Proceedings of the ASME 2007 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2007
September 4-7, 2007, Las Vegas, Nevada, USA

DETC2007-35741

INTERACTIVE IMAGE PROCESSING AND MANIPULATION

Stephen S. Nestinger, Graduate Research Assistant
Harry H. Cheng, Professor, ASME Fellow*
Integration Engineering Laboratory
Dept. of Mechanical and Aeronautical Engineering
University of California, Davis
Davis, California 95616
Email: hhcheng@ucdavis.edu

ABSTRACT

Electronic imaging informatics spans a diverse range of applications. These applications would benefit from an interpretive imaging platform, which allows dynamic manipulation and processing of electronic images. Ch is an embeddable C/C++ interpreter that provides an interpretive platform for C/C++ based scripts and programs. Combining Ch with ImageMagick provides the functionality for rapid development of user defined image manipulation and processing applications and scripts. The presented Ch ImageMagick package provides users with the ability to interpretively execute C code based on the ImageMagick C library. This article describes the integration of ImageMagick and Ch. The use of ImageMagick utilities in Ch scripts for rapid prototyping is illustrated. A Web-based example demonstrates the use of Ch and ImageMagick in C based CGI scripting to facilitate the development of Web-based applications involving image manipulation and processing.

1 Introduction

Electronic imaging informatics plays a significant role in today's information era. Imaging applications span a diverse range of fields including industrial visual inspection systems, medical imaging for diagnostics and prognosis, and astrophysics for theory verification. The wide spread use of imaging informatics requires the use of tools to modify and process specific image information. Such tools have been created for medical imaging [1–3], biology [4,5], computer vision [6,7], teaching [8–11], and custom image processing systems [12]. Some tools utilize the Internet as a front-end for image manipulation and processing [13–17].

Imaging technologies typically produce large image sets, which are then processed or manipulated before final delivery or storage. Normally, image processing and manipulation are done through an imaging application with a graphical user interface (GUI) such as Adobe Photoshop or the GNU Image Manipulation Program (GIMP). If done through a GUI-based application, manipulating a large set of images requires a cumbersome process of loading and manipulating each image individually. If the manipulation procedures for an entire image set are the same, it is easier to utilize a script to automatically iterate the procedure on individual images. Some GUI-based image manipulation applications provide limited built-in scripting capabilities. Users and developers not familiar with the specific built-in scripting language may stagnate due to the required learning curve. In addition, end-user needs are highly dynamic and require rapid changes to imaging operations. This requires continuous modification of the application specific internal scripts, which consumes developer time and effort. The use of a scripting environment based on a widely known popular language that supports image processing and manipulation operations would greatly reduce the time consumed during the development phase and in maintenance. This is especially true for developers in

*Address all correspondence to this author.

non-computer science and engineering disciplines who are not used to developing applications in a large number of different scripting languages. The best scripting environment would be a C based interpretive platform. C is a standard programming language used in all fields of science and engineering.

There are a few available image manipulation and processing application programming interfaces (APIs) based on C/C++. VXL [18] provides a collection of modular libraries for image manipulation and processing, one of which is the VIL C++ core library for loading, saving, and manipulating images in many common file formats. The Microsoft Vision SDK [19] is a library for writing programs to perform image manipulation and analysis on computers running Microsoft Windows operating systems. DIPlib [20] is a scientific image-processing C library that contains a large number of functions for image manipulation and processing. Gandalf [21] is a C based computer vision and numerical library with an image package that provides low-level image manipulation routines. LEADTOOLS [22], by Aurora Co., provides a set of commercial packages like the Raster Imaging Pro SDK, which contains a low level API, C++ Class Library and .NET Class libraries for image manipulation and processing. The Ch OpenCV package [7], a Ch binding to the popular OpenCV [23] library, was created to supply developers with interactive open architecture computer vision. Paintlib [24] is a portable C++ class library for image loading, saving and manipulation. However, these libraries and packages do not provide a set of utilities for rapid image application prototyping. Some of the packages are platform specific, which restricts developers, while others are relatively complicated and require a steep learning curve. An optimum image manipulation and processing library is one that not only provides robust utilities that can be quickly used in scripts, but also a C/C++ backend with the same functionalities as the utilities for application development.

ImageMagick is a widely used set of image manipulation utilities that also provides a C/C++ API backend [25]. ImageMagick provides a robust collection of tools and libraries for writing, reading, and manipulating images. It incorporates many different image formats including the popular JPEG, PNG, PDF, SVG, TIFF and others. Utilizing the ImageMagick API in a C based scripting environment provides users with the dynamic image functionality they require and is useful for applications dealing with a large set of images.

