A Visualization Toolkit for Teaching, Learning and Experimentation in Image Processing

Eugene Ageenko, Gaetano La Russa

{ageenko,larussa}@cs.joensuu.fi
Department of Computer Science, University of Joensuu, Box 111, FIN-80101 Joensuu, Finland

Abstract - Due to their intrinsic and abstract difficulties, Image Processing (IP) courses are in general of graduate and postgraduate level. For efficient IP teaching, a special focus needs be set on students’ experimental works. The scope is to allow students to concretize and try out the studied technology. Visualization tools for IP’s basic aspects are also needed by teachers that desire achieving better pedagogical results. Indeed, the visualization of the impact that an algorithm has on the visual data (such as e.g. filtering algorithm) is cognitively superior to the plain textual description. Students should then be able to actively and quickly complement their lectures by processing and manipulating images and even constructing personal IP tools. It is this hand-on experimentation that can significantly facilitate the learning of the mathematical concepts of image processing [10]. The toolkit presented in this article responds to the above requisites.

INTRODUCTION

Image Processing (IP) can be taught very effectively by complementing the basic lectures with computer laboratories where the participants can actively manipulate and process images [14].

Because of the widespread use of imaging (digital camera, scanners, etc.), there is an ever-increasing need to train specialists who are proficient with this new technology. Many universities are meeting this demand by offering courses in image processing (IP) but teaching IP is quite a challenging task. This might be the reason why image processing courses are in general of graduate and postgraduate level. IP courses have to deal with the concretization of various concepts that most of the times are of abstract mathematical nature. On the other hand, IP is also very practical discipline, and seeing the algorithms applied to the images especially in an interactive animated way has dramatic effects on the students.

According to authors’ experiences, an efficient teaching and learning of IP fundamentals is achieved if attention is paid to visual representation of the algorithms and the experimental work of the students. Visualization in IP can be considered a cognitive boosting factor while plain textual description alone would require a lot of imagination and interpretative skills. It is clear that better pedagogical results can be achieved if such visualization tools for the demonstration of the basic aspects of image processing are made available to teachers [5].

Modern commercial image manipulation tools (aka Photoshop) have complex user interface, sky-high prices, and moreover, they do not bring any insight on the algorithms they employ, when these are often offered as black-boxes and remain misery for the student. After using such tools the person became proficient in the tool but not in the IP, which of course might be suitable for Digital Designers but is not suitable for our goals. There has been a substantial effort by members of this community to create didactical tools for teaching IP [9]

In brief, there are [14]

– commercial packages for setting IP laboratories, e.g. MATLAB [4], and Khoros [3], their disadvantage of these approaches similar to that of commercial image manipulation tools [15]
– non-commercial platform independent C- or Java-based solutions and environments for image analysis and processing, such as ImageJ [2] or IPLab [14]
– collections of routines and classes without any environment, such as [11]-[13]
– collection of independent interactive applets such as HIPR [6]
– numerous scattered applets on the Web demonstrating this or that algorithm.

Sage and Unser claim that “the best way to truly understand an algorithm is obviously to code it and to test it”, which is not always true. Even though the students remember the algorithm, it does not necessary mean they understand “why” does it works, “what” precisely happens when it is applied and “when” it shall be best used.

IMAGE PROCESSING TOOLKIT

In this chapter we introduce the basic ideas of the Image Processing Toolkit (IPT), a visualization tool that is meant to concretize various abstract concepts of image processing. The tool and its making are also described in the second part of this chapter. The tool framework and the usability study are presented in the other chapters.
I. Introducing IPT

In principle the IPT interface and structure is quite simple. This is due to various motivations and factors, both of pedagogical and design reason. The IPT is a standalone tool because it is based on the standard Java platform, therefore it is independent of any third party solution or system. Because of its Java nature, IPT can be executed on any platform where Java virtual machine is installed. In addition to this, IPT is a GPL license and therefore is royalty and license free.

