Information technology Applications

ITA

2/2022

PAN-EUROPEAN UNIVERSITY
Faculty of Informatics

VZDELÁVANIE VEDA VÝSKUM
EDUCATION SCIENCE RESEARCH

SSKI
SLOVENSKÁ SPOLOČNOSŤ PRE KYBERNETIKU A INFORMATIKU PRI SAV
SLOVAK SOCIETY FOR CYBERNETICS AND INFORMATICS

International Journal of Information Technology Applications (ITA)

Volume 11, Number 2, December 2022

AIMS AND SCOPE OF ITA

The primary aim of the International Journal of Information Technology Applications (ITA) is to publish high-quality papers of new development and trends, novel techniques, approaches and innovative methodologies of information technology applications in the broad areas. The International Journal of ITA is published twice a year. Each paper is refereed by two international reviewers. Accepted papers will be available online with no publication fee for authors. The International Journal of ITA is being prepared for the bibliographic scientific database Scopus.

International Journal of Information Technology Applications (ITA)

Instructions for authors

The International Journal of Information Technology Applications is welcoming contributions related with the journal´s scope. Scientific articles in the range approximately 10 standard pages are reviewed by two international reviewers. Reports up to 5 standard pages and information notices in range approximately 1 standard page are accepted after the decision of editorial board. Contributions should be submitted via e-mail to the editorial office. The language of contributions is English. Text design should preserve the layout of the template file, which may be downloaded from the webpage of journal. Contributions submitted to this journal are under the author´s copyright responsibility and they are supposed not being published in the past.

Deadlines of two standard issues per year

paper submission deadline	– end of May/end of October
review deadline	– continuous process
camera ready deadline	– end of June/end of November
release date	– Summer/Winter

Editorial office address

Faculty of Informatics, Pan-European University, Tematínska 10, 851 05 Bratislava, Slovakia
juraj.stefanovic@paneurouni.com

Published by

Pan-European University, Slovakia, http://www.paneurouni.com
Paneurópska vysoká škola, n.o., Tomášikova 20, 821 02 Bratislava, IČO 36 077 429
Civil Association EDUCATION-SCIENCE-RESEARCH, Slovakia, http://www.e-s-r.org
OZ VZDELÁVANIE -VEDA-VÝSKUM, Andrusovova 5, 851 01 Bratislava,
IČO 42 255 180
Slovak Society for Cybernetics and Informatics (SSKI)
at the Slovak Academy of Sciences,
Slovenská spoločnosť pre kybernetiku a informatiku pri SAV (SSKI),
Ústav automobilovej mechatroniky, Fakulta elektrotechniky a informatiky STU,
Ilkovičova 3, 812 19 Bratislava 1, IČO 00 178 730, http://www.sski.sk

Electronic online version of journal

http://www.paneurouni.com/ITA visit Archive and Instructions for authors:
http://www.e-s-r.org

Print

Multigrafika s.r.o., Rajecká 13, 821 07 Bratislava

Subscription

Contact the editorial office for details.
Older print issues are available until they are in stock.

ISSN: 2453-7497 (online)
ISSN: 1338-6468 (print version) Registration No.: EV 4528/12

Contents

Editorial

Dear authors, dear readers,

introducing to you with a delay the next issue of our journal, Contributions provide new insights into the research, development and application of new methods and algorithms in both theoretical but especially in the practical domain. This issue focuses on the presentation of advanced methods and techniques in the broad field of applied computer science, new communication technologies, automotive mechatronics, virtual reality and the application of artificial intelligence for early detection of neurodegenerative diseases based on real data.

Thank you for your contributions, we welcome your latest research results and solutions and we look forward to further collaborations.

prof. Ing. Štefan Kozák, PhD.
ITA Editor-in-Chief

THE PROGRAMMABLE VIRTUAL MODEL OF THE MECHATRONIC SYSTEM USING NX 12 ENVIRONMENT

Filip Žemla, Ján Cigánek

Abstract:

The paper deals with the design and implementation of the programmable virtual 3D model of the mechatronic system. The paper offers an analysis of a programmable logic controllers, as well as an overview of Industry 4.0, specifically an analysis of the concept of a digital twin and technologies for the creation of a digital twin. The design of the digital twin of the mechatronic system consists of two steps. The first step is the creation of the virtual 3D model, the second step is the creation of an interactive communication between the virtual model and programmable logic controller. The resulting digital twin of the high bay warehouse is compared with a real laboratory model. The task is to design and create a digital twin based on a real model that will simulate its behavior as accurately as possible.

Keywords:

PLC, digital twin, Industry 4.0, virtual 3D model, NX 12, mechatronic system.

ACM Computing Classification System:

Information systems, Information Systems applications, Process control systems.

Introduction

Nowadays, we can observe that the term digital twin is becoming more and more popular in all industries. We can find its use when solving problems in real objects or processes. In the standard commissioning of automated machines and equipment, there are usually many unexpected and unforeseen problems that extend the time of commissioning with increasing costs. This is because the PLC programmer can only fully verify the program after the technology is physically connected.

The digital twin solves problems such as eliminating errors in the control software, mechanical damage during the implementation of the device or late detection of the problem in the earlier stages of the project. It allows the programmer to verify the functionality of the prototype, which was created by another designer. The programmer can also test the control program and the functionality of the entire system already in the development phase and have the program tuned during commissioning itself. This will prevent mechanical damage and further downtime caused by traditional commissioning.

A digital twin is a real-time virtual representation of a physical object or process. Using a digital twin, we can observe the movement of products and other objects in real time on a virtual model. They can then be analyzed and the best possible solution evaluated. The virtual model of the device can be used by anyone, anywhere in the world. Through it, the user can monitor individual processes of automated lines and detect errors remotely without being physically present. A digital twin can be a digital replica of an object in the physical world, e.g. a production line, a car, or even larger objects like buildings, entire cities. There are many applications and use cases for the digital twin, it is a very active area of research and innovation.

Programmable Logic Controller (PLC) is considered as a fundamental component of automation in industry. PLC is an industrial digital computer that is relatively small and adapted to control production processes in real time, such as production lines, robotic devices or any activity requiring high reliability [1]. Looking at the function of a PLC, it can be defined as a device that receives multiple inputs or signals from a physical system. It can receive information from connected sensors or input devices, then process them and trigger outputs based on pre-programmed parameters. Such devices can be networked with other PLC and SCADA systems [2].

1 Industry 4.0 and Digital Twin

The industry is currently exposed to a significant technological change called Industry 4.0. This term was first mentioned by the German Federal Government in 2011. It arose as a result of the development of Internet and the idea of connecting the systems of the physical real world with the virtual world. The main task of the change is to achieve fully automated production, capable of autonomously adapting to external influences, such as changes in demand. This can be achieved by connecting all production equipment in one global network with autonomous exchange of information [3].

Industry 4.0 or the fourth industrial revolution represents a global trend in automation and industrial processes, with the use of modern intelligent technologies, towards the exchange of data in production processes and the complete automation of machines. Industry 4.0, sometimes referred to as IIoT (Industrial Internet of Things) or intelligent manufacturing, combines physical production and operations with intelligent digital technology, machine learning and big data. It creates a more integrated and better connected ecosystem for companies focused on manufacturing and supply chain management. The term Industry 4.0 also includes four key aspects: cyber-physical systems (CPS), Internet of Things (IoT), Internet of Services (IoS) and smart factories.

CPS represent a significant evolutionary step. They are physical devices with built-in tools for digital data collection in real time, their processing and distribution, they are interconnected using the Internet. The combination of CPS, high-performance software and special user interfaces, integrated into digital networks, creates a completely new world of system functionality.

IoT is a concept referring to the connection between physical objects such as vehicles, machines and other objects with embedded electronics, software, sensors connected to a network. IoT enables the collection and exchange of data. Connected objects can be controlled remotely, through existing network infrastructures, and create opportunities for further direct integration of the physical world into computer systems.

The Internet of Services (IoS) represents an infrastructure using the Internet as a medium for offering and selling services. As a result, services become tradable goods. IoS provides the business and technical basis for advanced business models focused on the provision and use of services.

A smart factory uses technologies, solutions and approaches within Industry 4.0. It is a concept derived from IIoT, assuming the production environment as a fully automated and intelligent network of systems, enabling the management of equipment, machines and logistics chains in the production plant, without human intervention. A smart factory is a place where data is exchanged not only between production tools and machines, but also between all elements in the production technology chain.

Virtual Commissioning (VC) is the simulation and modeling of the production system for the purpose of developing and testing the system before physical commissioning. Engineers simulate processes before turning on the physical system, allowing them to verify that everything is working. VC uses a virtual model, which is an accurate 3D simulation of mechanical, electrical and control systems, to verify the physical functions of the system before implementation.

VC technology and applications have been developed to significantly reduce or even eliminate the physical process, shorten the required time and ultimately deliver significant cost savings. VC does not have to wait for all the hardware to start up. VC can begin before the programmer has any hardware available. Because all the components exist in the software model, engineers can start testing in the digital space.

Today we use two virtual commissioning methods, Software in Loop (SiL) and Hardware in Loop (HiL). Both methods work with a virtual model of the device, as a copy of the real device on which the control program is tested. Both methods result in real-time simulation. The virtual model simulates all processes and thus generates input and output signals, which are subsequently converted into digital and analog form and transferred to the control program [4].

In the Hardware in Loop (HiL) method, the automated device is converted into a virtual form. A real PLC station with appropriate communication interfaces is subsequently used to control this model. This method allows the programmer to verify the control program in a complex and precise way, and with its results to be closer to reality. The disadvantages of this method are the need for real hardware, i.e. PLC with the associated I/O interface and PC with a running virtual model.

The principle of the Software in Loop method is based on simulations of both the virtual device and the controlling PLC. The developed control algorithms run in a simulation environment on the development computer, in simulation or in real time, depending on the requirements. The programmer thus can test the control program without the need for additional hardware. It allows testing the software before initializing the hardware prototypes, which significantly speeds up the development cycle.

Over the years, the definition of a digital twin has evolved, but the basic idea has remained the same: a dynamic virtual software-generated representation of the corresponding physical systems and processes. Since the introduction of this concept, the amount of data and information that can be transferred between products has increased. With a virtual model, it is possible to simulate the behavior of various states and thus evaluate its performance. These models can be simplified so that only the geometry or other characteristics match, thus reducing their size and speeding up the simulation calculation. Such simplified models allow to simulate complex systems and their behavior in real time, with acceptable computing power [5].

There are several software tools from different manufacturers for virtual commissioning or digital twin. The leading manufacturer of simulation software is the Siemens company, with its products NX Mechatronics concept designer and Tecnomatix Process Simulate, which contain all the necessary tools for the complex simulation requirements of the project. Another suitable environment for virtual commissioning is Emulate 3D from Rockwell Automation.

2 Case Study

For the practical part of the paper, a kit from Fischertechnik called Automated High-Bay Warehouse was chosen (Fig.1). This is a laboratory model in which a pallet with a workpiece is automatically transported by a conveyor and placed in a position in the warehouse using a manipulator. The model consists of three main parts: a conveyor with a color sensor, a manipulator that moves in 3 axes, and a warehouse where pallets with workpieces are stored. There are also limit switches, sensors and encoders that protect the manipulator from mechanical damage.

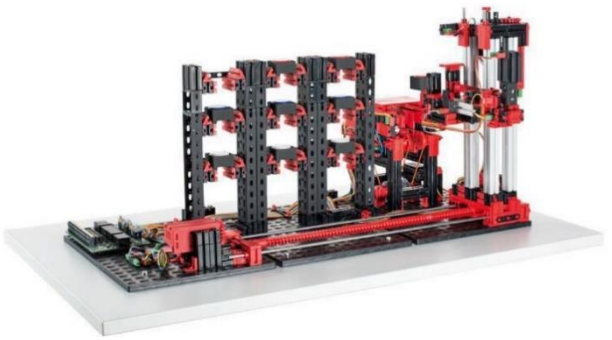


Fig.1. Laboratory model of High-bay warehouse.

VC of our high-bay warehouse was carried out using several environments: NX 12 and its MCD module with the running virtual model; TIA Portal V15.1 to configure individual PLC modules of Siemens S7-1500, and to create the control program for the high-bay warehouse and its visualization on HMI panel KTP700 Basic PN from Siemens; S7-PLCSim Advanced V3.0 environment to create a virtual instance of the PLC S7-1500 and to upload the created control program from the TIA Portal.

A. Creation of virtual 3D model

For creating a 3D model, we chose the NX 12 program from Siemens in the CAD module. The NX program is Computer-aided design (CAD), Computer-aided manufacturing (CAM) and Computer-aided engineering (CAE) software for detailed modeling of machines, from individual parts to large assemblies. In addition to modeling, it is possible to perform calculations, simulations, analyses, creation of drawing documentation, programming, etc. All these areas are interconnected, which increases the efficiency of the entire solution [6].

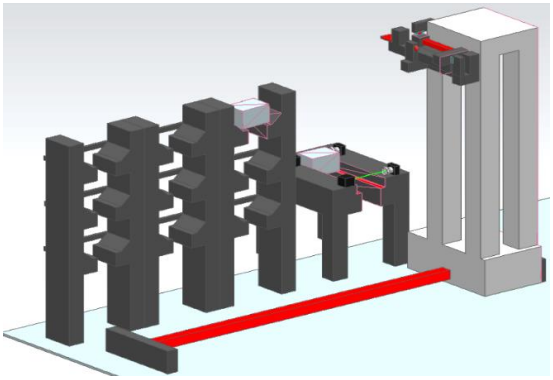


Fig.2. Virtual model of High-bay warehouse.

When creating the 3D model, we used real data from the Fischertechnik kit. With the help of a caliper, we obtained all characteristic and necessary dimensions by measuring, in order to maintain proportionality between the real model and the virtual model. We tried to simplify the individual details of the kit as much as possible, in order to speed up the subsequent simulations and simplify the modeling of the virtual model.

Our 3D model consists of several parts, which we will later combine into a single unit. The basic element is the manipulator, composed of three parts. The first part is the base that allows the manipulator to move up and down along Y axis. The other two parts of the manipulator are used for loading or unloading the pallet. One part is used for picking a pallet, thanks to which the manipulator can move forward and backward along Z-axis.

Another separate part related to the manipulator is the rail along the X axis. This is used to allow the manipulator to move left and right.

An equally important element is the warehouse, which we modeled as a separate component. In the warehouse, it was necessary to model 9 positions for pallet storage.

The last larger part is the conveyor, used to transport the pallet for the manipulator. The last step was the modeling of the limit switches and the sensors located on the conveyor, which consist of two separate components. In NX 12 environment, we combined individual functional elements and components into the resulting virtual model.

B. Creation of digital twin

The creation of the digital twin also took place in the NX 12 environment, in the Mechatronics Concept Designer (MCD) module. We uploaded the completed 3D CAD model of high-bay warehouse to the MCD application. We tried to revive the individual functional elements of 3D model in a virtual environment. MCD software is a module of the NX program that enables 3D modeling and interactive simulation of automation-related multi-body physics concepts. It allows creating a mechatronic 3D CAD model of an existing device, with the aim of mapping the physical and kinematic properties of the device. The simulation technology in MCD software is based on the game physics engine, based on simplified mathematical models, bringing the physical behavior of the real world into the virtual world.

The first step of the conversion was defining the movement coupling of individual functional elements. We assigned physical properties to the individual elements, simulating the real behavior of the body (option to choose manual assignment of the object's weight, location of the center of gravity, moment of inertia in individual axes or the choice of automatic assignment of values, based on the 3D model of the given part). Collision properties defining the mutual interaction of the bodies had to be assigned to the individual bodies.

The high-bay warehouse model contains four end sensors, two light sensors and drives ensuring movement in three axes, and the corresponding encoders. For all these elements it was also necessary to create a simulation in the virtual reality. For rectilinear movement of the conveyor, Transport Surface function was used, which will create a surface allowing objects to be moved in the selected direction and speed. Speed Control function was used to simulate the manipulator drive in all directions. Next, we created a simulation of the individual sensors. The Collision Sensor function was used for two light barriers on the conveyor to detect the intersection of the green beam with another object.

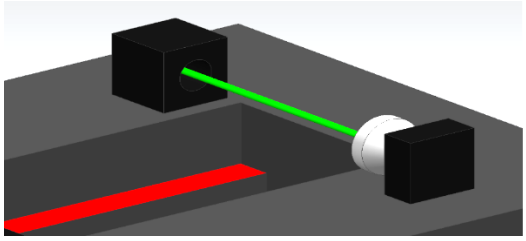


Fig.3. Light sensor (barrier) of virtual model.

It was also necessary to create collision sensors for the end sensors placed on the manipulator. These sensors detect the presence of the manipulator in the extreme position and prevent mechanical damage or collisions. Collision Sensor function ensures that the manipulator stops if it touches the limit switch.

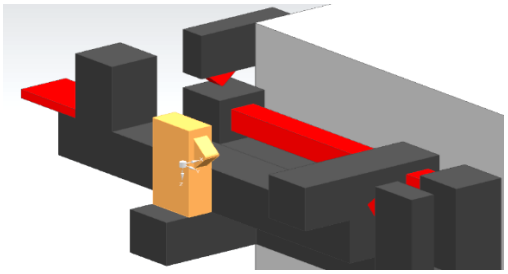


Fig.4. Limit switch of virtual model.

The laboratory model contains two encoders on the X and Y axes, which ensure the mapping of the current position of the manipulator. To simulate the encoders, we used Position Sensor function to measure the value of the displacement of the body on the given axes. The value of the encoders must be zero when the limit sensors are switched on, because the limit switches serve as reference values.

C. Connection to digital twin

In order to connect the virtual PLC and our digital twin, it was necessary to define the signals that ensure the connection between the virtual model and PLC. The creation of signals for PLC took place in NX 12 environment, in MCD module. Individual signals were created using Signal function. The output from PLC is the input in MCD, and conversely the input from PLC is the output in MCD. The names of the signals should be identical in MCD and in TIA Portal, as the Signal Mapping function will automatically allow to map the signals from MCD with the signals from PLC.

Bool or Double signals created in this way must then be connected to the corresponding functions controlling the functional elements of the laboratory model. We functionalized this connection using Runtime Expression tab, in which the behavior of the model can be defined during the simulation run. The signal connected to Speed Control function can control its activity, speed value, acceleration, simulating load conditions in the X, Y and Z axes.

To make the sensors functional during the simulation, Runtime Expression was used again. We chose the "triggered" option, because when the light beam is crossed or the limit switch is pressed, the signal value changes. Thus, we will be able to send the value from the sensor to the PLC.

D. Creation of PLC algorithm

The creation of the control algorithm took place in TIA Portal V15.1 environment. When creating a project, it is necessary to select correct PLC module and CPU type. Our kit consists of an S7-1500 series PLC with a CPU 1516-3 PN/DP from Siemens. Digital and analog inputs and outputs from a real lab model had to be added to CPU.

We also used memory variables to start specific sequences or part of the program, respectively to store a value. PLC program is stored in organizational blocks (OB). The organizational block forms the interface between the operating system and the program that is called cyclically by the system.

At first, we created "Safety" OB, which serves to safely stop the drives in extreme positions and prevents mechanical damage of individual parts. Here we used information from limit switches and pulse counting from encoders on individual axes. We created the algorithm for the reference of the manipulator in OB "Main", which ensures the introduction of the manipulator to the reference extreme position on the X and Y axis, and the initialization of individual auxiliary variables.

The automatic mode is used to automatically place the pallet in the warehouse. After placing the pallet on the conveyor and selecting a specific position in the warehouse using HMI panel, the manipulator will automatically store the pallet. It means the transfer of the pallet from the conveyor to the desired position in the warehouse. If the given position is already occupied, the manipulator unloads (moves) the pallet from the selected position to the conveyor.

The selection of the position is made through HMI screen (Fig.5), where the user can select one of 9 positions using a button. The user can thus decide which position he wants to load or unload. Loading or unloading from the position by the manipulator depends on the color of the button. If the color of the button is green, the user can store the pallet in the given position. If the pallet is red, it can be removed from the given position. We have created an OB called "HMI to PLC" for data transfer after pressing a button from HMI to PLC. Now we can identify whether the user wants to unload or store the pallet in the selected position.

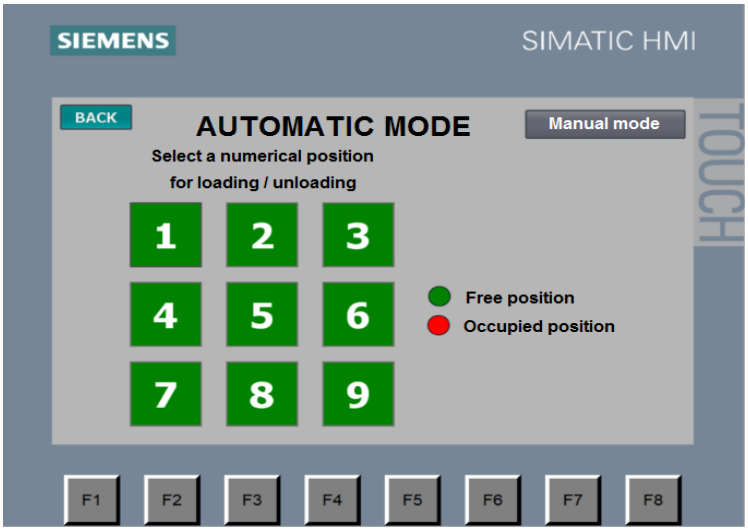


Fig.5. HMI for automatic mode.

After choosing the position, it is necessary to create an OB to map positions in the warehouse and to enter instructions for the manipulator: to load or to unload. For individual rows and columns, it was necessary to find out their individual coordinates. When moving the manipulator to a specific position, we use encoders to monitor the value on the X and Y axes.

In order to match the value in the virtual model and the value from the encoder, it was necessary to convert the measured pulses to milimetres. For the calculation, we used the length of the movement path in the X and Y axis, and pulses count of the real encoders. When the manipulator starts to move, we add or subtract value in the auxiliary variable according to the direction of movement, which we determine using the input signals.

In the case of the virtual model, we used Position Sensor function, which automatically measures the displacement value of the manipulator. At the start of the simulation, it was necessary to set the manipulator, to the starting positions, in order to activate the limit switches.

We send the value of the position from the virtual encoder directly to the PLC, using the connection of the signal and the auxiliary variable.

Next OB serves for unloading or loading a pallet to the selected position. The algorithm consists of four consecutive steps: a) arrival of the manipulator in front of the correct position, b) insertion of the manipulator into the warehouse, c) displacement of the manipulator along y axis, d) extension of the manipulator from the warehouse.

The manual mode is used for manual control and testing of the functionality of the manipulator, conveyor and individual functional elements. We can track the position of the manipulator on X and Y axes and we were able to map individual positions in the warehouse and to create an automatic mode. We can test the functionality of the safety stop of the drives, if the limit switch is turned on or if the position of the manipulator exceeds a critical limit. The movements of the manipulator in all directions are controlled using the buttons on HMI. In manual mode, the manipulator can be set to the reference position using the "Initialize" button.

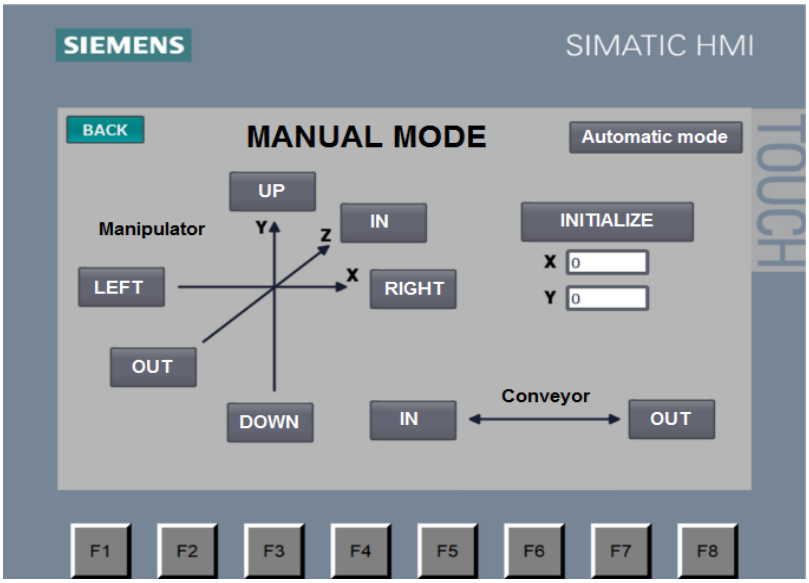


Fig.6. HMI for manual mode

E. Virtualization of PLC

PLC virtualization was executed in PLCSim Advanced V3.0, which allows creating a PLC instance replacing a real PLC. When creating a virtual PLC, in the user interface (Fig.7) we selected the PLCSIM option in Online Access option. We created an instance called my_PLC. In PLC type menu, Unspecified CPU 1500 option was selected, and then Start button was pressed. This step creates an instance with the specified name that represents the created virtual PLC.

