ELSEVIER

Advances in Engineering Software 35 (2004) 527–536

ADVANCES IN ENGINEERING SOFTWARE

www.elsevier.com/locate/advengsoft

Ch OpenCV for interactive open architecture computer vision

Qingcang Yu[a], Harry H. Cheng[a,*], Wayne W. Cheng[b], Xiaodong Zhou[b]

[a]*Integration Engineering Laboratory, Department of Mechanical and Aeronautical Engineering, University of California, One Shields Avenue, Davis, CA 95616, USA*
[b]*SoftIntegration, Inc., 216 F Street, 68 Davis, CA 95616, USA*

Received 17 May 2003; received in revised form 29 April 2004; accepted 21 May 2004

Available online 24 July 2004

Abstract

In this paper, design and implementation of an interactive open architecture computer vision software package called Ch OpenCV is presented. Benefiting from both Ch and OpenCV, Ch OpenCV has many salient features. It is interactive, capable of interface with binary static or dynamical C/C++ libraries, integrated with advanced numerical features and embeddable. It is especially suitable for rapid prototyping, web-based applications, and teaching and learning about computer vision. Applications of Ch OpenCV including web-based image processing are illustrated with examples.

Keywords: C/C++ interpreter; Ch; Open CV; Computer vision; Image processing

1. Introduction

In the past decades, the exponential growth of processor speed and memory capacity has led to dramatic broadening of research areas and its applications in computer vision. A considerable number of computer vision and image processing software packages have been developed for various applications. For computational speed, most of these software packages are written in C/C++. For example, TargetJr [1] and its successor VXL [2] are collections of C++ libraries, which provide a modular and portable platform for development of vision algorithms. Gandalf [3] is a computer vision and numerical library, which allows users to develop new portable applications. Manufactured by MVTec, HALCON is a commercial computer vision tool consisting of an image processing library with C and C++ interfaces [4]. The Delft Scientific Image Processing Library (DIPlib) [5] is another scientific image-processing C library. It contains a large number of functions for processing and analyzing multi-dimensional image data.

Many commercial image processing and computer vision software packages have also been developed. They provide high-level image processing and machine vision functions and display tools. Some software packages, such as LabView [6], can be accelerated with special image processing hardware. Developed by Microsoft, the vision SDK is a low-level C++ library for image manipulation and analysis [7]. Image processing toolkit in MATLAB [8] and digital image processing package in Mathematica [9] are convenient image processing tools, taking the advantage of many advanced numerical functions at the same time. Developed by Aurora Co., LEADTOOLS [10] is a set of commercial computer vision packages. It includes an Image Server, which can be used to build a web-based image processing server. But they do not include advanced functions such as moving objects tracking, pose recognition, face recognition and 3D reconstruction, etc. And it is complicated to interface with existing computer vision library and code in C/C++.

We have developed Ch OpenCV package for interactive open architecture computer vision [11]. Ch OpenCV is open source and freely available for downloading from the Internet. In this article, integration of Ch and OpenCV will be presented. First, we will outline the overview of OpenCV and Ch. Then we will highlight the salient features of Ch OpenCV package. Finally, we will present design, implementation, and application examples of Ch OpenCV.

* Corresponding author. Tel.: +1-530-752-5020; fax: +1-530-752-4158.
E-mail address: hhcheng@ucdavis.edu (H.H. Cheng).

0965-9978/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.advengsoft.2004.05.003

2. OpenCV and Ch

Recently, Intel Microprocessor Research Lab has developed an Open Source Computer Vision Library (OpenCV for short) [12,13] distributed under a BSD style license which allows for royalty free commercial or research use with no requirement that the user's code be free or open. OpenCV is supported under Windows and Linux, but the code is well behaved and has ported to many other operating systems. OpenCV contains an optimized collection of C libraries spanning a wide range of computer vision algorithms, including motion segmentation and pose recognition [14], multi-projector display system [15], object and face recognition, and 3D reconstruction, etc. The broad functional areas supported by OpenCV include:

- Basic structures and array manipulations.
- Image processing and analysis.
- Object structural analysis.
- Motion analysis and object tracking.
- Object and face recognition.
- Camera calibration and 3D reconstruction.
- Stereo, 3D tracking and statistically boosted classifiers.
- User interface and video acquisition support.

Ch is an interpreter that provides a superset of C with salient extensions [16]. Ch supports all features in the ISO 1990 C standard (C90). Existing C code can be executed in Ch without any modification and compilation. Ch supports many new features in C99 such as complex numbers, variable-length array, binary constants, IEEE 754 floating-point arithmetic, generic functions [17], and function name __func__. In addition, Ch provides a very high-level language environment and it is object-based. Ch supports classes, objects, and encapsulation in C++ for object-based programming with data abstraction and information hiding, as well as simplified I/O handling. Furthermore, Ch provides a universal shell for convenience and ease of use. It can be used as a login command shell similar to C-Shell, Bourne shell, Bash, tcsh, or Korn shell in Unix, as well as the MS-DOS shell in Windows. Ch has many built-in enhanced features for shell programming to automate repetitive tasks, rapid prototyping, regression testing, and system administration across different platforms. Ch is freely available [16].

