BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 68, No. 5, 2020
DOI: 10.24425/bpasts.2020.134662

Agent-based approach to the design of a multimodal interface for cyber-security event visualisation control

W. KASPRZAK, W. SZYNKIEWICZ*, M. STEFAŃCZYK, W. DUDEK, M. WĘGIEREK, D. SEREDYŃSKI, M. FIGAT, and C. ZIELIŃSKI

Warsaw University of Technology, Institute of Control and Computation Engineering, ul. Nowowiejska 15/19, 00-665 Warsaw, Poland

Abstract. Convenient human-computer interaction is essential to carry out many exhausting and concentration-demanding activities. One of them is cyber-situational awareness as well as dynamic and static risk analysis. A specific design method for a multimodal human-computer interface (HCI) for cyber-security events visualisation control is presented. The main role of the interface is to support security analysts and network operators in their monitoring activities. The proposed method of designing HCIs is adapted from the methodology of robot control system design. Both kinds of systems act by acquiring information from the environment, and utilise it to drive the devices influencing the environment. In the case of robots the environment is purely physical, while in the case of HCIs it encompasses both the physical ambience and part of the cyber-space. The goal of the designed system is to efficiently support a human operator in the presentation of cyberspace events such as incidents or cyber-attacks. Especially manipulation of graphical information is necessary. As monitoring is a continuous and tiring activity, control of how the data is presented should be exerted in as natural and convenient way as possible. Hence two main visualisation control modalities have been assumed for testing: static and dynamic gesture commands and voice commands, treated as supplementary to the standard interaction. The presented multimodal interface is a component of the Operational Centre, which is a part of the National Cybersecurity Platform. Creation of the interface out of embodied agents proved to be very useful in the specification phase and facilitated the interface implementation.

Key words: multimodal interface, cyber security visualisation, embodied agent.

1. Introduction

The degree of dependence of modern states on Information and Communications Technology (ICT) systems (including smart grid [1] and cyber-physical [2] systems) has increased to such a level that accidental or intentional damage to this infrastructure may have catastrophic consequences. Therefore, suitable tools are needed to detect, prevent and mitigate the effects of activities that violate the security of ICT infrastructure that is important for effective functioning of the country and its citizens. To this end, the National Cyber-Security Platform (NCP) is being created, which will enable on-going monitoring of the network operation and will ensure coordination on the national level of activities aimed at detecting and preventing unwanted situations in cyberspace. The main goal of NCP is to achieve comprehensive, country-wide view of cyber-threats to evaluate risk in real-time, as well as to monitor the current state of various essential services. National Cyber-Security Centres have been established in the majority of highly developed countries e.g. Great Britain [3], United States [4] or Netherlands [5]. They are responsible for national cyber-security, with primary focus on securing government networks, protecting critical national infrastructure, and assisting businesses and citizens in protecting their own systems.

Comprehensive network analysis requires the detection of correlation of events occurring in the cyberspace, situational awareness as well as dynamic and static risk analysis. To develop a useful and effective tool for such an analysis appropriate methods and techniques for multidimensional data visualisation are needed [6, 7]. Visualisation is one of the most useful tools for cyber-security operators. By gathering appropriate data and visualizing it logically, situational awareness is increased, and attacks can be easily communicated to the users who need this information and the attack can be addressed properly. This is underscored in [8] where the authors point out that clear understanding of the users' needs and addressing their requirements are critical factors to successfully develop insightful visualisations.

To assist analysts in detecting patterns in network activity data, and detecting anomalies, an interactive realtime dashboard, BubbleNet, for visualizing patterns in cyber-security data was developed [9]. This dashboard presents data in diverse views: a view of the location in the form of a map with spatial information encoded using Dorling-like cartograms [10], time view in the form of bar graphs and the view of attributes that graphically represent different data attributes.

In [11] a visual analysis system, Voila, for interactively detecting anomalies in spatiotemporal data from a streaming source was presented. The interactive graphical user interface of this system consists of eight main views, including macro and micro-maps, pattern time view and a feature inspection view. The user can select the area on the macro-map with a mouse and display the list of anomalies detected for this area

*e-mail: W.Szynkiewicz@elka.pw.edu.pl

Manuscript submitted 2019-12-13, revised 2020-05-18, initially accepted for publication 2020-07-05, published in October 2020

on the micro-map. It can also perform zooming and shifting operations and changing the display mode on both maps.

The VisIDAC [12] system enables real-time visualisation of 3-D data of security event log collection detected by intrusion detection systems installed in many networks. Event data are displayed in a graphical form on three panels: for global source networks, target networks, and global destination networks. In order to easily distinguish types of events, different shapes and colors are used. It helps security analysts to immediately understand the key properties of security events.

The development of a system to monitor computer network cyber-security entails the necessity to create a multimodal interface enabling intelligent control of multidimensional visualisation of events occurring in cyberspace. Research work on multimodal human-computer interfaces (HCI) has been ongoing for over 40 years [13–16]. The aim of this research is to develop methods and techniques of human-computer interaction that fully utilize the means of natural communication and human interaction with the environment. Multimodal interfaces are characterized by two basic features: combining many types of data and processing it all in real time under specified time constraints [13].

Since the appearance of Bolt's [17] "Put That There" demonstration system, which processed speech in parallel with touchpad pointing, a variety of new multimodal interfaces has emerged [13–15, 18].

The majority of the above mentioned interfaces uses standard input devices such as a keyboard or a mouse for adjusting the way the data is presented. Unfortunately an eight-hour work-shift imposes a great burden on the muscles and joints of the analyst. Those are generally categorised as musculoskeletal disorders [19–22]. Hence alternatives are looked for. The standard methods of interaction with such systems will be retained, but voice and gesture interaction at least partially alleviates the problem. The analyst while still working can get up and exercise for a while, reducing the stress cumulated in the muscles and joints. Moreover, when the operator presents the current situation to other people it is more convenient to use gesture and voice rather than command the system by using standard devices. Hence a multimodal interface not only allows analysts to adapt the presentation of the current network traffic to their requirements, but also reduces the negative impact on their wellbeing. Thus two main visualisation control modalities have been added. The first commands the system by utilising static and dynamic gestures and the second by using voice-based commands. Thus the interface interacts with an operator, being a part of the environment, similarly to that how a robot interacts with its environment. Hence the idea of using a similar approach to the design of the interface that has been used successfully to design robot controllers.

The approach followed here utilises embodied agents composed of real and virtual receptors and effectors as well as a control subsystem [23–26]. The structure of the embodied agents in the case of robotic systems and the interface is the same, also the activities of those agents are specified in the same way, however the environment had to be defined in a different way, thus the treatment of some of the receptors and effectors had to be modified. This paper, besides presenting the design method, which can be used to build other human-machine interfaces, presents the particular multimodal interface developed for the Operational Centre that is a part of the NCP. Thus also the design and implementation and performance of gesture recognition and speech-recognition modules composed of embodied agents is presented. The speech module is also responsible for speaker identification in order to limit the access to privileged users only, registered with the visualisation control system. The way in which security data is presented to the operator and the analysts has a significant impact on the efficiency of their work. Hence the design of the interface to such systems is of utmost importance.

There are several goals that this work wants to attain. The research goal is to verify whether the embodied agent-based system design methodology, that was initially applied to the design of robotics systems, can be utilised in the creation of smart human-computer interfaces? If yes, what modifications are required? The technology-oriented goal is to design smart HCI modules providing gesture recognition and speech recognition capabilities, so that they can be utilised in many other domains than just in cybersecurity system interfaces. The practical goal is organisational: to strengthen the safety and improve communication and operation speed of cybercsecurity centers in various institutions. Finally, there is an ergonomic goal. The proposed interface enables the analysts to work either sitting or standing. In the former case standard input devices will be used and in the latter gesture and voice will be utilised. Standing position alleviates back pain problems. Moreover, an analyst can present to the others the network situation standing far from the screen, thus without obstructing their view. Nevertheless, we need to highlight that this kind of command input is not treated by us as primary.

The structure of the paper follows the design steps advocated by software engineering: statement of requirements and functionality of the interface (Section 2), system specification including also the methodology used for that purpose (Section 3), the most important aspects of the implementation, i.e. interaction of the interface with the operating system and the voice and gesture recognition algorithms (Section 4), tests of the interface, especially the results of voice and gesture recognition (Section 5). Section 6 provides the conclusions derived from the conducted work and presents our future plans.

2. Functionality of the interface

The general concept and functionality of a designed system results from the requirements imposed on it. The requirements stem from the analysis of the categories of the users of the system and the operations that they will have to perform.

2.1. General concept and requirements. Both human-machine interfaces (HMI) and robotic systems act at the contact point between the software system and physical environment. Both types of systems get stimulus from the environment. This stimulus is detected by receptors. Subsequently the acquired

data is processed in accordance with the task of the system. Finally the system influences the environment through its effectors. In the case of the NCP the operator acts in the environment, he/she is the source of the stimulus and is the target of the actions of the system.

The first step in the design of a system is its distinction from the environment. It was assumed that the most natural form of interaction between the operator and the NCP is the one using voice and gesture communication. Hence the NCP interface was equipped with cameras and microphones. Moreover, the operator, when bootstrapping the system, has to configure it, and later on supervise it. For that purpose he/she uses standard input devices, such as a mouse, keyboard or touchpad. The feedback information to the operator is presented on the screen of the monitor, specifically in the windows appearing on it. The interaction takes place through the cursors shifted by the operator on the screen. Those input and output devices delimit the system. The operator acts in the physical environment, which is extended onto the monitor screen. The system, here the NCP and its interface, interacts with the operator using the enumerated devices and the windows, as well as cursors appearing on the screen.

Due to the fact that previous NCPs had used touchpads, keyboards and mice as input devices, it was assumed that the voice and gesture commands will be translated into the format customarily produced by those input devices. Gestures are produced by hands, thus the camera images must be acquired with adequate frequency. It was assumed that image and voice processing must be fast enough, so that the operator issuing a command will not notice a significant delay in the reaction. In voice analysis a delay occurs between the end of uttering a command and the termination of its analysis. It was assumed that the admissible delay is up to 1 s, because the voice analyser must wait approximately 200 ms to decide whether the utterance has been terminated or it will be continued, and only then can the voice analysis be conducted, what again needs time. In the case of image analysis the user controls the motion of the cursor, expecting a much faster reaction and continuity of its motion. Thus a single image must be analysed within 50 ms, and the reaction time must be below 100 ms. The structure of the operator interface must conform to those requirements.

The interface is configured using standard input devices and its normal operation is stimulated by an operator using voice or gesture commands. Moreover, it reacts to the operator commands by presenting data on the screen. Thus its structure may be defined with the use of embodied agents (see Sec. 3.1 or [27, 28]), which realise their tasks acquiring data from the environment using receptors and influence the environment through their effectors. However receptors and effectors must be defined adequately. Up till now embodied agents have been utilised to create robotic systems [24–26]. This paper shows their utility in the design of systems not directly related to robotics.

2.2. The tasks of the interface and categories of its operators. The interface to the NCP visualisation system plays three roles, related to different categories of platform operators. It:

- provides to the user (e.g. NCP manager) convenient control of visualisation of the information created by the platform,
- allows the service engineer to configure a given interface installation,
- enables the system administrator to manage resources (information about gestures, users and voice commands) and create models of gestures, voice commands and users.

These roles relate to three different aspects (modes) of the interface functioning: configuration, administration (modeling and data management) and control.

The **user** uses the interface to command the system. The **service technician** configures the equipment and parameterises the software executing image and speech analysis. The hardware and software configuration of each module consists of providing the values of internal parameters for a given installation of the visualisation system. The **administrator** manages: gesture models, command models and user data, but above all provides training data and supervises the learning process (model creation). This applies to both the management of information about the system users and the commands the system has to recognise. In the case of user data management, it is possible to append new users to the system or remove those who are not needed, and edit the data. In the case of commands, it is possible to add, remove or edit them.

Learning is used to define the assignments of gestures to commands and voice utterances to commands. Moreover, it is used to recognise the speaker to determine his/her eligibility. It is anticipated that different categories of users will be eligible to exert different influence over the system. The learning process takes place in two phases. First, samples in the appropriate form are collected, and then the respective classifier learns using them. Supervision over the learning process rests with the system administrator.

Control consists of recognizing user commands conveyed by voice or gesture. These commands are transformed into stimuli customarily generated by the mouse (cursor movement, clicking), keyboard (pressing a key) or the touch screen (touch). These stimulants affect the windows represented by agents or supplementary programs running on the computer. The stimuli are interpreted by these agents or programs as, for example, rotation of the image, zooming in or out of the view displayed in the window.

In the case of the presented interface, the gestures and voice commands are integrated and subsequently transformed into **visualisation control codes** accepted by the platform, so that their effects on the screen can be observed by the user in his/her preferred form.

3. Specification of the interface

Any complex system must be decomposed into manageable components. Usually hierarchical decomposition is used. In the case of the NCP interface a two layer decomposition has been used. The upper layer consisted of modules. The lower layer resulted from the subdivision of modules into embodied agents.