The IPT has been designed as open application interface, which allows the plug-ins to be made by any independent developers. All plug-ins are easily added to the toolkit and contribute to enrich its features. Even though the functionalities of IPT would grow, its software dimension remains of reasonable small size. Its small size allows a fast internet download and a fast web-access and start. The ITP does not require any installation and its user interface (which has a fast learning curve) is simple and visually intuitive. The point-and-click user interface is the essence of its functionalities, required to teach and study the IP subjects. Since the IPT does not have a multi-level and context menus, it requires only a very simple logic with WYSIWYG appearance (i.e. “What You See Is What You Get”). Nevertheless a basic level of computer knowledge is required to fully operate the tool with all the teaching/learning benefits.

The IPT is meant for teaching and/or learning image processing, therefore its usability and features are balanced for these scopes. For professional image processing and editing we are well aware that other professional commercial tools, such as MathLab or PhotoShop, could be used. It is the simplicity of IPT that makes it the correct tool for teaching and leanring element of IP university courses. For example, the SDI architecture (one panel with image input and image output and toolbar) allows users to interactively apply all available algorithms to the image. The resulting image can replace the original input image with a single mouse click and then being again processed with a new algorithm. The resulting final and intermediate images can be opened in secondary windows for demonstrations, comparisons, detailed observations or re-use. Therefore the produced effects of the algorithm in the output image can be immediately visualized and compared with the input image. The IPT helps to demonstrate how algorithms and/or their properties differentiate on the image by giving an immediate comparison of the images in the synchronous panning and zooming. The IPT allows conducting experiments with the algorithms by modifying the algorithm parameters, which modify the images. The toolkit visualizes the images as numeric 2D arrays (matrices) and allows modifying the existing algorithms or designing new ones by using the standard Java language.

The standard Java library [16] is well suited for signal and image processing operations [9], which frees the IPT TOOLKIT from dependency on third party solutions. We especially do not abstract from standard image types because we found it essential to discriminate them. Binary, Grayscale and Color image types

![Figure 1. The view on the ITP TOOLKIT, illustrating main application window and enlarged image view.](image)
are supported and are differentiated by each plug-in, which in turn will process only the image types it supports and accordingly to that type. Indeed morphological operation for binary images and grayscale image are different, as well as filtering algorithms. The execution time of algorithms implemented in standard Java is indeed slower that in ImageJ, however is quite sufficient unless the user operates on images larger than 1 Megapixel, which is quite exceptional for demonstration and experimentation needs.

II. The making of ITP TOOLKIT

The ITP is the result of a combined situation of needs. During the course of Image Processing there was a need to explain to the students the concepts of algorithms and image filters and how these are correlated to the image transformation (processing). Students did clearly have difficulties to understand how some generic (to them) concepts like of pixel definition or plug-in applications could determine a change in a picture. So, since the course had also a demo section where students had to conduct some practical work, it was decided to create a software where everyone had to participate creating its various components. This generated fruitful discussions and soon students found themselves involved in one of their most exciting course activities: the creation of what later on became the ITP! Under the supervision of the course lecturer the main points of the to-be software were defined. The ITP should have had a simple and intuitive user interface (see Figure 1), a dual view feature (to allow students to see with their own eyes what were the concrete effects of applied plug-in to an image), the software platform had to be such that its plug-ins could be freely downloaded from internet (hence a software referring to commonly accepted standard plug-ins), the major image format types had to be supported for image processing and, finally, the algorithms had to be customizable for experimentation purposes. Later other additional features were created and added to the original designed software: small windows could pop-up “freezing” a particular image processing phase (for quality comparison reasons or reference), a zooming feature would allow a close-up view of obtained effects on small particulars, a split-and-merge animation capable of “freezing” the process at wanted stages was implemented. Each plug-in can be customized via its properties’ pane, enabling experimentation with the plug-in. Multiple views of the images can be engaged. Views support properties to enlarge the image content and synchronously display the images to ease demonstration and comparison.

