Heliyon 5 (2019) e02560

Contents lists available at ScienceDirect

Heliyon

journal homepage: www.heliyon.com

Research Article

A Python-based laboratory course for image and video signal processing on embedded systems

Karina Jaskolka *, Jürgen Seiler, Frank Beyer, André Kaup

Multimedia Communications and Signal Processing, Friedrich-Alexander-University Erlangen-Nürnberg, 91058 Erlangen, Germany

ARTICLE INFO

Keywords:
Image and video signal processing
Laboratory course
Python
Embedded system
Computer science
Education

ABSTRACT

The usage of embedded systems is omnipresent in our everyday life, e.g., in smartphones, tablets, or automotive devices. These devices are able to deal with challenging image processing tasks like real-time detection of faces or high dynamic range imaging. However, the size and computational power of an embedded system is a limiting demand. To help students understanding these challenges, a new lab course "Image and Video Signal Processing on Embedded Systems" has been developed and is presented in this paper. The Raspberry Pi 3 Model B and the open source programming language Python have been chosen, because of low hardware cost and free availability of the programming language. In this lab course the students learn handling both hard- and software, Python as an alternative to MATLAB, the image signal processing path, and how to develop an embedded image processing system, from the idea to implementation and debugging. At the beginning of the lab course an introduction to Python and the Raspberry Pi is given. After that, various experiments like the implementation of a corner detector and creation of a panorama image are prepared in the lab course. Students participating in the lab course develop a profound understanding of embedded image and video processing algorithms which is verified by comparing questionnaires at the beginning and the end of the lab course. Moreover, compared to a peer group attending an accompanying lecture with exercises, students having participated in this lab course outperform their peer group in the exam for the lecture by 0.5 on a five-point scale.

1. Introduction

Embedded systems are used in many different areas like medical engineering (Mastinu et al., 2017), aeronautics (Sharp et al., 2010), water and energy supply (Raghunathan et al., 2005; Stoianov et al., 2007), transportation (Bernini et al., 2014; Castellanos et al., 2011), automotive industry (Bhat et al., 2017), and information and communication technology (Rupniewski et al., 2016). Moreover, in everyday life embedded systems are omnipresent, e.g., in smartphones or tablets. These devices contain usually an integrated camera and deal with challenging tasks, like the calculation of high dynamic range images (Tsai et al., 2014) or the detection of faces in real-time (Mao et al., 2017). Although, the performance of the embedded systems is limited because of size and mobility, the systems are used in many applications for image and video signal processing. To show these challenges to the students, a lab course seems to be the best way because of practical and illustrative presentation (Hodson, 1993; Hofstein and Lunetta, 1982, 2004; Lunetta et al., 2007; Schwartz, 1959). Currently, the lecture "Image, Video, and Multidimensional Signal Processing" and the supplements to this lecture are offered at the Friedrich-Alexander-University (FAU) Erlangen-Nürnberg. In this lecture, the basic understanding in processing of image and multidimensional data like interpolation, feature detection, segmentation, and transformation are explained. In the supplements, these topics are enlarged by different blackboard and computer (MATLAB) exercises. However, a laboratory course for signal processing on embedded systems was missing in the course catalog of the FAU. So, the new lab course "Image and Video Signal Processing on Embedded Systems" has been developed. In this paper, this lab course is presented and the benefits for students are shown.

Different definitions of an embedded system exists, e.g., Wang describes a medium-scale embedded system as a microprocessor with programming tools (debugger, simulator, integrated development environment), which typically possesses an operating system (OS) (Wang, 2017). By the definition of Vahid and Givargis an embedded system must be cheap, fit on a single chip, process data in real time, and consume minimum power to prevent a cooling fan. In addition, embedded systems are present in several common electronic devices, such as digital cameras, calculators, home security systems, washing machines, printers,

* Corresponding author.
E-mail address: karina.jaskolka@FAU.de (K. Jaskolka).

https://doi.org/10.1016/j.heliyon.2019.e02560
Received 2 August 2018; Received in revised form 13 March 2019; Accepted 27 September 2019

product scanners, etc (Vahid and Givargis, 2002). All compared systems fulfill both definitions and are classified as embedded system in (Abbot, 2018).

Some requirements for the choice of the embedded system for the new lab course are given. The handling of the system should be possible with display, mouse, and keyboard, and a user-friendly OS with a graphical user interface (GUI) should also be used. The needed connection for the additional hardware should exist and a connection to a network should also be possible. For the usage of Python, a Python interpreter should be present. Finally, the embedded system should be inexpensive, but capable for the experiments.

Different embedded systems exist, but not every of those fulfills these requirements. The BeagleBone Black (BeagleBoard.org Foundation, 2018) provides a processor with 1 GHz and 512 MB RAM, a micro-HDMI, USB 2.0 ports and an own OS. The Raspberry Pi Foundation provides six different embedded systems (Raspberry Pi Foundation, 2018). The Raspberry Pi 1 Model A has been introduced in February 2012 as first embedded system and has been replaced by the successor Raspberry Pi 1 Model A+ in November 2014 (Raspberry Pi Foundation, 2018). The latest model is the Raspberry Pi 3 Model B. It is a cost-effective, up-to--date and widespread embedded system. The BeagleBone Black and the Raspberry Pi 3 Model B fulfill all the requirements of the lab course. However, the BeagleBone Black is twice as expensive as the Raspberry Pi and, in addition, the Raspberry Pi has a better CPU and RAM. Thus, the Raspberry Pi is used in the newly developed lab course "Image and Video Signal Processing on Embedded Systems". In addition, the Raspberry Pi is so inexpensive that the students can also use the device at home for further experiments.

This paper is organized as follows. In Section 2 the laboratory equipment with hardware and software is presented. In Section 3 the learning objectives and the laboratory properties are explained. In Section 4 the laboratory experiments are described. In Section 5 the assessment of the lab course is shown and the last section concludes this paper.

2. Instrumentation

At the beginning of the lab course, all groups receive a box with the required hardware components: mouse, keyboard, HDMI to DVI adapter cable, power supply, Raspberry Pi 3 Model B, USB camera, and micro SD card. In the first experiment and before every experiment all the components are connected together. The displays are already available in the laboratory room. The setup is very easy and will be performed by the students in Experiment I. By this, the students learn the practical set-up work with embedded systems directly.

