OpenCV performance on MAEMO/MeeGo

Iliya Mironov, Diana Ilina
Nizhny Novgorod State University
Gagarin Ave, 23, Nizhny Novgorod, Russia
frimen3000@mail.ru, ilina.diana@gmail.com

Abstract

OpenCV is an open source effective image processing library. This library expends image and video processing capabilities of Maemo/MeeGo platform and can be used by various applications that process embedded cameras images and videos as well as any other images/videos. The library is cross-platform, and runs on Mac OS, Windows and Linux.[1][3]

The target of this project is analyzing the possibility of using OpenCV library with MAEMO/MeeGo devices and creating user application that use OpenCV. The tasks of our project are porting OpenCV to MAEMO/MeeGo, developing an example application that uses OpenCV, measuring resource consumption.

The main goal of our project is to create applications that use OpenCV algorithms for measuring the performance of mobile devices based on the platform MAEMO/MeeGo.[6][7]

During the project implementation will be identified depending on the performance of mobile devices from various image parameters.

Index Terms: OpenCV library, Maemo/MeeGo

I. INTRODUCTION

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real time computer vision. This is one of most effective and famous open source image processing libraries. It can perform many of transformations and calculations over images. This library is based on OpenGL and it allows you to improve efficiently of processing by using hardware capabilities of Maemo/MeeGo device.

Example applications of the OpenCV library are Human-Computer Interaction (HCI); Object Identification, Segmentation and Recognition; Face Recognition; Gesture Recognition; Motion Tracking, Ego Motion, Motion Understanding; Structure From Motion (SFM); Stereo and Multi-Camera Calibration and Depth Computation; Mobile Robotics.

Augmentation of functionality and of mobile devices allows them to solve tasks before such as image/video processing and pattern recognition. High photo/video cameras resolution and increasing computing power mobile devices graphic acceleration provide the possibility to out some objects (e.g. faces,) in the picture, to do automatic photo processing.

This article will introduce the basic types of OpenCV, how to use them and how to do some simple image transforms, all with example applications and documented sample C code.

The main objectives of our project is to: create test applications that use algorithms OpenCV, OpenCV algorithms adapted to the capabilities of mobile devices, as well as the creation of packages for developers for the platform MAEMO/MeeGo with different architectures. Our study will determine the ability to create innovative applications for mobile devices such as innovative user interfaces or games with the extended reality.

II. MAIN PART

OpenCV is a computer vision library widely used with many functions that enable a computer program to "see" and to make decisions based on what it finds.

Officially launched in 1999, the OpenCV project was initially an Intel Research initiative to advance CPU-intensive applications, part of a series of projects including real-time ray tracing and 3D display walls.

The first alpha version of OpenCV was released to the public at the IEEE Conference on Computer Vision and Pattern Recognition in 2000, and five betas were released between 2001 and 2005. The first 1.0 version was released in 2006. In mid 2008, OpenCV obtained corporate support from Willow Garage, and is now again under active development. A version 1.1 "pre-release" was released in October 2008, and a book by two authors of OpenCV published by O'Reilly Media went on the market that same month.

Starting from OpenCV 2.0 the new modern C   interface has been introduced. It is crisp (less typing is needed to code the same thing), type-safe (no moreCvArr* aka void*) and, in general, more convenient to use.

Basic modules library OpenCV: Cxcore - core libraries, CV - The module of image processing and computer vision, Highgui - Module I / O image and video, create the user interface. These modules we used to work on the project.[2]

Computer vision is still one of the most fundamental, widely studied, and largely unsolved problems in the image processing. However, this theme already have some achievements and the theme of this paper is how to use easily ready solutions. Many methods for the image processing have been suggested, and you can find them in the OpenCV library. The OpenCV library is an open source project, which provides for us many ready solutions. However, it's not a trivial task to use this library in the Qt environment. To this end we have developed a special class to transfer data from the OpenCV format (IplImage) to the Qt image format (QImage).

In this paper we provide performance measurements for the most used image processing methods provided by the OpenCV library.

To test performance we used following devices:

- Internet-tablet Nokia N900 (ARM Cortex-A8 CPU 600 MHz, RAM: 256 MB, Swap: 768 MB)[8]
- Netbook DNS (Intel Atom N450 1660 MHz, RAM 1024 MB)

We have investigated following OpenCV methods for the image processing:

- Smoothing filters: Blur, Blur no scale, Gaussian, Median
- Delineation filters: Ceny, Sobel.
- Threshold filter (on the value space)
- Fluxion computing (by the Laplas operator)

For this methods we have carried out performance tests and below we provide several plots showing dependences of the processing duration on the image resolution.

