Journal of Engineering and Technology for Industrial Applications, 2018. Edition. 13.Vol: 04
https://www.itegam-jetia.org
ISSN ONLINE: 2447-0228
DOI: https://dx.doi.org/10.5935/2447-0228.20180004

A hardware-software architecture for computer vision systems

Antônio Otavio Fernandes[1], Luiz Fernando E. Moreira[2], Glauber Tadeu S. Carmo[3], José M. Mata[4]

[1]Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Brasil.
[2] Invent Vision Sistemas de Imagem e Visão, Belo Horizonte, Minas Gerais, Brasil.
[3,4]Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Brasil.

Email: otavio@dcc.ufmg.br, luizf@ivision.ind.br, glaubertadeu@gmail.com, mata@dcc.ufmg.br

Received: January 16th, 2018.

Accepted: February 14th, 2018.

Published: March 30th, 2018.

Copyright ©2016 by authors and Institute of Technology Galileo of Amazon (ITEGAM).
This work is licensed under the Creative Commons Attribution International
License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/

ABSTRACT

Computer vision systems are based on capture and processing of images, from which information for the application is extracted. This paper presents a hybrid platform which can be easily configured and efficiently used for a large number of applications in computer vision. It consists of a dedicated processor integrated to the capture element, connected to a general-purpose computer for the high-level application processing. The camera module consists of a DSP executing a middleware which provides all the basic functions, such as image capture, low-level image processing, control, and communication. Experiments showed that good performance is obtained with this platform, and the framework presented may simplify the development of computer vision systems. An application for the hardware-software architecture presented here would be the validation and prototyping of computer vision systems.

Keywords: Computer vision platform, Low-level image processing, Smart camera.

I. INTRODUTION

Computer vision systems are based on capture and processing of images, from which information for the application is extracted. Applications include surveillance cameras, automated visual inspection, and microscopy [7]. Traditional computer vision systems consist of a camera connected to a general-purpose computer which runs the application. The result of the processing can activate complementary actuators or go to the output device (Figure 1). The whole system can be integrated in one component, the smart-camera, which includes image capture, processing, and output (Figure 2).

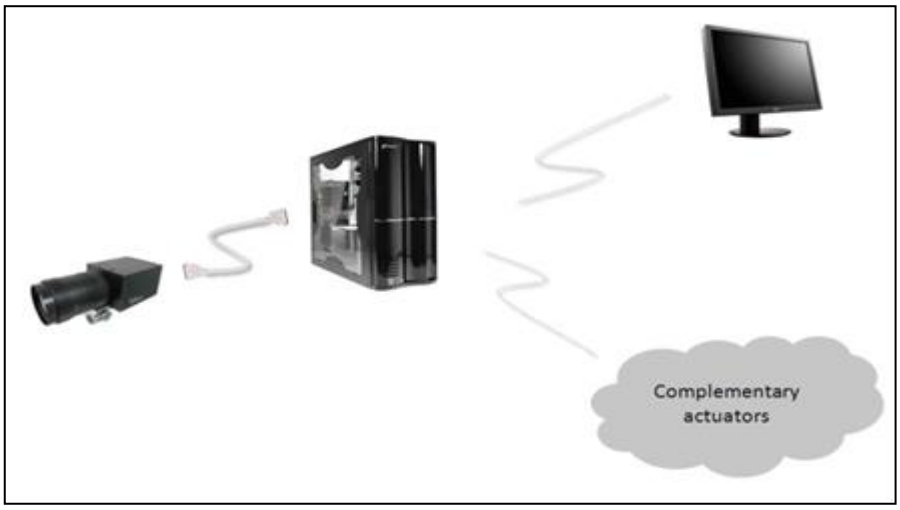

Figure 1: Traditional computer vision system.
Source: Authors, (2018).

