

**SECTION:** COMPUTER SCIENCE.

**SEKCJA:** INFORMATYKA.

**How to cite:** Bahatskyi, O. (2024). Using a Micro-PC to Process Video Information as a Component of Edge Computing, *Innovations and Global Solutions (Poland)*. (pp. 15-19). Futurity Research Publishing. <https://futuraity-publishing.com/international-conference-on-science-innovations-and-global-solutions-archive/>

## Using a Micro-PC to Process Video Information as a Component of Edge Computing

***Oleksii Bahatskyi***

*PhD in Technical Science, Senior Researcher*

*V.M. Glushkov Institute of Cybernetics of the NAS of Ukraine, Kyiv, Ukraine*

*<https://orcid.org/0000-0003-2969-6630>, Scopus Author ID: 57203882961*

*Email - [bagatskyi.o.v@gmail.com](mailto:bagatskyi.o.v@gmail.com)*

**Accepted:** June 2, 2024 | **Published:** June 14, 2024 | **Language:** English

**Abstract:** The increasing volume of video data from surveillance cameras worldwide necessitates expanded processing resources. This study explores the use of "smart" video cameras that perform initial data processing within peripheral computing systems, or Edge Computing. These cameras, part of a larger Smart system, transmit processed data for further analysis. We evaluate the potential of micro-PCs (similar to RaspberryPi, equipped with video sensors and network transmission capabilities via Ethernet or Wi-Fi) as components of intelligent cameras, focusing on their ability to handle primary data processing.

**Keywords:** Edge computing, smart video camera, Smart systems, video processing.

## Introduction

The use of modern hardware complexes enables the resolution of specific problems associated with processing photo and video materials and facilitates the creation of real-time object identification systems (notably, inexpensive video sensors generally do not produce images at rates exceeding 30 frames per second). Regarding hardware, micro-PCs such as the Raspberry Pi (Raspberry Pi Documentation, 2024) and similar devices can connect to networks via Ethernet or Wi-Fi. Typically, these microcomputers operate with an OS that includes compilers for programming languages, making them ideal for developing applications. For instance, a Linux-based OS comes equipped with built-in C and C++ compilers, which naturally supports the development of programs for data acquisition, processing, and interpretation. Moreover, using an operating system eliminates the need to develop driver routines that interact with hardware components, as the system is already configured with all necessary peripherals, allowing for their management through standardized input/output commands. The only system component requiring configuration or creation is the sensor driver. Therefore, the utilization of microcomputers for data processing allows for the development of "custom" hardware and software complexes that obviate the need for external data processing.

The **aim** of this work is to analyze the feasibility of using inexpensive micro-PCs as a component of a "smart" camera that performs primary processing of video sequences in real time.

## Research Results

Primary processing can involve, for example, the real-time delineation of contours in video sequences. All such methods of contour detection rely on a fundamental characteristic of the brightness signal—discontinuity. The most well-known method for identifying discontinuities is spatial filtering, which involves image processing using a square matrix. This matrix may be referred to as a mask, filter, kernel, window, or template. Typically, a window operator multiplies a defined part of the image by a matrix of coefficients, encompassing a broad range of operations on the image. To detect lines or contours, it is often advisable to employ the Laplacian operator, which calculates the second derivative of a function (Gonzalez & Woods, 2007).

The author proposes using and evaluating the processing speed of micro-PCs, specifically the "Orange Pi Lite" (512 MB RAM) (Orange Pi Lite, 2024) and the "Nvidia Jetson Nano" (Robotics and Edge Computing, 2024), neither utilizing CUDA, equipped with Ubuntu OS and paired with a Logitech C300 USB video camera set to a default resolution of 800x600. These proposed micro-PCs, based on quad-core SoCs, enable parallel data processing through their four ARM architecture-based hardware processor cores, potentially reducing data processing time from the video sensor. To harness this capability, it is necessary to develop software tools optimized for the available hardware resources.

The following software components can be utilized to process video data from the camera:

- ✚ A video sensor driver (either generic or custom-built) compatible with the Linux API v4l2 (Schimek et al., 2024), facilitating data reception and video sensor setting management. Typically, the operating system installs this driver automatically, requiring no modification;
- ✚ A routine dedicated to data processing, leveraging the available hardware computing resources (processor cores, memory, etc.) to produce a result;

- ✚ A subroutine responsible for outputting (or further transmitting) the result, enabling identification by the user or further processing of the received information. The output can be in the form of a video or photo file with specialized "marks" or as a string of characters (e.g., coordinate values).