Uploading a program created in TIA Portal environment to a PLC instance is the same as uploading to a real PLC, using Download to device option in TIA Portal. The dialog window offers the option of choosing the device to which the program should be recorded. In the line Type of the PG/PC interface we selected the bus type PN/IE, in the next line PG/PC interface we chose PLCSIM as the connection adapter. The last setting in Connection to interface/subnet line was Direct at slot '1 X1' option. After configuration, using Start search button, we found the created instance called my_PLC. After selecting the given device and pressing the load button, the program will be uploaded to this instance.

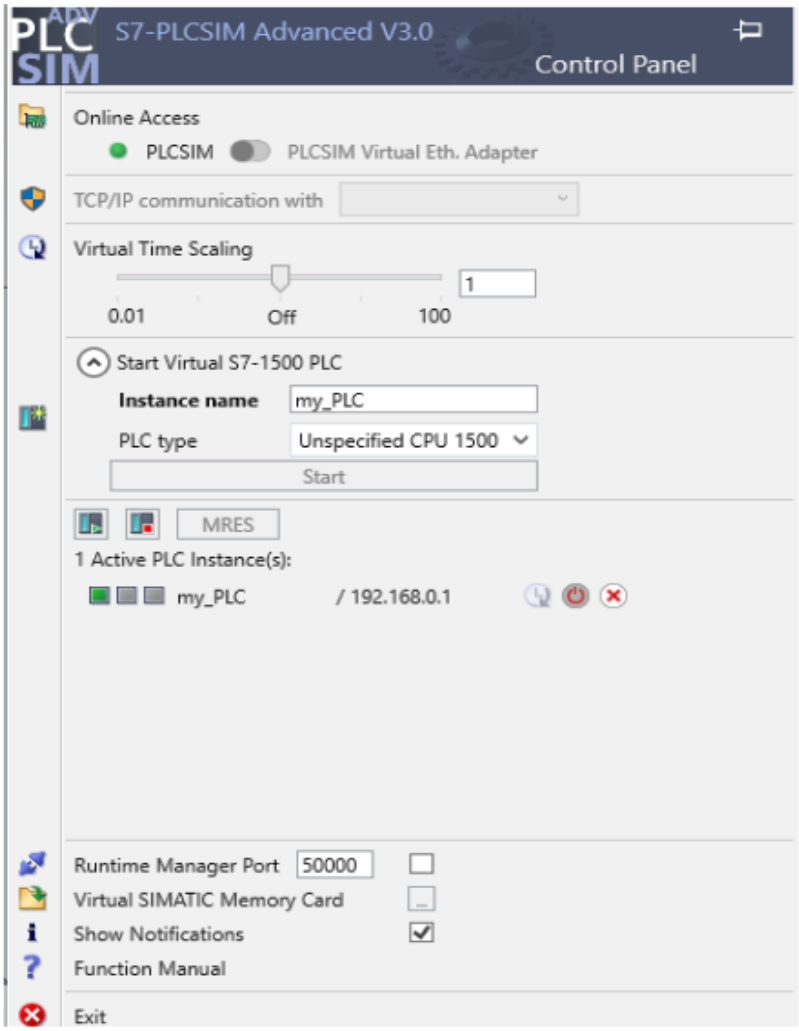


Fig.7. PLCSim Advanced.

F. Connection of virtual model and PLC

The connection of the signals of the virtual model and the signals in the PLC (running in PLCSim Advanced) was realized through the MCD module of NX 12 environment. In Automation tab, we selected External signal configuration function, which is used to upload PLC signals to MCD. In this dialog box, we selected PLCSIM Adv tab.

After selecting the PLC instance, all input and output signals of the virtual PLC are displayed in the lower part of the dialog window. When choosing IOMDB in Show line, memory type tags (auxiliary variables) will also be displayed.

Signal mapping function (Fig.8) was used to match the signals from PLC to the signal of the virtual model in MCD. In Signals section, the signals created in the virtual model are displayed on the left side, and the imported signals from the virtual PLC are on the right side.

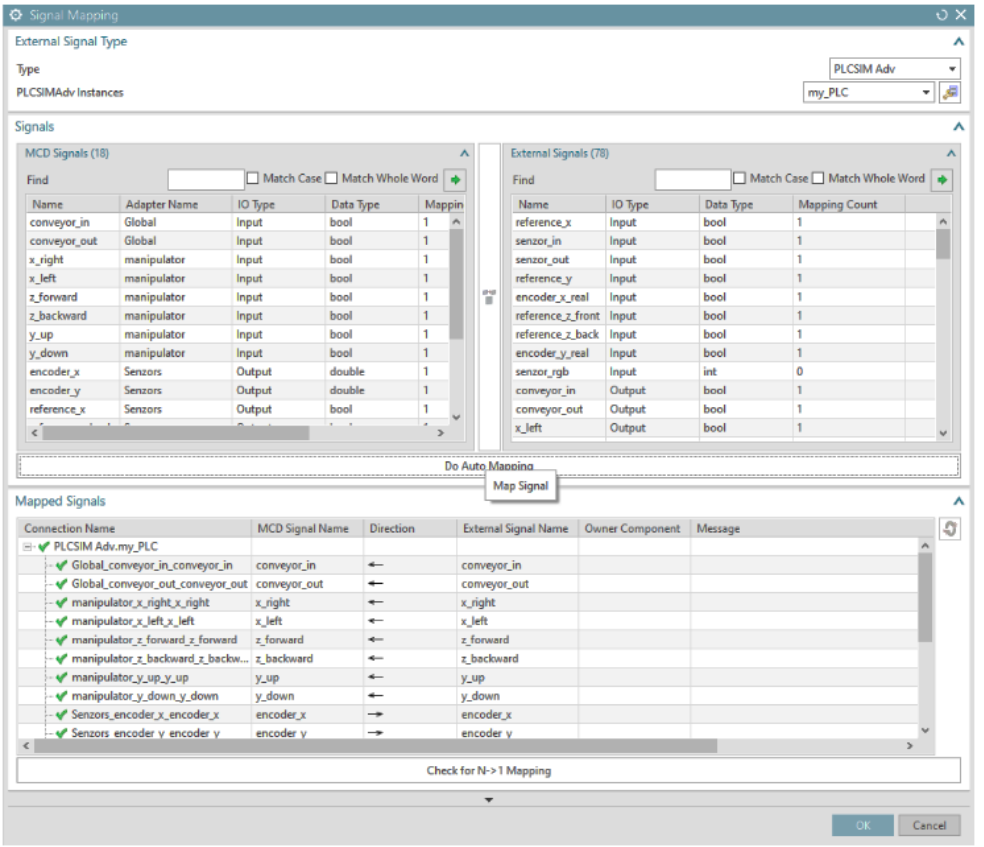


Fig.8. Signal connection between virtual model and virtual PLC.

To create a connection between individual signals, it is necessary to select one signal from the right side and its corresponding signal from the left side. We connect both signals using the central button. Several conditions must be met when connecting signals. The signals must be of the same data type and opposite I/O type. Therefore, the signals for controlling the drives were created as Input, and the sensor signals were defined as Output.

G. Simulation results

Starting the virtual model in NX MCD environment is possible only after starting the virtual PLC with the downloaded program. In NX, we launch the virtual model using the Play button. To control the high-bay warehouse, it is necessary to start the virtual HMI touch panel in TIA Portal. We verified the functionality of the virtual model using the manual mode, where we tested the control of individual manipulator drives. The course of the simulation was smooth and the behavior of the virtual model corresponded to the behavior of the real kit.

Finally, we verified using manual mode the functionality and control of individual drives in all axes of the manipulator, as well as conveyor drives. After verifying the functionality of all elements, a successful manipulator reference was made.

Using the automatic mode, we tested the loading of the pallet and its subsequent unloading from a random position in the warehouse (Fig.4).

After testing the functionality of the virtual model, we were able to compare the real and virtual model. In the case of comparing the functionality of PLC algorithm on a real and virtual model of a high-bay warehouse, we evaluate the behavior as identical. Individual positions in the warehouse were served by the manipulator as expected and without any problems in both models. All functional elements in the virtual model corresponded to the behavior in the real kit.

Conclusion

The realization of the digital twin was preceded by the creation of a 3D model. We created the virtual model in the CAD module of NX 12 environment. We tried to simplify the individual details of the virtual model as much as possible, in order to speed up subsequent simulations and simplify modeling. Subsequently, we uploaded the created 3D model to MCD module, where the conversion to a digital twin was executed. We tried to bring the individual functional elements of the virtual model to life in a virtual environment. It resulted in a programmable virtual model of the high-bay warehouse, where its inputs and outputs corresponded to the real kit.

PLC control algorithm consists of Automatic mode and Manual mode. With the help of the created virtual model and PLC program, a virtual commissioning was carried out, which verified the functionality of the model and the PLC algorithm. After the testing, we agreed the behavior of the virtual model and the real kit.

The virtual model of the high-bay warehouse could be improved by a more complex version of 3D model. At the same time, by adding more components to the virtual model, we can create more complex systems, for example by connecting with a robotic arm and other Fischertechnik kits to create an automated line in the virtual reality.

Acknowledgement

This work has been supported by the Slovak Grant Agency VEGA 1/0107/22, KEGA 039STU-4/2021 and Scientific Grant APVV-21-0125.

References

[1] A. Goel, “40 Important PLC Projects for Engineering Students”, Engineering, 2018. https://engineering.eckovation.com/plc-based-final-year-projects/

[2] T. Achyut, “PLC – Architecture/Hardware”. PLCAUTONETICS, 2020. https://www.plcautonetics.com/plc-architecture/

[3] J. Kruger, “Why are there still so many jobs?”, Siemens, 2020. https://new.siemens.com/global/en/company/stories/industry/why-are-there-still-so-many-jobs.html.

[4] J. Dzinic, Y. Charlie, „Simulation-based verification of PLC programs”, Department of Signals and Systems, Chalmers University of Technology. 2013.

[5] M. Grieves, “Digital Twin: Manufacturing Excellence through Virtual Factory Replication”. ResearchGate, 2015. https://www.researchgate.net/publication/275211047_Digital_Twin_Manufacturing_Excellence_through_Virtual_Factory_Replication.

[6] B. Lyndon, “Machine design software generates open source Mechatronics program code”, Automation, 2003.

Authors

Ing. Filip Žemla
Faculty of Electrical Engineering and Information Technology,
Slovak University of Technology in Bratislava, Slovakia
filip.zemla@icloud.com
Currently a student of doctoral studies at Slovak University of Technology in Bratislava. The main focus of his studies is oriented to virtualization and optimalisation of modern manufacture processes. His main skills are SCADA systems, database systems and front-end programming.

Ing. Ján Cigánek, PhD.
Faculty of Electrical Engineering and Information Technology,
Slovak University of Technology in Bratislava, Slovakia
jan.ciganek@stuba.sk
He was born in 1981 in Malacky, Slovakia. He received the diploma and PhD. degree in Automatic Control from the Faculty of Electrical Engineering and Information Technology, Slovak University of Technology (FEI STU) in Bratislava, in 2005 and 2010, respectively. He is now Assistant Professors at Institute of Automotive Mechatronics FEI STU in Bratislava. His research interests include optimization, robust control design, computational tools, SCADA systems, big data, and hybrid systems.

HIGH THROUGHPUT OPEN-SOURCE IMPLEMENTATION OF Wi-Fi 6 AND WiMAX LDPC ENCODER AND DECODER

Tomáš Páleník, Viktor Szitkey

Abstract:

This paper describes the design and C99 implementation of a free and open-source Low-Density Parity-Check (LDPC) codes encoder and decoder focused primarily on the Quasi-Cyclic LDPC (QC-LDPC) codes utilized in the IEEE 802.11ax-2021 (Wi-Fi 6) and IEEE 802.16-2017 (WiMAX) standards. The encoder is designed in two variants: the first one universal, the other a minimal memory usage design. The decoder provides a single- and multi- threaded implementation of the layered single-scan min-sum LDPC decoding algorithm both for floating point and fixed-point arithmetic. Both encoder and decoder are directly callable from MATLAB using the provided MEX wrappers but are designed to be simply used in any C project. A comparison of throughput and error performance with the recent commercial closed-source MEX implementation of an LDPC encoder and decoder introduced in MATLAB R2021b Communications Toolbox is provided. Source code portability to alternative non-x86 architectures is facilitated by using only the standard C99 constructs, GNU tools, and POSIX libraries. The implementation maintains low-memory requirements, enabling its deployment in a constrained-architecture in the context of Internet of Things. All source codes are freely available on GitHub under a permissive BSD license.

Keywords:

Layered decoding, LDPC direct encoding, MEX, min-sum decoding, QC-LDCP codes.

ACM Computing Classification System:

Mathematics of computing, Mathematical software, Mathematical software performance.

Introduction

After the introduction of the QC-LDPC codes in [1] and the desired direct-encoding [2] calculated using the sparse parity check matrix **H** without needing the code generator matrix G, QC-LDPC codes have been an integral part of modern communication standards for some time now. This ranges from IEEE 802.16 [3], through IEEE 802.11ax [4,5], DVB-S2 [6] up to the latest 5G 3GPP Release 17 TS 38.212 [7], and even the wired networking Ethernet IEEE 802.3 standard [8]. Because of their faster decoding, the ultra-high speed 200 Gbps and 400 Gbps IEEE 802.3-2018 prescribe the use of Reed Solomon (RS) codes [9] to fit the strict latency constraints. Even though the RS codes don't reach the exceptional near-capacity-limit error performance of LDPC, the research in this complementary area is still ongoing [10]. Arguably the most sophisticated LDPC designs are presented in the CCSDS specifications, such as [11] and [12], with intense research activity ongoing: Recent papers [13], [14], [15], propose novel hardware encoders, for the CCSDS standard QC-LDPC codes, achieving acceleration primarily by restructuring encoder structure and utilizing parallelism. [16] provides a comparison of several hard- and soft-decision decoding algorithms for the CCSDS defined (128512, 64256) QC-LDPC code,

while [17] deals with potential energy saving in a satellite link. [18] describes how the Automorphism Ensemble Decoding (AED) can be enabled for QC-LDPC codes to improve error performance for codes with shorter *N* (128 to 256) used in Wi-Fi, 5G, and CCSDS specifications and [19] compares the Belief propagation, Min-Sum and Neural Normalized Min-Sum (N-NMS) decoding performance in the context of line product codes (LPC) also defined by the CCSDS. [20] deals with modifying existing LDPC codes to incorporate desired run length properties and [21] presents a mathematical framework for constructing QC-LDPC codes with desired girth, providing examples based on the CCSDS-defined protographs.

In any case, the LDPC codes have evolved to become ubiquitous in almost all broadband wireless technologies. Implementations of encoders and decoders are also plentiful [22] - [26], with [27] giving an exhaustive overview of even more. These are all, however, hardware implementations: either ASIC designs intended to be integrated into modems, or FPGA-based. Complementing these are pure software implementations available at GitHub: [28] provides universal encoder/decoder C++ implementations, but doesn't include the practical, standard-specified **H** matrix designs. [29] provides the IEEE802.11 codes and separate MATLAB and C encoder/decoder implementations. [30] focuses on CCSDS LDPC codes and provides a MATLAB implementation. In [31] a C++ implementation defines a SIMD-acceleration using the AVX-512 vectorized arithmetic focused on the DVB-S2 and DVB-T2 standard. [32] provides a MATLAB implementation for the 5G New Radio TS38.212 encoder and decoder. [33] is an older C implementation focused on education with no standard **H** matrix designs, and [34] describes the results for a SIMD-accelerated x86-specific C++ encoder/decoder implementation using the Intel compiler and the optimized Math Kernel Library with focus on the CCSDS and DVB-S2 LDPC codes.

The common denominator of all these implementations is the lack of integration of fast C/C++ code (if available) with MATLAB. This integration is important, since it is impractical to write whole simulations in C or maintain two parallel (and potentially functionally different) versions for MATLAB and C/C++. The obvious solution would be the utilization of fast C/C++ code in MATLAB by implementing MEX-file wrappers that allow calling such a method directly from MATLAB. Such implementation exists as a commercial closed-source product and is part of the Communications Toolbox. In fact, the LDCP codes in MATLAB have a history on their own: the first introduction of LDPC in MATLAB came in 2007 in the form of the `dvbs2ldpc()` function, still available today. In 2012 the `comm.LDPCDecoder` and `comm.gpu.LDPCDecoder` system objects were introduced in R2012a [35], with the former now deprecated and replaced by the universal `ldpcDecode()` function, complemented by `ldpcEncode()`, both being new improved and optimized implementations introduced only recently in R2021b [36]. These come with the ability to expand practical QC-LDPC prototype matrices to sparse **H** matrices, but the task of obtaining the prototype matrices is left to the user. The implementations are available as binary MEX modules only, so it is hard to infer what advanced optimizations were made. Furthermore, for a researcher, who would like to test potential modifications to the encoding/decoding algorithms, while evaluating practical LDPC codes used in modern standards, such closed-source implementations are of little use.

This is where our implementation comes to play: it provides a relatively fast C99 implementation of a QC-LDPC encoder and decoder that can be integrated into any C project, directly containing practical LDPC codes utilized in the IEEE802.11ax-2021 [4] (actually defined in the earlier IEEE802.11-2020 specification [5]) and IEEE 802.16-2017 standards, while also providing MEX wrappers and supporting functions for seamless MATLAB integration for statistical evaluation of waterfall curves. Everything is available as completely free and open source code under the permissive BSD license. To facilitate portability, and even a potential ARM platform compatibility, no external libraries, such as the optimized Intel MKL, or ISA-specific constructs, such as intrinsics, were used.

The rest of the paper is organized as follows: the next section contains a brief recap of the QC-LDPC codes design along with some important specific parameters of the practical codes used in IEEE802.11ax and IEEE802.16-2017. The second section recaps the direct encoding algorithm and the fourth describes the two encoder implementations: one universal and another designed with minimal memory requirements in mind. These can be switched at compile-time. The next section then recaps the design and describes our implementation of the layered single-scan min-sum decoder algorithm, while the sixth section provides some thoughts on a fixed point arithmetic decoder design. The sixth section provides an error performance evaluation in the form of waterfall curves, along with the comparison of throughput of various decoder setups in contrast to the existing MATLAB Communications Toolbox implementation. The last section concludes the paper.

1 LDPC Codes in Modern Standards

Discovered by Gallager [37], the LDPC codes are linear block codes (LBC) with basic parameters [*N*, *K*] defined by their sparse parity check matrix **H**. The code generator matrix $\mathbf{G}^{K \times N}$ can be derived from $\mathbf{H}^{M \times N}$ (defining: $M = N - K$ and code rate $R = K / N$) , but it will unlikely be sparse and therefore unsuitable for practical high-throughput encoding. The encoding using the **G** matrix is, however, described very simply by multiplication of the data word – the vector **i** of length *K* bits by **G** to obtain the codeword **c** of length *N*. For a systematic code, the systematic part **s** of the codeword is a subvector of **c** equal to **i**. We may denote:

$$\mathbf{c}^{1\times N} = \left[\mathbf{s}^{1\times K} \quad \mathbf{p}^{1\times M}\right] = \mathbf{i}^{1\times K} \cdot \mathbf{G}^{K\times N}, \mathbf{s} = \mathbf{i} \tag{1}$$

The sparsity of the large **H** matrix not only provides for the excellent error correction properties of LDPC codes, it also allows for an efficient storage of **H** in memory: instead of using an array of size $M \times N$, only the indices of ones need to be stored. Throughout the rest of the paper we will use *n* for the column index (also known as the variable node index in soft-decoding context) and *m* for the row index of **H** (also known as the check-node index). Then for each $n \in \{ 1, 2, \ldots, N \}$ the set of row indices of ones in a given column of **H** may be called neighborhood of *n* and denoted *M(n)* while for each $m \in \{ 1, 2, \ldots, M \}$ the *N(m)* represents the set of column indices of ones in a given row *m*. Any sparse binary matrix **H** can be stored in a memory efficient way as an array of arrays *M(n)*, or array of arrays *N(m)*. The number of elements of $g(m) = |N(m)|$ may be called the degree of row *m*, while the symbol $f(n) = |M(n)|$ denotes the degree of column *n*. For practical irregular LDPC codes, the *g(m)* and *f(n)* are not constant over *m* and *n*, and their average and maximum values are of interest.

$$\mathbf{H} = \begin{bmatrix} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \cdots & \mathbf{P}_{0,N_b-1} \\ \mathbf{P}_{1,0} & \mathbf{P}_{1,1} & & \mathbf{P}_{1,N_b-1} \\ \vdots & & & \\ \mathbf{P}_{M_b-1,0} & \mathbf{P}_{M_b-1,1} & \cdots & \mathbf{P}_{M_b-1,N_b-1} \end{bmatrix} \tag{2}$$

The subclass of practical QC-LDPC codes used in communications standards [3-5] was deliberately constructed with such structure of the **H** matrix that allows for the encoding process to be completely and efficiently implemented directly using **H**. As shown in (2), the very large binary sparse matrix **H** is designed as a block matrix consisting of a grid of submatrices $\mathbf{P}_{i,j}$ of size $Z \times Z$.

Each of the submatrices may be a zero matrix or a permuted identity matrix, with the only allowed permutation being a cyclic shift, resulting with $\mathbf{P}_{i,j}$ circulant and sometimes called a cyclic shift matrix.

The constraints on $\mathbf{P}_{i,j}$ allow for a compressed representation of $\mathbf{H}$ in a form of a model matrix $\mathbf{H_{bm}}$ of size $M_b \times N_b$ (denoting $M_b = N_b - K_b$) where each element $\mathbf{H_{bm}}(i, j)$ represents the circulant permutation matrix $\mathbf{P}_{i,j}$ and $N = N_b \times Z$, $K = K_b \times Z$. By convention the value of $\mathbf{H_{bm}}(i, j)$ equal to -1 represent a zero submatrix $\mathbf{P}_{i,j}$, value 0 an identity submatrix, and any positive number represents identity matrix circularly shifted by $\mathbf{H_{bm}}(i, j)$. The compressed $\mathbf{H_{bm}}$ matrices are then directly specified by the standards with different model matrices defined for each codeword length N, as in [5], or common structure for the largest value of N given along with a corresponding transformation for all other supported values of N, as in [3]. In all cases the large sparse binary matrix $\mathbf{H}$ is obtained from $\mathbf{H_{bm}}$ by expanding each element of $\mathbf{H_{bm}}$ to a square matrix of size Z. For both standards of interest, the $\mathbf{H}$ matrices share further similarity: The $\mathbf{H_{bm}}$ can be further partitioned to three submatrices as follows (using the notation from [3]):

$$\mathbf{H_{bm}}^{M_b \times N_b} = \left[\mathbf{H_{b1}}^{M_b \times K_b} \quad \mathbf{H_{b2}}^{M_b \times M_b} \right] = \left[\mathbf{H_{b1}}^{M_b \times K_b} \quad \mathbf{h_b}^{M_b \times 1} \quad \mathbf{H_{b2'}}^{M_b \times (K_b - 1)} \right] \tag{3}$$

In terms of submatrix indexing the submatrices $\mathbf{H_{b1}}$, $\mathbf{h_b}$ and $\mathbf{H_{b2'}}$ may be defined as:

$$\mathbf{H_{b1}} = \mathbf{H_{bm}}[0,1,...,M_b - 1; 0,1,...,K_b - 1] \tag{4}$$

$$\mathbf{h_b} = \mathbf{H_{bm}}[0,1,...,M_b - 1; K_b] \tag{5}$$

$$\mathbf{H_{b2'}} = \mathbf{H_{bm}}[0,1,...,M_b - 1; K_b + 1,...,N_b - 1] \tag{6}$$

where $\mathbf{h_b}$ is the first column of matrix $\mathbf{H_{b2}}$. We use the a slightly mathematically imprecise indexing starting at zero instead of one. We do this in order to be compatible with the indexing defined in the standard [3].

The direct encoding algorithm further requires that the column vector $\mathbf{h_b}$ has a special structure containing two paired (equal) values in the first and last position and a single positive value among the remaining positions. All other elements are set to –1 indicating later expansion to zero matrices. This structure is required for the direct encoding described in later sections. Also required is the structure of the submatrix $\mathbf{H_{b2'}}$, where two zero diagonals within an all –1 matrix in $\mathbf{H_{b2'}}$ translate to double diagonal expanded submatrix of $\mathbf{H}$. Equation (7) shows the example structure of $\mathbf{h_b}$ and $\mathbf{H_{b2'}}$ for $N_b = 24$, $K_b = 18$ for the $R = 3/4$ QC-LDPC code defined in [3]:

$$\mathbf{H_{b2}}^{6\times 6} = \left[\mathbf{h_b}^{6\times 1} \quad \mathbf{H_{b2'}}^{6\times 5} \right] = \left[\begin{bmatrix} \mathbf{0} \\ -1 \\ \mathbf{80} \\ -1 \\ -1 \\ \mathbf{0} \end{bmatrix} \quad \begin{bmatrix} 0 & -1 & -1 & -1 & -1 \\ 0 & 0 & -1 & -1 & -1 \\ -1 & 0 & 0 & -1 & -1 \\ -1 & -1 & 0 & 0 & -1 \\ -1 & -1 & -1 & 0 & 0 \\ -1 & -1 & -1 & -1 & 0 \end{bmatrix} \right] \tag{7}$$

Both standards specify a set of supported code rates and codeword sizes that further influence the design of encoder and decoder implementation. Parameters for the IEEE 802.11ax standard are summarized in (Tab.1), while the IEEE802.16-2017 is summarized in (Tab.2).