3. New features of Ch OpenCV

Integrating Ch with OpenCV, Ch OpenCV extends OpenCV with the following salient features for computer vision.

Interactive. With Ch OpenCV, C/C++ programs with OpenCV can be executed interpretively without compilation interactive execution of C programs without the tedious edit/compile/link/debug cycle is especially appealing for rapid application development and deployment. Furthermore, the interpretive execution of programs without byte-code provides a potential use of mobile code in computer vision.

Unlimited libraries. Ch can seamlessly integrate different components. All existing C libraries and modules can be part of the Ch libraries using Ch SDK [18]. Therefore, the potential of Ch libraries is almost unlimited. All the previously mentioned computer vision packages are complementary to Ch OpenCV. This greatly enhances the ability and broadens the application areas available to OpenCV.

Powerful numerical computing. Ch is powerful in numerical computation. Many high-level numerical functions such as differential equation solving, integration, Fourier analysis, and 2D/3D plotting make Ch a very powerful language environment for solving engineering and science problems. This provides OpenCV with additional powerful numerical features for computer vision.

Web enabled. Like ASP and Java servlet, Ch CGI toolkit contains four classes named CRequest, CResponse, CServer and CCookie for the Common Gateway Interface (CGI) in web servers [19] Ch allows rapid development and deployment of web-based applications and services. It simplifies the implementation of web-based computer vision.

Embeddable. Unlike C/C++ compilers, Ch can be embedded as a scripting engine in C/C++ applications and hardware [20]. It relieves users from the burden of developing and maintaining a macro language or interpreter for many applications. Because of this advantage, Ch OpenCV is applicable in embedded computer vision.

4. Integration of Ch with OpenCV

4.1. Interfacing C libraries from Ch space

All existing binary static and dynamic C libraries and modules can be imported to Ch. Because the Ch space in scripting and C space in binary library have their own name spaces, a function in the C space cannot communicate directly with a function in the Ch space. However, by using a dynamically loaded library (DLL), a Ch program is able to extend its address space from the Ch address space to the binary C address space during execution, and call functions in the static or dynamic libraries.

To call a C function in static or dynamical library from Ch, it is necessary to create a wrapper function that can link the Ch and the underlying C function. A wrapper function must be able to do the following three things:

- It can be called from Ch and pass arguments to the C function in the library.
- It can call the function in the library.
- It can return a value from the function in the library to Ch.

In Ch, a wrapper function consists of a `chf` function in Ch space and a `chdl` function in C space. The interface of

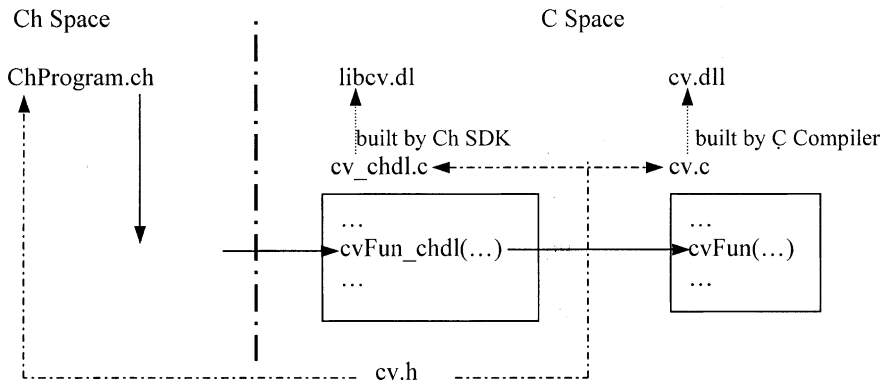

Fig. 1. Files and functions call in Ch and C spaces.

Ch to functions in binary libraries is illustrated in Fig. 1. In Fig. 1, a function named `cvFun ( )` in the dynamically linked library `CV.dll` which is built by a C compiler is called by a Ch program named ChProgram.ch.

To invoke the function cvFun( ) in a library, Ch first searches for a chf file with the same name as the function and file extension.chf, that is cvFun.chf in this case, and passes the proper arguments to it. The `cvFun.chf` file searches for a dl file, which contains `cvFun_chdl ( )` function and passes arguments to it. In `cvFun_chdl ( )` function, function `cvFun ( )` in the library will be invoked. Details about these functions and related files are described below.

