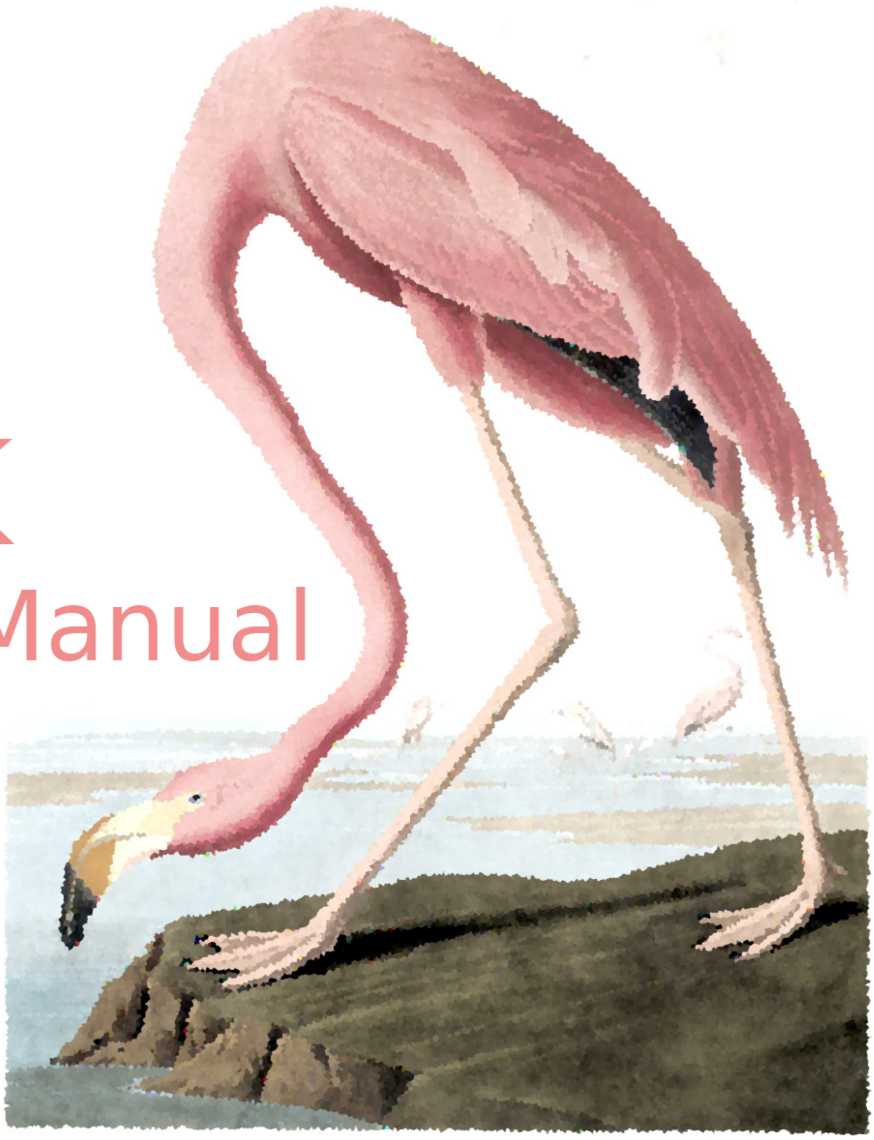
Pink

A User Manual



Michel Couprie, László Marak & Hugues Talbot

Version 0.1 as of April 15, 2013

Documentation is like sex:

when it is good, it is very, very good;

and when it is bad, it is better than nothing.[1].

À nos pauvres étudiants qui ont été forcés de souffrir pendant des heures sur ce logiciel.

[1] Attributed to Dick H. Brandon, pioneer in computer law.

Contents

Chapter 1

Introduction

PINK is an image processing library developed at ESIEE Paris for research and teaching purposes. It contains implementations of over 200 algorithms for image segmentation and filtering. Most of the operators come from mathematical morphology, but it contains operators from different fields. Pink is free software licensed under the CeCILL license.

We are interested in the continuous development of Pink. It has already been proven useful in many applications and we are constantly looking for new ones.

This manual aims at referencing most functions of PINK, hopefully in a didactic manner.

