IPOL
IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2019–10–08.
Submitted on 2019–02–18, accepted on 2019–05–31.
ISSN 2105–1232 © 2019 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2019.250

Watervoxels

P. Cettour-Janet[⋆,1], C. Cazorla[⋆,1], V. Machairas[2], Q. Delannoy[1], N. Bednarek[1,3], F. Rousseau[4], É. Decencière[2], N. Passat[1]

[⋆] P. Cettour-Janet and C. Cazorla are equal first authors
[1] Université de Reims Champagne-Ardenne, CReSTIC, France
[2] Centre de Morphologie Mathématique, Mines ParisTech, PSL Research University, Paris, France
[3] Service de médecine néonatale et réanimation pédiatrique, CHU de Reims, France
[4] IMT Atlantique, LaTIM U1101 INSERM, UBL, Brest, France

Communicated by Pascal Monasse and Sébastien Drouyer *Demo edited by* Sébastien Drouyer

Abstract

In this article, we present the n-dimensional version of the waterpixels, namely the watervoxels. Waterpixels constitute a simple, yet efficient alternative to standard superpixel paradigms, initially developed in the field of computer vision for reducing the space cost of input images without altering the accuracy of further image processing / analysis procedures. Waterpixels were initially proposed in a 2-dimensional version. Their extension to 3-dimensions—and more generally n-dimensions—is however possible, in particular in the Cartesian grid. Indeed, waterpixels mainly rely on a seeded watershed transformation applied on a saliency map defined as the linear combination of a gradient map and a distance map. We propose a description of the algorithmics of watervoxels in n-dimensional Cartesian grids. We also discuss its parameters and its time cost. A source code for 2- and 3-dimensional versions of watervoxels is provided, such as a 2-dimensional demonstrator. This article can be seen as the companion of the article "Waterpixels", published in 2015 in IEEE Transactions on Image Processing.

Source Code

Source codes for the described algorithms are provided in the web page of the article[1].

Keywords: waterpixels; superpixels; seeded watershed; 3-dimensional images; Cartesian grid

[1] https://doi.org/10.5201/ipol.2019.250

1 Introduction

Superpixels [11] have emerged as a relevant low-level, pre-processing approach for simplifying large images, especially in the context of computer vision applications, that require rapid (ideally real-time) operators compliant with embedded vision tools.

In theory, the idea of superpixels is rather simple: it consists of reducing the combinatorics of an image (i.e. the number of pixels) without altering neither the spectral information (pixel values) nor the pixel organization (pixel topology). In other words, superpixels aim at building a smaller, information lossless image from the initial image, thus allowing to reduce the time cost of further procedures acting on it.

In practice, satisfying the three constraints of (1) image size reduction, (2) information preservation (3) non-alteration of the structure, is difficult, since they are generally antagonistic.

In this context, many superpixel paradigms have been proposed. Among them, the SLIC superpixels [1] are considered as the state of the art. In the meantime, many other superpixel paradigms were investigated. Among these alternatives, the waterpixels, initially proposed in [6] and further compared to SLIC superpixels [7] exhibit good performances with an efficient, linear time cost.

The paradigm of waterpixels is mainly based on the watershed transformation [2, 13], and more precisely its marker-based variant that allows one to control the number and position of the regions forming the image partition. Basically, waterpixels consist of computing the seeded watershed transformation of a saliency map built as the linear combination of a gradient map (ensuring the coherence of the superpixels in terms of signal homogeneity and edge-fitting) and a distance map from a set of regularly sampled points (ensuring the geometrical coherence and topological consistency of the partition).

In the pioneering articles [6, 7], the waterpixels were defined in the 2-dimensional Cartesian grid. The proposed concepts remain, however, valid in any n-dimensional Cartesian grid, $n \geq 1$. In the current paper, that can be seen as the companion paper of [7], we describe the waterpixel construction in dimension n. A demonstrator for building 2-dimensional waterpixels, and a source code for building the so-called watervoxels in 3 dimensions is also provided.

2 Synthetic Description of the Method

Let $n \in \mathbb{N}$, $n > 0$ be the dimension of the considered image support. For all $i \in [\![1, n]\!]$, let $d_i \in \mathbb{N}$, $d_i > 0$ be the size of the image support in the i-th dimension. The support of the image is then the Cartesian product $\Omega = \prod_{i=1}^{n} [\![0, d_i - 1]\!]$, that is a finite subset of $\mathbb{Z}^n$. In particular, the size of this support is $|\Omega| = \prod_{i=1}^{n} d_i$.

