Université de Bordeaux

Master ISV, Image 3D
2015/2016

Image segmentation and image analysis - Practical work 2

Watershed

The watershed method is an efficient tool to analyze the topography of an image. It is used for image segmentation since it can compute a partition of the initial image. In this context, the watershed method is generally applied to an edge image where strong edges can be seen as ridges separating wide valleys (homogeneous regions). Figure 1 gives an example of watershed segmentation. You will implement an algorithm for computing the watersheds of an image (Vincent and Soille, IEEE PAMI, 1991). This algorithm is based on the idea of "flooding" the image to detect the watersheds. It can be synthesized as :

1. The flooding is gradual. At each step, the pixels of level h are processed. All the pixels of level $< h$ have been processed before. We thus have a set of incomplete basins and some boundary pixels (marked as `WATERSHED` in the code) which separate two neighboring basins.
2. The pixels of level h are then either added to an existing basin, or marked as boundary pixels, or recognized as belonging to a new basin.

The first step is realized by sorting all the pixels according to their value. The second step requires processing the pixels in a certain order.

a	b
c	d

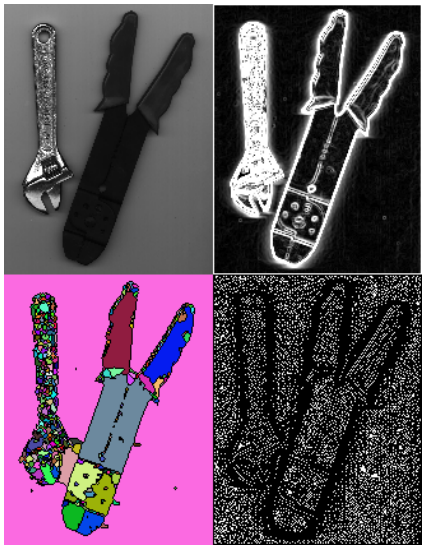

FIGURE 1 – (a) initial image, (b) image of the gradient norm (Sobel), (c) watershed, (d) regional minima (white).

Experimentation

The archive (http ://dept-info.labri.fr/∼vialard/Image3D/src/files2.tgz) contains :
— a directory `images` containing test images
— a directory `bin` containing the .class files of an `ImageJ` plugin computing watersheds.
— a directory `src` containing a part of the source files of the plugin.
Test the plugin with the following parameters :

Image	input of the algorithm	ϵ
laBaule_spot	gradient	5
laBaule_spot	gradient	25
tools	gradient	5
tools	gradient	20
tools	gradient	40

Apply a mean filter to the images and repeat the same tests. Do the same with a median filter.

What do you notice ? The parameter `epsilon` is used to consider pixels with similar altitude as having the same altitude. How is it used in practice ? Is the algorithm robust against the image noise ?

Computing the watersheds

The inputs of the algorithm are :
— an image of floats (pixels altitudes)
— a float ϵ (uncertainty on altitude)

The output of the algorithm is an image of integers : each pixel contains either a basin label or `WATERSHED` if it is on the boundary between two basins.

Edit and read carefully the class `Watershed` and particularly its method `computeWatershed`.

The algorithm follows the following steps, already written or to be completed (see also Figure 2 and the algorithm description in the article of Vincent and Soille).

1. The pixels are sorted by increasing order of altitude.
2. They are processed group by group : all the pixels between h and $h + \epsilon$ are considered to have the same altitude.
3. These pixels are `MASK`ed and you have to determine to which basin they belong.
4. The new MASKed pixels which touch a basin (labeled pixel) or a boundary (WATERSHED pixel) are processed first. They are said to be at distance 1. The other MASKed pixels will be at distance 2, 3, ... See (II) in the code `Watershed.java`.
5. These pixels are inserted in a queue. A fictitious pixel indicates the end of the queue and thus the beginning of a sequence of pixels with higher distances. See (III) in the code.
6. Given a pixel in the queue, it is processed according to its neighborhood. Implement the labeling of the current pixel (IV.TODO in the code). Some of the possible cases are illustrated in Figure 3. The following points are important :
   — Only the neighbors with a distance inferior to the current distance are tested (the other neighbors with the same distance as the current pixel are processed in parallel).
   — The four adjacent pixels are sequentially tested : the current pixel can thus be relabeled several times.
   — The pixel will be labeled either with a neighboring label or with the WATERSHED label.
   — The neighboring pixels with distance+1 are inserted in the queue. When they are discovered, their distance is 0 and they are MASKed.
7. The last step of the algorithm is the labeling of the news basins. Each pixel of the current altitude that has not been labeled is given a new label which is propagated to its non labeled neighbors. See (V. TODO).