In lieu of the desired functionality of an open environment image manipulation platform, the Ch ImageMagick package has been developed [26]. Ch is an embeddable C/C++ cross-platform interpreter. Use of Ch ImageMagick enhances the portability of the ImageMagick API. Typically, the use of ImageMagick API based applications on heterogeneous platforms requires recompiling the applications for each platform. Since Ch is platform independent, use of Ch ImageMagick makes user code truly portable across different platforms and readily to run without compilation. Ch also provides powerful numerical features that further enhance image manipulation and processing applications. These features are comparable to those in MATLAB [27] and Mathematica [28], which also provide image processing and manipulation toolkits. However, Ch provides these features in the framework of C, which makes C based vector and matrix operations more concise.

This article describes the use of the ImageMagick utilities in Ch and the creation of the Ch ImageMagick package. Two examples of using ImageMagick in Ch are given. The first example uses Ch's plotting capabilities, GTK, and ImageMagick to produce a GUI-based interactive plotting application. Users can modify the plotting algorithm and display an updated plot. The second example utilizes the Ch CGI toolkit with the ImageMagick utilities. The example demonstrates the ability to create a Web-based user interface for the dynamic modification of images. Users can upload a file and choose which modification and processing operations to apply. The resulting image is then displayed in a dynamically generated Web page.

2 Using the ImageMagick Utilities in Ch Scripts

Ch is an open architecture embeddable C/C++ interpreter originally developed by Cheng [29] for cross platform scripting [30], and numerical and embedded computing [31, 32]. Ch is an extension and enhancement of the most popular Unix/Windows C computing environment. Being platform independent, a Ch program developed on one platform can be easily executed on a different platform without the need of recompiling and linking.

Many features first implemented in Ch have been added to the latest C standard called C99 [33]. These features include, complex numbers, variable-length array (VLA), binary constants, and function name __func__. Ch also supports computational arrays as first-class objects as in Fortran 90 and MATLAB for linear algebra and matrix computations. Furthermore, Ch supports all features of the ISO C90 standard ratified in 1990 [34]. Computational arrays simplify the implementation of intensive computational image manipulation algorithms. Ch is especially suitable for Web-based client/server computing. Ch programs can be used through a common gateway interface (CGI) in a Web server and as dynamic applets executed through a Web browser.

As a scripting environment, Ch is a high-level software development language based on the standard C language that allows for quick and easy development and maintenance of trivial tools. Using Ch as a scripting language allows for the gluing together of individual modules for rapid application prototyping. Ch users can directly invoke system utilities that would require many lines of code in order to be implemented otherwise. Once the rapid prototyping phase has been completed, the commands and utilities in a Ch script can be replaced with equivalent C code. The code can then be run interpretively within Ch. This allows for the rapid testing of applications before they are com-

Table 1. Utilities provided with ImageMagick.

Utility	Description
animate	animate an image sequence on any X server.
compare	mathematically and visually annotate the difference between an image and its reconstruction.
composite	overlap one image over another.
conjure	interpret and execute scripts written in the Magick Scripting Language (MSL).
convert	image format conversion as well as resize, blur, crop, despeckle, dither, draw on, flip, join, and much more.
display	display an image or image sequence on any X server.
identify	describe the format and characteristics of one or more image files.
import	save any visible window on an X server and output it as an image file.
mogrify	resize an image, blur, crop, despeckle, dither, draw on, flip, join, re-sample, and much more.
montage	create a composite image by combining several separate images.
stream	a lightweight tool to stream pixel components of an image or portion of an image to any storage format.

piled, thus saving development time and effort. After the tests are complete and if desired, users can then compile their code for optimized performance.

ImageMagick provides a collection of utilities for writing, reading, and manipulating images. A list of these utilities and a summary of what they can do is provided in Table 1. These ImageMagick utilities can be used in a Ch script for the rapid development of an image manipulation and processing application. If necessary, ImageMagick utilities can be replaced using the ImageMagick C API, which contains all of the functionalities of the corresponding utility, and compiled for optimization.

3 Ch ImageMagick

Ch ImageMagick is a Ch binding to the ImageMagick C library. It makes all of the ImageMagick C library functionalities available in a C script. A list of functionalities of the ImageMagick C library is given in Table 2. Embedding Ch in graphics applications allows users or developers to dynamically generate and manipulate graphics at run-time. The Ch Software Development Kit (SDK) allows for the porting of C libraries and functions into Ch space by creating Ch packages that bind to the C space libraries and functions [35]. To call a C function in the ImageMagick C library in Ch, a wrapper function should be created first. The wrapper function consists of two parts: a *chf* function in Ch space and a *chdl* function in C space. The chdl functions are compiled into a dynamically loaded library, which is loaded during the execution of an image manipulation or processing program or script. The concept behind Ch ImageMagick is shown in Figure 1 where the function named *NewMagickWand()* in the ImageMagick C library is called by a Ch script or program in Ch space.