3.1. Embodied agents. In the classical approach an agent is defined as an entity acting autonomously, perceiving its environment, having a persistent nature, possessing the ability of adapting to the changes occurring in the environment, and moreover being able to assume the goal of another agent [29–32]. Nevertheless, here a more concise definition is assumed, at the same time stating it in more general terms. It has been assumed that an agent is an entity striving to execute a certain task, perceiving its environment and influencing it rationally. The accomplishment of the task is the agent's internal urge to act. The task is realised in the environment, taking into account the conditions prevailing in that environment. For that purpose the agent uses its receptors to collect data and its effectors to exert its influence upon the ambience. If the agent operates in a material environment it needs a physical body (embodiment), to be able to influence it, thus it is an embodied agent [27, 28]. The agent structure of the designed system will be presented further on, here it suffices to notice that the actions of the operator, treated as a part of the environment, are detected by such receptors as cameras and microphones, while the NCP influences the operator through windows appearing on the screen, thus both the elements of windows displaying information and the cursors are treated as effectors. The task of the interface is to recognise the operator commands and to transform them into adequate visualisation of the NCP data. The benefit of using embodied agent meta-model to define a particular system model (its specification) is that an associated formal notation can be employed that facilitates the description of both the structure and the activities of the system [24, 27, 28]. Moreover, the transformation of such a specification into the code of the system is fairly simple. The main elements of the used notation are described herein. Each agent or its subsystem is denoted by a central symbol from the set $\{a, c, e, r, C, E, R\}$, representing its type. Particular subsystems and their components use this central symbol, however are distinguished by specific indices. Agent a_j, where j is its name, is composed of: real effectors $E_{j,m}$ (m – effector designator) interacting with the environment (in the case of NCP interface those are the windows and cursors appearing on the screen), real receptors[1] $R_{j,l}$ (l – sensor designator) gathering data about the state of the environment (in the case of NCP interface those are: cameras, microphones and other input devices), and the control system C_j executing the task by influencing the effectors, and taking into account the readings obtained from the receptors. The control system C_j is composed of three types of subsystems: virtual effectors e_j, virtual receptors r_j and the control subsystem c_j. An agent may contain many real receptors $R_{j,l}$ and many virtual receptors $r_{j,k}$, where l and k are their designators. Similarly it may contain many real effectors $E_{j,m}$ and many virtual effectors $e_{j,n}$, where m and n are their names. Virtual receptors $r_{j,k}$ and virtual effectors $e_{j,n}$ can be treated as the means for presenting the environment and the real devices to the control subsystem in such a way that expression of the task becomes straight forward. In the case of the NCP interface virtual effectors transform complex operator commands into elementary commands affecting windows and cursors presented on the computer screen, while the virtual receptors transform the sensor readings into abstract concepts needed for concise expression of the realised task. Agents may also communicate with each other, thus they can form a network. Figure 1 presents the internal structure of an embodied agent.

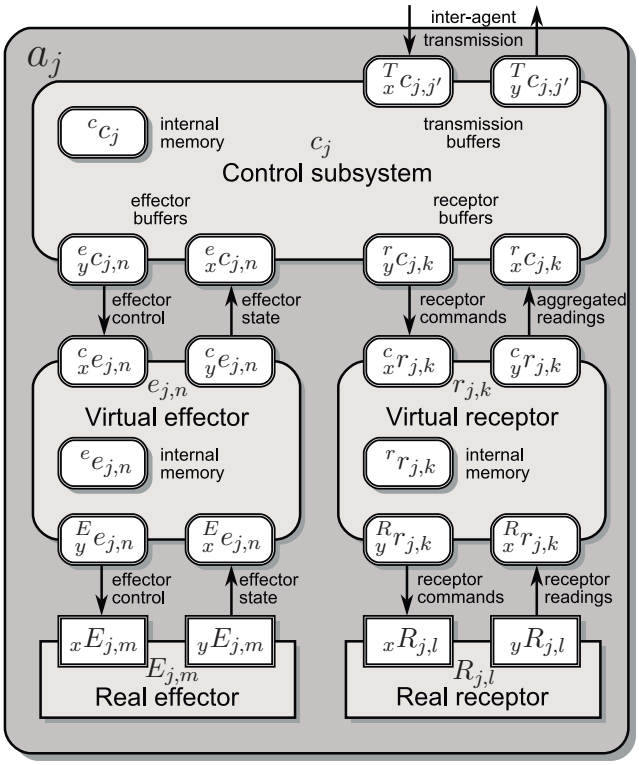

Fig. 1. General structure of an embodied agent a_j

Subsystems of agent a_j, i.e.: control subsystem c_j, virtual effectors $e_{j,n}$, virtual receptors $r_{j,k}$, real effectors $E_{j,m}$ and real receptors $R_{j,l}$ communicate using buffers (Fig. 1). Denotations of subsystem input buffers contain a left subscript x, the output buffers use y, while the internal memory does not have this subscript. The left superscript in those denotations points out the type of subsystem that the buffer interacts with.

Every subsystem acts according to the same pattern [24–26]. Each one contains a finite state machine (FSM). With each of its state a behaviour is associated. Each behaviour is parameterised by a transition function, which takes as arguments the values contained in the subsystem input buffers as well as its internal memory and produces values for the output buffers and the internal memory. Moreover, the behaviour is parameterised by two predicates: terminal condition and error condition. The behaviour is repeated as long as both of those predicates are false. Termination of the behaviour causes the FSM to transit to its next state, which is chosen based on the initial conditions labeling the directed arcs of the FSM graph. The activities realised by the behaviour depend solely on the definition of the transition function it is parameterised by. The behaviour pattern, besides the evaluation of the transition function, reads-in

[1] Strictly speaking those are the exteroceptors, as proprioceptors are directly coupled with effectors.

the new values into the input buffers and dispatches the data contained in the output buffers to the other subsystems.

The general design method consists in the execution of the following items in the order preferred by the designer. As the design procedure of any complex system is iterative, the order of the execution of its steps in each iteration might change. Taking into account the overall task of the defined system the designer must determine:

- necessary real effectors and receptors,
- number of agents and assign to them the real effectors and receptors (taking into account the transmission delays and the necessary computational power),
- individual tasks of each agent,
- virtual receptors and effectors for each agent, thus defining the concepts that the control subsystem will be using to express the task of the agent,
- finite state machine (FSM) switching the behaviours for each subsystem within each agent,
- behaviours executed in each FSM state, including their parameters: transition functions, terminal and error conditions.

Each behaviour is a template iteratively reading-in the arguments and writing-out the results of computation of the transition function, until the terminal or error condition (predicates) is satisfied. This step of the design procedure is neglected here for brevity. The names of behaviours have to suffice, instead of precise definitions of the mentioned functions.

3.2. Structure of the system. The system is composed of the **interface** modules and the **platform**, also being a module (Fig. 2). Each module of the interface is composed of **agents**. The interface consists of the following modules: presentation, video and audio. The number of interface modules results from the way the operator interacts with the interface and how the interface interacts with the platform. The user commands the system either by voice or by gestures, so there are two separate command sources, and hence two separate modules that process them: audio and vision. Regardless of the source, the commands must affect the operation of the platform without conflict. Therefore, a single platform controller, aggregating commands received from various sources, is optimal. This command aggregation is done by the presentation module. Hence the three-module interface structure. Moreover, the platform itself can be treated as a module composed of agents.

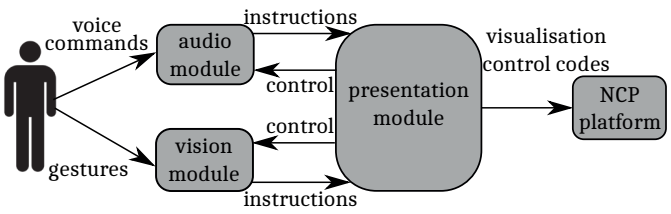

Fig. 2. Interactions between the interface modules and the platform

The **presentation module** is responsible for interpreting instructions obtained from modules processing the users' commands (i.e. video and audio modules). This module conveys the visualisation control codes to the platform, so that it can display the required information in the form demanded by the user. The presentation module accomplishes this by emulating specific input devices. As a result of this emulation, imitations of codes customarily produced by the input devices are transmitted to the agents representing windows, and in consequence, the desired effects appear on the screen. The **vision module** is used to process commands produced by gestures, while the **audio module** deals with voice commands and speaker identification.

3.3. Decomposition of the interface modules into agents. Each of the modules is composed of a set of agents. The structure of the presentation module is presented in Fig. 3, audio module in Fig. 4 and vision module in Fig. 5. Each of the three modules uses **window agents** a_α, where $\alpha \in \{\text{pGUI, aGUI\_init, aGUI\_user, aGUI\_cmd, aGUI\_ctrl, vGUI}\}$. They are responsible for producing windows visualised on the screen. The screen is treated as a part of the environment. Figure 6 defines the general structure of a window agent. Its real effectors are the boxes enabling the presentation of data on the screen, and its real receptors are the elements of the window sensitive to the stimulus generated by the real effectors of the agent a_{OID} (*Operator Interface Devices*). The graph of the FSM of the control subsystem of an agent a_α is presented in Fig. 7. In the first state, ${}^c\mathcal{S}_\alpha^{\text{init}}$, it initialises its memory and in the second, ${}^c\mathcal{S}_\alpha^{\text{control}}$, where it stays ever after, it reacts to the stimulus from the agent a_{OID} via its real receptors and to the other agents via inter agent communication buffers. The interaction between the a_{OID} and any of the window agents a_α is equivalent to communication through the environment,

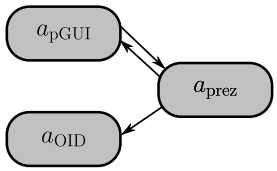

Fig. 3. Structure of the presentation module

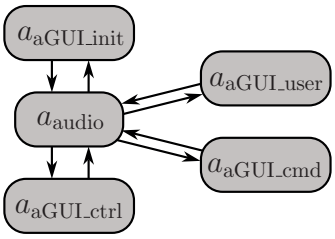

Fig. 4. Structure of the audio module

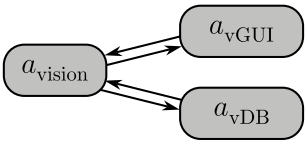

Fig. 5. Structure of the vision module

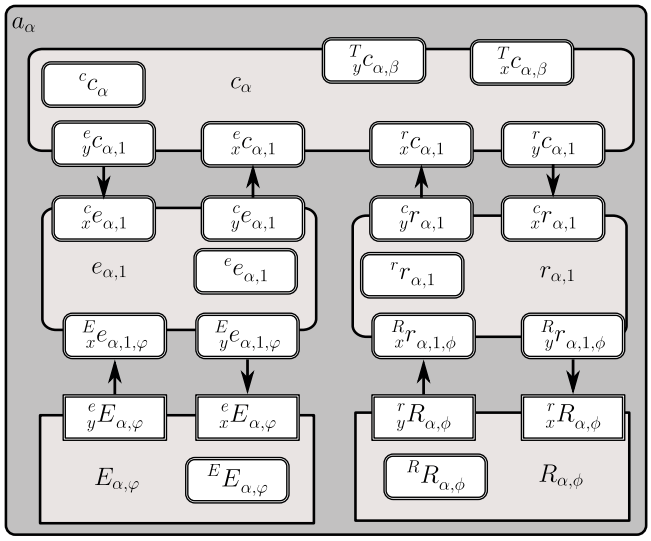

Fig. 6. General structure of a window agent a_α; where α is the name of the agent producing the window on the screen, β is the name of the agent with which a_α communicates, ϕ is a set of information presentation boxes of a window agent a_α (such as text fields), and φ is a set of action grabbers of a window agent a_α (such as buttons). The following pairs exist: $\alpha =$ pGUI and $\beta =$ prez, $\alpha =$ visionGUI and $\beta =$ vision, $\alpha =$ aGUI_ω and $\beta =$ audio (where $\omega \in$ {init, user, cmd, ctrl})

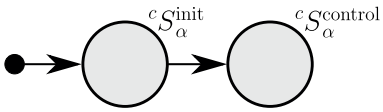

Fig. 7. FSM of the control subsystem of a window agent a_α

i.e. the screen, and is thus called **stigmergy** [33]. The real effectors of the agent a_{OID} represent the cursors, or speaking more precisely, stimuli generators, which produce stimuli usually caused by computer mouses, keyboards or touch screens. Its real receptors are: computer mouse, keyboard or touch screen. The agent a_{OID} interacts with a window agent a_α in the following way:

- a_{OID} in reaction to the stimulus produced by its real receptors, produces actions of its real effectors (i.e. cursors), which affect the real receptors of a window agent a_α (i.e. button or slider),
- window agent a_α perceives the stimulus via its real receptors and adequately reacts to it.

