ADVANCES IN CYBER-PHYSICAL SYSTEMS

Vol.7, Num.2, 2022

SOFTWARE SYSTEM FOR MOTION DETECTION AND TRACKING

Bohdan Tsiunyk, Oleksandr Muliarevych

Lviv Polytechnic National University, 12, Bandera Str., Lviv, 79013, Ukraine.
Authors' e-mails*: bohdan.tsiunyk.mkiks.2021@lpnu.ua; oleksandr.v.muliarevych@lpnu.ua;*

https://doi.org/10.23939/acps2022.__.___
Submitted on
© Tsiunyk B., Muliarevych O., 2022

Abstract: **The goal of the work is to develop a software system for motion detection of and tracking object. It consists of the user interface which is presented as a desktop application. This paper describes the process of developing a desktop software system stage using the latest technologies which will be relevant and easy to maintain in future development and upgrade. The technologies used in the development process, the systems and modules which were integrated into the project, the main approaches to software development, as well as an explanation of why this particular stack of technologies was preferred for the implementation of this software system have been described. To make sure that the developed desktop application meets common optimization requirements it has been tested for resource usage.**

Index Terms: **desktop architecture, development, Open CV, Python.**

I. INTRODUCTION

Today, more than ever before, our community of people must ensure its safety, the safety of its settlement, because the health and even the lives of its residents depend on it. This is one of the most important components of everyone's life because without the rest it makes no sense.

The relevance of the topic of home security lies in the implementation of certain security steps, electronics, equipment, and other means. To inform and protect everyone who risks becoming a participant or victim of an emergency, a home invasion, or theft in their home, when any threat to the health or life of people is detected. And to improve this process, so that ordinary people take a direct part in protecting security themselves and improving the well-being of their neighborhood, it was planned to create a system for monitoring the movement of surveillance cameras. This is the main goal, the primary task, which is solved in the software system of this work.

A motion detection system [1] is a collection of software-integrated tools and components, as well as hardware, that meets the needs of a specific group of users of a given software system and is stored in a common user database. Because of their flexibility in implementation, today's detection software systems can meet any user's needs.

According to the Automated Imaging Association (AIA), machine vision refers to all industrial and non-industrial applications in which a combination of hardware and software guide devices in the execution of their functions based on image capture and processing. [2]. Though industrial computer vision employs many of the same algorithms and approaches as academic/educational and governmental/military computer vision applications, the constraints are distinct. When compared to academic/educational vision systems, industrial vision systems require more robustness, reliability, and stability and typically cost much less than those used in governmental/military applications. As a result, industrial machine vision implies low cost, acceptable accuracy, high robustness, high reliability, and mechanical and thermal stability. Machine vision systems are reliant on [3] digital sensors protected inside industrial cameras with specialized optics to acquire images so that computer hardware and software can process, analyze, and measure various characteristics for decision-making. As an example, consider a fill-level inspection system at a brewery.

For years we've been reaping the small-scale rewards of Artificial Intelligence (AI) research [4]. Checks can now be read into an ATM without human intervention, auto-focusing cameras can recognize faces in a photograph, social media can auto-tag friends based on their faces, and more. As the world's access to computing power grows, so will AI's democratization. Every day, new real-world AI use cases are discovered, which is why metro star is constantly expanding our capabilities. The aim is to provide our partners with access to the same efficiencies that the rest of the world enjoys in their daily lives. In this blog, we will discuss some of the ways we have begun to use AI to assist our Defense clients. As an example, consider coffee. It's disappointing to brew an entire pot of coffee only to have your cup taste bitter. By allowing us to focus on the most important aspects of coffee or data in our world, AI can make every cup the perfect cup. We can now use machines to parse through hours of video data to highlight specific events of interest, translate foreign languages in seconds, and even understand commands to perform specific tasks when called upon. While we are still a long way from true generalized AI, modern Machine Learning (ML) has demonstrated that this technology is capable of solving specific, time-consuming tasks.

