CODING, PERFORMANCE AND IMPLEMENTATION COMPLEXITY DIFFERENCES BETWEEN OPENCV_1.X AND OPENCV_2.X

120044749
Brandon Dias*

*University of Brasília
Computer Science Department
Princípios de Visão Computacional

Email: dias.unb@gmail.com

Wednesday 30th March, 2016

Abstract— First practical work of Computer Vision Principles, a subject ministered by Professor Flávio de Barros Vidal. The experiment goes from foundations of OpenCV until the use of basic operations and mainstream functions, trying understand the impact of complexity reduction between the 2.x version of OpenCV and its precursors.

Keywords— OpenCV, Performance, Complexity, PVC

1 Goals

This practical work has the main focus on setting the mindset in the fresh-new students in the OpenCV environment. 2 main tasks were proposed: The first one was changing a "filter code", which was written in the first version of OpenCV, to the second version, paying attention on which advantages C++ brings over C; The second task was measure the time difference between both implementations.

2 Introduction

The OpenCV, Open Source Computer Vision, is a initiative to create a huge library of functions for all those students, developers, researches, entrepreneurs, companies, etc which their labor is over manipulating images inside Computer Vision's paradigm. Since it was released in 2000, openCV continuously grew and today has more than 2500 optimized algorithms, more than 47 thousand people of user community and estimated number of downloads exceeding 7 million"""".

As a free software, it doesn't fear to change, looking for better performance, even it's very basic core. And that's exactly what this experiment intends to do, by comparing the creation of 3 different images through `IplImage` concept and `Mat` concept. In the subsections below, it will be presented briefly the Struct and the Class, respectively. Also, the two algorithms used in the code: the filter, which sets the path for languages work on, and the time measurement function.

2.1 IplImage

It's a C language structure, called Struct. It implies in all variables inside be public and that it doesn't have Methodes by its own. As described by webpage of OpenCV:

> *"The IplImage is taken from the Intel Image Processing Library, in which the format is native. OpenCV only supports a subset of possible IplImage formats, as outlined in the parameter list above".*

2.2 Mat

A Class, with all that implies, since private variables until overload and polymorphism of its methods. Other main difference is how Mat, since it's a C++ construction, deals with memory allocation, keeping the programmer less concerned with parallel issues of his or her main focus of work. As described by webpage of OpenCV:

> *"Mat is basically a class with two data parts: the matrix header (containing information such as the size of the matrix, the method used for storing, at which address is the matrix stored, and so on) and a pointer to the matrix containing the pixel values (taking any dimensionality depending on the method chosen for storing) . The matrix header size is constant, however the size of the matrix itself may vary from image to image and usually is larger by orders of magnitude."*

2.3 Filters and Noise

As it is not the goal of this work, it's not to be spoken here. The only comprehension needed is that a noise is every distortion of the real representation of data sampling. Defying it like this,

the code shown in ?BElow? is a filter, an algorithm to reduce this distortions. It's clear that the core function access all elements in the image created. And it is all that matters, because this work wants to evaluate the speed of the computing of those actions.

3 Materials and Methodology

Computer Data:

- Ubuntu 14.04 LTS;
- S.O. 64-bit;
- Intel Core i5-2430M CPU @ 2.40GHz 4;
- 6GB RAM

3.1 Measuring Time: gettimeofday()

The used tool for evaluate time component of algorithm analysis, was the function gettimeofday(). It's referenced in linux documentation. The function gettimeofday() can get the time as well as a timezone.

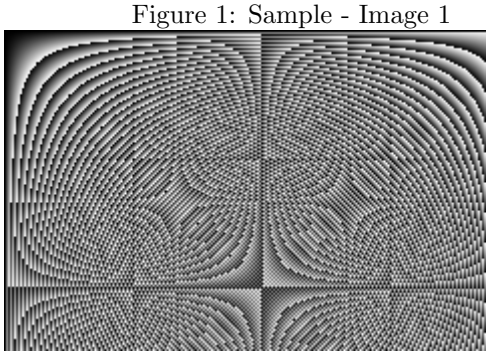
The acquisition was taken with 2 refinement: The generated figures are done in different archives, to avoid any delay caused by code itself (like add and print functions between main propose). And for each size of generated image, was taken 2 check points: the first one as soon as program initiate and right before showing results and closing; the other one, focus only on Image generation (loop for) to test reading/writting performance.

4 Results and Analysis

The results of each image are show in order of size. Note: each value is the integer median of 10 repetitions and the wait() was commented.

Image sample appearance:

Figure 1: Sample - Image 1



The results shows that the simplicity of code image 1280x720 can be really dominated by both algorithm.

Table 1: Milliseconds Performance

	1280x720		4320x1280		8640x4320	
cv1	98ms	05ms	133ms	129ms	418ms	271ms
cv2	104ms	06ms	141ms	38ms	365ms	255ms
	all	loop	all	loop	all	loop

In the image 4320x1280 is the best view for all performance gain. It's a lot faster, mainly when just loop iteration is observed.

In the third image, 8640x4320, the Mat() method is still faster, but suggest a convergence with iplImage();

OpenCV Documentation has it's own comparative between algorithm with Mat(). And the results are corroborated. OpenCV developers call these loop-access implementation of "Efficient Way".

5 Conclusion

Results shows that changes in openCV 1.x to 2.x really let performance better, manly for big matrices. But, more important, these work shows that openCV 2.x is the best option for those new researches and enthusiastic of Computer Vision, since it frees the coder from trouble by automatizing routines on code and cleaning the code for what matters most.