Table 1. Data word size *K*, codeword size *N*, and expansion factor *Z* for the IEEE 802.11ax. Word sizes divisible by 8 bits are shown in bold.

***N*(bit)**	***Z*(bit)**	***K* (bit)**			
		***R*=1/2**	***R*=2/3**	***R*=3/4**	***R*=4/5**
648	27	324	**432**	486	540
1296	54	**648**	**864**	972	**1080**
1944	81	972	**1296**	1458	1620

It is important to note that different sections of the Wi-Fi 6 standard define several different LDPC codes and that different versions of each standard may share some LDPC codes. In this section we will deal with the codes defined in the IEEE 802.11-2020 [5], since these provide the longest codeword sizes and best error correcting capabilities. These are then also reused in the more recent standard version IEEE 802.11ax-2021 [4].

As show in (Tab.1), all codeword lengths *N* are multiples of 8 bits. This is not true for all data word sizes *K*. The expansion factor (or subblock size) *Z* is deliberately chosen to take on values 27, 54 or 81, clearly with the implementation of a hardware encoder in the form of specifically designed ASIC circuit in mind. As discussed in later sections, this design choice will complicate a memory optimized C implementation. In terms of alignment of bits to blocks easily fitting standard C data types, a slightly better situation is with the IEEE802.16-2017 standard, shown in (Tab.2), where all data words and all codewords fit the 8bit-in-a-byte storage. Also depicted are all the possible values of the subblock size *Z*. Starting at 24 and incrementing by 4 up to 96, 10 out of the total 19 values are divisible by 8, facilitating an optimized binary encoder implementation. Values 32, 64, and 96 enable further optimization by using wider native C types.

Table 2. Data word size *K*, codeword size *N*, and subblock size *Z* for the IEEE 802.16-2017 standard. All data and codewords lengths in bits are multiples of 8. Only half of the *Z*s are divisible by 8, while all data word sizes fit a 8bit-in-a-byte storage. *Z* factors divisible by 8 are show in bold.

***N*(bit)**	***Z*(bit)**	***K* (Byte)**			
		***R*=1/2**	***R*=2/3**	***R*=3/4**	***R*=4/5**
576	**24**	36	48	54	60
672	28	42	56	63	70
768	**32**	48	64	72	80
864	36	54	72	81	90
960	**40**	60	80	90	100
1056	44	66	88	99	110
1152	**48**	72	96	108	120
1248	52	78	104	117	130
1344	**56**	84	112	126	140
1440	60	90	120	135	150
1536	**64**	96	128	144	160
1632	68	102	136	153	170
1728	**72**	108	144	162	180
1824	76	114	152	171	190
1920	**80**	120	160	180	200
2016	84	126	168	189	210
2112	**88**	132	176	198	220
2208	92	138	184	207	230
2304	**96**	144	192	216	240

2 QC-LDPC Direct Encoding

The special structure of the **H** matrix was designed to allow for the direct encoding algorithm where the encoding of a systematic code uses the **H** matrix instead of the code generator matrix **G**. For a practical C encoder implementation the model matrix $\mathbf{H_{bm}}$ is actually used, so the large sparse binary **H** matrix, or even permutation submatrices $\mathbf{P}_{i,j}$ are never actually stored anywhere in memory. Since the multiplication of some subvectors of **i** of length Z by matrix $\mathbf{P}_{i,j}$ is identical in result to cyclic shift, the actual operation to implement as a building block of the encoder is the cyclic shift of subvectors of size Z bits by $\mathbf{H_{bm}}(i, j)$ positions. The direct encoder algorithm originally introduced in [38] and described in sufficient detail also in [3] can be summarized as follows:

The information block **i** (also known as systematic vector s) is divided into K_b blocks of Z bits. The grouped **i** can be denoted as a matrix **u**, where each element u_i is a column vector:

$$\mathbf{u}^{Z\times K_b} = [u_0 \quad u_1 \quad \dots \quad u_{K_b-1}] \tag{8}$$

$$u_i = [s_{iZ} \quad s_{iZ+1} \quad \dots \quad s_{(i+1)Z-1}]^T \tag{9}$$

The parity sequence **p** is calculated in blocks of Z bits directly using the matrix $\mathbf{H_{bm}}$. Again, the parity sequence **p** can be expressed as a matrix in terms of blocks, where each element v_i is a column vector Z bits long:

$$\mathbf{p}^{Z\times M_b} = [v_0 \quad v_1 \quad \dots \quad v_{M_b-1}] \tag{10}$$

$$v_i = [p_{iZ} \quad p_{iZ+1} \quad \dots \quad p_{(i+1)Z-1}]^T \tag{11}$$

The slightly nonstandard notation for vectors u_i and v_i is used here in order to preserve notation in [3]. The encoding is divided into initialization and recursion. First the initialization is defined as follows:

$$v_0 = \mathbf{P}^{-1}_{\mathbf{H_{bm}}(y,K_b)} \cdot \left(\sum_{i=0}^{M_b-1} \sum_{j=0}^{K_b-1} \mathbf{P}_{\mathbf{H_{bm}}(i,j)} u_j \right) \tag{12}$$

Where $\mathbf{H_{bm}}(i, j)$ is an element of the (potentially scaled) model matrix $\mathbf{H_{bm}}$, $\mathbf{P_{Hbm}}_{(i, j)}$ represents the circulant permutation matrix, and multiplication by $\mathbf{P_{Hbm}}_{(i, j)}$ implements the cyclic shift operation on subvector u_j. The element $\mathbf{H_{bm}}(y, K_b)$ is the only nonnegative element of the subvector $\mathbf{h_b}$ outside of the paired values, as shown in the example in (7) with $\mathbf{H_{bm}}(y, K_b) = \mathbf{h_b}(2) = 80$. Note that the value of y is different for each $\mathbf{H_{bm}}$ and must be found on-the-fly by the encoder. The inverse permutation $\mathbf{P}^{-1}$ is just a cyclic shift in the opposite direction. All addition/sum operations are performed over GF(2).

After initialization and successful calculation of v_0, the encoder proceeds with iteration/recursion, finding the remaining parity subblocks v_i:

$$v_1 = \mathbf{P}_{\mathbf{H_{bm}}(0,K_b)} v_0 + \sum_{j=0}^{K_b-1} \mathbf{P}_{\mathbf{H_{bm}}(0,j)} u_j \;,\; i = 0 \tag{13}$$

$$v_{i+1} = v_i + \mathbf{P}_{\mathbf{H_{bm}}(i,K_b)} v_0 + \sum_{j=0}^{K_b-1} \mathbf{P}_{\mathbf{H_{bm}}(i,j)} u_j \;,\; i > 0 \tag{14}$$

Since the sums terms in (12):

$$\mathbf{S}_i \equiv \sum_{j=0}^{K_b-1} \mathbf{P}_{\mathbf{H_{bm}}(i,j)} u_j \ , i = 0 \ldots M_\mathrm{b} - 1 \tag{15}$$

are calculated during the initialization and reused during the iteration, it is advantageous to store their values for reuse. Since the elements of $\mathbf{H_{bm}}$ equal to -1 expand to all zero **P** submatrix, a multiplication by such a $\mathbf{P_{Hbm}}$ results to a zero term in sums and may be safely skipped in the implementation.

3 C99 Encoder Implementation

The mathematical definition of the encoder given in the previous section determines the design of the final C algorithm, while the actual values of code parameters *N*, *K*, *M*, and *Z* play an important role. This paper describes two encoder implementations. The first one is a universal array implementation, that supports all combinations of the code parameters, but requires more memory, since each bit is actually stored as a C array element (usually but not necessarily an 8bit-wide `unsigned char`). The advantage is a simple code, where bit accesses are directly supported by an array access operator. The disadvantage for actual transmission systems/modems processing blocks of truly binary data is the necessity to expand the data inside the encoder by a factor of 8, so that each bit is actually stored in a separate byte.

The second implementation is a bitmap (or packed-bits) encoder – a true binary encoder in that it handles binary data blocks as bitmaps, and accesses bits within bytes by using the bitwise operators. While it is possible to implement cyclic shifts even for a subblock of size *Z* defined in the Wi-Fi standard {27, 54, 81}, these require non-byte aligned memory accesses that hurt performance. This approach was tried and benchmarked, and the measured low throughput was then a reason for not implementing the bitmap encoder for these values of *Z*. On the other hand, the WiMAX standard provides several code parameters combinations, where all *N*, *K*, *M*, *Z* are divisible by 8 (or by 16 and even 32 and 64) which allows for a clean and fast C implementation with minimal memory requirements.

All encoder equations (12) - (15) were implemented directly, replacing the multiplication $\mathbf{H_{bm}}(i, j)$. u_j with a cyclic shift of vector u_j by $\mathbf{H_{bm}}(i, j)$ bit positions and each addition with a bitwise XOR. The C language doesn't directly support a construct that would access the underlying ISA rotation instructions, so two linear shifts of a variable are called and combined with a bitwise OR operator. This is one of the known drawbacks of the C language [39] and the task of compiling the three related C operations into one rotation instruction is left to the compiler.

Since the encoding algorithm is defined in terms of blocks of bits of size *Z*, that can span several bytes or multibyte words, array indexing and bitwise operators must be combined to implement the circular shift that is the basis of equations (12) - (15). The principle of the memory efficient bitmap encoder cyclic shift routine is demonstrated in (Fig.1), where three indices are defined: the bit index I_bBL within the block of size *Z*, the bit index within the word I_bW and the index I_W of the word within the array. Since the array is stored in memory addressed in bytes, the byte index I_B denotes the byte offset from the start of the array. It may be a convenient, but not necessary to set the word size to contain just one byte, making $I_\mathrm{W} = I_\mathrm{B}$. This choice is made in (Fig.1) for the sake of example simplicity. Let *wb* denote the word size in bits.

Also depicted in (Fig.1), is the observation that the shift *s* can be conceptually divided into two parts: the shift *sw* expressed in an integral number of words and the remaining shift of up to *wb* – 1 bits. Shown for selected values of *s* = 3, 11, 19, all congruent modulo *wb* = 8, the resulting

array elements, although placed on different array positions, are bitwise identical ([ ] denotes C array indexing): Out_s=11[1] = Out_s=3[0] = Out_s=19[2].

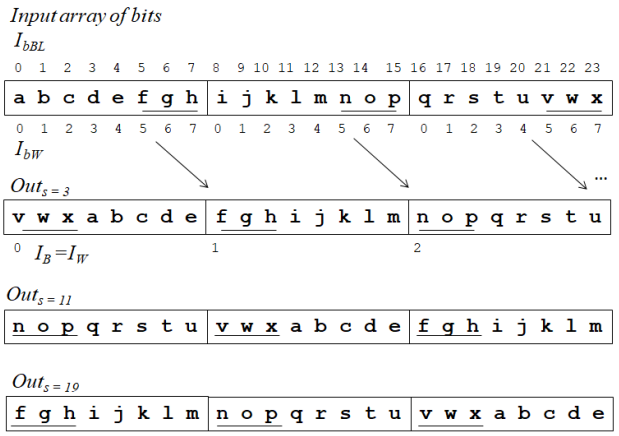


Fig.1. The array cyclic shift examples for word size wb = 8 bits, block size Z = 24 and selected shift value s = 3; 11; 19. Indices are: index of the bit within the block I_{bBL}, of the bit within the word I_{bW}, of the word I_W in the array, and the byte offset from the start of the array I_B (now same as I_W)).

The arbitrary cyclic shift by s positions can be represented as $s = sw \cdot wb + sb$ where sw = div(s, wb) is the partial rotation of the array in whole words (array elements) and sb = mod(s, wb) the residual rotation in bits (using standard integer division and modulo operations). Clearly $sb < wb$. This division serves the purpose of efficient implementation by a combination of two operations: First the whole array is shifted by sb bits in a way where two successive array elements are combined together into a resulting array element using bitwise operations and where the last array element is also used as the element preceding the first one. Then the rest of the shift of the array by sw words is implemented by reshuffling whole array using modulo indexing. These two steps can be combined together into a single scan of the input array. Algorithm A1 provides the resulting algorithm implementing an arbitrary bit shift on an array where zw is the number of words in array, and $Z = zw \cdot wb$.

Algorithm A1:

```
sW = shift / wb ;                   //shift in words
sb = shift % wb ;                   //remaining bits
prevSW = A[ ZW - 1 ] ;              //previous word
for( i = 0 ; i < ZW ; i++ ){
    j = ( i + sW ) % ZW ;           //DST word index
    curSW = A[ i ] ;                //source word
    hi = prevSW << wb - sb ;
    lo = curSW >> sb ;
    B[ j ] = hi | lo ;              //bitwise OR
    prevSW = A[ i ] ;
}
```

This algorithm is effective and works for any shift size s = 1to Z – 1. With the important limitation of the block size Z equal to a multiple of the word size wb. This algorithm will therefore be directly used for encoding half of the supported codeword sizes defined by the WIMAX standard, while being completely unsuitable for Wi-Fi 6.

4 Universal LDPC Decoder

In our LDPC decoder implementation, the algorithm choice was motivated by our efforts to produce an ideally exact, or at least practically close approximation of the decoding results obtained when using the widely popular but proprietary MATLAB decoder provided by the `ldpcDecode()` function of the MATLAB R2021b Communications Toolbox . This determined the algorithm choice to be the time proven min-sum algorithm defined in [40,41], which provides high throughput and is a good approximation to the much more computationally demanding posterior Log-Likelihood Ratio (LLR)-based Belief Propagation algorithm [42]. To save memory, the single-scan version of the min-sum algorithm, as described in [43] was implemented, more specifically the layered version with faster convergence as described in [44]. Two versions of the decoder were implemented: one using floating point arithmetic – the 32-bit `float` C type, and another, slightly faster and more memory efficient, fixed point arithmetic decoder, where the LLR approximations were stored as 16-bit `short int` values.

The paper by Huang [43] provides a concise overview of the min-sum algorithm, together with detailed description of its memory optimized single-scan improvement. Both algorithm variants are briefly summed up here: For a Gaussian channel and binary (BPSK) modulation, let x_n be the n-th transmitted bit, n_n the sample of Additive White Gaussian Noise (AWGN), and y_n the sample in the receiver. The modulation and channel transfer can be then described by (16) and the soft-decoding min-sum algorithm works with the LLR metric Z_n, defined by (17):

$$y_n = (-2x_n + 1) + n_n \tag{16}$$

$$Z_n^{(0)} = \ln \frac{p(x_n = 0 / y_n)}{p(x_n = 1 / y_n)} = 2y_n / \sigma^2 \tag{17}$$

where the σ^2 is the channel variance (may be omitted in the min-sum implementation) and the superscript (k) indicates the iteration number. Each of the iteration implements the following steps:

1. The horizontal scan – the check node update:
   For each m and each $N(m)$ calculate the check node message L_{mn}, based on the variable node messages Z_{mn} coming from all incident variable nodes except the output one (with index n).
   $$L_{mn}^{(k)} = \min_{n' \in N(m) \setminus n} \left| Z_{mn'}^{(k)} \right| \cdot \prod_{n' \in N(m) \setminus n} \operatorname{sgn}\left(Z_{mn'}^{(k)} \right) \tag{18}$$

2. The vertical scan - the variable node update:
   For each n and each $M(n)$ calculate the variable node message Z_{mn} coming from all incident check nodes, except the output one (with index m).
   $$Z_{mn}^{(k)} = Z_n^{(0)} + \sum_{m' \in M(n) \setminus m} L_{m'n}^{(k)} \tag{19}$$

3. For each n and each $M(n)$ calculate the posterior LLR estimate $Z_n^{(k)}$:
   $$Z_n^{(k)} = Z_n^{(0)} + \sum_{m \in M(n)} L_{mn}^{(k)} \tag{20}$$

4. Hard decision and termination:

For each codeword position n calculate the bit estimate (21) and if the orthogonality with the **H** matrix condition is satisfied: $\mathbf{H}\hat{\mathbf{x}} = \mathbf{0}$, terminate the decoding.

$$\hat{x}_n^{(k)} = \begin{cases} 0, & \text{if } Z_n^{(k)} > 0; \\ 1, & \text{otherwise} \end{cases} \tag{21}$$

The values of Z_{mn} are initialized by received channel samples: $Z_{mn}^{(0)} := Z_n^{(0)}$ and (19) can be easily rewritten using (20):

$$Z_{mn}^{(k)} = Z_n^{(k)} - L_{mn}^{(k)} \tag{22}$$

The single-scan min-sum algorithm modification takes advantage of the following observation: when eq. (18) is applied to a vector of values, the resulting vector only contains elements with two possible magnitudes: the minimum of the input elements, and the second minimum. If signs are stored separately from magnitudes, compressed as a bitmap in a single word, this property, exploited by the algorithm design in [43], can be used to drastically reduce the decoder memory requirements. For example, the maximum check node degree for the Wi-Fi 6 code R = 5/6 is max($g(m)$) = 20, so this optimization provides a saving of 18 magnitudes for each check node.Furthermore, as described in [43], only the check node messages L_{mn} need to be stored, immediately lowering memory requirements by almost 50%. Equation (18) can combined with (22) and rewritten as:

$$L_{mn}^{(k)} = \min_{n' \in N(m) \setminus n} \left| Z_n^{(k-1)} - L_{mn'}^{(k-1)} \right| \cdot \prod_{n' \in N(m) \setminus n} \operatorname{sgn}(Z_n^{(k-1)} - L_{mn'}^{(k-1)}) \tag{23}$$

To reduce the number of iterations necessary for obtaining the desired BER, a further modification of the computation – the layered decoding described in [44] was implemented. Without an in-depth description we can summarize it in the following way: The horizontal scan in (23) is not performed on each of the M check nodes in a single loop, but is organized in tiers - groups of check nodes of size Z block-rows of matrix **H**. The calculations of (20), (22) and (23) are then interleaved during each decoder iteration (k): (23) is evaluated for a block of Z rows/checks, after which the updated values Z_n are used in evaluation of (20) and (22) which in turn updates the inputs for the check nodes calculation (23) of the next tier. The algorithm then consists of two embedded iterations: the inner one alternating between the horizontal and vertical step, and the outer iteration over the M_b groups of check nodes/rows. These are sometimes denoted as super-iterations.

5 Fixed Point Implementation

For potential use of the LDPC decoder in constrained systems, especially with memory saving in mind, we also implemented a fixed-point arithmetic decoder, by default using the `int16_t` exact-width integer data type, introduced in C99 for the min-sum algorithm metrics. This usually but not necessarily compiles to `short int`. Because the decoding algorithm only uses the minimum, plus and minus operations, the fixed-point arithmetic can further be simplified by simply mapping of floating point values to an integer range. Our choice of the 16-bit representation is much wider than the various 8- and even fewer bits used in decoders described in literature, but the implementation is fully parametrized, so it is in theory easy to switch to a narrower 8bit representation. It is important to note that, when using narrow integer types, MATLAB uses a saturating arithmetic to minimize errors coming from signal values higher than the maximum level, while the standard C arithmetic uses the faster native wrap-around (or overflow) arithmetic.

Since it would be costly to check for overflow every time an addition is made, the primary way used in our implementation is to use a wider storage than necessary. In the case of the 16bit short `int16_t` C type, the floating point LLR values are actually sampled as Q_B bit integers, with the default value of $Q_B = 10$.With an extra sign bit, this effectively defines an 11bit signed range from -1023 to 1023. These values are stored within 16bit integers, leaving the 5 most significant bits free so when values accumulate during the min-sum algorithm operation, the overflow occurrences will be reasonably rare and not catastrophically effect the resulting BER. Logically we can expect this negative effect to be stronger with a growing number of summation elements, therefore codes with smaller rate should be affected more. This is confirmed in the results section.

In order to support memory constrained architectures, three different configurations with different memory requirements can be defined: Config A – a completely universal and memory unconstrained implementation, will be useful for MATLAB simulations. It supports all of the LDPC code parameters defined in (Tab.1) and (Tab.2) with an array encoder and floating point decoder. Config B present a memory-constrained implementation: a bitmap encoder and fixed-point decoder. It supports only a subset of compatible code (N, K) parameters, more specifically the half of the WiMAX code parameters, shown in bold in (Tab.2). Config C will be a minimal memory implementation with only one of the LDPC code parameters: $N = 576$ and $R = 1/2$ supported. (Tab.3) gives an overview of the memory requirements for all three implementations:

Table 3. Rounded memory requirements for various configurations in Bytes.

Configuration	ENC [B]	DEC [B]	Total [KiB rounded]
Config A	4896	162 432	164
Config B	864	84 096	83
Config C	240	10 080	11

An important aspect of our implementation is the seamless integration with MATLAB. While writing whole simulations in C is possible, and may provide very high throughput, the flexibility of MATLAB as a tool for rapid prototyping and testing of potential modifications to existing algorithms makes it an indispensable platform. Out of all the implementations available online [28] - [32] none provides both C-language implementation with its performance, and also MATLAB integration. The user can, however, sometimes choose between one of the two. Our implementation fills this niche in that it implements a C encoder and decoder that can be directly used in an actual communication system, while also allowing to run them unchanged from within the MATLAB environment. To achieve this, separate C source files implement the necessary MEX wrappers that make the encoder and decoder functions callable from MATLAB. All code is written in the C99 dialect supporting the portable representation of exact width integer types and tested with the GNU `c99` command, along with `g++`. If necessary, the encapsulation of functions to C++ objects should be a simple task.

For both encoder and decoder the necessary buffers are defined as statically allocated arrays and the important code parameters are written as constants (preprocessor macros) in an automatically generated header file `ldpc.h` and source file `ldpc.c`. This enables for a simple and readable code, where the analogy with the underlying mathematical equations is clearly visible. Furthermore, this design facilitates the shift of C code optimizations to the compiler, where they belong. The compilation to MEX file is done from inside the running MATLAB environment, and appropriate functions are provided to make this process completely transparent to the user, i. e. the user only has to call the provided functions, and doesn't need to care about the details of compilation.

Since many parameters are compiled in, there are some limitations to current usage: the adaptive change of code parameters during the simulation of an Adaptive Coding and Modulation (ACM) systems is not yet implemented, but is still possible. All that needs to be done is to call the compilation function at the beginning of simulation several times, one for each parameter set, and specify slightly different names for the resulting MEX modules. The m-file wrapper would then need to be extended to call a different MEX module for different code parameters. No C code modification would be needed for this extension

6 BER and Throughput Analysis

6.1 BER Evaluation

We evaluated the error correcting capabilities of our implementation for the QC-LDPC codes used in IEEE 802.11ax and IEEE 802.16-2017 communication standards using the classic approach of Monte-Carlo simulation producing the waterfall curve for the AWGN channel. The whole communication chain: LDPC encoder, channel and LDPC decoder was evaluated alongside the commercial Communication toolbox LDPC implementation introduced in recent MATLAB version R2021b (denoted further as COM). As shown in (Fig.2) with curves for selected code parameters, the error performance of our floating point implementation is visually indistinguishable from the toolbox implementation.

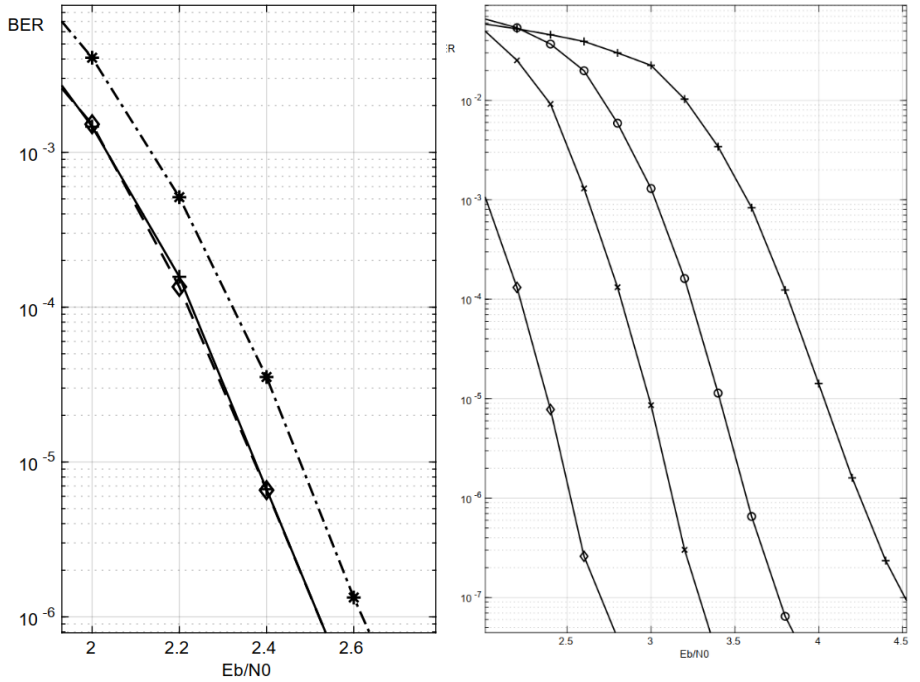


Fig.2. **a) (left)** Error performance for the Wi-Fi 6 R = 1/2 LDPC N = 1944 for maximum 10 decoder iterations along with the MATLAB Communications toolbox R2021b LDPC implementation (plus marker). Our floating point (diamond) result closely copies the COM implementation, fixed point decoder (star) shows some expected penalty. **b) (right)** Results for the IEEE 802.11-2020 defined LDPC codes for N = 1944 and maximum of 10 decoder iterations, Code rates R = 1/2 (diamond), R = 2/3 (x), R = 3/4 (o) and R = 5/6 (+).