First, the micro SD card with the pre-installed OS Raspbian Jessie (Raspberry Pi Foundation, 2018) is inserted in the SD card slot. After that, all the other hardware is connected by USB or HDMI with the Raspberry Pi as depicted in Fig. 1. The OS boots after connecting the power supply. The whole hardware is recognized instantly. The configuration of the system and the connection to the Wi-Fi is done in the first experiment, the camera is connected and tested in the fourth experiment. Updates are installed regularly by the students to keep the software up to date. We decided to use an USB camera instead of the Raspberry Pi camera module (Raspberry Pi Foundation, 2018), so the experiments can easily be transferred to other embedded systems. Furthermore, USB cameras are used in many industrial computer vision applications, e.g., Basler, an internationally leading manufacturer of high-quality cameras mostly uses USB as interface (Basler, 2019).

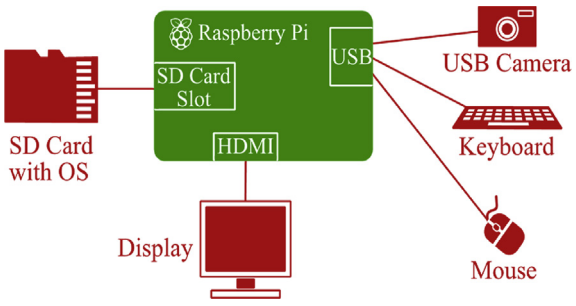


Fig. 1. Block diagram of the connected components of the workstation.

The students perform the whole setup on their own, so they learn how to handle sensitive hardware. They recognize, that an embedded system is a circuit board, which must be handled with care. Thus, such an embedded system should always be operated in its provided case.

2.1. Raspberry Pi 3 Model B

The Raspberry Pi 3 Model B is the third generation Raspberry Pi and is developed by the Raspberry Pi Foundation (Raspberry Pi Foundation, 2018). All necessary components like CPU, GPU etc. are placed on the Broadcom chip. The embedded system is equipped with following parts (Raspberry Pi Foundation, 2018):

- Quad Core 1.2 GHz Broadcom BCM2837 64 bit CPU with 1 GB RAM,
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board,
- 40 pin extended GPIO and status LEDs,
- 4 × USB 2.0 ports and 10/100 Mbit ethernet port,
- 4 pole stereo output and composite video port,
- CSI camera port for connecting a Raspberry Pi camera and DSI display port for connecting a Raspberry Pi touchscreen display,
- full size HDMI output and micro USB power input, and
- micro SD card port for loading the OS and storing data (underside).

However, the Raspberry Pi neither possess a non-volatile internal memory nor an interface for an internal harddisk. Moreover, there is no on-off switch. So, the power will be switched off when the power supply will be disconnected from the micro USB power input. Usually, the Raspberry Pi should be put into a case to protect the system.

2.2. Software

For the lab course, the officially supported operating system Raspbian Jessie in its newest version is used (Raspberry Pi Foundation, 2018). The OS is already installed on the micro SD card for the students because the installation takes a lot of time and is uninformative. The OS supports the whole hardware of the Raspberry Pi, and provides a desktop interface and some useful software. By putting the micro SD card into the card slot the OS will boot immediately after the power supply is plugged in. The used micro SD card can read up to 100 MB/s and write up to 90 MB/s. It is a Class 10 card and has a capacity of 32 GB. This card has been chosen because otherwise the update and installation process would take too long.

The experiments are prepared with the open source programming language Python (Python Software Foundation, 2018). The readability of a program has been the most important part during the development of Python. The language can easily be learned and supports the most common programming paradigms, e.g., object, aspect, and functional orientation. Python only needs a few keywords and the syntax is clear and easy to read. Python 2, Python 3, and the integrated development environment IDLE for Python are already installed on the OS.

Python was chosen because it is simple to read and to learn, it is open and free. Moreover, it consists of an extensive standard library, which aims at programming in general and contains specific modules for threading, networking, databases, etc. Furthermore, there are a lot of additional packages, e.g., for plotting and handling images, which are also used in the laboratory. As a student of a technical study program, Python serves as a good basis for future jobs, because many companies use Python as primary programming language (Cass, 2018).

Table 1
Overview of properties and costs of the lab course.

ECTS	2.5
Number of experiments	7
Length of experiment	Ca. 4 hours
Group size	2-3 students per group
Participating students	27
Costs for Raspberry Pi, power supply and case	44 Euro per device
Costs for additional hardware: USB camera, keyboard, mouse, micro SD card and HDMI-DVI cable	84 Euro per device
Costs for the server	1000 Euro
Total costs for lab course hardware for 30 students	2920 Euro

During the lab course various software and Python packages will be installed. A detailed description can be found in the individual experiments.

2.3. Server

For the lab course, a network connectivity is necessary for updating, downloading, and searching in the internet. Therefore, a server with a wireless network adapter is provided. The adapter sets up a Wi-Fi network, so the individual Raspberry Pis can establish a connection with the server and receive internet access. With FileZilla (Kosse, 2018), which is installed in the third experiment, the students can upload their generated code and results, and download provided data like the laboratory manual, a pre-compiled OpenCV library, and Python scripts. For Experiment IV to VII, parts of the Python source code are available on the server, so the preparation of the tasks is easier for the students. For updates, an internet access is necessary. Furthermore, the students should be enabled to search the internet for solving a problem or understanding details. This way, they learn how to solve a problem on their own by discussing and investigating together. The server is needed since the Raspberry Pi should not be connected directly to the university network, because of security reasons. Furthermore, the progress of the groups can be controlled and an adequate quality of service can be provided.