To invoke the function *NewMagickWand()*, Ch searches for the equivalent chf file, *NewMagickWand.chf* in this case, and passes the proper arguments to it. The *NewMagickWand.chf* searches through the dynamically loaded library for the equivalent chdl function, *NewMagickWand_chdl()*, and passes arguments to it. The *NewMagickWand_chdl()* function then invokes the *NewMagickWand()* function in the ImageMagick library. Any return arguments are passed back to the initial function call in Ch space.

Table 2. ImageMagick image manipulation features.

Function	Description
Format	convert an image to different formats
Transform	resize, rotate, crop, flip or trim an image
Transparency	render portions of an image invisible
Draw	add shapes or text to an image
Decorate	add a border or frame to an image
Special effects	blur, sharpen, threshold, or tint an image
Image calculator	apply a mathematical expression to an image or image channels
Text	insert descriptive or artistic text
Identification	acquire image format and attributes
Animation	create a GIF animation sequence from a group of images
Composite	overlap one image over another
Montage	juxtapose image thumbnails on an image canvas
Large image	read, process, or write mega- and giga-pixel image sizes

Once the libraries and functions have been ported over to Ch, there are three different ways that users can utilize the available image manipulation functions from the C space library. The first way is to take advantage of Ch's ability to interpret C/C++ at the prompt. Users can simply type in function names and commands at the prompt. This allows users to quickly try out certain application functionalities. The second way is to utilize Ch's ability to interpret C/C++ files by placing the user's interactive code into a C/C++ file and running that file as a program. C/C++ files can be readily run at the prompt by typing in the name of the file. This allows users to quickly modify chunks of code or a single

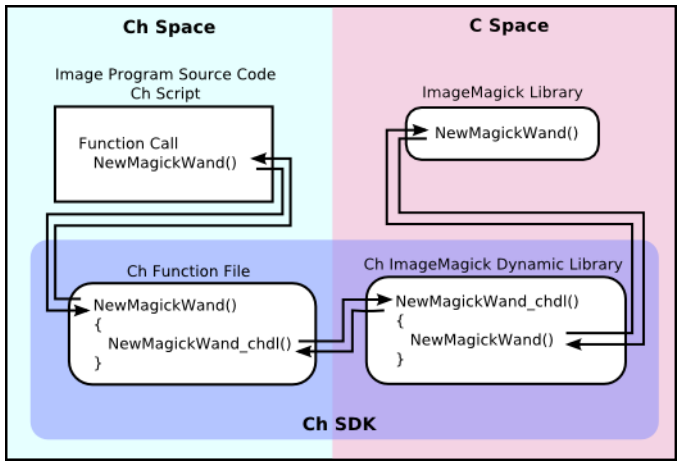

Figure 1. Concept of Ch SDK with Ch ImageMagick.

line of code and generate results without the need of recompiling and linking. Since Ch is also a shell, system command calls can also be placed in the file. The third way is to then compile their program into a binary executable, which allows users to further optimize their application if needed.

Ch has a wide variety of packages and toolkits that can be used along side Ch ImageMagick to further enhance user applications. With the Ch CGI Toolkit [36], Web-based applications can dynamically modify or process an image through the Internet. Web-servers can use Ch based scripts to quickly convert, scale or create user images on the fly, which keeps all of the lower level details away from the user. With the Ch GTK+ toolkit [37], users can create graphical user interfaces to their application using ImageMagick as a backend for image manipulation.

4 Sample Interactive Image Manipulation Applications

This section describes a few interactive image manipulation applications that demonstrate the practical use of the ImageMagick utilities and C library in Ch.

4.1 Interpretive Execution of Image Manipulation Programs

A user specific image manipulation application can be created in several ways. One method is to utilize Ch as a shell environment by which utilities can be readily invoked in a Ch script to manipulate images. The second method is to utilize Ch ImageMagick and make function calls directly to the ImageMagick API. Ch ImageMagick supports all C functions available from the ImageMagick library. A C/C++ program using these functions can be readily treated as a Ch script. The program can be directly executed in different command shells and integrated development environments (IDEs) without compilation and linking. If desired, the functions can be called directly within a Ch shell. This allows developers to quickly try out user defined functions or a procedure of functions before placing them into a file. Once the procedure is complete, the required sequence of functions can be placed into a file and run as a Ch script.

