S I V P

Scilab Image and Video Processing Toolbox*
version 0.4.0

Shiqi YU and **Shulin SHANG**

Free Software Association†,
Institute of Automation, Chinese Academy of Scienses

shiqi.yu@gmail.com, slshang@hitic.ia.ac.cn

September 5, 2006

*http://sivp.sourceforge.net/
†http://fsa.ia.ac.cn/

Contents

1 Introduction **3**
1.1 SIVP overview 4
1.2 Features of SIVP 4
1.3 Major improvements 5

2 Installation **5**
2.1 Installation on Linux 5
2.2 Installation on Windows 6

3 How to use SIVP **8**
3.1 Read an image file and show it 8
3.2 Write to an image file 9
3.3 Read frame from a video file 9
3.4 Create a video file 10
3.5 Edge detection 10
3.6 Histogram 11
3.7 More functions 12

4 Comparison with similar software **12**

5 Port SIVP to Windows **12**

6 Future work **18**

7 Acknowledgements **18**

1 Introduction

SIVP (Figure 1) is a toolbox designed for academic researchers. It is meant to be a useful, efficient, and free image and video processing toolbox for Scilab. Currently it has been downloaded and used by many researchers.

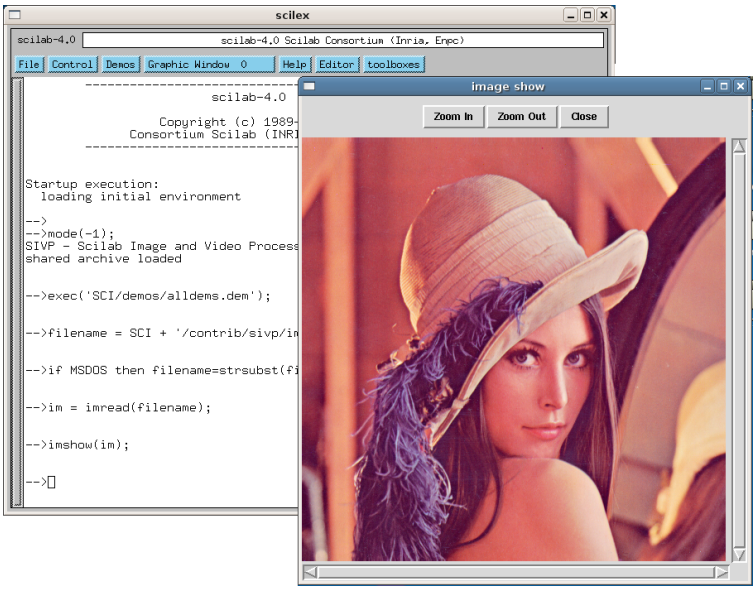


Figure 1: SIVP interface

Scilab is an excellent scientific software package for numerical computations. It provides a powerful open computing environment for engineering and scientific applications. Compared with Matlab, Scilab still does not have enough of high quality toolboxes. In image processing, there are many toolboxes, and SIP[1] toolbox is the best. SIP was developed based on ImageMagicx[2], but ImageMagicx is not good enough for image processing in research field. So we developed another toolbox based on OpenCV[3]. OpenCV is a good library for image processing which has high efficiency. Because video is just an image sequence, so we involved video processing in the toolbox.

SIVP is not only developed for Scilab Contest. The main purpose is developing a useful, efficient, and free image and video processing toolbox for academic researchers. SIVP is a free software and licensed under GPL (GNU General Public License). Everyone can get the source code from SIVP homepage (Figure 2), modify it and improve it.

[1]http://siptoolbox.sourceforge.net/
[2]http://www.imagemagick.org/
[3]http://www.intel.com/research/mrl/research/opencv/

