Graph Cut Library

Contributed by Pavel Matula
Last Updated Monday, 09 July 2012

Introduction

Graph Cut library - Gc
in short - is a library focusing on combinatorial
optimization via graph cuts and its use in digital image analysis, especially for finding optimal solutions to energy minimization based discrete labeling problems such as image segmentation. This research field
has become very popular in the last decade and many interesting
algorithms are built upon graph cuts. The library is being developed in C++ and places emphasis especially on speed and low memory usage as well as clean and extensible object-oriented design.

Algorithms

The library offers implementation of several popular algorithms from
the field. Here is the list of the most interesting ones:

- Maximum flow algorithms - The library includes a wide range
of the most popular maximum flow algorithms in image processing (such as Boykov-Kolmogorov or Push-Relabel methods) all of
which are highly optimized and exception safe. It contains implementations for both general directed graphs and grid graphs.

- Metric approximation - Implementation of Euclidean and Riemannian metric approximation via graph cuts.

- Multi-label discrete energy optimization - Implementation of popular alpha/beta swap and alpha expansion algorithms.

- Image segmentation - Implementation of the graph cut based minimization of the popular Chan-Vese and Mumford-Shah segmentation models.

Supported platforms

One of the primary goals of the library is to be cross-platform
and to avoid ugly platform or compiler specific hacks. Basically, it
should be possible to build the library on any platform using a C++ compiler
with decent support for modern standards (particularly templates). The library does not have any 3rd party library dependencies. We
have sucesfully compiled and used the library on the following
platforms:

- Microsoft Windows XP and newer, both x86 and x86-64 architectures, using Visual Studio compiler (versions 2008 and 2010 tested).

- Linux, both x86 and x86-64 architectures, using GCC compiler (several versions including 4.1 and above tested).

License

The library is licensed under the GNU Lesser General Public License. Its text is included with the library and is also accessible from the documentation, see The GNU General Purpose License and The GNU Lesser General Purpose License.
We also encourage you to reference this library if you use it. Finally, note that some of the algorithms implemented in this library may have their own licensing rules (like not being available for commercial use or requiring citation of some publications) and some may have patents pending. We tried to mention these in the detailed documentation of such algorithms, but we can't guarantee this information is complete, so be careful.

Authors

The library has been developed at Centre for Biomedical Image Analysis at the Faculty of Informatics, Masaryk University, Brno, Czech Republic. Currently it has only one author:

- Ondrej Danek, contact: ondrej.danek@gmail.com, web: http://www.ondrej-danek.net

Download

Downloads are free but protected by login and password which one may obtain after filling the Registration form.

- Gc library (April 26th 2012)

Documentation

Documentation can be generated from the sources using the Doxygen tool .

- Online documentation (generated April 26th 2012)

MATLAB interface

Currently the MATLAB interface is avilable for several graph-cut based segmentation algorithms. A compiled binary version of the interface for 32-bit and 64-bit Windows can be downloaded here:

- Gc MATLAB interface (March 5th 2012)

Sources are supplied together with the main library (see above). The interface has been compiled and tested in MATLAB version 2010a.

Chan-Vese segmentation example:

```
a = imread('img/cameraman.tif');
imshow(a)
an = gc_normalize_image(single(a));
seg1 = gc_chan_vese(an, 1, 10, 10);
figure; imshow(seg1)
seg2 =  gc_chan_vese(an, 1, 1, 1);
figure; imshow(seg2)
```

Results:

Mumford-Shah segmentation example:

```
a = imread('img/fruit.tif');
imshow(a)
an = gc_normalize_image(single(a));
seg = gc_mumford_shah(an, 4, 10);
figure; imshow(seg*64)
```

Results: