PINK image processing library

Michel Couprie

1 General description

PINK is an image processing and image analysis library developed at ESIEE Paris for research and teaching purposes.

Initially, PINK was a personnal project with the aim of providing a small group of users with a basis to develop image processing code in a simple and straightforward way. It was inspired in some ways by a library called Khoros: operators were coded in standard C, intermediate results were stored in files... The primary goal was to keep the library easy to develop, maintain and debug. At this stage, the idea of having a sophisticated graphical user interface like in Khoros was dropped. By the way, PINK is a recursive acronym in the manner of GNU, meaning “PINK Is Not Khoros”.

Gradually, the number of users/developpers increased and also did the need for interfaces. The library in now used routinely in ESIEE for teaching image processing and in the framework of student projects. It continues to be used as a research tool for developping and experimenting new algorithms and image processing chains, by the members of the LIGM/ESIEE team and some of our academic partners. It is also used outside ESIEE by a small number of industrial partners (CEA, SANOFI, Saint Gobain, EDF), who contributed financially to its development. The development team is mainly composed by Hugues Talbot, Laszlo Marak, Laurent Najman, Jean Cousty and myself.

The library is governed by the CeCILL license under French law and abiding by the rules of distribution of free software. It is distributed through the web site: `pinkhq.com` that has been created and is maintained by Laszlo Marak.

The download is available either in source form or in binary packages for the most popular Linux distributions. Microsoft Windows installer is also available. The source code is stored in a mercurial repository. One can clone it using the following command:

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

The repository is public and it will only require authorization for pushing.

PINK has a doxygen generated documentation. One can generate it from the source or access the latest documentation on the web site. The web site also features a mailing list and a bug tracker.

2 Content

Most existing image processing libraries concentrate on pixel-based operators and linear filtering methods like convolutions, FIR and IIR filters, diffusion and so on. PINK complements these with implementations of over 200 algorithms for image segmentation and non-linear filtering. Whereas most operators come from mathematical morphology, discrete geometry and discrete topology, some operators from other fields are implemented as well.

All the operators feature research-level, scientific journal-described algorithms with 3D extension where possible. Notable operators include skeletonization and topological thinning, mathematical morphology, watershed, maximum flow, total variation filtering, and more.

The list of available operators is structured according to the following categories (the number of operators is given in parentheses):

- Arithmetic operators (39)
- Format and type conversion (56)
- Mathematical morphology (45)
- Digital connectivity (41)
- Digital topology, binary (48)
- Digital topology, grayscale (33)
- Orders topology (27)
- Geometrical operators (68)
- Graphic primitives (21)
- Histogram-based operators (9)
- Signal processing (13)
- Statistics (3)
- Three-dimensional meshing (6)

Some categories, such as mathematical morphology, digital connectivity, digital topology... are quite developped, in reflect of the domains of expertise of our group. Other ones just contain a few useful operators that were needed in some applications.

3 Implementation

3.1 Core operators library

The development of the core image operators library was mostly done in standard C language. The main image data structure is quite simple and allows for the representation of the most commonly used image types: 1D, 2D, 3D, 4D, color images, image sequences, and the most commonly used pixel types: byte, short and long integer, float, double, complex. Not all operators accept all image and pixel types, as genericity was not a priority in this project. Some conversion operators are provided as a workaround to cope with this drawback.

3.2 Bash front end

All operators from the core library come with a basic interface to be used in shell window or scripts. The operator basically reads its input in file(s), process it, then writes its output in other file(s).

3.3 Python front end

It is build upon Boost Python and the Python Imaging Library, and uses plug-ins that use python-vtk, numpy and matplotlib. While the core foundation has been in development for over a decade, its Python front-end is recent work.

The PINK dedicated front end is exposed using Boost Python. The library is designed to behave natively in Python, with the functional programming paradigm. Operators are functions and the images are returned after the operation as results. This way complex image processing pipelines and algorithms can be easily scripted in Python. This enables an easy plugin creation or development of solutions for particular applications (pre-processing—segmentation—post-processing).

The Python front-end makes it suitable to use PINK together with other packages like SAGE, PyLab or Tkinter. This makes it possible to develop image segmentation and processing methods that are using graphical interfaces for parametrization. The operators can be demonstrated using Python's facilities in an interactive manner. This functionality is particularly useful as a learning tool.