4.2. Program in Ch space

With Ch OpenCV, the same C code using functions in the OpenCV library run in Ch without any modification. From an application developer's point of view, developing computer vision applications in C and Ch is the same. However, certain files originally developed for OpenCV need to be modified and added for Ch OpenCV. These files are described in this section.

4.2.1. Header file

The same header file can be used in both Ch and C spaces. But in Ch space, the following program statements are added into the `cv.h` header file for dynamically loading the binary OpenCV library.

```
#if defined(_CH_)
#pragram package <opencv>
#include <chdl.h>
LOAD_CHDL_CODE(cv,Cv)
#endif
```

The macro `LOAD_CHDL_CODE (cv,Cv)` defined in header file `chdl.h` invokes a function named dlopen( ), which locates and loads the DLL `libcv.dl` into the address space of the running process. The function returns a handle `_ChCV_handle` to the process which the process uses on subsequent calls to the function `dlsym ( )` and `dlclose ( )` which are described later. If an attempt to load the library fails, `dlopen ( )` returns `NULL` and an error message is printed. In the same macro, a function `atexit ( )` is set to close the DLL when the progress terminates.

4.2.2. Function file

When a function such as `cvFun ( )` is called, Ch will search for a function file with the same name of the function and file extension `.chf`, `cvFun.chf` in this example, according to the searching paths set in the Ch language environment. Program 1 shows an example of `chf` file.

```
#include <cv.h>
#include <dlfcn.h>

int cvFun(arg1, arg2) {
    void *fptr;
    int retval;

    fptr = dlsym(_ChCv_handle, "cvFun_chdl");
    if(fptr == NULL) {
        fprintf(_stderr, "Error: %s(): dlsym(): %s\n", __func__,
                dlerror());
        return retval;
    }
    dlrunfun(fptr, &retval, cvFun, arg1, arg2);
    return retval;
}
```

Program 1. The function file `cvFun.chf` file in Ch space.

We assume that the function cvFun ( ) returns a value of integer type.

The function call fptr = dlsym(_ChCv_handle, "cvAvg_chdl") locates the symbol cvFun_chdl within the DLL pointed to by handler _ChCV_handle. The application can then reference the data or call the function defined by the symbol using the function dlrunfun ( ). The function call dlrunfun(fptr, &retval, cvFun, Arg1, Arg2,…) runs the function found in the dynamically loaded object through the address pointed to by fptr, which is returned by function dlsym ( ). The second argument retval is the address of the return variable containing the value. So a Ch function can get the returned value after calling a function in the binary module. If the function doesn't have a return value (i.e. its return type is void), NULL should be used as the second argument. If the third argument is the function name itself, in this case, it is cvFun, Ch will check the number and type of the rest of the argument according to the function prototype. If the third argument is NULL, the argument check is ignored. Starting with the fourth argument, arguments of function cvFun ( ), passed from Ch program ChProgram.ch, will be passed to the chdl function cvFun_chdl ( ) in the DLL.

4.3. Program in binary in C space

File cv_chdl.c consists of chdl functions, which correspond to functions in OpenCV binary library one by one. These chdl functions are the bridges for passing arguments to functions in C space from Ch space. These functions also pass the returned values to Ch space from C space. Program 2 illustrates how these arguments are passed.

The chdl function takes no argument in the argument list if no argument is passed from the chf function. Otherwise, it takes one argument of type void * even if there is more than one argument passed. The argument varg is a pointer to actual argument list.

```
#include <cv.h>
#include <ch.h>

EXPORTCH int cvFun_chdl(void *varg) {
    va_list ap;
    int arg1;
    double arg2;
    int retval;

    Ch_VaStart(ap, varg);
    arg1 = Ch_VaArg(ap, int);
    arg2 = Ch_VaArg(ap, double);
    retval = cvFun(arg1, arg2);
    Ch_VaEnd(ap);
    return retval;
}
```

Program 2. The chdl file cv_chdl.c in C space.

The macro Ch_VaStart(ap, varg) initializes an object having type va_list ap for subsequent use by the macro Ch_VaArg ( ) and function Ch_VaEnd ( ). These macros and functions are defined in the header file ch.h. The Ch_VaArg ( ) macro expands an expression that has the specified type and value of the argument in the call. The first invocation of the Ch_VaArg macro after the Ch_VaStart macro (e.g. arg1 = Ch_VaArg(ap, int)) returns the value of the first argument passed from the chf function. If more than one argument is passed, successive invocations return the values of the remaining arguments in succession. In this example, we assume function cvFun ( ) in DLL accepts two arguments of int and double types.

The expression retval = cvFun(arg1, arg2) calls the function cvFun ( ) in the DLL in C space and saves the return value in the variable retval. The value of retval is obtained from the function in function file cvFun.chf in Ch space.

The macro Ch_VaEnd(ap) releases the memory associated to the object ap of type va_list.

4.4. Building dynamically loaded library

The chdl function is contained in file cv_chdl.c, which shall be used to build DLL libcv.dl. The makefile in Program 3 can be used to build libcv.dl.

The command ch dlcomp libcv.dl cv_chdl.c $(INC) creates the object file cv_chdl.obj from cv_chdl.c with command dlcomp. The argument libcv.dl indicates that the generated object file will be used to build dynamically the loaded library libcv.dl. The option $(INC) provides an additional search path for header files. The command ch dllink libcv.dl cv_chdl.obj $(LFLAG) builds the DLL libcv.dl from the object file cv_chdl.obj with command dllink. At the same time, OpenCV library cv.lib indicated by $(LFLAG) will also be linked.

```
OPENCV = C:/opencv
INC=/I$(OPENCV)/cv/include
LFLAG=$(OPENCV)/lib/cv.lib

target: libcv.dl

libcv.dl: cv_chdl.obj
    ch dllink libcv.dl cv_chdl.obj $(LFLAG)

cv_chdl.obj: cv_chdl.c
    ch dlcomp libcv.dl cv_chdl.c $(INC)

clean:
    del *.obj *.dl *.exp *.lib
```

Program 3. Makefile for building libcv.dl.

```
#include <cv.h>
#include <highgui.h>

int main( int argc, char** argv )
{
   IplImage *orig_image = 0, *des_image = 0;
   int threshold=120, maxValue=255;
   int thresholdType = CV_THRESH_BINARY;

    // open image and show it in "Source" window
    char* filename = argc==2 ? argv[1] : (char*)"baboon.jpg";
    if( (orig_image = cvLoadImage( filename, 1)) == 0 )  {
        printf("Can not load image file");
        return -1;
    }
    cvNamedWindow("Original", 0);
    cvShowImage("Original", orig_image);
    des_image = cvCloneImage( orig_image);

    // perform threshold and show result image
    cvThreshold(orig_image, des_image, threshold, maxValue,
thresholdType);
     cvNamedWindow("Threshold", 0);
    cvShowImage("Threshold", des_image);


    // release variables and exit.
    cvWaitKey(0);
    cvDestroyWindow("Original");
    cvDestroyWindow("Threshold");
    cvReleaseImage(&orig_image);
    cvReleaseImage(&des_image);
    return 0;
}
```

Program 4. Program `threshold.ch`.

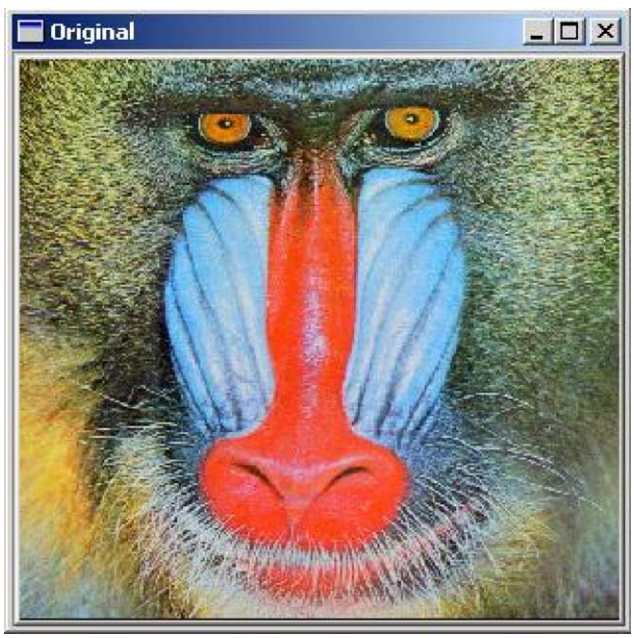

Fig. 2. The original image.

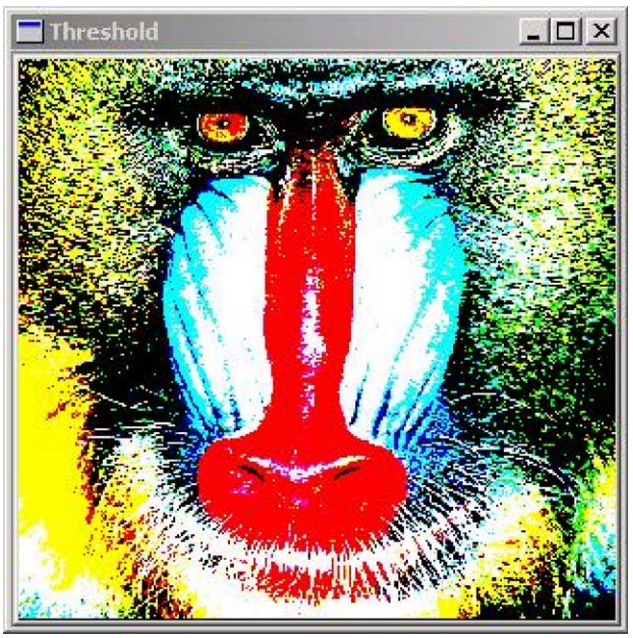

Fig. 3. The image after thresholding.

```
#include <cv.h>
#include <highgui.h>
#include <stdio.h>
#include <numeric.h>

int main(int argc, char** argv) {
    char windowname[] = "Gray image";
    IplImage *image1 = NULL, *image2 = NULL;

    // Load the source image.
    image1 = cvLoadImage(argc == 2 ? argv[1]:"baboon.jpg", 1);
    if(!image1) {
        printf("Image was not loaded.\n");
        return -1;
    }

    //convert to gray image and show in window
    image2 = cvCreateImage(cvSize(image1->width,image1->
             height), IPL_DEPTH_8U, 1);
    cvCvtColor(image1,image2, CV_BGR2GRAY);
    cvNamedWindow(windowname, 0);
    cvShowImage(windowname, image2);

    //perform histogram
    unsigned char x[128], y[image2->imageSize];
    linspace(x, 0, 255);
    memcpy(y, image2->imageData, image2->imageSize);
    histogram(y, x);

    //release variables and exit.
    cvWaitKey(0);
    cvReleaseImage(&image1);
    cvReleaseImage(&image2);
    cvDestroyWindow(windowname);
    return 0;
}
```

Program 5. Program `histogram.ch`.

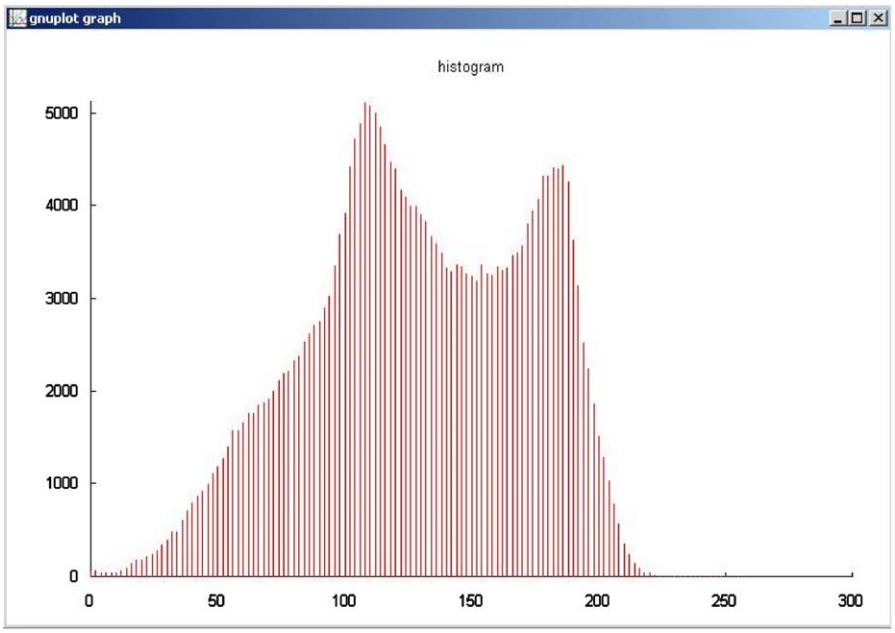

Fig. 4. The histogram of the image.

5. Application examples

Three examples are presented in this section to illustrate the usage and power of Ch OpenCV. The first example illustrates interactive execution of original application program OpenCV in Ch. The second example takes advantage of high-level graphical plotting and numerical features of Ch for image analysis in Ch OpenCV. The last example demonstrates interactive web-based image processing in Ch OpenCV.

5.1. Example 1

This example illustrates how thresholding is applied to each pixel of an image for image processing in Ch OpenCV The function `cvThreshold ( )` in OpenCV library has the following prototype.

```
void cvThreshold (const CvArr* src,
CvArr* dst, double threshold, double max-
Value, int thresholdType);
```

where `src` and `dst` is the source image and destination image, respectively. They must be of single-channel. The parameter `threshold` is the fixed threshold value. This function supports different kinds of thresholding types, which are indicated by parameter `thresholdType`. The macro `CV_THRESH_BINARY`, used in this example, is one of the thresholding types. In this type, if pixel value is large than `threshold`, the pixel value is changed to `maxvalue`. Otherwise, it becomes 0. Program 4 uses function `cvThreshold ( )` to process the image `baboon.jpg` in a JPEG file.