Do the tests of the first question with your own implementation.

Closing the boundaries between basins

The watershed that are computed with this algorithm are not always closed boundaries around regions. Explain why. Fill in the method `closeWatershed` to close the boundaries after the watershed computation.

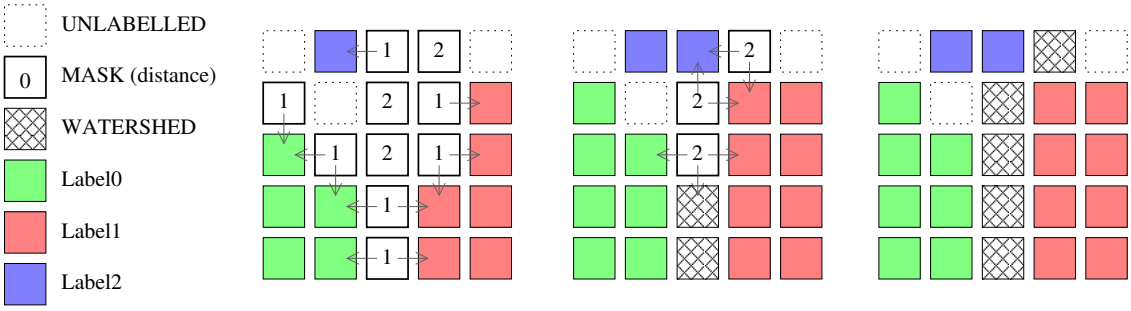

FIGURE 2 – A step of the algorithm. For each MASKed pixel, the arrows indicate the **only neighbors** that are taken into account for computing its label. (a) labels, (b) processing of the MASKed pixels at distance 1 from the boundary, (c) processing of the MASKed pixels at 2, (d) end of the iteration.

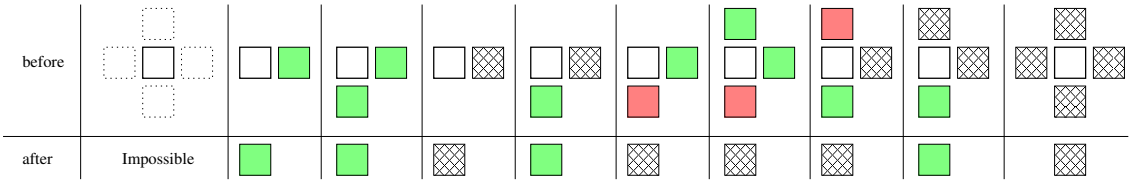

FIGURE 3 – Labeling of a MASKed pixel. The new label depends on the labels of the neighbors. Some configurations are illustrated.

The key idea is the following : for each pixel p of the image, if there is a neighboring pixel with an inferior label, then p is considered as a boundary pixel.

Is this closure algorithm independent from a rotation or symmetry of the image ?

Displaying the basins according to the mean value

Implement another way to display the basins, no more with a random color, but with the mean gray-level of the corresponding region in the initial image. Is there more than a visual interest in doing this ?

To go further...

Merging step

The algorithm often provides a lot of non significant small regions. Write a method of the class `Watershed` which aggregates the small regions to neighboring regions, by considering the lowest watershed first.

Markers

Another method to limit the number of small regions is to use markers. The user marks a group of pixels inside each region to be segmented. Basins that do not contain markers merge when they meet. If a basin contains a marker, it absorbs basins which don't. If two basins containing a marker meet, then, and only then, the watershed between them is built.