3.4. Presentation module. The presentation module is composed of three agents: a_{prez} (Fig. 8), a_{OID} (Fig. 9) and a_{pGUI} (Fig. 6). The control subsystem c_{prez} of the agent a_{prez} contains inter-agent communication buffers, connected not only to the other agents forming the presentation module, but also to a_{audio} and a_{vision}. The responsibility of a_{prez}, and thus its control subsystem c_{prez}, are as follows:

- transformation of the gesture identifiers (instructions), delivered by a_{vision}, into codes usually produced by input devices, such as mouse, keyboard, touch screen, and transmission of those codes to a_{OID},

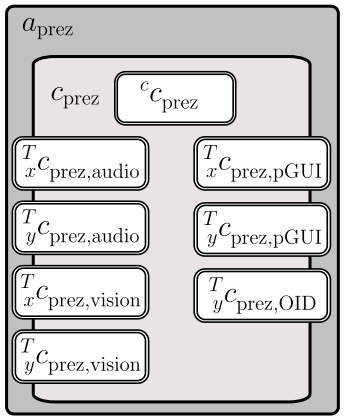

Fig. 8. Structure of the agent a_{prez}

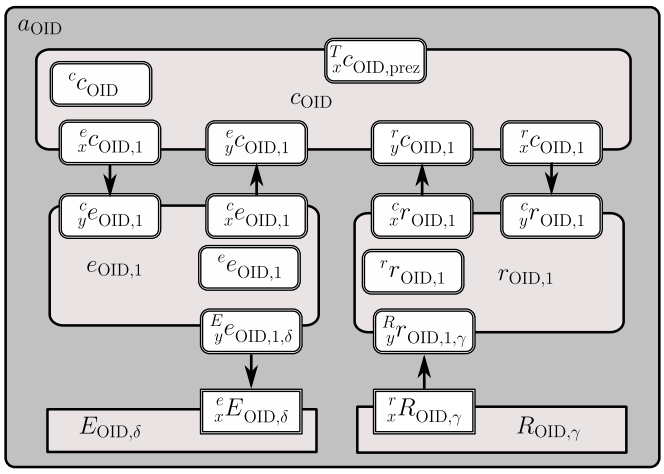

Fig. 9. Structure of the agent a_{OID}, where $\gamma \in$ { mouse, keyboard, touch screen}, $\delta \in$ {mainPointer, secPointer, keyExecutor}

- transformation of the voice command identifiers, delivered by a_{audio}, into codes usually produced by a keyboard and transmission of those codes to a_{OID},
- transmission of operation mode switch commands to a_{vision} and a_{audio} – these commands are delivered by the window agent a_{pGUI} of the presentation module,
- transmission of logging information to a_{pGUI} for visualisation on the screen – the logging information is provided by c_{prez} itself and by other agents: a_{vision} and a_{audio}.

The FSM of c_{prez} is presented in Fig. 10. In its first state, ${}^c\mathcal{S}^{\text{init}}_{\text{prez}}$, the memory of the control subsystem is initialised and in the second, ${}^c\mathcal{S}^{\text{control}}_{\text{prez}}$, information obtained from the other modules via inter-agent communication buffers is fused, and commands for the agent a_{OID} are generated. The agent a_{OID} drives three real effectors (cursors): $E_{\text{OID,mainPointer}}$ and $E_{\text{OID,secPointer}}$, as well as $E_{\text{OID,keyExecutor}}$. The last one produces the stimulus usually caused by a keyboard. The task of a_{OID} is to ini-

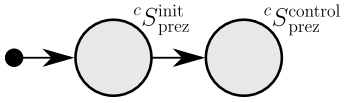

Fig. 10. FSM of the control subsystem c_{prez}

tiate actions triggered by input devices. The mentioned actions are caused by the real effectors $E_{\mathrm{OID},\delta}$, where $\delta \in \{\mathrm{mainPointer}, \mathrm{secPointer}, \mathrm{keyExecutor}\}$. Those effectors interact through stigmergy with the receptors of the agents presenting windows on the screen. The a_{OID} causes the actions in response to the requests from a_{prez} or to stimulus produced by a human operating the real receptors $R_{\mathrm{OID},\gamma}$ of a_{OID}, where $\gamma \in \{\mathrm{mouse}, \mathrm{keyboard}, \mathrm{touch\ screen}\}$.

The graph of the FSM of the control subsystem c_{OID} is presented in Fig. 11. In the first state, ${}^{c}\mathcal{S}^{\mathrm{init}}_{\mathrm{OID}}$, the memory of c_{OID} is initialised and in the second, ${}^{c}\mathcal{S}^{\mathrm{control}}_{\mathrm{OID}}$, commands obtained from $r_{\mathrm{OID},1}$ and a_{prez} (in the latter case via inter-agent communication buffers) are processed, and commands for real effectors are produced, e.g. to move a pointer called mainPointer or to execute a keystroke.

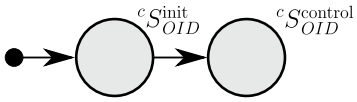

Fig. 11. FSM of the control subsystem c_{OID}

The agent a_{pGUI} produces a window displayed on the screen. It is responsible for:

- presenting the information delivered by a_{prez} (namely logging information),
- perceiving stimulus from the environment (i.e. the computer screen). This stimulus is produced by the agent a_{OID} as a response to the actions of the operator. This stimulus is sensed by the buttons, check lists and sliders of the window produced by the agent a_{pGUI} on the computer screen.

3.5. Audio module. The audio module consists of five agents:

- $a_{\mathrm{aGUI\_init}}$, which produces a window used to select subsequent windows: user management, command and control windows,
- $a_{\mathrm{aGUI\_user}}$, which creates the window used for the management of the audio module,
- $a_{\mathrm{aGUI\_cmd}}$, which forms the window for managing commands recognised by the audio module,
- $a_{\mathrm{aGUI\_ctrl}}$, which presents the window for management of recordings, model creation based on acquired sound recordings as well as control and
- a_{audio}.

The first four of them are window agents a_{α}, where $\alpha \in \{\mathrm{aGUI\_init}, \mathrm{aGUI\_user}, \mathrm{aGUI\_cmd}, \mathrm{aGUI\_ctrl}\}$ (Fig. 6). Each of them produces a single window having buttons and message boxes for displaying information. Out of the four enumerated window agents a_{α} only one is active at a time.

The control subsystem c_{audio} of the agent a_{audio} (Fig. 12) acquires data from the virtual receptor $r_{\mathrm{audio,mic}}$ and influences the virtual effector $e_{\mathrm{audio,ls}}$. It uses transmission buffers to communicate with the agent a_{prez} and the above-mentioned window agents of the audio module. The virtual receptor $r_{\mathrm{audio,mic}}$ aggregates data from the real receptor $R_{\mathrm{audio,mic}}$, which contains microphones, analog-to-digital converter and a software driver of this transducer installed in the operating system. The virtual

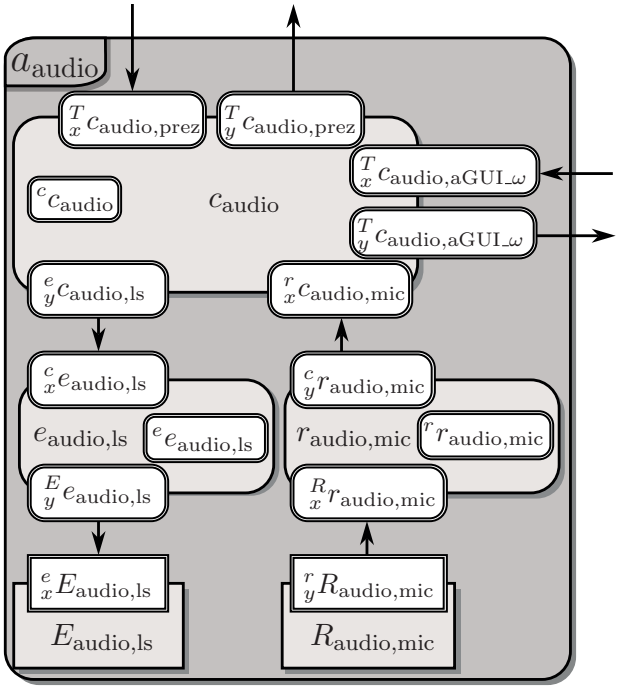

Fig. 12. Structure of the audio agent a_{audio}, where $\omega \in \{\mathrm{init}, \mathrm{user}, \mathrm{cmd}, \mathrm{ctrl}\}$

effector $e_{\mathrm{audio,ls}}$ controls the real effector $E_{\mathrm{audio,ls}}$, which represents a computer speaker or a stand alone sound system.

The audio module, and more specifically the control subsystem c_{audio} of the agent a_{audio}, enables the system administrator to create and manage models of commands and users, and enables the user to control the NCP using voice commands. The administrator influences the audio module by communicating with the agent a_{OID} using its receptors. The agent a_{OID} uses its effectors to interact with the audio module window agents a_{α} using stigmergy. Stigmergy has to be used, as the source of the stimulus is the operator.

The activities of window agents a_{α} consist in acquiring data from the real receptors monitoring the environment (e.g. window buttons), sending the relevant information to the agent a_{audio}, receiving data from a_{audio} and displaying it using its real effectors (appropriate message boxes).

The graph of the FSM presented in Fig. 13 defines what behaviours the control subsystem c_{audio} of agent a_{audio} executes. With each state of the FSM a single behaviour is associated. In the states ${}^{c}\mathcal{S}^{\mathrm{init}}_{\mathrm{audio}}$ and ${}^{c}\mathcal{S}^{\mathrm{manage}}_{\mathrm{audio}}$ c_{audio} waits for messages produced by the agent $a_{\mathrm{aGUI\_init}}$. Depending on the received message, the FSM of c_{audio} chooses the next state. In states ${}^{c}\mathcal{S}^{\mathrm{control}}_{\mathrm{audio}}$, ${}^{c}\mathcal{S}^{\mathrm{recordings}}_{\mathrm{audio}}$, ${}^{c}\mathcal{S}^{\mathrm{users}}_{\mathrm{audio}}$, ${}^{c}\mathcal{S}^{\mathrm{commands}}_{\mathrm{audio}}$ or ${}^{c}\mathcal{S}^{\mathrm{learning}}_{\mathrm{audio}}$ it executes the behaviour triggered by the delivered message and after that returns to the state it had been previously in. Behaviours create and destroy window agents a_{α} – causing their windows to appear on or disappear from the screen. Thus in ${}^{c}\mathcal{S}^{\mathrm{control}}_{\mathrm{audio}}$ the agent $a_{\mathrm{aGUI\_cmd}}$ is created, enabling voice command recognition, i.e. continuous audio acquisition, speech detection and analysis of each isolated portion of speech in order to recognise the speaker and the command. Depending on whether the speaker has been

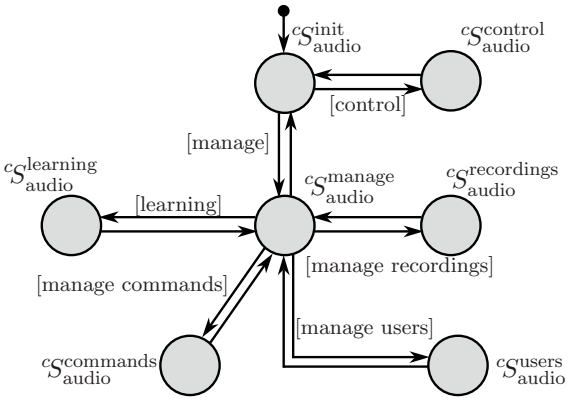

Fig. 13. The FSM switching behaviours of the control subsystem c_{audio} of the agent a_{audio}

detected or not, a dedicated command model is used (either a command model for the recognised user or a general command model). When the behaviour associated with the state ${}^{c}\mathcal{S}^{\text{control}}_{\text{audio}}$ terminates, the agent $a_{\text{aGUI\_cmd}}$ disappears from the audio module. The behaviour associated with ${}^{c}\mathcal{S}^{\text{manage}}_{\text{audio}}$ is responsible for deciding what types of resources the agent a_{audio} will manage. The behaviour associated with:

- ${}^{c}\mathcal{S}^{\text{commands}}_{\text{audio}}$ creates the agent $a_{\text{aGUI\_cmd}}$ enabling the management of the configuration files containing commands,
- ${}^{c}\mathcal{S}^{\text{users}}_{\text{audio}}$ creates the agent $a_{\text{aGUI\_user}}$ enabling the management of the configuration files containing user data,
- ${}^{c}\mathcal{S}^{\text{recordings}}_{\text{audio}}$ creates the agent $a_{\text{aGUI\_ctrl}}$ conducting a recording session for each of the selected users,
- ${}^{c}\mathcal{S}^{\text{learning}}_{\text{audio}}$ creates the agent $a_{\text{aGUI\_ctrl}}$ enabling the learning process, i.e. automatic creation of the user and command models based on previously acquired voice samples (during recording sessions with the participation of the users).

The recording session requires recording n samples (repetitions) for all commands loaded from the configuration file, where n is the number of samples defined by the administrator of the audio module. During the recording session, c_{audio} communicates with the virtual receptor $r_{\text{audio,mic}}$ in order to acquire the audio signal and with the agent's control subsystem $a_{\text{aGUI\_ctrl}}$ in order to transmit the recorded sample oscillogram, and also with the virtual effector $e_{\text{audio,ls}}$ to replay the recognised command using the speakers, and finally it creates a *wave* file with a recorded sample being part of the internal memory of the control subsystem c_{audio}.