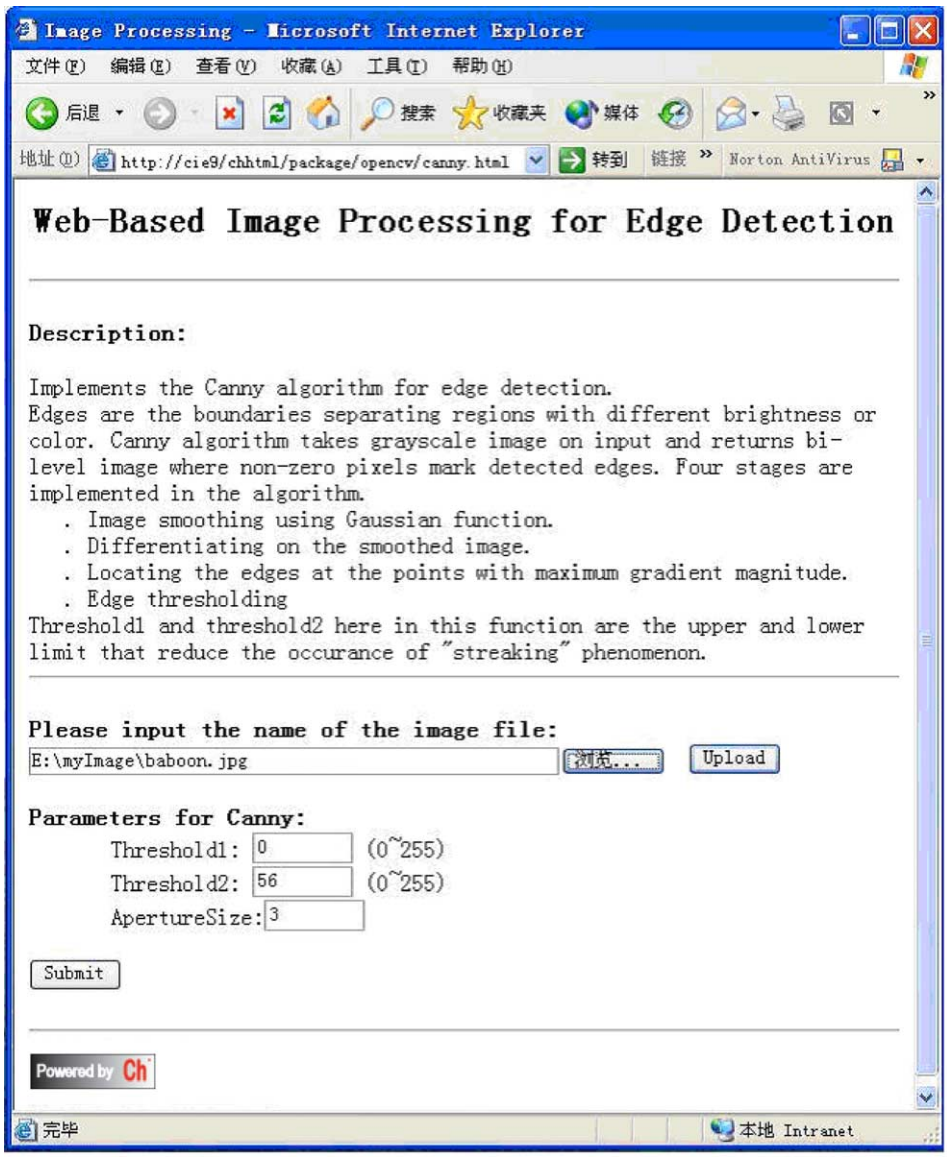

Fig. 5. The web-based image processing for edge detection.

Fig. 2 shows the original image. Fig. 3 is the result image after thresholding is applied. As shown in Program 4, the parameters of `threshold` and `maxValue` are set to 120 and 255, respectively.

5.2. Example 2

This example demonstrates the use of 2D/3D plotting and numerical features in Ch Function `histogram ( )` is used to calculate the histogram of a gray-scale image. The function `histogram ( )` has at least two arguments. The first argument is an array of data set. The second argument is the array, which contains the bins of the histogram. When the function called, the histogram plot is displayed on the screen.

An image with a 24-bit platter may contain millions of colors. In Program 5, the color image shown in Fig. 2 is loaded and converted to a gray-scale image first using function `cvCvtColor ( )`. Then the pixel data in image is read and passed to function `histogram ( )`. When Program 5 is executed, the histogram shown in Fig. 4 is displayed.

5.3. Example 3

This example demonstrates an application of Ch OpenCV for web-based image processing. The user can upload an image file such as `baboon.jpg` through the web page shown in Fig. 5. The uploaded image is processed using the Canny algorithm for edge detection. The parameters for edge detection are provided by the user through the web browser. Details for implementation of the Canny algorithm are described in the web page.