3. Design

The lab course is based on the successful completion of seven experiments. Therefore, 2.5 ECTS can be achieved, which consists of around four hours in the lab and around six hours for preparation and follow-up per experiment. Considering the requirements, teamwork and cooperation with the other students are strongly encouraged throughout the course. The students prepare the courses in groups of two in a weekly session. Some basic programming skills (e.g., MATLAB, C, etc.) and the participation in a basic lecture for signal processing are recommended for participating in the lab course. So, bachelor and master students can take part in the lab course. The following study paths take part at the lab course: Electrical, Electronics and Communication Engineering (EEI), Information and Communication Technology (IuK), Advanced Signal Processing and Communications Engineering (ASC), and Communications and Multimedia Engineering (CME). The students are normally at the end of their bachelor's degree or at the beginning of their master's degree. In the introduction part of the laboratory manual, all information

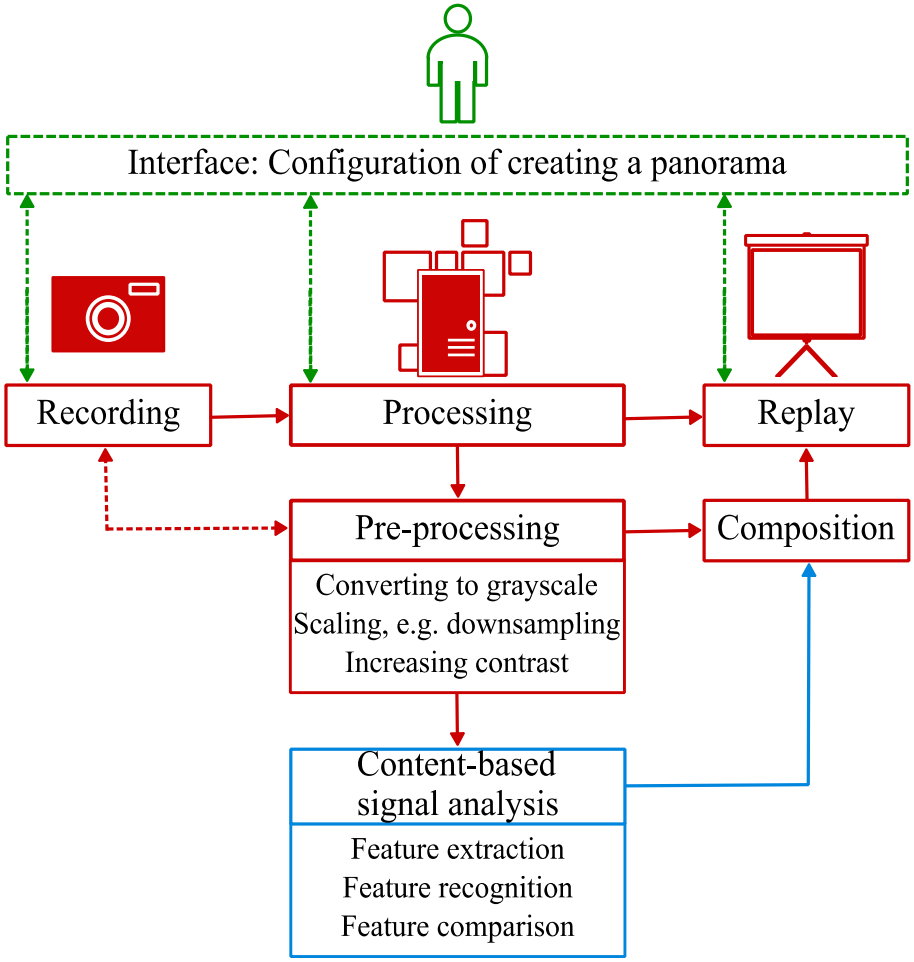


Fig. 2. Overview of the image signal process for creating a panorama.

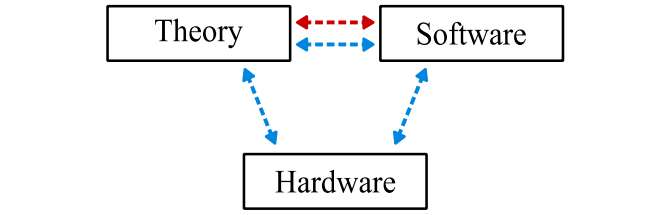


Fig. 3. Parts of a conventional lecture (theory and software) and a lab course (theory, software, and hardware).

for preparation and accomplishment of the lab course are summarized. Each student receives a printed manual at the beginning of the lab course. A digital copy of the laboratory manual is also available in the e-learning system of the university. Before every experiment, the students should study the manual at home to be prepared for the current experiment. Table 1 shows an overview of the properties and costs of the lab course. A Raspberry Pi 3 Model B, a power supply and a case for the Raspberry cost around 44 Euro. The additional hardware, like USB camera, keyboard, mouse, micro SD card and HDMI-DVI cable costs around 84 Euro. If some students want to use the Raspberry lab system at home, some of the additional hardware, like keyboard or mouse, does not have to be purchased. So, costs decrease for the use at home. For the laboratory, a server was purchased for around 1000 Euro.

The fact that no special requirements are necessary for the lab course affects the first experiments. The commissioning of the embedded system and the involvement in the laboratory network are the topics of the first experiment. In the second experiment, the fundamentals of Python are introduced. Furthermore, the typical path of image signal processing is passed in the following experiments. In Fig. 2 the red flowchart shows this typical path with recording, processing, composition and replaying. The processing part includes many options like converting to grayscale, scaling, increasing contrast, etc. In Fig. 3 the different parts of a conventional lecture and a lab course are depicted. While in a conventional lecture ("Image, Video, and Multidimensional Signal Processing") only the aspects theory and software are used, also the aspect hardware is considered in a lab course ("Image and Video Signal Processing on Embedded Systems"). So, these three aspects are weighted in the different experiments. The aspect software involves the independent implementation of the different algorithms or the updating process, hardware involves the independent usage of the different components like the camera, and theory involves the description and understanding of different methods and needed knowledge for the various tasks. The first experiment only contains the aspect hardware, the second only software. The following experiments combine all three aspects, however in different relations.

Table 2
Summary of the experiments prepared in the lab course.