As expected, the memory-constrained fixed point decoder (Config B) introduces an approximately 0,1 dB penalty. This effect is most visible for larger values of N and smaller code rates where the eq. (22) sums over large number of parity-check matrix rows; 972 rows for the code with results shown in (Fig.2). For smaller N = 576 and higher R = 5/6, the fixed-point decoder curve is getting even closer to the floating point implementation. During all the simulations at least 10000 errors were collected for each data point on the waterfall curve.

(Fig.2b) provides a comparison of results for various code rates defined for the N = 1944 LDPC code used in Wi-Fi 6, and (Tab.4) gives another perspective on the similarities of our implementation (MEX) and the existing closed-source implementation (COM). Along with the number of simulated bit transfers a direct comparison of the resulting absolute number of errors after decoding is given. More important relative error between the COM and MEX implementation is calculated (denoted Δ).

Table 4. The average number of iterations (nIter), number of decoded bits (in thousands), absolute number of errors of the commercial decoder (#E-COM) and our open source implementation (#E-MEX).

Eb/N0	nIter	Kbit	#E-COM	#E-MEX	Δ%
1.00	8.00	1152	131 877	131 873	3.03e-3
1.20	7.99	3456	313 767	313 768	3.19e-4
1.40	7.93	5760	346 651	346 651	0.0
1.60	7.73	8064	241 298	241 300	8.29e-4
1.80	7.17	10368	97 798	97 794	4.10e-3
2.00	6.36	12672	23 731	23 732	4.20e-3

As shown in the last column of (Tab.4), in terms of number of errors after decoding, the implementations give practically the same results. This is also true for the average number of decoder iterations, denoted nIter, so only the value for the MEX decoder is shown.

6.2 Throughput Evaluation

Since the LDPC decoder is by far the most computationally intensive task in our simulations, throughput evaluation was focused on the decoder function. Several different single- and multi-threaded decoder configurations were evaluated and throughput compared to the optimized MATLAB R2021b `ldpcDecode()` implementation (denoted as COM). Two x86-64 platforms were compared: MATLAB R2021b on Ubuntu 18.04LTS running on an 8 core/16 thread Intel Core i7-9800X CPU at 3.80GHz with Skylake-X architecture supporting the AVX-512 instruction set extension, and MATLABR2022a on Ubuntu 20.04LTS running on an 16 core/32 thread AMD Ryzen 9 5950X Processor.

(Tab.5) summarizes the data-bits throughput of the decoder for the WIMAX QC-LDPC code with rate R = 5/6, and codeword size N = 2304. Data were processed in blocks of 10 codewords per thread and the decoder was always set to perform a fixed number of 10 iterations in order to prevent comparing runtimes with varying iteration numbers. 32-bit floating point and 16-bit fixed point (integer) implementations were evaluated along with two different multithreading approaches: The first, more straightforward, implementation spawns worker threads each time the MEX function is run by MATLAB and destroys them during the same call, just after the block is decoded. This is denoted in (Tab.5) with the suffix: simple. A more sophisticated method, denoted as MTX, starts the worker threads once and then synchronizes to them each time the decoder MEX function is called by using POSIX thread conditional variables and mutexes.

Table 5. Throughput comparison for two CPUs: AMD and Intel, single- and multi-threaded implementation, floating- and fixed- point decoder, Commercial LDPC implementation (COM), and our implementation (MEX). MTX denotes a more sophisticated thread synchronization implementation using POSIX mutexes.

	LDPC Decoder implementation / CPU	**Throughput [Mbps]**
1	MEX float, single thread, Intel	1.61
2	MEX fixed, single thread Intel	2.15
3	COM single thread, Intel	2.62
4	MEX float, 16 thread simple, Intel	11.19
5	MEX float, 16 thread MTX, Intel	12.40
6	MEX fixed, 16 thread simple, Intel	13.19
7	MEX fixed, 16 thread MTX, Intel	13.44
8	COM Multi thread, Intel	2.10
9	MEX float, single thread, AMD	2.45
10	MEX fixed, single thread AMD	3.70
11	COM single thread, AMD	3.64
12	MEX float, 32 thread simple, AMD	36.46
13	MEX float, 32 thread MTX, AMD	37.76
14	MEX fixed, 32 thread simple, AMD	40.35
15	MEX fixed, 32 thread MTX, AMD	43.95
16	COM Multithreaded, AMD	3.33
17	CLI float, single thread AMD	6.31

As shown on lines 1 to 3, and 9 to 11 our single-threaded implementation reaches only about 60% of the otherwise equivalent Communications Toolbox function. This is not surprising, given the fact that the LDPC decoder in MATLAB is now a mature optimized implementation, which is only available as a closed-source MEX file. It is hard to infer what optimizations, such as the use of intrinsics or the optimized Intel MKL library, were made. Our implementation relies on the standard C99 language constructs only, which brings some throughput penalty but facilitates portability. The fixed point implementation improves the throughput slightly while sacrificing some error performance.

Since modern CPUs have been using many cores for a long time, the multi-threaded implementation is actually the one that matters. Here the advantage of our approach lies in the ability to fine-tune the number of threads that the user can specify explicitly, compared to the On/Off setting in the toolbox function. Lines 4 to 7 and 12 to 15 compare various multithreaded implementations on Intel and AMD platforms. The comparison with the toolbox function is given on lines 8 and 16, showing the throughput of the multithreaded version of the `ldpcDecode()` method to be an order of magnitude lower than of our implementation. The low throughput of the multithreaded toolbox decoder is somewhat surprising, and may indicate a bug in the implementation, potentially fixed in some later toolbox release. The actual use of multiple threads was checked by the OS-built-in system monitor utility.

What's a bit disappointing is the almost negligible throughput improvement of the more sophisticated MTX design, where the worker threads are running (or waiting) in parallel to the main MATLAB thread, and are synchronized to the main thread by means of POSIX conditional variables and mutexes. As shown in rows 5, 7, 13, 15 of (Tab.5), these actually represent so much overhead that such implementation, with its greatly complicated design, bring negligible benefit relative to the simple implementation shown in rows 4, 6, 12, and 14.

For more insight into how much the MATLAB environment affects performance, a command line version of the benchmark was run, compiled by the GNU `c99` CLI compiler. The result shown on line 17 of (Tab.5) indicate more than double the performance.

Conclusion

In this paper we described the details of our QC-LDPC encoder and universal LDPC decoder C99 implementations, which focus on (but are not limited to) the modern LDPC codes defined in the IEEE 802.11-2020 and IEEE 802.16-2017 standards. We provided error performance evaluation results comparable to a state-of-the art closed-source implementation provided by the MATLAB R2021b Communications Toolbox, along with comparison of throughput of various configurations on Intel and AMD platforms. The complete source code of the C99 encoder and decoder, along with MATLAB MEX wrappers and supporting MATLAB scripts are freely available at our GitHub page [46] published under a permissive BSD license.

Acknowledgement

This work was supported by the Slovak APVV Agency under cont. no. APVV-19-0436.

References

[1] TANNER, R. M., SRIDHARA, D., SRIDHARAN, A., et al. LDPC block and convolutional codes based on circulant matrices. IEEE Transactions on Information Theory, 2004, vol. 50, no. 12, pp. 2966-2984. DOI: 10.1109/TIT.2004.838370

[2] LI, Z., CHEN, L., ZENG, L., et al. Efficient encoding of quasi-cyclic low-density parity-check codes. IEEE Transactions on Communications, 2006, vol. 54, no. 1, pp. 71-81. DOI:10.1109/TCOMM.2005.861667

[3] IEEE Std. 802.16-2017, IEEE standard for air interface for broadband wireless access systems – Section 8.4.9.2.5: low density parity check (LDPC) code. IEEE New York (USA), 2018, pp. 1459 – 1463. ISBN 978-1-5044-4474-3

[4] IEEE Std. 802.11ax-2021, Part 11: wireless lan medium access control (MAC) and physical layer (PHY) specifications - Amendment 1: enhancements for high-efficiency WLAN. IEEE New York (USA), 2021. ISBN 978-1-5044-7389-7

[5] IEEE Std 802.11-2020, Part 11: wireless lan MAC and PHY specifications - Annex F: HT LDPC matrix definitions. IEEE New York (USA), 2021. p. 4130–4132. ISBN 978-1-5044-7283-8

[6] ETSI EN 302 307 V1.2.1, Digital Video Broadcasting (DVB): Second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications (DVB-S2). ETSI Sophia Antipolis Cedex (France), 2009.

[7] 3GPP TS 38.212 V17.4.0, 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; NR; Multiplexing and channel coding (Release 17). 3GPP Valbonne (France), 2022. Available at: http://www.3gpp.org

[8] IEEE Std. 802.3-2018, IEEE Standard for Ethernet 802.3-Section 7: 10G Ethernet, Chapter: 101.3.2.4 low density parity check (LDPC) forward error correction (FEC) codes. IEEE New York (USA), 2018, p. 321–330. DOI: 10.1109/IEEESTD.2018.8457469

[9] IEEE Std. 802.3-2018, IEEE Standard for Ethernet 802.3-2018 (Revision of IEEE Std 802.3-2015) - Section 8: 200G and 400G Ethernet Chapter: 119.2.4.6. IEEE New York (USA), 2018, p.67–69. DOI: 10.1109/IEEESTD.2018.8457469

[10] FARKAS, P., RAKUS, M. Decoding five times extended reed solomon codes using syndromes. Computing and Informatics, 2020, no. 6, vol 39, pp.1311–1335. ISSN 2585-8807 (online), DOI: 10.31577/cai_2020_6_1311

[11] CCSDS 131.0-B-3, Recommendation for Space Data System Standards - TM synchronization and channel coding - Recommended standard - Blue book. CCSDS Washington DC (USA), 2017, p. 51–55. Available at: https://public.ccsds.org/publications/bluebooks.aspx

[12] CCSDS 130.1-G-3: Report Concerning Space Data System Standards – TM synchronization and channel coding summary of concept and rationale – Informational report – Green book. CCSDS, Washington DC (USA), 2020, p.84–94.

[13] SOWMYA, G., KEERTHI, K., LALITKRUSHNA, J. T., et al. An architecture for efficient encoding of quasi cyclic LDPC codes and its implementation in FPGA. In IEEE 11th International Conference on Communication Systems and Network Technologies (CSNT), Indore (India), 2022, pp. 136-140. DOI: 10.1109/CSNT54456.2022.9787597

[14] LIU, J., FENG, Q. A Miniaturized LDPC Encoder: Two-layer architecture for CCSDS near-earth standard. IEEE Transactions on Circuits and Systems II: Express Briefs, 2021, vol. 68, no. 7, p. 2384–2388. DOI: 10.1109/TCSII.2021.3054334

[15] KANG, J., WANG, B., ZHANG, Y., AN, J. Enhanced partially-parallel LDPC decoder for near earth applications. In IEEE 11th International Conference on Electronics Information and Emergency Communication (ICEIEC), Beijing (China), 2021, p. 12-16. DOI: 10.1109/ICEIEC51955.2021.9463828

[16] RAJAGOPALAN, P., et al. Performance analysis of LDPC decoding algorithms for CCSDS tel-ecommand space data link protocol. In 2nd International Conference for Emerging Technology (INCET), Belagavi (India), 2021, p. 1-5. DOI: 10.1109/INCET51464.2021.9456163

[17] RAKUS, M., FARKAS, P. On possible energy savings with transmission supported via feedback channel in CubeSat transceiver. In Journal of Electrical Engineering, 2021, vol.72, no. 5, p. 337–342. DOI: 10.2478/jee-2021-0047

[18] GEISELHART, M., EBADA, M., ELKELESH, A. et al., Automorphism ensemble decoding of qua-si-cyclic LDPC codes by breaking graph symmetries. IEEE Communications Letters, 2022, vol. 26, no. 8, p. 1705-1709. DOI: 10.1109/LCOMM.2022.3174164

[19] NGUYEN, J., WANG, L., HULSE, C. et al., Neural normalized min-sum message-passing vs. viterbi decoding for the CCSDS line product code. In ICC 2022 - IEEE International Conference on Com-munications. Seoul (Korea), 2022, p. 2375–2380, DOI: 10.1109/ICC45855.2022.9838412

[20] FARKAS, P., RAKUS, M. Adding RLL properties to four CCSDS LDPC codes withouth increasing their redundancy. Accepted for publication in Computing and Informatics, 2023, ISSN 1335-9150

[21] SMARANDACHE, R., MITCHELL, D. G. M. A unifying framework to construct QC-LDPC Tanner graphs of desired girth. IEEE Transactions on Information Theory, 2022, vol. 68, no. 9, p. 5802-5822. DOI: 10.1109/TIT.2022.3170331

[22] JUNG, Y., CHUNG, C., KIM J., JUNG, Y. 7.7Gbps encoder design for IEEE 802.11n/ac QC-LDPC codes. In International SoC Design Conference (ISOCC), Jeju (Island), 2012, pp. 215–218. DOI: 10.1109/ISOCC.2012.6407078

[23] MAHDI A., PALIOURAS V. A low complexity-high throughput QC-LDPC encoder. IEEE Transac-tions on Signal Processing, 2014, vol. 62, no. 10, p. 2696–2708. DOI: 10.1109/TSP.2014.2314435

[24] KUN, C., QI, SHENGKAI, L., CHENGZHI, P. Implementation of encoder and decoder for LDPC codes based on FPGA. Journal of Systems Engineering and Electronics, 2019, vol. 30, no. 4, p. 642–650. DOI: 10.21629/JSEE.2019.04.02

[25] NGUYEN, T., NGUYEN, T., LEE, H. Efficient QC-LDPC encoder for 5G new radio. Electronics. 2019, vol.8, no. 6, p.668, ISSN: 2079-9292. DOI: 10.3390/electronics8060668

[26] YAO, X., LI, L., LIU, J., LI Q. A low complexity parallel QC-LDPC encoder. In IEEE MTT-S In-ternational Wireless Symposium (IWS). Nanjing (China), 2021, p. 1–3, DOI: 10.1109/IWS52775.2021.9499562

[27] LE GAL, B., JEGO, C. High-throughput multi-core LDPC decoders based on x86 processor. IEEE Transactions on Parallel and Distributed Systems, 2016, vol. 27, no. 5, p. 1373–1386. DOI: 10.1109/TPDS.2015.2435787

[28] WANG, A. Wangaran/LDPC Codes GitHub repository, 2015. [Online] Cited 2023-02-24. Available at: https://github.com/ wanganran/LDPC_codes/tree/master/LDPC_Code

[29] TAVILDAR, S. Tavildar/LDPC. GitHub repository, 2016, [Online] Cited 2023-02-24. Available at: https://github.com/ tavildar/LDPC

[30] SEA-WIND, cea-wind/LDPCC. LDPCC GitHub repository, 2018, [Online] Cited 2023-02-24. Available at: https://github.com/cea-wind/LDPCC/tree/master

[31] INAN., A. xdsopl/LDPC. GitHub repository, 2018. [Online] Cited 2023-02-24. Available at: https://github.com/xdsopl/LDPC

[32] MAUNDER, R. robmaunder/ldpc-3gpp-matlab. GitHub repository, 2018, [Online] Cited 2023-02-24. Available at: https://github.com/robmaunder/ldpc-3gpp-matlab

[33] RADFORD, N. radfordneal/LDPC-codes. GitHub repository, 2011, [Online] Cited 2023-02-24. Available at: https://github.com/radfordneal/LDPC-codes

[34] BERTRAND, L. G. blegal/Fast – Fast LDPC decoder for x86, GitHub repository, 2020. [Online] Cited 2023-02-24. Available at: https://github.com/blegal/Fast_LDPC_decoder_for_x86

[35] MATHWORKS MATLAB documentation: comm.gpu. LDPCDecoder, 2012. [Online] Cited 2023-02-24. Available at: https://www.mathworks.com/help/comm/ref/comm.gpu.ldpcdecoder-system-object.html

[36] MATHWORKS MATLAB documentation online: ldpcDecode, 2021. [Online] Cited 2023-02-24. Available at: https://www.mathworks.com/help/comm/ref/ ldpcdecode.html

[37] GALLAGER, R.G. Low-density parity-check codes, MIT Press Cambridge (USA), 1963. Available at: https://direct.mit.edu/ books/book/3867/Low-Density-Parity-Check-Codes DOI: 10.7551/mitpress/4347.001.0001

[38] YANG, M., RYAN, W.E., LI, Y. Design of efficiently encodable moderate-length high-rate irregular LDPC codes. IEEE Transactions on Communications, 2004, vol. 52, no. 4, p. 564-571. DOI: 10.1109/TCOMM.2004.826367

[39] CHISNALL, D. C Is Not a Low-level Language - Your computer is not a fast PDP-11. ACMqueue, 2018, vol. 16, no. 2. Available at: https://queue.acm.org/detail.cfm?id=3212479

[40] FOSSORIER, M., MIHALJEVIC, M., IMAI, H. Reduced complexity iterative decoding of low density parity check codes based on belief propagation. IEEE Transactions on Communications, 1999, vol. 47, p. 673–680. DOI: 10.1109/26.768759.

[41] CHEN, J., FOSSORIER, M. Density evolution of two improved BP-based algorithms for LDPC decoding. IEEE Communications Letters, 2002, vol. 6, p. 208–210. DOI: 10.1109/4234.1001666

[42] KSCHISCHANG, F.R., FREY, B.J., LOELIGER, H.A. Factor graphs and the sum-product algorithm. IEEE Transactions on Information Theory, 2001, vol. 47, no. 2, p. 498–519. DOI: 10.1109/18.910572

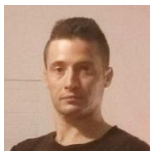
[43] HUANG, X. Single-scan min-sum algorithms for fast decoding of LDPC codes. In IEEE Information Theory Workshop - ITW '06. Chengdu (China), 2006, p. 140-143. DOI: 10.1109/ITW2.2006.323774

[44] HOCEVAR, D.E. A reduced complexity decoder architecture via layered decoding of LDPC codes. In IEEE Workshop on Signal Processing Systems. Austin–TX.(USA), 2004, p. 107-112. DOI: 10.1109/SIPS.2004.1363033

[45] INTEL CORP. USA. C++ Compiler Classic Developer Guide and Reference. [Online] Cited 2023-02-24. Available at: https://www.intel.com/content/www/us/en/develop/documentation/cppcompiler-developer-guide-and-reference/top/compiler-reference/intrinsics/intrinsics-for-avx-512-instructions.html

[46] PALENIK, T. YALDPC Toolkit, GitHub repository. [Online] Cited 2023-02-24. Available at: https://github.com/talenik/ YALDPC

Authors



Ing. Tomáš Páleník, PhD.
Slovak University of Technology in Bratislava, Slovakia
tomas.palenik@stuba.sk (corresponding author)
He received his Master's degree in 2006 from the STU and in 2010 he finalized his dissertation: "Communication system design based on an SDR platform: Exploiting the redundancy of an OFDM system" and received the Ph.D. degree in telecommunications. His research interests include digital communications systems, Orthogonal Frequency Division Multiplexing, Software Defined Radio, Error-Control Coding, IPv6 & IoT and graph algorithms in communications. During 2006–2008 he worked as an external consultant for Sandbridge-Technologies, N.Y., USA, where he implemented an LDPC decoder for an in-house developed multicore mobile processor SandBlaster. Later working in academia, he has participated in multiple research projects and also took part in the COST action IC1407. In 2019 he worked as a researcher and analyst at the AIT - Austrian Institute of Technology. He presented at several international conferences such as Wireless Innovation's SDR in Washington D.C., USA.
He is a member of the IEEE

Bc. Viktor Szitkey
Slovak University of Technology in Bratislava, Slovakia
xszitkey@stuba.sk
Research interests focuses on various technology implementation in SDR.
Student member of the IEEE.

International Journal
Information Technology Applications (ITA)

Volume 11, Number 2, December 2022

NEURAL-GENETIC CONTROL ALGORITHM FOR TWO-LINK ROBOT

Slavomír Kajan, Štefan Kozák

Abstract:

This paper deals with soft state control of non-linear dynamic model – robot. Soft methods based on neural networks and genetic algorithms demonstrate powerful problem solving ability. They are based on quite simple principles, but take advantage of their mathematical nature: non-linear iteration computation solutions. One of the ways of control of such non-linear systems is the use of neural networks as an effective controllers. In this paper a new methodology is proposed, where for neural controller structures and parameters are computed by the genetic algorithm (GA). The proposed approach is represents by direct neural controller using multilayer perceptron (MLP) network in feedback tracking control loop. The training method using GA allows find optimal adjustment of neural network weights so that high performance is obtained. The proposed control method is realized in Matlab/Simulink and demonstrated on typical non-linear systems (two-link robot).

Keywords:

Genetic algorithm (GA), MLP network, neural controller (NC), neural network, non-linear dynamic system, robot model.

ACM Computing Classification System:

Evolutionary algorithms, Artificial intelligence, Control systems.

Introduction

During the past decade there has been an intense interest in developing the soft computing methods (SCM) and techniques for a wide variety of scientific and engineering applications. The process control research in this area has been largely concerned with four SCM methods: knowledge-based systems, neural networks, fuzzy logic, genetic algorithms and various combinations of these techniques. In recent years, different fuzzy logic models are developed to cope with nonlinearities and uncertainties in many industrial systems. The fuzzy and neural models are mainly used to model system with complex physical structure. Soft computing methods can be used to optimize model parameters over a full range of input–output data. In recent years, genetic algorithm (GA) is widely used as an optimization method for training and adaptation of parameters in dynamical system. In many cases, the GA techniques are integrated in fuzzy logic and neural network structure as suitable optimization approach. GAs have many advantages over the conventional optimization methods. It does not require a complete system model and can be employed to globally search for the optimal solution. In literature as well as in practice applications of control systems occur, which are using artificial neural networks. The multilayer perceptron (MLP) neural network has good properties for direct control of non-linear systems (Chaozing Z., 1998).

For control of some classes of non-linear dynamical systems with advantage neural controllers (NC) are used. The neural network can be applied as a direct controller. It can emulate expert or another type of controller, it can be direct inverse controller, neuro-predictive controller or direct controller.

The main goal of this paper is to present an approach to the state control of an industrial robot using neural network and genetic algorithm. We know that the robots are characterized by a complex non-linear dynamical structure with un-modelled dynamics and unstructured uncertainties. These features make the designing of controllers for the robots a difficult task in the framework of state control. For the design of robot control are often used optimal control, linear-quadratic and neural control approaches. This article deals with last named type which is optimized by genetic algorithms.