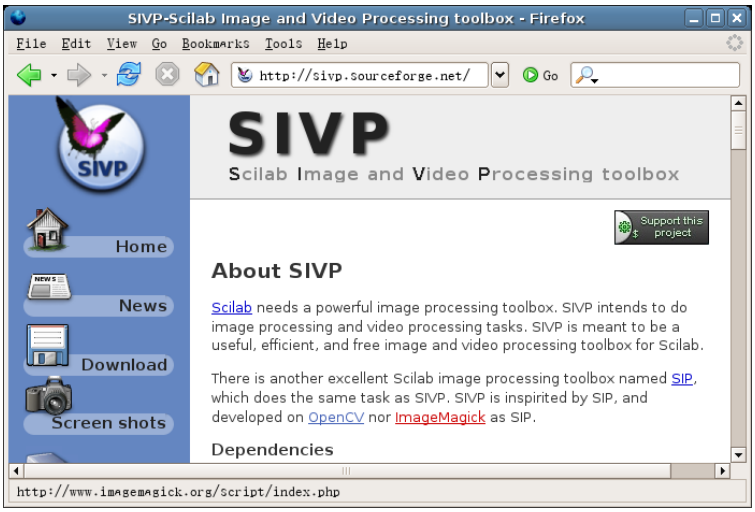


Figure 2: SIVP homepage

1.1 SIVP overview

The information for SIVP is listed in Table 1.

License	GNU General Public License (GPL)
Operating System	Linux, Windows, and other *nix system
Programming Language	C, TCL/TK
Interface Language	English
Dependencies	Scilab[a], OpenCV[b]
Homepage	http://sivp.sourceforge.net/
Project Page	http://sourceforge.net/projects/sivp/
Authors	Shiqi Yu, Shulin Shang
Downloads	2509[c]

[a]Scilab should be >= 3.1.1
[b]For video support on Linux, OpenCV should be compiled with FFMPEG.
[c]Up to Sep. 2, 2006, and only downloads on SourceForge.net

Table 1: SIVP overview

1.2 Features of SIVP

A lot of new features are added to SIVP version 0.4.0:

- Image I/O (Supported format: BMP, PNG, JPEG, TIFF, PBM, PGM, PPM, SR);
- Video I/O, camera read;

- Image type conversion;
- Spatial transformation functions;
- Image analysis and statistics functions;
- Image arithmetic functions;
- Linear Filtering;
- Morphological operations;
- Color space conversions.

1.3 Major improvements

Compared with SIVP version 0.2.6 submitted to Scilab Contest 2005, SIVP version 0.4.0 is improved in the following aspects.

- Call methods of all functions are modified to be the same with Matlab Image Processing Toolbox.
- Improved support for various data types. Now BINARY, UINT8, INT8, UINT16, INT16, INT32 and DOUBLE are well supported by SIVP.
- More functions added. There are 46 functions in SIVP 0.4.0 compared with 25 ones in SIVP 0.2.6.
- Fixed some bugs and SIVP becomes more robust.

2 Installation

SIVP can be easily installed whatever on Linux or Windows. Before your installation, you need to download relative software package from SIVP homepage http://sivp.sourceforge.net/, and then install it as described in the following subsections.

2.1 Installation on Linux

You can only need run `./configure, make,` and `make install` three commands to compile and install SIVP on Linux.

1. Uncompress the downloaded tar.gz file and enter SIVP source code directory.

   ```
   tar zxvf sivp-0.4.0.tar.gz
   cd sivp-0.4.0
   ```

2. Guess values for system-dependent variables and create Makefiles

   ```
   ./configure
   ```

   If `./configure` reports some package missed, please install them and rerun `./configure`.

3. Compile the source code.

   ```
   make
   ```

4. Install the toolbox.

```
make install
```

`make install` should be run as root.

2.2 Installation on Windows

We provide a friendly graphic user interface for SIVP Windows version installation. What you do are just to run the installer program and click mouse following the prompts. The installation is as Figure 3-7.