Experiment No.	Experiment Title	Topics and Learning Objectives
Experiment I	**Ready, Steady, Go:** Initial Operation of the Embedded System	• Connecting and handling the Raspberry Pi • Starting and configuring the operating system • Updating the software
Experiment II	**Hello Python:** Introduction to Python	• Motivation and introduction to Python • Programming with Python in version 2 and 3 • Understanding the concept of an algorithm
Experiment III	**Say Cheese:** Introduction to Image Signal Processing with Python	• Properties of digital images • Creating an image with Python Imaging Library (PIL) • Processing of the created image • Comparing the runtime of different algorithms
Experiment IV	**Take a Picture:** Image Signal Processing with a Camera	• Properties of digital imaging • Connecting and testing the USB camera • Application and understanding of different digital filters • Implementing algorithms for image enhancement
Experiment V	**From Machine's Point of View:** Introduction to Computer Vision	• Overview of typical features in images • Detection of edges • Comparison of different edge detectors
Experiment VI	**Great View:** Creating a Panorama, Part 1	• Introduction to panoramic imaging • Implementing the Harris and Stephens corner detector • Analyzing the results
Experiment VII	**Pixel Puzzle:** Creating a Panorama, Part 2	• Implementation of a user interface • Introduction to scale-invariant feature transform algorithm (SIFT) • Implementation of SIFT • Testing the program and analyzing errors

The lab course "Image and Video Signal Processing on Embedded Systems" imparts knowledge and fulfills the following learning objectives:

- Students learn how to handle different kind of hardware and understand the importance of a proper handling of the hardware.
- Students learn how to read and write the programming language Python as alternative to MATLAB.
- Students understand and explain the image signal processing path.
- Students classify and apply different kind of image filters.
- Students are able to create a whole program, from the idea to implementation and debugging.
- Students learn how to solve a problem on their own and are able to apply their knowledge.
- Students learn how to handle an operating system and different software on the embedded hardware, and understand the importance of periodically updating the software.

After successfully finishing the lab course, the students have obtained new knowledge and abilities, which they can use in their future professional careers. A detailed overview of the learning objectives with respect to each experiment is given in Table 2.

All experiments are identically constructed. The front page of the description includes a catchy title, a matching image, and an informative subtitle. In addition, motivation, aim, and thematically background are described on the first page. On the second page, content overview, requirements for hardware and software, some links for self-study and comments for the experiment are listed. The detailed description and tasks of the experiment begin on the third page. The given code and other commands are marked by a different font or are summarized in tables. Important chapters are labeled by an exclamation mark. The tasks are specially separated by a horizontal line. So, the laboratory manual illustrates the topics and aims of the lab course to the students in a clear and structured way. Experiments I to VI each contain an optional exercise, which is not necessary for passing the lab course. If a group has completed all tasks of an experiment before the end of time, these additional tasks can be carried out.

4. Study area

In the following subsections, the seven experiments are explained in detail. Table 2 shows a summary of the experiments, their topics, and their learning objectives. The experiments have been especially created for the Raspberry Pi, however, they are not bounded to it. They can also be solved on other embedded systems with different operating systems.

However, it is important that Python, the needed Python libraries, and OpenCV are installed on the system. So, the whole laboratory tasks are transferable.

4.1. Experiment I: Initial operation of the embedded system

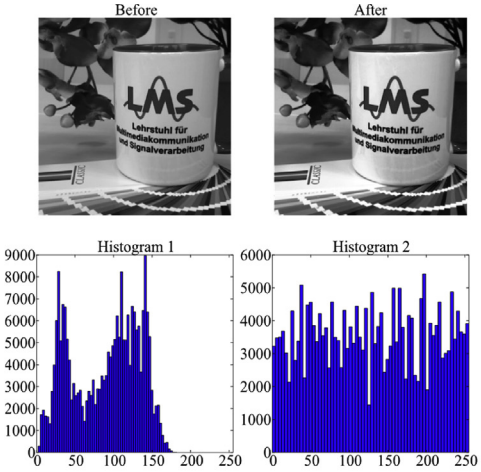
In the first experiment, the whole setup is put into operation after explaining the physical structure of the Raspberry Pi, as depicted in Fig. 1. The display, mouse, and keyboard are connected with the Raspberry Pi and the micro SD card with the current OS Raspbian Jessie is put into the micro SD slot. After booting the Raspberry Pi, the system is set up, i.e., changing hostname, keyboard layout, expanding filesystem, etc. Also a connection to the Wi-Fi is established and the OS is upgraded. The aspect hardware dominates this experiment. So, the students learn the handling with the Raspberry Pi. A simple introduction to the lab course is possible because of the low level of difficulty.

4.2. Experiment II: Introduction to Python

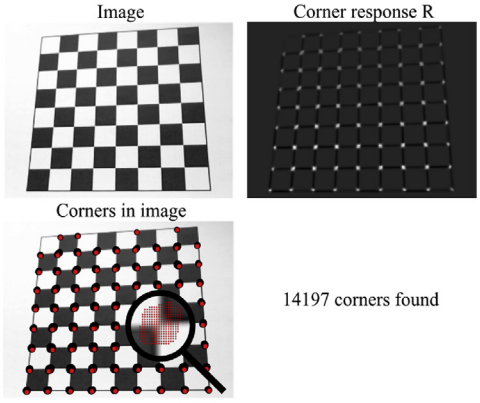
In this experiment, the programming language Python is introduced on the embedded system. The aspect software dominates this experiment. A motivation for learning Python, an introduction to the syntax, and the first tasks are part of this experiment. In addition, the students learn the difference between version 2 and version 3 of the programming language. The print function and the keywords of the different versions are used mainly to show the varieties. The understanding of the development of algorithms is taught by implementing small coding functions, e.g., the transfer of data is implemented in Python. So, the import and export of image data is introduced in a simple way. The students learn the basics of the programming language and enlarge their knowledge in the additional experiments. Furthermore, they learn the basics for developing a complete program.