In recent years, we have witnessed unprecedented advancements in the automatic analysis of visual data by computer algorithms, due to a series of factors that unleashed the potential of convolutional neural networks, The biologically inspired design of such models may explain part of their success: while the classic artificial neuron may only be an extreme simplification of the biological neuron, the increasing representational complexity learned by CNN may be more faithful to the layered structure of the lower areas of the human visual cortex. However, the road to achieving a level of artificial emulation of the human visual system high enough to interpret an environment in the same way that humans do remains long: while we can identify low- to high-level visual patterns from images and videos, artificial models largely miss the human capability [5] to make sense of this information, recognize semantic patterns, correlate to memory and experience, and so on. Furthermore, even the neural computational models we use are only loosely based on biological structures and connections: human visual analysis, for example, transmits information across cortical brain regions in both feedforward and feedback patterns.

Computer vision systems that provide vision and simulation via computers are used in many fields in daily life. The vision action can be performed by single, stereo, or multiple camera systems [6]. Using doubled or multi-camera systems, stereo systems can realize computer vision events. Stereo vision systems are visualization techniques that allow point coordinates to be reproduced in three dimensions on images captured by two different cameras. Stereo vision systems are typically dual in nature and based on multiple visions. These vision systems are used in a variety of applications, including portable autonomous robotic systems, 3D measurements, object tracking, the film industry, augmented reality, and object recognition [7].

Python is one of the few languages that can be both simple and powerful. It can be surprising at how easy it is to focus on the solution to the problem rather than the syntax and structure of the language you're programming in. Python's official introduction is: Python is a powerful programming language that is simple to learn. It has high-level data structures that are efficient and a simple but effective approach to object-oriented programming. Python's elegant syntax, dynamic typing, and interpreted nature make it an ideal language for scripting and rapid application development across a wide range of platforms.

Python is a straightforward and minimalistic programming language. Reading a good Python program is similar to reading English, albeit very strict English. Python's pseudo-code nature is one of its greatest strengths. It allows to focus on the problem solution rather than the language itself.

As you will see, Python is extremely easy to get started with. Python has an extraordinarily simple syntax, as it has been already mentioned. Python is an example of a FLOSS (Free/Libré and Open-Source Software). To put it simply, you can freely distribute copies of this software, read its source code, make changes to it, and incorporate parts of it into new free programs. FLOSS is based on the idea of a knowledge-sharing community. This is one of the reasons Python is so good; it was created and is constantly improved by a community of people who simply want to see a better Python. When writing programs in Python, you never have to worry about low-level details like managing the memory used by your program, etc. Because Python is open-source, it has been ported to (i.e., changed to work on) many platforms. Python can be used Python on GNU/Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE, and PocketPC. You can even use a platform to create games for your computer and iPhone, iPad, and Android [8].

A program written in a compiled language, such as C or C+, is converted from the source language, C or C+, into a language that your computer understands (binary code, i.e. 0s and 1s), using a compiler with various flags and options. The linker/loader software copies the program from the hard disk to memory and starts it when you run it. Python, on the other hand, does not require binary compilation. You just run the program directly from the source code. Internally, Python converts the source code into an intermediate form called bytecodes and then translates this into the native language of your computer, and then runs it. All of this makes using Python much easier because you don't have to worry about compiling the program, ensuring that the proper libraries are linked and loaded, and so on. This also makes your Python programs much more portable, as you can simply copy your Python program and run it on another computer!

Python supports both procedure-oriented and object-oriented programming. The program in procedure-oriented languages is built around procedures or functions, which are nothing more than reusable pieces of code. The program in object-oriented languages is built around objects that combine data and functionality. When compared to large languages like C++ or Java, Python has a very powerful but simple way of doing OOP.

If you need a critical piece of code to run very fast or want to have some piece of algorithm not be open, you can code that part of your program in C or C\++ and then use it from your Python program.