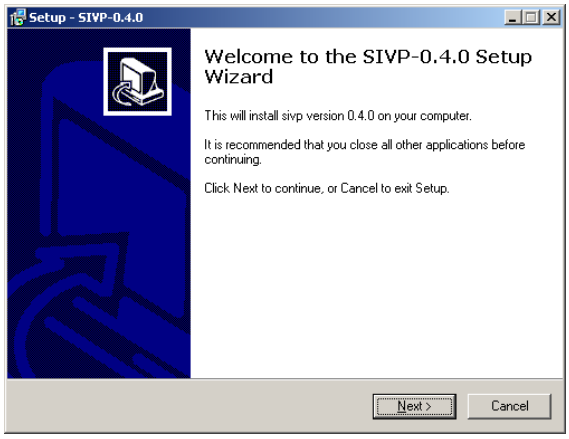


Figure 3: The 1^{st} step

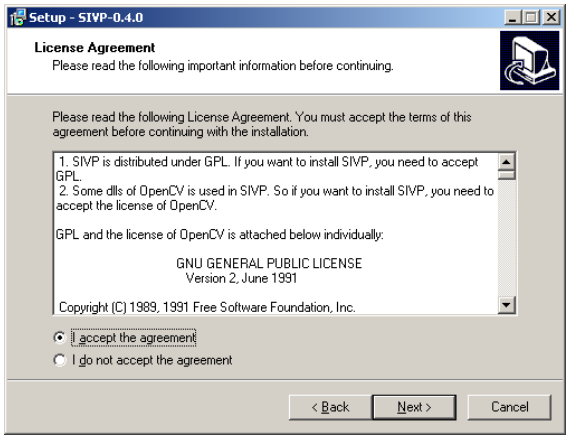


Figure 4: The 2^{nd} step

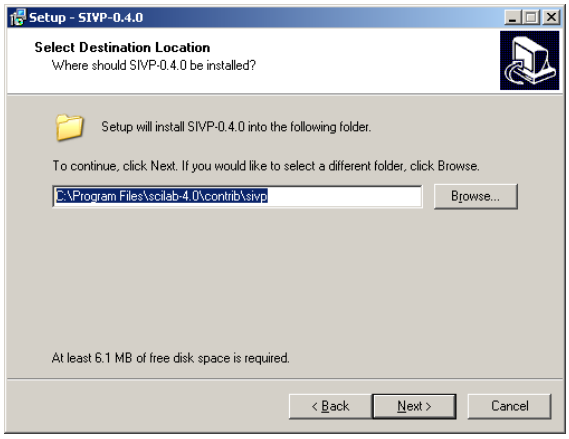


Figure 5: The 3^{rd} step

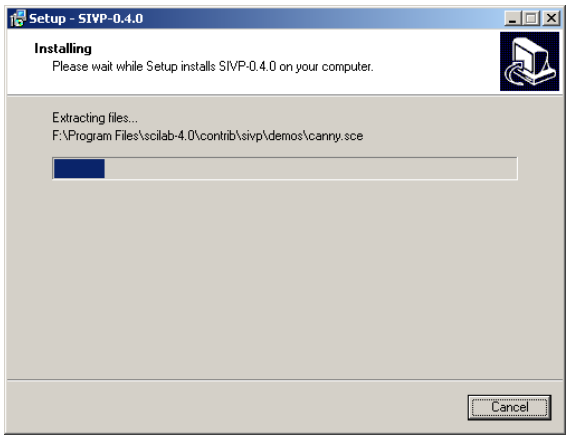


Figure 6: The 4^{th} step

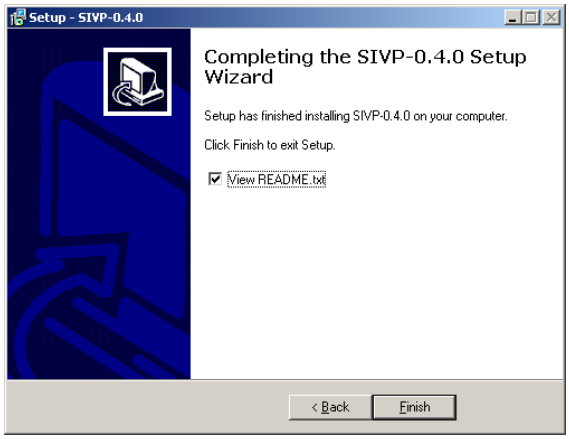


Figure 7: The 5^{th} step

3 How to use SIVP

After you install SIVP, an item "SIVP" will appear in "toolboxes" menu. SIVP can be loaded into Scilab by clicking "SIVP" menu item (Figure 8).