The learning process is performed almost exclusively by the control subsystem c_{audio}, which includes the functions necessary for decoding data from the *wave* format, pre-analysis and parametrisation of the speech signal, creating user and command models, and translating the created models into appropriate files. Encoded data in the *wave* format is sent to the control subsystem c_{audio}, which subsequently stores it within its internal memory. If necessary, c_{audio} can read the data from its internal memory and send it to the virtual receptor $r_{\text{audio,mic}}$, which decodes the data represented in the *wave* format into the form of a vector of signal samples (numbers), and in this form it stores the data while processing it.

3.6. Vision module The vision module consists of three agents (Fig. 5). The main one, a_{vision} (Fig. 14), is responsible for the gesture recognition task. It communicates with the a_{vGUI} and a_{vDB} (Fig. 15), which are responsible for the graphical user interface and database management respectively.

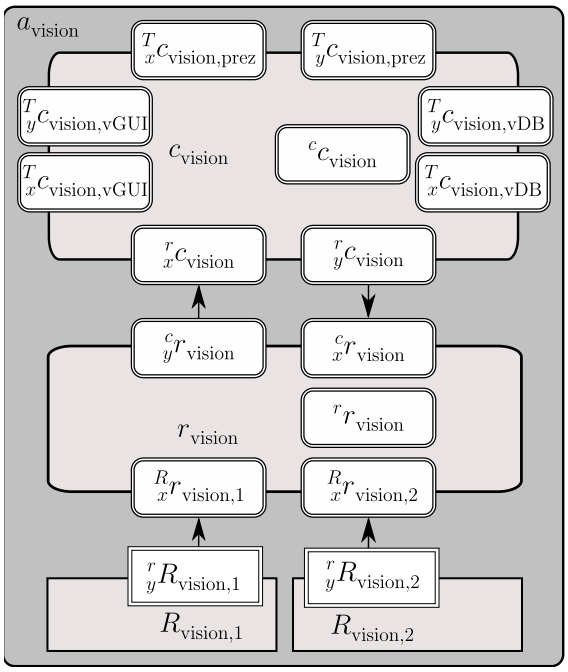

Fig. 14. Structure of the agent a_{vision}

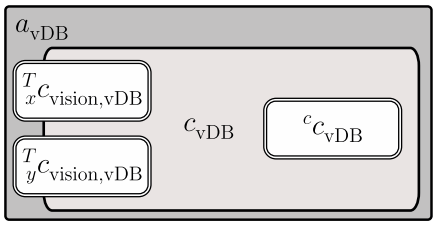

Fig. 15. Structure of the agent a_{vDB}

The agent a_{vision} contains the control subsystem c_{vision}, virtual receptor r_{vision} and two real receptors $R_{\text{vision,1}}$ and $R_{\text{vision,2}}$, being two RGB cameras in a stereo configuration. The virtual receptor r_{vision} is mainly responsible for preprocessing images obtained from the cameras $R_{\text{vision,1}}$ and $R_{\text{vision,2}}$. However, it also can input the calibration data from the control subsystem c_{vision}.

The database agent a_{vDB} is a purely computational agent, thus contains only the control subsystem c_{vDB}. The window agent a_{vGUI} assumes the role of the user interface producing a window. It possesses an effector and a receptor, similarly to other window agents a_{α} (Fig. 6). The agent a_{vision} communicates also with the agent a_{prez}, in order to deliver to it the recognized gestures.

The major activities of the agent a_{vision} are represented by the FSM (Fig. 16) of its control subsystem c_{vision}. The behaviour associated with the state ${}^{c}\mathcal{S}^{\text{init}}_{\text{vision}}$ initiates the commu-

nication with a_{vDB}. This behaviour gets the gesture models and other necessary information (e.g. camera calibration parameters) from the database agent a_{vDB}. After proper initialization the FSM proceeds to subsequent states, in which the agent locates the face (${}^{c}\mathcal{S}^{\text{face-localization}}_{\text{vision}}$) and the palms of the operator (${}^{c}\mathcal{S}^{\text{palm-localization}}_{\text{vision}}$). When the operator's face and palms are located the process of tracing them can be initiated. For that the FSM transits to the state ${}^{c}\mathcal{S}^{\text{gesture-recognition}}_{\text{vision}}$, with which a behaviour recognizing and interpreting gestures is associated. In the case of palm or face tracking failure, the FSM switches to one of the previous states, in which localisations of the palms and the face are conducted once more.

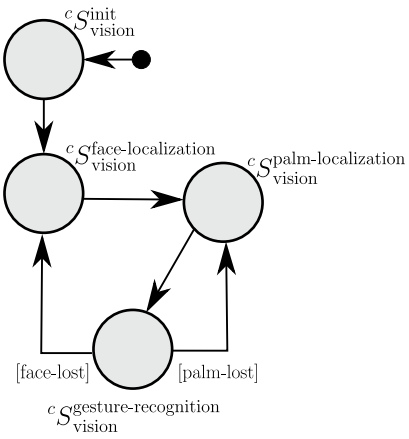

Fig. 16. The FSM switching behaviours of the control subsystem c_{vision} of the of agent a_{vision}

The FSM of the control subsystem c_{vGUI} of the agent a_{vGUI} has only one state. The behaviour associated with that state displays images sent by a_{vision}. Moreover, it sends to a_{vision} the requests resulting from the interaction through stigmergy with the agent a_{OID}.

Database agent a_{vDB} is a common storage for trained classifiers and calibration parameters. This information is retrieved by the vision agent a_{vision} during the startup. Agent a_{vDB} has only one state, in which it responds to all the data queries.

4. Implementation of the interface

The interface, as any software executed on a computer, must interact with its operating system, thus at the implementation stage this interaction must be defined. Moreover, the implementation defines the specific algorithms used to process data. In the case of the designed interface these are the voice and gesture recognition algorithms.

4.1. Interaction of the interface with the operating system. One of the fundamental implementation decisions, that must be made when designing a robotic system, is the determination of the number of computers that need to be used to deliver the necessary computational power to execute the task that the system is put to. In the case of a relatively low computational power requirements, the code of all agents included in the designed system can be located in one computer. This is the case of the NCP interface. Figure 17 presents the general principle governing the interaction between the agents (inter-agent communication) and between their subsystems (intra-agent communication), where all agents and subsystems are executed on the same computer. For the purpose of clarity only two agents are presented. They are

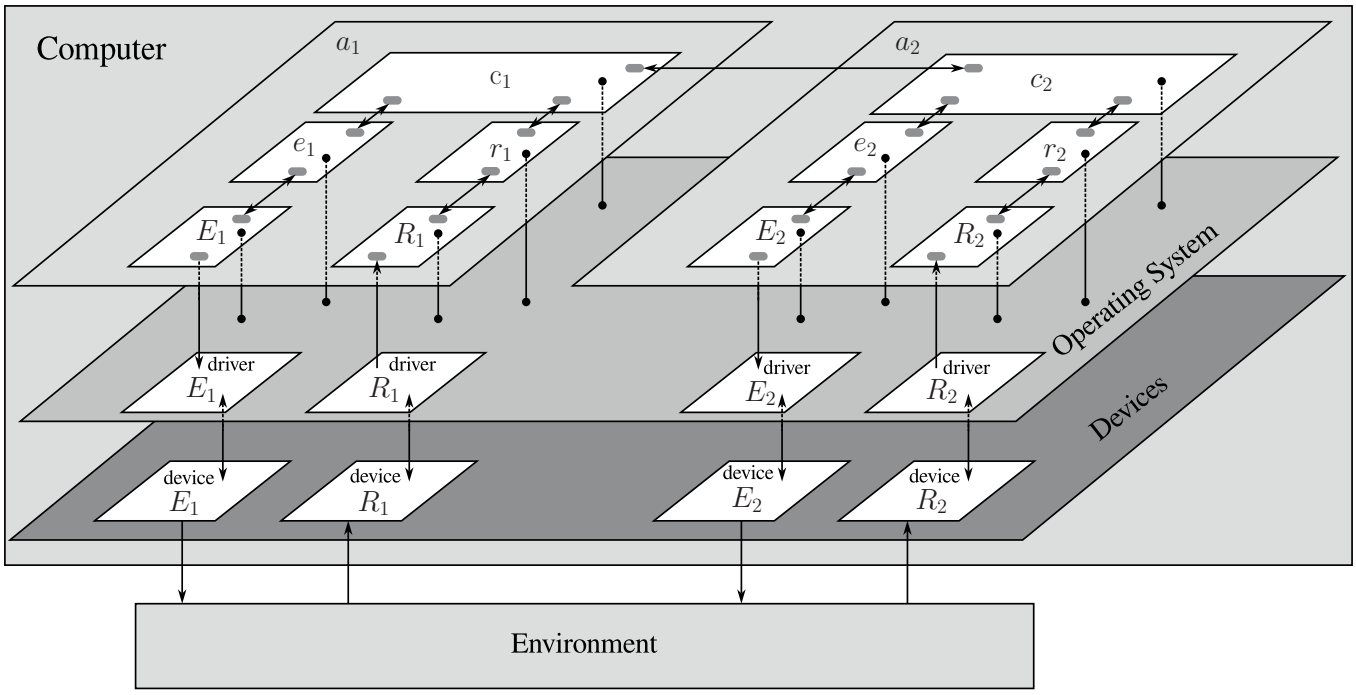

Fig. 17. Diagram of inter-agent and intra-agent communication used in the NCP interface

represented by a_1 and a_2, each having just one receptor and one effector. In the case of all agents residing on the same computer both the intra-agent and inter-agent communication takes place via the operating system. This figure also shows the interrelation between the subsystems and the drivers of devices, and between the drivers and hardware devices used by the NCP interface. The implementation assumes the creation of a three-layer structure, in which the upper layer is directly defined by the specification. The middle layer corresponds to the operating system. This layer consists of both the device drivers, being parts of real receptors and real effectors, as well as the software that organizes the transfer of information between threads and processes. The lowest layer consists of hardware devices, also being part of real receptors and real effectors. Those devices can also be computer based, thus they can contain software created by the hardware manufacturers, however that is hidden from the system designer.

4.2. Audio and speech analysis. The algorithms for audio signal processing are performed by the audio agent a_{audio}. It consists of the: control subsystem c_{audio}, virtual receptor $r_{\text{audio,mic}}$, virtual effector $e_{\text{audio,ui}}$, real receptor $R_{\text{audio,mic}}$ and real effector $E_{\text{audio,ui}}$. The virtual receptor $r_{\text{audio,mic}}$ listens to the incoming signals from the microphone $R_{\text{audio,mic}}$, when commanded to do so by c_{audio}. It performs signal acquisition, digitisation and recording, i.e. signal activity detection. The virtual receptor $r_{\text{audio,mic}}$ starts to record the signal when its amplitude exceeds the noise level and stops when a sufficiently long pause appears, or the recording buffer is full. Finally the buffer of digitised signal samples is sent by $r_{\text{audio,mic}}$ to the control subsystem c_{audio}. The control subsystem c_{audio} uses the digitised form of those signals in the speech pre-processing and feature detection pipeline. Besides the speech pre-processing and feature extraction (in short: speech parametrisation), the control subsystem of the audio agent performs also other speech-related behaviours (Fig. 13): encoding the digital speech signal to the Wave format and decoding it from this format (both done by the behaviour associated with state ${}^{c}\mathcal{S}^{\text{recordings}}_{\text{audio}}$), creating spoken command/phrase models using recorded examples (${}^{c}\mathcal{S}^{\text{learning}}_{\text{audio}}$), creating speaker models out of recorded examples (${}^{c}\mathcal{S}^{\text{learning}}_{\text{audio}}$), recognizing spoken commands/phrases and identifying registered speakers (${}^{c}\mathcal{S}^{\text{control}}_{\text{audio}}$). It also handles the input/output of speech models from/to the operating system file system. In the state ${}^{c}\mathcal{S}^{\text{learning}}_{\text{audio}}$ the model is created, thus it has to be memorized, hence input is required, while in the state ${}^{c}\mathcal{S}^{\text{control}}_{\text{audio}}$ this model has to be retrieved, thus input is necessary. The following subsections give a general explanation of the speech parametrisation pipeline used by the behaviours of the control subsystem c_{audio}.

4.2.1. Audio signal parametrisation. Both spoken command recognition and speaker identification tasks use the same speech pre-processing and feature detection pipeline (Fig. 18). As a result standard MFCC-based feature vectors are produced. In the FSM state ${}^{c}\mathcal{S}^{\text{control}}_{\text{audio}}$ (Fig. 13) the associated behaviour processes in real time a signal chunk containing an isolated spoken command. In the state ${}^{c}\mathcal{S}^{\text{learning}}_{\text{audio}}$ (Fig. 13) the chunk consists of a single wave-file content. Thus those behaviours use the transition functions executing the same computations, but on different data sources.