1 Dynamic Model of Robot

We shall consider the robot with the kinematics structure by (Fig.1), the dynamic model of which is described in the state space:

$$\begin{aligned} \dot{x}_1 &= x_2, \quad \dot{x}_2 = \frac{m_b}{m_r} x_1 x_4^{\,2} + \frac{K_1}{m_r} u_1 \\ \dot{x}_3 &= x_4, \quad \dot{x}_4 = -\frac{2 m_b x_1 x_2 x_4}{I_{23} + m_b x_1^{\,2}} + \frac{K_2}{I_{23} + m_b x_1^{\,2}} u_2 \end{aligned} \tag{1}$$

where $m_b = m_z + m_1,\ m_r = m_{rr} + m_b + m_2$

m_z = 35 kg, *is the mass of the weight,*
m_1 = 52 kg, *is the mass of the grasp head and part of the arm,*
m_{rr}= 62.5 kg, *is reduced mass of the gear,*
m_2 = 78 kg, *is the mass of the servomotors of the arm.*

K_1 = 281 Nm, K_2 = 291 Nm *are constants of the operational values.*

$$I_{23} = I_r + (m_2 + m_3 + m_4) r_0^{\,2}$$

I_r = 82.5 kgm^2 is reduced torque of the inertia of the electric servo-motor and the gear box.

m_3 = 90 kg, m_4 = 125 kg, r_0 = 250 mm

The elements of the state vector are:

$x_1 = r$[m]; $x_2 = \dot{r}$ [m.s^{-1}]
$x_3 = \varphi$ [rad]; $x_4 = \dot{\varphi}$ [rad.s^{-1}]

$u_1(t)$ and $u_2(t)$ are control action variables

where r is translation of the arm.
φ is rotation of the arm $(0,2\pi)$

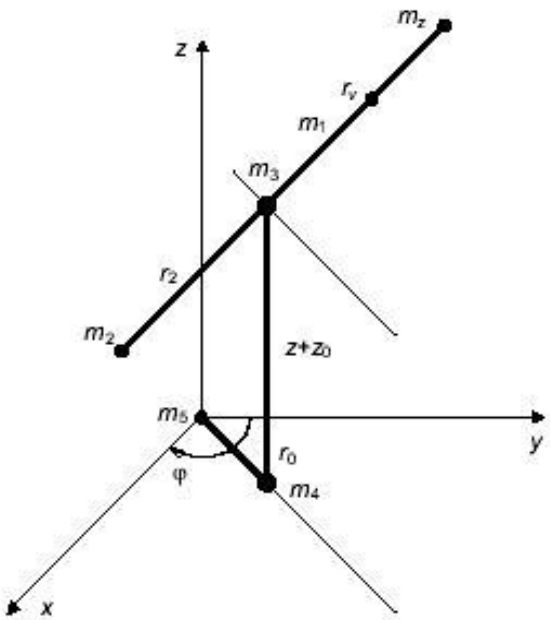


Fig.1. The kinematic scheme of the robot.

2 Neural Genetic Control Algorithm

In (Fig.2) the scheme of the neural control (NC) with optimisation of controller parameters using genetic algorithm is depicted. The main aim of euro-genetic control is computation of controller law parameters subject to the constraints and objective function defined by (3).

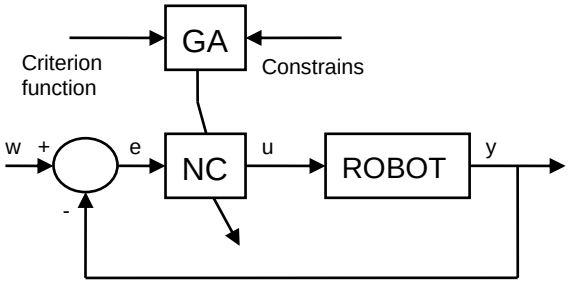


Fig.2. Block scheme of the neural – genetic control system.

Control action can be defined as nonlinear function

$$u(t) = f(e(t), y(t), y(t-1), y(t-2), W(t), t)$$

3.1 Neural Controller

Consider, the neural controller is represented by a multi-layer perceptron network (MLP) with a single hidden layer. This type of neural network is able to approximate any type of a arbitrally continuous non-linear function. The scheme of the proposed neural controller is shown in (Fig.3).

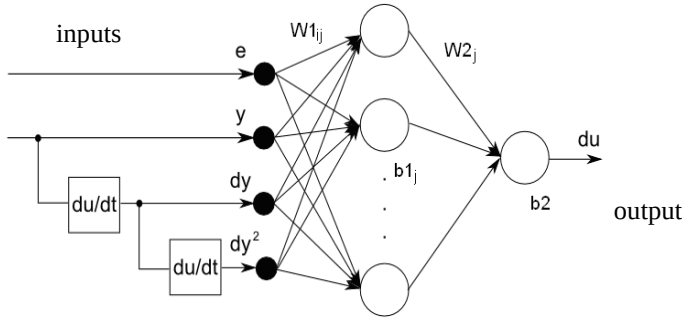


Fig.3. Scheme of the neural controller.

The inputs, states and output variables are the neural network are:

- the control errors $\boldsymbol{e_1(t)}$ and $\boldsymbol{e_2(t)}$, $\boldsymbol{e_1(t)} = w_1(t) - x_1(t)$ and $\boldsymbol{e_2(t)} = w_2(t) - x_3(t)$
- output process variable $\boldsymbol{y(t)}$,
- states of system $\boldsymbol{x(t)}=[x_1(t), x_2(t), x_3(t), x_4(t)]$.

where $w_1(t)$, $w_2(t)$ are reference values of arm translation and rotation. Outputs from the neural network are then the control values $u_1(t)$ and $u_2(t)$. The neural network with such inputs and outputs represents a non-linear state controller, where its outputs are a non-linear functions of its inputs.

In the hidden layer of the multilayer perceptron network (MLP) the hyperbolic tangent activation functions are used (tansig) in form

$$\varphi(a) = \frac{2}{1+e^{-2a}} - 1 \ . \qquad (2)$$

In the output layer linear activation function has been used. The optimized parameters are the weights between input and hidden layer $W1_{ij}$, weights between hidden and output layer $W2_{jl}$, biases in the hidden layer $b1_j$ and bias in output layer $b2_l$. For the initial setting of the neural controller parameters the Levenberg-Marquardt method is possible to use, where the data from a designed LQ controller as training data can be used.

3.2 Genetic Algorithm

A general scheme of the used GA can be described by following steps (Fig.4):

1. Initialization of the population of chromosomes.
2. Fitness evaluation of the population.
3. End if terminal conditions are satisfied.
4. Selection of parent chromosomes.

5. Crossover and mutation of the parents ⟶children.
6. Completion of the new population from the new children and selected members of the old population. Jump to the step 2.

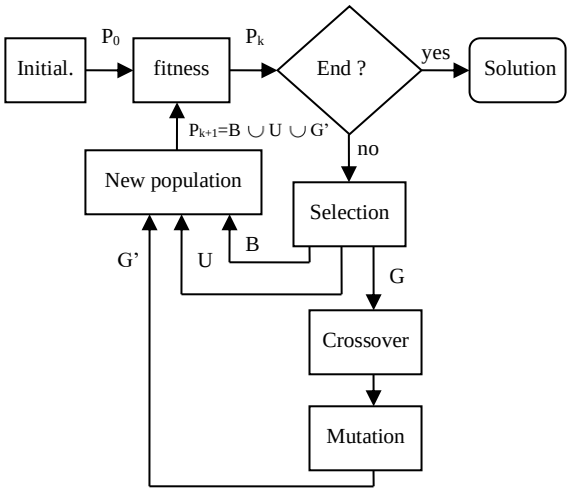


Fig.4. Block scheme of the used genetic algorithm.

The chromosome contains the set of neural network parameters - weights and biases and the optimised fitness function can use performance index in form (3), or other, where T is simulation time, e_1 and e_2 are the control errors, u_1 and u_2 are the control values and q_1, q_2, r_1, r_2 are weight constants.

$$J = \frac{1}{2}\int_0^T \left(q_1 e_1^2 + q_2 e_2^2\right)dt + \frac{1}{2}\int_0^T \left(r_1 u_1^2 + r_2 u_2^2\right)dt\,, \qquad (3)$$

After the initialization of the population, fitness of each chromosome of the population is evaluated. Fitness contains closed loop simulation with the model of the non-linear system and the neural controller and the performance index evaluation. The design procedure is based on the genetic algorithm (Fig.5).

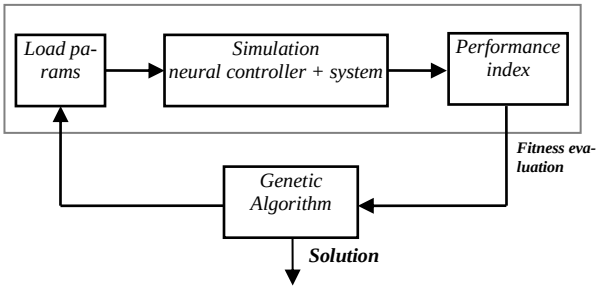


Fig.5. Block scheme of the GA-based neural controller design.

3 Simulation Results

As mentioned, for verification of the design approach NC the simulation model of robot (Fig.6) has been used. The non-linear simulation model of robot was created according to the equations (1). The simulation scheme of neural control of robot dynamic model is displayed in (Fig.7).

In case of the closed loop controller the neural controller with the MLP network with a single hidden layer is used [4]. The hidden layer contains 7 neurons for control of the robot model. The weights and biases of MLP network were optimised using GA with criterion function (3), where weights constants were setting as $q_1=1000$, $q_2=1000$, $r_1 = 0.1$, $r_2 = 0.1$.

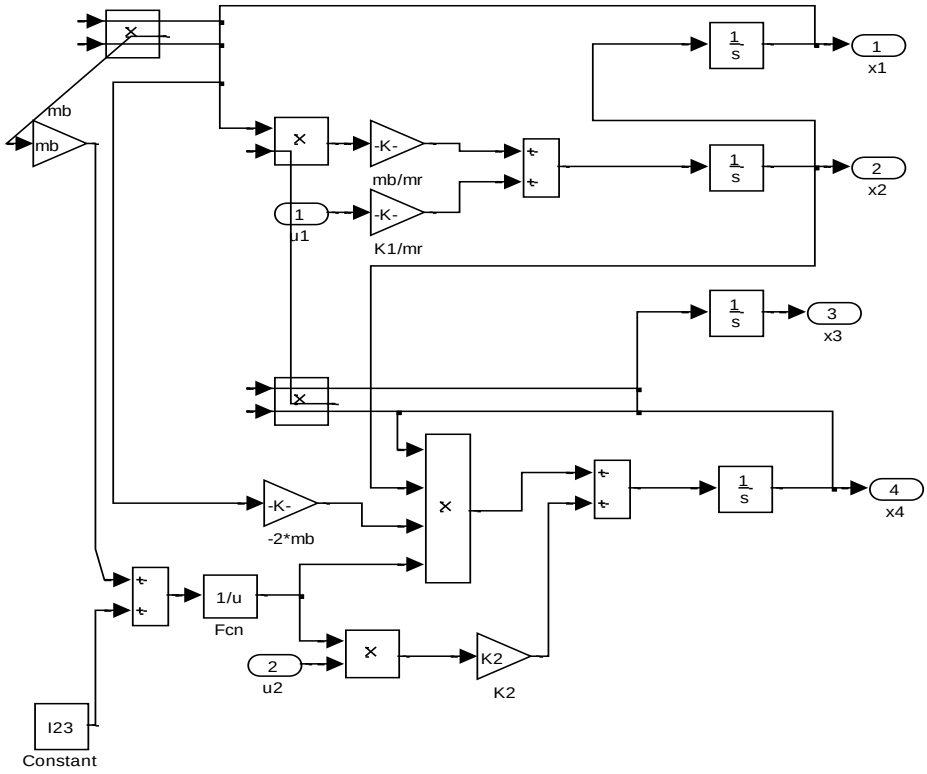


Fig.6. Simulation scheme of non-linear dynamic model of robot.

In (Fig.8) the simulation results as time-responses to arm translation x_1 and tracking of desired arm translation w_1 trajectory under the neural controller are compared. The trajectory of desired arm rotation w_2 is equal as in (Fig.8). The time-responses of system state variables in movement from state $x(0)$=[0 0 0 0] to state $x(T)$=[0.2 0 0.2 0] are displayed in (Fig.9) and (Fig.10). In these figures the simulation results as time-responses to arm translation x_1 and arm rotation x_3 are compared with desired arm translation w_1 trajectory and desired arm rotation w_2 trajectory. The time-responses of control variables u_1 and u_2 of neural controller are displayed in (Fig.11). For tracking of desired trajectory w_1 maximal value of absolute control error was 0.0027 and mean value was 0.0012. For tracking of desired trajectory w_2 maximal value of absolute control error was 0.0014 and mean value was 0.0006.

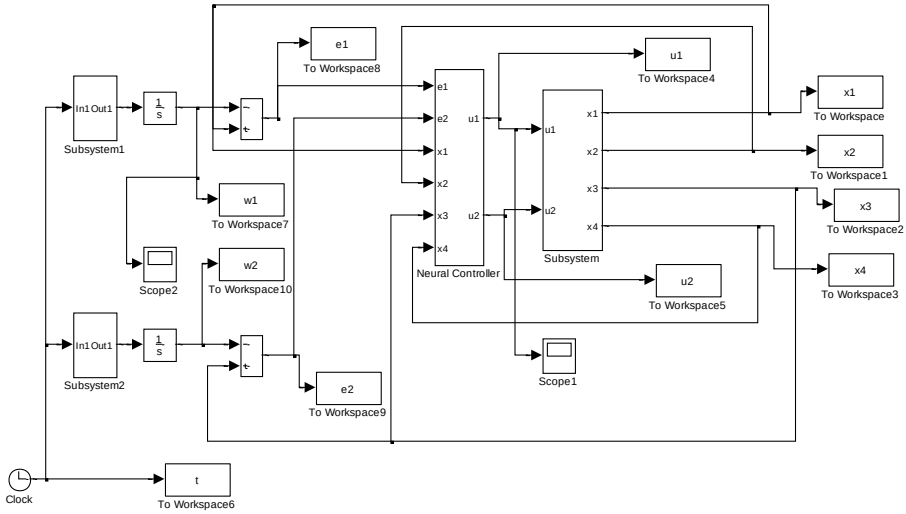


Fig.7. Simulation scheme of neural control of robot model.

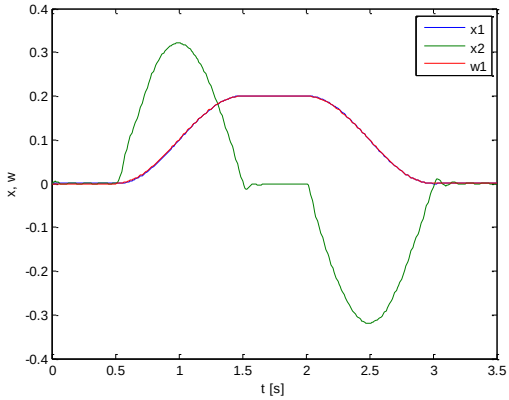


Fig.8. Time-responses of arm translation x_1 (x_2 is speed of x_1) for tracking of desired trajectory w_1

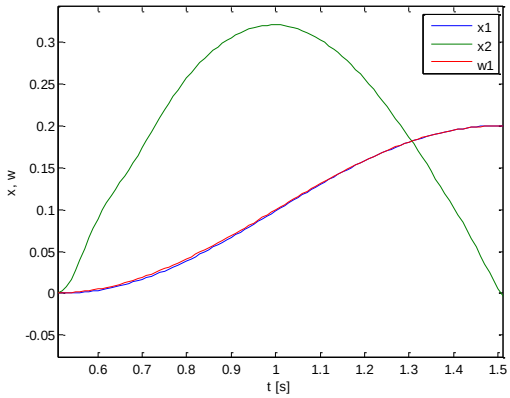


Fig.9. Detail of time-responses of arm translation x_1 (x_2 is speed of x_1) for tracking of desired trajectory w_1

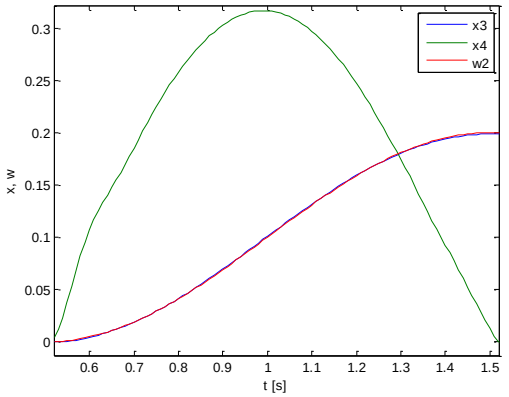


Fig.10. Detail of time-responses of arm rotation x_3 (x_4 is speed of x_3) for tracking of desired trajectory w_2

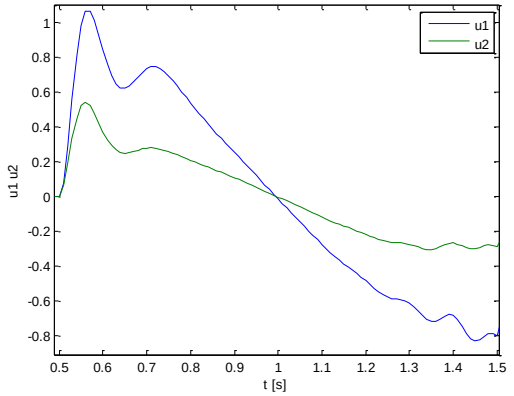


Fig.11. Detail of time-responses of control variables u_1 and u_2 under neural controller.

Conclusions

The new hybrid intelligent control methods based on neural-genetic approach presents an efficient tool for handling plants with complex dynamics as well as unstable inverse systems, time-varying time delays, occasional open-loop instability, plant model miss-matches, different uncertainties, etc. Neural controllers are able to provide high performance in control of non-linear systems. Hybrid soft computing methods based on genetic algorithms are an efficient means for neural controller parameters computation. The obtained numerical and graphical control results demonstrated in paper demonstrate that the hybrid ANN-GA control approach is well formulated and can be effectively implemented to control for robot.

Acknowledgements

The work has been supported by the grant agency VEGA no.1/0937/14.

References

[1] Chaozing, Z. (1998). Control and Dynamic System. Academic Press. New York.

[2] Z. Dideková, S. Kajan, Neural Control of Non-Linear Processes Designed by Genetic Algorithms, In: ELITECH `09 : 11th Conference of Doctoral Students. Bratislava, STU in Bratislava FEI, 2009. - ISBN 978-80-227-3091-4.

[3] R. Gašparík, Optimálne riadenie manipulátora, Diplomová práca, FEI STU Bratislava, 2007

[4] A. Jadlovská, Modelovanie a riadenie dynamických procesov s využitím neurónových sietí (Edícia vedeckých spisov. FEI TU Košice, ISBN 80-8894122-9, 2003, in Slovak language).

[5] A. Jadlovská, Using Forward and Inverse Neural Models for Solving Optimal Tracking Problem of Non-Linear System, Journal of Electrical Engineering, Vol. 55, No. 5-6, 2004, pp. 150-155

[6] I. Sekaj, Genetic Algorithm Based Controller Design,
In: 2nd IFAC conference Control System Design'03, Bratislava, 2003.

[7] M. T. Hagan, M. B. Menhaj, Training Feedforward Networks with the Marquardt Algorithm, Submitted to the IEEE Proceedings on Neural Network, 1994.

[8] D. E. Goldberg, Genetic Algorithms in Search, Optimisation and Machine Learning (Addisson-Wesley, 1989).

[9] I. Sekaj, Evolučné výpočty a ich využitie v praxi (Iris, Bratislava, 2005, in Slovak language).

[10] I. Sekaj, Evolutionary Based Controller Design, In: Evolutionary Computation, Book edited by: Wellington Pinheiro dos Santos, ISBN 978-953-307-008-7, pp. 239-260, October 2009, In-Tech, Vienna, Austria. 2009.

[11] R. C. Dorf, Modern Control Systems (Addison-Wesley publishing Company, 5th edition, 1990).

Authors

Ing. Slavomír Kajan, PhD.
Institute of Robotics and Cybernetics,
Faculty of Electrical Engineering and Information Technology,
Slovak University of Technology in Bratislava, Slovakia
slavomir.kajan@stuba.sk

His research interests are artificial intelligence, neural networks and applications of AI methods in medicine, control systems and robotics.

prof. Ing. Štefan Kozák, PhD.
Faculty of Informatics,
Pan-European University in Bratislava, Slovakia
stefan.kozak@paneurouni.com

His research interests include system theory, linear and nonlinear control methods, numerical methods and software for modeling, control, signal processing, IoT, IIoT and embedded intelligent systems for digital factory in industry and medicine.

International Journal
Information Technology Applications (ITA)

Volume 11, Number 2, December 2022

COMPARISON OF FIRST-PRINCIPLES AND EXPERIMENTAL VEHICLE MODELS

Dávid Mikle, Juraj Račkay

Abstract:

This article deals with the development and comparison of different vehicle models to be used in vehicle dynamics control of an electric all-wheel drive vehicle in the future. The aim is to verify the accuracy of lateral and longitudinal motions of selected vehicle models based on two different approaches. Using first principles describing basic vehicle dynamics, single and the twin track vehicle models are derived in form of systems of nonlinear differential equations. Using experimental identification, five vehicle state space models of orders are identified based on measurements on a real vehicle. The experiments performed were three different test maneuvers. Vehicle models are then compared using the measured data with the simulation results in MATLAB.

Keywords:

Vehicle dynamics, vehicle model, state-space model, MATLAB, systemIdentification toolbox.

ACM Computing Classification System:

Dynamic systems control.

1 Introduction

A significant challenge in electric vehicles with all-wheel drive is the way how to control whole powertrain to improve vehicle dynamics while increasing safety and stability. This can be achieved by various control strategies with different complexity. Each of these strategy is based on physical laws and mathematical descriptions of vehicle motion – vehicle model.

A wide range of simplified vehicle dynamics models is available in the literature, which are able to accurately represent basics of the force and moment dynamics [1]. However, in case of critical vehicle situations, it is essential to assume a more descriptive model considering the couplings between vehicle components. The application of multi-body dynamics for the analysis of vehicle handling problems was firstly discussed in [2], which was later applied in several complex multibody dynamics models in [3] and [4]. A complete multi-body model of the vehicle including the suspension geometry and tire characteristics was introduced in [5]. Simultaneously several multi DOF nonlinear multi-body dynamic models that can accurately represent almost all physical characteristics of the vehicle including suspension and tire dynamics has been proposed in [6] and [7]. But all these models have huge demands in computation power. To scale down the computational requirement, an intermediate multi-body model was proposed by authors of [8], which was well accepted in automotive industry for less critical applications.

However, these models are rarely used in their raw form, because of demand of simultaneously solve the combined differential or algebraic system of equations, which brings convergence issues with an even more computational requirements to the system.

Therefore a dominant amount of vehicle control strategies, for instance, the side slip control, yaw control, and trajectory control are based on a linearized version of vehicle operating condition known as the single track model. Example of step by step derivation of such a model can be found in [9]. In this model the vehicle is considered in its most simplified form, neglecting tire's side sliding, all lifting, rolling and pitching motion and assuming constant mass distribution on the axles. Of course, these simplifications come at the cost of reduced accuracy in the model as compared with actual vehicle motion, but model outputs still fits reality enough for purposes of vehicle control.

2 Vehicle Model Development

In vehicle planar movement analysis, with neglecting the internal forces, vehicle has 6 degrees of freedom (DOF). Number of DOF can be additionally reduced by several assumptions when considering only longitudinal, lateral and yaw motion. With used simplifications, in next chapters we consider vehicle of 3 DOF.

A Single track model derivation

Single-track model (Fig.1) describes well basic drive processes without much effort in modeling and parametrization. In this paper for single track model representation, we will take derivation of model from work of Efremov [9].

This model is used to describe planar vehicle motion, with next simplifications:

- All lifting, rolling, and pitching motion is neglected.
- Vehicle mass is concentrated at the center of gravity.
- Front and rear tires are represented as one single tire on each axle.
  Imaginary contact points of tires and surface are assumed to lie along the center of axles.
- Pneumatic trail and aligning torque resulting from a side-slip angle of a tire are neglected.
- Mass distribution on the axles is assumed to be constant.

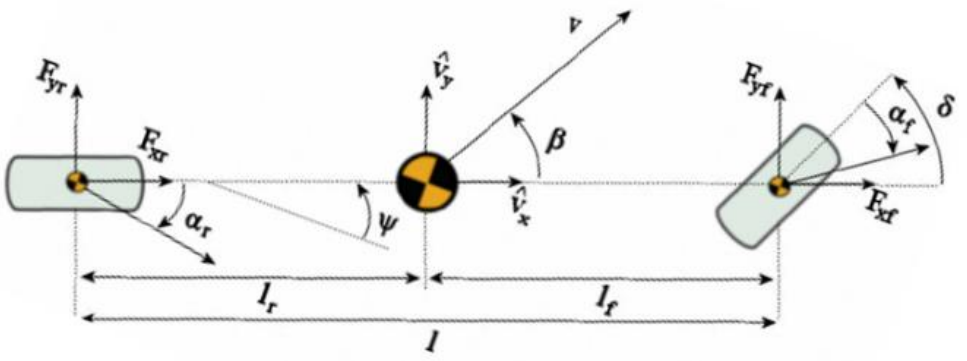


Fig.1. Single track model.

With assumed simplifications above, for considered vehicle three equations of motion exists:

$$F_x = -mv\left(\dot{\beta} + \dot{\Psi}\right)\sin(\beta) + m\dot{v}\cos(\beta) \quad (1)$$

$$F_y = -mv\left(\dot{\beta} + \dot{\Psi}\right)\cos(\beta) + m\dot{v}\sin(\beta) \quad (2)$$

$$M_Z = I_Z\ddot{\Psi} \quad (3)$$

Where, *m* is the vehicle's mass, *v* is the velocity of the center of gravity COG of the vehicle, β is the side-slip angle, ψ is the yaw angle, I_Z is the moment of inertia of the vehicle around the *z* axis. On the other side of the equations, there are forces acting on the COG of the vehicle along with *x* (F_X) and *y* (F_Y) axes and the moment acting around the *z* axis (M_Z).