4.3. Experiment III: Introduction to image signal processing with Python

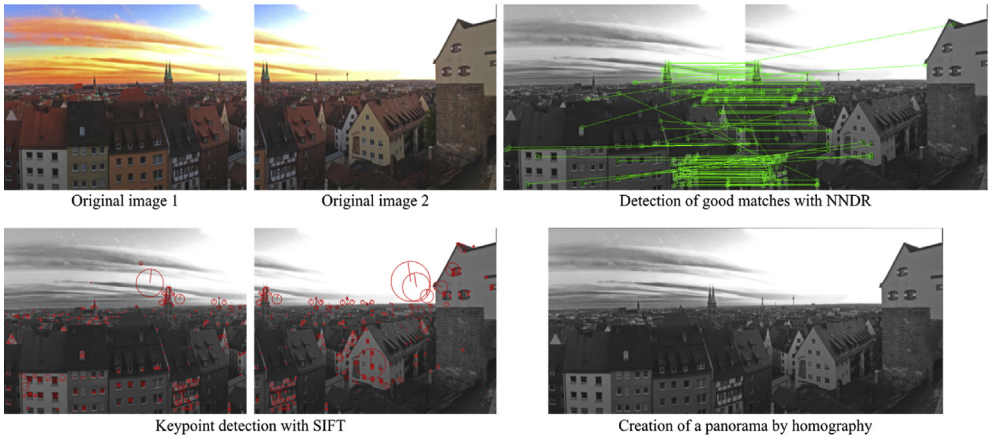
In the third experiment, the fundamentals of image signal processing with Python are introduced. The path of the image signal processing is explained in detail (Fig. 2, red flowchart). The recording is the conversion of an analogous light signal into a digital representation. Each pixel possesses a value for the color information, the number of pixels in horizontal or vertical direction represent the resolution of an image. The substantial task of the processing part is to eliminate the errors caused while recording an image. Therefore, different pre-processing steps like converting to grayscale, scaling the image, or increasing contrast can be performed. The goal is to process the data in a way which prevents the viewer from perceiving errors during the replay. The properties of images like resolution and color are also described by a coordinate system of a bitmap and different color spaces. This theoretical description is the foundation for later computer vision experiments. The aspects software and theory dominate this experiment. At the beginning, Python Imaging Library (PIL) (Lundh and Ellis, 2018) and FileZilla are installed. PIL adds support to Python for opening, saving and changing different image file formats. To learn and train more Python, a chessboard is created and modified with the PIL on different ways. The students learn how to save



(a) Experiment IV: Histogram equalization.



(b) Experiment VI: Harris and Stephens corner detector.



(c) Experiment VII: Feature matching.

Fig. 4. Example images of the result plots of the Experiments IV, VI and VII.

and open an image, and how to transpose and enframe the created cheeseboard. The runtimes of the different implementations are measured and compared. Thus, the students figure out how challenging image signal processing on an embedded system can be. This experiment benefits the understanding of a general implementation of functions. The more general a function is implemented, the easier the post-processing and the usage.

4.4. Experiment IV: Image signal processing with a camera

In this experiment the camera is connected to the Raspberry Pi before the properties of digital imaging are explained. At the beginning, Numerical Python (NumPy) (NumPy Developers, 2018), Python for Science (SciPy) (SciPy Developers, 2018) and Matplotlib (Hunter et al., 2018) are installed. NumPy is an effective library for displaying and calculating N-dimensional arrays. SciPy expands Python by many efficient mathematical functionalities. Matplotlib is a comprehensive and efficient library for plotting different images and graphs. After the installation, the students take some pictures while changing the parameters brightness and contrast. The poor quality of the images should be a motivation for applying some digital filters to the images for improving the quality. After converting the acquired images into grayscale, the distribution of the pixel values is shown with the help of a histogram. The students should realize that depending on the setting of brightness, different pixel value areas dominate in the histogram. For improving the dynamic range of the image, a histogram equalization should be implemented and applied to the different images. The implementation of the algorithm is getting more complex. Thus, to compare the calculated results some plots with the correct results are included in the laboratory manual. To increase the level of difficulty, less source code is given in this experiment. The focus in this experiment is on recording and pre-processing the images in the path of image signal processing (Fig. 2, red flowchart). Furthermore, it is shown, that efficient pre-processing methods are necessary for complex computer vision algorithms. The students also understand how digital imaging is working in theory and practically. All three aspects are present in this experiment, however, the aspect software dominates.

In Fig. 4a, a captured image with a logo is used to demonstrate the histogram equalization. Before, the values of the images range from around 0 to 175 with two peaks at around 30 and 125. After the equalization, the whole value range is used uniformly and the contrast is higher. The reflection in the cup is better recognizable and the color fan is illustrated with more shades of gray and is more distinguishable.

4.5. Experiment V: Introduction to computer vision

In the fifth experiment, an introduction to computer vision is given. Before the students can begin with the experiment, the Python library OpenCV version 2.4.13.3 is installed (OpenCV team, 2018). OpenCV is an open and free C++ library for computer vision tasks. It has been designed for computational efficiency and real-time applications. Since the compilation of this packet takes several hours on the Raspberry Pi, a pre-compiled version of OpenCV is provided on the server, which has to be installed. OpenCV contains a lot of computer vision algorithms which will be used in this and the following experiments. The image signal processing in Fig. 2 (red flowchart) will be extended by the content-based signal analysis (blue flowchart). After pre-processing, the image signal will be analyzed by extraction, recognition, and comparison of features in images, and the images will be composed for replaying. The experiment focuses on feature recognition. Therefore, typical features in an image are introduced and different methods for edge and corner detection are implemented in this experiment. A horizontal and vertical gradient operator, the Sobel operator, the Laplace operator, and the Laplacian of Gaussian operator are implemented and compared (Ziou and Tabbone, 1998). Advantages and disadvantages of the different operators are shown on images taken by the students. So, a filter can improve the detection of edges and corners. This experiment is mathematically demanding, so new Python elements are not introduced. In addition, the students understand the theory of features and test different edge detectors. Thus, the aspect theory dominates this experiment.

On the left side of Fig. 5, a Python code fragment of the implemented Sobel operator is depicted. In this exercise, the students implement their own Sobel filter and for comparison use the provided Python Sobel filter. Different possible rotations of the filter should be tested to see that the Sobel filter is able to identify not only horizontal and vertical edges but also diagonal structures. Furthermore, the students learn that the Sobel filter outperforms the gradient operator.

4.6. Experiment VI: Creating a panorama, part 1

In this experiment, the fundamentals of panoramic imaging are explained. In a panorama, a broader range from a scene is captured than a human eye can perceive or a camera can capture. For creating a panorama, several single images are captured and stitched together, while the shooting location of the camera stays the same for all images. Motion in a scene, different illumination, or perspective distortion cause problem and errors in a panorama. The distortion can be eliminated by a correct projection. Therefore, unique features have to be found in the captured images and compared with each other. After finding some matches, the images can be warped and combined. Detecting unique features in an image is necessary for a good panorama. So, the students are implementing the Harris and Stephens corner detector (Harris and Stephens, 1988). To illustrate the corner detection, different plots of a synthetic and a noisy image are made. Thereby, also a 3D plot is implemented by the students. However, the Harris and Stephens corner detector finds way too many corners in a synthetic image and thus, it is not a good detector for creating a panorama. The level of difficulty for the theory of the algorithm and the mathematical description is high. So, new Python elements

```python
def sobelFilter1(npImg):
    Fm = np.zeros( npImg.shape )
    Fn = np.zeros( npImg.shape )
    ndim.sobel( npImg, 1, Fm )
    ndim.sobel( npImg, 0, Fn )
    magnitude = sqrt( Fm**2 + Fn**2 )
    phase = np.arctan2( Fn, Fm )
    return Fm, Fn, magnitude, phase

def sobelFilter2( npImg, Dm, Dn):
    Fm = np.zeros( npImg.shape )
    Fn = np.zeros( npImg.shape )
    ndim.correlate( np)mg, Dm, Fm )
    ndim.correlate( npImg, Dn, Fn )
    magnitude = sqrt( Fm**2 + Fn**2 )
    phase = np.arctan2( Fn, Fm )
    return Fm, Fn, magnitude, phase
```

```python
def computeHarrisCorners(Fm, Fn, k=0.04, sigma=3):
    Mmm = filters.gaussian_filter(Fm*Fm, sigma)
    Mmn = filters.gaussian_filter(Fm*Fn, sigma)
    Mnm = Mmn
    Mnn = filters.gaussian_filter(Fn*Fn, sigma)

    detM = Mmm*Mnn - Mmn**2
    traM = Mmm + Mnn
    R = detM - k*traM*traM

    return Mmm, Mmn, Mnm, Mnn, detM, traM, R

def detectCorners( R, threshold=0.5 ):
    cornersCandidates = np.uint8((R > R.max() * threshold))
    cornerPos = np.array(cornersCandidates.nonzero()).T
    return cornerPos
```

Fig. 5. Python code fragments for the Experiments V (left) and VI (right).

are also not introduced in this experiment. However, the students learn the theory of creating a panorama. In addition, the aspects theory and software dominate this experiment.

In Fig. 4b, the original image of a chessboard, the corner response function R of the Harris and Stephens corner detector (Harris and Stephens, 1988), and the detected corners are shown. R is negative in edge regions, positive in corner regions and almost zero in flat regions. So, the corners of the chessboard are marked with a white dot in the upper right image. Around 14000 corners can be found in this example, however, some corners in the top of the chessboard cannot be recognized. A zoom into the chessboard depicts why so many corners can be found. In the zoomed in area around 264 possible corners are detected. Thus, the Harris and Stephens corner detector is not a reliable detector for creating a panorama and should be replaced by more advanced features.

Furthermore, on the right side of Fig. 5 the Python code of the Harris and Stephens corner detector is depicted. With the support of the paper of Harris and Stephens (Harris and Stephens, 1988), the students implement the detector gradually. For the corner response function R the determinant and the trace of the matrix elements are used. With the value of the response function R, a specific pixel can be defined as edge, corner, or flat. By thresholding the response function R, the corners in the image can be detected and used for further processing.

4.7. Experiment VII: Creating a panorama, part 2

In the last experiment, the final algorithm for creating a panorama is implemented as shown in Fig. 2. An interface (green dashed flowchart) enables the user to configure essential parameters and to control the whole process of the algorithm. In the first step, single frames are taken with the USB camera. Pre-processing with conversion to grayscale, downsampling and increasing contrast, feature extraction and feature recognition are done by the scale-invariant feature transform (SIFT) (Lowe, 2004). After that, the images are stitched together and can be replayed on the Raspberry Pi. In the first step, the scale-space extrema detector is used for feature extraction, i.e., the grayscale image is convolved with a Gaussian filter at different scalings, the differences of successive blurred images are taken, and then maxima and minima (keypoints) are detected in the difference of Gaussian. The found keypoints are characterized by a descriptor vector, i.e., magnitude and angle are describing the keypoints. The detected keypoints are compared by nearest neighbor distance ratio (NNDR), i.e., the ratio of the distance to the two nearest neighbor is calculated. A good match is found, if the ratio is smaller than 0.8, i.e. 90 % of false positive and less than 5 % of true positive are eliminated (Lowe, 2004). The good matches are used for calculating the homography matrix. To eliminate occurring outliers, the random sample consensus (RANSAC) is further applied (Fischler and Bolles, 1981). After calculating and optimizing the homography matrix, the panorama can be created. At the end, a load test for the RAM of the Raspberry Pi is performed. Therefore, a panorama with different resolutions is created while the used RAM is observed. If the resolution is set too high (about 1920×1080 pixels), the Raspberry Pi will be stretched to its limits, i.e., there is no freely avaible memory left and the Raspberry Pi crashes. Thus, the students learn that an embedded system has its limitation in memory and performance. In this experiment the students learn to create a whole program, and how to analyze errors.

In Fig. 4c the individual steps of the program are shown with two images. First, the images are taken with the USB camera (top left). After that, the images are converted into grayscale and the keypoints are detected in both images with SIFT and marked with red circles (bottom left). The bigger a circle, the more important this keypoint is for further processing. Good matches are found by NNDR and connected by a green line (top right). However, not every connection is a good one, like the connection between the corner of the right house and a tower, which will be eliminated by RANSAC. In the end the two images are scaled and combined to a panorama by the calculated homography matrix (bottom right).

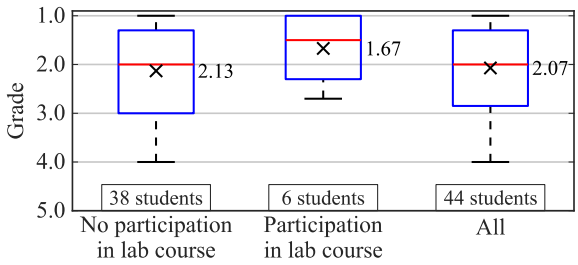


Fig. 6. Boxplot of the linked lecture "Image, Video, and Multidimensional Signal Processing". The black crosses indicate the mean value. The grades range from 1 to 5, whereby 1 is the highest and 5 is the lowest grade.