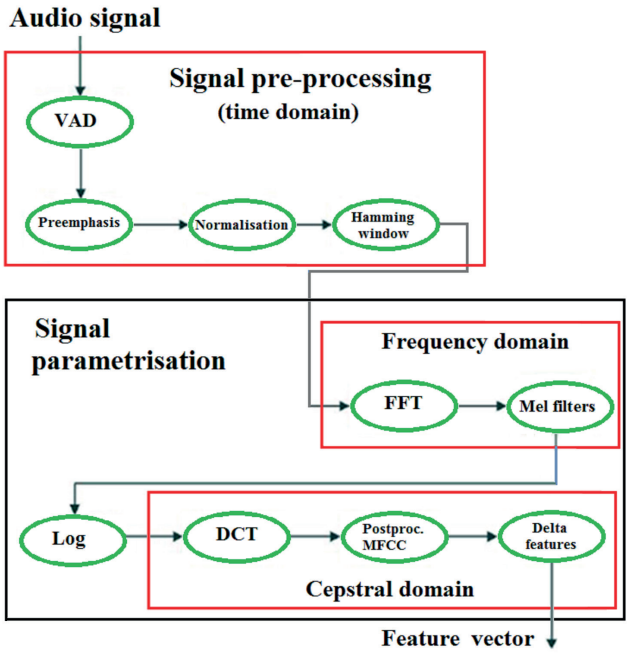

Fig. 18. The audio signal pre-processing and parametrisation pipeline

The first four steps of parametrisation are performed in the time domain: voice activity detection (VAD), preemphasis filtering, amplitude normalisation of the entire signal chunk, signal segmentation into uniformly distributed frames and the multiplication of frames by a Hamming window. Every frame is transformed by a Fast Fourier Transform into the frequency domain. In this domain the amplitude vector of Fourier coefficients is band-filtered by a set of triangle band-passing filters distributed according to the Mel frequency scale. After mapping the obtained feature vector values to a logarithmic scale the inverse DFT (in fact the inverse cosine transform only, as the data is real-valued only) is applied to every feature vector, leading to cepstral features. A subvector of the first 18 elements, called MFCC coefficients, in the cepstral domain is selected next. After postprocessing (the so-called liftering and long-time mean subtraction) gradients of the MFCCs with respect to time are obtained and the final feature vector is formed. It consists of 38 elements: frame total energy and delta of such energy, 18 MFCCs and 18 delta MFCCs.

Although, due to license reasons, we decided to make our own implementation of the feature detection pipeline, for an interested reader several existing open-source libraries can be recommended. The early speech recognition framework `sphinx-4` (Sun Microsystems, 2004) [34] contains, among others, a Java language implementation of functions for speech signal feature extraction. A more recent project `KALDI` [35] has appropriate functions implemented in C++. The library `Loudia7` [36] also can be used (under a GPLv3 licence). If the speaker recognition toolkit `ALIZE` [37] is to be used, it works

with the speech signal processing toolkit `SPro` [38] – its version 5 has a BSD license, while later versions, an MIT license.

4.2.2. Spoken command modelling and recognition. It has been assumed that spoken command recognition suits the isolated speech recognition paradigm. The command dictionary is limited to 20–50 items. Thus, the audio module deals with a "small" speech recognition problem. Additionally, as the command recognition task is associated in the same module with the speaker recognition task, it is assumed, that some training samples of the commands, spoken by the trusted speaker, will be recorded. Thus, a "closed" dictionary can be assumed here and no phoneme coder nor phonetic transcription module need to be implemented. This assumption leads to a simplified structure of the spoken command recognition system (Fig. 19), compared to a more universal solution for an open dictionary case. The command models take the form of feature maps, while the recognition process utilizes a modified DTW (dynamic time warping) algorithm.

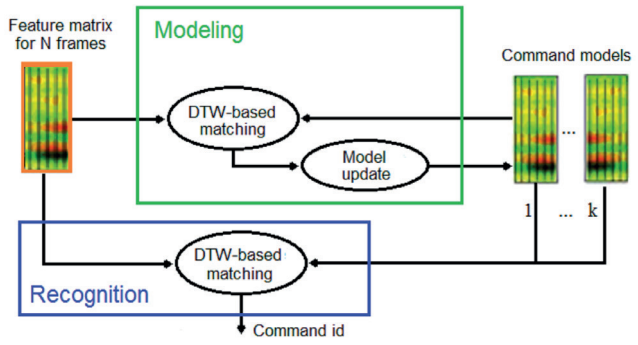

Fig. 19. Structure of spoken command modelling and recognition based on feature matrices and DTW matching

4.2.3. Speaker voice modelling and recognition. Typical speaker recognition approaches emerge from pattern recognition and machine learning domains. They employ computational techniques such as GMM clustering, SVM classification, stochastic factor analysis, probabilistic LDA and neural networks [39]. The basic approach, called UBM-GMM, finds GMM mixtures in the feature vector space to represent a universal background model for all speakers and models of individual, selected speakers to be identified [40]. The GMM-SVM approach operates on supervectors of features, resulting from a composition of cluster representatives, to enable the SVM classifier to learn. "Joint Factor Analysis", "i-vectors" and PLDA methods are based on factor analysis, performed in the space of feature supervectors (the first two methods) or i-vectors (the PLDA method). Recently, Deep Neural Networks are demonstrating their applicability to speaker recognition [41].

The implementation of a speaker recognition module can be based on the open source library provided by the project ALIZE [37], developed at the University of Avignon under the GNU Lesser General Public License (LGPL). Besides the basic UBM-GMM implementation, the library contains also more advanced techniques such as SVM, Nuisance Attribute Projection and Factor Analysis. However, the presented audio module contains our own implementation of the standard UBM-GMM approach (Fig. 20).

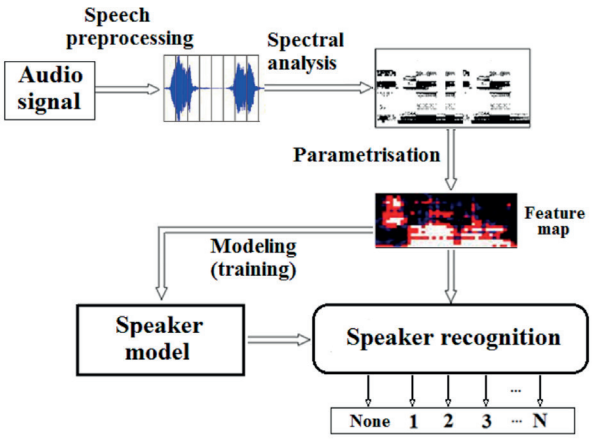

Fig. 20. The UBM-GMM approach to speaker modelling and recognition

4.3. Vision module implementation. The agent a_{vision} is responsible for the whole image processing and gesture recognition task. Its virtual receptor r_{vision} aggregates and processes the data gathered from the cameras. In the processed images (with calculated depth information) the operator's palms are tracked and their gestures are recognized by the control subsystem c_{vision}. Those are subsequently sent to the presentation module to trigger imitations of mouse and keyboard events.

4.3.1. Image rectification. The virtual receptor r_{vision} uses the calibration parameters (acquired from the control subsystem c_{vision} of the agent a_{vision} provided to it by the agent a_{vDB}) to process images acquired by the cameras. First, the distortion is removed from the images, and then both pictures are rectified in such a way, that a given point in space is mapped to the same vertical coordinate in both images.

4.3.2. Depth estimation. Based on two rectified images, using the SGBM algorithm (*Semi Global Block Matching*, [42]), the disparity map is calculated. The calculated differences in pixel positions in the left and right image can then be used to estimate the distance from the camera to each point (using the calculated disparity value). Having the z coordinate and camera intrinsic parameters the x and y coordinates can then be calculated.

4.3.3. Face and hand regions detection. To detect the operator the face detection algorithm is employed. For this, the well known and widely used cascade detector is used [43]. To make the process robust to dynamic background (with other people moving around) out of all detected faces only the one closest to the camera is taken into account. If the face was properly localized in the previous frames, instead of time consuming redetection, it is only tracked. In the region where the face was detected

the features of the face are subjected to a matching process, i.e. eyes, nose, lips and chin are used [44]. This information is then utilised to estimate the face oval, and its interior is used to calculate the operator's skin color model. The obtained skin color model is used to segment the input image in order to find the palms. In this case, to initialize the tracker, the operator has to be in the calibration pose (palms at the head height) for one second. After that both palms are tracked and re-detection is only required, if tracking fails.

4.3.4. Static gestures. Static gestures are used by the interface to imitate mouse events, e.g. open hand is the "neutral" gesture, thumb up or down can trigger scrolling, fist triggers mouse drag (holding left mouse button down) etc. Currently the interface is prepared to recognize six static hand gestures: open hand, fist, thumb up and down, pointing with one or two fingers (V shape). As the control subsystem detects and tracks palm regions, it is not necessary to implement another palm detection algorithms, only the classifier is required. For this task, the CNN (convolutional neural network) is used. This technique has proven its efficiency in multiple vision-related tasks, from image classification and object detection [45], through face recognition [46], to medical applications [47].

The CNN structure resulted from the fact that we had started from the exemplary network (TinyDarknet), provided by the author of the Darknet library [48], and subsequently modified it to fit our needs. Palm region, cut out of a larger image, is resized to a 50×50 pixel area and is used as the input to the network. Input dimensions depend on the actual size of the palm in the image. As the operator stands about 3 meters away from the camera, the palm in the image is smaller than 100 px. Another problem is the size of the training dataset. As this dataset is of limited size, we have to limit the number of trainable network parameters. The network contains 7 convolutional layers. Instead of the final fully connected layer another convolution network is used (with six filters), where each filter corresponds to one gesture. Six planes produced by this layer are fed through the global average pooling layer, producing the volume of size 6×1. Final network output is produced by the softmax layer (Fig. 21). To train the classifier about 3000 palm images were used. The data was also further augmented (rescaled, sheared etc.).

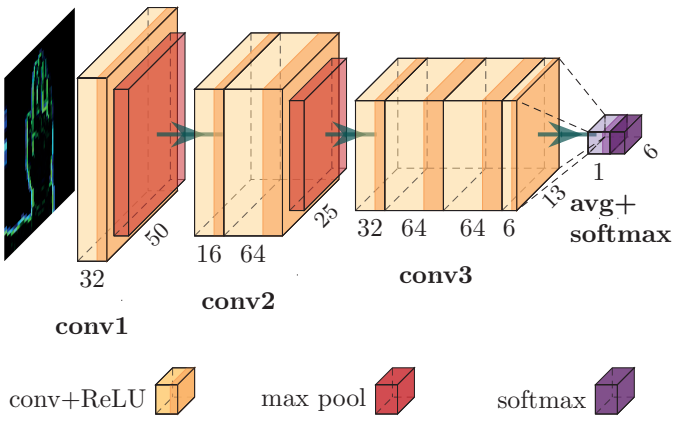

Fig. 21. The structure of the neural network classifying static gestures

4.3.5. Dynamic gestures. Apart from static gestures, it is also possible to use sequences of palm positions treating them as a dynamic gesture (e.g. waving the hand to the left or right). To recognize such gestures and to find the appropriate reference trajectory in the training set the DTW (dynamic time warping) algorithm was used [49]. Training of dynamic gestures is much easier than the static ones. Only a small number of reference trajectories per gesture is required (currently 5). The recognized dynamic gesture identifiers are sent to the presentation module, where they are translated into, for example, back and forward keyboard codes.

Dynamic gestures are interpreted "on demand", i.e. the operator has to keep the right palm out of the work area and only the left one is active. From that moment on, every position of the left palm is recorded and stored in the buffer with a capacity to store at the most 50 last positions. After receiving a new frame the whole buffer is interpreted as a trajectory and DTW is used to find the closest reference gesture. After finding the match with the score lower than the defined threshold, the gesture identifier is sent to the presentation module. To avoid multiple triggers of the same gesture in subsequent frames, there is a one second delay after successful recognition (new positions are recorded, but the gestures are not interpreted).

5. Tests of the interface

The tests of the interface focused on voice and gesture command execution, conducted both separately and in conjunction on an integrated system.

5.1. Spoken command and speaker recognition tests. The audio module has been tested in two scenarios: off-line tests in a dominantly "closed set" scenario (only commands or speakers are recognized that are registered and modelled by the audio module), and on-line tests in an "open set" scenario (both registered and unregistered commands and speakers can appear). Every registered speaker will be called to be *genuine*, while an unregistered – an *imposter*.

5.1.1. Audio sample sets. Initially, samples were recorded by 3 stationary microphones and (two different) mobile microphones in two sessions. In the first one, genuine speakers (11 persons: 8 men and 3 women) were recorded, while in the second – imposter speakers (9 persons – all men). Later, in the third recording session, samples of 22 commands were recorded by a mobile microphone, coming from 3 speakers. Let us denote the three datasets as follows:

- **set 1:** training and test samples of 10 commands, acquired by stationary microphones. Let us further split this set into two subsets: **subset 1A** contains genuine speaker samples, while **subset 1B** is composed of imposter speaker samples;
- **set 2:** training and test samples of 10 commands recorded by mobile microphones. Let us further split it into **subset 2A** that contains genuine speaker samples and **subset 2B** filled with imposter speaker samples.
- **set 3**: training and test samples of 22 commands recorded by a mobile microphone.