The resulting system of nonlinear differential equations describing the steering angle projection and vehicle dynamics can be written as follows:

$$\dot{\beta} = -\dot{\psi} + \frac{1}{mv}\left(F_y\, cos(\beta) - F_x\, sin(\beta)\right) \quad (4)$$

$$\dot{v} = \frac{1}{m}\left(F_y\, sin(\beta) + F_x\, cos(\beta)\right) \quad (5)$$

$$\ddot{\psi} = \frac{1}{I_z}M_z \quad (6)$$

For this system sum of forces acting on vehicle in each direction can be derivated as:

$$F_x = F_{xf}\,\cos\left(\delta_f\right) - F_{yf}\,\sin\left(\delta_f\right) + F_{xr}\,\cos(\delta_r) - F_{yr}\,\sin(\delta_r) \quad (7)$$

$$F_y = F_{xf}\,\sin(\delta_P) + F_{yf}\,\cos\left(\delta_f\right) + F_{xr}\,\sin(\delta_r) + F_{yr}\,\cos(\delta_r) \quad (8)$$

$$M_z = l_f\left(F_{xf}\sin\left(\delta_f\right) + F_{yf}\cos\left(\delta_f\right)\right) - l_r\left(F_{xr}\sin(\delta_r) + F_{yr}\cos(\delta_r)\right) \quad (9)$$

Where δ_f , δ_r are steering angles of the front and the rear wheel, l_f , l_r is the distance from the vehicle's COG to the front and rear axle. Forces F_{Yf} , F_{Yr}, F_{Xf} , and F_{Xr} are forces acting on tires.

The tire dynamics is described by famous tire model Pacejka Magic formula, which can be used for estimation not only the lateral and longitudinal forces' impact on a tire, but also all the torques acting on a wheel around all axis. It has a straightforward calculation, and the same formula is used to estimate all the forces and torques using different sets of coefficients. The general Simplified Pacejka Magic formula has the following equation:

$$F = DF_Z\sin\left(C\ arctg(B\alpha - E(B\alpha - arctg(B\alpha)))\right) \quad (10)$$

Where *D*, *C*, *B*, and *E* is the set of shaping coefficients, F_Z is a wheel-load and α is tire side slip angle.

All these equation were implemented in single track Simulink model to verify performance of the derivated vehicle model.

B Twin track model derivation

For twin track model, in our previous work with Račkay [10], we developed Efremous single track model to cover all four wheels of vehicle. This model is based on the same three equations motion (1), (2) and (3), however in equation of acting forces (7), (8) and (9) we expand wheels elements from front *f* and rear *r* wheel to front right *fr*, front left *fl* and rear right *rr* and left *rl* wheels to cover remain dynamics. Expanded equations are in form:

$$F_x = F_{x_{PP}}\cos(\delta_{PP}) + F_{x_{LP}}\cos(\delta_{LP}) + F_{x_{PZ}}\cos(\delta_{PZ}) + F_{x_{LZ}}\cos(\delta_{LZ}) - F_{y_{PP}}\sin(\delta_{PP}) - F_{y_{LP}}\sin(\delta_{LP}) - F_{y_{PZ}}\sin(\delta_{PZ}) - F_{y_{LZ}}\sin(\delta_{LZ}) \quad (11)$$

$$F_y = F_{x_{PP}}\,sin(\delta_{PP}) + F_{x_{LP}}\,sin(\delta_{LP}) + F_{x_{PZ}}\,sin(\delta_{PZ}) + F_{x_{LZ}}\,sin(\delta_{LZ}) + F_{y_{PP}}\,cos(\delta_{PP}) + F_{y_{LP}}\,cos(\delta_{LP}) + F_{y_{PZ}}\,cos(\delta_{PZ}) + F_{y_{LZ}}\,cos(\delta_{LZ}) \quad (12)$$

$$M_z = l_P\{F_{x_{PP}}\,sin(\delta_{PP}) + F_{y_{PP}}\,cos(\delta_{PP}) + F_{x_{LP}}\,sin(\delta_{LP}) + F_{y_{LP}}\,cos(\delta_{LP})\} - l_Z\{F_{x_{PZ}}\,sin(\delta_{PZ}) + F_{y_{PZ}}\,cos(\delta_{PZ}) + F_{x_{LZ}}\,sin(\delta_{LZ}) + F_{y_{LZ}}\,cos(\delta_{LZ})\} + b_P\{F_{x_{PP}}\,cos(\delta_{PP}) - F_{y_{PP}}\,sin(\delta_{PP}) + F_{x_{PZ}}\,cos(\delta_{PZ}) - F_{y_{PZ}}\,sin(\delta_{PZ})\} - b_L\{F_{x_{LP}}\,cos(\delta_{LP}) - F_{y_{LP}}\,sin(\delta_{LP}) + F_{x_{LZ}}\,cos(\delta_{LZ}) - F_{y_{LZ}}\,sin(\delta_{LZ})\} \quad (13)$$

Again all these equation were implemented in twin track Simulink model to verify performance of the derivated vehicle model.

3 Vehicle Model Identification

Vehicle model identification is an alternative process to model derivation, where you identify models with different representation from measured data. It is recommended to start by estimating the parameters of simpler models structures and if the model performance is poor, you gradually increase the complexity of the model structure. Ultimately, you choose the simplest model that best describes the dynamics of your system.

Vehicle itself is a complex systems with multiple inputs and multiple outputs (MIMO) and such systems are often more challenging to model because of couplings between several inputs and outputs. MIMO models are often covered via state-space representations, since the model structure complexity is easier to deal with [11]. State-space model use state variables to describe a system by a set of first-order differential or difference equations, rather than by one or more nth-order differential or difference equations. The general state-space description has the following form:

$$\dot{x} = Ax + Bu \quad (14)$$

$$y = Cx + Du \quad (15)$$

(14) is called state equation with state vector *x*, (15) is output equation, *u* is input vector, *y* is output vector, coefficient *A* is system matrix, *B* is control input matrix, *C* is output matrix and *D* is feedforward matrix.

The state-space model structure is a good choice for quick estimation because it requires only one user input, the model order, n. In Matlab, model identification is supported in system identification toolbox [11] with user-friendly GUI.

In this work we identified five state space models of 3rd, 4th, 5th, 6th and 7th model order from measured experiments.

4 Experiments

For experiments we used conventional vehicle with front wheel drive, equipped with multiple sensors. For position measurements was used self-build RTK GNSS receivers witch absolute accuracy of approx. 20mm, attached on each axle. We modified receiver's chips to be able to perform measurement at 20Hz frequency, what is in navigation systems pretty high performance. We also used a 9 DOF inertial measurement unit from XSENSE. As a vehicle output we measured accelerations and angular velocities in each directions witch frequency of 100 Hz. Position of used sensors is shown at following pictures:

Fig.2. Position of GNSS antennas (Top), position of inertial measurement unit (Bottom).

In experiments we also performed a reading of CAN Bus of vehicle. It required a small intervention to vehicle's electric wiring, because a Volkswagen concern cars have switchable CAN bus on OBD2 connector, which means that it is not possible to read any data directly from it, because it requires a data polling from CAN Bus gateway. So we connected directly on CAN BUS wires in dashboard connectors and soldered wires to them. For data logging we used a self-build control unit with CAN transceiver with C++ library build for parse all data.

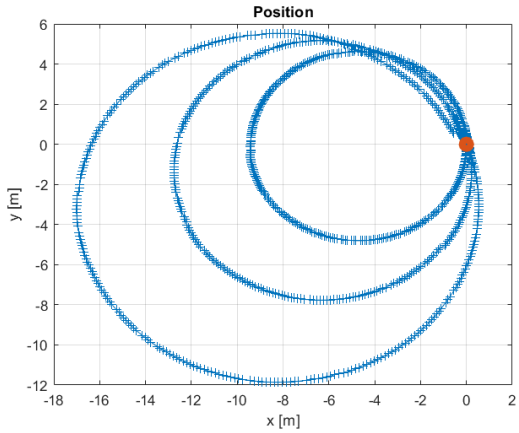


Fig.3. Trajectory of constant steer maneuver for multiple steering angles.

As experiments we performed a three different test maneuvers. First we performed a constant steer drive for different steering wheel angles at about 6 km/h, shown in (Fig.3). From trajectory of this maneuver we measured the real cornering radius for each steering wheel angle and from Ackerman geometry we calculated actual steering angles of the wheels. Then by polynomial regression we identified equation of steering wheel factor characteristic.

Next we performed a step steer maneuver (Fig. 4), where we accelerate with straight wheels from rest to about 30km/h and then applied a steering step to minus 15 degree. Data from this maneuver we used for model identification in MATLAB.

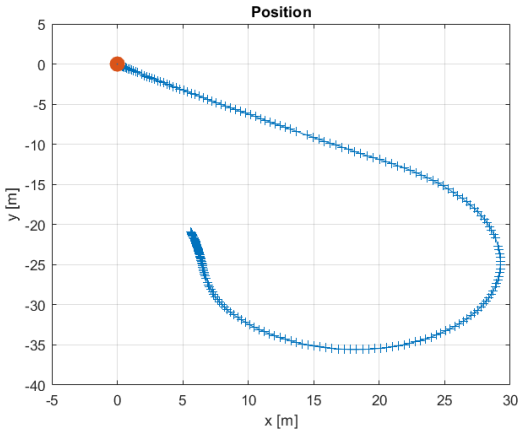


Fig.4. Trajectory of step steer maneuver.

Finally a third maneuver was single line change (Fig. 5), where we accelerate with straight wheels from rest to about 60km/h and the applied a minus 7degree input and immediately changed it to plus 7degree and then back to straight wheel. Data from this maneuver we used for verification of simulated models outputs.

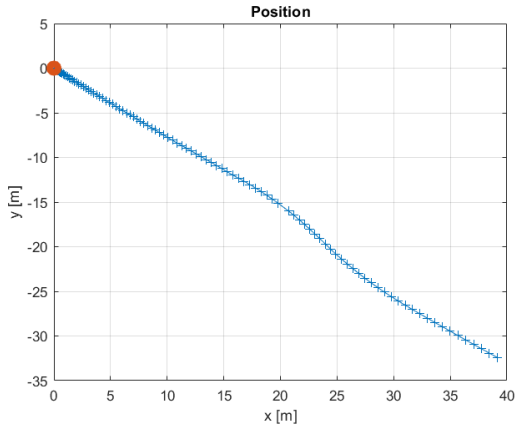


Fig.5. Trajectory of single line change maneuver.

5 Simulation

To verify derivated and identified vehicle models, we have simulated data from single line change experiment as test maneuver in MATLAB, to simulate the dynamic responses of models. The aim of this simulation is to verify the accuracy of lateral and longitudinal motion of models.

We simulated this maneuver with single track, twin track and identified state space models with 3rd – 7th order. Results of line change simulations are shown in the following figures.

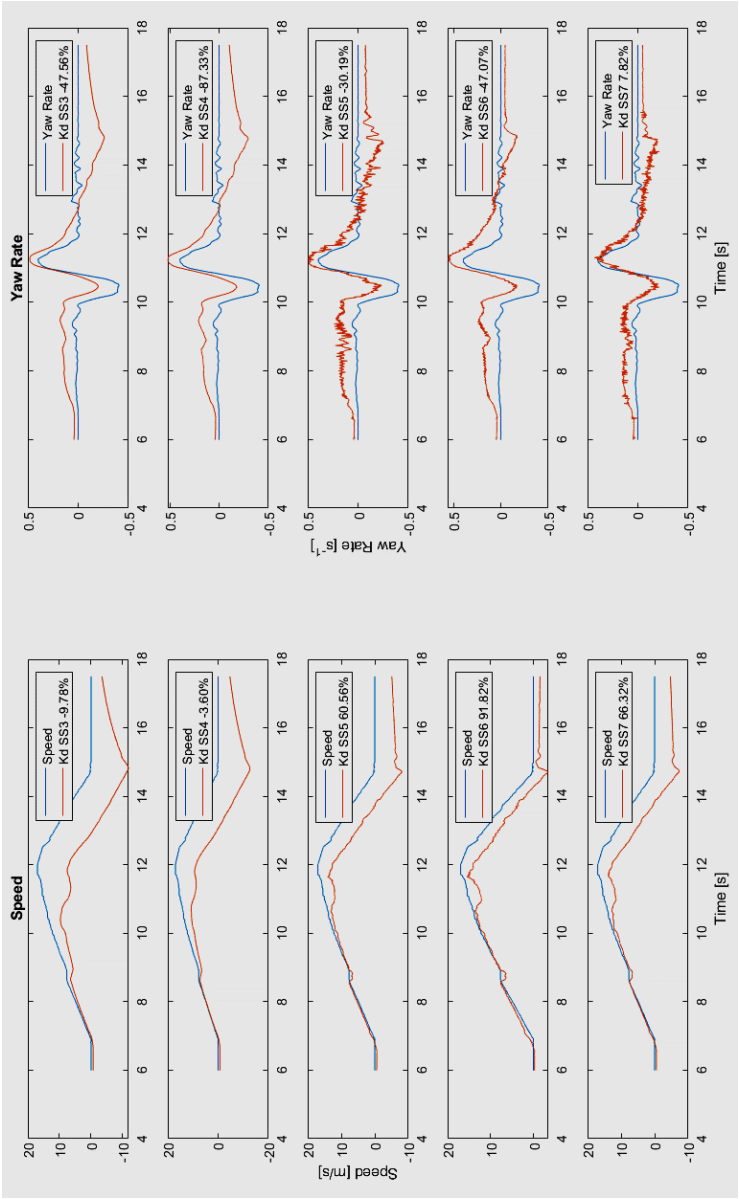


Fig.6. Comparison between measured vehicle speed (Left) and yaw rate (Right) and simulated results from identified State Space model with 3rd order SS3(1st row), 4th order SS4(2nd row), 5th order SS5(3rd row), 6th order SS6(4th row) and 7th order SS7(5th row).

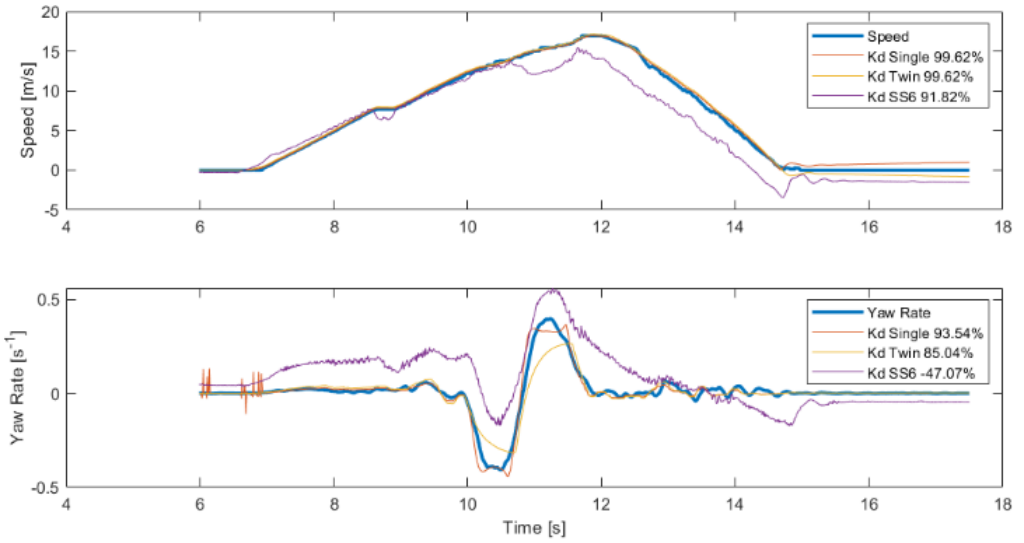


Fig.7. Comparison between measured vehicle speed (Top) and yaw rate (Bottom) and simulated results from single and twin track model and SS6.

From (Fig.6) with results of simulation of all identified models we can see that models with 3rd and 4th order (SS3, SS4) has a poor results, where coefficient of determination was negative number in both simulated outputs. Models with 5th, 6th and 7th order has significantly better results at simulated Speed, where model SS6 has Kd = 91.82%, what can be considered as great result. Also these models has slightly better result in simulated Yaw rate, however cause slight oscillations. In overall, from these results we can consider SS6 as a best identified state space vehicle model.

In (Fig.7) we can see comparison of model SS6, single and twin track model with actual measured data. From results we can see that derivated single and twin track model can even with many considered simplification pretty accurately represent longitudinal and lateral motion of vehicle. Coefficient of determination in simulated Speed for both models is approximately 99% and simulated Yaw rate is 85.05% for twin track and 93.54% for single track model.

6 Conclusion

The paper presents two most common vehicle model design approaches – a theoretical derivation based on first principles, and experimental identification, and their simulation-based comparison. In the design of any vehicle model, the most important objective is to obtain desired accuracy considering applied simplifications. Effectiveness accuracy of longitudinal and lateral motion of the developed models has been verified via line change simulations.

Based on simulation results, we can conclude that the developed single track and twin track vehicle models work as expected despite having used several simplifications in the theoretical background. The developed models are now ready to be implemented in the control system of the racing car of the Slovak formula student team STUBA Green Team.

Acknowledgement

The paper was supported by the Ministry of Education, Science and Sport of the Slovak Republic under the project KEGA 010STU-4/2023.

References

[1] M. Ehsani, Electric, hybrid, and fuel cell vehicles, introduction. In Transportation technologies for sustainability pp. 492–493, 2013. Springer New York, ISBN 978-1-4614-5844-9.

[2] R. R. McHenry, An analysis of the dynamics of automobiles during simultaneous cornering and ride motions. Computer-Aided Design, 1(3), pp.19–32, 1969. https://doi.org/10.1016/S0010-4485(69)80082-5.

[3] M. V. Blundell, The modelling and simulation of vehicle handling part 1: Analysis methods. Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics, 213(2), pp. 103–118, 1999. https://doi.org/10.1243/1464419991544090.

[4] M. V. Blundell, The modelling and simulation of vehicle handling part 4: Handling simulation. Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics, 214(2), pp. 71–94, 2000. https://doi.org/10.1243/1464419001544250.

[5] W. Kortm, Review of multibody computer, codes for vehicle system dynamics. Vehicle System Dynamics, 22(sup1), pp. 3–31, 1993. https://doi.org/10.1080/00423119308969463.

[6] S. Hegazy, H. Rahnejat, K. Hussain, Multi-body dynamics in fullvehicle handling analysis. Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics, 213(1), pp. 19–31, 1999. https://doi.org/10.1243/1464419991544027.

[7] A. A. Shabana, Dynamics of multibody systems. Cambridge University Press, 2009. https://doi.org/10.1017/cbo9781107337213.

[8] M. Jaiswal, G. Mavros, H. Rahnejat, P. D. King, A multi-body dynamics approach for the study of critical handling manoeuvres on surfaces with uneven friction. TU Delft, 2007. ISBN 9789081176811.

[9] EFREMOV, Denis. Single-Track model derivation. CVUT Praha, (2018).

[10] Račkay, J. Matematické modelovanie jazdnej dynamiky vozidla. bachelor thesis in Slovak language, STU Bratislava, 2020.

[11] L. Ljung, System Identification Toolbox User's guide. MathWorks, 2020.

Authors

M.Sc. Dávid Mikle
Institute of Automotive Mechatronics
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology in Bratislava, Slovakia
david.mikle@stuba.sk

He received the B.Sc. degree in Automotive Mechatronics and the M.Sc. degree in Applied Mechatronics and Electromobility from the Slovak University of Technology in Bratislava, Slovakia, in 2016 and 2018. During the M.Sc. degree, he was part of the slovak formula student team STUBA Green Team. Currently he is an interrupted Ph.D student in Mechatronic Systems and lecturer in subject Dynamics of Vehicles at the same university. His research interests include vehicle dynamics modeling and control, applied to electric vehicles.

MSEng. Juraj Račkay
Institute of Automotive Mechatronics
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology in Bratislava, Slovakia
xrackay@stuba.sk

He received the Bc. degree in Automotive Mechatronics and the MSEng. degree in Applied Mechatronics and Electromobility from the Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Slovakia, in 2020 and 2022, respectively. Presently, he is a PhD student at the same university. His research interests include vehicle dynamics modelling and control of an electric vehicle traction system.

SIMULATION-BASED MODEL CONTROL USING STATIC HAND GESTURES IN MATLAB

Slavomír Kajan, Jozef Goga

Abstract:

This paper deals with the domain of simulation-based models control using static hand gestures in the MATLAB environment. The aim of this paper was to design an algorithm for visual static hand gesture recognition with high classification accuracy. For this recognition task, different convolutional neural network models (CNN) were tested. For the successful training of CNN, stochastic backpropagation of error was used. Training of CNN was implemented on the graphic card using toolboxes such as Neural Network and Parallel Computing from the MATLAB program package. For the training and testing of CNN a database of 35 static hand gestures was used. The proposed CNN gesture recognition system has been implemented in the simulation scheme due to the need of setting different model parameters.

Keywords:

Hand gesture recognition, neural networks, graphic card, parallel computing, MATLAB.

ACM Computing Classification System:

Image recognition, artificial intelligence, parallel computing.

1 Principle of Gesture Recognition

The hand gesture recognition itself can be implemented in a several consecutive steps. General scheme of hand gesture recognition system is shown in (Fig.1). In this recognition task, the Kinect v2 sensor was used.

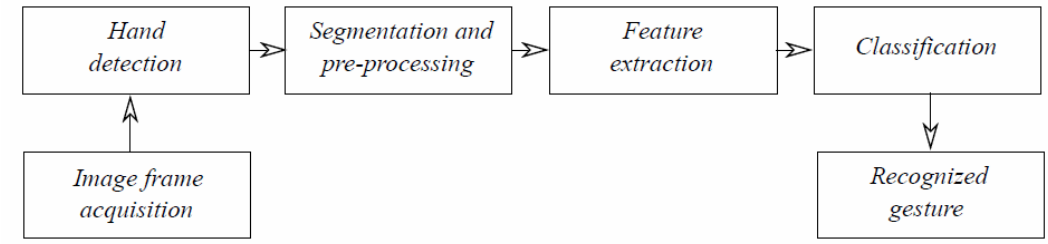


Fig.1. General scheme of gesture recognition system.

The captured image from the sensors is processed and stored in appropriate form. Subsequently, the gesture in an acquired frame may be segmented or image features may be extracted from the entire input frame [1]. Under the extraction of features, we understand the evaluation of quantitative or statistical indicators which represent given gesture based on the suitable metrics.

It can be the number of stretched fingers, angles between fingers, fingertip markings, fingertips positions [3], histograms, Voronoi diagrams, and other statistical and quantitative indicators. These extracted features are an input to a computational model, whose job is to correctly classify the given gesture. With regard to the complexity of this task, in this paper we dealt only with the recognition of static hand gestures. Hand gesture recognition and associated problems such as hand segmentation were elaborated in many papers. Some authors have used a color-based image analysis approach such as a color histogram based on statistical methods [3], thresholding the tints of a color model [4], or gesture capturing with color gloves [5] that are easier to segment. Approaches based on depth analysis are in general more successful than color-based methods, but those approaches assume the hand is the closest object in the frame. One possible solution is mapping from the depth data to a corresponding part of a color image [3], or hand segmentation based on the distance limited by color bracelet [6].

Use of the latest deep neural network models does not always improve classification accuracy, which ranges from 70% - 90%, depending on the specific architecture of the neural network [2][8 - 10]. In this paper, we used different architectures of convolutional neural networks for this challenging static hand gesture recognition task.

2 Gesture Recognition Using Convolutional Neural Network

The general structure of the convolutional neural network (CNN) is displayed in (Fig.2).

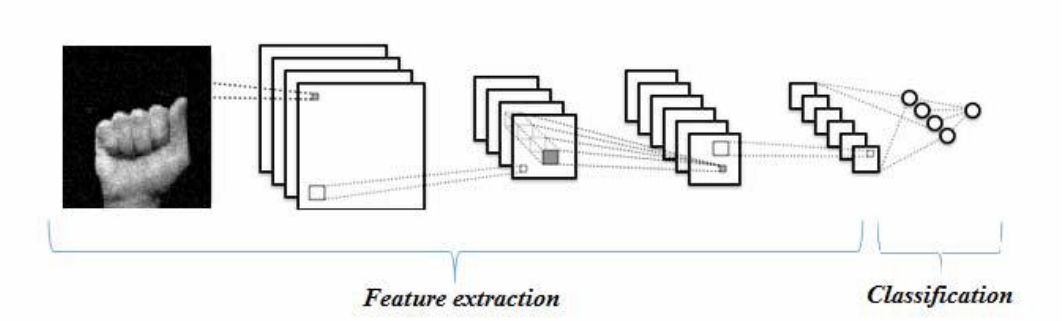


Fig.2. The general architecture of convolutional neural network used for gesture recognition.

Convolutional neural networks are designed specifically for pattern recognition with a large degree of invariance to shift, change of scale or other forms of distortion. These properties are gained through learning process. However, the structure of convolutional neural networks involves certain forms of constraints. The most commonly used types of layers in the network architecture are convolutional, pooling, and fully-connected layers. By arranging these computational layers, we create the overall architecture of the convolutional neural network.

