Scale-Invariant Sketch Tokens for Edges and Ridges v.1.1

Matlab code for extracting scale-invariant edge and ridge tokens.

Iasonas Kokkinos ⟨jkokkin@stat.ucla.edu⟩
http://www.stat.ucla.edu/~jkokkin
Release date: 9/5/2008

1 Introduction

This code is an implementation of T. Lindeberg's scale-invariant primal sketch [5, 4] and is used for object detection in [2] and shape analysis in [3].

The code takes as input a grayscale image and convolves it with Gaussian kernels of increasing size. Using this scale-space, edge and ridge contours are found as maxima in space and scale of appropriate differential operators. These contours are then post-processed to obtain a sparse set of straight line segments.

Apart from computing T. Lindeberg's primal sketch, additional functionalities that are generally useful include:

- Efficient Gaussian filtering with a mex implementation of R. Deriche's IIR filters [1].
- A line tracking algorithm along the lines of Nevatia and Babu [7].
- The line segmentation algorithm used in D. Lowe's SCEPRO system [6].

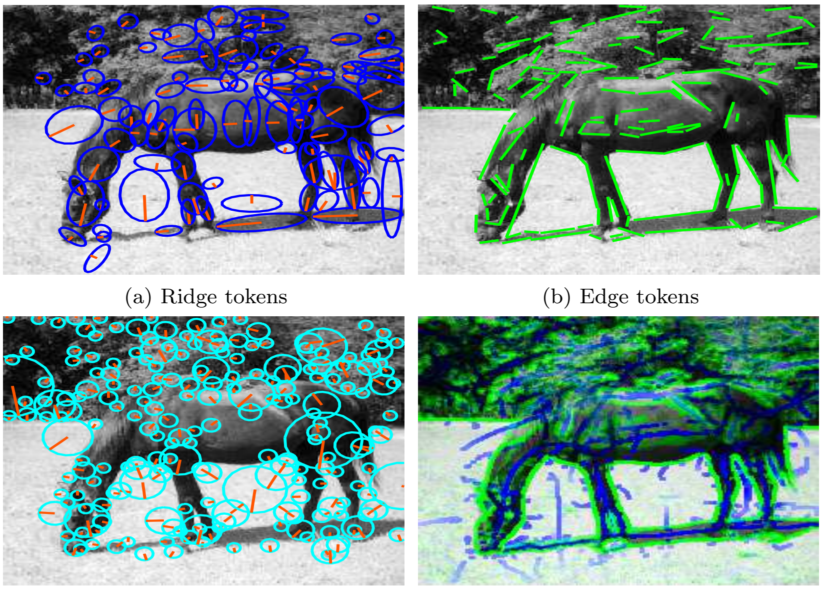
2 Usage

You first need to place the root folder and all of its subfolders in the Matlab path. For this go from the Matlab command window to File→Set Path→Add with Subfolders and choose the folder 'PS_primal_sketch'.

The input to the code is a grayscale image normalized to lie in [0,1], e.g.:

```
input_image = imread('horse010.jpg');
input_image = double(input_image)/256;
```

Figure 1: Input image.



(a) Ridge tokens (b) Edge tokens

(c) Blob tokens (d) Ridge and Edge contours.

Figure 2: Scale-invariant primal sketch contours and tokens. For *ridges*, the length and orientation of the orange lines indicate the feature size and symmetry axis respectively. The ellipse width indicates ridge scale. *Edges* are found at fine scales so we illustrate them as line segments, instead of ellipses. Finally, *blobs* are depicted as circles, where the length and orientation of the lines amount to the scale and symmetry axis of the token.