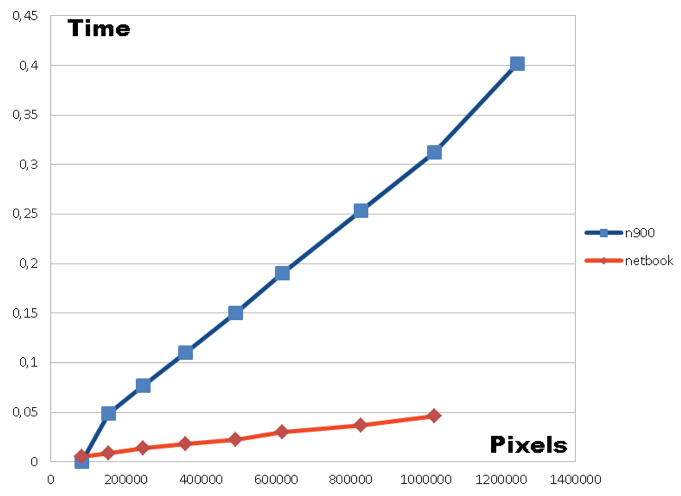

Fig.1 Performance testing results for the Blur filter

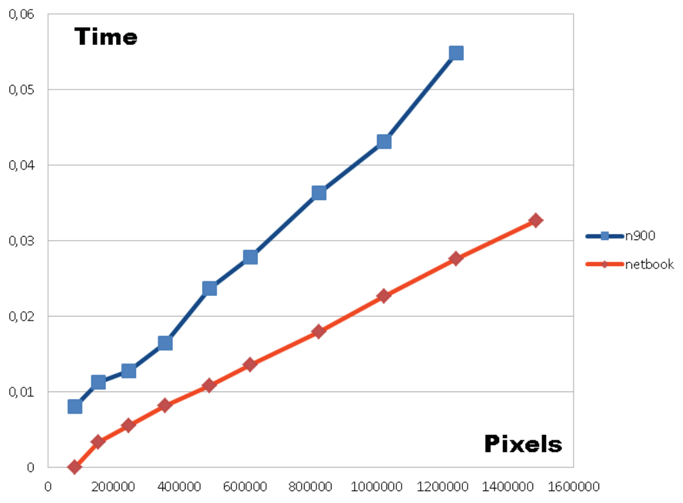

Fig.2 Performance testing results for the Blur no scale filter

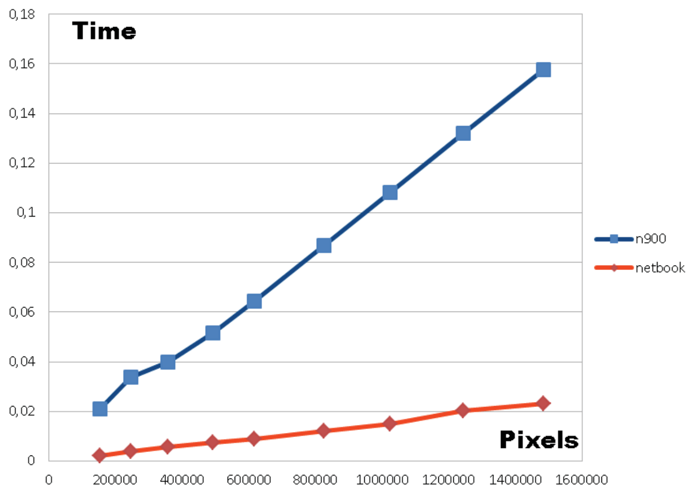

Fig.3 Performance testing results for the Gaussian filter