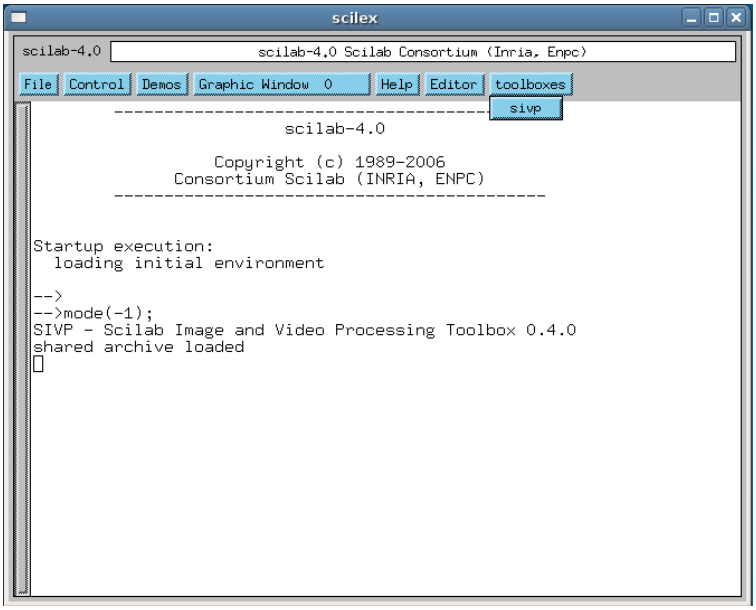


Figure 8: Load SIVP into Scilab

3.1 Read an image file and show it

The following image formats are supported by SIVP, and you can use `imread` function to read image files.

- Windows bitmaps - BMP, DIB;
- JPEG files - JPEG, JPG, JPE;
- Portable Network Graphics - PNG;
- Portable image format - PBM, PGM, PPM;
- Sun rasters - SR, RAS;
- TIFF files - TIFF, TIF.

```
im=imread(SCI+'/contrib/sivp/images/lena.png');
imshow(im); //show the image
```

The result is shown in Figure 9.

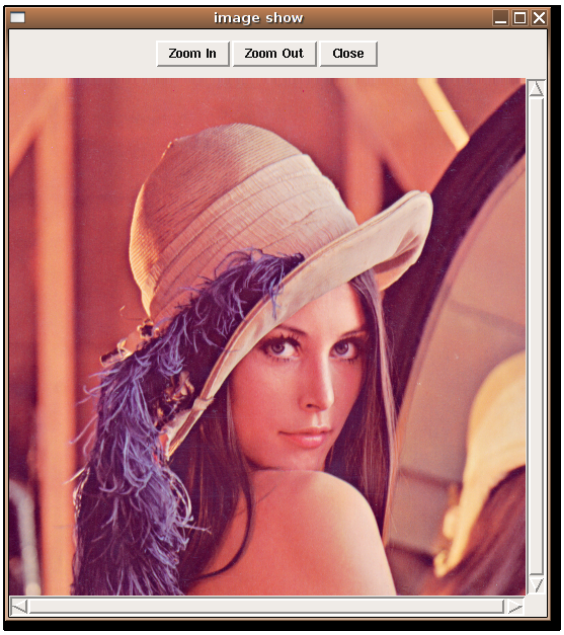


Figure 9: Showing an image

3.2 Write to an image file

`imwrite` function can write an image matrix to an image file. The following code write `im` to image file "`image.bmp`";

```
imwrite{im, 'image.bmp'};
```

The image file format is decided by the extension of filename.

3.3 Read frame from a video file

If you want to read the frame in a video file, you need to open the video file using function `aviopen` first, and then use `avireadframe` to get a frame. At last you should close the video file using `aviclose`. The following code open a video file and read the first 100 frames.

```
n = aviopen('video.avi'); //open the video file
for idx=1:100,
    im=avireadframe(n);   //get a frame
    imshow(im);           //show the frame
end;
aviclose(n);              //close the file
```

