UDC 004.932 **doi: 10.20998/2522-9052.2020.2.13**

K. Dergachov, L. Krasnov, O. Cheliadin, R. Kazatinskij

National Aerospace University "Kharkiv Aviation Institute", Kharkiv, Ukraine

VIDEO DATA QUALITY IMPROVEMENT METHODS AND TOOLS DEVELOPMENT FOR MOBILE VISION SYSTEMS

Abstract. Subject of study. The article proposes new methods of input and preliminary processing of video data for web- and specialized pi-cameras in monocular and stereo vision systems based on Raspberry Pi microcomputers to improve the quality of work of modern mobile vision systems. This approach is always relevant, since the design of modern vision systems constantly requires new non-trivial hardware, algorithmic and software solutions. **Objectives.** The goals are to compare quality indicators of the known methods of input and preliminary processing of video data in vision systems and to develop new methods and algorithms providing better speed, the necessary frame resolution and the independence of the frame brightness from changes in scene illumination while reading video data. **Methods used.** The paper formulates a comprehensive criterion for improving the quality of the Raspberry Pi microcomputer video input and preprocessing video data. Based on the accepted quality indicators (video input speed, resolution, and stability indicators of average brightness of the current frames of the received video stream), video input and preprocessing algorithms that satisfy the specified requirements are synthesized. This allows to find the optimal method for processing video data and to overcome the contradiction of reducing the input speed due to the need of increasing the resolution of video frames for each project. The created universal program for input and preliminary processing of these data allowed to obtain quantitative estimates of the effectiveness of the developed algorithms and formulate recommendations for their further use. All this allows you to significantly increase the efficiency of using Raspberry Pi microcomputers in modern mobile vision systems. **The results obtained** are the basis for the creation of a universal software product for high-speed input (in real time) and preliminary processing of video data for face detection and recognition systems, as well as stereo vision systems. **Conclusions.** The conducted experimental studies confirmed the efficiency and effectiveness of the proposed methods and algorithms for high-speed input of video data with different values of the resolution of the frame and the ability to adaptively adjust its brightness. Based on the created methods and algorithms, various options for its software implementation are proposed. This allows us to recommend the results for practical use. Prospects for further research include the expansion of the vector of criteria for assessing quality and features for optimizing video data, as well as the creation of new algorithms and various versions of programs based on them.

Keywords: Raspberry Pi microcomputer; algorithms and program codes for input and preprocessing of video data; OpenCV library functions.

Introduction

Rapidly developing mobile application engineering actively uses vision systems (here in after – VS) based on small-sized video recorders and micro-computers.

The need for such systems is especially big in robotics, when equipping unmanned aerial vehicles, automobile vehicles etc.

Subject of study. When designing mobile small-sized VS, designers of hardware and software systems must first solve a number of serious problems associated with choosing an economical autonomous power supply, a microcomputer with sufficient speed and memory capacity, suitable video recorders and modern software.

In addition, even at the stage of preliminary design, it is necessary to introduce quality criteria for the received video information, which correspond to accepted standards and allow using the available hardware and software resources with maximum efficiency.

The main difficulty in building modern mobile VS is the lack of a unified approach to the selection of suitable microcomputers, cameras, and methods for inputting and pre-processing video data in the conditions of limited resources of the processing system for speed. Therefore, we will conduct a comparative analysis of known methods and means, and also consider in detail new technical solutions. Such a task is quite modern and relevant, since designing effective VS requires new approaches, non-trivial hardware, algorithmic and software solutions.

The goal of this study is analysis of well-known methods of input and preliminary processing of video data during VS implementation, development and creation of new algorithms and software tools guaranteed to ensure high quality of the original video data and real-time processing of video information.

Assessment of the quality and effectiveness of new methods and means of obtaining and processing video data must be carried out in laboratory conditions, and the reliability of the results should be checked by statistical analysis of the data.

1. Problem statement

Criteria for assessing the quality of the source video in VS. A systematic analysis of existing methods of input and processing of video data in various VS is possible only on the basis of objective quality criteria. To create them, we introduce the following indicators, define and decode the corresponding abbreviations:

- FPS – (Frame per second). This value characterizes the speed of changing / reading frames of the recorded video stream. The maximum FPS value is limited by the passport data of the camcorder used;
- FR – (Frame Resolution) describes the resolution of the video frame in pixels (320x240, 640x480, 1280x960, etc.);
- FMB – (Frame medium brightness). This is an indicator of the average level of brightness of the frame, objectively reflecting the degree of illumination of the scene at the current time.

These are the most important indicators mainly

determine the quality indicators of the video processing system in VS.

Traditional methods of capturing video data significantly slow down the processing speed of video data, which often seriously limits the ability to work in real time. However, using multithreading can significantly reduce the impact of Input/Output latency, leaving the main thread without blocking. Therefore, there is the possibility of increasing FPS up to the technological limit of the camcorder used.

It is clear that in solving problems of video data visualization, the desire to improve quality by increasing the resolution of the frame (FR) is always justified. However, it should be noted that a large increase in the number of pixels in the frame will inevitably lead to a slowdown (decrease in the FPS). At the same time, a compromise can be found experimentally - using the proposed new effective algorithms for capturing and entering video data.

Note that the knowledge of the video stream frame medium brightness (FMB) in conditions of high variability of the illumination of the scene is very useful. This allows, by performing threshold procedures, to set the mode to enable / disable contrasting (equalization) of the frames of the video stream. In addition, with a low level of frame brightness (by comparing FMB with a predetermined threshold) in long-term video surveillance systems, it is convenient to turn on/off the backlight of the camcorder.

Based on the accepted quality indicators, the authors will further formulate recommendations on the construction of high-quality video input algorithms for modern mobile VS and give examples of their practical implementation.

2. Review of the literature

Key resources for building mobile VS. Consider more carefully and evaluate in detail all aspects of projects for the construction of modern mobile VSs systems. In this case, we will be guided by the accepted criteria for the quality of video data and rely on the available data of modern literature and internet resources.