The gateway routine PS00___primal_sketch can then be called as:

```
[ridges,edges,blobs,contours] = PS00___primal_sketch(input_image);
```

This gives (a) the ridge, edge and blob tokens and (b) the contours formed by the ridge/edge maxima points. We can access all the information related to the token coordinates and attributes (scale, orientation, point set); e.g. typing 'ridges' on the Matlab prompt gives:

```
ridges =
           lines: {1x98 cell}
      merit_geom: [1x98 double]
    orientations: [1x98 double]
          ratios: [1x98 double]
          scales: [1x98 double]
             c_m: [1x98 double]
             c_n: [1x98 double]
            ener: [1x98 double]
```

The ridge structure thus contains information related to:

- Location ('c_m,c_n' fields)
- Orientation, length and elongation ('orientations','scales','ratios' fields)
- Average energy ('energy' field)
- Raw sketch information ('lines' field)

To visualize the results, you can use the following commands:

```
% display ridge lines as ellipses. Width is proportional  to scale.
show_ridges_on_image(input_image,ridges);
% display edges lines as strokes.
show_edges_on_image(input_image,edges);
% display blobs as circles.
show_ellipses_on_image(input_image,blobs);

% form contour strength images
contour_ridge      = PSzz_unzip_contour(contours{1});
contour_edge       = PSzz_unzip_contour(contours{2});
% and superimpose them on the input_image
colored_sketch     = colored_contours(input_image,contour_ridge,contour_edge);
imshow(colored_sketch);
```

2.1 Demonstration Programs

Three demos are included to show the code's functionality at three different levels.

- demo1.m shows how to call the code if one is only interested in applying it as a preprocessing step for object detection.
- demo2.m shows how one can tune the code's internal workings by modifying its parameters. The parameters are defined in PS0z1__settings_sketch, PS0z1__settings_tokens. You can override these settings as shown in demo2.m.
- demo3.m shows how to manipulate and visualize the intermediate steps used for primal sketch computation. It shows:
  - The evolution of the normalized feature strength as a function of scale.
  - The contours of joint maxima in scale and space.
  - The connected components of these contours.
  - The curve segments obtained by Lowe's algorithm and the corresponding line strokes.
  - A set of markers obtained from ridge tokens.

2.2 Accessing Intermediate results

The 'raw data' related to each token can be accessed by using the command

```
[crd_x,crd_y,energy,width]  = PSzz_token_points(edges,k);
```

which gives the locations and energy/width information for the points composing the k-th edge (or ridge) token.

Further, PS00___primal_sketch can return the connected components of the ridge/edge maxima contours. These again can be accessed with the command

```
[crd_x,crd_y]  = PSzz_token_raw_information(component,k_ind);
```

where k_ind is again the index into the component we want.

3 Code Internals

3.1 Documentation

You can see the function call hierarchy on /doc/primal_sketch/graph.html and browse through the code starting from /doc/menu.html. This documentation was generated using G. Flandin's m2html package, distributed from the Mathworks site.

3.2 Conventions

There are several conventions I use, which help control the coding and debugging processes; knowing these will help understand and utilize the code.

3.2.1 Filename Format

The first few letters of the filenames reflect the order in which the files are called [1]. This helps organize the files and keep track of who calls whom during debugging. For this you need to:

- Open a Matlab editor and an explorer window at the root directory, and drag all the files from the explorer into the Matlab editor.
- Put the file-tab bar on one horizontal column: go on bar →right click→bar position→left/right.
- Sort the files by name: go on the bar→right click→alphabetize.

This helps rapidly find the file you need.

3.2.2 Structure manipulation

I extensively use my expand_structure and compress_structure routines within the code. These are used primarily to unclutter the code, particularly when many related variables are used. As a working example, consider the variables:

```
im_sc, d_x, d_y, d_xx, d_yy, d_xy
```

in PS1z2__get_gaussian_jet. They are bundled in a single structure as follows:

```
fields_wt = {'im_sc','d_x','d_y','d_xx','d_xy','d_yy'};
compress_structure;
gauss_jet = structure;
```

which is equivalent to the code:

```
gauss_jet.im_sc = im_sc;
gauss_jet.d_x   = d_x;
gauss_jet.d_y   = d_y;
...
```

Note that the 'fields_wt' and 'structure' variable names should always be used in conjunction with the 'compress_structure' script.