History

Pink is Not Khoros

Related software

These days, image processing is a technology and something such as PINK is best used as a component among others. For optimal use, the following packages should be installed:

- imview
- Python (version 2.7 preferred) ; with the following packages:
  - numpy
  - scipy
  - matplotlib
  - python-vtk
  - python-image (PIL)
  - python-image-tk
- Doxygen
- ActiveTcl 8.3
- VTK
- MPlayer
- Gnuplot

Other software may prove useful:

- OpenCV
- ITK

Contributors

Pink is the result of many thousands of hours of work, and includes contributions from this (non-exhaustive) list of people

Code under the main CeCILL license:

- Michel Couprie : main author, initial design
- László Marak (ujoimro) : library, port to Python, continuous maximum flows, Total-Variation denoising, Python front-end, native Microsoft Windows port.
- Laurent Najman : localextrema, saliency
- Hugues Talbot : fmm, fast morphological operators, region growing; this documentation.
- Jean Cousty : redt 3d (reverse euclidean distance transform - algo de D. Coeurjolly), watershedthin, opérateurs sur les graphes d'arêtes (GA), minimum cost forest (MSF), waterfall, recalagerigide translateplane
- Xavier Daragon: dist, distc (Quadratic Euclidian Distance in 3D)
- André Vital Saude: radialopening, divers scripts tcl, hma
- Nicolas Combaret: toposhrinkgray, ptselectgray
- John Chaussard: lballincl, cropondisk, shrinkondisk
- Christophe Doublier: zoomint
- Hildegard Koehler: lintophat
- Cédric Allène: gettree, histolisse, labeltree, nbcomp, pgm2vtk, seuilauto
- Gu Jun: maxdiameter
- Sébastien Couprie: mcsplines.c
- Rita Zrour: medialaxis (exact Euclidean medial axis - algorithm of Rémy Thiel), dist, distc (Quadratic exact Euclidean distance - algorithm of Saito-Toriwaki, in 2D)
- Laurent Mercier: gestion d'un masque dans delaunay
- Benjamin Raynal: parallel 3D thinning
- Nivando Bezerra: parallel grayscale thinning

Code under different free software licenses:

- David Coeurjolly: lvoronoilabelling.c
- Dario Bressanini: mcpowell.c
- Andrew W. Fitzgibbon: lbresen.c
- Lilian Buzer: lbdigitalline.cxx

Chapter 2

Tutorial

This chapter explains how to start PINK and how to use it.

2.1 Starting PINK and related software

2.2 Examples

We will show a lot of Python code:

2.2.1 Code highlighting

Testing Listings:

```
#!/usr/bin/env python
#

print("Hello World of Pink/Python\n")
```

Testing fancyvrb

Program 1: hello in Python

```
#!/usr/bin/env python
#

print("Hello World of Pink/Python\n")
```

2.2.2 Dynamic content

Some code and its output: The following program produces the plot from Fig. 2.1.

Program 2: A plotting example

```
import numpy as np
import matplotlib.pyplot as plt

fig = plt.figure()
ax1 = fig.add_subplot(111)
t = np.arange(0.01, 10.0, 0.01)
s1 = np.exp(t)
ax1.plot(t, s1, 'b-')
ax1.set_xlabel('time (s)')
ax1.set_ylabel('exp')
```

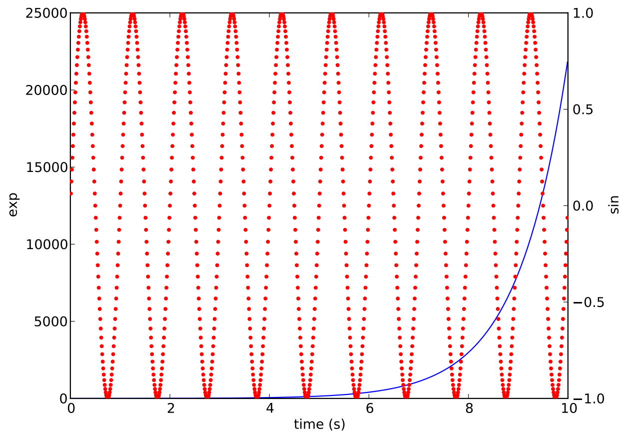


Figure 2.1: The functions sin and exp together from program 2.

```

ax2 = ax1.twinx()
s2 = np.sin(2*np.pi*t)
ax2.plot(t, s2, 'r.')
ax2.set_ylabel('sin')
#plt.show()
```

2.2.3 Image processing

An example of processing?