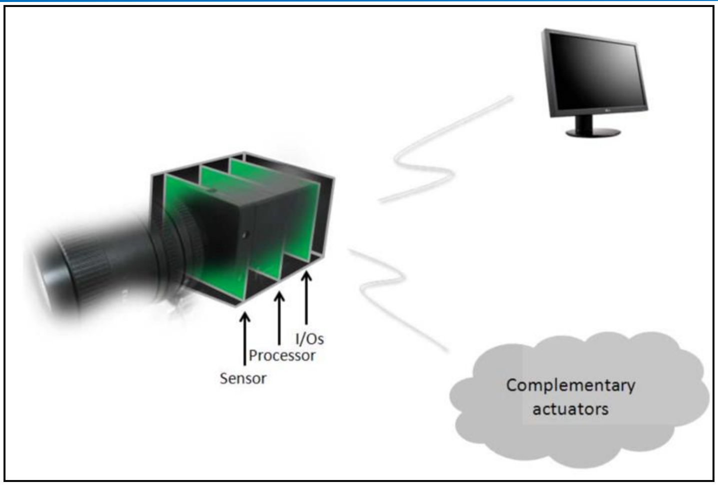

Figure 2: Smart camera.
Source: Authors, (2018).

Image representation requires a huge amount of memory, and image processing requires strong processing power. In the traditional vision system, the time for transmission between camera and computer may be extremely long. In the smart-camera, communication camera-processor is direct, significantly reducing the time between capture and processing.

Expressive image processing power is obtained through the use of dedicated processors and especial architectures. On the other hand, high-level applications run better on general purpose computers. The development of applications in these dedicated platforms requires from the developer detailed knowledge of all its resources; high-level applications may not fit well in these dedicated processors.

We propose a hybrid architecture, consisting of a dedicated processor integrated to the capture element, for the low-level image processing, connected to a general-purpose computer for the high-level application processing (Figure 3).

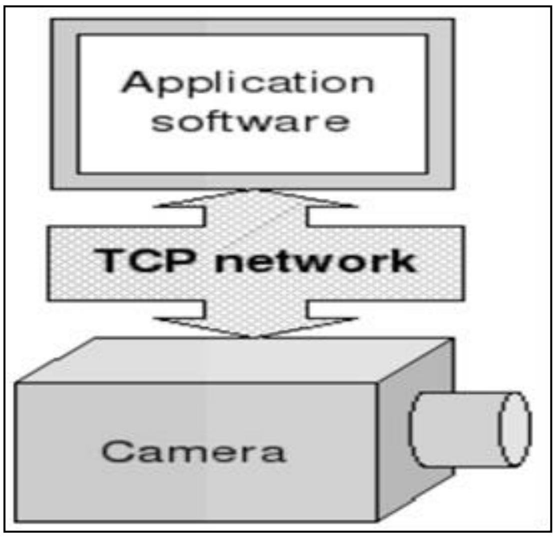

Figure 3: Hybrid architecture.
Source: Authors, (2018).

In order to ease the programming of the dedicated processor, we provide a framework for the development of applications, which did not require detailed knowledge of the platform. The framework consists of a middleware running in the dedicated processor and configuration software running in the general-purpose processor. The middleware provides all the basic functions, such as image capture, low-level image processing, control, and communication.

II. COMPUTER VISION SYSTEMS

II.1. REQUIREMENTS

Of the five senses, vision is the one that provides most of the data we receive. Its function provides us a detailed description of the surrounding tridimensional world, which changes constantly. Although it involves a huge amount of information and complex processing, the human visual system (eye-brain) can interpret this information easily. Efforts have been made to get machines to do this work, but the power of these machines is still far from the power of the human vision [6].

Computer vision can be seen as inverse computer graphics. In computer graphics the main motivation is the generation of images, transforming abstract descriptions into images. In computer vision the abstract description is obtained from the image, allowing object recognition. The description of a scene, besides identification of each object, includes position, color, orientation, and other aspects.

In order to understand the difficulties of implementing machine vision, let’s consider the object recognition problem. Recognition involves image capture by the camera, grabbing image (digitize and store), preprocess (suppress noise and regularize the image data), and pattern recognition. These tasks are complex and demand large space and processing time.