The convolutional layer is the main computing block in the overall network architecture. Its input is usually 3-dimensional image tensor, which contains 3 color image channels. As the title of this layer suggests, a discrete convolution of input with the kernel is performed there. When computing, we move the kernel in the direction of the width and height of the input image with the selected step, for all their mutual positions, creating a feature map. By learning, these feature maps are activated when different image patterns are detected, such as edges at a certain angle, color clusters, and others.

The pooling layer performs sub-sampling of the input tensor, thereby reducing the size of the feature map, but retaining the most important information contained therein. This greatly reduces the spatial magnitude of the feature map, as well as the number of parameters and the computational difficulty of the neural network.

The neurons in the fully-connected layer of the convolutional network have, as the name suggests, all connections to the neurons in the previous layer. This is, therefore, a classical multilayer perceptron network. Outputs from convolutional and pooling layers represent high-level features extracted from input images. These features are an input into the fully-connected layer of the convolutional network, and its role is to correctly classify them.

3 Training and Testing of the Convolutional Neural Network

For the training and testing of CNN a database of 35 static hand gestures was used. This database contains static gestures of the American Sign Language (ASL), which was changed due to the dynamic characters "J" and "Z". The database was created by 65 volunteers (60 men and 5 women), which consists of 5 frames per gesture [6]. Overall, we created 175 frames (35 gestures times 5 frames) for color, infrared and depth images. A total of 525 images per person. The resulting database has 34 125 images (65 people times 525 frames). Data from 50 people was used in the training process and data from 15 people was used in the testing process.

These images were then modified into a form suitable for training the convolutional neural network. In the original color image from the Kinect sensor, with a resolution of 1920x1080 pixels, we only segmented the hand area. In this way, we created a square image suitable for training with a resolution of 640x640 pixels. Segmentation of hand gestures was also done for original depth and infrared frames. The original resolution of 512x424 pixels was adjusted to a gesture frame with a size of 156x156 pixels.

Fig.3. A preview of frames in the database.

The first tested architecture of CNN (Table 1) consist of three convolutional layers. The second tested architecture (Tab.2) has the same number of convolutional layers, but fully-connected layer with 50% dropout was added. This omission is used to make the neural network better distribute trained information throughout the network as any neuron may be omitted in the next epoch. The last tested architecture (Tab.3) has been extended to four convolutional layers, where kernel size is the same in all layers.

For the successful training of CNN, stochastic backpropagation of error was used [13]. Training of CNN was implemented on the graphic card using toolboxes such as Neural Network and Parallel Computing from the MATLAB program package. In Figure 4 is shown the comparison of classification score during the training process for all tested neural network architectures. The comparison of classification accuracy of those architectures is displayed in Table 4. Examples of learned features acquired by transition of the random input color image through individual convolutional layers of trained CNN with architecture A-2 is shown in Figure 5 [6].

Table 1. First architecture of CNN – A-1.

No.	*Layer type*	*Parameters*
1	Image Input	156x156x3 images
2	Convolution	3x3@18 kernels with stride [1 1]
3	ReLU	-
4	Max Pooling	2x2 max pooling with stride [2 2]
5	Convolution	6x6@36 kernels with stride [1 1]
6	ReLU	-
7	Max Pooling	2x2 max pooling with stride [2 2]
8	Convolution	9x9@72 kernels with stride [1 1]
9	ReLU	-
10	Max Pooling	2x2 max pooling with stride [2 2]
11	Fully-Connected	35 neurons, fully-connected layer
12	Softmax	-
13	Classification Output	35 classes, cross-entropy error

Table 2. Second architecture of CNN – A-2.

No.	*Layer type*	*Parameters*
1	Image Input	156x156x3 images
2	Convolution	5x5@18 kernels with stride [1 1]
3	ReLU	-
4	Max Pooling	2x2 max pooling with stride [2 2]
5	Convolution	6x6@36 kernels with stride [1 1]
6	ReLU	-
7	Max Pooling	2x2 max pooling with stride [2 2]
8	Convolution	9x9@72 kernels with stride [1 1]
9	ReLU	-
10	Max Pooling	2x2 max pooling with stride [2 2]
11	Fully-Connected	35 neurons, fully-connected layer
12	ReLU	-
13	Dropout	50% dropout
14	Fully-Connected	35 neurons, fully-connected layer
15	Softmax	-
16	Classification Output	35 classes, cross-entropy error

Table 3. Last architecture of CNN – A-3.

No.	*Layer type*	*Parameters*
1	Image Input	156x156x3 images
2	Convolution	5x5@18 kernels with stride [1 1]
3	ReLU	-
4	Max Pooling	2x2 max pooling with stride [2 2]
5	Convolution	5x5@36 kernels with stride [1 1]
6	ReLU	-
7	Max Pooling	2x2 max pooling with stride [2 2]
8	Convolution	5x5@72 kernels with stride [1 1]
9	ReLU	-
10	Max Pooling	2x2 max pooling with stride [2 2]
11	Convolution	5x5@72 kernels with stride [1 1]
12	ReLU	-
13	Max Pooling	2x2 max pooling with stride [2 2]
14	Fully-Connected	128 neurons, fully-connected layer
15	ReLU	-
16	Dropout	50% dropout
17	Fully-Connected	35 neurons, fully-connected layer
18	Softmax	-
19	Classification Output	35 classes, cross-entropy error

Table 4. Comparison of different CNN architectures – classification score (accuracy).

Architecture	*Accuracy [%]*	
	Train	*Test*
A-1	100.00	72.14
A-2	90.63	84.43
A-3	98.44	81.14

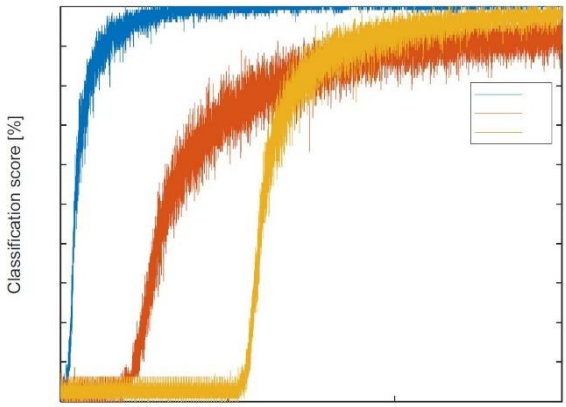


Fig.4. Classification score during the training process for different CNN architectures.

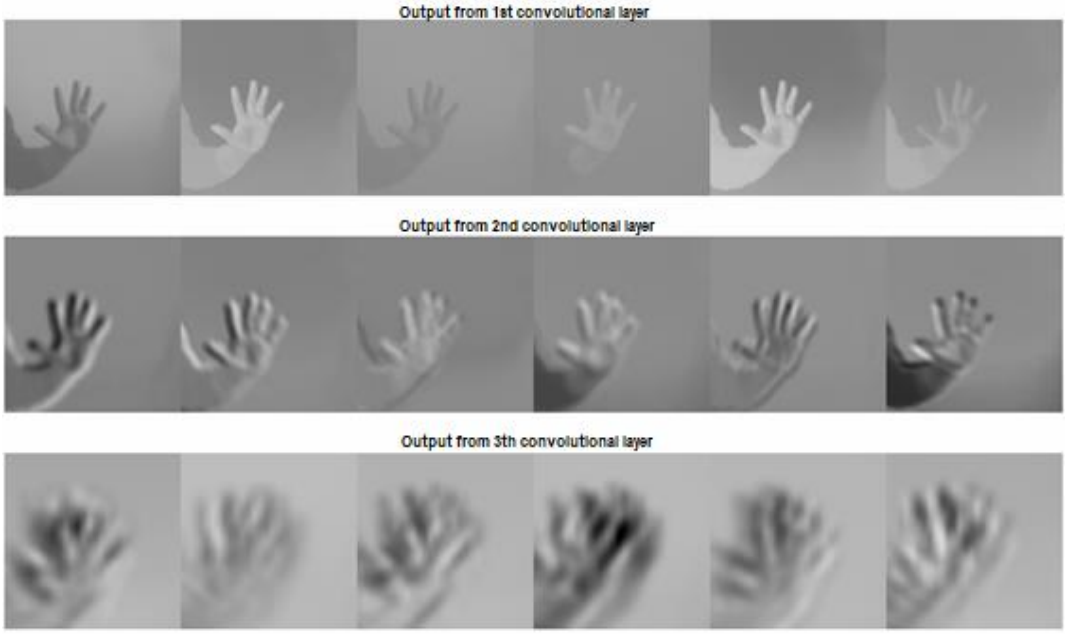


Fig.5. 6 examples of learned features by convolutional layers
of trained CNN with architecture A-2.

4 Simulation-based Model Control Using Static Hand Gesture

4.1 Example of Hand Gesture Recognition Using Kinect SDK

The Kinect functions are accessed using the From Video Device block in the Simulink graphical simulation environment. This block allows you to obtain images from an RGB or depth camera along with the tracking metadata. For the hand gesture recognition, we used the metadata HandRightState property, which identifies a recognized right-hand gesture. In (Fig.6) are displayed hand gestures, which can be recognized by Kinect SDK. In (Fig.7) is shown simulation scheme for PID control loop, in which setpoint is changed based on a right-hand gesture recognized by the Kinect sensor [6].

Fig.6. Standard hand gestures of Kinect SDK.

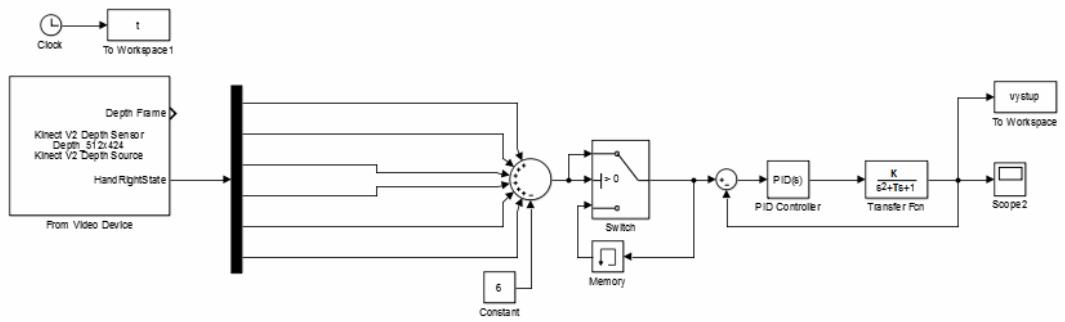


Fig.7. Simulation scheme for a PID control loop
(setpoint is changed based on a right-hand gesture recognized by the Kinect sensor)

4.2 Example of Hand Gesture Recognition Using CNN

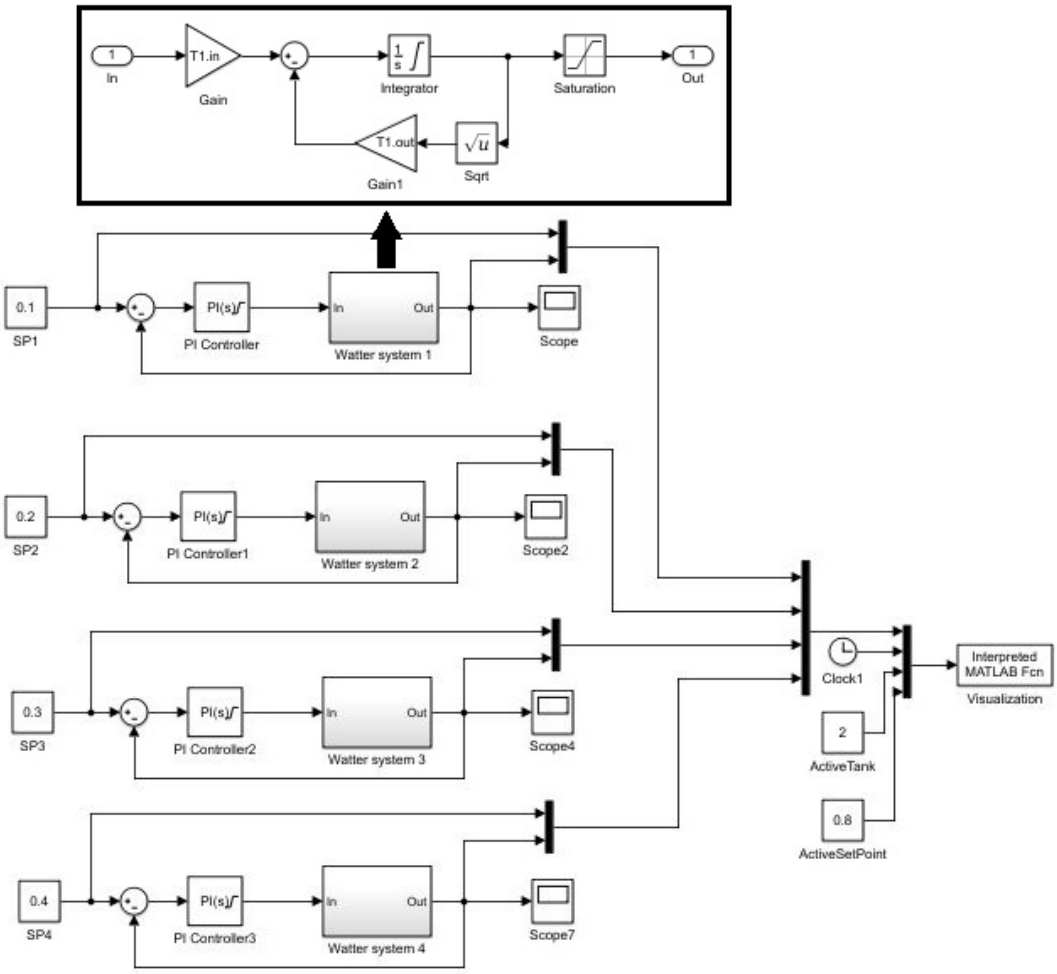


Fig.8. Simulation scheme of water tanks system with PI controllers.

A trained CNN with A-2 architecture was used to control the simulation model. To demonstrate hand gesture recognition, a simulation model of four independent water tanks with PI controllers was created [6]. We designed the entire control system scheme according to the Model-View-Controller (MVC) software architecture. MVC divided the program into 3 independent units (Simulation Model, Visualization, Controller) with minimal interconnections.

The simulation model is represented by a simulation scheme of four independent water tanks (Fig.8). In the simulation scheme are four independent control loops with PI controllers for level control in tanks. Inputs to the simulation model are the desired level values (SetPoints) in the tanks.

For a better presentation of the results achieved and the current state of simulation, a simple visualization was created (Fig.9). Simulation model is connected to the water tank visualization (Fig.8), to which it sends the desired level (SP) and current level values in the individual tanks. Additionally, active tank number (ActiveTank), new desired value (ActiveSetPoint) and simulation time are sent to the visualization subroutine.

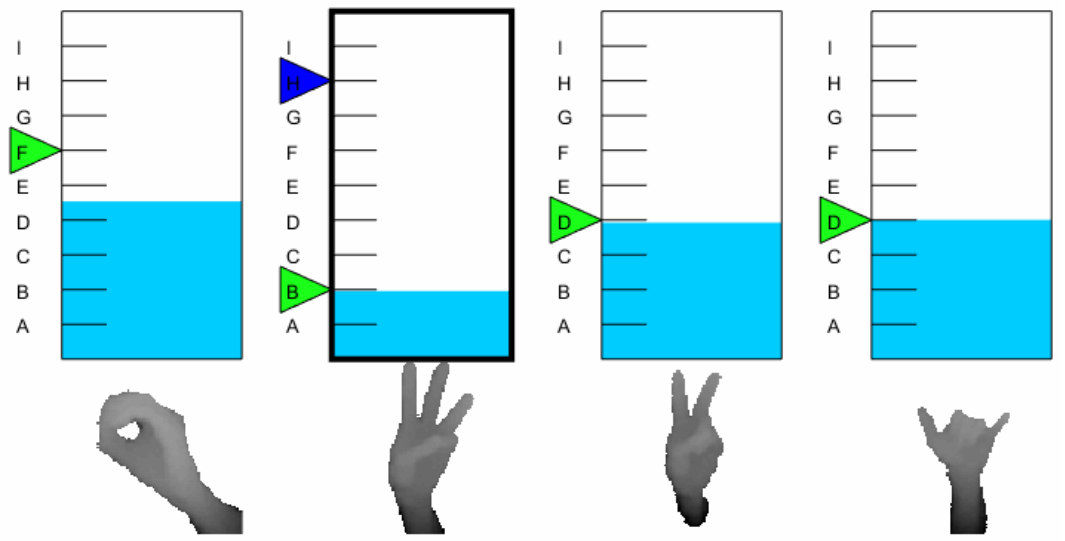


Fig.9. Visualization of the water tanks.

The last part of the MVC is a controller, that is linked to all MVC, Kinect sensor, and the neural network. Controller is running independently from the simulation. In the cycle, images and metadata from the Kinect sensor are collected and the simulation-based model is controlled. Both hands are used in the control process. The left hand is used to control parameter changes. Actions such as start and stop of right hand recognition, confirmation, and cancellation of set points are controlled by lefthand gesture. The right-hand gesture controls the desired level value of the selected tank. In the first phase of the simulation, the gesture symbolizing a particular tank is expected (Fig.9). When the gesture is recognized, the algorithm comes into the second phase - adjusting the desired level on the selected tank. Setpoint value can by controlled by ASL alphabet gestures from A to I.

The main subject of the final simulation test was to verify accuracy of the gesture recognition. We have created a test script with 14 gestures. 3 people participated on the testing phase, and everyone repeated the test 3 times. The average gesture recognition accuracy was 92.86%.

5 Conclusion

Various architectures of CNN were tested in this paper. The CNN classification models showed a very good classification accuracy. The proposed CNN gesture recognition system has been implemented in the simulation scheme due to the need of setting different model parameters. After successful testing of different CNN architectures, we verified the suitability of their use in the static hand gesture recognition task such as simulation-based control.

Acknowledgement

The research described in this paper was done within the project No. 1/0202/23 of the Slovak Grant Agency VEGA.

References

[1] P. K. Pisharady, M. Saerbeck. Recent methods and databases in vision-based hand gesture recognition: A review. Computer Vision and Image Understanding, 2015, 141: 152-165.

[2] A. Tang, K. Lu, Y. Wang, J. Huang and H. Li, A realtime hand posture recognition system using deep neural networks., ACM Transactions on Intelligent Systems and Technology (TIST), 2015, 6(2):21

[3] M. Van den Bergh and L. Van Gool. Combining rgb and tof cameras for real-time 3d hand gesture interaction. Applications of Computer Vision (WACV), 2011 IEEE Workshop on. IEEE, 6–72.

[4] M. Fagiani, E. Principi, S. Squartini, and F. Piazza. A new system for automatic recognition of italian sign language. Neural Nets and Surroundings, 2013, 69–79.

[5] R. Y. Wang and J. Popović. Real-time hand-tracking with a color glove, ACM transactions on graphics (TOG), 2009, vol. 28, 63

[6] Z. Ren, J. Yuan, and Z. Zhang. Robust hand gesture recognition based on fingerearth mover's distance with a commodity depth camera. Proceedings of the 19th ACM international conference on Multimedia, 2011, ACM:1093–1096

[7] S. Kajan, D. Pernecký, A. Hamad. Hand gesture recognition using multilayer perceptron network. 23th Annual Conference Proceedings, Technical Computing Prague, 2015

[8] G. Strezoski, D. Stojanovski, I. Dimitrovski, and G. Madjarov. Hand gesture recognition using deep convolutional neural networks.

[9] P. Molchanov, S. Gupta, K. Kim, and J. Kautz. Hand gesture recognition with 3D convolutional neural networks, 2015, Proceedings of the IEEE conference on computer vision and pattern recognition workshops, 1–7.

[10] P. Barros, S. Magg, C. Weber, and S. Wermter. A multichannel convolutional neural network for hand posture recognition. 2014, International Conference on Artificial Neural Networks, 403–410.

[11] F. Špaldoň. The control of simulation models using Kinect sensor. Bachelor thesis FEI STU in Bratislava, 2017, (in Slovak language)

[12] J. Goga, F. Špaldoň, S. Kajan, J. Pavlovičová, and M. Oravec. Static hand gesture database of FEI STU Bratislava. http://www.uim.elf.stuba. sk/kaivt/MLgroup, 2017.

[13] M. Beale, M. Hagan, H .Demuth. Neural Network Toolbox, User's Guide, 2017

Authors

Ing. Slavomír Kajan, PhD.
Institute of Robotics and Cybernetics,
Faculty of Electrical Engineering and Information Technology,
Slovak University of Technology in Bratislava, Slovakia
slavomir.kajan@stuba.sk
His research interests artificial intelligence, neural networks and applications of AI methods in medicine, control systems and robotics.

Ing. Jozef Goga, PhD.
Faculty of Electrical Engineering and Information Technology,
Slovak University of Technology in Bratislava, Slovakia
jozef.goga@stuba.sk
Currently at the Institute of Robotics and Cybernetics at the Faculty of Electrical Engineering and Information Technology of STU in Bratislava. His research interests include artificial neural networks, computer vision and biometrics.

International Journal
Information Technology Applications (ITA)

Volume 11, Number 2, December 2022

DETECTION OF PARKINSON'S DISEASE WITH MACHINE LEARNING SUPPORT

Zuzana Képešiová, Štefan Kozák, Eugen Ružický, Alfréd Zimmermann, Richard Malaschitz

Abstract:

Parkinson's disease (PD) is a neurodegenerative disorder causing partial or complete loss of motor reflexes and speech. It affects the patients significantly in their daily life. The accurate diagnosis of PD is quite complex, requires a lot of resources, and equipment and is time-consuming for diagnosis, especially in its initial stage's occurrence of the disease. To help the medical experts and researchers diagnose PD, we developed a machine learning approach based on the simple descriptive tasks captured in audio on the smartphone. The dataset for demonstrations consists of over 1500 patients with approximately 9% of patients diagnosed with PD. Hence the imbalance of the dataset the evaluation metrics such as Mathews correlation coefficient (MCC), sensitivity - specificity, and ROC curve were picked to describe the selected machine learning algorithms performance. The analysis of tested results in a single approach resulting with 73.49% (52.53% - 91.41%) MCC.

Keywords:

Machine learning, medical voice recordings, Parkinson disease, early detection, classification.

ACM Computing Classification System:

Machine learning.

1 Introduction

Parkison's disease (PD) is the second most common neurological disease, right after Alzheimer disease. At present, there is no cure nor a known root for PD, only a treatment, that can ease the symptoms and increase the quality of life of those affected. The symptoms of PD might demonstrate as problems with movement when the person is slow or even sometimes seems to be still. A well-known sign is rigidity (especially wrist, shoulder, and neck), and imbalance of neurotransmitters, dopamine, and acetylcholine. PD is mostly a movement disorder, but other weakenings also commonly develop, including psychiatric problems such as depression and dementia. Other noticeable sign of PD is the quality of speech.

The association of speech disorders with Parkinson's disease has been verified in various studies [1] [2]. Other studies have shown the progression of the disease is linked to a decrease in voice performance over time [3]. Therefore, the speech samples are ideal for the detection of the PD, while the data can be easily collected. Studies on the PD are usually focused on voice problems. Such practices also assist in the early diagnosis of the disease. People with Parkinsonism have vocal disorders affecting their speech in areas like volume level, the correct pronunciation of syllables, etc [4].

A Parkinson disease detection in machine learning related studies is being detected from various data sources such as handwriting database [5], dataset consisting of local field potential (LFP) recordings [6], gait features [7], single-photon emission computed tomography (SPECT) images [8], vocal recordings converted into images [9], and speech recordings converted into features [10]. In the previous study [11] we have shown a first comparative analysis of the machine learning algorithms for the early detection of PD, while in this paper we focus on a more specific treats between PD and speech symptoms with enlarged dataset focused on more specific tasks.

The paper is organized as follows. Section 2 describes the obtained dataset and its features, and introduces machine learning methods for classification, scaling methods of data, sampling methods for imbalanced datasets, and feature selection methods. Section 3 shows experimental results for the comparative analysis of proposed machine learning algorithms in a combination of sampling methods, scaling methods, and selection of features.

2 Machine Learning Methods

This section navigates through technical information about the analysis and the developed mathematical model. It starts with the dataset and its description, continues with data scaling methods, hence the imbalance of the dataset it continues with the sampling methods and finishes with the proposed machine learning algorithms to be analyzed.