A CGI program upload.ch in Ch uploads image files in the web server. After an image file is uploaded and parameters for the Canny algorithm are set, another CGI program `canny.ch` is invoked for edge detection and the result image shown in Fig. 6 is sent back to the user.

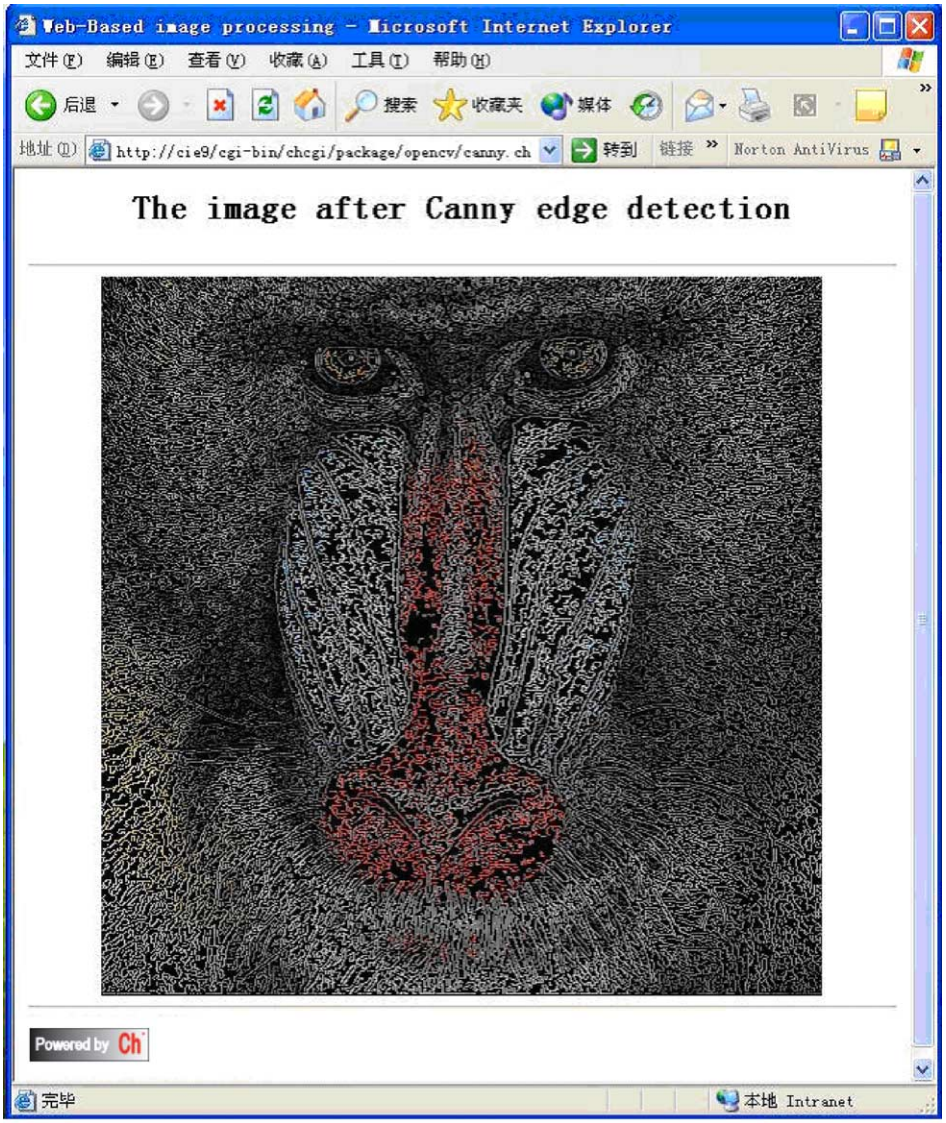

Fig. 6. The result page after Canny edge detection.

```
#!/bin/ch
   #include "imgproc.h"

   int main(void)
   {
     IplImage *image = 0, *cedge = 0, *gray = 0, *edge = 0;
     char * filedir = "C:/Inetpub/wwwroot/cgi-
                      bin/chcgi/package/opencv/temp/";
     char * nfiledir = "http://cie9/cgi-
                      bin/chcgi/package/opencv/temp/";
     string_t sfilename, dfilename;
     string_t sName, dName;
     int paraNum = 4;
     int num;
     chstrarray name, value;

     num =Request.getFormNameValue(name, value); // get CGI value
     if(num<paraNum ) {
        errorHandler("Not enough parameter passed from html file.");
     }

     strcpy(sName, value[0]);
     dName = stradd("res_", value[0]);
     sfilename = stradd(filedir, sName);
     dfilename = stradd(filedir, dName);
     nfilename = stradd(nfiledir, dName);

     if( (image = cvLoadImage( sfilename, 1)) == 0 )  {
        errorHandler("Can not load the original image file\n");
        return -1;
     }

     // Retrieve user defined parameter via CGI
     double threshold1=0, threshold2=56;  // default value
     int aperture=3;                      // default value
     threshold1 = atofCheck(value[headLen]);
     threshold2 = atofCheck(value[headLen+1]);
     aperture = atoiCheck(value[headLen+2]);

     // Canny functions
     cedge = cvCreateImage(cvSize(image->width,image->height),
             IPL_DEPTH_8U, 3);
     gray = cvCreateImage(cvSize(image->width,image->height),
             IPL_DEPTH_8U, 1);
     edge = cvCreateImage(cvSize(image->width,image->height),
             IPL_DEPTH_8U, 1);
     cvCvtColor(image, gray, CV_BGR2GRAY);
     cvCanny(gray, edge, threshold1, threshold2, aperture);
     cvZero( cedge );
     cvCopy( image, cedge, edge );
     cvSaveImage(dfilename, cedge);
     //release variable and exit
     cvReleaseImage(&image);
     cvReleaseImage(&gray);
     cvReleaseImage(&edge);

     //return result
     Response.setContentType(contenttype);
     Response.begin();
     Response.title(title);
     fprintf stdout << ENDPRINT
         <H2><CENTER>The image after Canny edge detection </H2>
          </CENTER> <HR>
         <center><img src="$(nfilename)"></center>
     ENDPRINT
```

Program 6. The CGI program `canny.ch`.

```
fprintf stdout << ENDPRINT
     <HR>
     <A HREF = "http://www.softintegration.com/">
     <img src= "http://cie9/cgi-
     bin/chcgi/package/opencv/poweredbych.gif"
     border=0 alt="Powered by Ch"> </a>