The Python Standard Library is enormous. It can assist you with regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, FTP, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI (graphical user interfaces), and other system-specific tasks. Remember that all of this is available wherever Python is installed. Python's Batteries Included philosophy is known as this. Python is a fascinating and powerful programming language. It has the right combination of performance and features to make writing Python programs both enjoyable and simple.

Because it can reconstruct local or full-field 3D profile information in a convenient, fast, and easy-to-implement manner, vision-based 3D technology has a wide range of applications in many fields, including 3D shape measurement, visual inspection, medical assistance, robot visual guidance, and so on. It is the foundation for calibrating the parameters of industrial cameras in order to obtain internal or external parameters for subsequent measurement or detection tasks in the application of such vision-based technologies. The precision of measurement is directly determined by the accuracy of camera calibration, especially in industrial precision measurement. Direct linear transformation (DLT), the Tsai method, and Zhang's plane calibration method are examples of common camera calibration methods. Using 3D space points, the DLT method can calibrate the basic camera perspective model. However, the DLT method typically ignores lens distortion. Based on the radial constraint, Tsai's method employs a two-step calibration strategy. To begin, this method employs a radial alignment constraint to compute the closed solution of the camera's external parameters. The internal parameters must then be solved. If the camera lens is not distorted, the closed solution of internal parameters can be directly obtained. If there is radial distortion, nonlinear iterative optimization is used to solve all of the remaining parameters, including the radial distortion coefficient. The DLT method and the Tsai method typically impose strict constraints on the spatial pose of the calibration object, whereas Zhang's method imposes no constraints on the spatial pose of the calibration object. A flexible calibration technique allows the camera or calibration target to be placed arbitrarily during the calibration process. Zhang's method first employs the undistorted model to solve the closed solution of the camera's internal and external parameters. The camera's internal and external parameters, including radial and tangential distortion coefficients, are then further optimized using this closed solution as the initial value and the minimization of the sum of squares of reprojection errors as the objective function. Zhang's method has currently become the most popular camera calibration method due to its high flexibility and adaptability [9].

OpenCV (Open-Source Computer Vision Library) is a free and open-source software library for computer vision and machine learning. OpenCV was created to provide a common infrastructure for computer vision applications and to speed up the incorporation of machine perception into commercial products. Because OpenCV is an Apache 2 licensed product, it is simple for businesses to use and modify the code. The library contains over 2500 optimized algorithms, including a comprehensive set of classic and cutting-edge computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, and stitch images together to produce a high-resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken with flash, follow eye movements, recognize scenery, and establish markers to override the algorithm. The OpenCV user community numbers over 47 thousand people, and the estimated number of downloads exceeds 18 million. Companies, research groups, and government agencies all make extensive use of the library. Most CV applications require image input. Most generate images as output. A camera as an input source and a window as an output destination may be required for an interactive CV application. Image files, video files, and raw bytes are all possible sources and destinations. Raw bytes, for example, could be transmitted over a network connection or generated by an algorithm if we incorporate procedural graphics into our application. Let's take a look at each of these options [10].

The human brain divides the vision signal into numerous channels that feed various types of information into your brain. Your brain has an attention system that identifies important parts of an image to examine while suppressing examination of other areas in a task-dependent manner. There is a massive amount of feedback in the visual stream that is currently unknown. Associative inputs from muscle control sensors and all other senses are common, allowing the brain to draw on cross-associations formed over years of living in the world. The feedback loops in the brain return to all stages of processing, including the hardware sensors (the eyes), which mechanically control lighting via the iris and tune reception on the retina's surface. However, in a machine vision system, a computer receives a grid of numbers from the camera or disk and that's it. There is no built-in pattern recognition, automatic focus and aperture control, or cross-associations with years of experience. Most vision systems are still fairly primitive. The computer "sees" only a grid of numbers. Any given number within that grid has a high noise component and thus provides little information on its own, but this grid of numbers is all the computer "sees." Our task now is to convert this noisy grid of numbers into perception: “side mirror”.

The last but not least step in application development is testing. It enables us to detect and fix common bugs early in the development process. Typically, testers perform this type of work. Testers are those who are in charge of application testing. They should simulate all possible user workflow scenarios. It allows to go through all of the functions that the application offers, testing them all.