A Dataset and Features

The collected dataset is containing over 1500 recorded samples. The data is consisting of 2 main data sources: basic information about the patient such as age, sex, education, etc; speech recordings statistics such as number of gaps, length of the gaps between words, number of used words, phonetism, etc; resulting in over 66 000 various features. The features are being computed from vocal recordings of subjects describing the seen images shown by the examiner. There are 2 types of images: small (62 images) and big (5 images). While small images depict one object or action, the big images consist of sceneries, which require more attention to detail. The dataset containes subjects falling into one or more of following groups:

- No neurodegenerative disorder
- Mild Cognitive Impairment (MCI)
- Alzheimer disease
- Parkinson disease
- Other neurodegenerative disorder

The dataset was cleaned up of low-quality recordings to ensure the transformation algorithms to extract the features correctly. Later, the subjects with diagnosed MCI and / or Alzheimer disease were filtered out of the dataset for early detection of Parkinson disease experiments. The dataset was filtered not only horizontally, but also vertically by selecting only 40 small images and 2 big images for the experiments.

B Scaling Methods

Many machine learning algorithms require scaling the input data to converge to the optimum as fast and smooth as possible. During the experimentation we used several different scaling methods.

Standard scaler, also known as z-score, standardize features by removing the mean and scaling to unit variance.

Min-max scaler transforms given values to fit into a given range set up to <0,1>.

Max-abs scaler changes each feature by its maximum absolute value resulting in values set up in a maximum range of <-1,1>.

Robust scaler removes the median and scales the data according to the quantile range, a measure of statistical dispersion

A Quantile transformer maps a variable's probability distribution to another probability distribution. The quantile function ranks or smooths out the relationship between observations and can be mapped onto other distributions, such as the uniform or normal distribution [12].

Yeo–Johnson transformer applies a transformation to stabilize variance and to make data more normal distribution-like, more Gaussian-like. Yeo–Johnson transformation as a power transformer can optimize the negative values in contradiction to the Box-Cox transformation method [13].

L2 normalizer, also known as Euclidean normalization of the group of L^p function spaces, makes the sum of the squares always be up to 1 normalizing the values.

C Sampling Techniques

The dataset consists of only 9% of positive cases for PD, which makes it unbalanced and makes it harder for the machine learning algorithms to learn efficiently, as there are not many samples with the class describing the PD. Some of the available methods dealing with such problem is to artificially create the underclassed samples or in contradiction, cut numbers of overclassed samples. In the paper, we propose four techniques, which may help the algorithms to find the links needed between the features [14].

Random oversampling is a method, where the minority class is evened out by enlarging the dataset with random copies of the minority class to equal the number of minority class compared to the majority class.

Synthetic Minority Over-sampling Technique (SMOTE) as the over-sampling method is a kind of data augmentation, where the minority class samples are not only randomly copied but are synthetically created. SMOTE picks samples close in the feature space from the minority class based on K-Neighbors and artificially creates a new sample with the feature values lying between the neighbors [15].

Adaptive Synthetic algorithm (ADASYN) oversample the dataset in a similar way as SMOTE as it is its extension. In contradiction to SMOTE, ADASYN generates synthetic samples inversely proportional to the density of the samples in the underclass area [16].

TomekLinks is in contradiction to an undersampling technique, where the majority class lowers the number of samples by deletion. TomekLinks uses the modified Condensed Nearest Neighbor method to choose which samples to delete. This modification creates so-called Tomek-link pair of the samples with different output classes and together they have the smallest Euclidean distance to each other in the feature space. This small distance creates close neighbors which implies high noise samples or very close samples to the minority class and therefore they are therefore removed [17].

D Machine Learning Algorithms

For the experiments we propose the comparison of the following machine learning algorithms: Extremely randomized trees, Random Forest, Linear regression, Logistic regression, Ridge regression, Passive Aggressive, Support vector classifier (SVC), Stochastic gradient descent (SGD), Light gradient-boosting machine (LGBM), Perceptron, Linear discriminant analysis (LDA), and k-nearest neighbors (KNN).

Extremely randomized trees propose an ensemble method of randomizing strongly both attribute and cut-point choice while splitting a tree node. The algorithm fits randomized decision trees on various batches of the dataset and utilizes averaging to improve the outcome accuracy [18].

The next tree-based method is called Random Forest. It utilizes the ensembling of decision trees method. In a classification task, the result of a random forest depends on most of the predicted class from all the trees of the forest [19].

The second class of machine learning algorithms is well known as linear algorithms. Linear algorithms predict the outcome of the data by a linear combination of the features and compute the weight of each feature to what extent it affects the predicted outcome. The best known is Linear regression, which fits the linear model with weights to minimize the residual sum of squares between the samples from the dataset and the predicted outcomes. Linear regression predicts the outcome value, which may lay outside of range <0,1> and therefore the input must be scaled properly.

Logistic regression is estimating a probability of the outcome and therefore is used for the classification problem more frequently. The logistic regression sums the weighted input and passes it through the sigmoid activation function resulting in a value in the range of <0,1>.

A Ridge regression extends the typical linear regression augmented by a penalty on the size of the coefficients resulting in a machine learning algorithm used for solving a problem of highly correlated independent variables.

For large-scale learning, there is a subclass of linear algorithms called Passive Aggressive. As the name of the subclass denotes, it is composed of two main parts: passive – if the prediction is correct, keep the model as it is; aggressive – on the other hand, if the prediction is not correct, then update the model. Passive Aggressive models use a regularization parameter to penalize the model in a case of an incorrect prediction [20]. In this comparative analysis, we chose a Passive Aggressive model utilizing a hinge loss function.

A support vector machine (SVM) is a machine learning method used for a large variety of tasks, which includes a classification problem ending in the term Support vector classifier (SVC). SVM maps the training samples to points in the space in order to maximize the gaps between given categories. The samples to be predicted are later mapped into the same space and depending on which side of the gap they fall the gap they belong to.

Gradient descent is an optimization method, where the main technique is a performance of minimizing cost function $J(\theta)$ with the parameter θ updating parameters in the reverse direction of the gradient of the cost function $\nabla\_\theta\ J(\theta)$ with respect to the parameters [21]. In the study, we used an SVM with Stochastic gradient descent (SGD).

Light gradient-boosting machine (LGBM) is based on decision tree algorithms. LGBM implements a highly optimized histogram-based decision tree learning algorithm. The LightGBM algorithm utilizes two novel techniques called Gradient-Based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) which allow the algorithm to run faster while maintaining a high level of accuracy [22].

Perceptron is an algorithm used as a classifier. In a binary classification problem, it maps the input vector onto a single binary output value. The output value depends on the sum of weights multiplied with the input in a dot product manner regularized with the bias factor [21].

A Linear discriminant analysis (LDA) tries to find a projection vector that can enlarge the distance of samples from different classes and reduce the distance of samples from the same class [23]. It may be used not only for a classification task but also for a dimensional reduction with the use of covariance parameters.

A possibility how classifying the samples is to place them into one of the defined groups. The groups might be created by a clustering algorithm known as the k-nearest neighbors (KNN) algorithm. In a contradiction to the previous algorithms, a KNN is a non-parametric supervised learning method, wherein in a classification task the goal is to assign the provided sample to the class based on the votes of its n closest neighbors and their class reference [24].

3 Experiments and Results

A Evaluation Metrics

Due to the high imbalance of the used dataset conventional evaluation metrics would not describe the efficiency of the trained model accurately. Therefore, we utilized the following evaluation metrics: Matthew's correlation coefficient (MCC), precision, and recall. An MCC is based on a confusion matrix and can be described as accuracy for the imbalanced dataset. The specificity metric defines how many negative samples were predicted correctly, while the sensitivity metric describes how many positive samples have been predicted correctly. The evaluation metrics are defined by the following formulas:

$$\mathrm{MCC} = \frac{tp \times tn - fp \times fn}{\sqrt{(tp+fp)(tp+fn)(tn+fp)(tn+fn)}}, \quad (1)$$

$$\mathrm{Specificity} = \frac{tn}{tn+fp}, \quad (2)$$

$$\mathrm{Sensitivity} = \frac{tp}{tp+\ fn}, \quad (3)$$

tn denotes true negatives, tp are true positives, fn are false negatives, fp are false positives [25]

B Experimental Results

The provided dataset of medical voice recordings transformed into the statistics combined with the basic information about patient with imbalanced data distribution over the Parkinson's disease categorization with 9% of positive and 91% of negative cases was divided into training and validation subsets in ratio of 80% training - 20% validation data 5 times in total utilizing stratified cross validation. The data split via stratified cross-validation ensures all samples to be included in validation split and provides better understanding of the absolute results of the algorithms. The data were divided with age and neurodegenerative disorder in mind.

(Tab.1) is providing a table of the 20 best results for the combination of the scaling method, the machine learning algorithm, and the sampling method. The results are sorted by the total MCC from the highest to lowest. The total values of the provided metrics are computed based on the prediction of the samples, while the samples were in the validation fold. The ranges of the metrics are computed across the cross-validation folds.

Regardless of the sampling method, the best results yields Linear Regression scaled by Quantile Transformer. For this Algorithm, the best sampling method is ADASYN. Other valuable algorithms used for successful early detection of Parkison's disease are Ridge, and LGBM regardless of the sampling method, and SVM, LDA and Logistic Regression in combination with None or TomekLinks sampling method and in use of Quantile Transformer. Overall, Quantile Transformer has proven to be the best scaling method for this task, especially in a combination with Linear Regression.

Table 1. comparative analysis results of selected best 20 settings.

Scaling Method	ML Algorithm	Sampling Method	Accuracy	MCC	Sensitivity	Specificity
Quantile Transformer	Linear Regression	ADASYN	0.9434 (0.913-0.9826)	0.7349 (0.5253-0.9141)	0.7753 (0.4285-0.8928)	0.9664 (0.9257-0.995)
Quantile Transformer	Linear Regression	None	0.9452 (0.913-0.9782)	0.7336 (0.5253-0.8969)	0.7391 (0.4285-0.9259)	0.9733 (0.9257-0.995)
Quantile Transformer	Linear Regression	Random Over Sampler	0.9434 (0.913-0.9739)	0.7298 (0.5253-0.8741)	0.7536 (0.4285-0.8928)	0.9693 (0.9356-0.9852)
Quantile Transformer	Linear Regression	TomekLinks	0.9426 (0.9043-0.9782)	0.7265 (0.5161-0.8969)	0.7536 (0.4642-0.9259)	0.9683 (0.9207-1.0)
Quantile Transformer	Linear Regression	SMOTE	0.9408 (0.9043-0.9695)	0.7253 (0.5253-0.8614)	0.7753 (0.4285-0.9285)	0.9634 (0.9158-0.9852)
Quantile Transformer	Ridge	TomekLinks	0.9391 (0.9043-0.9608)	0.6909 (0.4856-0.7989)	0.6521 (0.4285-0.8571)	0.9782 (0.9603-1.0)
Quantile Transformer	LGBM Regressor	ADASYN	0.94 (0.9173-0.9739)	0.6841 (0.5478-0.8691)	0.5724 (0.4285-0.7777)	0.9901 (0.9801-1.0)
Quantile Transformer	SVM	TomekLinks	0.9373 (0.8956-0.9608)	0.6833 (0.4208-0.8047)	0.6521 (0.3571-0.9285)	0.9762 (0.9554-0.995)
Quantile Transformer	Linear Discriminant Analysis	None	0.9382 (0.9-0.9652)	0.6806 (0.4538-0.8228)	0.6159 (0.3928-0.8571)	0.9822 (0.9653-1.0)
MaxAbs Scaler	LGBM Regressor	SMOTE	0.9382 (0.9173-0.9565)	0.6795 (0.5744-0.7798)	0.6086 (0.4814-0.8214)	0.9832 (0.9702-1.0)
Quantile Transformer	LGBM Regressor	TomekLinks	0.9382 (0.9043-0.9608)	0.6748 (0.4395-0.8066)	0.5724 (0.2142-0.75)	0.9881 (0.9653-1.0)
MaxAbs Scaler	LGBM Classifier	TomekLinks	0.9382 (0.9086-0.9565)	0.6725 (0.4846-0.7746)	0.5434 (0.3571-0.6428)	0.992 (0.9801-1.0)
MaxAbs Scaler	LGBM Regressor	ADASYN	0.9365 (0.9043-0.9521)	0.6725 (0.4613-0.7798)	0.6159 (0.3571-0.8214)	0.9802 (0.9702-0.9901)
Standard Scaler	LGBM Regressor	ADASYN	0.9365 (0.9086-0.9608)	0.6725 (0.5046-0.8066)	0.6159 (0.4285-0.7777)	0.9802 (0.9554-1.0)
Quantile Transformer	Linear Discriminant Analysis	TomekLinks	0.9356 (0.9-0.9608)	0.6714 (0.4538-0.7982)	0.6304 (0.3928-0.8571)	0.9772 (0.9554-0.995)
Quantile Transformer	Logistic Regression	TomekLinks	0.9356 (0.8956-0.9608)	0.6686 (0.4208-0.7989)	0.6159 (0.3571-0.8571)	0.9792 (0.9653-1.0)
Quantile Transformer	Ridge	ADASYN	0.9365 (0.9-0.9608)	0.6678 (0.4401-0.7989)	0.5869 (0.3571-0.7857)	0.9841 (0.9702-1.0)
Quantile Transformer	Ridge	None	0.9365 (0.9-0.9608)	0.6678 (0.4401-0.7989)	0.5869 (0.3571-0.7857)	0.9841 (0.9702-1.0)
Quantile Transformer	Ridge	Random Over Sampler	0.9365 (0.9-0.9608)	0.6678 (0.4401-0.7989)	0.5869 (0.3571-0.7857)	0.9841 (0.9702-1.0)
Quantile Transformer	Ridge	SMOTE	0.9365 (0.9-0.9608)	0.6678 (0.4401-0.7989)	0.5869 (0.3571-0.7857)	0.9841 (0.9702-1.0)

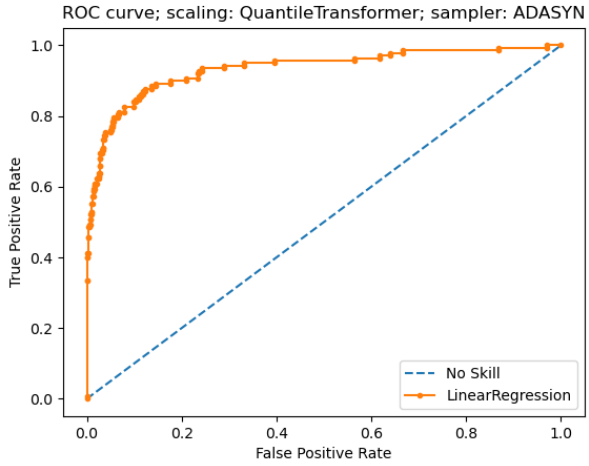


Fig.1. ROC curve of a combination of Quantile Transformer sampling method, Linear Regression machine learning. algorithm and ADASYN sampling method.

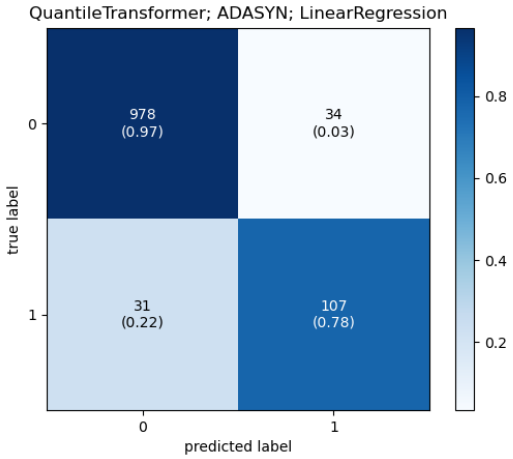


Fig.2. Confusion matrix of the selected model, linear regression with Quantile transformed data with ADASYN sampling method used.

(Fig.1) shows ROC curve of a combination of Quantile Transformer sampling method, Linear Regression machine learning algorithm and ADASYN sampling method. ROC curve shows a great decision skill of the selected algorithm when keeping a high true positive rate while keeping low false positive rate.

(Fig.2) shows a confusion matrix of the selected combination of Quantile Transformer sampling method, Linear Regression machine learning algorithm and ADASYN sampling method. As it is depicted on the cinfusion matrix, we can see the algorithm is very succcesfull determine the healthy patient, while keeping the high performance detecting a patient with PD.

4 Conclusion

In this paper, performance evaluation of machine learning algorithms in a combination with sampling and scaling methods was proposed and verified done. The aim of analysis the performance evaluation is to find the best modified algorithm available to solve the complex problem, and in this study for early detection of Parkinson's disease based on voice recordings of patients describing the images. These patients were describing the provided images divided into two groups: small and big images, differentiating on a scale of the descriptive manner. These voice recordings showing the light mental exercises were preprocessed and statistically described in a combination with basic information about a patient available. Out of all provided machine learning algorithms, and scaling and sampling methods the best performing model was chosen as a linear regression with a threshold with a Quantile transformed input data and with ADASYN sampling method resulting in 94.34% (91.30%-98.26%) accuracy, 73.49% (52.53%-91.41%) MCC, 77.53% (42.85%-89.28%) sensitivity, and 96.64% (92.57%-99.50%) specificity being able to successfully predict the Parkinson's disease in most of the positive subjects and with suspicion for healthy subjects to be monitored and followed up with a doctor. The proposed approach and efficient algorithmic processing can be a suitable and effective means for early diagnosis of neurodegenerative diseases.

Acknowledgement

This paper was supported by Early Warning of Alzheimer and other neurodegenerative diseases from Operational Programme Integrated Infrastructure (EU Cohesion Fund) - Code n. ITMS2014+/313022V631 and by Research of Advanced Algorithms and Modeling of Processes in Domains of Applied Informatics from Grant Agency Academia Aurea (GAAA) - Code n. GAAA/2022/1.

References

[1] J. Gamboa, F. J. Jiménez-Jiménez, A. Nieto, J. Montojo, M. Ortí-Pareja, J. A. Molina, E. García-Albea and I. Cobeta, "Acoustic voice analysis in patients with Parkinson's disease treated with dopaminergic drugs," *J Voice*, pp. 314-320, September 1997.

[2] A. K. Ho, J. L. Bradshaw and R. Iansek, "For better or worse: The effect of levodopa on speech in Parkinson's disease," *Mov Disord.*, vol. 23, no. 4, pp. 574-580, 15 March 2008.

[3] B. Harel, M. Cannizzaro and P. J. Snyder, "Variability in fundamental frequency during speech in prodromal and incipient Parkinson's disease: a longitudinal case study," *Brain Cogn.*, vol. 56, no. 1, pp. 24-29, October 2004.

[4] B. Sakar , M. Isenkul, C. Sakar, A. Sertbas, F. Gurgen, S. Delil, H. Apaydin and O. Kursun, "Collection and analysis of a Parkinson speech dataset with multiple types of sound recordings" *IEEE J Biomed Health Inform*, vol. 17, no. 4, pp. 828-834, 2013.

[5] P. Drotár, J. Mekyska, I. Rektorová, L. Masarová, Z. Smékal and M. Faundez-Zanuy, "Evaluation of handwriting kinematics and pressure for differential diagnosis of Parkinson's disease" *Artificial Intelligence in Medicine*, vol. 67, pp. 39-46, 2016.

[6] A. T. Connolly, W. F. Kaemmerer, S. Dani, S. R. Stanslaski, E. Panken, M. D. Johnson and T. Denison, "Guiding deep brain stimulation contact selection using local field potentials sensed by a chronically implanted device in Parkinson's disease patients" in *2015 7th International IEEE/EMBS Conference on Neural Engineering (NER)*, 2015.

[7] F. Wahid, R. K. Begg, C. J. Hass, S. Halgamuge and D. C. Ackland, "Classification of Parkinson's Disease Gait Using Spatial-Temporal Gait Features" *IEEE Journal of Biomedical and Health Informatics,* vol. 19, no. 6, pp. 1794-1802, 2015.

[8] T. Pianpanit, S. Lolak, P. Sawangjai, T. Sudhawiyangkul and T. Wilaiprasitporn, "Parkinson's Disease Recognition Using SPECT Image and Interpretable AI: A Tutorial" *IEEE Sensors Journal,* vol. 21, no. 20, pp. 22304 - 22316, 2021.

[9] M. Wodzinski, A. Skalski, D. Hemmerling, J. R. Orozco-Arroyave and E. Nöth, "Deep Learning Approach to Parkinson's Disease Detection Using Voice Recordings and Convolutional Neural Network Dedicated to Image Classification," in *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC),* 2019.

[10] J. S. Almeida, P. P. Rebouças Filho, T. Carneiro, W. Wei, R. Damaševičius, R. Maskeliūnas and V. H. C. de Albuquerque, "Detecting Parkinson's disease with sustained phonation and speech signals using machine learning techniques," *Pattern Recognition Letters,* vol. 125, pp. 55-62, 2019.

[11] Z. Képešiová, Š. Kozák, E. Ružický, A. Zimmermann and R. Malaschitz, "Comparative Analysis of Advanced Machine Learning Algorithms for Early Detection of Parkinson Disease," in *2022 Cybernetics & Informatics (K&I)*, Visegrád, Hungary, 2022.

[12] B. M. Bolstad, R. A. Irizarry, M. Åstrand and T. P. Speed, "A comparison of normalization methods for high density oligonucleotide array data based on variance and bias," *Bioinformatics,* vol. 19, no. 2, pp. 185-193, January 2003.

[13] I.-K. Yeo and R. A. Johnson, "A new family of power transformations to improve normality or symmetry" *Biometrika,* vol. 87, no. 4, pp. 954-959, December 2000.

[14] "Training and assessing classification rules with imbalanced data" *Data Mining and Knowledge Discovery,* vol. 28, pp. 92-122, 2014.

[15] N. V. Chawla, K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Oversampling Technique," *Journal of Artificial Intelligence Research,* vol. 16, pp. 321-357, 2002.

[16] H. He, Y. Bai, E. A. Garcia and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *IEEE International Joint Conference on Neural Networks,* 2008.

[17] D. Devi, S. k. Biswas and B. Purkayastha, "Redundancy-driven modified Tomek-link based undersampling: A solution to class imbalance," *Pattern Recognition Letters,* vol. 93, pp. 3-12, 2017.

[18] P. Geurts, D. Ernst and L. Wehenkel, "Extremely randomized trees" *Machine Learning,* vol. 63, pp. 3-42, 2006.

[19] K. Polat, "A Hybrid Approach to Parkinson Disease Classification Using Speech Signal: The Combination of SMOTE and Random Forests," in *Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT),* 2019.

[20] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz and Y. Singer, "Online Passive-Aggressive Algorithms," *Journal of Machine Learning Research,* vol. 7, pp. 551-585, 2006.

[21] Z. Képešiová, J. Cigánek and Š. Kozák, "Driver Drowsiness Detection Using Convolutional Neural Networks," in *2020 Cybernetics & Informatics (K&I)*, 2020.

[22] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye and T.-Y. Liu, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in *31st Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, USA, 2017.

[23] J. Wen, X. Fang, J. Cui, L. Fei, K. Yan, Y. Chen and Y. Xu, "Robust sparse linear discriminant analysis," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 29, no. 2, pp. 390-403, 2018.

[24] A. R. Lubis, M. Lubis and Al-Khowarizmi, "Optimization of distance formula in K-Nearest Neighbor method," *Bulletin of Electrical Engineering and Informatics,* vol. 9, no. 1, pp. 326-338, 2020.

[25] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics,* vol. 21, 2020.

Authors

Ing. Zuzana Képešiová, PhD.
Sygic, Ltd.
zuzana.kepesiova@gmail.com
research and applications worker Sygic, Ltd. as machine learning specialist. Her professional focus is mostly research and programming in intelligent systems focusing on pattern recognition, optimization, big data analysis and prediction.

prof. Ing. Štefan Kozák, PhD.
Faculty of Informatics
Pan-European University in Bratislava, Slovakia
stefan.kozak@paneurouni.com
His research interests include system theory, linear and nonlinear control methods, numerical methods and software for modeling, control, signal processing, IoT, IIoT and embedded intelligent systems for digital factory in industry and medicine.

Assoc. Prof. RNDr. Eugen Ružický, PhD.
Faculty of Informatics
Pan-European University in Bratislava, Slovakia
eugen.ruzicky@paneurouni.com
His research interests include applied informatics, system analysis, modelling, visualisation and applications in medicine.

RNDr. Alfréd Zimmermann
AXON PRO Ltd. Bratislava, Slovakia
zimmermann@axonpro.sk
He is owner and CEO of the company. Professional and scientific interests include artificial intelligence in general and natural language processing.

Richard Malaschitz
AXON PRO Ltd. Bratislava, Slovakia
richard.malaschitz@axonpro.sk
Research worker at AXON PRO, data preprocessing for research projects.

List of Reviewers

Issue 2/2022, in alphabetic order

doc. Ing. Petr Drahoš, PhD. Slovak University of Technology in Bratislava, Slovakia
doc. Ing. Oto Haffner, PhD. Slovak University of Technology in Bratislava, Slovakia
prof. ing. Alena Kozáková, PhD. Slovak University of Technology in Bratislava, Slovakia
doc. Ing. Erik Kučera, PhD. Slovak University of Technology in Bratislava, Slovakia
prof. Danica Rosinová, PhD. Slovak University of Technology in Bratislava, Slovakia

ISSN: 2453-7497 (online)
ISSN 1338-6468