Computers for mobile VS. To meet the needs of this market segment, a fairly large number of microcomputer models (Lego, Intel Galileo, Android IOIO OTG, Arduino, etc.) are now available. However, the Raspberry Pi platform holds strong leadership in this series [1 - 4]. It is represented by a line of models having various hardware implementations at affordable prices. The best of them are comparable in performance to desktop PCs. So the Raspberry Pi 4 Model B has 4 GB of RAM, a fast 4-core processor (1.5 GHz), support for two displays with a resolution of up to 4K, Gigabit Ethernet, USB 3.0, Wi-Fi, Bluetooth 5.0 and power via USB -C (5V/3A). Most importantly, the Raspberry Pi 4 has two USB 3.0 ports. They are 10 times faster compared to USB 2.0 and are well suited for connecting fast peripheral units (flash drives, web-cameras, etc.).

Digital video recorders (DVRs) for mobile VS. Typically, mobile VS are equipped with low-cost, low-resolution, non-calibrated web cameras that can be connected to the Raspberry Pi via USB ports. This method of organizing video surveillance has several disadvantages. The main one is that when synthesizing stereo vision systems, you cannot use one USB bus. This eliminates the possibility of camera synchronization and significantly reduces the total system bandwidth. At the same time, it is difficult to take into account the desynchronization due to the multitasking of modern operating systems and the functioning of the task scheduler. Camera desynchro-nization of not more than 10 ms is acceptable [7].

Specialized pi-cameras for Raspberry Pi boards have higher quality than web-cameras. They are used for monocular video surveillance systems. For example, the Raspberry Pi Camera Module v2 is a high-quality Sony IMX219 image sensor with a fixed focus. It connects to the microcomputer board using a special CSI interface (Camera Serial Interface). The sensor measures 25mm × 23mm × 9mm and weighs 3g. A short ribbon cable is used to connect to the Raspberry Pi. Key features of this sensor are:

- 8 megapixel camera capable of taking photos with a resolution of 3280 × 2464 pixels;
- Video recording with a resolution of 1080 p 30 FPS, 720 p 60FPS, 640 × 480 p 60/90FPS;
- Software is supported in the latest version of the Raspbian Operating System;

Here and further these abbreviations are used: p – pixel, FPS – frame per second.

Note that for a long time there was no technical solution for the implementation of a stereo pair with two pi-cameras. This is due to hardware design limitations − standard Raspberry Pi models contain only one camera port. To date, the problem has been resolved by the efforts of Arducam, which released the HAT stereo camera for the Raspberry Pi [13]. It allows to connect two 5-megapixel OV5647 or two 8-megapixel pi-cameras IMX219 to a standard Raspberry Pi board via an additional expansion board and simultaneously capture images or video. It is very important that both cameras are fully synchronized (accurate to units of ns). Further details on the use of this technology will be described below.

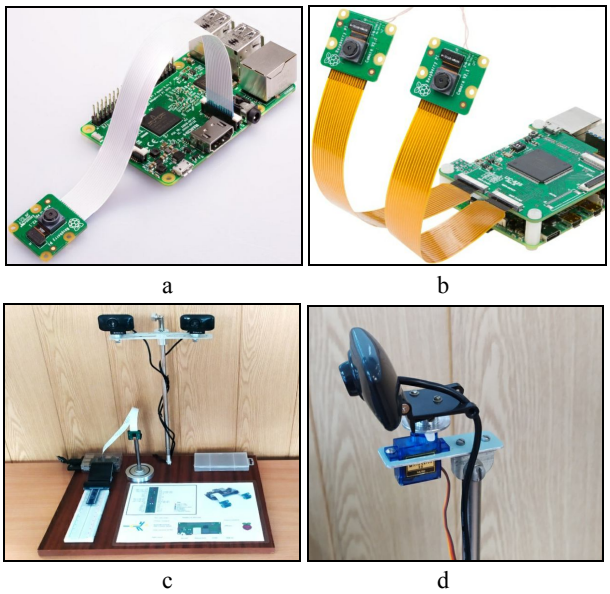
Programming tools. Raspberry Pi computer-based vision systems software typically uses the developer-recommended Raspbian operating system and the Python programming language. Python, as a modern object-oriented language, is most often used to program General Purpose Input/Output (GPIO) I/O ports on the Raspberry Pi and is part of the Raspbian operating system. It is possible to connect a large number of specialized libraries to Python, expanding the capabilities of solving the problems of necessary applications. It is especially important that in conjunction with Python you can use OpenCV − a library of algorithms for computer vision, image processing and general-purpose numerical algorithms with open source [6-12]. The convenience of working with Python also consists in the possibility of working with the Windows operating system (on a personal computer) and on the Raspberry Pi microcomputer with the Raspbian operating system. However, it is necessary to ensure that the versions of the libraries used in the project are completely identical.

In addition to the listed requirements in VS, it is necessary to carry out data input from video cameras in real time, and at the same time carry out complex video processing. In monocular video surveillance systems, for example, this is the task of detecting and recognizing faces, and in stereoscopic systems it is the task of constructing depth maps and volumetric reconstruction of the scene. In severe conditions of limited resources (hardware and software), they often resort to multi-threaded processing of the original video data, which can significantly improve the performance of video processing.

3. Materials and methods

Tools for constructing video data input algorithms. When performing the work, the authors used the following hardware and software resources:

Hardware. To conduct experimental studies, a laboratory bench was created on the basis of the Raspberry Pi 3 Model B microcomputer and a set of one or two web cameras and a pi camera. The appearance of the laboratory bench is shown in FIG. 1.



a b c d

Fig. 1. Fragments of the laboratory installation

Identical FC-250 web-cameras (FPS = 30 at resolution FR = 1280x960) are designed to create stereo pairs with the ability to connect them to a computer via USB ports.

