# Programming with Intel IPP (Integrated Performance Primitives) and Intel OpenCV (Open Computer Vision) Under GNU Linux

## A Beginner's Tutorial

by Jérôme Landré
j.Landre@iutlecreusot.u-bourgogne.fr

version 0.4 - July 2003

*Université de Bourgogne*
**Institut Universitaire de Technologie**
*Laboratoire Électronique, Informatique et Images*
U.M.R. C.N.R.S. 5158
12, rue de la Fonderie
71 200 Le Creusot
FRANCE
tél. +33 (0)3–85–73–10–00
fax +33 (0)3–85–73–10–99
http://iutlecreusot.u-bourgogne.fr




To Life, Love, Dreams and Peace...

# Contents

# Chapter 1

# Introduction

*"Every man has two nations and one of them is France."*
**Benjamin FRANKLIN — 1706-1790**

## 1.1 Foreword

Writing a book, especially a tutorial is not an easy task. Teaching is a very interesting job but also a very difficult one because you need to give interesting information on a subject to people that are not used with this subject. You need to be clear and precise to make your assistance understand what you want to explain. I want to be as precise as possible but without any warranty of success. Your judgment will be the best way for me to see if I was good or not.

Because english (and US english) is not my mother language, I want to apologize for any mistake encountered in the pages of this manual. I have tried to do my best to produce a good written english, however any comment is welcome to improve its quality.

## 1.2 About this tutorial

This manual is dedicated to the use of Intel signal and image processing libraries: IPP (Integrated Performance Primitives) and OpenCV (Open Computer Vision). The aim of this book is not to enter deeply into details but to introduce the concepts used in these libraries.

IPP is a commercial low-level library for signal processing, image processing and matrix computation developed by Intel Corporation. It offers functions that are optimized to run on Intel Processors (Pentium, Pentium II, Pentium III, Pentium 4 and Itanium) and to take in account special abilities such as MMX (MultiMedia eXtensions), SSE (Streaming Single instruction multiple data Extensions), and SSE2 cabled instructions. So you need to have an Intel processor on our computer in order

to use these libraries. It is divided into three libraries: signal processing, image processing and small matrix computing.

OpenCV is a free library developed by Intel Corporation. It proposes high-level computer vision and image processing functions. As it is a free and open library, it is not reserved to Intel processors and it can be built over a lot of architectures.

Mixing IPP and OpenCV together gives users a very powerful set of functions to work in the computer vision, signal and image processing domain. This book describes the way to mix together these libraries to obtain a very useful image processing development platform.

## 1.3 Intel IPP

This tutorial is not intended to replace Intel official manual [2] which is the absolute reference on IPP. IPP stands for Intel Integrated Performance Primitives, it is a signal processing, image processing and matrix calculation library developed by Intel Corporation.

IPP is not a free library, it comes under an Intel licensing policy which is explained at Intel website [3]. IPP offers to programmers a wide range of low-level functions which are optimized when used on an Intel processor (from Pentium to Itanium). It is really a good library for signal, image, video and sound processing with very good performances due to optimized instructions.

## 1.4 Intel OpenCV

OpenCV (Open source Computer Vision library) is an open source library developed by Intel Corporation. The official manual [1] is freely available on Internet OpenCV website [4]. This library allows high level functions for computer vision and image processing.

Image processing, computer vision algorithms are proposed to programmers in order to create powerful applications in the domain of digital vision. OpenCV offers many high-level datatypes such as sets, trees, graphs, matrices... OpenCV is opensource to run on many computer platforms.

## 1.5 Prerequisites

To understand this tutorial, a few knowledge about GNU Linux is recommended. You will need to be root (administrator) of your Linux system during the installation process. GNU Linux has known many changes and the visual interface (KDE, GNOME...) looks more and more beautiful. But I am an "old school Linux" student who does a lot of operations by hand, not because I am mad (even if I am a little) but because I do not know how to do with visual tools included in recent Linux distributions.

The knowledge of several Unix commands like cd (change directory), cp (copy), ls (list directory content) should help you to understand the different steps to build an application using Intel libraries. You will need to use a Linux shell to compile your programs and to launch them. I have used "bash" which is the default shell under several Linux distributions.

All the examples presented in this tutorial introduction have been written in C language and compiled with "gcc". A general knowledge about C programming is important to understand the programming methods used with IPP and OpenCV. Readers must have a good understanding of pointers in the C programming language because IPP and OpenCV use them often (as all the other applications written in C).

## 1.6 Presentation of chapters

Each chapter of this tutorial gives one or more example(s) of C code. I used "gcc" C compiler to build each example of this manual but I think these examples should work under another C compiler.

Chapter 2 will introduce the installation of IPP and OpenCV under GNU Linux. It will explain how to install IPP binaries and how to compile OpenCV sources. In chapter 3, OpenCV basic functions will be described with several examples to test this library. IPP functions will be presented in chapter 4. Chapter 5 will give a conclusion on IPP and OpenCV.

# Chapter 2

# Installation under GNU Linux

*"Tout ce que je sais, c'est que je ne sais rien."*
*"All I know is I know nothing."*
**Blaise PASCAL — 1623-1662**

## 2.1 IPP installation

IPP is not free, you must register and pay to download it, the official download site is given in the appendix A of this manual. The library is given under the form of a ".tar" archive file and you need a license file ".lic" to enable the library.

Installing IPP library under GNU Linux is not very difficult, it is given under the form of pre-compiled binaries, so you just need to install them and it works. To install IPP library, you must be root. I use a RedHat 9.0 distribution but I think the code I give could work on other Linux based distributions without any problem (but I have no warranty about this !). I have installed IPP 3.0 version of this file from a cd-rom. You can also install a downloaded version from Internet.

Before installing, we must precise several choices I did. First of all, the libraries will be installed under /home/intel directory. Why ? When I install Linux, I always create two partitions: "/" and "/home". By this way, I separate the system ("/" partition) and the users data ("/home" partition). So when I need to install a new version of Linux, I just format the "/" partition keeping the "/home" partition unchanged. And then I do not need to reinstall OpenCV and IPP at each installation of a new Linux version.

Another thing we have to choose is the working directory, it will be located in "/home/jeje/ippopencv". This is my working directory but of course, you can use another path for yours. "ippopencv" contains the sources of the C programs, a Makefile (detailed further in this

manual) and another directory "images" with all the images used in the examples.

At last, the installation starting directory is "/tmp". It is a common directory in which users can read and write easily and I used it to start the installation. So you must copy (using "cp") your OpenCV and IPP sources archives in it with the license file for IPP given by Intel Corporation when you register IPP.

In order to be clear, before each command launched an empty line will be added. A last thing, you need to have "lesstif" installed to build OpenCV. The installation of "lesstif" is not detailed in this manual, but "lesstif" is present on many Linux distributions under the form of three "rpm" packages: "lesstif-client", "lesstif-devel" and "lesstif-mwm". You will probably also need "fltk" and "ffmpeg" libraries to compile the examples for video processing. Please refer to their respective web sites for explanations about their installation.

---

```
[jeje@rebec jeje]$ pwd
/home/jeje

[jeje@rebec jeje]$ su -
Password:

[root@rebec root]# cd /tmp

[root@rebec tmp]# ll
total 113268
-rw-rw-rw-    1 jeje     jeje     106199040 jan  1  1970 ipp30lin.tar
-rw-rw-rw-    1 jeje     jeje          304 jan  1  1970 l_ipp_63608027.lic
-rw-rw-rw-    1 jeje     jeje      9653883 jan  1  1970 OpenCV-0.9.5.tar.gz

[root@rebec tmp]# tar xvf ipp30lin.tar
installdata
install.sh
ipp_license

[root@rebec tmp]# ll
total 217084
-rwxrwxrwx    1 root     root     106168907 jan 16  2003 installdata*
-rwxrwxrwx    1 root     root         5678 jan 16  2003 install.sh*
-rw-rw-rw-    1 jeje     jeje     106199040 jan  1  1970 ipp30lin.tar
-rw-r--r--    1 root     root        12512 jan 16  2003 ipp_license
-rw-rw-rw-    1 jeje     jeje          304 jan  1  1970 l_ipp_63608027.lic
-rw-rw-rw-    1 jeje     jeje      9653883 jan  1  1970 OpenCV-0.9.5.tar.gz

[root@rebec tmp]# ./install.sh
RPM shows no installed Intel packages.

Which of the following would you like to install?
1.  Intel(R) Integrated Performance Primitives Version 3.0
x.  Exit

--- type '1' then press 'Enter'

Intel(R) Integrated Performance Primitives Version 3.0

-------------------------------------------------------------------------------
Please carefully read the following license agreement.  Prior to installing the
software you will be asked to agree to the terms and conditions of the following
license agreement.
-------------------------------------------------------------------------------
Press Enter to continue.

--- Press 'Enter'
```

```
IMPORTANT - READ BEFORE COPYING, INSTALLING OR USING.
Do not copy, install, or use the Materials provided under this license agreement
 ("Agreement"), until you have carefully read the following terms and conditions.

By copying, installing, or otherwise using the Materials, you agree to be
bound by the terms of this Agreement.  If you do not agree to the terms
of this Agreement, do not copy, install, or use the Materials.

End User License Agreement for the Intel(R) Integrated Performance Primitives
3.0 for Linux*

1. LICENSE DEFINITIONS:

[...continued...]

11. APPLICABLE LAWS: [...] You may not export the Materials in violation
 of applicable export laws.

Enter 'accept' to continue, 'reject' to exit:

--- type "accept" then press 'Enter'

Where do you want to install to? Specify directory starting with '/'. [/opt/intel]

--- type "/home/intel" then press 'Enter'

/home/intel
What rpm install options would you like?  [-Uvh --replacefiles]

--- Press 'Enter'

-------------------------------------------------------------------------------
Intel(R) Integrated Performance Primitives Version 3.0
Installing...
Preparing...                ########################################### [100%]
   1:intel-ipp              ########################################### [100%]
Installation successful.

-------------------------------------------------------------------------------

Press Enter to continue.

--- Press 'Enter'


The following Intel(R) products and related products are installed.

Intel(R) Integrated Performance Primitives Version 3.0


Which of the following would you like to install?
1.  Intel(R) Integrated Performance Primitives Version 3.0
x.  Exit

--- Press 'x' then press 'Enter'

Successfully installed:

Intel(R) Integrated Performance Primitives Version 3.0
  (intel-ipp-3.0p-2.i386.rpm)

Exiting...

[root@rebec tmp]# ll /home/intel
total 8
drwxr-xr-x    8 root     root         4096 jui 23 13:57 ipp/
drwxr-xr-x    2 root     root         4096 jui 23 13:57 licenses/

[root@rebec tmp]# ll /home/intel/ipp
total 96
drwxr-xr-x    2 root     root         4096 jui 23 13:57 doc/
drwxr-xr-x    2 root     root         4096 jui 23 13:57 include/
-r--r--r--    1 root     root         2416 jan 16  2003 ippindex.htm
-r--r--r--    1 root     root        25429 jan 16  2003 ipplic.htm
```

```
-r--r--r--    1 root     root        18092 jan 16  2003 ippnotes.htm
-r--r--r--    1 root     root          699 jan 16  2003 ippredist.txt
-r--r--r--    1 root     root         8724 jan 16  2003 ippstart.htm
drwxr-xr-x    2 root     root         4096 jui 23 13:57 lib/
drwxr-xr-x    2 root     root         4096 jui 23 13:57 sharedlib/
drwxr-xr-x    7 root     root         4096 jui 23 13:57 tools/
drwxr-xr-x    5 root     root         4096 jui 23 13:57 training/
-r-xr-xr-x    1 root     root         1755 jan 16  2003 uninstall.sh*
```

---

Now IPP library directory is /home/intel/ipp. But you cannot still use the library because it is not in the ld path where Linux looks for libraries, we will now install OpenCV and see later how to put IPP library in the path using ldconfig.

During the next part of the process, you will install OpenCV which you need to compile because it is not given as a binary installation but under a source tree form. We consider you are still in the "/tmp" directory in the next part of the process.

## 2.2 OpenCV installation

OpenCV installation under GNU Linux is more difficult than IPP because in OpenCV, you need to compile the sources.