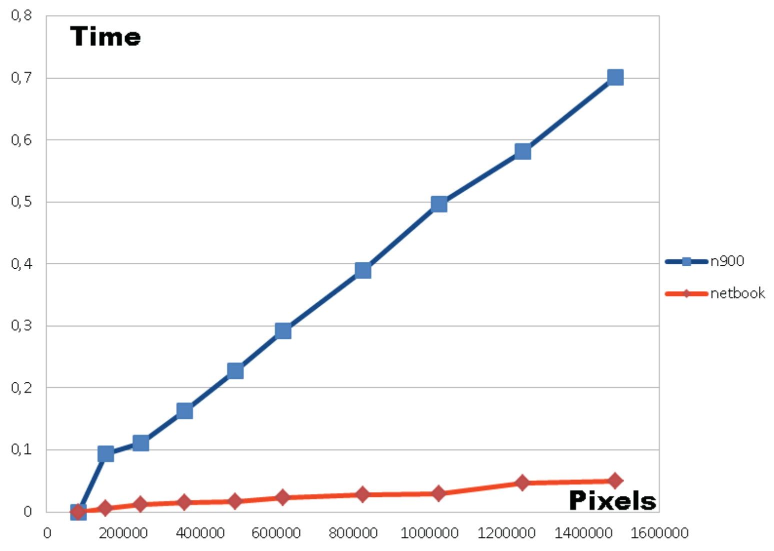

Fig.4 Performance testing results for the Median filter

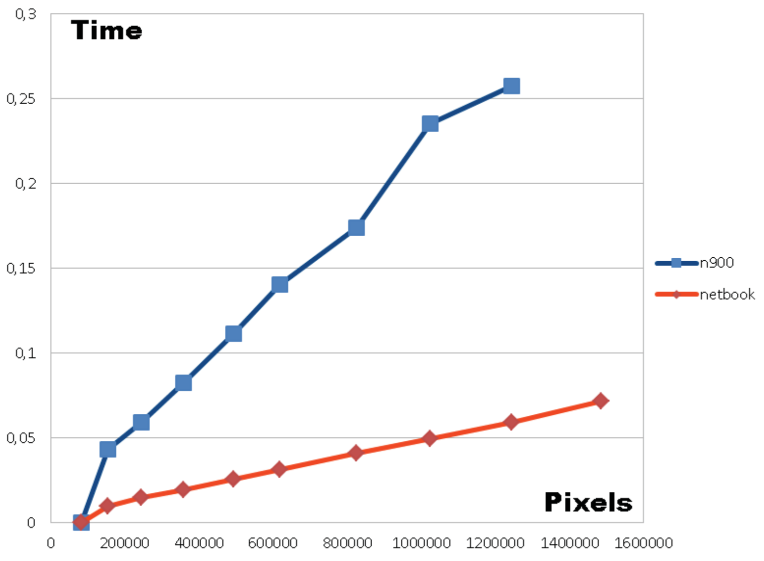

Fig.5 Ceny

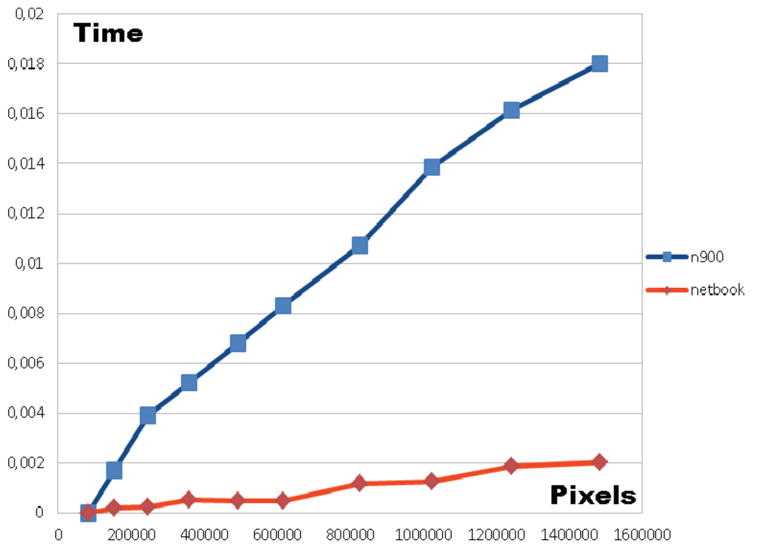

Fig. 6 Threshold

Bellow we provide examples of Ceny and Threshold methods application obtained with Nokia N900 device.