In the same way, in PS1z3__get_feature_strength the code:

[1] The PSL* files were written as a separate module and therefore do not strictly follow this convention.

```
structure = gauss_jet; expand_structure;
```

does the same thing with

```
im_sc = gauss_jet.im_sc;
d_x   = gauss_jet.d_x;
d_y   = gauss_jet.d_y;
..
```

4 Change log

4.1 v1.1

- Fixed a bug in the detection of maxima along orientations; this results in fewer false positives for ridges.
- Added a wrapper for other people's boundary detectors. I have found the Berkeley group's edge detector to be more robust to clutter, so here is how you can use it:

  ```
  edge_map =  pbBGTG(input_image);  %% use the berkeley detector
  thresh_energy = .1;               %% set threshold for rejecting edge tokens
  [maxima,edges,edge_skeletons] =  ...
  PSzz___turn_edgemap_to_tokens(edge_map,thresh_energy);  %% turn into tokens
  ```

  'maxima', 'edges', and 'edge_skeletons' are the same data-structures as the ones returned by the original primal sketch code.

4.2 Dependencies

The code has the following dependencies on mex files:

- The mex file iir_gauss.cpp located in the 'filtering' folder. Binaries are provided for Windows XP. Type mex iir_gauss.cpp on the Matlab prompt to compile on different OS'. Compilation works for both the Visual Studio and the Matlab built-in compiler.
- The mex implementation of k-d trees by Guy Schecter distributed on the Mathworks site. Contains binaries for Windows, Mac OS X, and Redhat Linux as well as source code.

5 Todo List

Some extensions/improvements that should eventually be implemented include:

- Finding the intersection of the 2D surfaces defining the edge/ridge locations [4] using a more accurate algorithm, e.g. marching lines.
- Handling triple points when finding continuous edge/ridge contours.
- Dealing with large gaps in the edge/ridge maps.
- Introducing a multi-resolution scheme to improve time efficiency.
- Introducing corners/junctions.
- ...

6 Acknowledgements

Financial support for this software has been provided by the the FP6 European Network of Excellence MUSCLE, under contract no. IST-FP6-507752.

Most of the code was developed for the paper [2] while I was a graduate student in the Computer Vision, Speech Communication and Signal Processing Group, in the School of ECE in the National Technical University of Athens, Greece: `http://cvsp.cs.ntua.gr/`

I am currently using this code in my research at the Center for Image and Vision Sciences in UCLA: `http://civs.stat.ucla.edu/`

7 Contact

You may find more information about my research at:
`http://www.stat.ucla.edu/~jkokkin`
Please contact me for any bugs/fixes/suggestions at: `jkokkin@stat.ucla.edu`

8 Terms

If you use our code in your research, please cite [2].

References

[1] Deriche, R. Recursively Implementing the Gaussian and its Derivatives. Tech. Rep. 1893, INRIA, Unite de Recherche Sophia-Antipolis, 1993.

[2] Kokkinos, I., Maragos, P., and Yuille, A. Bottom-Up and Top-Down Object Detection Using Primal Sketch Features and Graphical Models. In *IEEE Conf. Computer Vision and Pattern Recognition* (2006).

[3] Kokkinos, I., and Yuille, A. Unsupervised Learning of Object Deformation Models. In *Int.'l Conf. on Computer Vision* (2007).

[4] Lindeberg, T. Edge Detection and Ridge Detection with Automatic Scale Selection. *Int.'l Journal of Computer Vision 30*, 2 (1998).

[5] Lindeberg, T. Feature Detection with Automatic Scale Selection. *Int.'l Journal of Computer Vision 30*, 2 (1998).

[6] Lowe, D. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence 31*, 3 (1987), 355–395.

[7] Nevatia, R., and Babu, K. Linear feature extraction and description. *Computer Graphics and Image Processing 13*, 3 (July 1980), 257–269.