There are 2 (two) types of testing: manual and automation. The advantage of automation tests is that they are run independently from human participation so they can be run even at night, on weekends, in one word – all the time without exceptions. But the disadvantage is that writing automation tests takes pretty a lot of time and development resources.

Manual testing is another type of testing in which all of the testing process is done by humans. Because it is more flexible, it is more reliable and safer than automation testing. During the development process, manual testing is frequently performed. When developers implement a piece of functionality, it can be manually tested right away.

The performance testing of the developed application should be performed with special instruments. One such instrument is Desktop Studio Profiler. It allows monitoring of how the Desktop application uses the CPU resources, RAM, and network and how it affects battery conditions [11]. The main characteristic of such a system is the amount of memory, which should be from 200 MB to 400 MB.

II. PROBLEM STATEMENT

Today it is shown that many solutions have been developed that will allow consumers to monitor and track the state of the external environment using a camera in their home, neighborhood, or a separate place where moving objects or people should not enter, such as a safe in a bank or private sector. In case of violation of the visibility zone, the system should notify the user in case of attack, movement, or other actions. However, there is still no generally accepted solution that would automate this entire process.

The main feature of the system described in this work is that the user needs only one device with a built-in camera to use this functionality in the routine that will be described in this article. In the event of situations when something goes wrong, such as an attack on the private sector, the user receives appropriate messages and notifications.

The functionality described above does not exhaust the capabilities of this system for monitoring and detecting moving objects. Of course, safety and health are among the main, if not the most important, factors of human well-being, but not only emergencies can be of interest to today's society.

The feature of the existing software system and the difference from analogs is the fully corrected and instant response of moving objects and notification of movement messages to the user. This means that the user can relax as long as the safety and security system is activated.

This system is super-fast and processes images using a minimum of resources for this.

III. PURPOSE OF THE WORK

The purpose of this work is to create a software complex for motion detection and situation tracking in the private sector or a surveillance camera using the latest technologies for the development of this type of system, namely the user part in the form of a desktop application using Open CV and Python for cross-platform.

Another important requirement for the system is program optimization. As this kind of computation and user interface uses video stream rendering, there are often problems with RAM usage and CPU load. The current program must meet the requirements for standard desktop programs. The amount of RAM for standard desktop programs ranges from 200 to 500 MB. The calculation time is usually 0.1ms for the detection of moving objects.

The result application should be a working system for motion detection and tracking objects that meets all the requirements described above and allows users to monitor the situation in a defined area where security surveillance is conducted.

IV. ALGORITHM FOR CONVERTING VIDEO THREAD INTO CV OBJECT THE SOFTWARE SYSTEM

The main requirement and idea for the implementation of this software system was the use of modern technologies, which are preferred today for the implementation of such types of systems. The programming language used to develop the entire system is Python, and the development environment is Pycharm IDE.

As a result of studying pros and cons, it was decided to use the following technology stack for desktop implementation: Python, Open CV, and Pycharm.

By connecting a surveillance device, the user receives a video stream. Next, the Open CV algorithm converts the video format of the stream into CV video capture and CV frame settings. Debugging the format and framerate, we get the out object for further use. Fig.1.