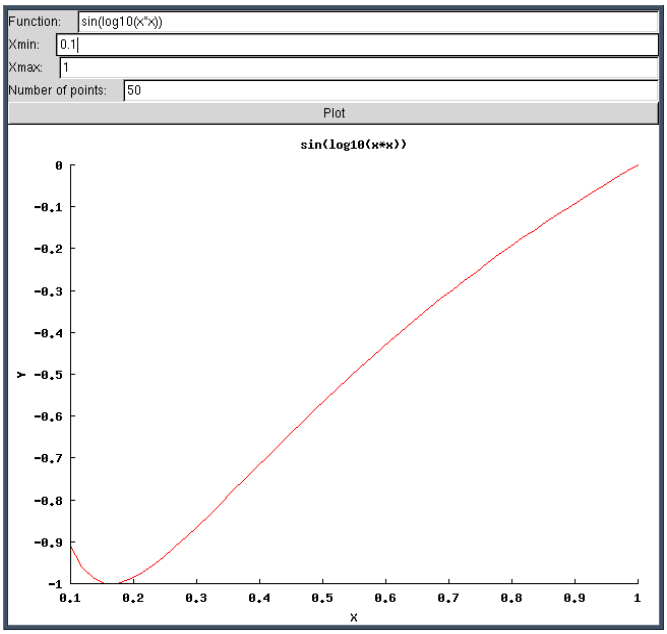

Figure 2. A graphical user interface for plotting functions in Ch.

```
...
plot.data2D(x,y);
plot.outputType(..., "plot.png");
plot.plotting();
convert plot.png plot.xpm
// or system("convert plot.png plot.xpm");
pixmap =
   gdk_pixmap_create_from_xpm(..., "plot.xpm");
gdk_draw_pixmap(...);
...
```

Figure 3. Using ImageMagick utilities for image conversion in a Ch script.

Figure 2 shows a graphical user interface for plotting functions launched by a Ch program using GTK. The application shows a common use of image manipulation in which images are converted into a different format in order to be utilized. The application uses Ch's plotting capabilities to produce a PNG format image, which is then converted into a image format that can be handled by GTK's drawable interface. A snippet of the code used is shown in Figure 3.

```
MagickWand *mw;
mw = NewMagickWand();
MagickReadImage(mw,"plot.png");
MagickWriteImages(mw,"plot.xpm", MagickTrue);
mw = DestroyMagickWand(mw);
```

Figure 4. Using ImageMagick C library for image conversion in a Ch script.

Two methods can be used to complete the conversion from the PNG to XMP format. The first method is to utilize ImageMagick's *convert* utility as a shell command as shown below.

```
convert plot.png plot.xpm
```

The interpretive execution of the application first creates the plot in the PNG file *plot.png* and then converts it to a XPM format by invoking *convert* as a shell command as shown in Figure 3. The second method uses the MagickWand API available in the ImageMagick library. The program in Figure 4 can be used to replace the *convert* command in Figure 3. The code first declares a MagickWand object pointer, which is initialized using the call to the *NewMagickWand()* function. The desired image is then read in using the function *MagickReadImage()* and written out in a different format with the function *MagickWriteImage()*. The MagickWand object is then destroyed.

4.2 Ch and ImageMagick Web applications

This section introduces a Web-based image system using Ch and ImageMagick. It also presents how to implement Web-based image manipulation and processing applications based on the ImageMagick utilities, Ch, Ch ImageMagick, and Ch CGI.

4.2.1 Implementation of Ch and ImageMagick Web-based interfaces Using the ImageMagick utilities, Ch, Ch ImageMagick and Ch CGI, Web-based interactive image manipulation and processing applications can be dynamically created and easily implemented. Figure 5 gives an overview of the Web-based image manipulation concept.

The implementation of the Ch and ImageMagick Web-based interface is based on the use of a standard HTTP Web server. By using a standard Web server, there is no need to develop a specific in-house image manipulation and processing server daemon. HTTP based systems are coordinated through a server client compliance by which client requests are sent to the server as HTML documents. Parameters encoded in HTML documents are extracted by a Ch CGI script, which causes the invocation of corresponding Ch ImageMagick scripts or programs. In a Web-based image manipulation and processing system, clients generally upload an image or a set of images and specify the manipulation and processing procedures to be applied to the image or the set of images. Ch CGI scripts are then invoked with the resulting image or images displayed in a Web browser.

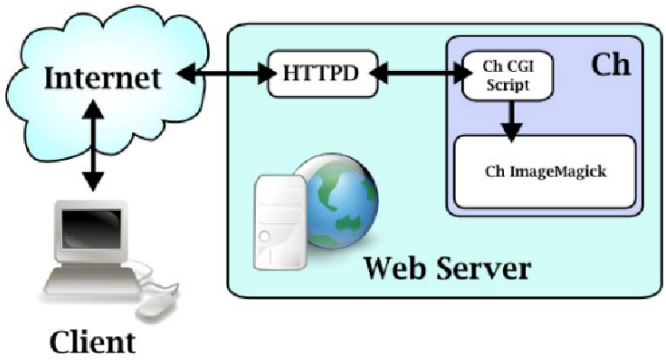

Figure 5. Interactive Web-based image manipulation application.