An important feature of most of the applications of computer vision is that they take place in real time: machine vision must be able to keep up with the ongoing process.

There is a need for tools (hardware and software) to help the developer of computer vision systems. Hardware platforms must achieve all the application requirements, and software tools should ease the programming task. It is desirable that a computer vision system include other benefits like modularity, portability, extensibility, and configuration and operation facilities. This demand led to studies for a better understanding of the steps of the vision process and to new computer vision development models, new hardware and software architectures, new dedicated processors, and new design techniques.

II.2. OPERATIONS

A computer vision system consists basically of three elements: capture, processing and output. Figure 4 shows the block diagram of a smart camera [3].

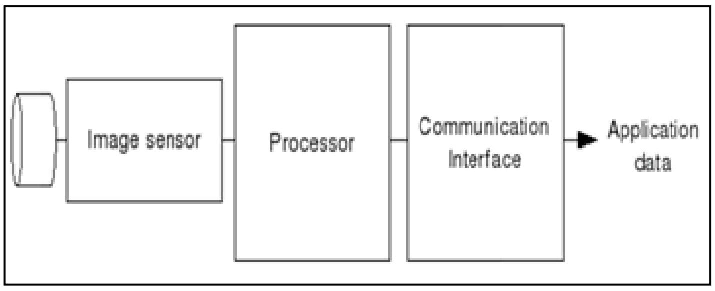

Figure 4: Block diagram of a smart camera.
Source: Authors, (2018).

Capture corresponds to transformation of the image into an abstract description; this is done by sensors (cameras). The main characteristics of available sensors are resolution (number of pixels), speed (frames per second), and number of colors.

The tasks of a computer vision application may be classified into three levels (low, medium, and high), according to the type of processing, the data access mode, and the control mode [12]. The type of processing refers to the used resources and data structures [5][9][12].

Low-level operations work directly with the image or part of it, being usually called image processing. They are characterized by the large amount of data processed, although acting on a few pixels each time. Access to the image is usually made in an ordered and predictable form. There is possibility of executing these operations in parallel. Smooth and convolution are examples of low-level operations.

Medium-level operations usually operate only on a region of the image; the aim is to extract and to organize information, like in the segmentation operation. Most of the time access to the image is also made in an ordered and predictable form, but the possibility of parallelism is not always evident.

High-level operations make part of the application; they operate on data structures extracted from the image, obtaining information for decision making. They consist of complex and sequential algorithms; access to the image information is random and non-deterministic. An example of this kind of operation is object recognition.

The complete processing in a computer vision application involves three steps. In the first step, low-level operations are applied to the captured image, with the aim to eliminate noise, and to get better contrast and other aspects that emphasize a given characteristic important for the application. The second step consists of extraction of information relevant to the application, using medium-level operations. In the third step decisions dictated by the application are made, using high-level operations, based on the information obtained in the second step.

II.3. ARCHITECTURE

Analyzing the process of a computer vision application, one can perceive that the different tasks demand different efforts from the various computational resources. One can conclude that a single processor architecture may not be able to carry all these operations efficiently; there is a need for a hybrid processing configuration, with specific architectures for each level [11].

Architectures for low-level operations are heavily explored, due to their specific characteristics and the large amount of data involved. There is a trend to use SIMD (single instruction multiple data) parallel architectures for low-level processing, and a second architecture for medium and high-level operations. Digital signal processors (DSPs) have also been used for low-level operations, with good performance [2]. Architectures for medium and high-level operations have many characteristics in common with general purpose processors.

Another important aspect is the availability of software, which should be easy to use and able to fully explore the hardware capabilities. There are several software packages for computer vision, like OpenCV (Open Source Computer Vision) [8], which is a library of programming functions.