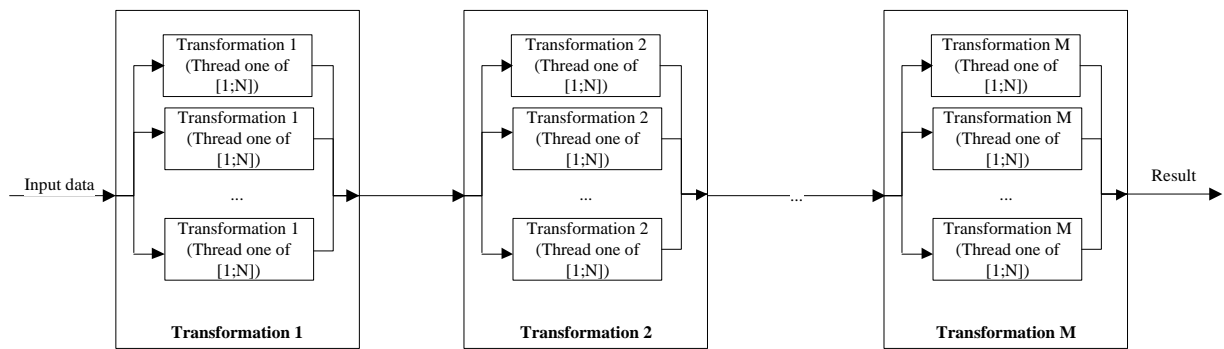
The efficiency of this routine depends on various factors (I/O driver, data transfer protocol, hardware support for data transfer protocols, etc.), which are not analyzed in this work.

Clearly, the subroutine responsible for data processing typically requires the most time; therefore, utilizing parallel data processing is optimal for obtaining results. Figure 1 illustrates a program model designed specifically for parallel data processing. This model is based on the Chain of Responsibility (Project Management Institute, Inc., 2024) and ThreadPool (Archiveddocs, 2024) patterns.

The Chain of Responsibility is a behavioral design pattern that allows requests to be passed sequentially through a series of transformations; this pattern links all handler objects into a single "chain" or "pipeline." Conversely, ThreadPool is a straightforward pattern for managing parallel processing, facilitating efficient hardware utilization to process large data volumes, albeit with some considerations regarding data organization. The principle of using these two patterns is straightforward—to minimize processing time, a "pipeline" is created where transformations are performed sequentially. Simultaneously, data for specific transformations are programmatically distributed across multiple "cores" for parallel processing, as depicted in Figure 1. Additionally, to accelerate processing, it is advisable to employ non-blocking parallelization. It should be noted that each transformation generates an object storing the intermediate result, which requires hardware resources for storage. Consequently, some transformations should not be parallelized to enhance the performance of a whole program.

**Figure 1**

*The proposed software model of data processing for micro-PC (parallelization for data flow)*



Source: author own development

The software implementation created for speed evaluation comprises five components (although, as mentioned earlier, not all parts should be parallelized):

- ✚ A subroutine for reading data from the video sensor, utilizing standard v4l2 APIs to capture real-time data at an 800x600 resolution.
- ✚ A resolution emulation routine used to achieve resolutions other than 800x600 by adding or removing lines and columns in the image to respectively increase or decrease the resolution.
- ✚ A subroutine employing a 3x3 Laplace mask in "non-blocking" parallelization mode, written in C++ version 11. To minimize execution time, the `std::async` function is used, into which the already received data is loaded.

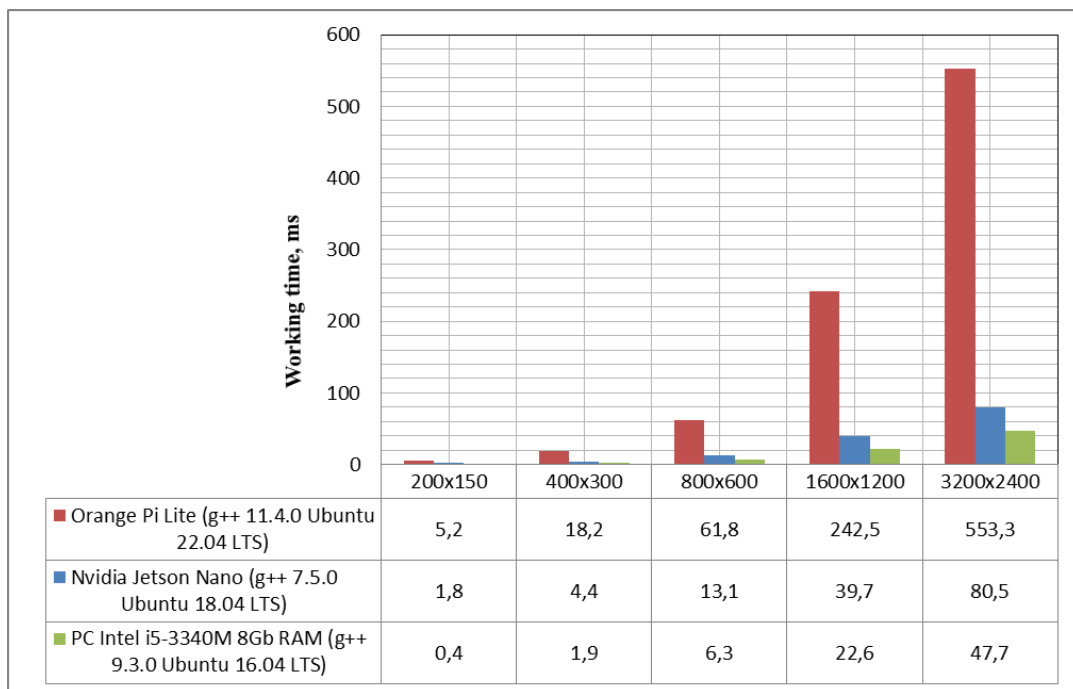
- ✚ A subroutine for object search—determining the boundaries and center of the object to either locate it or confirm its absence. Initially, methods for identifying the maximum brightness values characteristic of the object are employed, followed by determining its position in two-dimensional space within the photo. Subsequently, the center of the object is calculated.
- ✚ A subroutine for outputting the obtained results, utilizing `std::boost` for manipulating graphics files.