The design features of the stereo pair are shown in FIG. 1, c. When cameras are rigidly mounted relative to each other, it is possible to change the height of the stereo pair, rotate it 360° around the vertical axis. Each camera can be rotated at a small angle in the horizontal and vertical planes. But the main thing is the adjustment of the distance between the cameras using a special control line and the installation of this distance with an accuracy of fractions of mm. This is very important when changing the base distance.

A single pi-camera (in our case, the previously described Sony IMX219 Exmor camera) is used for a monocular vision system. It is mounted on a separate tripod (Fig. 1, c) and connected to the Raspberry Pi through a special CSI video input. This significantly reduces the load on the CPU compared to connecting cameras via USB (Fig. 1, a). This 8-megapixel sensor allows you to capture, record, broadcast video and support the following video formats: 1080p (30FPS), 720p (60FPS), 640 × 480p (90FPS).

The appearance of a HAT stereo camera based on two Sony IMX219 video sensors is shown in FIG. 1, b. In Fig. 1, d another example is shown - using a web-camera with a USB connection for video surveillance purposes. This camera is mounted on a special platform, the rotation of which with the help of a servo drive can be carried out through the connector of the hardware I/O ports of the GPIO of the Raspberry Pi computer.

Main programming resources. When analyzing various methods of inputting video data, we will focus on the use of the Python programming language and available internal and external resources. The main feature of writing program codes in Python is the formation of the necessary features of the project by importing its own packages (for example, numpy, pip,pi-camera) and libraries (`Pillow`, `Matplotlib` etc.). It also significantly increases the software resources of the processing system and the connection of external libraries. In our case, this is the OpenCV library [8 − 11], the imported resources of which are relatively small for solving the assigned tasks:

```
# import the necessary packages
from imutils.video import VideoStream
from imutils.video import FPS
import numpy as np
import argparse
import imutils
import datetime
import time
import cv2
```

Connecting these (and in some cases other packages and modules) does not put a significant load on the processor and creates good prerequisites for processing data in real time.

Pay attention to the use of relatively rarely used resources (`imutils` package and `argparse` module).

`Imutils` is a set of convenient functions for simplifying basic image processing operations, such as shifting, rotating, resizing, displaying images in `Matplotlib`, using OpenCV and Python. The aka `imutils` is also used when creating a multi-threaded Python class. It provides access to the computer's built-in webcam (external camera with USB connection or pi-camera) using the OpenCV video capture functions. This allows you to significantly increase the FPS by creating a new stream, which only polls the camera for reading new frames, and the main stream processes the current frame. Next, we consider this technology in more detail. Argparse is a Python module for handling options and command line arguments with which the script is called. In our task, it is rational to use the argparse module to convert video files to regular images and save them using the OpenCV library.

Methods of input and preprocessing of video data. Next, we consider the main options for building a video input system and their preliminary processing.

1. The classical method of inputting and processing video data from a web camera in monocular

vision systems is carried out using the OpenCV function for video capture `cv2.VideoCapture (0)`. Here is a snippet of Python code for the practical implementation of this method:

```
cv2.VideoCapture(0)
while True:
    (grabbed, frame) = stream.read()
    ...
    ...
    cv2.imshow("Frame", frame)
```

However, the `VideoCapture` video capture function and the `read()` data reading method block the main stream of program code for processing video data until the frame is read from the camera device and returned to the main program. Unfortunately, this method, which is distinguished by simplicity, is often the main obstacle − it limits the ability to process the video stream in real time.

2. The Raspberry Pi hardware input method is implemented using the function of pi-camera. However, this results in a limitation of the FSP due to the action of the software load. This method does not allow the use of two or more cameras. Note also that a time delay is required to process the display of frames. The program code for this method is:

```
camera = PiCamera()
camera.framerate = 32.
Stream = camera.capture_continuous(
rawCapture,format = "bgr",use_video_port =
True)
time.sleep(2.0)
for (i,f) in enumerate(stream):
frame = f.array
    ...
    ...
cv2.imshow("Frame", frame)
```

Note that the `VideoCapture` and `PiCamera()` functions block the main stream of program code for processing video data until the frame is read from the camera device and returned to the main program. Unfortunately, this method, which is distinguished by simplicity, is often the main obstacle − it limits the ability to process the video stream in real time.

3. The method of multithreaded input of video data involves the creation of the `VideoStream()` class to transfer the reading of the frames of a web camera or usb device to another stream, completely separate from the main Python script. This allows you to continuously read frames from the I/O stream while the main stream processes the current frame. Once the main thread has finished processing the next frame, it just needs to extract the current frame from the I/O stream. This is achieved without waiting for blocking I/O operations. The program code for the VideoStream multi-threaded input class is shown below:

```
fvs=VideoStream(usePiCamera=
   =args["picamera"] > 0).start()
time.sleep(2.0)
while True:
  frame = vs.read()
  cv2.imshow("Frame",frame)
fvs.stop()
```

Creating an FPS Class. An important addition to multi-threaded video capture functionality is the definition of the FPS class. This class is used to visualize and evaluate data entry speed. It provides quantitative evidence that multithreading does increase FPS.

```
fps = FPS().start()
while fvs.more():
  ...
...
  fps.update()
fps.stop()
printfps()
```

The results of calculating the current FPS values are displayed on the Raspberry Pi monitor screen.

Creating the FR class. Frame Resolution is a very important parameter of the frame, the FPS of the video stream directly depends on the choice of it. For example, the largest possible resolution of a pi camera is FR = 1280x960. In this case, FPS reaches 30 frames per second.