2.2.4 Binary image processing

2.2.5 Grey-level segmentation

2.2.6 Discrete geometry

2.2.7 3D image processing

2.3 Pink and other image-related software

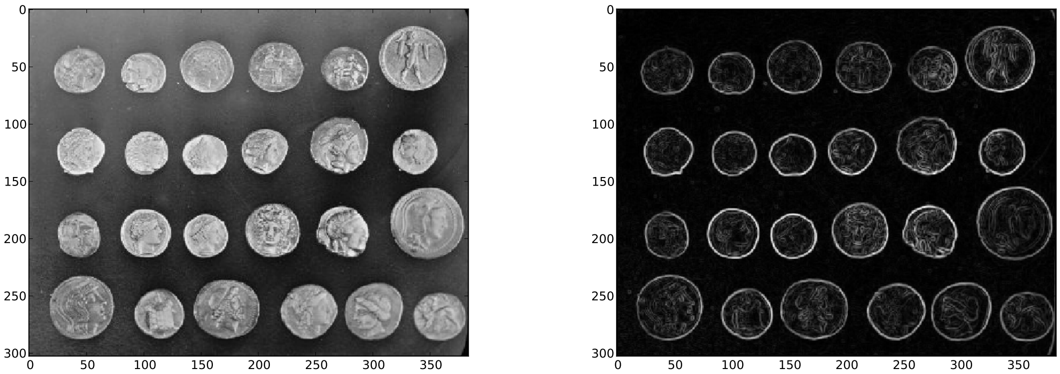
2.3.1 Pink and numpy

2.3.2 Pink and scikit

2.3.3 Pink and VTK

2.3.4 Pink and ITK

2.4 Exercises



(a) (b)

Program 3: An image processing example

```

from skimage import data, filter, io

# not hard !!
def sampleproc(image):
    edges = filter.sobel(image)
    return edges

## saving procedure
def saveimages(path1, path2):
    inputimg = data.coins()
    outputimg = sampleproc(inputimg)
```

Figure 2.2: Program 3 and its result.

Chapter 3

Complementary python tools: numpy, scipy, scikit-image, etc

This chapters describes working with complementary python tools, such as numpy, scipy, the scikits, etc.

In recent years, python has become a credible replacement for Matlab, thanks to a huge community effort in porting basic numerical facilities to python. Other efforts such as those of Enthought, a company that provides a well-designed distribution of python oriented towards parallel numerical tools, have made python a performance tool suitable for many enterprise-level applications.

The aim of this section is to show how these tools complement Pink.

3.1 Converting from/to numpy

The *lingua franca* of numerical computing under python is numpy. Pink provides means to swap back and forth between its internal format and numpy arrays.

Program 4: Numpy conversion

```
#import pink
#import numpy as np

# creation of some 3D data
myarray = np.array(np.arange(0,5*5*5)).reshape(5,5,5)
myarray
# convert array -> pink
myimage = pink.numpy2pink(myarray)
# convert pink -> array
newarray = pink.pink2numpy(myimage)
newarray
# variance calculation
newarray.var(axis=0)
```

The plan for future versions of Pink is to make this conversion automatic and transparent, so that for the Python user, Pink image data is always a numpy array.

3.2 Basic numpy usage

Appendix A

Compiling and installing PINK

While not as difficult as it may seem, obtaining, compiling and installing PINK is currently an imperfect process.

We recommend you try to get a binary version from `http://pinkhq.com`. If you cannot find what you want there, then by all means get the Git version:

```
hg clone https://www.pinkhq.com/hg/pink
```

You may need to work around your local firewall, but this particular arcane branch of knowledge is outside the scope of this document.

A.1 Git branches

There may be several branches of interest to you. Please read the mercurial `hg` documentation to get their names and revision history. Various GUI tools exists that may make this task easier. Typically the main branch is the one that is known to compile and work on the platform we do use. However several development branches are currently active. If you are involved in PINK development, you may know which branch is of interest to you. If you feel adventurous, by all means try the branch that is most recent. Once you have made your choice, compiling is the next task.

A.2 Compiling

For compiling PINK, you need the following components

- CMake version

LICENSE

List of Figures

List of Tables