Let $\mathbb{V}$ be a normed vector space, that will allow us to associate a value to each point of the image support. The existence of a norm on $\mathbb{V}$ is indeed required for defining a notion of gradient on the image. For instance, $\mathbb{V}$ may be any gray-scale space ($\mathbb{R}$ or $\mathbb{Z}$) but also a color (RGB, HSV), a multi-/hyperspectral space, or more generally a vector space.

The image I is a mapping from Ω to $\mathbb{V}$, namely

$$\begin{array}{rcccl} I & : & \Omega & \to & \mathbb{V} \\ & & x & \mapsto & I(x) = v \end{array} \tag{1}$$

The set of all such images is noted $\mathbb{V}^{\Omega}$.

The gradient map ∇ on the set of images $\mathbb{V}^{\Omega}$ is defined, for each image I of $\mathbb{V}^{\Omega}$ as

$$\begin{array}{rcccl} \nabla I & : & \Omega & \to & \mathbb{R}_+ \\ & & x & \mapsto & \nabla I(x) = g \end{array} \tag{2}$$

and gives, for each point x of Ω an information on the norm g of the derivatives of I at x. Many definitions of discrete gradients can be designed on $\mathbb{Z}^n$. In general, they rely on finite differences with respect to the norm on $\mathbb{V}$, and can be computed in linear time $\mathcal{O}(|\Omega|)$ with respect to the size of the image support.

Let $M \subseteq \Omega$. The distance map $\Delta(M)$ from M in Ω is defined as

$$\begin{array}{rcccl} \Delta(M) & : & \Omega & \to & \mathbb{R}_+ \\ & & x & \mapsto & \min_{m \in M} \|x - m\| \end{array} \tag{3}$$

for a given norm $\|\cdot\|$ in $\mathbb{R}^n$ or $\mathbb{Z}^n$. For well chosen norms and/or subsets M, such distance maps can be computed in linear time $\mathcal{O}(|\Omega|)$ with respect to the size of the image support.

Let $S = \{s_j\}_{j=1}^k \subseteq \Omega$ be a set of k distinct points of Ω, with $1 \leq k \leq |\Omega|$, used as seeds within Ω. Let $F \in (\mathbb{R}_+)^\Omega$ be a positive gray-level mapping on Ω. The seeded watershed $W(F, S)$ is defined as the mapping

$$\begin{array}{rcccl} W(F,S) & : & \Omega & \to & [\![1, k]\!] \\ & & x & \mapsto & W(F,S)(x) = w \end{array} \tag{4}$$

where w is the index of the seed point $s_w \in S$ such that x belongs to the catchment basin of s_w after the watershed transformation of F with the set of seeds S. In particular, $W(F, S)$ provides a partition of Ω into k connected regions; this partition is total if the watershed is computed in a graph-based fashion, by considering the watersheds on the adjacency links between the points of Ω, following for instance classical adjacencies defined in digital topology [5]. The watershed transformation can be computed in linear time $\mathcal{O}(|\Omega|)$ with respect to the size of the image support of F [9, Chapter 3].

Let $I \in \mathbb{V}^\Omega$ be the image to be processed. We choose the way of building the gradient ∇ and the distance map Δ, as meta-parameters. We define the set of markers $M \subseteq \Omega$ and the set of seeds $S \subseteq \Omega$ (in particular, the size k of S and M and the way they are spatially organized) and the trade-off coefficient $\alpha \in [0, 1]$ between the influence of the gradient and the distance map. Then, the watervoxel transformation $\mathcal{W}$ of I is defined as

$$\mathcal{W}(I, M, S) = W(\alpha \cdot \nabla I + (1 - \alpha) \cdot \Delta(M), S) \tag{5}$$

This watervoxel transformation can be computed in linear time $\mathcal{O}(|\Omega|)$ with respect to the size of the image support of I.