To select and control the frame size, a special Resolution class was created containing the maximum frame resolution and its proportions that are most acceptable for a given project. Here is a code snippet for implementing this procedure:

```
fvs=VideoStream(usePiCamera(0)] >.start()
while fvs.more():
    resolution.frame(fvs)
  frame1=resolution.init(1,1)
  frame2=resolution.init(2,1)
  cv2.imshow("Frame1",frame1)
  cv2.imshow("Frame2",frame2)
...
...
fvs.stop()
```

Stabilization of the contrast of video data. The registration of video data usually occurs against a background of various kinds of interference with a constant dynamic change in the background of the observed scene. In this case, one of the dominant negative factors is the variability of illumination. Note that these can be both rapidly changing lighting conditions and slow changes caused by, for example, twilight. All this leads to poorly controlled variations in the contrast of the frames, and, consequently, to a deterioration in the quality of processing. Recall that usually the change in brightness values for video data is in the range of numbers 0 ÷ 255. This corresponds to the uint8 data format. To overcome these difficulties (poorly controlled changes in frame contrast depending on illumination), the authors proposed to convert the original video sequence from the RGB color space to the YUV space using the function

```
img_yuv = cv2.cvtColor
(img, cv2.COLOR_BGR2YUV).
```

The range of RGB values is [0 ÷ 255] for each component, and ranges are used for the YUV color space

- Y → [0 ÷ 255];
- U → [-112 ÷ 112];
- V → [-157 ÷ 157].

A distinctive feature of the YUV color space [5] is that it uses an explicit separation of information about brightness and color. Color is represented in the form of three components - luminance (Y) and two color difference (U and V).

After translating the RGB frame of the video sequence into the YUV color space, it performs the equalization procedure (increasing contrast) of only the Y component using the function

```
img yuv[:,:,0] =
cv2.equalizeHist(img_yuv[:,:,0]),
```

and then the frame is inversely converted from YUV format to RGB format:

```
img_output=cv2.cvtColor
(img_yuv,cv2.COLOR_YUV2BGR).
```

In this case, the color balance is maintained unchanged, since the color difference components U and V were not transformed.

Note that the transition from the RGB color space to the YUV space allows you to simply estimate the average level of brightness of the frame by the Y component. The most objective and stable indicator in our opinion is the indicator FMB (Frame medium brightness), which is calculated as

$$FMB = \frac{1}{M \cdot N} \cdot \sum_{i=0}^{N} \sum_{j=0}^{M} Y(i,j)$$

where $Y(i,j)$ – two-dimensional array of numbers defining the brightness of the pixels of the image frame size $M \times N$. This indicator is all the more useful because at low levels of frame brightness in video surveillance systems it is convenient to use the procedure of comparing it with a pre-set threshold for automatically turning on / off the scene illumination system.

Thus, the use of the equalization procedure ensures the alignment of the histogram of the brightness distribution and leads the average brightness of the image to a value (FMB = 127), regardless of what this indicator was for the original image. This makes it possible to stabilize the level of average brightness of frames, and therefore, eliminate the factors of instability of illumination of the scene, which negatively affect the further quality of video processing.

4. Experiments

To evaluate the effectiveness of the proposed methods for improving the quality of video data, a number of experimental studies were carried out using the DVRs shown in Fig. 1, and the specialized program "Test programm". It is written in Python using the appropriate resources of the OpenCV library. This program is based on the use of input and preprocessing algorithms described in our work, received using web or pi-cameras of the original video data into the Raspberry Pi microcomputer. A generalized UML diagram of the operation of this program is shown in Fig. 2.

Given the complex nature of the research, the program included the possibility of organizing various versions of the video capture and video input algorithm (classic with the `cv2.VideoCapture(0)` function, accelerated by the hardware of the Raspberry Pi computer, as well as the fastest multithreaded input using a specially created class `VideoStream`). In addition, the program can pre-set the required screen resolution, as well as in conditions of insufficient scene illumination - the mode of stabilizing the brightness of the frame.

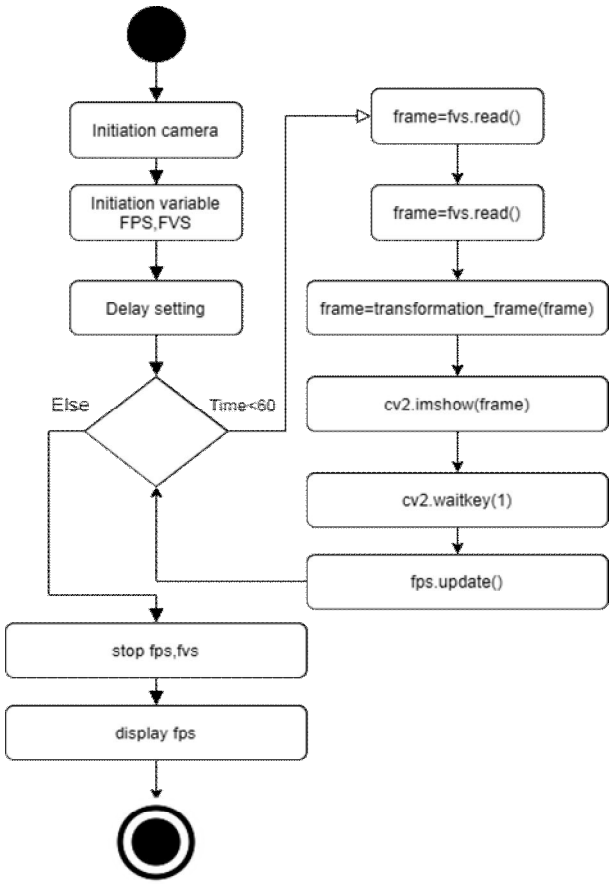


Fig. 2. UML diagram of program activities

The «Test program» also provides a number of service options. First of all, it is the ability to display the input video stream on the screen using the `cv2.imshow (frame)` function.

The `transformation_frame(frame)` function allows you to change the format of frames and their sizes, and, if necessary, set the mode for stabilizing the brightness of the frame. In addition, it is envisaged to apply special inscriptions in the frame field with information about the current FPS value of the video stream, frame sizes and stabilization of the brightness of the scene image.

Note that to establish the operating mode of the `VideoStream` class (capture the required number of frames of the input video stream), a time interval of 1 s was selected, and to obtain stable and reliable FPS estimates, the averaging interval was 10 s.