---

```
[root@rebec tmp]# tar xvzf OpenCV-0.9.5.tar.gz
OpenCV-0.9.5/
OpenCV-0.9.5/README
OpenCV-0.9.5/AUTHORS
OpenCV-0.9.5/COPYING
OpenCV-0.9.5/ChangeLog
OpenCV-0.9.5/INSTALL

[...continued...]

OpenCV-0.9.5/samples/c/pic2.png
OpenCV-0.9.5/samples/c/pyramid_segmentation.c
OpenCV-0.9.5/samples/c/minarea.c
OpenCV-0.9.5/_dsw/
OpenCV-0.9.5/_dsw/Makefile.am
OpenCV-0.9.5/_dsw/Makefile.in
OpenCV-0.9.5/_dsw/OpenCVlight.dsw

[root@rebec tmp]# cd OpenCV-0.9.5

[root@rebec OpenCV-0.9.5]# ll
total 980
-rw-rw-r--    1 root   root     146559 mar  5 14:35 aclocal.m4
drwxrwxrwx    7 root   root       4096 mar  5 14:36 apps/
-rw-rw-r--    1 root   root       2995 jan 21  2003 AUTHORS
-rw-rw-r--    1 root   root        909 nov 11  2002 autogen.sh
-rw-rw-r--    1 root   root      78742 fv 27 20:38 ChangeLog
-rwxr-xr-x    1 root   root      39715 mar  4 14:17 config.guess*
-rwxr-xr-x    1 root   root      29483 mar  4 14:17 config.sub*
-rwxrwxr-x    1 root   root     362486 mar  5 14:36 configure*
-rw-rw-r--    1 root   root       7196 mar  5 14:35 configure.in
-rw-rw-r--    1 root   root       1892 ao  8  2001 COPYING
drwxrwxrwx    5 root   root       4096 mar  5 14:36 cv/
drwxrwxrwx    5 root   root       4096 mar  5 14:36 cvaux/
-rw-rw-r--    1 root   root       2234 mar  5 14:36 cvconfig.h.in
-rwxr-xr-x    1 root   root      12123 mar  4 14:17 depcomp*
drwxrwxrwx    3 root   root       4096 mar  5 14:36 docs/
drwxrwxrwx    2 root   root       4096 mar  5 14:36 _dsw/
```

```
-rw-rw-r--    1 root   root       8247 nov 27  2002 INSTALL
-rwxr-xr-x    1 root   root       5569 mar  4 14:17 install-sh*
-rw-rw-r--    1 root   root     138394 jun 25  2002 ltmain.sh
-rw-rw-r--    1 root   root       2061 nov 27  2002 Makefile.am
-rw-rw-r--    1 root   root        410 nov 27  2002 Makefile.bcc
-rw-rw-r--    1 root   root        342 nov 27  2002 Makefile.gcc
-rw-rw-r--    1 root   root        426 nov 27  2002 Makefile.icl
-rw-rw-r--    1 root   root      17051 mar  5 14:35 Makefile.in
-rw-rw-r--    1 root   root        419 nov 27  2002 Makefile.vc
-rwxr-xr-x    1 root   root      10270 mar  4 14:17 missing*
-rwxr-xr-x    1 root   root       1801 mar  4 14:17 mkinstalldirs*
-rw-rw-r--    1 root   root         13 nov 22  2002 NEWS
-rw-rw-r--    1 root   root       1143 nov 27  2002 opencv-config.in
-rw-rw-r--    1 root   root       3137 mar  5 14:36 OpenCV.spec
-rw-rw-r--    1 root   root       3144 nov 27  2002 OpenCV.spec.in
drwxrwxrwx    4 root   root       4096 mar  5 14:36 otherlibs/
-rw-rw-r--    1 root   root         19 nov 22  2002 README
drwxrwxrwx    3 root   root       4096 mar  5 14:36 samples/
drwxrwxrwx    4 root   root       4096 mar  5 14:36 tests/
-rw-rw-r--    1 root   root       2558 fv 26 18:36 THANKS
-rw-rw-r--    1 root   root       1980 nov 26  2002 TODO
drwxrwxrwx    3 root   root       4096 mar  5 14:36 utils/

[root@rebec OpenCV-0.9.5]# ./configure --prefix=/home/intel/opencv-0.9.5
--with-x --with-gnu-ld
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu
checking target system type... i686-pc-linux-gnu
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for gawk... gawk
checking whether make sets ${MAKE}... yes
checking for g++... g++
checking for C++ compiler default output... a.out
checking whether the C++ compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C++ compiler... yes
checking whether g++ accepts -g... yes
checking for style of include used by make... GNU
checking dependency style of g++... gcc3
checking for gcc... gcc
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking dependency style of gcc... gcc3
checking how to run the C preprocessor... gcc -E
checking for ranlib... ranlib
checking for ld used by GCC... /usr/bin/ld
checking if the linker (/usr/bin/ld) is GNU ld... yes
checking for /usr/bin/ld option to reload object files... -r
checking for BSD-compatible nm... /usr/bin/nm -B
checking whether ln -s works... yes
checking how to recognise dependant libraries... pass_all
checking command to parse /usr/bin/nm -B output... ok
checking for ANSI C header files... yes
checking for sys/types.h... yes
checking for sys/stat.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for memory.h... yes
checking for strings.h... yes
checking for inttypes.h... yes
checking for stdint.h... yes
checking for unistd.h... yes
checking dlfcn.h usability... yes
checking dlfcn.h presence... yes
checking for dlfcn.h... yes
checking for ranlib... (cached) ranlib
checking for strip... strip
checking for objdir... .libs
checking for gcc option to produce PIC... -fPIC
checking if gcc PIC flag -fPIC works... yes
checking if gcc static flag -static works... no
```

```
checking if gcc supports -c -o file.o... yes
checking if gcc supports -c -o file.lo... yes
checking if gcc supports -fno-rtti -fno-exceptions... yes
checking whether the linker (/usr/bin/ld) supports shared
libraries... yes
checking how to hardcode library paths into programs... immediate
checking whether stripping libraries is possible... yes
checking dynamic linker characteristics... GNU/Linux ld.so
checking if libtool supports shared libraries... yes
checking whether to build shared libraries... yes
checking whether to build static libraries... no
checking whether -lc should be explicitly linked in... no
creating libtool
checking for X... libraries /usr/X11R6/lib, headers /usr/X11R6/include
checking whether to build debug version (no optimization)... no
configure: TARGET=i686-pc-linux-gnu
checking jpeglib.h usability... yes
checking jpeglib.h presence... yes
checking for jpeglib.h... yes
checking for jpeg_destroy_decompress in -ljpeg... yes
checking zlib.h usability... yes
checking zlib.h presence... yes
checking for zlib.h... yes
checking for gzopen in -lz... yes
checking png.h usability... yes
checking png.h presence... yes
checking for png.h... yes
checking libpng/png.h usability... yes
checking libpng/png.h presence... yes
checking for libpng/png.h... yes
checking for png_read_rows in -lpng... yes
checking for png_get_valid... yes
checking for png_set_tRNS_to_alpha... yes
checking tiff.h usability... no
checking tiff.h presence... no
checking for tiff.h... no
checking for /usr/X11R6/include/Xm/Xm.h... yes
checking for main in -lm... yes
checking ffmpeg/avcodec.h usability... no
checking ffmpeg/avcodec.h presence... no
checking for ffmpeg/avcodec.h... no
checking for gethostbyname... yes
checking for connect... yes
checking for remove... yes
checking for shmat... yes
checking for IceConnectionNumber in -lICE... yes
configure: creating ./config.status
config.status: creating Makefile
config.status: creating OpenCV.spec
config.status: creating _dsw/Makefile
config.status: creating cv/Makefile
config.status: creating cv/include/Makefile
config.status: creating cv/src/Makefile
config.status: creating cvaux/Makefile
config.status: creating cvaux/include/Makefile
config.status: creating cvaux/src/Makefile
config.status: creating docs/Makefile
config.status: creating samples/Makefile
config.status: creating samples/c/Makefile
config.status: creating otherlibs/Makefile
config.status: creating otherlibs/highgui/Makefile
config.status: creating otherlibs/cvcam/Makefile
config.status: creating otherlibs/cvcam/include/Makefile
config.status: creating otherlibs/cvcam/src/Makefile
config.status: creating otherlibs/cvcam/src/unix/Makefile
config.status: creating tests/Makefile
config.status: creating tests/trs/Makefile
config.status: creating tests/trs/include/Makefile
config.status: creating tests/trs/src/Makefile
config.status: creating tests/cv/Makefile
config.status: creating tests/cv/src/Makefile
config.status: creating apps/Makefile
config.status: creating apps/cvcsdemo/Makefile
config.status: creating apps/cvcsdemo/pictures/Makefile
```

```
config.status: creating apps/vmdemotk/Makefile
config.status: creating apps/cvlkdemo/Makefile
config.status: creating apps/cvlkdemo/pictures/Makefile
config.status: creating apps/HaarFaceDetect/Makefile
config.status: creating apps/haartraining/Makefile
config.status: creating apps/haartraining/src/Makefile
config.status: creating utils/Makefile
config.status: creating utils/cvinfo/Makefile
config.status: creating opencv-config
config.status: creating cvconfig.h
config.status: executing depfiles commands
config.status: executing default commands


[root@rebec OpenCV-0.9.5]# make
make  all-recursive
make[1]: Entre dans le rpertoire `/tmp/OpenCV-0.9.5'
Making all in cv
make[2]: Entre dans le rpertoire `/tmp/OpenCV-0.9.5/cv'
Making all in src
make[3]: Entre dans le rpertoire `/tmp/OpenCV-0.9.5/cv/src'
source='camshift.cpp' object='camshift.lo' libtool=yes \
depfile='.deps/camshift.Plo' tmpdepfile='.deps/camshift.TPlo' \
depmode=gcc3 /bin/sh ../../depcomp \

[...continued...]

creating cvtest
make[4]: Quitte le rpertoire `/tmp/OpenCV-0.9.5/tests/cv/src'
make[4]: Entre dans le rpertoire `/tmp/OpenCV-0.9.5/tests/cv'
make[4]: Rien  faire pour `all-am'.
make[4]: Quitte le rpertoire `/tmp/OpenCV-0.9.5/tests/cv'
make[3]: Quitte le rpertoire `/tmp/OpenCV-0.9.5/tests/cv'
make[3]: Entre dans le rpertoire `/tmp/OpenCV-0.9.5/tests'
make[3]: Rien  faire pour `all-am'.
make[3]: Quitte le rpertoire `/tmp/OpenCV-0.9.5/tests'
make[2]: Quitte le rpertoire `/tmp/OpenCV-0.9.5/tests'
make[2]: Entre dans le rpertoire `/tmp/OpenCV-0.9.5'
make[2]: Rien  faire pour `all-am'.
make[2]: Quitte le rpertoire `/tmp/OpenCV-0.9.5'
make[1]: Quitte le rpertoire `/tmp/OpenCV-0.9.5'


[root@rebec OpenCV-0.9.5]# make install
Making install in cv
make[1]: Entre dans le rpertoire `/tmp/OpenCV-0.9.5/cv'
Making install in src
make[2]: Entre dans le rpertoire `/tmp/OpenCV-0.9.5/cv/src'
make[3]: Entre dans le rpertoire `/tmp/OpenCV-0.9.5/cv/src'
/bin/sh ../../mkinstalldirs /home/intel/opencv-0.9.5/lib
mkdir -p -- /home/intel/opencv-0.9.5/lib
 /bin/sh ../../libtool --mode=install /usr/bin/install -c  libopencv.la
/home/intel/opencv-0.9.5/lib/libopencv.la
/usr/bin/install -c .libs/libopencv-0.9.so.5.0.0
/home/intel/opencv-0.9.5/lib/libopencv-0.9.so.5.0.0
(cd /home/intel/opencv-0.9.5/lib && rm -f libopencv-0.9.so.5 &&
ln -s libopencv-0.9.so.5.0.0 libopencv-0.9.so.5)
(cd /home/intel/opencv-0.9.5/lib && rm -f libopencv.so &&
ln -s libopencv-0.9.so.5.0.0 libopencv.so)
/usr/bin/install -c .libs/libopencv.lai
/home/intel/opencv-0.9.5/lib/libopencv.la
PATH="$PATH:/sbin" ldconfig -n /home/intel/opencv-0.9.5/lib
----------------------------------------------------------------------
Libraries have been installed in:
   /home/intel/opencv-0.9.5/lib

If you ever happen to want to link against installed libraries
in a given directory, LIBDIR, you must either use libtool, and
specify the full pathname of the library, or use the `-LLIBDIR'
flag during linking and do at least one of the following:
   - add LIBDIR to the `LD_LIBRARY_PATH' environment variable
     during execution
   - add LIBDIR to the `LD_RUN_PATH' environment variable
     during linking
```

```
   - use the '-Wl,--rpath -Wl,LIBDIR' linker flag
   - have your system administrator add LIBDIR to '/etc/ld.so.conf'

See any operating system documentation about shared libraries for
more information, such as the ld(1) and ld.so(8) manual pages.
----------------------------------------------------------------------
/bin/sh ../../mkinstalldirs /home/intel/opencv-0.9.5/include
mkdir -p -- /home/intel/opencv-0.9.5/include
make[3]: Quitte le rpertoire '/tmp/OpenCV-0.9.5/cv/src'
make[2]: Quitte le rpertoire '/tmp/OpenCV-0.9.5/cv/src'
Making install in include

[...continued...]

make[2]: Entre dans le rpertoire '/tmp/OpenCV-0.9.5'
/bin/sh ./mkinstalldirs /home/intel/opencv-0.9.5/bin
mkdir -p -- /home/intel/opencv-0.9.5/bin
 /usr/bin/install -c opencv-config
/home/intel/opencv-0.9.5/bin/opencv-config
make[2]: Rien  faire pour 'install-data-am'.
make[2]: Quitte le rpertoire '/tmp/OpenCV-0.9.5'
make[1]: Quitte le rpertoire '/tmp/OpenCV-0.9.5'
[root@rebec OpenCV-0.9.5]# ll /home/intel
total 12
drwxr-xr-x    8 root     root         4096 jui 23 13:57 ipp/
drwxr-xr-x    2 root     root         4096 jui 23 13:57 licenses/
drwxr-xr-x    5 root     root         4096 jui 23 14:32 opencv-0.9.5/
[root@rebec OpenCV-0.9.5]# ll /home/intel/opencv-0.9.5
total 12
drwxr-xr-x    2 root     root         4096 jui 23 14:32 bin/
drwxr-xr-x    3 root     root         4096 jui 23 14:32 include/
drwxr-xr-x    2 root     root         4096 jui 23 14:32 lib/
[root@rebec OpenCV-0.9.5]# cp -R docs/ /home/intel/opencv-0.9.5/
[root@rebec OpenCV-0.9.5]# ll /home/intel/opencv-0.9.5/
total 16
drwxr-xr-x    2 root     root         4096 jui 23 14:32 bin/
drwxr-xr-x    3 root     root         4096 jui 23 14:34 docs/
drwxr-xr-x    3 root     root         4096 jui 23 14:32 include/
drwxr-xr-x    2 root     root         4096 jui 23 14:32 lib/
[root@rebec OpenCV-0.9.5]# cd /home/intel/
[root@rebec intel]# ll
total 12
drwxr-xr-x    8 root     root         4096 jui 23 13:57 ipp/
drwxr-xr-x    2 root     root         4096 jui 23 13:57 licenses/
drwxr-xr-x    6 root     root         4096 jui 23 14:34 opencv-0.9.5/
[root@rebec intel]# ln -s opencv-0.9.5 opencv
[root@rebec intel]# ll
total 12
drwxr-xr-x    8 root     root         4096 jui 23 13:57 ipp/
drwxr-xr-x    2 root     root         4096 jui 23 13:57 licenses/
lrwxrwxrwx    1 root     root           12 jui 23 14:35 opencv -> opencv-0.9.5/
drwxr-xr-x    6 root     root         4096 jui 23 14:34 opencv-0.9.5/
```