For the subset 1A (genuine speakers, stationary microphones) for every genuine speaker and every command, 12 utterances were recorded by stationary microphones (at least two times for every of the 3 different microphones used). For subset 2A (genuine speakers, mobile microphones other 9 utterances (3 training and 6 test samples) were recorded by on-head mobile microphones. In subset 1B (imposter speakers) and subset 2B each, 6 samples per command and speaker were included. Thus, the first set (1A + 1B) contained 660 training and 660 test samples, coming from 11 genuine speakers, and 540 test samples – from imposter speakers, while the second set (2A + 2B) contained 330 training samples and 660 test samples from genuine speakers, and another 540 test samples from imposter speakers.

While the purpose of sets 1 and 2 was to test and verify the performance of both solutions – command recognition and speaker recognition – for many different users, set 3 was dedicated to obtain a better statistics of the command recognition performance for a larger number of commands and samples per command, recorded by the mobile microphone.

The ten commands were composed of two Polish words each, e.g.: *obraz ogólny* (general image), *najważniejsze zagrożenia* (most important hazards), *ostatnie zgłoszenia* (last submission). Commands from 7 to 10 had one word in common ("sector") differing only by one or two syllables of the second word (*transportowy, medyczny, bankowy, rządowy*).

In the studied scenario, the goal of command recognition is to support a user in the selection of choices displayed in a window appearing on the screen (e.g. selecting folders within a window or picking window items and manipulating them). Obviously, the commands are application-dependent and one dictionary may be completely replaced by another, if another window type is visualised, but the number of choices for most computer windows should be similar. We assume that distinguishing around 20 commands per window should be sufficient for a given application.

5.1.2. Spoken command recognition (off-line tests). Every one of the three sample sets 1A, 2A and 3 (for genuine speaker samples) was split into two parts: the training subset and the test subset. The split into training and test samples was as follows: for 12 samples of every command in 1A the split was 6:6, for 9 samples of every command in 2A, the split was 3:6, for 20 samples of every command in dataset 3, the split was 6:14. The subsets 1B and 2B (coming from imposter speakers) contain additional test samples only. Models of spoken commands were created, based on the training samples in this set obtained from all the speakers.

The recognition procedure has to consider both a closed set (test samples come from the limited set of commands) and an open set case (test commands other than modelled can appear). This leads us to a decision rule that combines two conditions, each one controlled by a different threshold value. First, the possible winner among modelled commands is selected, on the basis of a "softmax" rule (controlled by a belief threshold). Next, the winner is conditionally accepted, if the matching quality between the test sample and the selected model is higher than the score threshold.

In the following, we report the maximum recognition rate - obtained when there is no quality score threshold applied (appropriate for the closed-set case) and give also detailed statistics related to the distribution of the score threshold.

The spoken command recognition performance is evaluated by the use of typical criteria: ROC and CMC (cumulative match curve) are shown and EER (equal error rate) point is found at the intersection of curves representing FRR (false acceptance rate) and FAR (false rejection rate). In Fig. 22 this is shown for the test set 1A, while in Fig. 23 for the entire set 1 (1A + 1B).

Using the 10-command model, established for the training subset in dataset 1A, we obtained a 94.2% rank-1 recognition rate for the test subsets of 1A (stationary microphones, genuine speakers only), When test samples coming from unregistered (i.e. imposter) speakers (subset 1B) were added, this rate has dropped to 86.0%.

In Fig. 24 command recognition results are shown for the test set 2A, while in Fig. 25 for the entire set 2 (2A + 2B). With a 10-command model, established for the training subset in dataset 2A (the mobile microphone samples), we obtained a rank-1 recognition rate of 92.6%. When imposter samples from subset 2B were added, the rank-1 recognition rate dropped to 85.2%.

The point EER was found at 15.5% and 24.8%, for both cases of the stationary microphone tests, or 15.8% and 25.4%, for both tests of the mobile microphone.

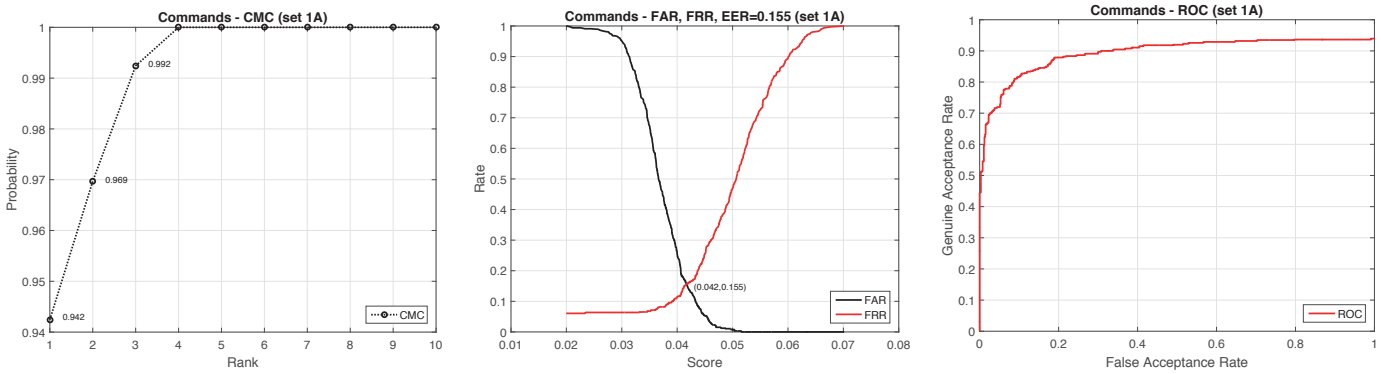

Fig. 22. Spoken command recognition results for the subset 1A of audio samples (stationary microphones, genuine speakers only) – rank-1 accuracy is 94.2%

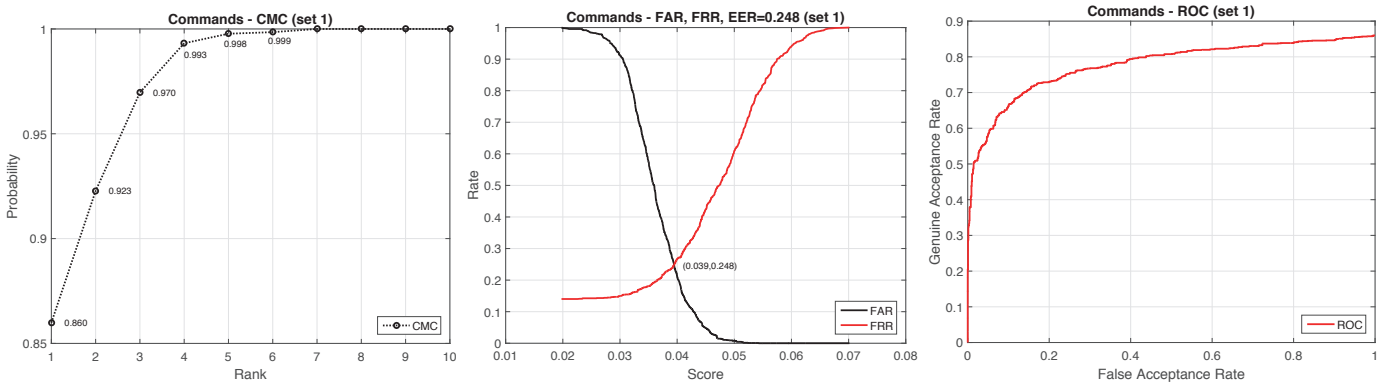

Fig. 23. Spoken command recognition results for the entire set 1 of audio samples (stationary microphones, both genuine and imposter speakers) – rank-1 accuracy is 86.0%

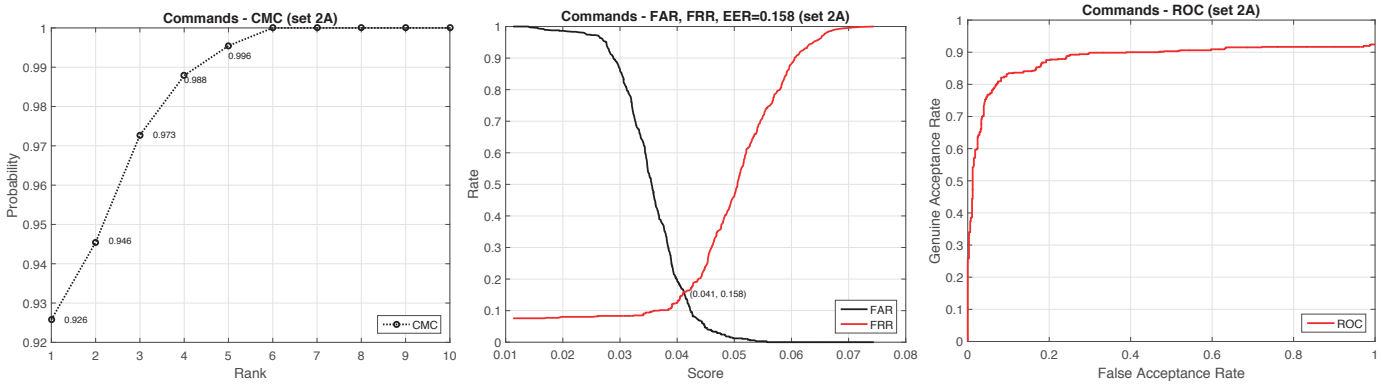

Fig. 24. Spoken command recognition results for the subset 2A of audio samples (mobile microphones, genuine speakers only) – rank-1 accuracy is 92.6%

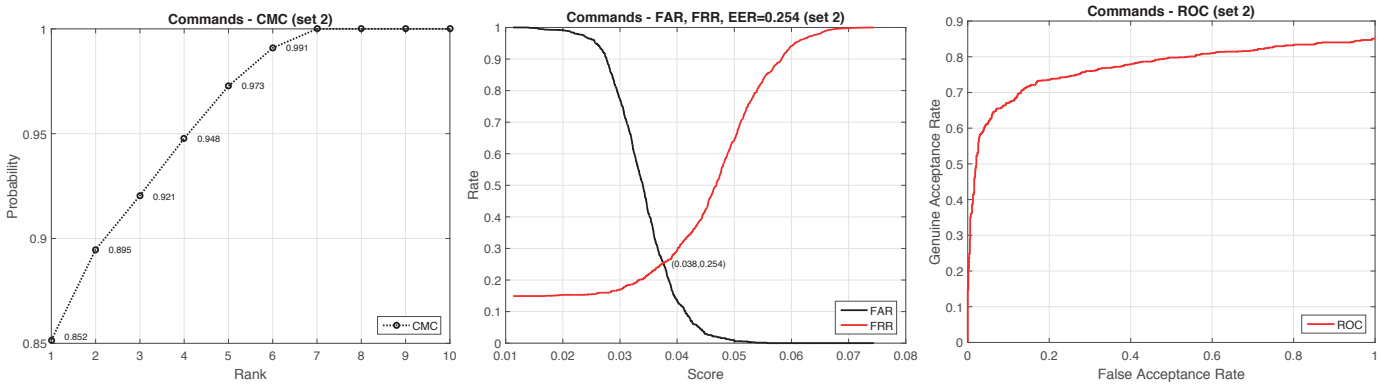

Fig. 25. Spoken command recognition results for the entire set 2 of audio samples (mobile microphones, both genuine and imposter speakers) – rank-1 accuracy is 85.2%

The recognition errors mainly stem from commands no. 7 and no. 10, which differ from the commands no. 9 and no. 8 only by single syllables. These dependencies between commands lead to an empiric parametrisation of class weights for the sensor fusion process.

The influence of a higher number of registered commands on the recognition rate can be illustrated by results obtained for the dataset 3 (Fig. 26). The rank-1 recognition rate reached is 90.6%, while the EER point was established at 0.176 (i.e 17.6%). Both numbers represent only a small drop in the quality of the results, when compared with the results obtained for the 10-command dictionary.

The above test results lead us to the strategy of setting the score threshold sufficiently high, in order to reduce the expected false acceptance rate to, let us say, 1%. Then the false rejection rate is expected to be in the range of 25–30%. This requires from the user careful pronunciation of commands and sometimes their repetition (infrequently).

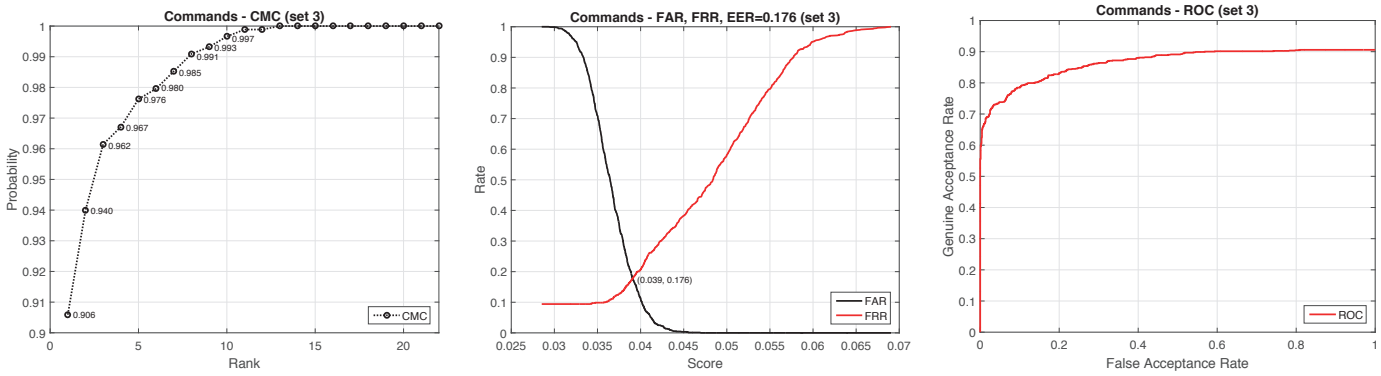

Fig. 26. Spoken command recognition results for the dataset 3 of audio samples (mobile microphones, 22 commands) – rank-1 accuracy is 90.6%

5.1.3. On-line command recognition tests. The on-line recognition tests were conducted under live signal acquisition conditions by three unregistered persons. They have not contributed to any of the models – neither the command nor the speaker model. Every unregistered user has spoken every command 10 times to every of the three microphones (two stationary and one mobile).