3 Algorithmics and Parameters

From an algorithmic point of view, the watervoxel transformation relies on three standard transformations: (1) the gradient, (2) the distance map and (3) the watershed transformation. These three transformations are simply combined together in order to provide the watervoxels, as expressed in Equation (5). In other words, the algorithmic bricks of the watervoxel transformation can be seen as meta-parameters. They are discussed hereafter, together with the actual parameters, namely the density of markers and the way to define seeds, and the trade-off between gradient and distance maps.

In this section, the discussion is carried out under the most general hypotheses, in terms of image dimensions and value spaces. In the next section, implementation details corresponding to the source codes and demonstrator will be provided in 2- and 3-dimensions, and for gray-level and color images.

3.1 Distance Map and Markers Definition

The distance map defined in Equation (3) depends on the chosen norm $\|\cdot\|$. Since we work in $\mathbb{Z}^n$, that is viewed as the discrete analogue of $\mathbb{R}^n$, it is relevant to consider the space of ℓ_k norms $\|\cdot\|_k$,

defined for $z = (z_i)_{i=1}^n \in \mathbb{R}^n$ as

$$\|z\|_k = \big(\sum_{i=1}^{n} |z_i|^k\big)^{\frac{1}{k}}, \tag{6}$$

for $k \in \mathbb{N}$, $k > 0$ and

$$\|z\|_\infty = \lim_k \|z\|_k = \max_{i \in [\![1,n]\!]} \{|z_i|\}. \tag{7}$$

In general, we mainly focus on the norms ℓ_1, ℓ_2 and ℓ_∞, corresponding to the Manhattan, Euclidean and Tchebychev distances.

The induced distance map Δ on Ω, that is a part of $\mathbb{Z}^n$, depends on the position of the chosen set of markers M. In the general case of $\mathbb{Z}^n$, with $n \geq 3$, the only configurations of markers leading to regular tilings of the image support are indeed the Cartesian samplings. In other words, we choose a gap distance $\delta \in \mathbb{N}$, $\delta > 0$ that should separate the markers of M in each dimension or, equivalently, we define these markers as the points of $(\delta\mathbb{Z})^n$; the density of markers is then δ^{-n}. The induced Voronoi tiling, with respect to any ℓ_k is composed of hypercubes of size δ^n still organized in a Cartesian way. (Note that it is, of course, possible to consider alternatively less regular tilings, for instance inspired from cristallographic configurations [12].)

Then, the distance map $\Delta(M)$ is simply obtained by reproducing in each of these hypercubes, the local distance map corresponding to the ℓ_k distance of each point to the center of the hypercube. This computation has a constant cost at each point, as it is directly obtained from the coordinates of the points vs. those of the center point.

It is worth mentioning that we can combine Δ with an increasing transformation, for instance the square $v \mapsto v^2$ or exponential mapping $v \mapsto e^v$, in order to increase the weight of high distances.

Finally, it is important to keep in mind that the values of Δ will have to be normalized in order to relevantly define their linear combination with the gradient values. In particular, the highest value of Δ (that should be set to 1 in the normalization process) is obtained at the points located on $(\delta\mathbb{Z} + \frac{\delta}{2})^n$ if δ is even and $(\delta\mathbb{Z} + \frac{\delta-1}{2})^n$ if δ is odd.

3.2 Gradient Computation

The distance map $\Delta(M)$ has to be combined with a gradient map or, more precisely, a map ∇I providing the norm of the gradient of the image I. The way of building the gradient on a discrete image is not unique. In practice, it mainly depends on the composition of the discrete approximation of the derivatives of I along the n orthonormal vectors e_i ($i \in [\![1, n]\!]$) of the canonical basis of $\mathbb{Z}^n$. Basically, such derivative in dimension i is generally approximated at x by a finite difference

$$\nabla_i I(x) = \|I(x) - I(x + e_i)\|, \tag{8}$$

where $\|\cdot\|$ is the norm that endows the normed vector space $\mathbb{V}$. Discrete versions of the norm of the gradient can then be designed by combining these n derivatives obtained in each dimension. In other words, we simply have

$$\nabla I(x) = \|(\nabla_i I(x))_{i=1}^n\|_k \tag{9}$$

This way of computing the norm of the gradient is consistent with any dimension n.