---

As you can see in the last part of the installation process, I have created a symbolic link between the real directory "opencv-0.9.5" and the virtual one "opencv" using the "ln -s" command. It can be very helpful to test new versions of OpenCV. When you will install another version of the library, you will only need to change the opencv virtual link to link to the new version to test your programs.

As for IPP, we need to tell to Linux where it will find OpenCV library binaries. We will see it in the next part.

## 2.3 Linux and libraries. . .

Now that we have installed IPP and OpenCV, we need to specify to Linux where are the libraries in order to use them. Linux uses the file "/etc/ld.so.conf" to know the path of libraries needed at runtime for each executable launched. So we must modify this file to make Linux able to find our new libraries.

I used "vi" to modify "/etc/ld.so.conf" file, but you can use many others text editors (such as kedit, xedit, gedit...) especially if you are not familiar with "vi". Because you modifiy a part of the system (the path for dynamic libraries), ou need to be root to do so. Once "/etc/ld.so.conf" modified, you must use the command "ldconfig -v" to make Linux understand there are two more paths for dynamic libraries.

---

```
[root@rebec intel]# more /etc/ld.so.conf
/usr/X11R6/lib
/usr/lib/qt3/lib

[root@rebec intel]# vi /etc/ld.so.conf

[root@rebec intel]# more /etc/ld.so.conf
/usr/X11R6/lib
/usr/lib/qt3/lib
/home/intel/ipp/sharedlib
/home/intel/opencv/lib

[root@rebec intel]# ldconfig -v
/lib:
libacl.so.1 -> libacl.so.1.0.0
libattr.so.1 -> libattr.so.1.0.0
libgcc_s.so.1 -> libgcc_s-3.2.so.1
libproc.so.2.0.7 -> libproc.so.2.0.7
libpam_misc.so.0 -> libpam_misc.so.0.75
libpam.so.0 -> libpam.so.0.75
libpopt.so.0 -> libpopt.so.0.0.0
libuuid.so.1 -> libuuid.so.1.2

[...continued...]

/home/intel/ipp/sharedlib:
libippvmw7.so -> libippvmw7.so
libippvmpx.so -> libippvmpx.so
libippvmm6.so -> libippvmm6.so
libippvmi7.so -> libippvmi7.so
libippvma6.so -> libippvma6.so
libippvm64.so -> libippvm64.so
libippvm.so -> libippvm.so
libippvcw7.so -> libippvcw7.so
libippvcpx.so -> libippvcpx.so
libippvcm6.so -> libippvcm6.so
libippvci7.so -> libippvci7.so
libippvca6.so -> libippvca6.so
libippvc64.so -> libippvc64.so
libippvc.so -> libippvc.so
libippsw7.so -> libippsw7.so
libippsrw7.so -> libippsrw7.so
libippsrpx.so -> libippsrpx.so
libippsrm6.so -> libippsrm6.so
libippsri7.so -> libippsri7.so
libippsra6.so -> libippsra6.so
libippsr64.so -> libippsr64.so
libippsr.so -> libippsr.so
libippspx.so -> libippspx.so
libippsm6.so -> libippsm6.so
libippsi7.so -> libippsi7.so
```

```
libippscw7.so -> libippscw7.so
libippscpx.so -> libippscpx.so
libippscm6.so -> libippscm6.so
libippsci7.so -> libippsci7.so
libippsca6.so -> libippsca6.so
libippsc64.so -> libippsc64.so
libippsc.so -> libippsc.so
libippsa6.so -> libippsa6.so
libipps64.so -> libipps64.so
libipps.so -> libipps.so
libippmw7.so -> libippmw7.so
libippmpx.so -> libippmpx.so
libippmpw7.so -> libippmpw7.so
libippmppx.so -> libippmppx.so
libippmpm6.so -> libippmpm6.so
libippmpi7.so -> libippmpi7.so
libippmpa6.so -> libippmpa6.so
libippmp64.so -> libippmp64.so
libippmp.so -> libippmp.so
libippmm6.so -> libippmm6.so
libippmi7.so -> libippmi7.so
libippma6.so -> libippma6.so
libippm64.so -> libippm64.so
libippm.so -> libippm.so
libippjw7.so -> libippjw7.so
libippjpx.so -> libippjpx.so
libippjm6.so -> libippjm6.so
libippji7.so -> libippji7.so
libippja6.so -> libippja6.so
libippj64.so -> libippj64.so
libippj.so -> libippj.so
libippiw7.so -> libippiw7.so
libippipx.so -> libippipx.so
libippim6.so -> libippim6.so
libippii7.so -> libippii7.so
libippia6.so -> libippia6.so
libippi64.so -> libippi64.so
libippi.so -> libippi.so
libippcvw7.so -> libippcvw7.so
libippcvpx.so -> libippcvpx.so
libippcvi7.so -> libippcvi7.so
libippcva6.so -> libippcva6.so
libippcv64.so -> libippcv64.so
libippcv.so -> libippcv.so
libippcore64.so -> libippcore64.so
libippcore.so -> libippcore.so
libippacw7.so -> libippacw7.so
libippacpx.so -> libippacpx.so
libippacm6.so -> libippacm6.so
libippaci7.so -> libippaci7.so
libippaca6.so -> libippaca6.so
libippac.so -> libippac.so
libippac64.so -> libippac64.so
/home/intel/opencv/lib:
libcvcam-0.9.so.5 -> libcvcam-0.9.so.5.0.0
libhighgui-0.9.so.5 -> libhighgui-0.9.so.5.0.0
libcvaux-0.9.so.5 -> libcvaux-0.9.so.5.0.0
libopencv-0.9.so.5 -> libopencv-0.9.so.5.0.0
/lib/i686: (hwcap: 0x8000000000000)
librt.so.1 -> librt-2.2.5.so
libm.so.6 -> libm-2.2.5.so
libc.so.6 -> libc-2.2.5.so
libpthread.so.0 -> libpthread-0.9.so

[root@rebec intel]# exit

[jeje@rebec jeje]$
```

---

Once you have finished the installation, you go back to your simple user account by exiting the root shell. Then you can create your own directory structure to build your IPP and OpenCV applications.

## 2.4 Directory structure

To test our installation and all the examples of C code, we must create a directory in which all the examples will be placed. So a directory called "ippopencv" is created (using "mkdir") and a subdirectory containing all the images is also created ("images").

After that, I have copied my images ".jpg" and ".bmp" from the "/tmp" directory to the images directory. You will need two images to test our system, I have chosen a bitmap image ("example.bmp") and a JPEG image ("example.bmp") to illustrate the examples. These are images from the french "Mont Saint-Michel" in Normandy (which is a very nice place to visit during your holidays).

---

```
[jeje@rebec jeje]$ mkdir ippopencv

[jeje@rebec jeje]$ cd ippopencv

[jeje@rebec ippopencv]$ mkdir images

[jeje@rebec ippopencv]$ ll
total 4
drwxr-xr-x    2 jeje     jeje         4096 jui 23 14:45 images/

[jeje@rebec ippopencv]$ cp /tmp/*.jpg images/

[jeje@rebec ippopencv]$ cp /tmp/*.bmp images/

[jeje@rebec ippopencv]$ ll images
total 372
-r-xr-xr-x    1 jeje     jeje       360054 jui 23 16:46 example.bmp
-r-xr-xr-x    1 jeje     jeje        14848 jui 23 16:46 example.jpg
```

---

The difference between the ".bmp" file and the ".jpg" file is that the JPEG format is compressed with loss of information. That's why the size of the ".bmp" file is so big compared to the ".jpg" file.

Now we are ready to begin with our examples, first under OpenCV, then under IPP, and finally under these two libraries together.

# Chapter 3

# Basic OpenCV

*"Un bon croquis vaut mieux qu'un long discours."*
*"A picture is worth a thousand words."*
**Napoléon $I^{er}$ — 1769-1821**

## 3.1 General information

OpenCV uses the "IplImage" structure to create and handle images. This structure has a lot of fields explained as they will appear in examples. Several fields are more important than others. For instance "width" is the width of the IplImage, "height" the height, "depth" the depth in bits and "nChannels" the number of channels (one for gray-levels images and three for color images).

---

```
typedef struct _IplImage {
    int  nSize;         /* sizeof(IplImage) */
    int  ID;            /* version (=0)*/
    int  nChannels;     /* Most of OpenCV functions support 1,2,3 or 4 channels */
    int  alphaChannel;  /* ignored by OpenCV */
    int  depth;         /* pixel depth in bits: IPL_DEPTH_8U, IPL_DEPTH_8S,
   IPL_DEPTH_16S, IPL_DEPTH_32S, IPL_DEPTH_32F
   and IPL_DEPTH_64F are supported */
    char colorModel[4]; /* ignored by OpenCV */
    char channelSeq[4]; /* ditto */
    int  dataOrder;     /* 0 - interleaved color channels,
   1 - separate color channels.
                           cvCreateImage can only create interleaved images */
    int  origin;        /* 0 - top-left origin,
                           1 - bottom-left origin (Windows bitmaps style) */
    int  align;         /* Alignment of image rows (4 or 8).
                           OpenCV ignores it and uses widthStep instead */
    int  width;         /* image width in pixels */
    int  height;        /* image height in pixels */
    struct _IplROI *roi;/* image ROI. if NULL, the whole image is selected */
    struct _IplImage *maskROI; /* must be NULL */
    void  *imageId;     /* ditto */
    struct _IplTileInfo *tileInfo; /* ditto */
    int  imageSize;     /* image data size in bytes
                           (==image->height*image->widthStep
                           in case of interleaved data)*/
```

```
    char *imageData;  /* pointer to aligned image data */
    int  widthStep;   /* size of aligned image row in bytes */
    int  BorderMode[4]; /* ignored by OpenCV */
    int  BorderConst[4]; /* ditto */
    char *imageDataOrigin; /* pointer to very origin of image data
                              (not necessarily aligned) -
                              needed for correct deallocation */
}
IplImage;
```

---

Each function name in OpenCV starts with "cv", for instance "cvCreateImage", "cvSobel", "cvAdd"... In the HTML documentation, you will find only the name of each function without the prefix "cv". You must add it to use any function of OpenCV.

OpenCV comes with a graphical interface called highGUI (high graphical user interface). This graphical interface is very important because you need it under OpenCV to visualize your images but also under IPP because IPP has no graphical interface. Now let's go programming...

## 3.2 Creating an image

Our first program will simply allow us to try our installation of OpenCV with a classical and small example which consists in creating an image, filling it with simple values and displaying it on the screen under X-window.

Our first program must be called "opencv0.c" and must be located in "ippopencv" (our main directory).