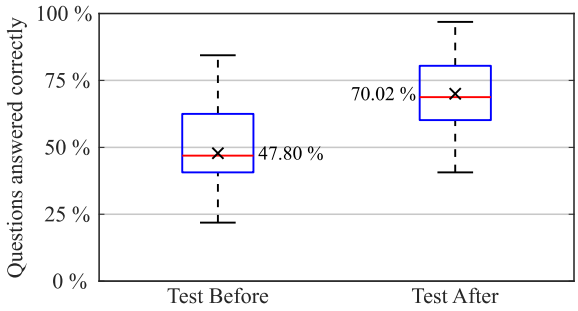


Fig. 7. Boxplot of the tests before and after the lab course. The black crosses indicate the mean values.

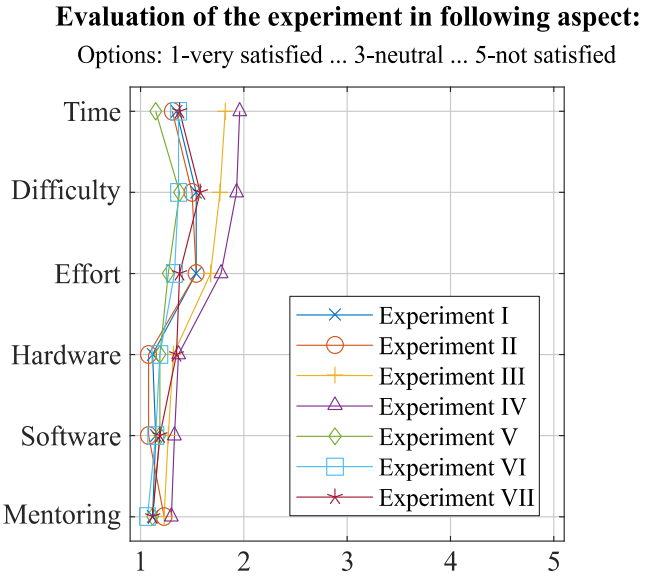


Fig. 8. Average student evaluation of the individual experiments of the lab course.

Table 3
Average student evaluation of the lab course.

Evaluation questions	Average
For these questions: 1-excellent … 3-fair … 5-very poor	
How would you evaluate the handling with the Raspberry?	1.41
How would you evaluate the laboratory manual?	1.33
How would you evaluate the experiments?	1.33
How would you evaluate the organization of the laboratory?	1.19
Overall, how would you evaluate the laboratory?	1.30
For these questions: 1-too high … 3-about right … 5-too low	
The time spent in the laboratory was:	2.59
The time spent for preparation was:	2.78

5. Results & discussion

At the Faculty of Engineering of the FAU, every semester an evaluation of the different lectures, tutorials, seminars, and laboratories takes places. Therewith, the students have the opportunity to contribute to the improvements of teaching at the FAU. The "Image and Video Signal Processing on Embedded Systems" lab course has also been evaluated by the evaluation system of the university. The students have rated the lab course with a 1.15, whereby 1 is the highest and 5 is the lowest grade. This is much higher than the average voting of 2.09 for all lab courses at the FAU and leads to a third place out of in total 50 in the category lab courses at the FAU. The consent for collecting the data and publishing these results is given by the Evaluation Board and the managing Officer.

To ensure that the lab course can improve the students' knowledge about image and video signal processing, the grades of the exam of the linked lecture "Image, Video, and Multidimensional Signal Processing" are analyzed. Since the content of the lecture and the laboratory largely overlap, the knowledge in image and signal processing can be improved by participating in the lab course. Altogether, 44 students have participated in the exam and 27 have participated in the lab course. Altogether, 6 out of these have participated in both the exam and the lab course. The exam was performed after the lab course, so the impact of the lab course for learning the topics of image and video signal processing can be evaluated. In Fig. 6 the performance of the different groups of students are depicted as boxplot. The central red mark indicates the median value, the blue bottom and top edges of the boxes the 25th and 75th percentiles, respectively, and the black whiskers the most extreme data points. The black crosses indicate the mean value. The grades ranges from 1 to 5, whereby 1 is the highest and 5 is the lowest grade. It can be seen that the students who participated in the lab course perform better with a grade of 1.67 than the peer students (average grad 2.13). Evidently, the students seem to memorize the content better, because they learn it not only in theory but also practically.

Furthermore, before and after the lab course a test with 15 questions about image and video signal processing on embedded systems was performed. 27 students have participated in the lab course and have taken part in the test. Thus, it can be seen how the students have improved in this topic during the lab course. In Fig. 7 the results of both tests are depicted as a boxplot. The description of the boxplot can be found in the previous section. It can be seen, that the students have been able to improve their knowledge about image and video signal processing. On average, the students could answer around 47.80 % of the questions in the entrance test, and around 70.02 % in the second test. The lowest performance for the entrance test is 21.88 % and the highest is 84.38 %. For the second test both values could be outperformed with 40.63 % and 96.88 %.

In addition, each experiment as well as the whole lab course have been evaluated by the students using an evaluation sheet. 27 students took part in this evaluation again. In Fig. 8 the evaluation of the individual experiments is shown. The students have evaluated the experiments in the aspect of time, difficulty, effort, hardware, software, and mentoring. All aspects were rated between 1 and 1.96 on a five-point scale. In Table 3 the questions and results of the evaluation of the whole lab course are summarized. The handling with the Raspberry Pi, the laboratory manual, the experiments and the organization of the lab course have been evaluated between excellent and good. Overall, the lab course has been evaluated with 1.30. The time spent in the laboratory and for preparation has been evaluated between "about right" and "high". For all test results described above the consent for publication was given by the participating students.

Overall, the effectiveness of the lab course can be confirmed by comparing the grades of the exam of the linked lecture and by the results of the test before and after the lab course. Furthermore, the lab course has been self-evaluated by the students as very good.

6. Conclusion

In this paper the newly developed lab course "Image and Video Signal Processing on Embedded Systems" has been presented. The Raspberry Pi 3 Model B has been chosen as embedded system because of property, availability, and cost-efficiency. Different programming tasks have to be solved by the groups on the Raspberry Pi in the programming language Python. Python has been chosen because of readability, easy learning, and clearness. In addition, Python is an open source programming language. The requirements and the seven different experiments that have to be prepared are presented in this paper. Furthermore, the learning objectives were highlighted in the paper. The students learn the handling with hardware and software, to read and write in Python, the image signal processing path, and the development of a program. The improvement of the students in image and video signal processing was verified by the grades of the exam of the linked lecture and the assessment of a test before and after the lab course. Furthermore, the lab course has been evaluated by the participating students in the evaluation system of the university and in a self-developed evaluation sheet. Overall, in both evaluation systems the lab course has been rated between excellent and very good. The lab course can easily be extended by further experiments. In the future it is planned to add two new experiments. The first one will deal with object detection and the other one with background replacement. So, the aspect of *video* signal processing will be given more weight in the lab course.