5. Results

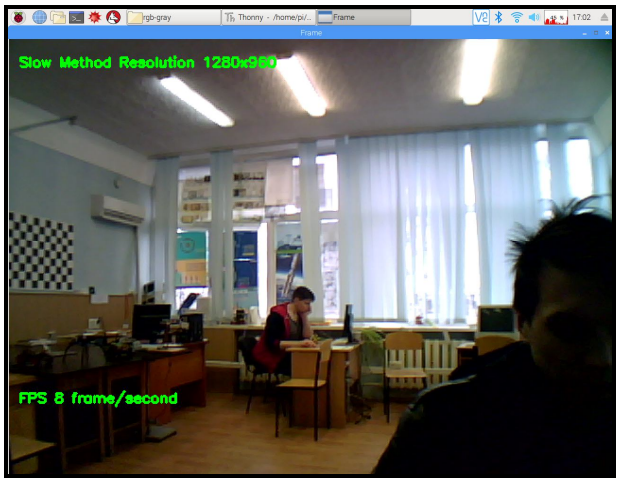
When planning the experiment, the task was to identify and study the relationship between the nature of changes in the FPS performance indicator and the resolution of the FR frames of the video stream entered into the Raspberry Pi microcomputer for various data input and preprocessing algorithms.

In addition, the task was to study the degree of influence on the speed of the procedure for entering additional load on the computer processor due to the algorithm for stabilizing the brightness of the video stream frames. Based on these data, it was necessary to formulate recommendations on the optimal

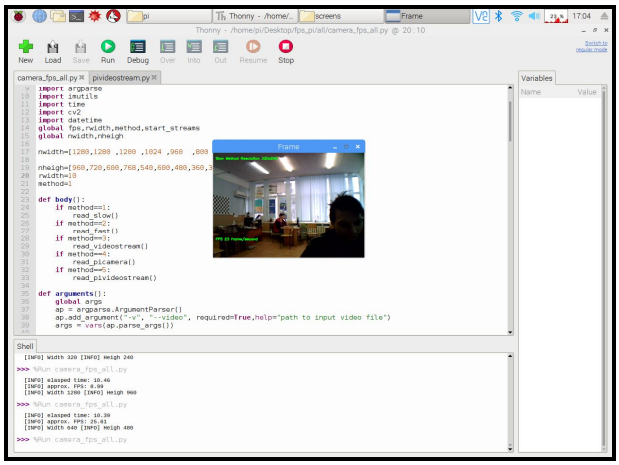
configuration of software for input and preliminary processing of received video data.

No less important is the task of convenient and clear visualization of the obtained experimental data.

In Fig. 3 two options for the formation of frames of a video stream with different resolutions and different aspect ratios of the frame are shown. In the first case, at a resolution of 1280 × 960 (Fig. 3, a), the frame almost completely covers the area of the working window. The inscription on the bottom left of the frame displays the current FPS value. When viewing a video stream, the FPS value changes in real time. The inscription in the upper left corner shows which video input method is used in this case (Slow method and Fast method appropriately).



a



b

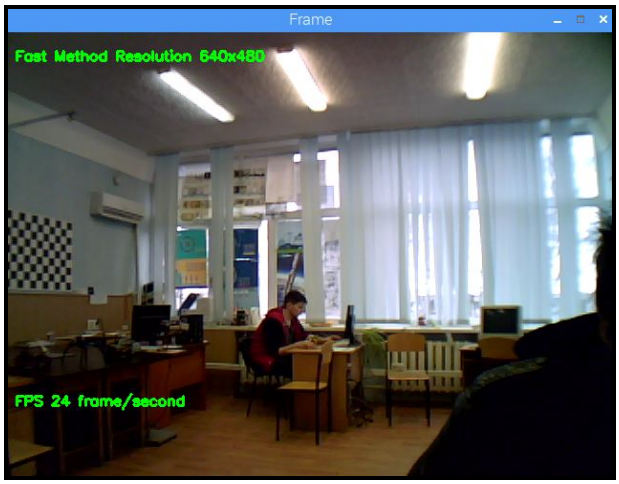
Fig. 3. Video stream frames formed with different resolutions: a – 1280 × 960; b – 320 × 240

In Fig. 3, b, a frame of a video stream with a resolution of 320 × 240 is displayed on a computer screen. Such permission is usually used in robotics. Due to the small area of this frame, you can easily see the screen fragment of the program code, and in the lower left corner of the command window the current calculated FPS values. Here is a fragment of such data:

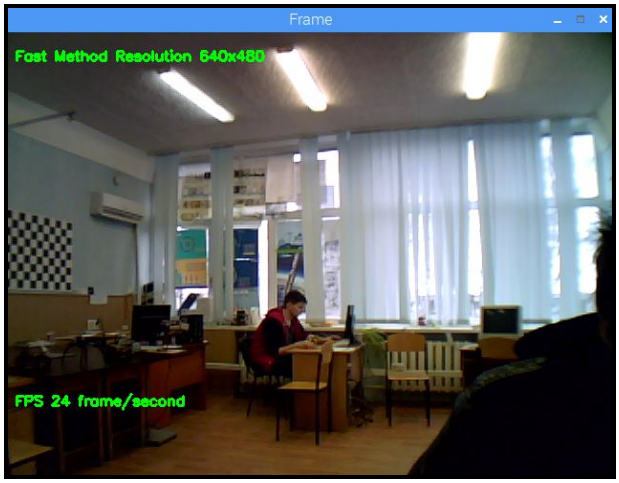
```
[INFO] elasped time: 10.67
[INFO] approx. FPS: 8.53
[INFO] Width 1280
[INFO] Heigh 960
```

When comparing the FPS in the examples shown in Fig. 3 shows that the transition from low resolution to large leads to a significant decrease in input speed − FPS decreases from 23 frame/second to 8. Potentially possible (passport FPS) is 30 frame / second).