Here again, it is important to keep in mind that the gradient map will have to be normalized in order to be combined with the distance map. In particular, the maximal value of $\nabla I(x)$ is $\|(\max_{u,v \in \mathbb{V}} \|u - v\|)_{i=1}^n\|_k$, and should be set to 1 in the normalized gradient map.

In the case where $\mathbb{V}$ has specific properties, for instance when $\mathbb{V} = \mathbb{R}$ or $\mathbb{Z}$, alternative strategies may also be considered for defining gradient heuristics, for instance via morphological operators.

3.3 Seeded Watershed and Seeds Definition

There exist various kinds of watershed algorithms. In particular, the first ones proposed in the literature provided as output a partial partition of the discrete image, since they defined watersheds—i.e. frontiers between the catchment basins—by subsets of pixels/voxels. Recent algorithms (see [9, Chapters 3, 9] and references therein) now define interpixel/voxel watersheds, by considering the image support via the non-oriented graph modeling its topological structure. Under these hypotheses, the watershed transformation is formulated as a graph-cut problem that consists of defining a partition of the vertices (i.e. the pixels/voxels) of the graph, by the dual removal of edges (i.e. the interpixel/voxel adjacency links located on the watersheds).

Here, we rely on such a graph-based paradigm. This requires to define an adjacency relation on Ω, and more generally on $\mathbb{Z}^n$. The framework of digital topology provides us with standard adjacency relations, for instance the $(2n)$- and (3^n-1)-adjacencies that rely on the ℓ_1 and ℓ_∞ norms, and that lead to the well-known 4- and 8-adjacencies in $\mathbb{Z}^2$ and the 6- and 26-adjacencies in $\mathbb{Z}^3$. It is important to avoid adjacencies that could lead to spatially non-coherent results. For instance, the (3^n-1)-adjacency could lead to intersections between regions, in violation of the Jordan theorem. Considering the $(2n)$-adjacency, i.e. following the policy of well-composedness [3] is then a safe and easy strategy. Other solutions, proposed e.g. by perfect fusion grids [4] or regular images [10] may be alternatively considered.

The watershed transformation has the property of providing oversegmented results. Indeed, in its basic version, any point corresponding to a locally minimal value within the graph leads to a distinct catchment basin. In other words, the cardinality of the partition is equal to the number of local minima in the processed saliency map, with an oversized result in case of noisy input.

In the case of the watervoxels, we wish to obtain the same number of regions within the partition as the number of markers of M initially defined. In addition, for the sake of geometrical and topological coherence, these regions should have positions, size and relationships similar to the ones of the Voronoi tiling of Ω induced by the markers of M. Depending on the application, we can either use the set of markers M as set of seeds, or determine an alternative set of seeds S. (In particular, one may note that the markers of M may not be local minima of the saliency map.)

It is important to choose, for each marker m of M a seed point s of S such that s is located in the Voronoi cell of m, and not too close from the border of this cell, in order to avoid that many seeds be connected, which may yield undesired border effects. To tackle this issue, it is possible to constrain s to be at least at a given distance from the border of the Voronoi cell, by imposing a security distance (see Algorithm 2). Two strategies can be considered for defining s. First, one can define s, within the Voronoi cell, as the regional minimum for the saliency map $\alpha \cdot \nabla I + (1-\alpha) \cdot \Delta(M)$ located the closest to m. Second, one can choose s among the regional minima of the gradient map ∇I within the central part of the Voronoi cell [7]. Note that in this second case, it is also possible to recompute a distance map Δ_S from these new seed points. This new distance map can then substituted to Δ in the linear combination, leading to the so-called m-marker strategy proposed in [7].

3.4 Trade-off Parameter

The trade-off parameter $\alpha \in [0,1]$ allows one to control the influence of the gradient map versus the distance map, in the linear combination forming the saliency map involved in the watershed transformation process.

Following Equation (5), the higher α, the higher the importance of the gradient over the distance map. In the extreme cases, for $\alpha = 0$, we only consider the distance map, and the watervoxel partitioning will exactly map the Voronoi tiling of the support Ω with respect to the markers M. By contrast, for $\alpha = 1$, we only consider the gradient map, and the watervoxel partitioning is exactly

the seeded watershed transformation induced by the set of seeds S.