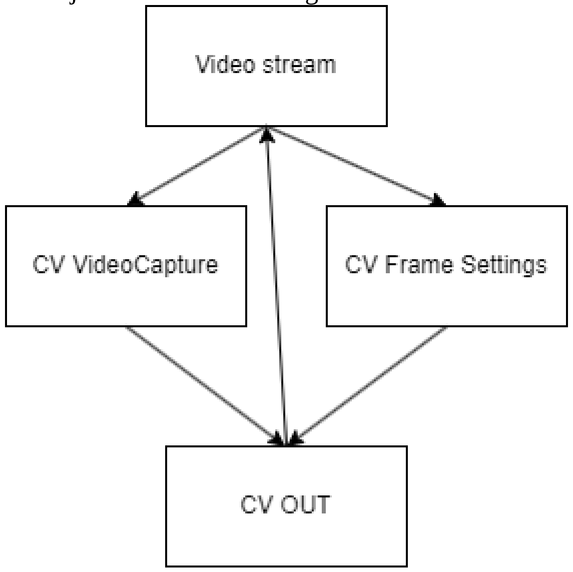


Fig. 1. Video thread converting scheme

Last but not least part of application development is its testing. Moreover, the testing part is one of the most important in the development process as it allows us to intercept common bugs and fix them right in the development stage. The performance test is used.

V. DEVELOPMENT OF THE DESKTOP APPLICATION AND USER INTERFACE

Even though web and mobile applications appear to have taken over the software development market, there's still demand for traditional graphical user interface (GUI) desktop applications.

Creating a desktop interface consists of the following stages: creating graphical user interfaces with Python, connecting the user's events on the app's GUI with the app's logic, organizing the proper project layout, create a fully functional GUI application.

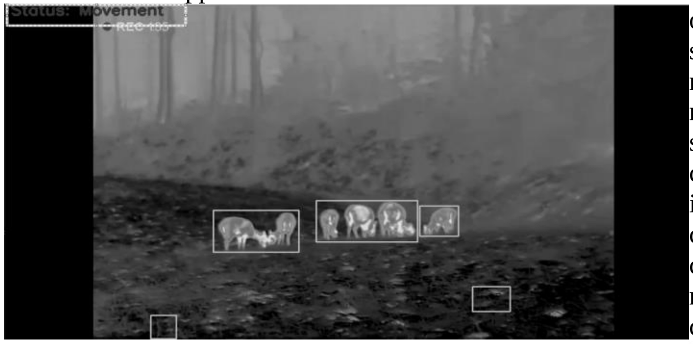


Fig. 2. Visualization of movement detection status

The above code creates a Window, the visible code is very important, without that the UI is going to run but will be invisible, with width and height as specified, with a title of “feed”. And a Text that is centered in the parent (which happens to be the window), the text displayed is “Hello World”, a pixel size of 24px.

Formally, we had a property string that received our current time string from Python, now we create a property to receive the object from python. There are not that many types. UI converts Python base types into a bool, int, double, string, list, object, and var and can handle every Python type.

First of all, the user interface should be as simple and user-friendly as possible without displaying the processes that take place under the hood of the entire system. The user interface should display the monitoring status of the external area, that is, the video stream and moving objects should be illuminated. Also, the system should display the state of the system, in the case of motion detection, a motion message should be displayed on a part of the screen. (see Fig. 2.)

The data parameter can be used to include data files in the app folder. It's a list of tuples and the tuple always has two items, the target path, which we will be including, and the destination path, which should be stored in the application’s folder. The destination path must be relative. It is placed right there with the app’s executables, making it an empty string (‘’), to be in a nested folder within the application’s folder. The object CV out should be placed in the same folder where there is the whole software system.

The result of the working system for motion detection and tracking moving objects is presented in Fig. 3.

VI. ALGORITHM FOR CREATING A SOFTWARE SYSTEM

For software systems, models can be learned from behavioral traces, available specifications, knowledge of experts, and other such sources. Software models help to steer testing and model checking of software systems. The model inference techniques extract structural and design information of a software system and present it as a formal model. This chapter briefly discusses the passive model inference and goes on to present the active model inference of software systems using the algorithm. This algorithm switches between model inference and testing phases. In the model inference phase, it asks membership queries and records answers in a table to conjecture a model of a software system under inference. In the testing phase, it compares a conjectured model with the system under inference. If a test for a conjectured model fails, a counterexample is provided which helps to improve the conjectured model. Different counterexample processing methods are presented and analyzed to identify an efficient counterexample processing method.