A CGI enabled Web server is able to receive client requests, execute applications on the server, and send execution results back to the client. HTML interactions are based on static information. CGI allows Web servers to output dynamic Web-based information according to client requests. CGI applications can be written in any language allowed by the host. Scripting languages are preferred based on the ability to quickly modify and easily debug and maintain scripts as compared to compiled programs. With Ch, C/C++ programs become scripts. The Ch CGI toolkit provides four easy-to-use classes, CResponse, CRequest, CServer, and CCookie, which provide member functions similar to the API in active server page (ASP) and Java server page (JSP). Member functions of these classes can be directly integrated into image manipulation and processing applications. The transfer of user required input is accomplished through a fill-out form on the Web page. The user parameters are extracted by Ch CGI programs and directly passed to image manipulation and processing programs. The image manipulation and processing programs then generate dynamic Web pages with the desired results. A Ch CGI script using Ch ImageMagick or the ImageMagick utilities can be created to allow for on-line image manipulation and processing.

4.2.2 Web-based interactive image manipulation and processing An example of a Web-based manipulation and processing interface using Ch and ImageMagick is shown in Figure 6. The source code for this example is available on the Web [38]. The Web-based interface is simple and can be accessed by any user with access to a Web browser. The process starts with the user uploading an image they would like to manipulate or process. In the example, the file chosen is called *lenna.tif*. Afterward, the user can then choose from a list of manipulation options. The options available through the Web-based interface are listed in Table 3 and are only a selected few of the available features in ImageMagick.

The example shows that the convert option was selected to convert the image from the input format of TIFF to an output

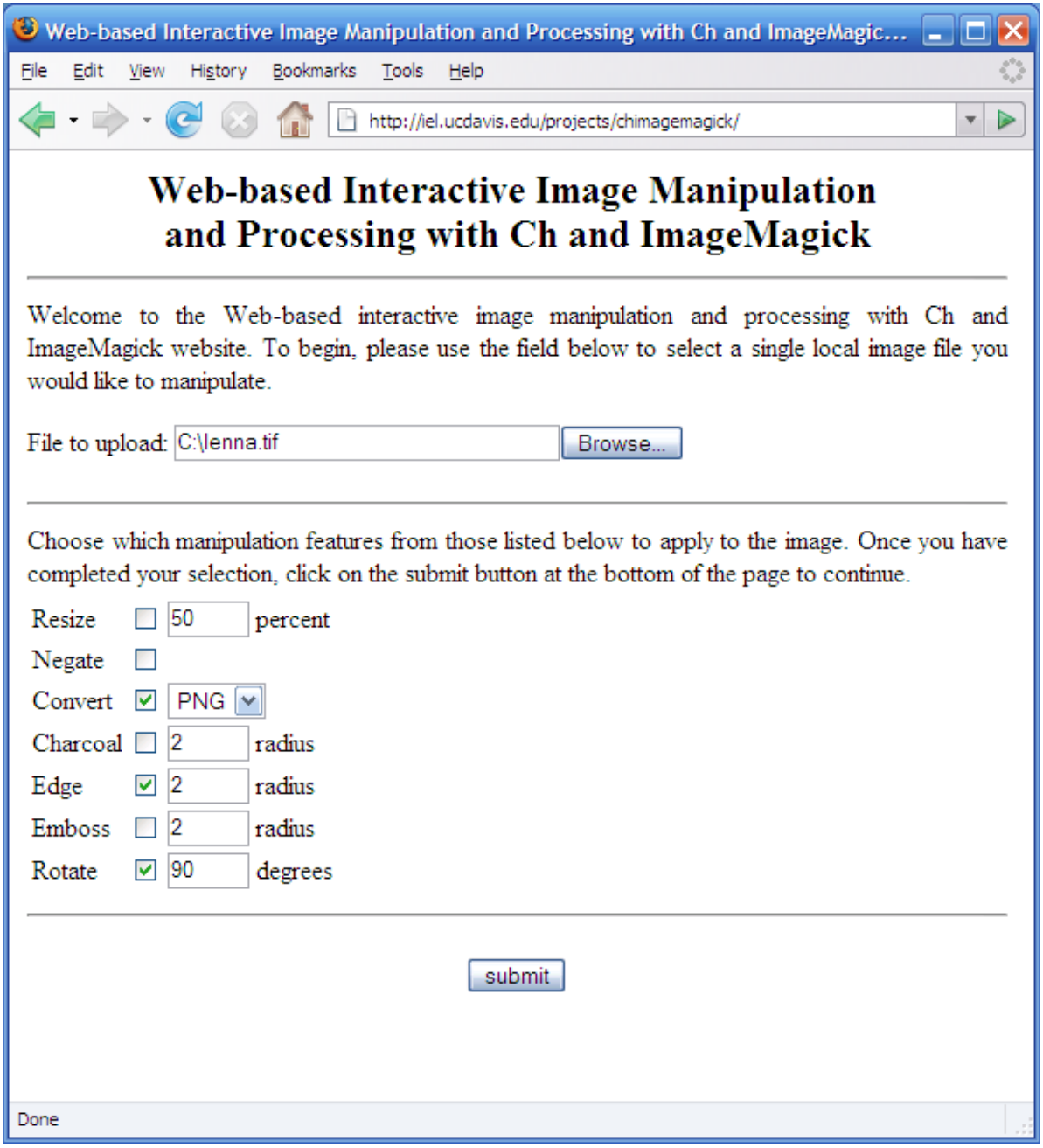

Figure 6. Interactive Web-based image manipulation and processing upload form.

