IM, an Imaging Toolkit

Antonio Scuri

The author is a senior developer at the Computer Graphics Technology Group (Tecgraf) of the Pontifical Catholic University of Rio de Janeiro (PUC-Rio). He is responsible for the Portable User Interface toolkit (IUP), the Canvas Draw graphics library (CD), the Imaging toolkit (IM), and other tools for desktop application development.

Contact with the author: scuri*@tecgraf.puc-rio.br*

Abstract

IM is a toolkit for Digital Imaging. It provides support for image capture, several image file formats and many image processing operations. Image representation includes scientific data types and attributes. Animation, video and volumes are supported as image sequences. The main goal of the library is to provide a simple API and abstraction of images for scientific applications. It has a very flexible license and can be used for public and commercial applications. The library and its API are implemented in C, but it has a binding to the Lua language.

This article will present a small overview of the library concepts and show a few examples.

Getting Started

IM is based on four concepts: Image Representation, Storage, Processing and Capture. Image Visualization is a task that it is left for a graphics library, like OpenGL or the Canvas Draw toolkit. Figure 1. Illustrates the relationship of these concepts.

Image Representation describes the image model and its details: which color systems are going to be used, which data types, how the data is organized in memory and how other image characteristics are managed.

Image Storage describes the file format model and how images are loaded and saved. Image Capture describes how to obtain an image from a capture device. And Image Processing describes the image processing operations.

There are multiple ways to implement these concepts, but there is no common definition in the literature. Although there is a standard called Programmer's Imaging Kernel System (PIKS), it is a very complete and also very complex standard, so it is difficult to implement and not very popular.

The idea behind IM was to create a toolkit that was not complex, neither big nor widespread, but one that can be used as a solid base to the development of thesis and dissertations, as for

commercial applications. As this environment is very heterogeneous, the IM project chooses some directives:

- Portability (Windows and UNIX)
- C API
- Totally Free and Open Source
- Focus in Scientific Applications
- Easy to Learn
- Easy to Reuse

Considering these directives there are only a few similar toolkits, but none can be considered equivalent. The most popular like VTK or OpenCV are too complex, too big, or to hard to reuse code.

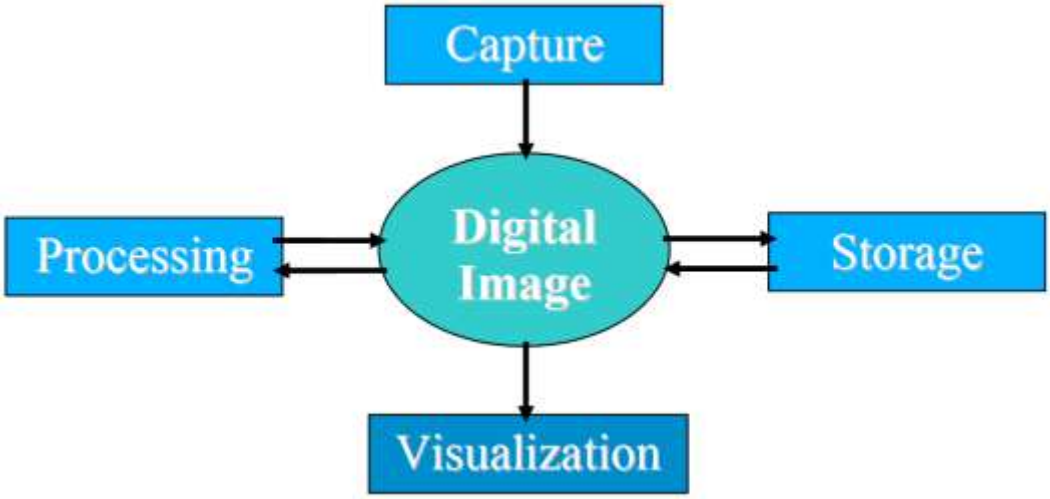

Figure 1. IM Toolkit Concepts

Image Representation

In IM only four parameters define the raw image data: width, height, color mode and data type.