4 Implementation

4.1 Description

The proposed implementation of the watervoxel algorithm is based on the following successive steps:

1. Image loading and storage in a Numpy array. The supported image formats are NIFTI, JPEG and PNG.
2. Computation of the gradient norm of the image.
3. Definition of the markers and computation of the distance map.
4. Computation of the linear combination of the distance map and the gradient:

$$\alpha \cdot image\_gradient + (1 - \alpha) \cdot distance\_map \tag{10}$$

5. Computation of the markers.
6. Computation of the marker-based watershed on the linear combination of the gradient norm and of the distance map.
7. Result saving as NIFTI (for 3D images) or PNG (for 2D images).

4.2 Requirements

The proposed implementation depends on various Python libraries (see `requirements.txt` file in the provided source code). These libraries are:

- nibabel (2.3.1), to read and write the NIFTI files.
- numpy (1.15.4), to perform matrix computation.
- scipy (1.1.0), to compute the gradient.
- scikit-image (0.14.1), to compute the watershed and to read and write 2D images.
- matplotlib (3.0.2), to plot the results.

The README file explains how to install the packages required to fulfill these dependencies in a virtualenv, parsing the file `requirements.txt` with pip.

4.3 Inputs

The proposed implementation requires the following input information:

- The path of the image to be computed.
- The size of the supervoxels (i.e. δ).
- The trade-off parameter (i.e. α) for the linear combination of distance map and gradient.

- The gradient type to be used (magnitude gradient, morphological gradient, dilation-image, image-erosion, inverted_image, inverted_plus_erosion).
- The security distance (in case of use of the second strategy; see Section 3.3).
- The result file name.
- The structuring element (i.e. the shape of the structuring element which is used if we compute a morphological gradient).
- The type of distance map (Euclidean, Manhattan, Tchebychev).
- The m-marker Boolean parameter which determines whether we use or not the m-marker recomputed distance map.

4.4 Pseudo-code

We provide, in Algorithm 1, the pseudo-code for the whole watervoxel computation process. In particular, it summarizes the discussion of Section 3.

In Algorithm 2, we provide the pseudo-code for the computation of markers.

For each Voronoi cell, we set a label to the greatest area with minimum value and set the remainder to zero. In the first case, the minimal area is searched in the linear combination of the gradient and the distance map. In the second case, we set a security distance on the edges of the cell and we search the minimum of the gradient only (see Section 3.3).

5 Results

In this section, we illustrate the behavior of the watervoxels. In particular, we compare them versus results obtained with SLIC. Our purpose here is not to carry out a comparative study between both. (In the 2-dimensional case, a comparative study was already proposed in [7].) Once again, our purpose is not to claim the superiority of one superpixel paradigm over the other, but to emphasize their behaviour differences. We recall, however, that the watervoxel method runs in guaranteed linear time.

5.1 2-dimensional Example

We first compare the two superpixel approaches on a 2-dimensional color image; see Figure 1 (the same image was involved in illustration examples in [7]). The dimensions of this image are 321 $\times$ 481 pixels.

For the watervoxel computation: the distance δ is 9 pixels and we set α to 0.7 which means that the gradient information is $\simeq 2.5$ times stronger than the distance map information. The markers are computed using, on the one hand, the first strategy, i.e. as the minima of the combined map (see Section 3.3); and on the other hand the m-marker strategy, i.e. as the minima of the gradient map with a further recomputed distance map.

For the SLIC computation, we used the Scipy implementation. We consider a similar segmentation, i.e. $\frac{321\times481}{9^2} \simeq 1900$, and a compactness equal to 100.

One can observe that both SLIC and classical waterpixel results exhibit a globally similar result. Slight differences can be observed, however. For instance, one can see that the frontiers of superpixels that are adjacent to a strong gradient are completely modified in SLIC, whereas those with waterpixels have their boundary modified only on the strong gradient areas. In other words, the waterpixels

Algorithm 1: Watervoxel computation (Watershed.py)

Input:
δ: distance between seeds/supervoxel size
α: weighting coefficient
grad: gradient function
secure_dist: security distance
dist_type: distance type
m_marker: Boolean *m*-marker parameter

begin

$matrix = load\_image(I)$
$grad\_norm = ||grad(matrix)||$ N.B.: we normalize the gradient norm between 0 and 100