In this "closed-set" case ("closed" with respect to the command dictionary) the acceptance rate observed, when the acceptance threshold was set according to the EER point, was 82% for the stationary microphones and 73% for the mobile one. This is in line with the command recognition rate of ca. 75% observed for the EER point in the off-line tests with datasets 1 and 2.

5.1.4. Speaker recognition results. Preliminary tests have shown, that making speaker recognition decisions on the base of a single command leads to a relatively high error rate. Thus, a procedure that accumulates speaker identification scores over the time of several commands was implemented. In particular, the first decision is made after the evaluation of 3 commands. Up to 10 past command scores are accumulated by the decision procedure. Hence, when 30 genuine test samples are available for every single speaker, up to 28 identification decisions will be generated.

The voice recognition procedure is of type "classification with an open set of speakers". It first selects the winner from among possible registered candidates and unknown speakers represented by the UBM. Now a relative distance threshold is used in the final acceptance step. Thus, three possible results of a single recognition process are distinguished: *failure, no decision, acceptance*. If for a given test sample the selected best model is in fact the true genuine speaker model, then a *failure* cannot be generated. But still a *no decision* result is possible. In order to get the *acceptance* answer, the actual distance of current sample to the selected genuine speaker model must be shorter than the distance to the UBM model by several percent. In practice, we also experimented with a second distance threshold, created for a "softmax"-like decision. Here, the distances of the current sample to those of all of the registered users are combined into a single distribution and a relative distance to the winning model must significantly differ from the average distance.

In Fig. 27 speaker recognition results are shown for the test-set 1, while in Fig. 28 similar results are presented for the test-set 2. We are not presenting the results for the genuine subsets 1A and 2A, as they were perfect from the point of view of the recognition rate. With both genuine and imposter speakers the rank-1 (genuine) recognition rates were (again) 100% (for stationary microphones) and 91% (for mobile microphones). The EER points were found at 6.9% and 12.2%, respectively.

Relative scores are evaluated, between the genuine user score and the UBM model score (e.g. the ratio between the distance of the test sample to the UBM model and the distance of the

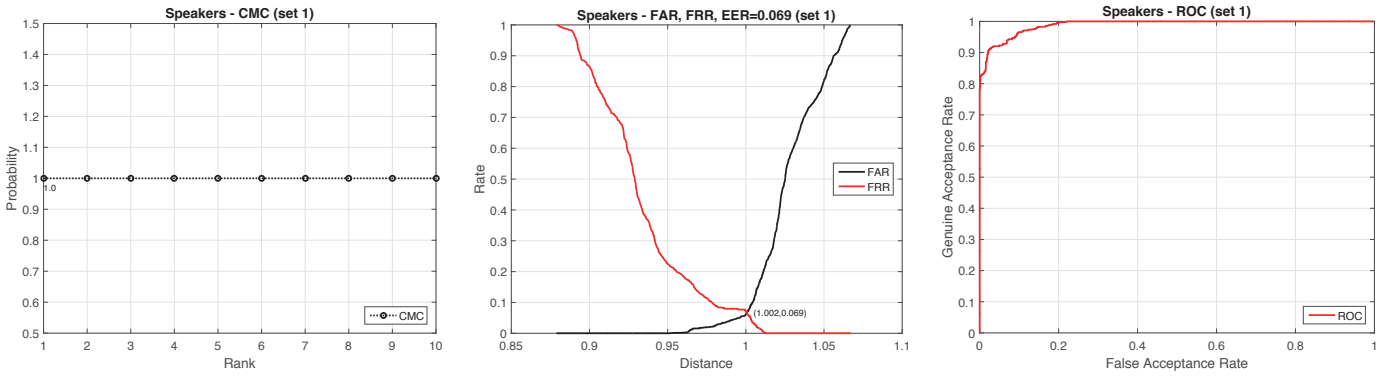

Fig. 27. Speaker recognition results for the dataset 1 of audio samples (stationary microphones, 11 genuine and 9 imposter speakers, modelling based on 10 commands) – rank-1 genuine accuracy is 100%

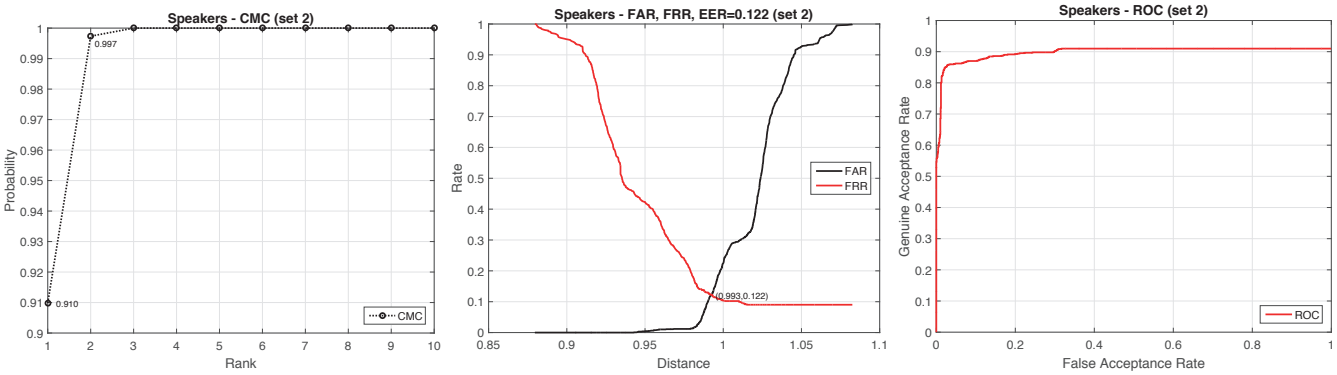

Fig. 28. Speaker recognition results for the dataset 2 of audio samples (mobile microphones, 11 genuine and 9 imposter speakers, modelling based on 10 commands) – rank-1 genuine accuracy is 91%

test sample to the speaker model). The average value for all test s amples is 1.059 for the test set 1 and 1.064 for the test set 2. In other words, the genuine scores exceeded the UBM scores by 5.9% – 6.4%. The relative distance of the "genuine" model to the "imposter" models has nearly always been greater than the one to the UBM distance. It ranged from 1.047 do 1.164, whereas the distance to UBM was found to be from 1.023 to 1.092.

5.1.5. Deep learning in speech recognition. Current research in audio signal processing is dominated by deep neural network techniques. We have started our work with a CNN deep network (e.g. VGG16) for spectrogram data processing, with the idea preferably to apply it in the context of speaker recognition. By *learning from scratch* or applying the *transfer learning* technique to an existing model (a CIFAR10 model [50] additionally trained on several hundreds of audiobooks, made available by the Librivox service embedding vectors have been created for either a complete command spectrogram of for selected part of sequences representing phonemes. For final speaker recognition, a multi-class SVM classifier has been learned using the *scikit-learn* package. The overall results were unsatisfactory: the system could learn only up to five classes (speakers), while the recognition rates for a closed-set of five speakers were well below the rates achieved by our classic approach for 11 speakers. For set 1 the best genuine recognition rate was 87% and 76% for test 2. This is worse by 13–15% than our classic case results, reported above.

In this work both audio and video modules were required to work robustly, while being trained on small data sets only, and had to be implemented without third-party libraries, in order to assure a practical verification of the proposed system design concept. Thus, classic techniques have been preferred here. Nevertheless, in our future work a phoneme coder is going to be developed, that makes the audio module more versatile and of universal use. For this purpose, we expect to apply a CNN network for phoneme classification and also use large speech datasets (for example Voceleb2 [51]).

5.2. Gesture recognition tests. The gesture recognition part of the system, recognizes dynamic gestures using pre-existing, well-tested solution. Because of that, this section focuses only on testing static gesture recognition (Fig. 29). The CNN was trained using a set composed of more than 3000 original gesture images. Testing was done using a set of other almost 2000 images. Both sets were acquired from three different operators in two recording sessions. The images had been gathered as a video sequence, what resulted in subsequent images being similar to each other. In consequence, training data had much smaller variability than it seemed from just looking at its size.

Fig. 29. Sample static gestures

The training was done with the Adam optimizer, with the learning rate of 0.001 and the categorical cross entropy loss. The training process is presented in Fig. 30. After 15 epochs the network reached more than 97% accuracy on the training set and scored 95.7% accuracy on the testing set. The confusion matrix for the testing dataset, along with the CMC curve, are presented in Fig. 31.

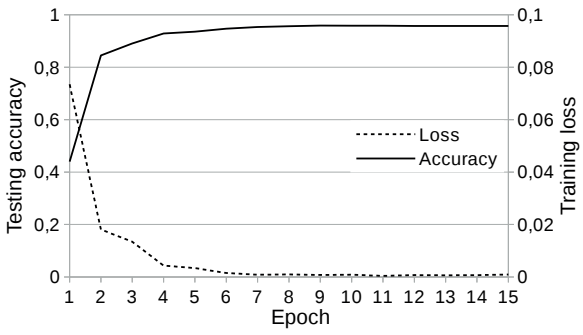

Fig. 30. Training process of the CNN

Apart from accuracy testing using image datasets, live tests were performed to confirm the proper functioning of the gesture

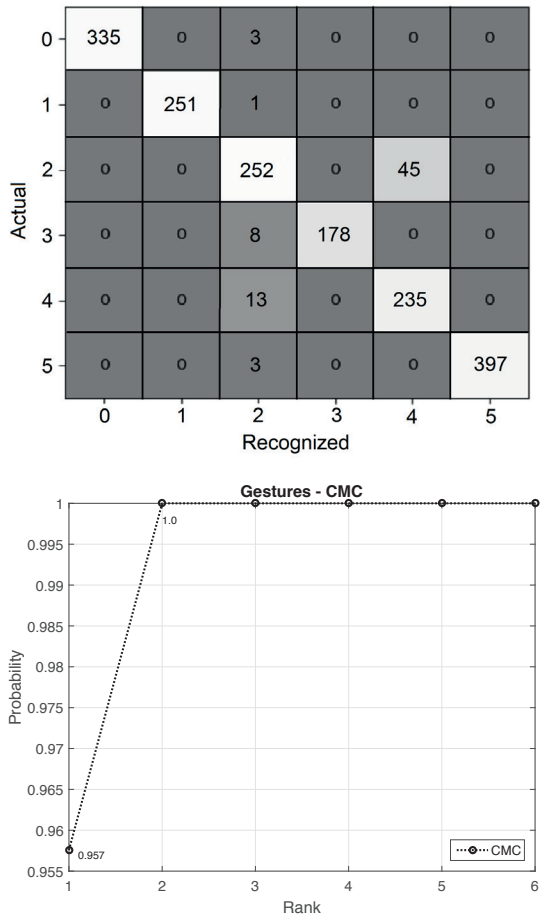

Fig. 31. Top: confusion matrix for static gesture recognition: rows present actual classes and columns represent recognition results. Bottom: CMC plot for the process (accuracy: 95.7%)

recognition system in connection with the presentation module. During this trial, the operator was asked to execute some operations, similar to those conventionally performed using a computer mouse. Those operations were, e.g., to navigate on a map (zoom, scroll, rotate), browse some web pages presented in multiple tabs and navigate between them, navigate in a text etc. All of the implemented functions worked as expected, but the users had some suggestions, pertaining mainly to the comfort of use. Standing in front of the screen with hands up for longer periods is tiring and the hands quickly become numb. This problem can be solved by defining the operating area in such a way that rising hands will not be necessary. Moreover, the gesture recognition mode of operation of the interface is not the only one intended for use by the operator. It is assumed that it will be used as supplementary input.

Another possible problem discovered during tests are spurious actions caused by unintended gestures done by the operator gesticulating while talking to others. This could be solved by checking whether the user is facing the screen or not, by either detecting the head orientation or gaze tracking techniques (e.g. [52]).

There is also a possibility to extend the vision system by providing face recognition software [53] to aid the operator identification process. Information from the vision system can additionally support voice recognition. It can be used to narrow down the choice of possible voice models, based on the recognized face (making the speaker recognition faster or even changing the task from recognition to only verification). It can also be used to keep track of the user's identity during periods of silence, so that operator change can be detected, which in turn can reset the user recognition system.