Fig. 4 shows two 640x480 video frames inserted into a Raspberry Pi computer using the multi-stream `VideoStream` method. Due to insufficient lighting of the scene (Fig. 4a), the procedure for contrasting (equalizing) frames was introduced into the preliminary processing of the video stream (Fig. 4b). It was previously mentioned that this requires translating the RGB frame of the video sequence into the YUV color space, followed by the equalization of the luminance component Y and returning it to the RGB color space. It is clear that this creates an additional load on the processor and negatively affects the input speed − FPS decreases from 24 to 19 frame/second. However, the image quality of the frame is increasing.



a



b

Fig. 4. Stabilization of the brightness of the frames of the video stream

Fig. 5 shows the dependences of the rate of input of FPS video data using a web-camera depending on the selected resolution of the frame. Here we present possible standard options for the names of the formats and the resolution of the frames, which are recommended to be established when creating and

organizing the input of the video stream, and also determine the aspect ratio of the frame on the screen:

QVGA – Frame Resolution = 320×240 → (4 : 3);
HVGA – Frame Resolution = 640×240 → (8 : 3);
HVGA_1 – Frame Resolution = 320×480 → (2 : 3);
nHD – Frame Resolution = 640×360 → (16 :93);
VGA – Frame Resolution = 640×480 → (4 : 3);
SVGA – Frame Resolution = 800×600 → (4 : 3);
qHD – Frame Resolution = 960×540 → (16 : 9);
XGA – Frame Resolution = 1024×768 → (4 : 3);
WXVGA – Frame Resolution = 1200×600 → (2 : 1);
HD 720p – Frame Resolution = 1280×720 → (16 : 9);
Full HD – Frame Resolution = 1920x1080 → (16 : 9).

The analysis was carried out for formats with a relatively low resolution (from QVGA – 320 × 240 to Full HD - 1920 × 1080). A comparative analysis of the effectiveness of the two input methods (classical and multithreaded) showed the undoubted advantage of the latter.

This is especially true with high resolution frames. So, for example, at HD 720p resolution, the FPS of the fast method is 17.60, which is about twice as high as the slow method (FPS = 9.76). Such data are undoubtedly useful for practical use.

As in the previous study, we studied data on the characteristics of the input rate of a video stream using various methods (slow and fast). But at the same time, for video surveillance, a specialized pi-camera was used in conjunction with the Raspberry Pi microcomputer. The dependences of FPS values on the resolution of video frames are shown in Fig. 6. Obviously, the use of the method of multithreaded input (Fast method) in this case significantly exceeds the capabilities of the slow method (Slow method).

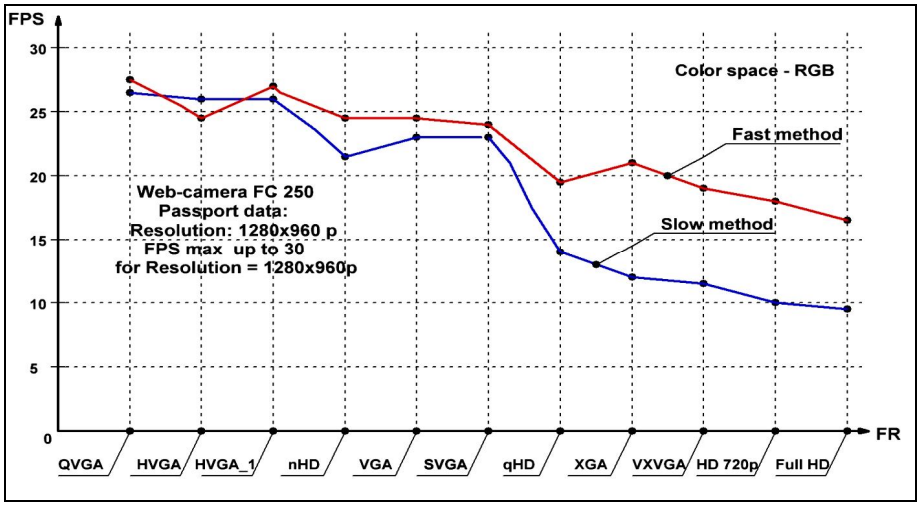


Fig. 5. Indicators of video input speed for various resolutions of video stream frames

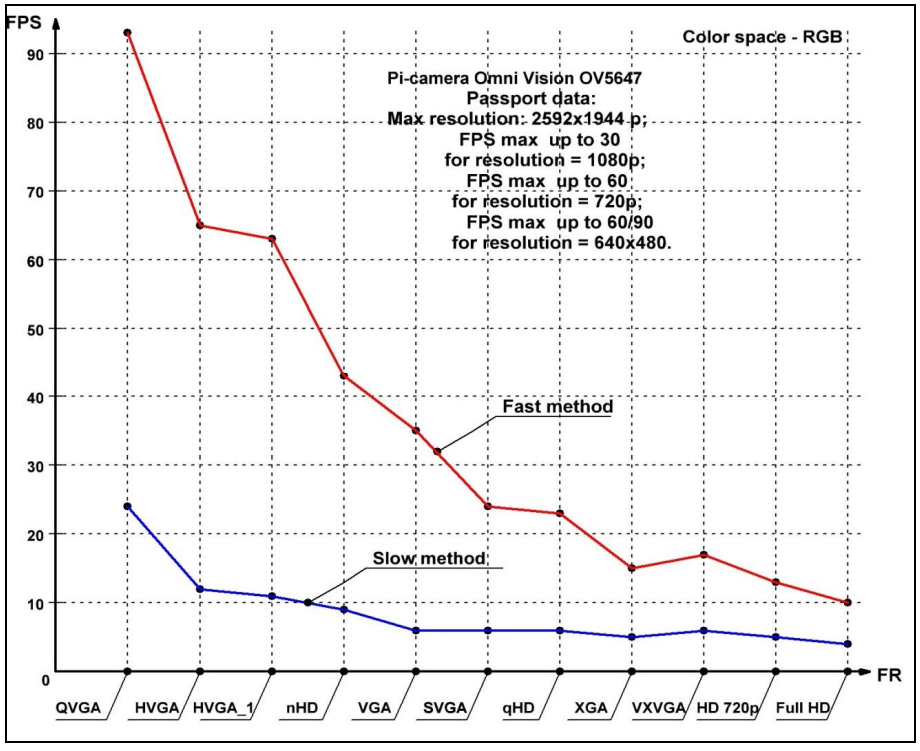


Fig. 6. Indicators of video input speed for pi-cameras

This is especially noticeable at low resolution values of the frame. For example, with QVGA resolution (320 × 240), the data input speed increases by 3.5 times and approaches the potential capabilities of the pi-camera in terms of speed. Based on these data, we can conclude that in practically important applications it is advisable to use pi-cameras and the Fast method for this task, based on the use of the multi-stream class VideoStream.