---

```
//
// opencv0.c - creating and displaying an image using Intel OpenCV
//

#include "cv.h" // includes OpenCV definitions
#include "highgui.h" // includes highGUI definitions
#include <stdio.h>// includes C standard input/output definitions

int main()
{
IplImage *cvImg; // image used for visualisation
CvSize imgSize; // size of visualisation image
int i = 0, j = 0;

imgSize.width = 640; // visualisation image is
imgSize.height = 480; // 640x480 pixels
// creation of a 8 bits depth gray image
cvImg = cvCreateImage( imgSize, 8, 1 );

// image is filled with gray values
// corresponding to the 256 modulus
// of their position in the image
for ( i = 0; i < imgSize.width; i++ )
for ( j = 0; j < imgSize.height; j++ )
((uchar*)(cvImg->imageData + cvImg->widthStep*j))[i] =
( char ) ( ( i * j ) % 256 );

cvNamedWindow( "Testing OpenCV...", 1 ); // creation of a visualisation window
cvShowImage( "Testing OpenCV...", cvImg ); // image visualisation
```

```
cvWaitKey( 0 ); // wait for key

cvDestroyWindow( "image" ); // close window

cvReleaseImage( &cvImg ); // memory release before exiting the application

return( 0 ); // stopping our first program
}
```

---

This program creates a gray level image using structure IplImage (which is the type of each image under OpenCV) and fills it with gray values before displaying it. The user must press a key to stop the application. To access pixels values, you must use the given pointer descriptor "((uchar*)(cvImg->imageData + cvImg->widthStep*j))[i]". This allows to access to the pixel of coordinates [i,j] ([col,row]) in the image (because it is a gray image, the formula is more complex in three channels images). For more information, see the FAQS of OpenCV. It describes very well pixel and matrix elements access.

To compile this program, we must use the following command line. In this case, we assume that our previous installation worked, that "/etc/ld.so.conf" has been modified and that "ldconfig -v" worked.

---

```
[jeje@rebec ippopencv]$ ll
total 8
drwxrwxr-x    2 jeje     jeje         4096 jui 23 17:50 images
-rw-rw-r--    1 jeje     jeje         1147 jui 23 16:46 opencv0.c

[jeje@rebec ippopencv]$ gcc -I/home/intel/opencv/include -L/home/intel/opencv/lib
-lopencv -lhighgui -lstdc++ opencv0.c -o opencv0

[jeje@rebec ippopencv]$ ll
total 56
drwxrwxr-x    2 jeje     jeje         4096 jui 23 17:50 images
-rwxrwxr-x    1 jeje     jeje        46502 jui 23 16:46 opencv0
-rw-rw-r--    1 jeje     jeje         1147 jui 23 16:46 opencv0.c
```

---

Compilation is made using gcc, "-I/home/intel/opencv/include" tells the compiler where to find include files ".h" for OpenCV, "-L/home/intel/opencv/lib" gives the library path. We must use "-lstdc++" which is the standard C++ library in order to compile properly because OpenCV is based on this standard library. If there are no errors, you should obtain an executable file called opencv0, you must run it to see the result as explained as follows.

---

```
[jeje@rebec ippopencv]$ ll
total 56
drwxrwxrwx    2 jeje     jeje         4096 jui 23 16:44 images
-rwxrwxr-x    1 jeje     jeje        46502 jui 23 16:37 opencv0
-rw-rw-r--    1 jeje     jeje         1147 jui 23 16:33 opencv0.c

[jeje@rebec ippopencv]$ ./opencv0
```

```
--- your first image must appear on the screen ---
--- press a key to exit ---

[jeje@rebec ippopencv]$
```

Your first OpenCV image must appear on the screen, the result is shown on figure 3.1.

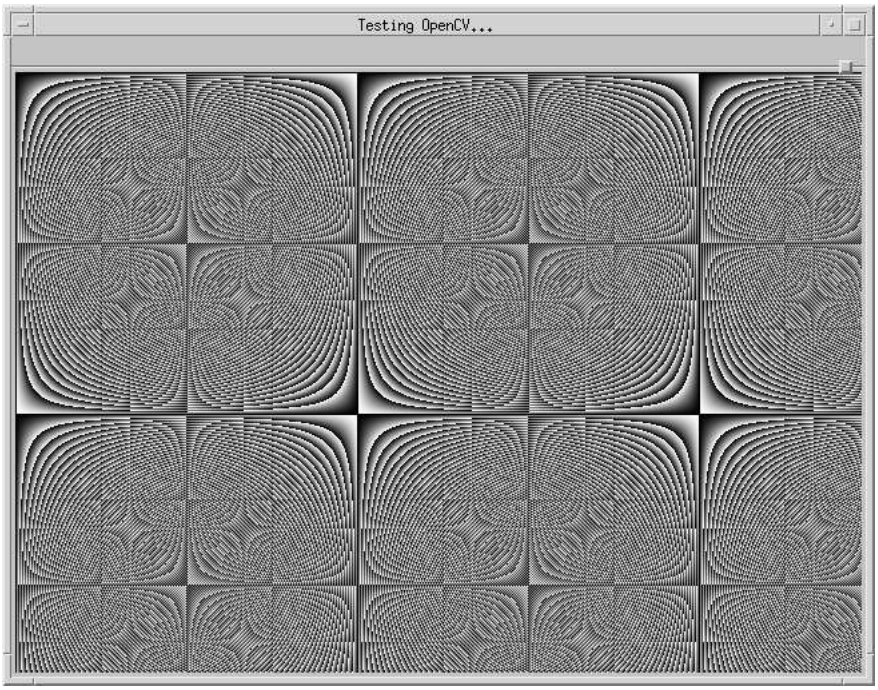


Figure 3.1: Our first image created with OpenCV

## 3.3 Loading and displaying files

Our next example is important because instead of creating an image, we are going to load a bitmap (".bmp") file and a JPEG (".jpg") file. HighGUI (high Graphic User Interface) gives the possibility to load, save and display images on the screen, which is impossible directly with IPP. So we will see later how IPP uses highGUI to display images under X.

```
//
// opencv1.c - loading and displaying two images using OpenCV
//

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

```
// file example.bmp in our images directory
char name0[] = "images/example.bmp";
// file example.jpg in our images directory
char name1[] = "images/example.jpg";

int main()
{
// new IplImage structure img0
IplImage* img0 = NULL;
// new IplImage structure img1
IplImage* img1 = NULL;

// example file example.bmp is loaded as img0
img0 = cvLoadImage( name0, -1 );
// example file example.jpg is loaded as img1
  img1 = cvLoadImage( name1, -1 );

// a visualization window is created with title image0
cvNamedWindow( "image0", 1 );
// a visualization window is created with title image1
cvNamedWindow( "image1", 1 );

// img0 is shown in window image0
cvShowImage( "image0", img0 );
// img1 is shown in window image1
cvShowImage( "image1", img1 );

// wait for key to close the windows
cvWaitKey(0);

// memory release for img0 before exiting the application
cvReleaseImage( &img0 );
// memory release for img1 before exiting the application
cvReleaseImage( &img1 );

return(0);
}
```

Compile your program and run it with:

```
[jeje@rebec ippopencv]$ gcc -I/home/intel/opencv/include -L/home/intel/opencv/lib
-lopencv -lhighgui -lstdc++ opencv1.c -o opencv1

[jeje@rebec ippopencv]$ ./opencv1

--- your images must appear on the screen ---
--- press a key to exit ---

[jeje@rebec ippopencv]$
```

Two windows might appear on your X screen. Press any key to close them. The result is shown on figure 3.2.

## 3.4 Makefile

The command line to compile our application is quite long, so we can define a "Makefile" file to use the "make" command to build automatically our programs. You define "BIN" to be the list of programs to compile and for each, you define the list of files that will launch compilation when changed. For instance, opencv1 depends on opencv1.c and

Makefile, so when one of these files change, the next "make" command will launch the compilation of opencv1.

I wrote "Makefile" using "vi", but again you can use every text editor you know under Linux. The last part of the file is useful to clean the directory with a "make clean" command. By default, "make" uses "make all". But you can ask the compilation of any file alone, for instance "make opencv1" will compile only "opencv1.c" if necessary.

---

```
[jeje@rebec ippopencv]$ vi Makefile

[jeje@rebec ippopencv]$ more Makefile
CC      = gcc -O2 #-march=i686 -mcpu=i686
INC     = -I/home/intel/opencv/include/opencv # opencv include path
LIB     = -L/home/intel/opencv/lib # opencv libs path
OPT     = -lopencv -lhighgui -lstdc++ # standard libraries needed to link

BIN     = opencv0 opencv1 # executables to create

all:    $(BIN) # make all creates opencv0 and opencv1

opencv0: opencv0.c Makefile
                $(CC) opencv0.c $(INC) $(LIB) $(OPT) -o $@
opencv1: opencv1.c Makefile
                $(CC) opencv1.c $(INC) $(LIB) $(OPT) -o $@
clean:
                rm *~ *.o $(BIN) core


[jeje@rebec ippopencv]$ make
gcc -O2  opencv0.c -I/home/intel/opencv/include/opencv  -L/home/intel/opencv/lib
-lopencv -lhighgui -lstdc++  -o opencv0
gcc -O2  opencv1.c -I/home/intel/opencv/include/opencv  -L/home/intel/opencv/lib
-lopencv -lhighgui -lstdc++  -o opencv1

[jeje@rebec ippopencv]$ ll
total 112
drwxrwxr-x    2 jeje     jeje         4096 jui 23 19:01 images
-rw-------    1 jeje     jeje          721 jui 23 18:53 Makefile
-rwxrwxr-x    1 jeje     jeje        46502 jui 23 18:53 opencv0
-rw-rw-r--    1 jeje     jeje         1147 jui 23 18:52 opencv0.c
-rwxrwxr-x    1 jeje     jeje        46584 jui 23 18:53 opencv1
-rw-------    1 jeje     jeje         1158 jui 23 18:52 opencv1.c

[jeje@rebec ippopencv]$ ./opencv1

--- your images must appear on the screen ---
--- press a key to exit ---

[jeje@rebec ippopencv]$
```

---

With this "Makefile", you only need to type "make" and the compile process will begin automatically.

## 3.5 JPEG and BMP are different

Our next example proposes to apply an image processing function to a loaded image and to display the result. To understand that ".bmp" files and ".jpg" differ (because of jpeg lossly compression), we will substract the two images previously loaded and display the result on the screen.

---

```
//
// opencv2.c - difference between two images with scaling of the result
//

#include "cv.h"
#include "highgui.h"
#include <stdio.h>

// file example.bmp in our images directory
char name0[] = "images/example.bmp";
// file example.jpg in our images directory
char name1[] = "images/example.jpg";

int main()
{
// new IplImage structure img0
IplImage* img0 = NULL;
// new IplImage structure img1
IplImage* img1 = NULL;
// new IplImage structure result image res
IplImage* res = NULL;
// size of our result image res
CvSize imgSize;

// example file "example.bmp" is loaded as img0
img0 = cvLoadImage( name0, -1 );
// example file "example.jpg" is loaded as img1
img1 = cvLoadImage( name1, -1 );

// visualisation image is the same size than img0
imgSize.width = img0->width;
imgSize.height = img0->height;
// creation of a 8 bits depth color image
res = cvCreateImage( imgSize, 8, 3 );

// substract img0 from img1 and store the result in res
cvSub( img0, img1, res, 0 );

// scale result (which is small) not to see a black image res
cvConvertScale( res, res, 30, 0 );

// a visualization window is created with title "image"
cvNamedWindow( "image", 1 );

// res is shown in window "image"
cvShowImage( "image", res );

// wait for key to close the window
cvWaitKey(0);

// memory release for img0 before exiting the application
cvReleaseImage( &img0 );
// memory release for img1 before exiting the application
cvReleaseImage( &img1 );
// memory release for img1 before exiting the application
cvReleaseImage( &res );

return(0);
}
```

---

You have to add an "opencv2" entry in the "BIN" line of our "Makefile" and to define "opencv2".

---

```
[jeje@rebec ippopencv]$ vi Makefile

[jeje@rebec ippopencv]$ more Makefile
CC      = gcc -O2 #-march=i686 -mcpu=i686
```

```
INC     = -I/home/intel/opencv/include/opencv # opencv include path
LIB     = -L/home/intel/opencv/lib # opencv libs path
OPT     = -lopencv -lhighgui -lstdc++ # standard libraries needed to link

BIN     = opencv0 opencv1 opencv2 # executables to create

all:    $(BIN) # make all creates opencv0, opencv1...

opencv0: opencv0.c Makefile
                $(CC) opencv0.c $(INC) $(LIB) $(OPT) -o $@
opencv1: opencv1.c Makefile
                $(CC) opencv1.c $(INC) $(LIB) $(OPT) -o $@
opencv2: opencv2.c Makefile
                $(CC) opencv2.c $(INC) $(LIB) $(OPT) -o $@
clean:
                rm *~ *.o $(BIN) core

[jeje@rebec ippopencv]$ make
gcc -O2  opencv0.c -I/home/intel/opencv/include/opencv  -L/home/intel/opencv/lib
-lopencv -lhighgui -lstdc++  -o opencv0
gcc -O2  opencv1.c -I/home/intel/opencv/include/opencv  -L/home/intel/opencv/lib
-lopencv -lhighgui -lstdc++  -o opencv1
gcc -O2  opencv2.c -I/home/intel/opencv/include/opencv  -L/home/intel/opencv/lib
-lopencv -lhighgui -lstdc++  -o opencv2

[jeje@rebec ippopencv]$ ./opencv2

[jeje@rebec ippopencv]$
```

The difference is computed using "cvSub" and the result is scaled (because it is too small) by a factor of 30. The difference between "example.bmp" and "example.jpg" is shown on figure 3.3.

## 3.6 Color conversion using OpenCV

This example converts a RGB color image into the HSV colorspace and displays the result on the screen.

