Marvin – A Tool for Image Processing Algorithm Development

Gabriel Ambrósio Archanjo, Fabio Andrijauskas, Danilo Rossetto Muñoz
Marvin Project
{leirbag.arc,biozit,munozdanilo}@gmail.com

Abstract

This work presents, Marvin, tool for image processing algorithm development. This tool is composed by a framework and an image manipulation program that supports extensibility through plug-ins. The framework, used to develop the plug-ins, provides features for image manipulation, algorithm analysis and integration with the base application. Marvin is suitable for image processing algorithm development and prototyping and for education purpose.

1. Introduction

Image processing is increasingly used for many purposes and usually involves the development of algorithms to filter and extract information from images. Nowadays, many tools for image manipulation are available and they allow users to manipulate images and develop their own algorithms for image processing, such as GIMP [4] and Photoshop [5].

Marvin, as an alternative tool for image manipulation and image processing algorithm development, has an important differential from the majority of these kinds of tools, focused on image manipulation. Marvin is also an image manipulation program, but it is focused on the development of image processing algorithms. For this purpose, Marvin provides an easy-to-use white-box framework, that requires the knowledge of the framework internal structure from the plug-in developers [1]. The framework provides features to facilitate the development of image processing algorithms, and it is open source and multi-platform, considering that it is developed in Java.

2. Plug-ins

Every Marvin feature to manipulate images is externally implemented through plug-ins. The framework defines the interfaces and provides features to manipulate the images. A Marvin plug-in must implement the interface MarvinPluginImage that specifies which methods must be implemented to manipulate an image. Once Marvin plug-ins were implemented in Java, after the plug-in compilation, the class file must be copied to the plug-ins folder that is inside the Marvin folder. Every plug-ins located in that folder will be automatically loaded into Marvin image manipulation program in the next execution.

In many cases, it is interesting to use a plug-in inside another and the Marvin framework allows plug-ins integration. The plug-in that implements an edge detection algorithm, for example, uses the gray scale plug-in in the beginning of the process.

3. Performance Meter

Many image processing algorithms can be divided in few processes. An algorithm, for example, may be divided in three processes to find and segment the interest points of an image: representing the image in gray scale, finding the interest points and segmenting the image[2]. Each process has a different computational cost and the performance meter allows developers to analyses the performance of the entire algorithm and their processes individually, considering the number of steps and the execution spent time. Thus, it is possible a better understanding of the algorithm performance and facilitates optimizations, since is visible what algorithm processes are costly and most interesting to be optimized. Another performance meter application is to determine the complexity of an algorithm, Big-O notation, analysing the number of steps executed for different image resolution.

4. Plug-in history

In some cases to achieve some result in an image, it is necessary to use a few processes. In the beginning of an image processing method, compensations can be done, such as noise and blur reduction[3]. Preparing an image for pattern recognition, for example, in some applications it is interesting to remove noise, increase contrast and emphasize edges. For these cases that multiple processes are used, Marvin History stores all plug-ins and their configurations applied to an image. The history can be exported as an image strip containing the configuration and resulting images of each used plug-in. This feature allows future analyses of used processes and facilitates a continuous development of image processing processes.

5. Integrated Graphical User Interface

The plug-ins can have user specified attributes that determines how it will work and the Marvin framework provides features to integrate a GUI (Graphical User Interface) with these attributes. Thus, the plug-in developer sets the relation between the plug-in attributes and the interface components, added to the plug-in window. When the component value is changed, the associated attribute value is changed automatically. The developer does not worry about handling component events.

The Figure 1 shows a plug-in interface using a integrated GUI with a plug-in window, a filters history and the performance meter displaying the result of Interest Point plug-in.

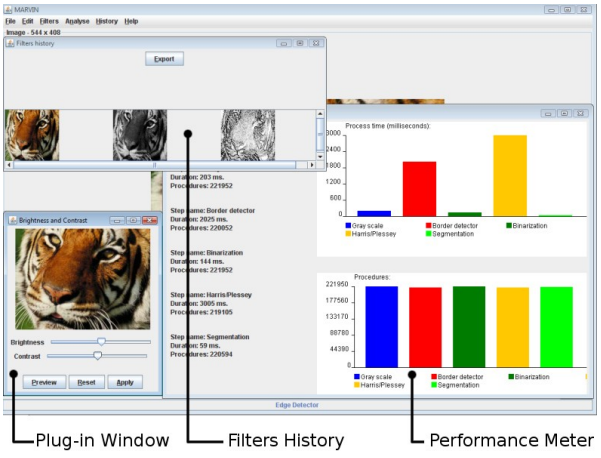

Figure 1. Marvin image manipulation program interface.

6. Current plug-ins

Currently, there are plug-ins developed for the Marvin application that provide the following features: image segmentation considering interest points, edge detection, image halftoning, Gaussian blur, image difference, image blending, steganography, color intensity histogram, brightness and contrast manipulation, sepia, mosaic, flipping, cropping, gray scale and negative colors conversion. Some of these plug-ins are contributions of third-party developers.

7. Conclusions and Future Work

Marvin is a tool composed by a framework and an image manipulation program. Features, such as image manipulation methods, integrated graphics user interface, filter history, plug-in integration and performance meter allow analysis, study, prototyping and development of image processing algorithms.

Marvin is developed in Java and it is open source. Every feature for image processing are developed externally through plug-ins, also in Java, by the open source community and are available at http://marvin.incubadora.fapesp.br/.

For the next version, new statistical charts support, new interface components and multi-image processing for motion analysis are expected.

8. References

[1] M.E. Fayad, D.C. Schmidt, "Object-Oriented Application Frameworks", *Communications of the ACM*, USA, 1997.

[2] C. Harris, M. STEPHENS, "A combined corner and edge detector", *Proceedings of The Fourth Alvey Vision Conference*, Manchester, pp 147-151. 1988.

[3] GONZALES Rafael, WOODS Richard, *Digital image processing*, Second Edition Pretenci Hall, 2001

[4] The GNU Image Manipulation Program – Official Website. http://www.gimp.org/. Accessed June 26, 2008.

[5] Photoshop CS3 Editions – Official Website. http://www.adobe.com/products/photoshop/. Accessed June 26, 2008.