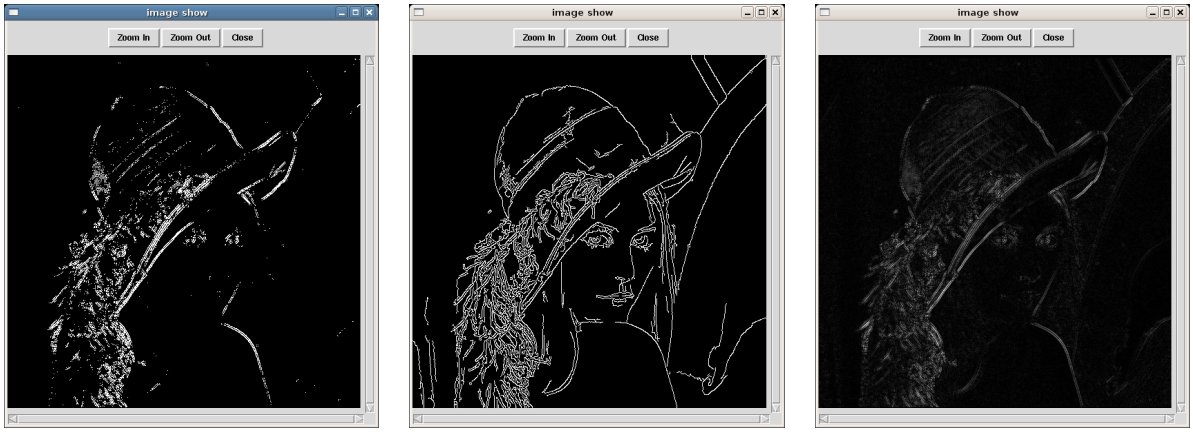
3.4 Create a video file

If you want to write a sequence of images to a video file, you need to create the file using function `avifile`, and then add frame using function `addframe` one by one. At last you should close the file using `aviclose`. an example is as follows.

```
im = imread('lena.png');
n = avifile('lena.avi', [200;200], 30);
for ii=1:300
    ims = im(200:399, ii:ii+199, :);
    addframe(n, ims);
end
aviclose(n);
```

3.5 Edge detection

Function `edge` can detect edges in an images using various operators. Currently `sobel`, `prewitt`, `log`, `fftderiv` and `canny` are supported by this function. More operators will be added in future.



(a) Sobel operator (b) Canny operator (c) Sobel operator (no threshold)

Figure 10: Edge detection

The following source code will use `sobel`, `canny` and `sobel` but no threshold to detect edges in image `im`.

```
im = imread('lena.png');
im = rgb2gray(im);

E = edge(im, 'sobel');
```

```
imshow(E);

E = edge(im, 'canny', [0.06, 0.2]);
imshow(E);

E = edge(im, 'sobel', -1);
imshow(mat2gray(E));
```

The results are shown in Figure 10.

3.6 Histogram

Function `imhist` can be used to get the histogram of an image. If display arguments are given, the function can show the histogram in a figure (Figure 11).

```
im = imread('lena.png');
im = rgb2gray(im);
[count, cells]=imhist(im);
[count, cells]=imhist(im, 10);
scf(0); imhist(im, 10, '');
scf(1); imhist(im, 10, 0.5);
scf(2); imhist(im, 10, 'green');
```

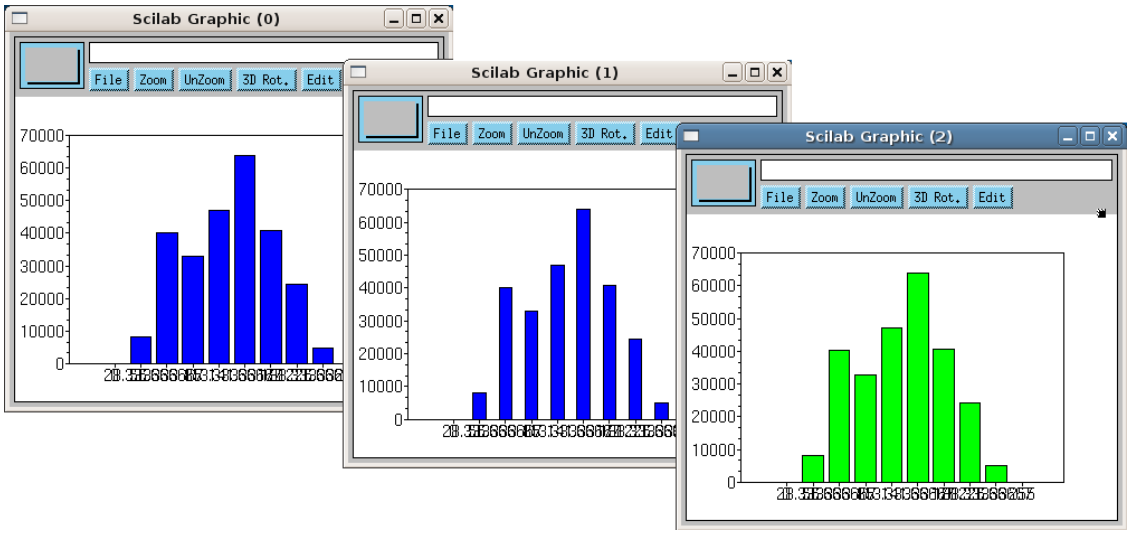


Figure 11: Histograms

3.7 More functions

If you want to know more functions of SIVP, please refer SIVP online document. By menu $Help \rightarrow Help\ browser \rightarrow SIVP \cdots$, all manuals for SIVP functions are shown in a window (Figure 12).