```
//
// opencv3.c - Color conversion
//

#include "cv.h"
#include "highgui.h"
#include <stdio.h>

// file example.bmp in our images directory
char name[] = "images/example.bmp";

int main()
{
// new IplImage structure img
IplImage* img = NULL;
// new IplImage structure res
IplImage* res = NULL;
// size of our result image res
CvSize imgSize;

// example file example.bmp is loaded as img
img = cvLoadImage( name, -1 );

// result size is the same as loaded image size
imgSize.width = img->width;
imgSize.height = img->height;
```

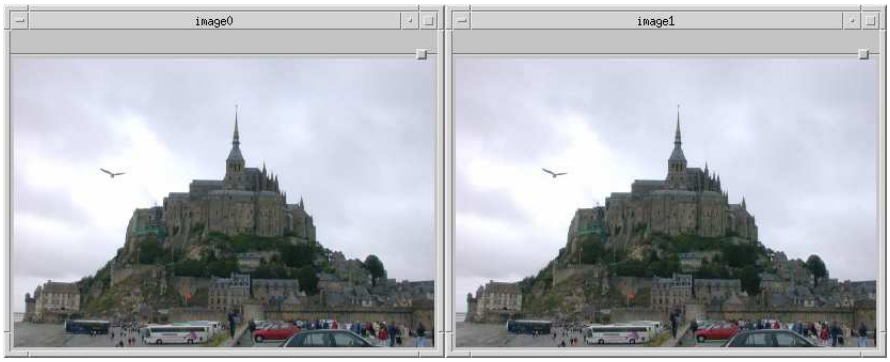


Figure 3.2: Loading and displaying a ".jpg" file and a ".bmp" file.

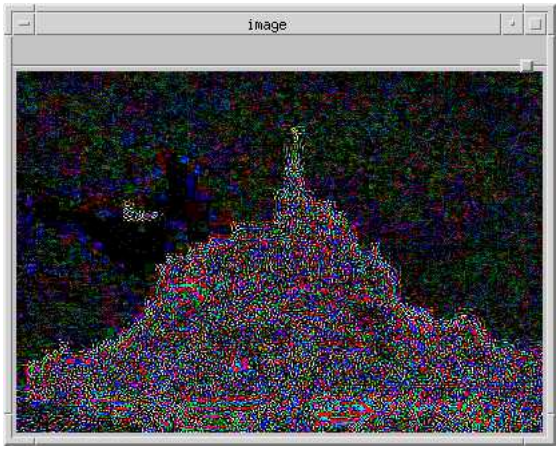


Figure 3.3: Difference between lossy compression in "JPEG" and no compression in "BMP".

```
// creation of result image res
res = cvCreateImage( imgSize, img->depth, img->nChannels );

// color conversion from RGB to HSV colorspace
cvCvtColor( img, res, CV_RGB2HSV );

// a visualization window is created with title "image"
cvNamedWindow( "image", 1 );

// res is shown in window image
cvShowImage( "image", res );

// wait for key to close the window
cvWaitKey(0);

// memory release for img before exiting the application
cvReleaseImage( &img );
// memory release for res before exiting the application
cvReleaseImage( &res );

return(0);
}
```

Figure 3.4 shows the HSV colorspace conversion of our original image. Our Makefile changes:

```
[jeje@rebec ippopencv]$ vi Makefile

[jeje@rebec ippopencv]$ more Makefile
CC      = gcc -O2 #-march=i686 -mcpu=i686
INC     = -I/home/intel/opencv/include/opencv # opencv include path
LIB     = -L/home/intel/opencv/lib # opencv libs path
OPT     = -lopencv -lhighgui -lstdc++ # standard libraries needed to link

BIN     = opencv0 opencv1 opencv2 opencv3 # executables to create

all:    $(BIN) # make all creates opencv0, opencv1...

opencv0: opencv0.c Makefile
                $(CC) opencv0.c $(INC) $(LIB) $(OPT) -o $@
opencv1: opencv1.c Makefile
                $(CC) opencv1.c $(INC) $(LIB) $(OPT) -o $@
opencv2: opencv2.c Makefile
                $(CC) opencv2.c $(INC) $(LIB) $(OPT) -o $@
opencv3: opencv3.c Makefile
                $(CC) opencv3.c $(INC) $(LIB) $(OPT) -o $@
clean:
                rm *~ *.o $(BIN) core

[jeje@rebec ippopencv]$ make
gcc -O2  opencv0.c -I/home/intel/opencv/include/opencv  -L/home/intel/opencv/lib
-lopencv -lhighgui -lstdc++  -o opencv0
gcc -O2  opencv1.c -I/home/intel/opencv/include/opencv  -L/home/intel/opencv/lib
-lopencv -lhighgui -lstdc++  -o opencv1
gcc -O2  opencv2.c -I/home/intel/opencv/include/opencv  -L/home/intel/opencv/lib
-lopencv -lhighgui -lstdc++  -o opencv2
gcc -O2  opencv3.c -I/home/intel/opencv/include/opencv  -L/home/intel/opencv/lib
-lopencv -lhighgui -lstdc++  -o opencv3

[jeje@rebec ippopencv]$ ./opencv3

--- your image must appear on the screen ---
--- press a key to exit ---

[jeje@rebec ippopencv]$
```

We have seen several examples to show how OpenCV works. Now, we will introduce OpenCV and see what kind of interactions are possible between the two libraries.

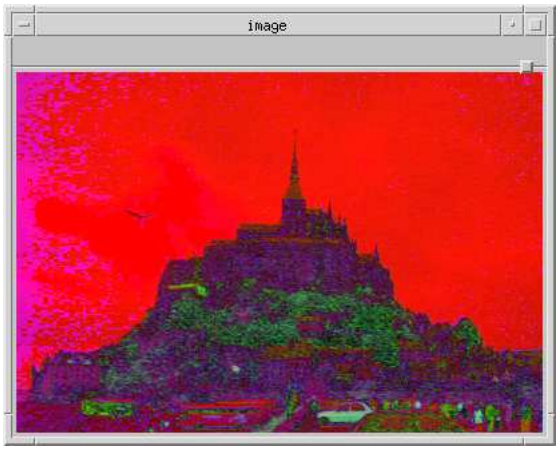


Figure 3.4: Color conversion between RGB and HSV colorspaces.

# Chapter 4

# Basic IPP

*"Enseigner, ce n'est pas remplir un vase, c'est allumer un feu."*
*"Teaching is not filling a vase, it is lightning a fire."*
**Michel de Montaigne — 1533-1592**

## 4.1 General information

IPP is a low-level library developed by Intel Corporation in order to give to programmers a set of signal processing, image processing and matrices optimized functions. All the IPP functions take advantage of Intel processors capabilities such as MMX, SSE and SSE2 functions.

Compared to OpenCV, IPP functions are faster but they only work on Intel processors. IPL (Intel Image Processing Library) was the ancestor of IPP. In IPP, the basic structure IplImage is no longer used because in a optimized application, the reading and writing of IplImage fields is a time consuming task.

The basic structure of IPP is the "ipp image" which is just a pointer to a specific kind of data. This pointer is only an address in the memory and you need to manage its size and properties outside the structure. This choice make the programmer more responsible for the management of his (or her) images. But in the other hand, internal IPP functions have only to manage a pointer which implies important time saving necessary to optimized applications.

There are three parts in IPP library:

- signal processing functions,
- image processing functions,
- small matrix operations.

Each function starts with the prefix "ipp" followed by the part of the IPP used i.e. "ipps" for signal processing function, "ippi" for image processing function and "ippm" for matrix functions. The name of the

functions follows and after it we find the kind of data on which the function will occur. The different types of data used in the library are shown in table "Data types" below. At last we find several descriptors depending on the type of the function that are shown in table "Descriptors" below.

| | |
|---|---|
| Ipp8u | unsigned byte (8 bits) |
| Ipp8s | signed byte (8 bits) |
| Ipp16u | unsigned word (16 bits) |
| Ipp16s | signed word (16 bits) |
| Ipp16sc | signed complex word (16 bits) |
| Ipp32u | unsigned double word (32 bits) |
| Ipp32s | signed double word (32 bits) |
| Ipp32sc | signed complex double word (32 bits) |
| Ipp32f | signed double word floating point number (32 bits) |
| Ipp32fc | signed complex double word floating point number (32 bits) |
| Ipp64s | signed quad word (64 bits) |
| Ipp64f | signed quad word floating point number (64 bits) |

Data types

| | |
|---|---|
| A | data contains an alpha channel |
| Cn | data is made up of n discrete interleaved channels (1, 2, 3, 4) |
| C | channel of interest (COI) is used in the operation |
| D2 | image is two dimensional |
| I | operation is in-place |
| M | uses mask roi for source and destination image |
| Pn | data is made up of n discrete planar (non-interleaved) channels, with a separate pointer to each plane |
| R | uses region of interest (ROI) |
| Sfs | Saturation and fixed scaling mode is used |

Descriptors

For instance the function "ippiColorToGray_32f_C1R" means the function ColorToGray from image processing ("ippi") that operates on 32f (four bytes floating point numbers) on a single channel image ("C1") and uses region of interest (ROI) ("R").

**Do not be afraid of IPP functions !** Their names seem pretty unreadable the first time you see them. But after a while using these functions, you will find it very simple and very clear, because for each function you have the type of data involved, the part of the library used, the type of the result and the operation performed.

There are many functions working either with integer and real datatypes. But there are some functions that work only with one of

these types. For instance, wavelets transform functions only work with real (”Ipp32f”) values. So before using a function, you will need to know how it works. All the functions available under IPP are described in the IPP reference manual[2].

As I already mentioned, IPP does not have a graphic interface, this is why it is important to use OpenCV (highgui) to visualize results. All examples in this chapter will use highgui to read, display and write images. As the structure IplImage is no longer used in IPP, you must be very careful linking IPP and OpenCV structures and pointers. Many errors can occur during this task.

An ipp image is just a pointer to a table of data. Figure 4.1 shows really what is an IPP image composed of only one channel (gray image). Figure 4.2 shows an IPP image in the case of a three channels (color) image. In many IPP functions, you will be ask the pointer to the data (”ptr” in our figures) but also the size (width and height) and the bytestep of the your image. The datatype of the IppiSize structure is given below.

```
typedef struct {
    int width;
    int height;
} IppiSize;
```

In figure 4.1, for instance, the pointer to the image data is ptr. The size of the image is width=9 and height=7. The bytestep is the number of bytes that are necessary to go from one pixel ([row,col]) to the same pixel of the next line ([row+1,col]) in your image. And as the data is one channel (gray level image), the bytestep is ”width*sizeof(data)”. Because you need to go forward of ”width*sizeof(data)” bytes in order to retrieve the pixels at the same row on the next line of your image. ”sizeof(data)” depends of course on the datatype of your image. If it is a ”Ipp8u” image, the data length is one byte. So ”sizeof(data)” will be ”sizeof(Ipp8u)” that is one, so bytestep will be ”width*1” so ”width” (=9 in our example). If your prefer working with Ipp32f 32 bits floating points data (it depends on your application), your bytestep will be ”width*sizeof(Ipp32f)” that is ”width*4” (=36 in our example).

For three channels (color) images, it is the same except that one pixel needs three cells (three channels, [R,G,B] in our example) instead of one (figure 4.2). The size of the image is now width=3 and height=7. The bytestep will be ”width*3*sizeof(data)” because of the three channels. So if you choose ”Ipp8u” data, bytestep will be ”width*3*1” (=9 in our example). And it will be ”width*3*sizeof(Ipp32f)” for ”Ipp32f” image that is ”width*3*4” (=36 in our example).

So please be careful about the size and especially the bytestep of your images. It can be source of multiple errors very difficult to detect at runtime because your code will compile but due to a bad bytestep, your program will crash or will not do exactly what you want.

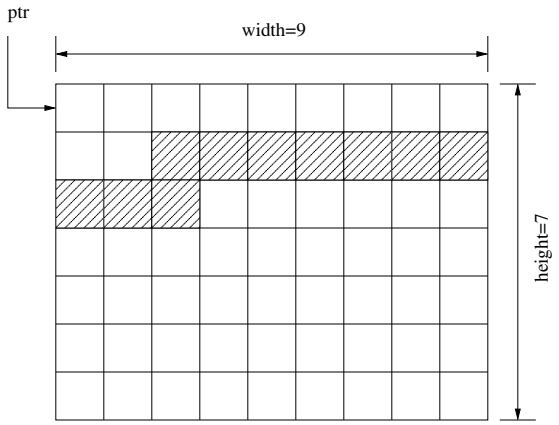


Figure 4.1: IPP image for a gray-level (one channel) image. The bytestep is represented by hatched pixels.

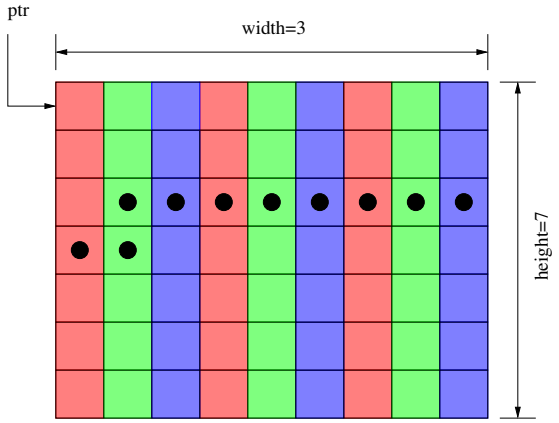


Figure 4.2: IPP image for a color (three channels) image. Be careful, as a pixel needs three values (one per channel), the width is 3 (and not 9). The bytestep must be multiplied by 3 to work ! Here the bytestep is represented by black circles.

## 4.2 A small example to begin with

Our first example of IPP is the creation of an image. But we will need OpenCV to display the result. The listing given below is our first program using IPP so it is called "ipp0.c" in our directory "ippopencv".