format of PNG, the edge option was selected to detect edges using a radius of 2 pixel counts, and the rotate option was selected to rotate the image by a positive 90 degrees. The selected options and there arguments, if any, are embedded in the HTML as parameters. When the user clicks on the submit button at the bottom of the page, these parameters are encoded by the client browser and sent to the HTTP server. The HTTP server then decodes the parameters via the Ch CGI program using the Ch CGI member function CRequest::getFormNameValue(). Since the Ch CGI toolkit is a set of C++ classes, CGI programs can be easily integrated with ImageMagick so that the system parameters can be directly passed to applications without any additional interface. The activation of the CGI program commences the image manipulation and processing according to the parameters specified. The CGI program then dynamically creates a Web page to display the resulting image. Different manipulation procedures are chosen and the resulting output is shown in Figure 7. The dynamically created output Web page gives the command used to

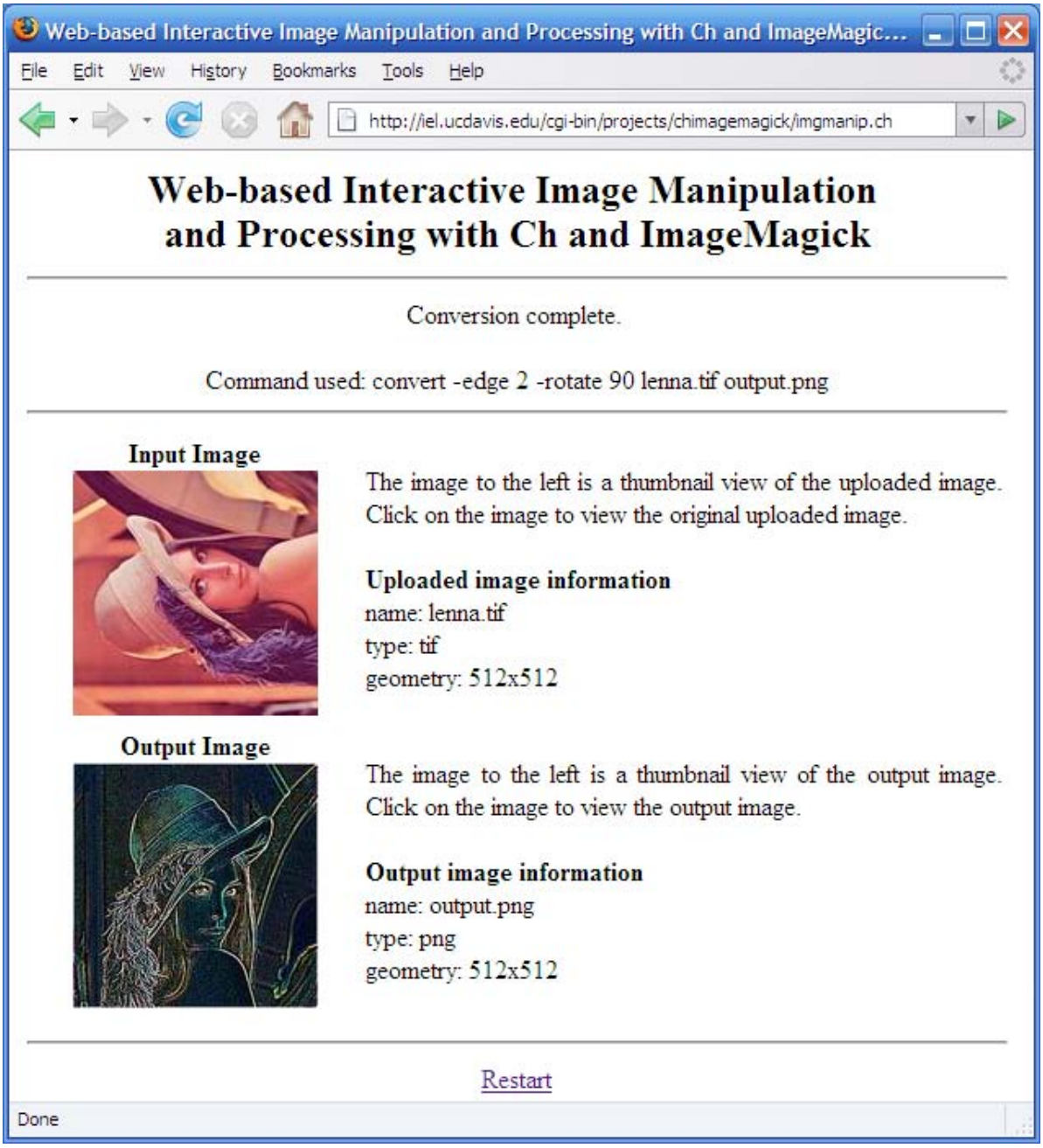

Figure 7. Interactive Web-based image manipulation and processing results.

manipulate and process the uploaded image from the user. The command used is given as follows.

```
convert -edge 2 -rotate 90 lenna.tif output.png
```

The command indicates that the *convert* ImageMagick utility was invoked to manipulate and process the input image *lenna.tif* and the output was written to *output.png*. The other options applied to the image were edge and rotate with arguments of 2 and 90, respectively. Using commands directly in a CGI script, Ch simplifies the development of CGI scripts and Web-based applications.