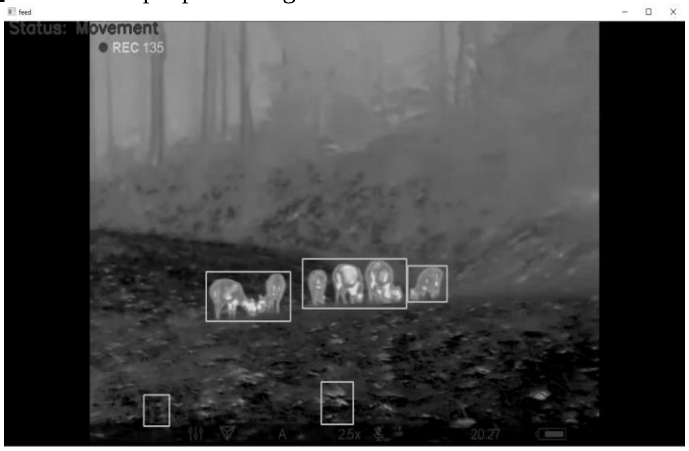


Fig. 3. Result of software system work

Software systems are part of our everyday life and they become more complex day by day. The ever-growing complexity of software and high-quality requirements pose tough challenges to quality assurance. The quality of a software system can be measured by software testing. However, if manually done, testing is a time-consuming and error-prone task. Especially test case design and test execution are the most cost-intensive activities in testing. In the previous 20 years, many automation tools have been introduced for automating test execution by using test scripts. However, the effort for creating and maintaining test scripts remains. Model-based testing (MBT) aims at improving this part by systematizing and automating the test case design. Thereby, test cases or automatable test scripts can be generated systematically from test models.

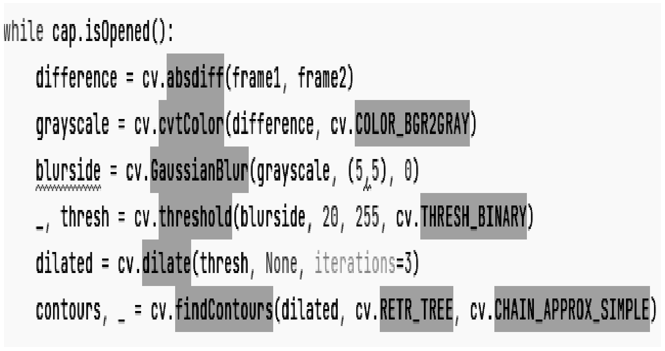
For this software system, all components such as video thread, logic function, user interface, input devices, and cameras should work properly to see the real result.

VII. ALGORITHM FOR TRACKING MOVING OBJECTS

This software system uses Open CV and Python as a core. Conceptually, a byte is an integer ranging from 0 to 255. In all real-time graphic applications today, a pixel is typically represented by one byte per channel, though other representations are also possible. An OpenCV image is a 2D or 3D array of the array type. An 8-bit grayscale image is a 2D array containing byte values. A 24-bit BGR image is a 3D array, which also contains byte values. We may access these values by using an expression, such as image[0, 0] or image[0, 0, 0]. The first index is the pixel's y coordinate or row, 0 being the top. The second index is the pixel's x coordinate or column, 0 being the leftmost. The third index (if applicable) represents a color channel. For example, in an 8-bit grayscale image with a white pixel in the upper-left corner, the image[0, 0] is 255. For a 24-bit BGR image with a blue pixel in the upper-left corner, image[0, 0] is [255, 0, 0].

The next problem facing computer vision is noise. To typically deal with noise is by using statistical methods. For example, it may be impossible to detect an edge in an image merely by comparing a point to its immediate neighbors. But if it looks at the statistics over a local region, edge detection becomes much easier. A real edge should appear as a string of such immediate neighbor responses over a local region, each of whose orientation is consistent with its neighbors. It is also possible to compensate for noise by taking statistics over time. Still, other techniques account for noise or distortions by building explicit models learned directly from the available data. For example, because lens distortions are well understood, one need only learn the parameters for a simple polynomial model in order to describe—and thus correct almost completely—such distortions. The actions or decisions that computer vision attempts to make based on camera data are performed in the context of a specific purpose or task. We may want to remove noise or damage from an image so that our security system will issue an alert if someone tries to climb a fence or because we need a monitoring system that counts how many people cross through an area in an amusement park. Vision software for robots that wander through office buildings will employ different strategies than vision software for stationary security cameras because the two systems have significantly different contexts and objectives. As a general rule: the more constrained a computer vision context is, the more it can rely on those constraints to simplify the problem and the more reliable our results will be.