III. DSCAM PLATFORM

We developed a hybrid platform which can be easily configured and efficiently used for a large number of applications in computer vision. It consists of a dedicated processor integrated to the capture element, for the low-level image processing, connected to a general-purpose computer for the high-level application processing (Figure 5).

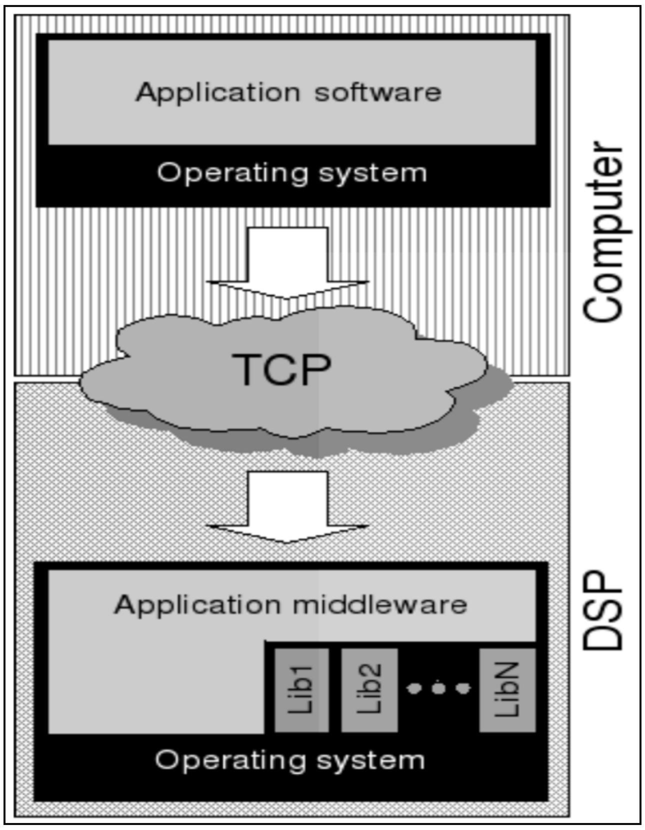

Figure 5: DSCAM platform.
Source: Authors, (2018).

The camera module consists of a DSP executing a middleware which provides all the basic functions, such as image capture, low-level image processing, control, and communication. The middleware is responsible for management of all the tasks executed in this module. The execution flow is defined in a file sent by the application module. There is a library of image processing functions; most of the functions were implemented with the use of the package OpenCV.

The DSP Blackfin 537 from Analog Devices [1] provided the best cost/benefit relation for the intended applications. Access to the DSP interfaces is simplified by the use of the platform Ez-Kit BF537. Management of hardware and software is provided by the mini operating system uClinux [10].

The configuration and operation software, running on the general-purpose processor, offers to the user an interface that allows access to all the platform resources and ways to define the flow of execution of the operations.

The independence between these two modules allows independent development and extension. Due to the modular structure of the platform, extensions can be easily incorporated. We included, for example, user authentication.

Communication with the sensor can be done in the Blackfin DSP through the PPI interface. In our prototype, this communication was not implemented; experiments were made considering that the image was already stored in a file.

IV. EXPERIMENTS

Two experiments were made, running different benchmarks on different platforms. The first experiment had the goal of comparing different platforms, and the second one had the goal of measuring the performance of the camera module (low-level operations). For the first experiment, we used part of the DARPA image understanding benchmark [13], and for the second experiment we used a simple computer-generated image on which low-level operations could be applied. Reference [4] presents a detailed description of the experiments.

Five different platforms were used for the experiments, including a PC (personal computer) with a 1.8GHz Celeron processor, SuSE Linux 9.2 operating system, DSP (digital signal processor) Blackfin 537, uClinux embedded Linux

1. PC + Linux;
2. PC + middleware + Linux;
3. DSP;
4. DSP + uClinux;
5. DSP + uClinux + middleware.

As we were interested in the performance of the processing and control operations, access to input/output devices was no included in the measurements. Image representation was previously loaded in the memory.