From the technical point of view, the images in PINK are exposed in Python as objects. The operators are functions, with images as first parameters. As result, they return a processed image. Pixels, if necessary, can be accessed with intuitive operators. This means that simple algorithms for experimenting can be implemented directly in Python with speed as the only constraint. Notice that, unlike with the bash front end, intermediate results are not stored in files, resulting in important time gains.

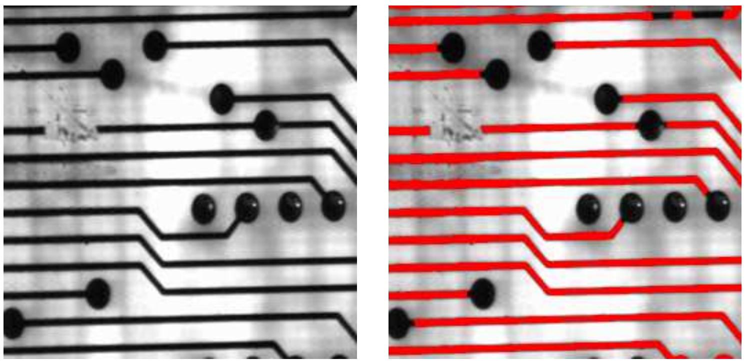
By design new C/C++ operators can be exposed using only a few lines of C++ code, completely omitting third party languages. Indeed, the complete library is implemented using C/C++ and Python. It has been ported to most of the popular Linux distributions and it also runs on OSX and Microsoft Windows.

4 Example

Here is a basic example of a Python script that calls PINK morphological operators to extract the wires in a printed circuit board image. Examples of input image and result are displayed in figure 1.

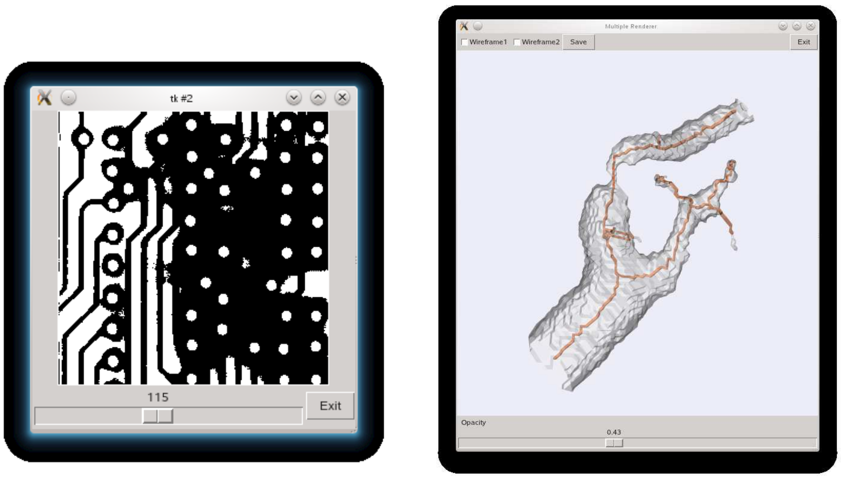
```
def FindTheWires(image, threshold):
    binary = pink.threshold(image, threshold, 0, 255)
    inv = pink.inverse(binary)
    eros = pink.erosball(inv, 2)
    filtered = pink.geodilat(eros, inv, 8)
    filled = fill_the_holes(filtered)
    open = pink.openball(filled, 6)
    joints = pink.geodilat(open, filled, 8, 1)
    result = filled - joints
    return result
```

Thanks to the Python-TK GUI it is fairly easy to create interactive programs that apply a treatment on any given image, under the control of a parameter, and allow the user to visualize the effect of parameter changes. In this sample code, a simple threshold is applied, but the same works for arbitrarily complex user-defined procedures. See illustration in figure 2(a).



(a) (b)

Figure 1: (a) Image of a PCB. (b) Result of the procedure FindTheWires (in red).



(a) (b)

Figure 2: (a) Example using the Python-TK GUI. (b) 3D visualisation using VTK-Python.

```
Im = pink.cpp.readimage("circuit2.pgm")

def binarise(value)
  global Im
  return pink.threshold(Im, value)

pink.manipulate(circuit, 0, 255)
```

Pink supports 3D visualization with VTK-Python. Meshes can be generated from 3D objects (skeletons, segmented objects...). Figure 2(b) illustrates the visualization of both original data and result of a treatment (filtered skeleton).