A processed image can be easily reused and another operation continued (forming the sequence of operation). At any step the image can be opened in a separate window to form a chain of the action for the visual demonstration (see Figure 6) or for the reuse of the images in further operations. As shown before, views can be enlarged and synchronously panned for detailed observation.

THE IPT FRAMEWORK

The Java technology gives very powerful opportunities to develop complex projects and applications by decomposing them into independent parts that are communicating between each other by the means of well defined interfaces [13]. The architecture of the ITP conforms to the well known design patterns and paradigms in object-oriented programming.

I. Framework architecture

Plug-in based architecture of the ITP lets to easy extend existing functionality with new implemented image processing algorithms. Plug-in is a Java class that implements special interfaces or extends one of two abstract classes.

The first class is a prototype for a simple one-step plug-in where image processing is performed in one step (e.g. simple filtering operation). It receives an image as the input parameter and returns the processed image which is subsequently displayed in the application. The second abstract class is the base class for the animated plug-ins, in which an image processing operation is performed iteratively. Each of the iteration makes some changes to the image resulting in animation in the main application. The plug-in management is a task of main framework that has different User Interface UI elements such as menu or buttons to control workflow of plug-ins execution.

II. Plug-in architecture

The plug-in architecture gives the capability to develop and add new image processing algorithms to the ITP without any changes in the source code of the main framework. Simultaneously, the main framework can be separately updated resulting e.g. in a new updated user interface, extra functionality or new plug-in capabilities accessible to all plug-ins automatically. The process of plug-in development is entirely independent from the main framework and can be performed by any person familiar with basics of Java language [7]. Sample plug-ins are supplied with the framework for illustrative purposes.

The image processing algorithm shall be implemented solely in the main processing method of the plug-in. The main framework communicates with the plug-in using these methods as well as few other methods used to retrieve informative about the plug-in (such as it’s name, description, capabilities, supported image types and such), see Figure 2.
Animated plug-in is a plug-in that performs iterated image processing and visualization of this process at runtime while usual plug-ins only return result image. An example of animated plug-in is a region growing algorithm that is implemented in the framework. This plug-in animates region growing process on the image from seed point on the base of specified rule and relationship between neighbor pixels. Each iteration step consists from image processing and immediate displaying the result image. On each step animated plug-in knows the initial source image (that is given by main framework) and the arbitrary number of result images from previous steps (that plug-in must keep inside itself) to process next iteration. Development of animated plug-ins is independent from the rest of the framework as well as in the case of non-animated one-step plug-ins. Communication protocol between main ITP framework and animated plug-in can be represented by the following steps:

1. Start animation.
2. Is animation complete? If YES – return.
4. Display result image.
5. Go to step 2.

THE USABILITY STUDY

The leading concept of the ITP functionality is quite simple: both lists of source images and plug-ins are maintained by the system for effective operations and transformations. The plug-ins perform different processing operations over images. The task of the framework is to be a manager and a conceptual mediator between the plug-in functionality and the user’s needs and expectations. The wrong doing threshold has been reduced so that it is almost impossible to commit errors. The user interface components will be disabled in the case that an improbable (almost impossible) operative error occurs.

The simple user interface provides intuitive understanding of the functions of the ITP TOOLKIT, allowing an easy use of its integrated tools. As it is mentioned above, the educational purpose of this tool is to help teachers of image processing courses to visualize various image filter algorithms. The interface of the toolkit consists of two toolbars: (main and operations), status-bar, and two viewer panes. The toolbars are used for the selection of the appropriate tools and execution of the plug-ins. The viewer panes display the source and processed image. The status bar contains the information about the processing image and the system state (Figure 5).