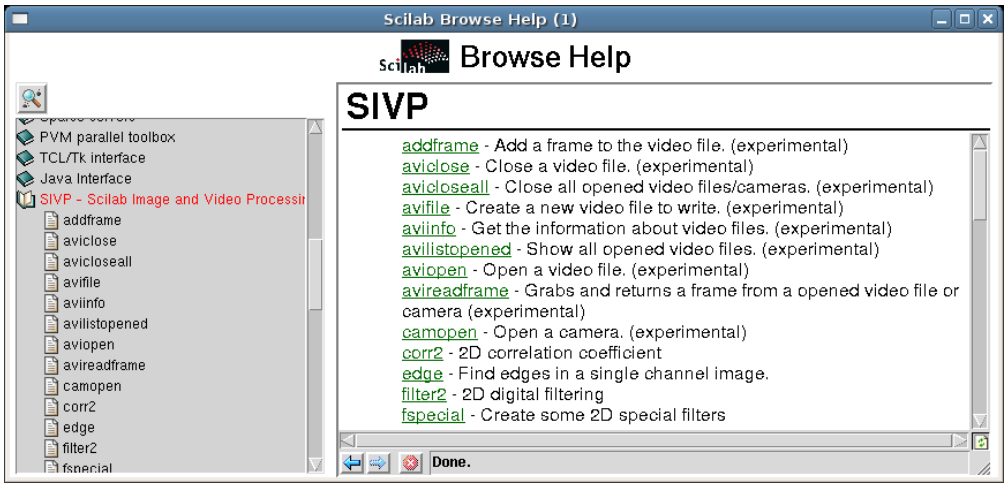


Figure 12: Online help

4 Comparison with similar software

The software in comparison[4]:

- SIP, Scilab Image Processing toolbox which has been widely used.
- Matlab Image Processing toolbox.
- SIVP.

Table 2 lists some comparison results of SIP, Matlab and SIVP. Compared with other software, SIVP is more superior than the other two in most aspects.

5 Port SIVP to Windows

SIVP was developed on Linux system. SIVP must be ported to Windows for using SIVP on Windows. Normal users can just download binary Windows version from SIVP home page, and no need to port it to Windows.

[4]All software ran on the same PC(P4 3.0G CPU, 512M RAM);SIP and SIVP ran on Linux and Matlab ran on Windows.

	SIP	**Matlab**	**SIVP**
The number of supported image format	>90	15	10
The number of supported video format	unsupported	1 for Linux; 5 for Windows	about 50 (depends on the number of installed codecs)
Open source	Yes	No	Yes
Reading 240 352×240 PNG color images (Figure 13)	6.78s	6.04s	4.94s
Reading 240 352×240 PNG color images (Figure 14)	1.85s	2.04s	0.24s
Sobel edge detection(512×512 gray iamge, Figure 15)	0.34s	0.32s	0.04s
Reading video frames(320×240 rawvideo, 100 frames, Figure 16)	-	0.45s	0.31s
Showing color image (512×512, Figure 17)	0.19s	0.21s	0.95s