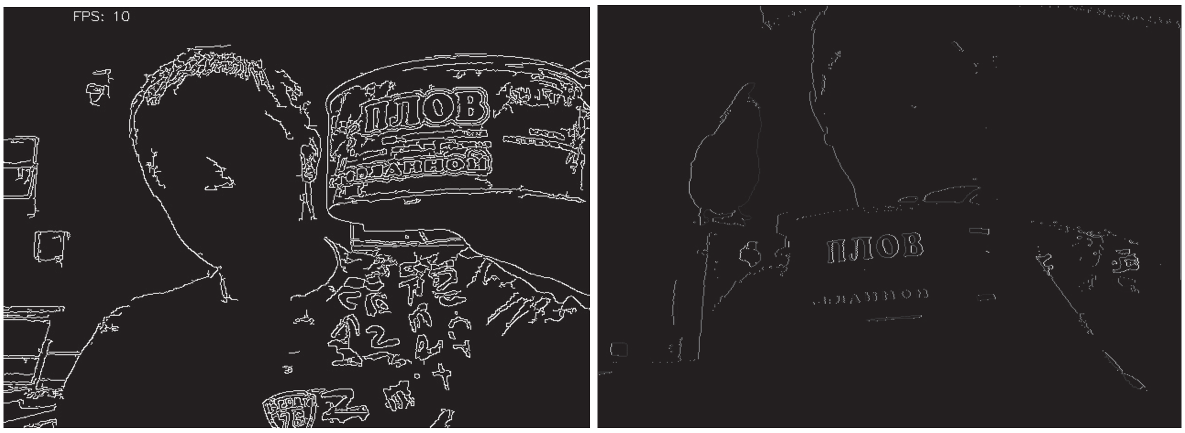

Fig.7 Results of Ceny image processing (leftward) and Threshold image processing (rightward)

Thus, according to our experimental plots we can tell that for any an OpenCV library's method we can define that image size which provide the same time of image processing for a Nokia N900 devic e and a netbook. As result we can take high performance on mobile devices via scaling of destination image.

Using the realized tests code we have made an application which process images from a camera's video dataflow via the afore-cited methods. This application is realized by means of the Qt with using of the OpenCV library. To adequate OpenCV library using in the Qt there is implemented methods to transfer data from the OpenCV image format (IplImage) to the Qt image format (QImage) and vice versa. Also to the functionality of our application we have added ability to save a single image and a video stream fragment. Via this application we have measured FPS (computed frames per second) for our two devices, which results of we provide following in Fig. 7.

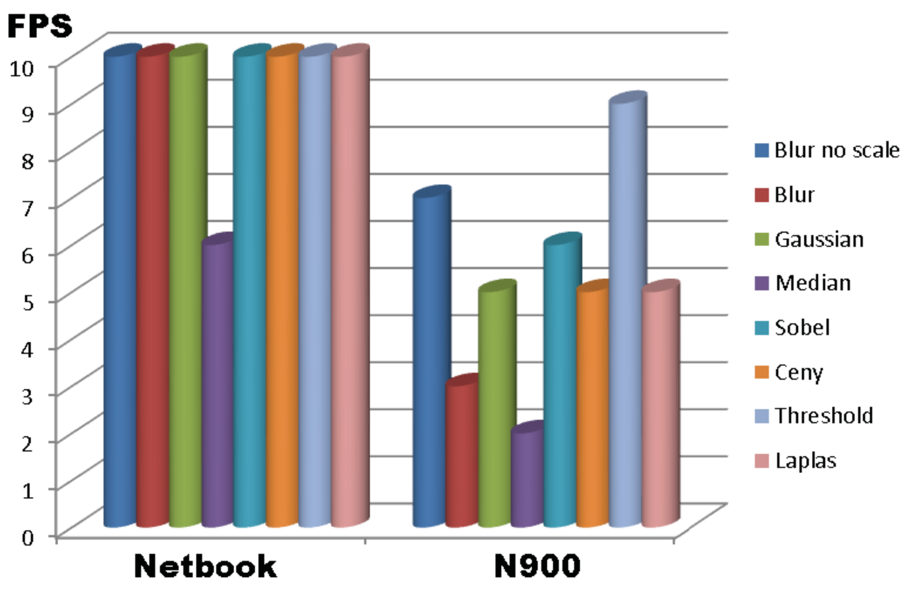

Fig. 7 FPS (computed frames per second) for different filters

image resolution is 640x480

III. CONCLUSION

As result of performed work we have tested the performance of the most used methods from the OpenCV library. Our tests showed ability to use this library on mobile phones in real projects. In spite of capacities of rapid single image processing, using the OpenCV library leaves be hard whereas there is low FPS on mobile phone. It confirms our experimental plots, which show us that optimum FPS will be reach at the points of crossing our plots.

Finally, we can tell that right now it is real to create a commercial application for image processing, such as an object identification, handwritten or typed text recognition from photo, denoising and image quality improving, photo processing.

REFERENCES

[1] OpenCV, http://opencv.willowgarage.com/wiki/

[2] OpenCV, http://opencv.willowgarage.com/documentation/cpp/introduction.html

[3] Bradsky G., Kaehler A. Learning OpenCV - O'Reilly, 2008.- C. 1 - ISBN 978-0-596-51613-0

[4] ARM EABI port, http://wiki.debian.org/ArmEabiPort

[5] Why ARM's EABI matters, http://www.linuxfordevices.com/c/a/Linux-For-Devices-Articles/Why-ARMs-EABI-matters/

[6] MeeGo, http://meego.com/

[7] MAEMO, http://maemo.org/

[8] Nokia N900, http://maemo.nokia.com/n900