The operation cycle starts with the loading of the image in the tool. All loaded or created images in the toolkit can always be accessed using the main toolbar’s combobox. (see Figure 3). Another combobox (on the right) maintains the list of all accessible plug-ins. Depending on whether the selected plug-in supports the processing of the selected image, the entries of the combobox list will be enabled or disabled. The functionality of these comboboxes is very simple and intuitive.

Image viewers are components for displaying the images and allow their resize and scroll. The main idea of having two image viewers is that the left viewer presents the source image for the plug-in processing while the right image viewer always presents the result of the plug-in processing.

The operation toolbar contains buttons for the different actions and operations in the ITP framework (see Figure 4):

1. Pan tool – used for the image movement, using the mouse for simply dragging with automatic synchronization of the images in all open image viewers.
2. Zoom tool – used to obtain a magnified view of the source and result images, which always displays the results of the applied plug-ins in addition to other always-on-top windows.
3. Seed point – a tool for the selection seed point, which is used, for example, in a region growing plug-in.
4. Copy plug-in property window, where the parameters of the image processing algorithm can be always modified.
5. Copy plug-in information window, which describes the plug-in and displays its capabilities and attributes..
6. Copy of the source image in an additional viewer window.
7. Copy target (processed) image into the place of the source image in order to continue the operation cycle.

The intuitive user interface allows, via the use of the few buttons, the management and execution of the plug-ins. RGB, grayscale and binary images are supported in the ITP TOOLKIT but obviously different plug-ins are intended for different image types. There are built-in capabilities to convert one type of image to another one, as, for example, in the case that it would be needed to process a RGB image by “threshold” plug-in but only grayscale types of the image are supported.
Teachers can use the ITP framework in classes to open several parallel views of a processed image and demonstrate the effect that different algorithms (or same algorithm with different parameters) have in details. Similarly, students can repeat teacher’s experiments or try-and-learn new ones. For example, Figure 6 shows a captured sequence of split and merge of one image. A zoom function has been used in the capturing of the given images. A list of the most recent images is kept in the configuration file and displayed in the menu. The tool provides a function to bookmark own favorite images.

Another additional advantage that teachers of Java programming technology can have of this tool is by using it as a framework for getting students’ practical experience feedbacks in developing image processing plug-ins. Novices can study Java programming language and verify their developing know-how and understanding of different image processing algorithms. In addition, the ITP TOOLKIT gives the proper framework for testing and experimenting in image processing algorithms, providing those practical visual feedbacks that easily can be compared to similar algorithms’ outcomes. Each plug-in has a UI property component where different controls for the tuning of the algorithm parameters are placed. Researchers can therefore combine different algorithms and methods trying to obtain new effects. The interfaces and technologies are open, cross-platform and at the current time there are some few dozens of implemented plug-ins and image processing algorithms that can be use as examples for a personalized creative development [1].

**CONCLUSION**

As we have shown in this article, the ITP TOOLKIT is an advanced educational tool that uses intuition and concept correlation to help concretizing abstract concepts related to image processing algorithms. The ITP, from the very beginning of its experimental phases, has proved to be a powerful tool in image processing courses. It has also proven to be an efficient learning framework environment for students wishing to study Java technology. The use of ITP improves the mastering of design applications in object oriented environment and provides new tutoring/learning approaches to this field.

The ITP TOOLKIT is an extendable and cross-platform. Its user interface is simple and its applications are tailored for understanding image processing. The tool has practically worked as a concept convener, helping students to acquire complex ideas of Java technology and image processing. One of the most powerful aspects of IPT is the possibility to verify algorithms’ validity and functionality in image processing. A new and more
refined version of IPT is under-development. This new version is expected to become a solid cornerstone in the image processing courses utilizing visual tools.

ACKNOWLEDGEMENTS

Authors express their gratitude to software developers Vitali Diatchkov and Andrei Mihaila for the framework development.

REFERENCES


http://www.nist.gov/lspix/imlab/labs.html


[16] The Sun Java Series, Detailed information can be found at http://java.sun.com/docs/books/