The color mode is a combination of the color space and some additional flags. Color spaces can be:

- RGB (nonlinear)
- MAP (indexed RGB)
- GRAY (luma or non color intensity)
- BINARY (black & white)
- CMYK (nonlinear)
- YCBCR (ITU-R 601)
- LAB (nonlinear)

- LUV (nonlinear)
- XYZ (linear)

Raw data organization is defined with additional flags:

- with alpha channel or not
- top down or bottom up orientation
- packed or separated components

The data types are a subset of the standard C data types:

- BYTE
- USHORT
- INT
- FLOAT
- COMPLEX FLOAT

The raw data buffer is always byte aligned and each component is stored sequentially in the buffer following the specified packing. For example, if a RGB image is 4x4 pixels it will have the following organization in memory:

RRRRRRRRRRRRRRRRGGGGGGGGGGGGGGGGBBBBBBBBBBBBBBBB - for non packed components

RGBRGBRGBRGBRGBRGBRGBRGBRGBRGBRGBRGBRGBRGBRGB - for packed components

If a pointer uses the correct data type, then to locate the pixel at line y, column x, component d simply write:

```
if (is_packed)
   value = idata[y*width*depth + x*depth + d]
else
   value = idata[d*width*height + y*width + x]
```

This data organization is restricted to raw data buffers. For the high level image processing functions, we created a structure called **imImage** that eliminates the extra flags and assume bottom up orientation and separated components.

The conversion between image data types or color spaces and all the image processing functions are defined only for the **imImage** structure.

Listing 1. Processing an imImage data

```
for (int d = 0; d < image->depth; d++)
{
   unsigned char* idata = (unsigned char*)image->data[d];
   for (int y = 0; y < height; y++)
   {
      for (int x = 0; x < width; x++)
```

```
            {
               int offset = y * width + x;
               idata[offset] = 255;
            }
         }
      }
```

Image Storage

Essentially all the file formats save the same image data. There is no such thing like a GIF image, instead we have a color indexed image that can be saved in a file with a GIF format, or a TIFF format, etc. However the compression encoding can be lossy and degrade the original image. The point is that file formats and image data are two different things.

A file format is a file organization of the image data and its attributes. The IM toolkit storage model considers all the file formats under the same model, including image, video, animation, stacks and volume file formats. When there is more than one image each one is treated as an independent frame. Each frame can have its own parameters and set of attributes.

The most popular file formats are supported: TIFF, BMP, PNG, JPEG, GIF and AVI. Image metadata is supported by using generic name based attributes with variable data types.

Listing 2. Loading Raw Image Data

```
    /* loads the first image in the file, using no bitmap
conversion and the same data organization as in the file. */
    imFile* ifile = imFileOpen(file_name, &error);
    imFileReadImageInfo(ifile, 0, &width, &height, &color_mode,
&data_type);
    imFileReadImageData(ifile, data, 0, -1);
    imFileClose(ifile);
```

Listing 3. Loading an imImage

```
    /* loads the first image in the file, using no bitmap
conversion, but with the imImage data organization. */
    imImage* image = imFileImageLoad(file_name, 0, &error);
```

Image Processing

It is used the simplest model possible for image processing: a function with input data, output data and control parameters.

The operations have usually one or more input images, and one or more output images. In-place operations are also available, so many operations can use the same data for input and output. The input and output data type, color mode or size may be different depending on the operation. All of these details are described in each function documentation.

There is no ROI (Region Of Interest) management, but you can crop, process, then insert the result in the original image.

About a hundred Image Processing operations are available, including geometric, arithmetic, histogram and convolution operations.

Listing 4. Hough Lines

```
imProcessCanny(in,out,stddev);
imProcessHysteresisThreshold(in,out,low,high);
imProcessHoughLines(in,out);
imProcessLocalMaxThreshold(in,out,size,min);
```

Listing 5. Image Analysis

```
imProcessSliceThreshold(in, out, level1, level2);
imProcessPrune(in, out, connect, size1, size2);
imProcessFillHoles(in, out, connect);
imAnalyzeFindRegions(in, out, connect);
imAnalyzeMeasureArea(in, area);
imAnalyzeMeasurePerimeter(in, perim);
```

Image Capture

The capture support is designed for live video. It is not designed for passive digital cameras that only transfer the already taken pictures. Valid sources includes: USB cameras (like most Webcams), Firewire (IEEE 1394) cameras, and analog video capture boards, including TV Tuners. These are called devices.

The capture functions allow you to list the available devices, connect to a device, configure the device, and retrieve an image.

Currently, Image Capture is implemented only in Windows. It is independent of the other libraries and can capture from more than one source at the same time.

On the Web

- IM - An Imaging Toolkit
  http://www.tecgraf.puc-rio.br/im/
- CD - Canvas Draw, a 2D graphics library
  http://www.tecgraf.puc-rio.br/cd/
- The Programming Language Lua
  http://www.lua.org/
- IMLAB - An Experimental System for Image Processing
  http://www.tecgraf.puc-rio.br/~scuri/imlab/

Final Remarks

IM is a new and unexplored solution for an imaging toolkit, and it is proposed as a simple and clean one. It has a different approach, its organization was designed so it can be used for teaching imaging concepts and its image processing operations source code can be easily reused. But to became a real solution it needs to be evaluated by a larger number of users.