6. Conclusions

This paper presents a multimodal interface for controlling the visualisation of the security data collected by the National Cybersecurity Platform. The interface specification is based on an embodied agent methodology, hence demonstrating that the methodology used to design robotic systems is also useful for designing non-robotic ones. The network of cooperating agents renders the structure of the designed system open. This enables a relatively simple extension of the structure, and thus increases its capabilities, e.g. new modes of communication with the interface such as head or face-based gestures or other biometric modalities can be appended. In addition, the current modes can be extended by adding new voice commands and gestures.

The methodology based on embodied agents used to design the system proved its effectiveness. The resulting system specification, relying on the separation of concerns paradigm, significantly facilitated its implementation. The multi-layer system decomposition into: modules, agents, FSMs and behaviours simplified the division of labour between the members of the implementation team, thus reducing the time necessary to obtain a working system.

The advantage of having a multimodal interface to a cybersecurity platform, that employs recognition of speech signals and images, is the alleviation of the strain that the operators are subjected to during their daily work, which eventually leads to musculoskeletal disorders. Moreover, such an interface can authenticate the users. Currently only voice authentication has been implemented. Face recognition is subject to further work. In comparison to the use of customary input devices this biometric feature is a significant benefit.

Currently, the interface is subjected to testing. The experiments showed high effectiveness of the gesture recognition procedure. An innovative element of the audio module is that it combines spoken command recognition with trusted speaker recognition in one module. This enables the identification of the speaker, or just the determination whether the speaker is a man or a woman. Hence, the command recognition unit can use a dedicated model for every registered speaker or separate models for women and men. The test results clearly indicate that the effectiveness of the command recognition significantly increases when using a dedicated speaker model or an assigned gender model. Because we cannot expect to have large collections of samples obtained from different speakers, therefore, to improve the recognition rates (for spoken commands) and the score margins (in speaker recognition), the best parameter settings (e.g., number of features, feature weights) are chosen experimentally.

Currently the audio agent is being enhanced to make it more universal, i.e. applicable to other domains, and moreover, to limit the training sample acquisition requirements. A phoneme coder is being designed and a phonetic transcription and phonetic modelling of commands is under development. Deep-learning based techniques will be employed at acoustic level (i.e. for feature extraction) in command recognition, but also will be applied to speaker modelling and speaker recognition. Components of the current system (performing feature extraction and speaker modelling/recognition) are going to be replaced by ANNs – preferably CNNs and recurrent DNNs. Keyword-based speaker recognition is assumed to be fused with text-independent speaker recognition, whenever applicable in the given domain.

Gesture recognition system, in the presented form, is good enough for the task that it was initially defined for. The list of dynamic gestures is easily extendible by providing a few training samples of the new gesture. Static gesture classifier, on the other hand, relies heavily on big training datasets. This makes it hard to extend with new gestures, if the target operator decides to do so. One of the planned extensions is to use the intermediate stage for static gesture recognition, relying on hand-pose detection (detection of 21 hand joints for palm and fingers). Based on this we can build the actual gesture recognition on top of joint positions. For the first part (joint detection) we can use one of the available networks (e.g. Google Mediapipe). The second stage can be implemented as a pure classifier, with joint positions treated as a (small) feature vector. This approach would enable the operator to extend the system with new gestures by showing just a few examples (less than twenty is planned).

Acknowledgements. Work done as part of the CYBERSECIDENT/369195/I/NCBR/2017 project supported by the National Centre of Research and Development in the frame of CyberSecIdent Programme.

References

[1] W. Wang and Z. Lu, "Cyber-security in the smart grid: Survey and challenges", *Comput. Netw.* 57 (5), 1344–1371 (2013).

[2] W. Dudek and W. Szynkiewicz, "Cyber-security for mobile service robots–challenges for cyber-physical system safety", *J. Telecommun. Inf. Technol.* 2019 (2), 29–36 (2019).

[3] NCSC-UK, *The National Cyber Security Centre, United Kingdom*, https://www .ncsc.gov.uk/, [Online: accessed on May 5, 2019].

[4] NCSC-USA, *The National Cybersecurity and Communications Integration Center, USA*, https://ics-cert.us-cert.gov/, [Online: accessed on May 5, 2019].

[5] NCSC-NL, *The National Cyber Security Centre, Netherlands*, https://www.ncsc.nl/, [Online: accessed on May 5, 2019].

[6] H. Shiravi, A. Shiravi, and A. Ghorbani, "A survey of visualization systems for network security", *IEEE Trans. Vis. Comput. Graph.* 18 (8), 1313–1329 (2012).

[7] A. Sethi and G. Wills, "Expert-interviews led analysis of eevi–a model for effective visualization in cyber-security", in *2017 IEEE Symposium on Visualization for Cyber Security (VizSec)*, 2017, pp. 1–8.

[8] D. M. Best, A. Endert, and D. Kidwell, "7 key challenges for visualization in cyber network defense", in *Proceedings of the Eleventh Workshop on Visualization for Cyber Security*, *ser. VizSec'14*, Paris, France, 2014, pp. 33–40.

[9] S. McKenna, D. Staheli, C. Fulcher, and M. Meyer, "Bubblenet: A cyber security dashboard for visualizing patterns", in *Eurographics Conference on Visualization (EuroVis)* vol. 35, 2016, pp. 281–290.

[10] M. Bostock, *Pseudo-Dorling cartogram*, https://bl.ocks.org/mbostock/4055892/, [Online: accessed on 5-04-2019], 2015.

[11] N. Cao, C. Lin, Q. Zhu, Y. Lin, X. Teng, *et al.*, "Voila: Visual anomaly detection and monitoring with streaming spatiotemporal data", *IEEE Trans. Vis. Comput. Graph.* 24 (1), 23–33 (2018).

[12] B. Song, J. Choi, S.-S. Choi, and J. Song, "Visualization of security event logs across multiple networks and its application to a CSOC", *Cluster Comput*, 1–12 (2017).

[13] B. Dumas, D. Lalanne, and S. Oviatt, "Human machine interaction", in, D. Lalanne and J. Kohlas, Eds., ser. Lecture Notes in Computer Science. Springer, 20095440, ch. Multimodal Interfaces: A Survey of Principles, Models and Frameworks, pp. 3–26.

[14] A. Jaimes and N. Sebe, "Multimodal human–computer interaction: A survey", *Comput. Vis. Image. Underst.* 108 (1), 116–134 (2007), Special Issue on Vision for Human-Computer Interaction.

[15] M. Turk, "Multimodal interaction: A review", *Pattern Recognit. Lett.* 36, 189–195 (2014).

[16] S. Oviatt, B. Schuller, P. Cohen, D. Sonntag, G. Potamianos, *et al.*, Eds., *The Handbook of Multimodal-Multisensor Interfaces, Volume 1: Foundations, User Modeling, and Common Modality Combinations*, ser. ACM Books Series. Association for Computing Machinery (ACM), 2017.

[17] R. A. Bolt, ""Put-That-There": Voice and gesture at the graphics interface", in *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '80, Seattle, Washington, USA, 1980, pp. 262–270.

[18] S. Oviatt, "Ten myths of multimodal interaction", *Commun. ACM* 42 (11), 74–81 (1999).

[19] N. Heneghan, G. Baker, K. Thomas, D. Falla, and A. Rushton, "What is the effect of prolonged sitting and physical activity on thoracic spine mobility? An observational study of young adults in a UK university setting", *BMJ Open* 1–6 (2018).

[20] J. Wahlström, "Ergonomics, musculoskeletal disorders and computer work", *Occup. Med. (Lond)* 55 (3), 168–176 (2005).

[21] P. Y. Loh, W. L. Yeoh, and S. Muraki, "Impacts of typing on different keyboard slopes on the deformation ratio of the median nerve", in *Proceedings of the 20th Congress of the International Ergonomics Association (IEA 2018)*, S. Bagnara, R. Tartaglia, S. Albolino, T. Alexander, and Y. Fujita, Eds., Cham: Springer, 2019, pp. 250–254.

[22] M. Tiric-Campara, F. Krupic, M. Biscevic, E. Spahic, K. Maglajlija, *et al.*, "Occupational overuse syndrome (technological diseases): Carpal tunnel syndrome, a mouse shoulder, cervical pain syndrome", *Acta Inform. Med.* 22 (5), 333–340 (2014).

[23] M. Janiak and C. Zieliński, "Control system architecture for the investigation of motion control algorithms on an example of the mobile platform Rex", *Bull. Pol. Ac.: Tech.* 63 (3), 667–678 (2015).

[24] C. Zieliński, T. Kornuta, and T. Winiarski, "A systematic method of designing control systems for service and field robots", in *19-th IEEE International Conference on Methods and Models in Automation and Robotics, MMAR*, IEEE, 2014, pp. 1–14.

[25] C. Zieliński, M. Stefańczyk, T. Kornuta, M. Figat, W. Dudek, *et al.*, "Variable structure robot control systems: The RAPP approach", *Rob. Auton. Syst.* 94, 226–244 (2017).

[26] C. Zieliński, T. Winiarski, and T. Kornuta, "Agent-based structures of robot systems", in *Trends in Advanced Intelligent Control, Optimization and Automation*, J. Kacprzyk and et al., Eds., ser. Advances in Intelligent Systems and Computing vol. 577, 2017, pp. 493–502.

[27] T. Kornuta and C. Zieliński, "Robot control system design exemplified by multi-camera visual servoing", *J. Intell. Rob. Syst.* 77 (3–4), 499–524 (2013).

[28] C. Zieliński, M. Figat, and R. Hexel, "Communication within multi-fsm based robotic systems", *J. Intell. Rob. Syst.* 93 (3), 787–805 (2019).

[29] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach. Third Edition*, Upper Saddle River, N.J.: Prentice Hall, 2010.

[30] M. Wooldridge, "Agent-based software engineering", in *Software Engineering. IEE Proceedings*, IET144, 1997, pp. 26–37.

[31] M. Wooldridge, "Intelligent agents," in *Multiagent Systems*, G. Weiss, Ed., Cambridge, MA, USA: MIT Press, 1999, pp. 27–77.

[32] L. Padgham and M. Winikoff, *Developing Intelligent Agent Systems: A Practical Guide*, John Wiley & Sons, 2004.

[33] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, New York, Oxford: Oxford University Press, 1999.

[34] W. Walker, P. Lamere, P. Kwok, B. Raj, R. Singh, *et al.*, *Sphinx-4: A flexible open source framework for speech recognition*, http://cmusphinx.sourceforge.net/sphinx4/, [Online: accessed on Oct 12, 2016].

[35] Kaldi, *The kaldi project*, http://kaldi.sourceforge.net/index.html, [Online: accessed on Oct 10, 2018].

[36] Loudia, *The loudia library*, https://github.com/rikrd/loudia, [Online: accessed on Oct 15, 2018].

[37] Alize, *The alize project*, http://alize.univ-avignon.fr, [Online: accessed on Oct 15, 2018].

[38] G. Gravier, *Spro (speech signal processing toolkit)*, https://gforge.inria.fr/projects/spro, [Online: accessed on Oct 18, 2018].

[39] M.-W. Mak and J.-T. Chien, *Machine learning for speaker recognition*, http://www.eie.polyu.edu.hk/~mwmak/papers/IS2016-tutorial.pdf, [Online: accessed on Oct 20, 2018].

[40] T. Marciniak, R. Weychan, A. Stankiewicz, and A. Dąbrowski, "Biometric speech signal processing in a system with digital signal processor", *Bull. Pol. Ac.: Tech.* 62 (3), 589– 594 (2014).

[41] G. Hinton et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups", *IEEE Signal Process. Mag.* 29 (6), 82–97 (2012).

[42] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information", *IEEE Trans. Pattern Anal. Mach. Intell.* 30 (2), 328–341 (2008).

[43] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features", in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 20011*, 2001, pp. I–I.

[44] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1867–1874.

[45] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks", in *Advances in neural information processing systems*, 2015, pp. 91–99.

[46] O. M. Parkhi, A. Vedaldi, A. Zisserman, *et al.*, "Deep face recognition", in *bmvc*1, 2015, p. 6.

[47] M. Grochowski, A. Kwasigroch, and A. Mikołajczyk, "Selected technical issues of deep neural networks for image classification purposes", *Bull. Pol. Ac.: Tech.* 67 (2), 363–376 (2019).

[48] J. Redmon, *Darknet: Open Source Neural Networks in C*, http://pjreddie.com/darknet/, 2013–2016.

[49] N. E. Gillian, R. B. Knapp, and M. S. O'Modhrain, "Recognition of multivariate temporal musical gestures using n-dimensional dynamic time warping", in *Proceedings of the International Conference on New Interfaces for Musical Expression NIME*, Norway, 2011.

[50] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning", *J. Big Data* 3 (1), 9 (2016).

[51] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition", *arXiv preprint arXiv:1806.05622*, 2018.

[52] A. Wojciechowski and K. Fornalczyk, "Single web camera robust interactive eye-gaze tracking method", *Bull. Pol. Ac.: Tech.* 63 (4), 879–886 (2015).

[53] J. Bobulski, "Multimodal face recognition method with twodimensional hidden markov model", *Bull. Pol. Ac.: Tech.* 65 (1), 121–128 (2017).