The DARPA image understanding benchmark specifies algorithms that involve all three levels of operations on images; it is used to validate processing architectures, especially parallel architectures. Its algorithms refer to applications close to real practice, involving location of rectangular objects in an image.

The first experiment was divided into three parts, comparing the execution of the DARPA benchmark algorithms on different platforms. In the first part, we compared platforms 1 and 2, in the second part platforms 4 and 5, and in third part, platforms 1 and 4.

Analyzing the results, we concluded:

- we can neglect the overhead introduced by the middleware, in both PC and DSP;
- medium-level or high-level algorithms have better performance on the PC;
- low-level algorithms have similar performance on both platforms;
- uClinux introduces an overhead when there are many mathematical operations;
- cache memory size has influence on the performance.

In the second experiment, we compared platforms 3 and 4, trying to evaluate the performance of the camera module, running low-level algorithms on simple images. Analyzing the results, we concluded again that uClinux introduces an overhead, due to mathematical and logical operations, cache use, and other factors.

V. CONCLUSION

In this work, we explored hardware and software platforms for computer vision systems. In especial we explored the software architecture for the camera module. Our experiments showed once more that dedicated processors and especial architectures, like DSPs, are indicated for low-level image processing, whereas medium and high-level operations can be executed with good performance on general-purpose processors.

A framework as presented here may simplify the development of computer vision systems. The middleware running in the camera module introduces no significant overhead to the system. Special attention must be given to the performance of the embedded operating system used.

An application for the hardware-software architecture presented here would be the validation and prototyping of computer vision systems. Due to the modularity and extensibility of the platform, prototyping of computer vision applications would be simplified.

VI. REFERENCES

[1] Analog Devices, "**Blackfin processors**," [Online]. Available: http://www.analog.com/en/embedded-processing-dsp/blackfin/processors/index.html

[2] D. Baumgartner, P. Rossler, W. Kubinger, "**Performance benchmark of dsp and fpga implementations of low-level vision algorithms**," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", Minneapolis, June 2007.

[3] A. N. Belbachir, "**Smart cameras**," Springer, 2009.

[4] G. T. S. Carmo, "**DSCAM: uma plataforma hardware-software para operações de visão computacional**," M.S. Thesis, Universidade Federal de Minas Gerais, Brasil, June 2009.

[5] A. Choudhary, J. Patel, N. Ahuja, "**Netra: a hierarchical and partitionable architecture for computer vision systems**," in IEEE Transactions on Parallel and Distributed Systems, Vol. 4, no. 10, pp. 1092-1104, October 1993.

[6] E. R. Davies, "**Machine Vision: Theory, Algorithms, Practicalities**," Morgan Kaufmann, 3rd edition, 2005.

[7] InventVision, **Sistemas de Imagem e Visão**. www.inventvision.com.br.

[8] OpenCV: **Open Source Computer Vision, version 2.1**, April 2010, [Online]. Available: http://opencv.willowgarage.com/wiki

[9] N. K. Ratha, A. K. Jain, "**Computer vision algorithms on reconfigurable logic arrays**," in IEEE Transactions on Parallel and Distributed Systems, Vol. 10, no. 1, pp. 29-43, January 1999.

[10] uClinux, "**Embedded Linux/Microcontroller Project**," [Online]. Available: http://www.uclinux.org/

[11] C. L. Wang, P. Bhat, V. Prasanna, "**High-performance computing for vision," in Proceedings of the IEEE**, Vol. 84, no. 7, pp. 931- 946, July 1996.

[12] C.Weems, "**Architectural requirements of image understanding with respect to parallel processing**," in Proceedings of the IEEE, Vol. 79, no. 4, pp. 537-547, April 1991.

[13] C.Weems, E. Riseman, A. Hanson, A. Rosenfeld, "**The DARPA image understanding benchmark for parallel computers**," in Journal of Parallel and Distributed Computing, Vol. 11, no. 1, pp. 1-24, January 1991.