The given examples, of course, do not cover all the features of the problem under consideration. You can explore the effect on input speed and frame quality and other factors. However, due to limited publication volumes, we recommend that you do this yourself.

Conclusions

A new integrated approach to the problem of improving the quality of video processing in modern mobile VS based on Raspberry Pi microcomputers is proposed. Jointly optimized indicators of the speed of capture and processing of video data, the resolution of the video frame and the stability indicators of the brightness of the frame regardless of the lighting conditions of the scene. The effectiveness of the proposed video processing algorithms and methods for synthesizing program codes was studied experimentally.

The scientific novelty. The analysis showed a significant improvement in the quality of the recorded video stream using the proposed algorithms. For the first time, new methods for processing video data were obtained on the basis of objective optimization criteria using modern programming tools.

The practical significance. In the future, the results can be used to create various projects based on mobile VS. Based on the results of the research, it is easy to create the necessary high-quality software.

Prospects for further research. The authors consider it most appropriate to complete work in this area to conduct detailed studies of the characteristics of input and preliminary processing of video data from two cameras to form a modern mobile stereo vision system on the basis of the Raspberry Pi computer.

References

1. (2020), *The official documentation site for working with a Raspberry Pi computer*, available to: https://www.raspberrypi.org;
2. (2020), *The official site of the Python language*, available to: http://python.org/;
3. (2020), *The official website of the developers of the OpenCV library*, available to: http://opencv.org;
4. (2019), *Synchronized Dual Camera for Raspberry Pi 4*., Published by Lee Jackson on September 16, 2019, available to: https://www.arducam.com/dual-camera-hat-synchronize-stereo-pi-raspberry/;
5. Fedorov, D.Ju. (2016), *Fundamentals of programming using the Python language as an example*, Simvol-Pljus, SPb., 176 p.
6. Lutc M. (2011), *Python Programming*, Simvol-Pljus, SPb., 992 p.
7. Protasov, S.I., Kurgalin, S.D. and Kryloveckij A.A. (2011), "Using webcams as a source of stereo pair flow", *Vestnik VGU, Serija: sistemnyj analiz i informacionnye tehnologii*, Voronezh, No. 2
8. Linda G., Shapiro and George C., Stockman (2001), *Computer Vision*, Prentice Hall, 580 p.
9. Joseph, Howse and Joe Minichino (2015), *Learning OpenCV 3 Computer Vision with Python*, Second Edition, Packt Publishing, ISBN: 978-1-78528-977-4.
10. Saurabh, Kapur (2017), *Computer Vision with Python 3*, Packt Publishing, ISBN: 978-1-78829-976-3.
11. Prateek, Joshi (2015), *OpenCV with Python By Example*, Packt Publishing, ISBN: 978-1-78528-393-2.
12. Dergachov, K., Krasnov, L., Cheliadin, O. and Zymovin, A. (2018), "Adaptive algorithms of face detection and effectiveness assessment of their use", *Advanced Information Systems*, Vol. 2, No. 3, DOI: https://doi.org/10.20998/2522-9052.2018.3.02.
13. Dergachov, K., Krasnov, L., Cheliadin, O. and Plakhotnyi, O. (2019), "Web-cameras stereo pairs color correction method and its practical implementation", *Advanced Information Systems*, Vol. 3, No. 1, pp. 29-42, DOI: https://doi.org/10.20998/2522-9052.2019.1.06

Received (Надійшла) 24.03.2020
Accepted for publication (Прийнята до друку) 20.05.2020

Відомості про авторів / About the Authors

Дергачов Костянтин Юрійович – кандидат технічних наук, доцент, завідувач кафедри систем управління літальними апаратами, Національний аерокосмічний університет імені М.Є. Жуковського «ХАІ», Харків, Україна;
Kostiantyn Dergachov – Candidate of Technical Science, Associate Professor, Head of Aircraft Control Systems Department, National Aviation University "Kharkiv Aviation Institute", Kharkiv, Ukraine;
e-mail: kosv.v@ukr.ua; ORCID ID: http://orcid.org/0000-0002-6939-3100.

Краснов Леонід Олександрович – кандидат технічних наук, старший науковий співробітник, доцент кафедри систем управління літальними апаратами, Національний аерокосмічний університет імені М.Є. Жуковського «ХАІ», Харків;
Leonid Krasnov – Candidate of Technical Science, Senior Research, Associate Professor of Aircraft Control Systems Department, National Aviation University "Kharkiv Aviation Institute", Kharkiv, Ukraine;
e-mail: leonid.krasnov.1947@gmail.com; ORCID ID: http://orcid.org/0000-0003-2607-8423.

Челядін Олександр Олександрович – аспірант кафедри систем управління літальними апаратами, Національний аерокосмічний університет імені М.Є. Жуковського «ХАІ», Харків, Україна;
Oleksandr Cheliadin – Doctoral Student of Aircraft Control Systems Department, National Aviation University "Kharkiv Aviation Institute", Kharkiv, Ukraine;
e-mail: lorianua33@gmail.com; ORCID ID: http://orcid.org/0000-0002-1201-6240.

