. The interaction manager is also responsible for controlling the edge-device capabilities such as computer vision algorithms and robotic algorithms.

Finally, an advantage of proposed architecture is that its easy expandable by new algorithms and approaches. We mentioned an example in future work in Section 6 where we described the use of environmental sensors for environment classification or semantic segmentation task, Neduchal et al. (2019). In the next phase of our research, we focus on these particular extensions in order to develop full system for a mobile robot.

ACKNOWLEDGEMENTS

This research was supported by the Technology Agency of the Czech Republic, project No. TN01000024.

REFERENCES

Budkov, V.Y., Prischepa, M., Ronzhin, A., and Karpov, A. (2010). Multimodal human-robot interaction. In *International Congress on Ultra Modern Telecommunications and Control Systems*, 485–488. IEEE.

Chýlek, A., Šmídl, L., and Švec, J. (2019). Multimodal Dialog with the MALACH Audiovisual Archive. In *Proc. Interspeech 2019*, 3663–3664.

Deng, J., Guo, J., Xue, N., and Zafeiriou, S. (2019). Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4690–4699.

Garage, W. (2012). Robot operating system (ros).

Gorostiza, J.F., Barber, R., Khamis, A.M., Malfaz, M., Pacheco, R., Rivas, R., Corrales, A., Delgado, E., and Salichs, M.A. (2006). Multimodal human-robot interaction framework for a personal robot. In *ROMAN 2006-The 15th IEEE International Symposium on Robot and Human Interactive Communication*, 39–44. IEEE.

Guo, J., Deng, J., Lattas, A., and Zafeiriou, S. (2021). Sample and computation redistribution for efficient face detection. *arXiv preprint arXiv:2105.04714*.

Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2117–2125.

Lucignano, L., Cutugno, F., Rossi, S., and Finzi, A. (2013). A dialogue system for multimodal human-robot interaction. In *Proceedings of the 15th ACM on International conference on multimodal interaction*, 197–204.

Mizuchi, Y. and Inamura, T. (2017). Cloud-based multimodal human-robot interaction simulator utilizing ros and unity frameworks. In *2017 IEEE/SICE International Symposium On System Integration (SII)*, 948–955. IEEE.

Neduchal, P., Bureš, L., and Železný, M. (2019). Environment detection system for localization and mapping purposes. *IFAC-PapersOnLine*, 52(27), 323–328.

Neduchal, P. and Flídr, M. (2016). Development of a laboratory framework for testing simultaneous localization and mapping approaches. *IFAC-PapersOnLine*, 49(25), 493–498.

Neduchal, P., Gruber, I., and Železný, M. (2020). Indoor vs. outdoor scene classification for mobile robots. In *International Conference on Interactive Collaborative Robotics*, 243–252. Springer.

Neduchal, P. and Železný, M. (2021). Environment classification approach for mobile robots. In *Proceedings of 15th International Conference on Electromechanics and Robotics" Zavalishin's Readings"*, 421–432. Springer.

Pražák, A., Loose, Z., Psutka, J.V., Radová, V., Psutka, J., and Švec, J. (2021). Live TV Subtitling Through Respeaking. In *Proc. Interspeech 2021*, 2339–2340.

Stanislav, P., Šmídl, L., and Švec, J. (2016). An automatic training tool for air traffic control training. In *Interspeech 2016*, 782–783.

Švec, J., Ircing, P., and Šmídl, L. (2013). Semantic entity detection from multiple ASR hypotheses within the WFST framework. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, 84–89. IEEE, Olomouc, Czech Republic. doi: 10.1109/ASRU.2013.6707710.

Švec, J. and Šmídl, L. (2010). Prototype of czech spoken dialog system with mixed initiative for railway information service. In P. Sojka, A. Horák, I. Kopeček, and K. Pala (eds.), *Text, Speech and Dialogue*, 568–575. Springer Berlin Heidelberg, Berlin, Heidelberg.

Tihelka, D., Hanzlíček, Z., Jůzová, M., Vít, J., Matoušek, J., and Grůber, M. (2018). Current state of text-to-speech system artic: AÂ decade of research on the field of speech technologies. In P. Sojka, A. Horák, I. Kopeček, and K. Pala (eds.), *Text, Speech, and Dialogue*, 369–378. Springer International Publishing, Cham.

Yongda, D., Fang, L., and Huang, X. (2018). Research on multimodal human-robot interaction based on speech and gesture. *Computers & Electrical Engineering*, 72, 443–454.

Zhu, Y., Cai, H., Zhang, S., Wang, C., and Xiong, Y. (2020). Tinaface: Strong but simple baseline for face detection. *arXiv preprint arXiv:2011.13183*.

Šmídl, L., Chýlek, A., and Švec, J. (2016). A Multimodal Dialogue System for Air Traffic Control Trainees Based on Discrete-Event Simulation. In *Proc. Interspeech 2016*, 379–380.

Švec, J., Bulín, M., Pražák, A., and Ircing, P. (2018). UWebASR - Web-based ASR engine for Czech and Slovak. In *CLARIN Annual Conference 2018 Proceedings*, 190–193.