Using CV absdiff find the difference between the initial state of the frame and the current one. The next step is to convert the difference into the BGR2GRAY color it needs for the GuassianBlur function to track changes between two frames. In cases like noise removal, erosion is followed by dilation. As erosion removes white noises, it also shrinks our objects. So, we dilate it. Since the noise is gone, they won't come back, but our object area increases. It is also useful in joining broken parts of an object. Using CV dilated filter the image from noise and then found contours of moving objects in a live stream. (see Fig. 4.)



```
while cap.isOpened():
    difference = cv.absdiff(frame1, frame2)
    grayscale = cv.cvtColor(difference, cv.COLOR_BGR2GRAY)
    blurside = cv.GaussianBlur(grayscale, (5,5), 0)
    _, thresh = cv.threshold(blurside, 20, 255, cv.THRESH_BINARY)
    dilated = cv.dilate(thresh, None, iterations=3)
    contours, _ = cv.findContours(dilated, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)
```

Fig. 4. Part of the coding scheme

OpenCV grew out of an Intel Research initiative to advance CPU-intensive applications. Toward this end, Intel launched many projects including real-time ray tracing and 3D display walls. One of the authors working for Intel at that time was visiting universities and noticed that some top university groups, such as the MIT Media Lab, had well-developed and internally open computer vision infrastructures—code that was passed from student to student that gave each new student a valuable head start in developing his or her own vision application. Instead of reinventing the basic functions from scratch, a new student could begin by building on top of what has come before.

VIII. APPLICATION TESTING

To test the current system, it was decided to use manual testing as it is more flexible and reliable. And, as a result of testing, all the found bugs and problems were fixed.

In the process of performance testing, it was decided to use Studio Profiled instrument as it is built into the IDE, so there is no need to use any other external tool.

There are requirements for desktop application's memory usage that applications should meet. The “Standard” applications should use up to 500 MB of Random-access memory (RAM), the “Media-intensive” – from 400 MB to 700 MB, and the “Huge” – from 800 MB and up to 1200 MB.

To test the application's performance, it is important to make it work at full strength. This type of testing is called stress testing. Stress testing is used to get the stability of the tested system or application. During the test, there are performed operations that bring the use of resources by the application to the maximum.

In this case, the most resource-intensive operations are the calculation of the location of the moving object relative to the entire image and the live conversion of frames from the video stream into a color filter. So, during testing in stress mode, the maximum value of CPU load was 12% and the maximum use of RAM was 450 MB. (see Fig. 5). The rate of transformation is up to 0.02 milliseconds per frame.

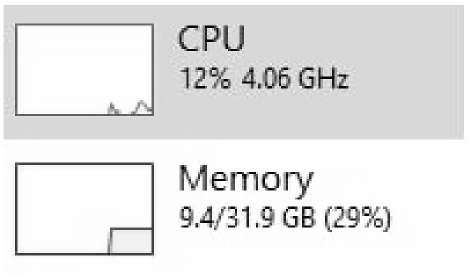


Fig. 5. Performance test in stress mode

IX. CONCLUSION

As a result of this work, a software system for motion detection and environmental tracking was created using the latest technologies (which will be relevant and easy to maintain in future development) for the development of this type of system, namely the desktop side in the form of a cross-platform user interface and using Python frameworks.

In addition, Open CV services were integrated for authorization and live monitoring. The idea of a fully customizable system for any video input device was also implemented. This meant that end users could independently configure the monitoring and tracking system. This was a really important aspect of system functionality. Another important requirement for the system was application optimization. Since the desktop interface used video stream rendering, there were often problems with the use of RAM.