For the future we intend to complete the support for Linux specific libraries, add some scientific file formats and improve the image processing library.

This work was developed at Tecgraf by means of the partnership with PETROBRAS/CENPES.

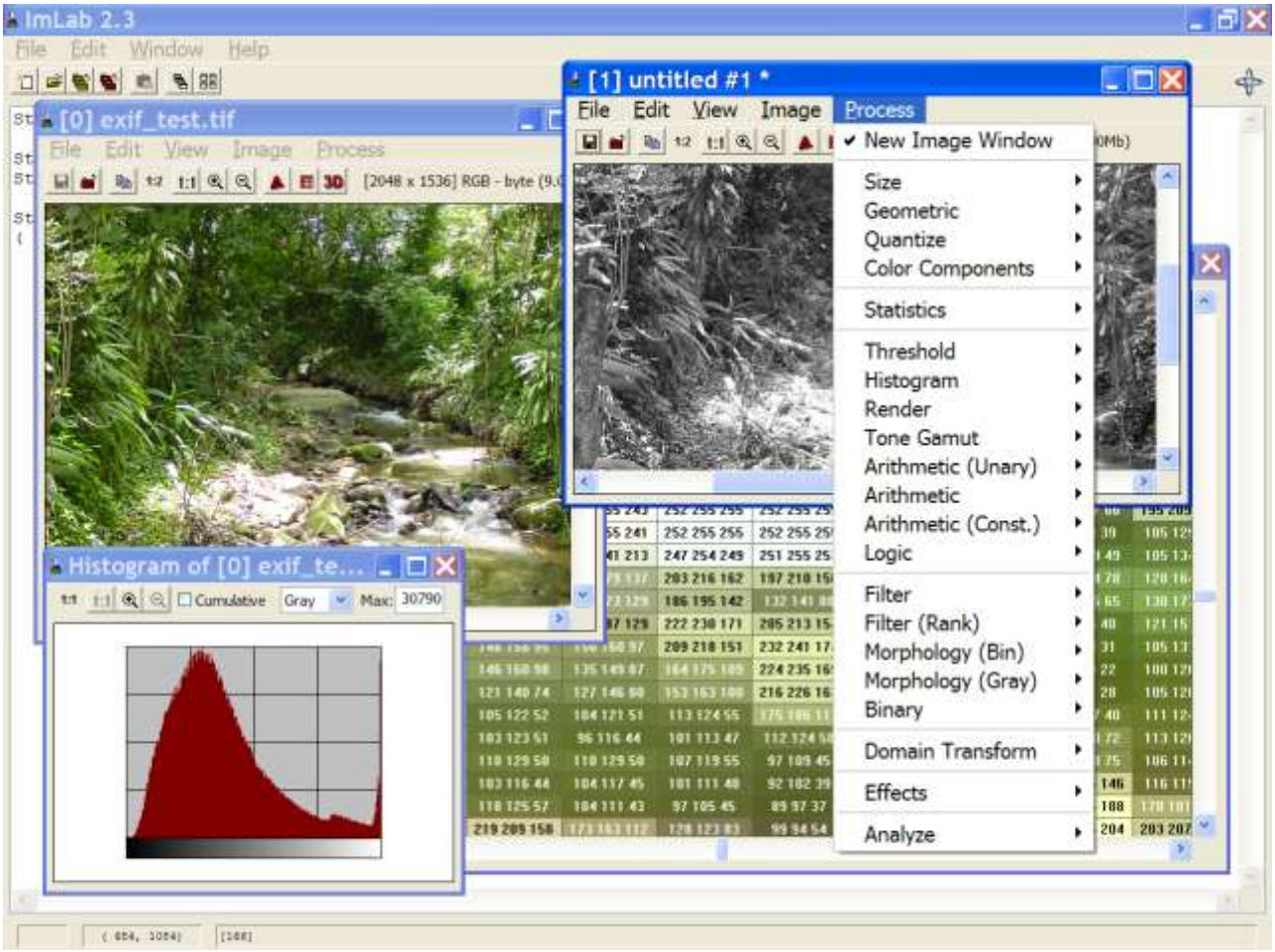

Figure 2. IMLAB - An Experimental System for Image Processing