```
//
// ipp0.c - creating an image using IPP and displaying it using OpenCV highgui
//

#include "cv.h"
#include "highgui.h"
#include "ipp.h"
#include <stdio.h>

// file example.bmp in our images directory
char name[] = "images/example.bmp";

int main()
{
Ipp8u *gray = NULL; // we define an Ipp8u image pointer gray
IppiSize size; // size of our image

IplImage* img = NULL; // new IplImage structure img
CvSize sizeImg;

int i = 0, j = 0;

size.width = 640;
size.height = 480;
gray = (Ipp8u *) ippsMalloc_8u( size.width * size.height );

for ( i = 0; i < size.height; i++ )
{
for ( j = 0; j < size.width; j++ )
{
*( gray + i * size.width + j ) =
(Ipp8u) abs( 255 * cos( (Ipp32f) ( i * j ) ) );
}
}

sizeImg.width = size.width;
sizeImg.height = size.height;
img = cvCreateImage( sizeImg, 8, 1 );
        cvSetImageData( img, gray, sizeImg.width );

// create window with title "image"
cvNamedWindow( "image", 0 );

// display img in window "image"
cvShowImage( "image", img );

// wait for key to close the window
cvWaitKey(0);

// destroy open window
cvDestroyWindow( "image" );

// memory release with IPP
ippsFree( gray );
// memory release with OpenCV
img->imageData = NULL;
cvReleaseImage( &img );

return( 0 );
}
```

Because the compilation requires both IPP and OpenCV, we must modify our Makefile in order to take in account the new libraries. So please edit our Makefile and add the correct lines. Be careful because the lines are very long, do not do any mistake or it will not work properly. Our new Makefile is made of two parts, one for OpenCV and one for IPP.

We link OpenCV and IPP with all our examples. It will work even if OpenCV examples do not need IPP. In this case, IPP libraries are simply not linked with our application.

---

```
[jeje@rebec ippopencv]$ more Makefile
CC      = gcc -O2 #-march=i686 -mcpu=i686
INC     = -I/home/intel/opencv/include/opencv
 -I/home/intel/ipp/include # include path
LIB     = -L/home/intel/opencv/lib -L/home/intel/ipp/sharedlib # libs path
OPT     = -lopencv -lhighgui -lcvaux -lcvcam -lippcore -lippac -lippi -lippj
-lippmp -lippm -lippsc -lippsr -lipps -lippvc -lippvm -lippcv -lstdc++
-lpthread # standard libraries needed to link (on the same line in Makefile)

BIN     = opencv0 opencv1 opencv2 opencv3 ipp0 # executables to create

all:    $(BIN) # make all creates opencv0, ..., ipp0, ...

# opencv examples

opencv0: opencv0.c Makefile
                $(CC) opencv0.c $(INC) $(LIB) $(OPT) -o $@
opencv1: opencv1.c Makefile
                $(CC) opencv1.c $(INC) $(LIB) $(OPT) -o $@
opencv2: opencv2.c Makefile
                $(CC) opencv2.c $(INC) $(LIB) $(OPT) -o $@
opencv3: opencv3.c Makefile
                $(CC) opencv3.c $(INC) $(LIB) $(OPT) -o $@

# ipp examples

ipp0: ipp0.c Makefile
                $(CC) ipp0.c $(INC) $(LIB) $(OPT) -o $@

# "make clean" cleans the directory
clean:
                rm *~ *.o $(BIN) core

[jeje@rebec ippopencv]$ make
gcc -O2  opencv0.c -I/home/intel/opencv/include/opencv -I/home/intel/ipp/include
-L/home/intel/opencv/lib -L/home/intel/ipp/sharedlib
-lopencv -lhighgui -lcvaux -lcvcam -lippcore -lippac -lippi -lippj -lippmp -lippm
 -lippsc -lippsr -lipps -lippvc -lippvm -lippcv -lstdc++ -lpthread  -o opencv0
gcc -O2  opencv1.c -I/home/intel/opencv/include/opencv -I/home/intel/ipp/include
-L/home/intel/opencv/lib -L/home/intel/ipp/sharedlib
-lopencv -lhighgui -lcvaux -lcvcam -lippcore -lippac -lippi -lippj -lippmp -lippm
 -lippsc -lippsr -lipps -lippvc -lippvm -lippcv -lstdc++ -lpthread  -o opencv1
gcc -O2  opencv2.c -I/home/intel/opencv/include/opencv -I/home/intel/ipp/include
-L/home/intel/opencv/lib -L/home/intel/ipp/sharedlib
-lopencv -lhighgui -lcvaux -lcvcam -lippcore -lippac -lippi -lippj -lippmp -lippm
 -lippsc -lippsr -lipps -lippvc -lippvm -lippcv -lstdc++ -lpthread  -o opencv2
gcc -O2  opencv3.c -I/home/intel/opencv/include/opencv -I/home/intel/ipp/include
-L/home/intel/opencv/lib -L/home/intel/ipp/sharedlib
-lopencv -lhighgui -lcvaux -lcvcam -lippcore -lippac -lippi -lippj -lippmp -lippm
 -lippsc -lippsr -lipps -lippvc -lippvm -lippcv -lstdc++ -lpthread  -o opencv3
gcc -O2  ipp0.c -I/home/intel/opencv/include/opencv -I/home/intel/ipp/include
-L/home/intel/opencv/lib -L/home/intel/ipp/sharedlib
-lopencv -lhighgui -lcvaux -lcvcam -lippcore -lippac -lippi -lippj -lippmp -lippm
 -lippsc -lippsr -lipps -lippvc -lippvm -lippcv -lstdc++ -lpthread  -o ipp0

[jeje@rebec ippopencv]$ ./ipp0
```

```
--- our image must appear on the screen ---
--- press a key to exit ---

[jeje@rebec ippopencv]$ ll
total 268
drwxrwxr-x    2 jeje     jeje         4096 jui 24 15:08 images
-rwxrwxr-x    1 jeje     jeje        47680 jui 24 15:32 ipp0
-rw-r--r--    1 jeje     jeje         1277 jui 24 13:35 ipp0.c
-rw-------    1 jeje     jeje         1035 jui 24 15:32 Makefile
-rwxrwxr-x    1 jeje     jeje        47510 jui 24 15:32 opencv0
-rw-rw-r--    1 jeje     jeje         1193 jui 24 14:54 opencv0.c
-rwxrwxr-x    1 jeje     jeje        47552 jui 24 15:32 opencv1
-rw-------    1 jeje     jeje         1168 jui 24 14:48 opencv1.c
-rwxrwxr-x    1 jeje     jeje        47651 jui 24 15:32 opencv2
-rw-------    1 jeje     jeje         1648 jui 23 18:51 opencv2.c
-rwxrwxr-x    1 jeje     jeje        47578 jui 24 15:32 opencv3
-rw-------    1 jeje     jeje         1156 jui 23 18:45 opencv3.c

[jeje@rebec ippopencv]$
```

So your first IPP program should show you a beautiful image like shown in figure 4.3.

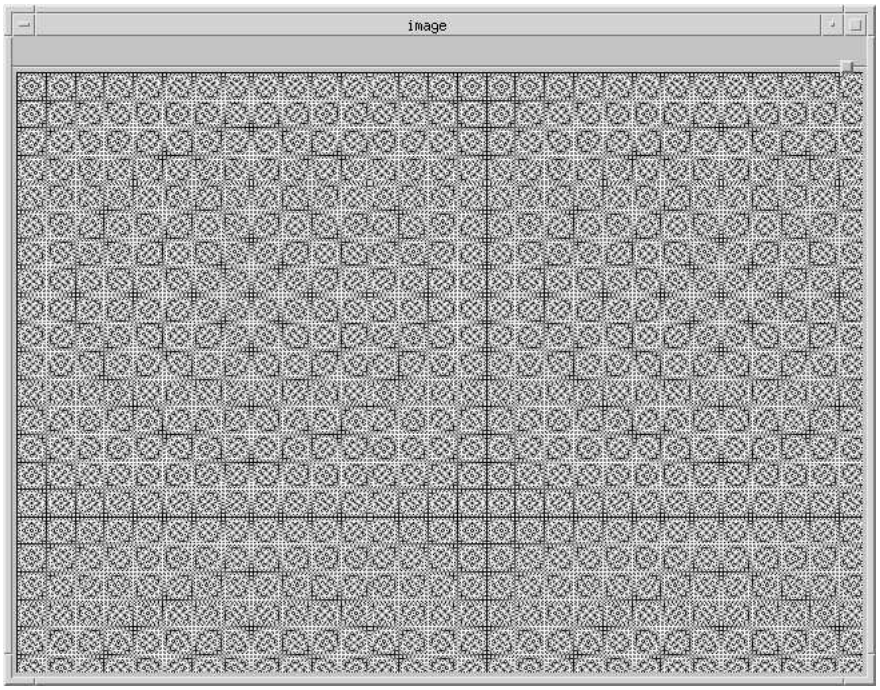


Figure 4.3: Creating an IPP image and displaying it with OpenCV. The result is very strange, isn't it ?

## 4.3 Color conversion under IPP

Let's try several color conversion functions under IPP. File "ipp1.c" must be in our directory "ippopencv".

```
//
// ipp1.c - loading, color conversion and display of an image using IPP and Opencv
//

#include "cv.h"
#include "highgui.h"
#include "ipp.h"
#include <stdio.h>

// file example.bmp in our images directory
char name[] = "images/example.bmp";

// functions defined after main()
void openWindow_8u( Ipp8u *img, IppiSize *size, int nChannels,
    char *name, int wait );
void closeWindow_8u( char *name );

int main()
{
// define Ipp8u image pointer rgb
Ipp8u *ipprgb = NULL;
// size of our images
IppiSize size;
// pointers to our result images
Ipp8u *ippxyz = NULL, *ipphls = NULL, *ippluv = NULL;

IplImage* img = NULL; // new IplImage structure img

int i = 0, j = 0;

img = cvLoadImage( name, -1 ); // load a BMP ipprgb image into img

size.width = img->width; // size of IPP images is the same
size.height = img->height; // as read image img

// memory allocation for rgb
ipprgb = ippsMalloc_8u( size.width * size.height * 3 );

// the pointer of our OpenCV data is our IPP image pointer
ippiCopy_8u_C3R( img->imageData, size.width * 3, ipprgb, size.width * 3, size );

// memory allocation for xyz
ippxyz = (Ipp8u *) ippsMalloc_8u( size.width * size.height * 3 );
// memory allocation for hls
ipphls = (Ipp8u *) ippsMalloc_8u( size.width * size.height * 3 );
// memory allocation for luv
ippluv = (Ipp8u *) ippsMalloc_8u( size.width * size.height * 3 );

// rgb2xyz
ippiRGBToXYZ_8u_C3R( ipprgb, size.width * 3, ippxyz, size.width * 3, size );
// rgb2hls
ippiRGBToHLS_8u_C3R( ipprgb, size.width * 3, ipphls, size.width * 3, size );
// rgb2luv
ippiRGBToLUV_8u_C3R( ipprgb, size.width * 3, ippluv, size.width * 3, size );

openWindow_8u( ipprgb, &size, 3, "rgb", 0 ); // display rgb image
closeWindow_8u( "rgb" ); // close rgb image

openWindow_8u( ippxyz, &size, 3, "xyz", 0 ); // display xyz image
closeWindow_8u( "xyz" ); // close xyz image

openWindow_8u( ipphls, &size, 3, "hls", 0 ); // display hls image
closeWindow_8u( "hls" ); // close hls image

openWindow_8u( ippluv, &size, 3, "luv", 0 ); // display luv image
closeWindow_8u( "luv" ); // close luv image

ippsFree( ipprgb ); // memory release with IPP
ippsFree( ippxyz ); // memory release with IPP
ippsFree( ipphls ); // memory release with IPP
ippsFree( ippluv ); // memory release with IPP
cvReleaseImage( &img ); // memory release with OpenCV
```

```
return( 0 );
}

// openWindow_8u used to display IPP images with OpenCV highgui

void openWindow_8u( Ipp8u *img, IppiSize *size, int nChannels,
    char *name, int wait )
{
IplImage *cvImg; // cv image used to display
CvSize sizeCv; // cv size structure

Ipp8u *tmp; // IPP temporary image

sizeCv.width = size->width; // define size
sizeCv.height = size->height; // of th displayed image

// create a cv image to be displayed
cvImg = cvCreateImage( sizeCv, IPL_DEPTH_8U, nChannels );

// create temporary image so that img is unchanged
tmp = ippsMalloc_8u( size->width * size->height * nChannels );

// copy img into tmp
ippiCopy_8u_C3R( img, size->width * nChannels,
 tmp, size->width * nChannels, *size );

// set cv image to contain tmp
cvSetData( cvImg, (void *) tmp, sizeCv.width * nChannels );

cvNamedWindow( name, 1 ); // create window
cvShowImage( name, cvImg ); // displays cv image

cvWaitKey( wait ); // wait for key (==0) or wait milliseconds
cvReleaseImage( &cvImg ); // free memory cv image and also tmp
}

// closeWindow_8u used to close a previously opened window

void closeWindow_8u( char *name )
{
cvDestroyWindow( name ); // close cv window and free memory
}
```

---

Because we will need to display results of IPP functions, we have created two functions. "openWindow_8u" that opens a window and displays an IPP image and "closeWindow_8u" that closes a window. These functions will be helpful in all further examples. Figure 4.4 shows the result of our program.

Several changes are necessary in our "Makefile". First of all, add a "ipp1" entry in the "BIN" line. Then create an "ipp1" entry in the "ipp examples" list. When done, you have to compile whith "make" and to launch with "./ipp1".

---

```
[jeje@rebec ippopencv]$ vi Makefile

[jeje@rebec ippopencv]$ more Makefile
CC      = gcc -O2 #-march=i686 -mcpu=i686
INC     = -I/home/intel/opencv/include/opencv
 -I/home/intel/ipp/include # include path
LIB     = -L/home/intel/opencv/lib -L/home/intel/ipp/sharedlib # libs path
OPT     = -lopencv -lhighgui -lcvaux -lcvcam -lippcore -lippac -lippi -lippj
-lippmp -lippm -lippsc -lippsr -lipps -lippvc -lippvm -lippcv -lstdc++
-lpthread # standard libraries needed to link (on the same line in Makefile)
```

```
BIN     = opencv0 opencv1 opencv2 opencv3 ipp0 ipp1 # executables to create

all:    $(BIN) # make all creates opencv0, ..., ipp0, ...