ENDPRINT
Response.end();

return 0;
```

Program 6 (*continued*)

The source code of CGI script `canny.ch` in the web server is listed in Program 6.

6. Conclusions

In this paper, design and implementation of open source Ch OpenCV have been described. Ch OpenCV provides a powerful environment for interactive open architecture computer vision. It allows the same programs to be executed interpretively for script computing or compiled using a C compiler for fast execution. It is especially useful for rapid prototyping, teaching, student learning of computer vision, and web-based image processing. Sample applications of Ch OpenCV for rapid prototyping and web-based image processing have been presented in this paper. Ch OpenCV contains salient features from both Ch and OpenCV. For example, Ch OpenCV is embeddable in other application programs. It is freely available and has potential for many applications in computer vision.

References

[1] TargetJr. http://www.esat.kuleuven.ac.be/~targetjr/
[2] VXL. http://vxl.sourceforge.net/
[3] Gandalf. http://gandalf-library.sourceforge.net/
[4] HALCON. http://www.mvtec.com/
[5] Delft Scientific Image Processing Library. http://www.ph.tn.tudelft.nl/DIPlib/
[6] National Instrument Company. http://www.ni.com/
[7] Microsoft Research Center. http://research.microsoft.com/projects/VisSDK/
[8] MATLAB Image Processing Toolkit 3.2. http://www.mathworks.com/products/image/
[9] Mathematica Digital Image Processing. http://www.wolfram.com/products/applications/digitalimage/
[10] Aurora Company. http://www.hallogram.com/leadtool/imgsrv/
[11] Ch OpenCV. http://www.softintegration.com/products/thirdparty/opencv/
[12] Bradski GR. The OpenCV library. Dr Dobb's J 2000;November: 120–5.
[13] OpenCV. http://www.intel.com/research/mrl/research/opencv/
[14] Bradski GR, Davis JW. Motion segmentation and pose recognition with motion history gradients. Mach Vis Appl 2002;13:174–84.
[15] Yang R, Gotz D, Hensley J, Towles H, Brown MS. PixelFlex: a reconfigurable multi-projector display system. IEEE Proceeding of the Conference on Visualization, San Diego, October 21–26; 2001. p. 167–74.
[16] Cheng HH. The Ch language environment user's guide. SoftIntegration, Inc.; 2002. . Available at http://www.softintegration.com.
[17] Cheng HH. Scientific computing in the Ch programming language. Sci Programming 1993;2(3):49–75.
[18] SoftIntegration, Inc.. The Ch language environment SDK user's guide. 2002. Also see . Available at http://www.softintegration.com/products/sdk/chsdk/.
[19] SoftIntegration, Inc., The Ch language environment cgi toolkit user's guide. Also see http://www.softintegration.com/products/toolkit/cgi/
[20] SoftIntegration, Inc., Embedded Ch user's guide. Also see http://www.softintegration.com/solution/embedded