The software implementation was tested on two micro-PCs, the "Orange Pi Lite" and "Nvidia Jetson Nano." A ThinkPad Edge E530 laptop (Intel i5-3340M, 8GB RAM) served as the speed benchmark (Lenovo Group Limited, 2024).

Figure 2 displays the results of the software tests. To estimate the average time for different resolutions, as shown in this figure, 10 values were collected, and the mean value for each resolution was then calculated.

**Figure 2**

*Dependencies of processing time on image sizes by using different hardware platforms*



Source: author own development

Analyzing the results shown in Figure 2, it is evident that the micro-PC "Orange Pi Lite" can serve as a component of an intelligent camera in peripheral computing systems only for processing video data at speeds not exceeding 16 frames per second (i.e., 62.5 ms) at a resolution of 800x600. For higher resolutions, using this micro-PC is impractical, whereas for lower resolutions, its use remains feasible.

Conversely, the "Nvidia Jetson Nano" has sufficient resources to process video at a resolution of 800x600 and 30 frames per second. However, when increasing the resolution to 1600x1200, the processing speed achievable by this micro-PC should not exceed 25 frames per second (i.e., 40 ms).

## Conclusions

The results suggest the feasibility of using "smart" video cameras based on micro-PCs for the primary processing of video information in peripheral computing systems. However, preliminary testing to assess the specific capabilities of the micro-PC for primary processing and the selection of an appropriate resolution is necessary.

## References

- Gonzalez, R. C., & Woods, R. E. (2007). *Digital image processing* (3rd ed.). Pearson.
- Lenovo Group Limited. (2024). *Edge E530 (ThinkPad)*. Lenovo.com. Retrieved June 13, 2024, from <https://pcsupport.lenovo.com/us/en/products/laptops-and-netbooks/thinkpad-edge-laptops/thinkpad-edge-e530>
- Archiveddocs. (2024). *Thread pool design pattern*. Microsoft.com. Retrieved June 13, 2024, from <https://learn.microsoft.com/en-us/archive/technet-wiki/13245.thread-pool-design-pattern>
- Robotics and Edge Computing. (2024). NVIDIA Jetson nano. NVIDIA. Retrieved June 13, 2024, from <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/product-development/>
- Orange Pi Lite (2024). Orangepi.org. Retrieved June 13, 2024, from <http://www.orangepi.org/html/hardWare/computerAndMicrocontrollers/details/Orange-Pi-Lite.html>
- Project Management Institute, Inc. (2024, June 13). The Chain Of Responsibility Pattern. Pmi.org. Retrieved June 13, 2024, from <https://www.pmi.org/disciplined-agile/the-design-patterns-repository/the-chain-of-responsibility-pattern>
- Raspberry Pi for industry (2024). Raspberrypi.com. Retrieved June 13, 2024, from <https://www.raspberrypi.com/for-industry/>
- Schimek, M. H., Dirks, B., Verkuil, H., Rubli, M., & Walls, A. (2024). *Video for Linux Two API specification*. LinuxTV.org. Retrieved June 13, 2024, from [https://www.linuxtv.org/downloads/legacy/video4linux/API/V4L2\\_API/spec-single/v4l2.html](https://www.linuxtv.org/downloads/legacy/video4linux/API/V4L2_API/spec-single/v4l2.html)