# opencv examples

opencv0: opencv0.c Makefile
                $(CC) opencv0.c $(INC) $(LIB) $(OPT) -o $@
opencv1: opencv1.c Makefile
                $(CC) opencv1.c $(INC) $(LIB) $(OPT) -o $@
opencv2: opencv2.c Makefile
                $(CC) opencv2.c $(INC) $(LIB) $(OPT) -o $@
opencv3: opencv3.c Makefile
                $(CC) opencv3.c $(INC) $(LIB) $(OPT) -o $@

# ipp examples

ipp0: ipp0.c Makefile
                $(CC) ipp0.c $(INC) $(LIB) $(OPT) -o $@

ipp1: ipp1.c Makefile
                $(CC) ipp1.c $(INC) $(LIB) $(OPT) -o $@

# "make clean" cleans the directory
clean:
                rm *~ *.o $(BIN) core
```

```
[jeje@rebec ippopencv]$ make
gcc -O2  opencv0.c -I/home/intel/opencv/include/opencv -I/home/intel/ipp/include
-L/home/intel/opencv/lib -L/home/intel/ipp/sharedlib
-lopencv -lhighgui -lcvaux -lcvcam -lippcore -lippac -lippi -lippj -lippmp -lippm
 -lippsc -lippsr -lipps -lippvc -lippvm -lippcv -lstdc++ -lpthread  -o opencv0
gcc -O2  opencv1.c -I/home/intel/opencv/include/opencv -I/home/intel/ipp/include
-L/home/intel/opencv/lib -L/home/intel/ipp/sharedlib
-lopencv -lhighgui -lcvaux -lcvcam -lippcore -lippac -lippi -lippj -lippmp -lippm
 -lippsc -lippsr -lipps -lippvc -lippvm -lippcv -lstdc++ -lpthread  -o opencv1
gcc -O2  opencv2.c -I/home/intel/opencv/include/opencv -I/home/intel/ipp/include
-L/home/intel/opencv/lib -L/home/intel/ipp/sharedlib
-lopencv -lhighgui -lcvaux -lcvcam -lippcore -lippac -lippi -lippj -lippmp -lippm
 -lippsc -lippsr -lipps -lippvc -lippvm -lippcv -lstdc++ -lpthread  -o opencv2
gcc -O2  opencv3.c -I/home/intel/opencv/include/opencv -I/home/intel/ipp/include
-L/home/intel/opencv/lib -L/home/intel/ipp/sharedlib
-lopencv -lhighgui -lcvaux -lcvcam -lippcore -lippac -lippi -lippj -lippmp -lippm
 -lippsc -lippsr -lipps -lippvc -lippvm -lippcv -lstdc++ -lpthread  -o opencv3
gcc -O2  ipp0.c -I/home/intel/opencv/include/opencv -I/home/intel/ipp/include
-L/home/intel/opencv/lib -L/home/intel/ipp/sharedlib
-lopencv -lhighgui -lcvaux -lcvcam -lippcore -lippac -lippi -lippj -lippmp -lippm
 -lippsc -lippsr -lipps -lippvc -lippvm -lippcv -lstdc++ -lpthread  -o ipp0
gcc -O2  ipp1.c -I/home/intel/opencv/include/opencv -I/home/intel/ipp/include
-L/home/intel/opencv/lib -L/home/intel/ipp/sharedlib
-lopencv -lhighgui -lcvaux -lcvcam -lippcore -lippac -lippi -lippj -lippmp -lippm
 -lippsc -lippsr -lipps -lippvc -lippvm -lippcv -lstdc++ -lpthread  -o ipp1
[jeje@rebec ippopencv]$ ./ipp1

--- our image must appear on the screen ---
--- press a key to exit ---

[jeje@rebec ippopencv]$
```

## 4.4 Filtering and saving an image

Let's try filtering functions under IPP. File "ipp2.c" must be in our directory "ippopencv". This example will load an image, transform it to floating point data, filter it with Laplace edge detection kernel, convert it back to integer data and save it on the disk.

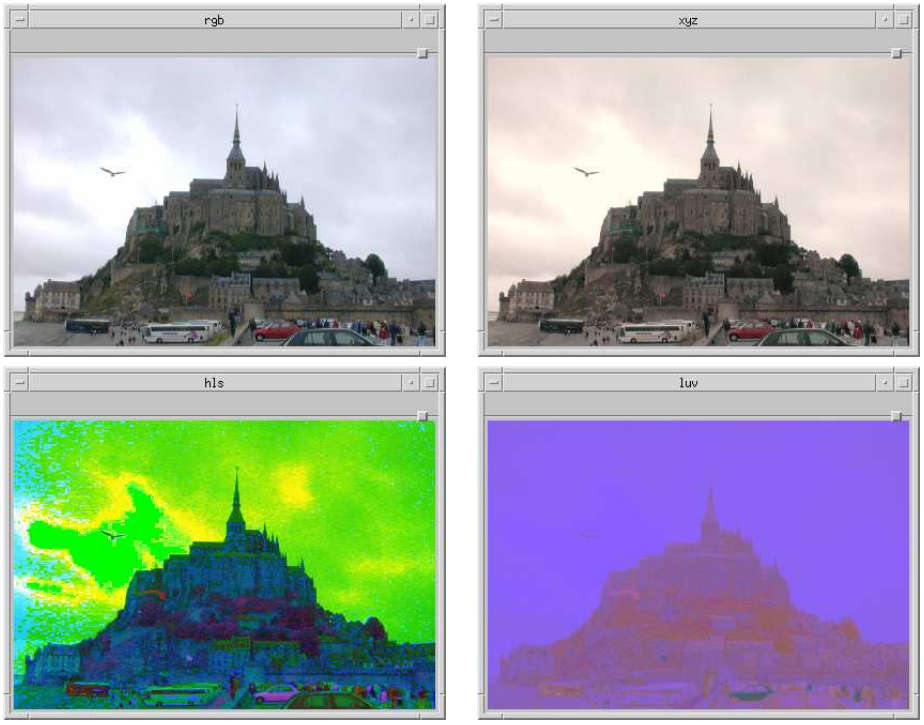


Figure 4.4: Color conversions with IPP displayed by OpenCV highgui.

```
//
// ipp2.c - load, convert to real, filter, convert to integer and save an image
//

#include "cv.h"
#include "highgui.h"
#include "ipp.h"
#include <stdio.h>

// functions defined after main
void openWindow_8u( Ipp8u *img, IppiSize *size, int nChannels,
    char *name, int wait );
void closeWindow_8u( char *name );
void saveImage_8u( Ipp8u *img, IppiSize *size, int nChannels, char *name );

int main( int argc, char **argv )
{
char name[ 255 ]; // file names
Ipp8u *img8u = NULL; // we define an Ipp8u image pointer
Ipp8u *laplace8u = NULL; // we define our laplace8u image pointer
IppiSize size; // size of our image
Ipp32f *img32f = NULL; // pointer to our Ipp32f real data image
Ipp32f *laplace32f = NULL; // pointer to our Ipp32f real data image
Ipp32f min[ 3 ], max[ 3 ]; // min and max of our Ipp32f image used to scale
IplImage* cvImg = NULL; // new IplImage structure img

if ( argc != 2 )
{
printf( "Sorry, use this function with an image filename\n" );
printf( "For instance, \"$ ./ipp2 images/example.bmp\"\n" );
exit( -1 );
}

strcpy( name, argv[ 1 ] );

cvImg = cvLoadImage( name, -1 ); // load an image into cvImg

size.width = cvImg->width; // size of IPP images is the same
size.height = cvImg->height; // as read image img

// memory allocation for img8u
img8u = ippsMalloc_8u( size.width * size.height * 3 );
// memory allocation for laplace8u
laplace8u = ippsMalloc_8u( size.width * size.height * 3 );

// copy data from OpeCV IplImage to IPP Ipp8u image
ippiCopy_8u_C3R( cvImg->imageData, size.width * 3, img8u, size.width * 3, size );

// --- first part: Laplace filter on our Ipp8u image

// Laplace filter 8u
ippiFilterLaplace_8u_C3R( img8u, size.width * 3, laplace8u, size.width * 3, size,
 ippMskSize3x3 );

openWindow_8u( laplace8u, &size, 3, "laplace_8u", 0 ); // display laplace8u image
closeWindow_8u( "laplace_8u" ); // close laplace8u image

strcpy( name, "images/laplace_8u.jpg" );
saveImage_8u( laplace8u, &size, 3, name ); // save image

// --- second part: Laplace filter on our Ipp32f image

// memory allocation for img32f
img32f = ippsMalloc_32f( size.width * size.height * 3 );
// memory allocation for laplace32f
laplace32f = ippsMalloc_32f( size.width * size.height * 3 );

// conversion
ippiConvert_8u32f_C3R( img8u, size.width * 3, img32f, size.width * 3 * 4, size );

// Laplace filter 32f
ippiFilterLaplace_32f_C3R( img32f, size.width * 3 * 4,
laplace32f, size.width * 3 * 4, size, ippMskSize3x3 );
```

```
ippiMinMax_32f_C3R( laplace32f, size.width * 3 * 4,
    size, &( min[ 0 ] ), &( max[ 0 ] ) );
ippiScale_32f8u_C3R( laplace32f, size.width * 3 * 4, img8u, size.width * 3,
size, min[ 0 ], max[ 0 ] );

openWindow_8u( img8u, &size, 3, "img8u", 0 ); // display img8u image
closeWindow_8u( "img8u" ); // close img8u image

strcpy( name, "images/laplace_32f.jpg" );
saveImage_8u( img8u, &size, 3, name ); // save image

ippsFree( laplace8u ); // memory release with IPP
ippsFree( img8u ); // memory release with IPP
ippsFree( img32f ); // memory release with IPP
cvReleaseImage( &cvImg ); // memory release with OpenCV

return( 0 );
}

// openWindow_8u used to display IPP images with OpenCV highgui

void openWindow_8u( Ipp8u *img, IppiSize *size, int nChannels,
    char *name, int wait )
{
IplImage *cvImg; // cv image used to display
CvSize sizeCv; // cv size structure

Ipp8u *tmp; // IPP temporary image

sizeCv.width = size->width; // define size
sizeCv.height = size->height; // of th displayed image

// create a cv image to be displayed
cvImg = cvCreateImage( sizeCv, IPL_DEPTH_8U, nChannels );

// create temporary image so that img is unchanged
tmp = ippsMalloc_8u( size->width * size->height * nChannels );

// copy img into tmp
ippiCopy_8u_C3R( img, size->width * nChannels,
 tmp, size->width * nChannels, *size );

// set cv image to contain tmp
cvSetData( cvImg, (void *) tmp, sizeCv.width * nChannels );

cvNamedWindow( name, 1 ); // create a window
cvShowImage( name, cvImg ); // displays cv image

cvWaitKey( wait ); // wait for a key (==0) or wait milliseconds
cvReleaseImage( &cvImg ); // free memory cv image and also tmp
}

// closeWindow_8u used to close a previously opened window

void closeWindow_8u( char *name )
{
cvDestroyWindow( name ); // close cv window and free memory
}

// saveImage_8u used to save an IPP image

void saveImage_8u( Ipp8u *img, IppiSize *size, int nChannels, char *name )
{
IplImage *cvImg; // cv image used to save
CvSize sizeCv; // cv size structure