Казатинський Роман Євгенович – магістрант кафедри систем управління літальними апаратами, Національний аерокосмічний університет імені М.Є. Жуковського «ХАІ», Харків, Україна;

Roman Kazatinskij – Postgraduate Student of Aircraft Control Systems Department, National Aviation University "Kharkiv Aviation Institute", Kharkiv, Ukraine;
e-mail: kazatinskij.roman@gmail.com; ORCID ID: http://orcid.org/0000-0002-7098-715X.

Розробка нових методів і засобів підвищення якості відеоданих в мобільних системах технічного зору

К. Ю. Дергачов, Л. О. Краснов, О. О. Челядін, Р. Є. Казатинський

Анотація. Предмет вивчення. У статті запропоновано нові методи введення і попередньої обробки відеоданих для web- і спеціалізованих pi-камер в системах монокулярного і стереозору на базі мікрокомп'ютерів Raspberry Pi для підвищення якості роботи сучасних мобільних систем технічного зору. Такий підхід завжди актуальний, оскільки проектування сучасних систем технічного зору постійно вимагає нових нетривіальних апаратних, алгоритмічних і програмних рішень. **Мети.** Метою є порівняльний аналіз показників якості відомих методів введення і попередньої обробки відеоданих в системах технічного зору та розробка нових методів і робочих алгоритмів, що забезпечують більшу швидкодію при читанні відеоданих, необхідний дозвіл кадрів і незалежність яскравості кадру від змін освітленості сцени. **Методи**. В роботі сформульовано комплексний критерій підвищення якості введення в мікрокомп'ютер Raspberry Pi і попередньої обробки відеоданих. На підставі прийнятих показників якості (швидкості введення відеоданих, роздільної здатності та показників стабільності середньої яскравості поточних кадрів прийнятого відеопотоку) синтезовані алгоритми введення і попередньої обробки відеоданих, що задовольняють заданим вимогам. Це дозволяє для кожного проекту знайти оптимальний метод обробки відеоданих і подолати протиріччя зменшення швидкості введення через необхідність збільшення роздільної здатності відеокадрів. Створена універсальна програма введення і попередньої обробки цих даних дозволила отримати кількісні оцінки ефективності розроблених алгоритмів і сформулювати рекомендації щодо їх подальшого використання. Все це дозволяє істотно підвищити ефективність використання мікрокомп'ютерів Raspberry Pi в сучасних мобільних системах технічного зору. **Результати.** Отримані результати покладені в основу при створенні універсального програмного продукту для швидкісного введення (в режимі реального часу) і попередньої обробки відеоданих в системах детектування і розпізнавання осіб, а також систем стереозору. **Висновки**. Проведені експериментальні дослідження підтвердили працездатність і ефективність запропонованих методів і алгоритмів швидкісного введення відеоданих з різними значеннями роздільної здатності кадру і можливістю адаптивної регулювання його яскравості. На базі створених методів і алгоритмів запропоновані різні варіанти їх програмної реалізації. Це дозволяє рекомендувати отримані результати для практичного використання. Перспективи подальших досліджень передбачають розширення вектора критеріїв оцінки якості та ознак для оптимізації відеоданих, а також створення нових алгоритмів і різних варіантів програм на їх основі.

Ключові слова: мікрокомп'ютер Raspberry Pi; алгоритми і програмні коди введення і попередньої обробки відеоданих; функції бібліотеки OpenCV.

Разработка новых методов и средств повышения качества видеоданных в мобильных системах технического зрения

К. Ю. Дергачёв, Л. А. Краснов, А. А. Челядин, Р. Е. Казатинский

Аннотация. Предмет изучения. В статье предложены новые методы ввода и предварительной обработки видеоданных для web- и специализированных pi-камер в системах монокулярного и стереозрения на базе микрокомпьютеров Raspberry Pi для повышения качества работы современных мобильных систем технического зрения. Такой подход всегда актуален, поскольку проектирование современных систем технического зрения постоянно требует новых нетривиальных аппаратных, алгоритмических и программных решений. **Цель.** Целью является сопоставительный анализ показателей качества известных методов ввода и предварительной обработки видеоданных в системах технического зрения и разработка новых методов и рабочих алгоритмов, обеспечивающих большее быстродействие при чтении видеоданных, необходимое разрешение кадров и независимость яркости кадра от изменений освещенности сцены. **Методы.** В работе сформулирован комплексный критерий повышения качества ввода в микрокомпьютер Raspberry Pi и предварительной обработки видеоданных. На основании принятых показателей качества (скорости ввода видеоданных, разрешающей способности и показателей стабильности средней яркости текущих кадров принимаемого видеопотока) синтезированы алгоритмы ввода и предварительной обработки видеоданных, удовлетворяющие заданным требованиям. Это позволяет для каждого проекта найти оптимальный метод обработки видеоданных и преодолеть противоречие уменьшения скорости ввода из-за необходимости увеличения разрешающей способности видеокадров. Созданная универсальная программа ввода и предварительной обработки этих данных позволила получить количественные оценки эффективности разработанных алгоритмов и сформулировать рекомендации по их дальнейшему использованию. Все это позволяет существенно повысить эффективность использования микрокомпьютеров Raspberry Pi в современных мобильных системах технического зрения. **Результаты.** Полученные результаты положены в основу при создании универсального программного продукта для скоростного ввода (в режиме реального времени) и предварительной обработки видеоданных в системах детектирования и распознавания лиц, а также систем стереозрения. **Выводы.** Проведенные экспериментальные исследования подтвердили работоспособность и эффективность предложенных методов и алгоритмов скоростного ввода видеоданных с разными значениями разрешающей способности кадра и возможностью адаптивной регулировки его яркости. На базе созданных методов и алгоритмов предложены различные варианты их программной реализации. Это позволяет рекомендовать полученные результаты для практического использования. Перспективы дальнейших исследований предполагают расширение вектора критериев оценки качества и признаков для оптимизации видеоданных, а также создание новых алгоритмов и различных вариантов программ на их основе.

Ключевые слова: микрокомпьютер Raspberry Pi; алгоритмы и программные коды ввода и предварительной обработки видеоданных; функции библиотеки OpenCV.