The current program met the requirements for standard cross-platform programs. RAM usage for standard cross-platform desktop interfaces was between 200MB and 400MB, up to 350.3MB under stress. The use of the proposed algorithm ensured the use of 350 MB of memory under the worst conditions which was quite a satisfactory result for such a development. The resulting application was a working system that allowed users to monitor the situation at their observation post and directly participated in the security of their observation post.

References

[1] Shubham Kumar; Jonathan Mi. *et al*. (2021). “Human-Inspired Camera: A Novel Camera System for Computer Vision”, *International SoC Design Conference (ISOCC)*, pp. 1–20. doi: 10.1109/ISOCC53507.2021.9613914.

[2] Xin Zhang; Shuo Xu. *et al*. (2020). “Research on Image Processing Technology of Computer Vision Algorithm”, *International Conference on Computer Vision, Image and Deep Learning (CVIDL)*, pp. 20–25. doi: 10.1109/CVIDL51233.2020.00030.

[3] Ubiratan Ramos, Maurício Edgar Stivanello, Marcelo Ricardo Stemmer (2020). “Adaptable Architecture for the Development of Computer Vision Systems in FPGA”, *IEEE Latin America Transactions*, pp. 8–14. doi: 10.1109/TLA.2020.9400438.

[4] Cheng Jiang; JiaQi Sun. *et al.* (2022). “Exploring a Computer Vision and Artificial Intelligence-based Approach to Sit-and-reach Distance Measurement”, *2022 3rd International Conference on Computer Vision, Image and Deep Learning & International Conference on Computer Engineering and Applications (CVIDL & ICCEA)*, pp. 478–489. doi: 10.1109/CVIDLICCEA56201.2022.9825271.

[5] Honeye Rahmani; S. Mahmoud Taheri *et al.* (2020). “From Brain Decoding To Brain-Driven Computer Vision”, *International Conference on Machine Vision and Image Processing (MVIP)*, pp. 320–321. doi: 10.1109/CVIDLICCEA56201.2022.9825271.

[6] Emre DANDIL (2019). “Computer Vision Based Distance Measurement System using Stereo Camera View”, *International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pp. 1–5. doi: 10.1109/ISMSIT.2019.8932817.

[7] Kshitij Meena; Manish Kumar. *et al.* (2020). “Controlling Mouse Motions Using Eye Tracking Using Computer Vision”, *International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp. 10–16. doi: 10.1109/ICICCS48265.2020.9121137.

[8] A Byte of Python (2022). [Electronic resource]. – Access mode: https://homepages.uc.edu/~becktl/byte_of_python.pdf. (Accessed: September 29, 2022).

[9] Junshu Zhang; Jindong Zhang. *et al*. (2020). “A perspective transformation method based on computer vision”, *International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, pp. 15–16. doi: 10.1109/ICAICA50127.2020.9182641.

[10] Learning OpenCV (2022). [Electronic resource]. – Access mode: https://www.bogotobogo.com/cplusplus/files/OReilly%20Learning%20OpenCV.pdf. (Accessed: September 29, 2022).

[11] Serhii Kundys, Bohdan Havano, Mykola Morozov (2022). “Software System for Monitoring the Situation in the Settlement”, *Advances in Cyber-Physical Systems*, 7(1), pp. 38-45. doi: 10.23939/acps2022.01.038.

Bohdan Tsiunyk is a student who is currently receiving a B.S. degree in Computer Engineering at Lviv Polytechnic National University and a senior AQA engineer at Epam Systems. His research interests include computer vision, machine learning, computer engineering, and the development of Python frameworks.

Oleksandr Muliarevych Ph.D. in Computer Systems and Components, Associate Professor at the Computer Engineering Department at the Lviv Polytechnic National University. His research interests include distributed highly scalable microservice systems, swarm intelligence, IoT, cloud computing, parallel computing technologies, computer vision, machine learning, and multi-agent systems applications.