Table 2: Comparison with similar software

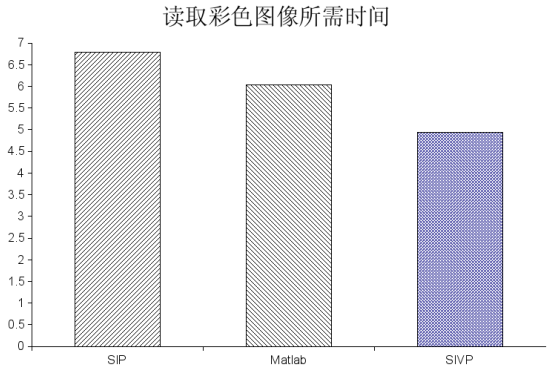


Figure 13: The time for reading color images

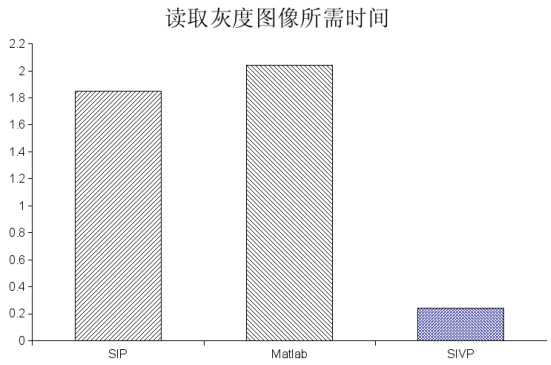


Figure 14: The time for reading gray images

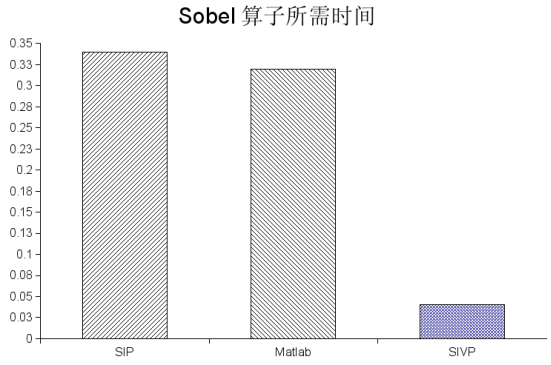


Figure 15: The time for sobel edge detection

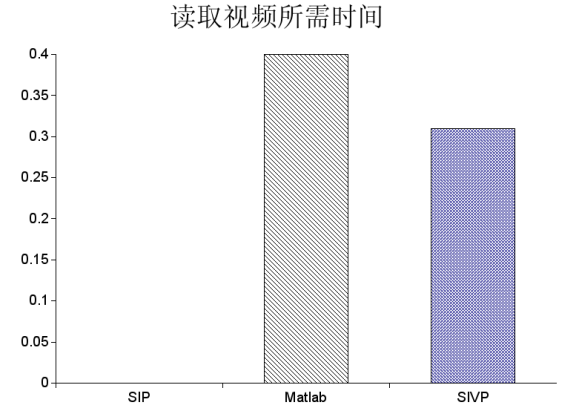


Figure 16: The time for reading video frames

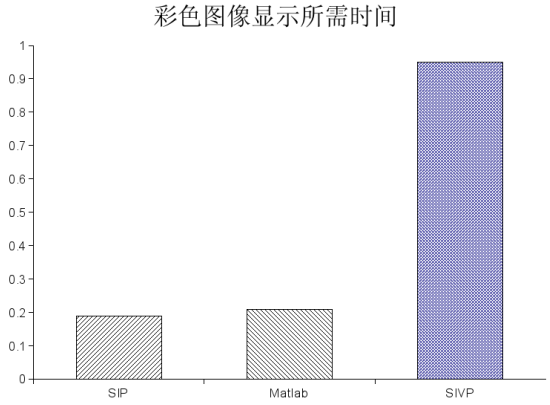


Figure 17: The time for showing a color image

MinGW (Minimallist GNU for Windows) is a set of development tools for generating native Windows code. It enables you to compile this toolbox using configure and make, similar to Unix/Linux.

There are some specific things that you must set up by hand in order to compile this toolbox with MinGW. The whole process consists in the following steps

1. Vist www.mingw.org, download and install the following packages: MinGW, MSYS, and MSYS-DTK. Then update autoconf to 2.59(msys-autoconf-2.59) and automake to 1.8.2(msys-automake-1.8.2). Install libxslt-1.1.14+.win32.
2. Install gtk-win32-devel-2.6.x in `C:\GTK` (download it from http://gladewin32.sourceforge.net/), execute commands below under MSYS:

```
cp /c/GTK/bin/* /usr/local/bin/
cp /c/GTK/share/aclocal/pkg.m4  /usr/local/share/aclocal/
```

3. Your scilab installation must be in `c:\scilab` (/c/scilab in MSYS), because this path is hardcoded in the build process. Please move your scilab installation to that place or else put an entry in the /etc/fstab file inside MSYS.
4. Inside /c/scilab/bin (MSYS path), we must rename the libraries "LibScilab.*" to lowercase. Even though Windows is case insensitive, this causes problem to programs inside MSYS. This step can be quickly done inside the MSYS shell typing this:

```
mkdir /tmp/scilibs
mv /c/scilab/bin/LibSci* /tmp/scilibs
cd /tmp/scilibs
for i in `ls Lib*`; do
   mv $i /c/scilab/bin/`echo $i | tr A-Z a-z`;
done
```