Ipp8u *tmp; // IPP temporary image

sizeCv.width = size->width; // define size
sizeCv.height = size->height; // of th displayed image

// create a cv image to be displayed
cvImg = cvCreateImage( sizeCv, IPL_DEPTH_8U, nChannels );
```

```
// create temporary image so that img is unchanged
tmp = ippsMalloc_8u( size->width * size->height * nChannels );

// copy img into tmp
ippiCopy_8u_C3R( img, size->width * nChannels,
 tmp, size->width * nChannels, *size );

// set cv image to contain tmp
cvSetData( cvImg, (void *) tmp, sizeCv.width * nChannels );

cvSaveImage( name, cvImg ); // save image to disk

cvReleaseImage( &cvImg ); // free memory cv image and also tmp
}
```

---

Our "Makefile" has to be changed to compile "ipp2".

---

```
[jeje@rebec ippopencv]$ vi Makefile

[jeje@rebec ippopencv]$ ll
total 324
drwxrwxr-x    2 jeje     jeje         4096 jui 24 16:30 images
-rwxrwxr-x    1 jeje     jeje        47680 jui 24 16:31 ipp0
-rw-r--r--    1 jeje     jeje         1277 jui 24 13:35 ipp0.c
-rwxrwxr-x    1 jeje     jeje        48310 jui 24 22:07 ipp1
-rw-r--r--    1 jeje     jeje         3530 jui 24 22:07 ipp1.c
-rw-r--r--    1 jeje     jeje         3547 jui 25 15:22 ipp2.c
-rw-------    1 jeje     jeje         1187 jui 25 15:23 Makefile
-rwxrwxr-x    1 jeje     jeje        47510 jui 24 16:31 opencv0
-rw-rw-r--    1 jeje     jeje         1193 jui 24 14:54 opencv0.c
-rwxrwxr-x    1 jeje     jeje        47552 jui 24 16:31 opencv1
-rw-------    1 jeje     jeje         1168 jui 24 14:48 opencv1.c
-rwxrwxr-x    1 jeje     jeje        47651 jui 24 16:31 opencv2
-rw-------    1 jeje     jeje         1648 jui 23 18:51 opencv2.c
-rwxrwxr-x    1 jeje     jeje        47578 jui 24 16:31 opencv3
-rw-------    1 jeje     jeje         1156 jui 23 18:45 opencv3.c

[jeje@rebec ippopencv]$ more Makefile
CC      = gcc -O2 #-march=i686 -mcpu=i686
INC     = -I/home/intel/opencv/include/opencv
 -I/home/intel/ipp/include # include path
LIB     = -L/home/intel/opencv/lib -L/home/intel/ipp/sharedlib # libs path
OPT     = -lopencv -lhighgui -lcvaux -lcvcam -lippcore -lippac -lippi -lippj
 -lippmp -lippm -lippsc -lippsr -lipps -lippvc -lippvm -lippcv -lstdc++
 -lpthread # standard libraries needed to link

BIN     = opencv0 opencv1 opencv2 opencv3 ipp0 ipp1 ipp2 # executables to create

all:    $(BIN) # make all creates opencv0, ..., ipp0, ...

# opencv examples

opencv0: opencv0.c Makefile
                $(CC) opencv0.c $(INC) $(LIB) $(OPT) -o $@
opencv1: opencv1.c Makefile
                $(CC) opencv1.c $(INC) $(LIB) $(OPT) -o $@
opencv2: opencv2.c Makefile
                $(CC) opencv2.c $(INC) $(LIB) $(OPT) -o $@
opencv3: opencv3.c Makefile
                $(CC) opencv3.c $(INC) $(LIB) $(OPT) -o $@

# ipp examples

ipp0: ipp0.c Makefile
                $(CC) ipp0.c $(INC) $(LIB) $(OPT) -o $@
ipp1: ipp1.c Makefile
                $(CC) ipp1.c $(INC) $(LIB) $(OPT) -o $@
ipp2: ipp2.c Makefile
                $(CC) ipp2.c $(INC) $(LIB) $(OPT) -o $@
```

```
# "make clean" cleans the directory
clean:
                rm *~ *.o $(BIN) core

[jeje@rebec ippopencv]$ ./ipp2 images/example.bmp

[jeje@rebec ippopencv]$ ll images
total 412
-r-xr-xr-x    1 jeje     jeje       360054 jui 25 19:08 example.bmp
-r-xr-xr-x    1 jeje     jeje        14848 jui 25 19:08 example.jpg
-rw-rw-r--    1 jeje     jeje        13683 jui 25 19:09 laplace_32f.jpg
-rw-rw-r--    1 jeje     jeje        20896 jui 25 19:09 laplace_8u.jpg

[jeje@rebec ippopencv]$
```

---

This example loads an image with OpenCV "cvImg" and copies it to an IPP image ("img8u"). This image is filtered to find its edges using a Laplacian kernel ("laplace8u") which is displayed. It is saved as "images/laplace_8u.jpg". Then it converts our image from "Ipp8u" to "Ipp32f" ("img32f"). The Laplacian kernel is then applied to our "Ipp32f" image giving an "Ipp32f" image ("laplace32f"). This image is converted back to an Ipp8u image by scaling its values between a min and a max value (per channel). Then it is displayed. At last, it is saved as "images/laplace_32f.jpg". The resulting images are shown in figure 4.5.

For the "Ipp32f" function, the result is not very good because of the information lost in the conversion (scaling) process. This example illustrates the difference between "Ipp8u" and "Ipp32f" images especially for the bytestep value.

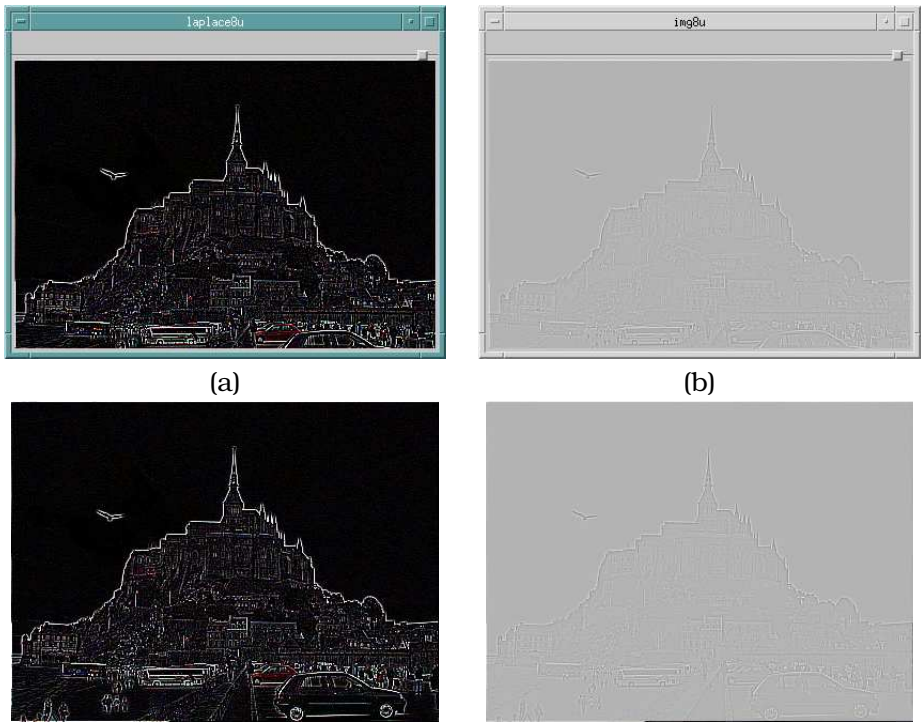


Figure 4.5: Edge detection by Laplacian kernel under IPP. (a) and (b) are the results on the screen. (c) and (d) are the files written on the disk.

# Chapter 5

# Conclusion

*"Pour être grand, il faut avoir été petit."*
*"To be great, you must have been small."*
**La chanson de Guillaume — fin du $XI^{ème}$ ou milieu du $XII^{ème}$ siècle**
**Guillaume's song — end of $XI^{th}$ or middle of $XII^{th}$ century**

Intel IPP library used in combination of OpenCV library is a very good tool for image and video processing. The speed of each operation takes advantage of MMX and SSE, SSE2 instructions (under Intel processors), each operation is very fast.

Low-level functions can be made under IPP and many high-level image processing methods and algorithms are present in OpenCV. So IPP and OpenCV are complementary tools to build efficient and effective image processing and computer vision applications.

This tutorial needs to live so please feel free to send me e-mails to tell me what is good and what is not. During this year, I will finish my Ph. D. thesis so I will probably not have much time to write version 0.5 of this manual, but I will take your corrections in consideration as soon as possible.

I want to apology for any mistake due to my bad american english writing. Please send me corrections.

Thank you for using this manual, I spent a lot of time to write it, reading it is the best way to respect my work...

# Appendix A

# Internet websites

***Intel IPP***

- Intel Corporation website - http://www.intel.com
- Intel Premier support website - http://premier.intel.com

***OpenCV***

- Open Computer Vision (OpenCV) sources, download site - http://sourceforge.net/projects/opencvlibrary
- Open Computer Vision (OpenCV) mailing list and group - http://groups.yahoo.com/group/OpenCV

# Appendix B

# Libraries overview

## B.1 Intel Performance Primitives (IPP)

IPP is a set of low-level functions for signal processing, image processing and small matrices computation. All the functions are optimized for Intel processors (Pentium, Pentium II, Pentium III, Pentium 4, Xeon, Itanium) and use special instructions such as MMX (MultiMedia eXtension), SSE (Streaming Single instruction multiple data Extensions), and SSE2. It is composed of three main parts.

- **Signal processing**
  - Signals generation,
  - Logical and arithmetic operations,
  - Conversions,
  - Statistics,
  - Filtering and convolution,
  - FFT, DFT, DCT, wavelets,
  - speech recognition,
  - Sound coding and decoding (MP3). . .
- **Image processing**
  - Images generation,
  - Logical and arithmetic operations,
  - Colorspaces conversions,
  - Thresholding and comparison,
  - Morphological operations,
  - Filtering,
  - FFT 2D, DFT 2D, DCT 2D, wavelets 2D,
  - Statistics,

  - Geometric tranformations,
  - JPEG and JPEG2000 coding and decoding,
  - H.263 video decoding,
  - MPEG4 coding and decoding. . .

- **Matrix computation**
  - Matrices generation,
  - Logical and arithmetic operation,
  - Matrix product, transposition, trace,
  - Linear equation system solving. . .

## B.2 Open Computer Vision Library (OpenCV)

OpenCV is an open library, developed by Intel Corporation. It offers to programmers a set of high-level functions and algorithms portable on many platforms.

- Basic operations,
- Statistics and moments computing,
- Image processing and analysis,
- Structural analysis,
- Motion analysis and object tracking,
- Pattern recognition,
- 3d reconstruction and camera calibration,
- Graphic interface and acquisition. . .

# Appendix C

# Copyright information

I wrote the first version of this tutorial during my August holidays in 2002. I have not been paid for this work and I don't intend to be. My purpose in only to help the Linux programmers community to use Intel IPP and OpenCV libraries easily. Perhaps Intel Corporation will find my tutorial very good and will propose me to work for them. Why not ? ;-)
This manual is free and no charge is required to use it and redistribute it. The C code is freely usable but is given without any warranty on your hardware or software. You use it at your own risk and the writer of this tutorial cannot be responsible for any problem encountered due to errors in this manual.

- Intel, Pentium, and MMX are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and others countries.
- OpenCV is an open library of Intel Corporation.
- Other names and brands are property of their respective owner.

I you have any remark on this tutorial, please send me an e-mail, my address is given in the front page.

# Bibliography

[1] *Open source computer vision library - reference manual*, INTEL Corporation, 1999-2001.

[2] *Integrated performance primitives for intel architecture - reference manual*, INTEL Corporation, 2000-2001.

[3] INTEL Corporation, *Intel website*, http://www.intel.com.

[4] _____, *Opencv website*, http://sourceforge.net/projects/opencvlibrary.

**HISTORY** (approximative due to the long time between two versions)

The writing of this manual took a long time because I had to prepare my Ph.D. thesis in image database indexing and retrieval using a multiresolution research tree. As my Ph.D. will be finished, I will have more time to go on with this manual.

- Version 0.4 written in July 2003 — First version published in the OpenCV mailing list.
  - IPP 3.0
  - OpenCV 0.9.5
- Version 0.3 written in March 2003.
  - IPP 3.0beta with update
  - OpenCV 0.9.4
- Version 0.2 written in December 2002.
  - IPP 3.0beta
  - OpenCV 0.9.3
- Version 0.1 written in August 2002.
  - IPP 2.0
  - OpenCV 0.9.3

**TODO**

- IPP 4.0beta has been announced, not yet tried.
- Add a specific part on video acquisition functions under OpenCV.
- Add more OpenCV examples (matrices operations, image processing, algorithms. . . ).
- Add more IPP examples (small matrices operations, Fourier transforms, MP3 and MPEG4 decoding. . . ).
- Add biggest part of code used during my Ph.D. thesis:

- Wavelet transform using IPP 3.0
- Principal Components Analysis (or Karhunen-loève transform) using matrix operations under OpenCV.
- 3D visualisation program.