Declarations

Author contribution statement

Karina Jaskolka: Conceived and designed the experiments; Performed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data; Wrote the paper.

Jürgen Seiler: Conceived and designed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data; Wrote the paper.

Frank Beyer: Conceived and designed the experiments.

André Kaup: Conceived and designed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data.

Funding statement

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Competing interest statement

The authors declare no conflict of interest.

Additional information

No additional information is available for this paper.

References

Abbot, D., 2018. Linux for Embedded and Real-Time Applications. Newnes.

Basler, A.G., 2019. Basler the Power of Sight. Retrieved from. https://www.baslerweb.com.

BeagleBoard.org Foundation, 2018. BeagleBone Black. Retrieved from. http://beagleboard.org/black.

Bernini, N., Bombini, L., Buzzoni, M., Cerri, P., Grisleri, P., 2014. An embedded system for counting passengers in public transportation vehicles. In: Proc. IEEE/ASME 10th International Conference on Mechatronic and Embedded Systems and Applications, pp. 1–6.

Bhat, A., Samii, S., Rajkumar, R., 2017. Practical task allocation for software fault-tolerance and its implementation in embedded automotive systems. In: Proc. IEEE Real-Time and Embedded Technology and Applications Symposium, pp. 87–98.

Cass, S., 2018. The 2017 Top Programming Languages. Retrieved from. https://spectrum.ieee.org/computing/software/the-2017-top-programming-languages.
Castellanos, J.C., Susin, A.A., Fruett, F., 2011. Embedded sensor system and techniques to evaluate the comfort in public transportation. In: Proc. 14th International IEEE Conference on Intelligent Transportation Systems, pp. 1858–1863.
Fischler, M.A., Bolles, R.C., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM 24 (6), 381–395.
Harris, C., Stephens, M., 1988. A combined corner and edge detector. Proc. Fourth Alvey Vis. Conf. 147–151.
Hodson, D., 1993. Re-thinking old ways: towards a more critical approach to practical work in school science. Stud. Sci. Educ. 22 (1), 85–142.
Hofstein, A., Lunetta, V.N., 1982. The role of the laboratory in science teaching: neglected aspects of research. Rev. Educ. Res. 52 (2), 201–217.
Hofstein, A., Lunetta, V.N., 2004. The laboratory in science education: foundations for the twenty-first century. Sci. Educ. 88 (1), 28–54.
Hunter, J., Dale, D., Firing, E., Droettboom, M., 2018. Matplotlib. Retrieved from. http://www.matplotlib.org.
Kosse, T., 2018. FileZilla - the Free FTP Solution. Retrieved from. https://filezilla-project.org/.
Lowe, D.G., 2004. Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. 60 (2), 91–110.
Lundh, F., Ellis, M., 2018. Python Imaging Library (PIL). Retrieved from. http://www.pythonware.com/products/pil.
Lunetta, V.N., Hofstein, A., Clough, M.P., 2007. Learning and teaching in the school science laboratory: an analysis of research, theory, and practice. Handb. Res. Sci. Educ. 393–441.
Mao, H., Yao, S., Tang, T., Li, B., Yao, J., Wang, Y., 2017. Towards real-time object detection on embedded systems. IEEE Trans. Emerg. Top. Comput. 1-1.
Mastinu, E., Doguet, P., Botquin, Y., Håkansson, B., Ortiz-Catalan, M., 2017. Embedded system for prosthetic control using implanted neuromuscular interfaces accessed via an osseointegrated implant. IEEE Trans. Biomed. Circuits.Syst. 11 (4), 867–877.
NumPy Developers, 2018. NumPy. Retrieved from. http://www.numpy.org.
OpenCV team, 2018. OpenCV. Retrieved from. http://www.opencv.org.
Python Software Foundation, 2018. Python. Retrieved from. https://www.python.org/.
Raghunathan, V., Kansal, A., Hsu, J., Friedman, J., Srivastava, M., 2005. Design considerations for solar energy harvesting wireless embedded systems. In: Proc. Fourth International Symposium on Information Processing in Sensor Networks, pp. 457–462.
Raspberry Pi Foundation, 2018. Raspberry Pi - Teach, Learn, and Make with Raspberry Pi. Retrieved from. https://www.raspberrypi.org/.
Rupniewski, M., Mazurek, G., Gambrych, J., Nałęcz, M., Karolewski, R., 2016. A real-time embedded heterogeneous GPU/FPGA parallel system for radar signal processing. In: Proc. Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress, pp. 1189–1197.
Schwartz, R.F., 1959. Laboratory: its scope and philosophy. IRE Trans. Edu. 2 (4), 120–122.
SciPy Developers, 2018. SciPy. Retrieved from. https://www.scipy.org.
Sharp, D.C., Bell, A.E., Gold, J.J., Gibbar, K.W., Gvillo, D.W., Knight, V.M., Weismuller, S.P., 2010. Challenges and solutions for embedded and networked aerospace software systems. Proc. IEEE 621–634.
Stoianov, I., Nachman, L., Madden, S., Tokmouline, T., 2007. PIPENET: a wireless sensor network for pipeline monitoring. In: Proc. 6th International Symposium on Information Processing in Sensor Networks, pp. 264–273.
Tsai, W.K., Lai, C.J., Sheu, M.H., Chen, T.H., 2014. High dynamic range image based on block-based edge strength for embedded system design. In: Proc. Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pp. 329–332.
Vahid, F., Givargis, T., 2002. Embedded System Design: A Unified Hardware/Software Introduction. John Wiley & Sons.
Wang, J., 2017. Real-Time Embedded Systems. John Wiley & Sons.
Ziou, D., Tabbone, S., 1998. Edge detection techniques-an overview. Pattern Recognition and Image Analysis C/C of Raspoznavaniye Obrazov I Analiz Izobrazhenii 8, 537–559.