5. Install your opencv in `c:\opencv` (/c/opencv in MSYS). Make a directory "opencv" under current directory, create opencv.sh and opencv.pc files in above said opencv directory according to the text below. Revise the 097 to actual installed opencv version number. Put an entry in /etc/fstab inside MSYS and revise the opencv directiry name of the opencv/opencv.sh script if opencv is installed it elsewhere At last, run opencv.sh in MSYS

```
#-----------------------------------------------
#opencv/opencv.pc
#-----------------------------------------------

# Package Information for pkg-config
prefix=/usr/
exec_prefix=${prefix}/bin
libdir=${prefix}/lib/opencv
includedir=${prefix}/include/opencv

Name: OpenCV
Description: Intel(R) Open Source Computer Vision Library
Version: 0.9.7
Libs: -L${libdir} -lcxcore -lcv -lhighgui -lcvaux
Cflags: -I${includedir}
```

```
#-----------------------------------------------
#opencv/opencv.sh
#-----------------------------------------------

#!/bin/sh
rm *.def *.dll.a
echo EXPORTS > cxcore.def
nm /c/opencv/lib/cxcore.lib | grep ' T _' | sed 's/.* T _//' >> cxcore.def
dlltool --def cxcore.def --dllname cxcore097.dll --output-lib libcxcore.dll.a

echo EXPORTS > cv.def
nm /c/opencv/lib/cv.lib | grep ' T _' | sed 's/.* T _//' >> cv.def
dlltool --def cv.def --dllname cv097.dll --output-lib libcv.dll.a

echo EXPORTS > cvaux.def
nm /c/opencv/lib/cvaux.lib | grep ' T _' | sed 's/.* T _//' >> cvaux.def
dlltool --def cvaux.def --dllname cvaux097.dll --output-lib libcvaux.dll.a

echo EXPORTS > highgui.def
```

```
nm /c/opencv/lib/highgui.lib | grep ' T _' | sed 's/.* T _//' >> highgui.def
dlltool --def highgui.def --dllname highgui097.dll --output-lib libhighgui.dll

echo EXPORTS > cvcam.def
nm /c/opencv/lib/cvcam.lib | grep ' T _' | sed 's/.* T _//' >> cvcam.def
dlltool --def cvcam.def --dllname cvcam097.dll --output-lib libcvcam.dll.a

if test ! -d "/usr/lib/opencv"; then
   mkdir /usr/lib/opencv
fi
cp lib*.a /usr/lib/opencv
cp opencv.pc /usr/lib/opencv

cp /c/opencv/bin/cxcore097.dll /usr/bin
cp /c/opencv/bin/cv097.dll /usr/bin
cp /c/opencv/bin/cvaux097.dll /usr/bin
cp /c/opencv/bin/highgui097.dll /usr/bin
cp /c/opencv/bin/cvcam097.dll /usr/bin

if test ! -d "/usr/include/opencv"; then
   mkdir /usr/include/opencv
fi
cp /c/opencv/cxcore/include/*.h /usr/include/opencv
cp /c/opencv/cv/include/*.h /usr/include/opencv
cp /c/opencv/cvaux/include/*.h /usr/include/opencv
cp /c/opencv/otherlibs/highgui/*.h /usr/include/opencv
cp /c/opencv/otherlibs/cvcam/include/*.h /usr/include/opencv
```

6. Enter the toolbox's toplevel directory, export environment variables SCI and PKG_CONFIG_PA for example,

```
export SCI=/c/scilab
export PKG_CONFIG_PATH=/usr/lib/opencv
```

where "/usr/lib/opencv" is the file opencv.pc located. Then type:

```
./configure
make
make install
```

7. For some versions of libtool/automake you may have to install the DLL of the toolbox by hand:

```
cp src/.libs/libsivp*.dll /c/scilab/contrib/sivp/lib/libsivp.dll
```

8. Copy the cv097.dll, cxcore097.dll, highgui097.dll to SCI/contrib/sivp/lib by hand. For example,

```
cp /c/opencv/bin/cxcore097.dll /c/scilab/contrib/sivp/lib
cp /c/opencv/bin/cv097.dll /c/scilab/contrib/sivp/lib
cp /c/opencv/bin/highgui097.dll /c/scilab/contrib/sivp/lib
```

Now enter Scilab and click the menuitem sivp button in toolbox.

6 Future work

SIVP will be developed in future and work on the following aspects to make SIVP a more useful and robust toolbox.

- Optimizing image showing.
- Add more functions.
- Optimizing Windows version.
- Recruiting more developers for SIVP.
- Merging with SIP if possible. All developers of SIVP and SIP work together to develop a better toolbox.

7 Acknowledgements

We would like to thank SourceForge.net providing us a platform for managing the project. We also thank Richardo Fabbri answering out questions, LIAMA for the support, and other known and unknown friends who have helped us. Besides, thanks to Scilab group adding a link to SIVP home page on Scilab home page to let more people know SIVP.