A snippet of the CGI code is shown in Figure 8. After *convert* is invoked to apply the manipulation procedures imposed by the client, a thumbnail view of the resulting image is dynamically generated based on the size of the output image. The CGI code uses the ImageMagick utility *identify* to get the width of the out-

Table 3. Options provided by the Web-based image manipulation and processing form.

Option	Description
resize	resize the image based on a percentage entered by the user
negate	invert the image or reverses all of the colors in the picture
convert	convert the file to a desired format
charcoal	change the image to simulate a charcoal drawing
edge	enhance edges within the image with a convolution filter of a given radius
emboss	emboss the image
rotate	apply image rotation based on the degrees entered by the user

```
...
value = `identify -format "%w" $outname`;
if(atoi(value) > 150)
  convert -resize x150 $outname thumbnail2.jpg
else
  convert $outname thumbnail2.jpg
...
```

Figure 8. Sample of the CGI code used in the We-based application.

put image. The name of the output image is stored in the variable *outname* and referenced as a script variable using the symbol '$'. The output image width is obtained using the *format* option passing it a format argument of "%w" where '%w' is a modifier for image width. The output from the shell command is stored in the string variable *value* and used in the *if* statement by passing it through *atoi()*. If the width of the output image is greater than 150 pixels, the ImageMagick utility *convert* is used to resize the output image to have a width of 150 pixels. The height of the image is scaled automatically using the original aspect ratio. If the output image has a width less than 150 pixels, the image is not resized. In either case, the *convert* utility is used to convert the output image into a JPG file called *thumbnail2.jpg*. The thumbnail view is then used in the output Web page and linked to the output image for viewing.

5 Conclusions

This article described the utilization of ImageMagick's image manipulation and processing functionalities in Ch. The open source Ch ImageMagick package was introduced. It supports all C functions available in the ImageMagick C library. A C/C++ program using these functions can be readily treated as a Ch script and run directly from the Ch shell without the need of compiling and linking. The functionality of the Ch ImageMagick package was further enhanced by using other available Ch packages. Since Ch ImageMagick is scriptable, embeddable, and truly platform independent, it is a convenient tool for the rapid prototyping of image manipulation and processing applications, Web-based applications, and ideal for teaching image processing and manipulation.

Ch, Ch ImageMagick, and Ch CGI are freely available and can be downloaded from the Internet [29]. A methodology for the implementation of Web-based image manipulation and processing systems based on Ch, ImageMagick, and Ch CGI was introduced in this article. The method described in this article can be followed to produce a low cost and easy to maintain Web-based image manipulation and processing system. Ch with ImageMagick gives users vast versatility and quick image manipulation program prototyping capabilities.

REFERENCES

[1] Lobo-Stratton, G., Mercer, T., and Polman, R., 2006. "Patient exam data reconciliation tool". *Journal of Digital Imaging,* **19**, pp. 60–65.

[2] Crane, J. C., Crawford, F. W., and Nelson, S. J., 2006. "Grid enabled magnetic resonance scanners for near real-time medical image processing". *Journal of Parallel and Distributed Computing,* **66**(12), Dec, pp. 1524–1533.

[3] Shen, H., Nelson, G., Nelson, D. E., Kennedy, S., Spiller, D. G., Griffiths, T., Paton, N., Oliver, S. G., White, M. R. H., and Kell, D. B., 2006. "Automated tracking of gene expression in individual cells and cell compartments". *Journal of the Royal Society Interface,* **3**(11), Dec, pp. 787–794.

[4] Philippsen, A., Schenk, A. D., Signorell, G. A., Mariani, V., Berneche, S., and Engel, A., 2007. "Collaborative EM image processing with the IPLT image processing library and toolbox". *Journal of Structural Biology,* **157**(1), January, pp. 28–37.

[5] Hohn, M., Tang, G., Goodyear, G., Baldwin, P., Huang, Z., Penczek, P. A., Yang, C., Glaeser, R. M., Adams, P. D., and Ludtke, S. J., 2007. "SPARX, a new environment for Cryo-EM image processing". *Journal of Structural Biology,* **157**(1), January, pp. 47–55.

[6] Biancardi, A., Cantoni, V., Codega, D., and Pini, M., 1997. "An interactive tool for C.V. tutorials". In Proceedings of the Fourth IEEE International Workshop on Computer Architecture for Machine Perception (CAMP '97), pp. 170–174.

[7] Yu, Q., Cheng, H. H., Cheng, W. W., and Zhou, X., 2004. "Ch OpenCV for interactive open architecture computer vision". *Advances in Engineering Software,* **35**(7-8), pp. 527–536.

[8] Mason, T. P., Applebaum, E. L., Rasmussen, M., Millman, A., Evenhouse, R., and Panko, W., 2000. "Virtual temporal bone: Creation and application of a new computer-based

teaching tool". *Otolaryngology-Head and Neck Surgery,* **122**(2), February, pp. 168–173.

[9] Bogdanova, I., Vandergheynst, P., and Kunt, M., 2002. "Virtual classroom for multimedia teaching on WWW". In Proceedings of the 32nd ASEE/IEEE Frontiers in Education Conference.

[10] Bamberger, R. H., 1994. "Portable tools for image processing instruction". In Proceedings of the IEEE International Conference on Image Processing (ICIP-94), Vol. 1, pp. 525–529.

[11] Chan, K. L., 1992. "An interactive image processing system". In Proceedings of the IEEE 'Communications on the Move' (Singapore ICCS/ISITA '92), Vol. 1, pp. 366–369.

[12] Martin, S., and Alves, J. C., 2005. "A high-level tool for the design of custom image processing systems". In Proceedings of the IEEE 8th Euromicro conference on Digital System Design (DSD'05).

[13] Young, N., Chang, Z., and Wishart, D. S., 2004. "GelScape: a Web-based server for interactively annotating, manipulating, comparing and archiving 1D and 2D gel images". *Bioinformatics,* **20**(6), April, pp. 976–978.

[14] Bozinov, D., 2003. "Autonomous system for Web-based microarray image analysis". *IEEE Transactions on Nanobioscience,* **2**(4), December, pp. 215–220.

[15] Zeng, H., Fei, D. Y., Fu, C. T., and Kraft, K. A., 2003. "Internet (WWW) based system of ultrasonic image processing tools for remote image analysis". *Computer Methods and Programs in Biomedicine,* **71**(3), July, pp. 235–241.

[16] Drap, P., and Grussenmeyer, P., 2000. "A digital photogrammerty workstation on the WEB". *ISPRS Journal of Photogrammetry and Remote Sensing,* **55**(1), February, pp. 48–58.

[17] Chen, G., Yi, H., and Ni, Z., 2005. "MIPP: A Web-based medical image processing system for stent design and manufacturing". In Proceedings of the IEEE Interntional Conference on Services Systems and Services Management (ICSSSM '05), Vol. 2, pp. 1484–1488.

[18] *VXL*. http://vxl.sourceforge.net.

[19] *Microsoft Research Center*. http://research.microsoft.com/vision.

[20] *DIPlib*. http://www.ph.tn.tudelft.nl/DIPlib.

[21] *Gandalf*. http://gandalf-library.sourceforge.net.

[22] *LEADTOOLS*. http://www.leadtools.com.

[23] INTEL. *Open Source Computer Vision Library (OpenCV)*. http://www.intel.com/technology/computing/opencv.

[24] *paintlib*. http://www.paintlib.de/paintlib.

[25] ImageMagick. http://www.imagemagick.org.

[26] ChMagick. http://www.imagemagick.org/ChMagick.

[27] MATHWORKS, INC. *MATLAB*. http://www.mathworks.com.

[28] WOLFRAM RESEARCH, INC. *Mathematica*. http://www.wolfram.com.

[29] *Ch — an Embeddable C/C++ Interpreter*. http://www.softintegration.com.

[30] Cheng, H. H., 2006. "Ch: A C/C++ interpreter for script computing". *C/C++ User's Journal,* **24**(1), Jan., pp. 6–12.

[31] Cheng, H. H., 1993. "Scientific computing in the Ch programming language". *Scientific Programming,* **2**(3), Fall, pp. 49–75.

[32] Cheng, H. H., 1993. "Handling of complex numbers in the Ch programming language". *Scientific Programming,* **2**(3), Fall, pp. 76–106.

[33] ISO/IEC, 1999. *International Standard: Programming languages - C*. Geneva, Switzerland.

[34] ISO/IEC, 1990. *International Standard: Programming languages - C*. Geneva, Switzerland.

[35] SoftIntegration. *The Ch Language Environment – SDK User's Guide*. SoftIntegration, Inc. http://www.softintegration.com.

[36] SOFTINTEGRATION, INC. *Ch CGI Toolkit*. http://www.softintegration.com/products/toolkit/cgi/.

[37] SOFTINTEGRATION, INC. *Ch GTK+ Toolkit*. http://www.softintegration.com/products/toolkit/gtk/.

[38] *Ch ImageMagick Example Website*. http://iel.ucdavis.edu/projects/chimagemagick.