N.B.: if dimension is greater than one of the image dimensions, δ = min dimension
if $\delta > min(dimensions(I))$ **then**
  $\delta = min(dimensions(I))$
$distance\_map = compute\_dist\_map(I, \delta, dist\_type)$ N.B.: we normalize the distance map between 0 and 100

$comb = \alpha * grad\_norm + (1 - \alpha) * distance\_map$
if $secure\_dist > 0$ **then**
  $markers = Markers(grad\_norm, \delta, secure\_dist)$
  if *m_marker is True* **then**
    $distance\_map = compute\_dist\_map(markers, \delta, dist\_type)$
    $comb = \alpha * grad\_norm + (1 - \alpha) * distance\_map$
else
  $markers = Markers(comb, \delta, secure\_dist)$ N.B.: See Algorithm 2
$Result = Watershed(comb, markers)$
$Save\_result\_as\_nifti(Result)$

Algorithm 2: Markers computation (Markers.py)

Input:
δ: distance between seeds/supervoxel size
secure_dist: security distance
I: Image
begin
  Decomposition of the image in Voronoi cells (depending on δ)
  $Marker\_label = 0$ N.B.: Initialization of the marker label number
  N.B.: (x_lim, y_lim, z_lim) are the front upper left corner coordinates of a cubical Voronoi cell
  for $x\_lim = indice\_x\_min$ **to** $indice\_x\_max$ **do**
    for $y\_lim = indice\_y\_min$ **to** $indice\_y\_max$ **do**
      for $z\_lim = indice\_z\_min$ **to** $indice\_z\_max$ **do**

$$thumbnail = I[x\_lim + secure\_dist : (x\_lim + \delta) - secure\_dist,$$
$$y\_lim + secure\_dist : (y\_lim + \delta) - secure\_dist,$$
$$z\_lim + secure\_dist : (z\_lim + \delta) - secure\_dist]$$

        $ind = where(min(thumbnail))$ N.B.: flat zone of minimal value and of greatest area in the Voronoi cell

        $Markers[tuple(ind)] = Marker\_label$
        $Marker\_label = Marker\_label + 1$
  return $Markers$

remain more regular (or, equivalently, more dependent to the underlying distance map structure), whereas the SLIC superpixels experiment data-driven side effects. By contrast, the waterpixels with m-markers lead to a much less structured superpixel partitioning, that better captures fine gradient details (see, e.g. cloud regions), but with a less regular behaviour in actually constant areas (see, e.g. water regions). Depending on the targeted application, either one or the other behaviour may be prefered.

5.2 3-dimensional Example

In a second example, we present supervoxel partitioning results on a 3-dimensional image, namely a magnetic resonance image (MRI) of a human brain.

The dimensions of this image are $240 \times 320 \times 32$ voxels, and a binary mask was applied in order to remove the extracranial part (in other words, the background has a homogeneous null value).

For the watervoxel computation: the distance δ is set to 9 voxels and α is set to 0.8, which means the gradient information is 4 times stronger than the distance map information. The markers map is computed using the first strategy (see Section 3.3).

For the SLIC computation, we used the Scipy implementation. We consider similar segments, i.e. $\frac{240\times320\times32}{9^3} \simeq 3400$, and a compactness equal to 100.

Based on a visual inspection, we observe that the classical watervoxels have a satisfactory behavior, with a good fitting on the salient edges between different kinds of tissues, and a regular shape in homogeneous areas. The m-marker version of watervoxels exhibits the same properties already observed in the 2D case.

Acknowledgements

The research leading to these results has been supported by the ANR MAIA project, grant ANR-15-CE23-0009 of the French National Research Agency (http://recherche.imt-atlantique.fr/maia) and the American Memorial Hospital Foundation.

Image credits

- Figure 1: The Berkeley Segmentation Dataset and Benchmark [8].
- Figure 2: Epirmex dataset. Epirmex is part of the French epidemiologic study Epipage 2[2].

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, *SLIC superpixels compared to state-of-the-art superpixel methods*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 34 (2012), pp. 2274–2282. https://doi.org/10.1109/TPAMI.2012.120.

[2] S. Beucher and C. Lantuéjoul, *Use of watersheds in contour detection*, in International Workshop on Image Processing Real-Time Edge Motion Detection/Estimation, Proceedings, 1979, pp. 17–21.

[2] http://epipage2.inserm.fr

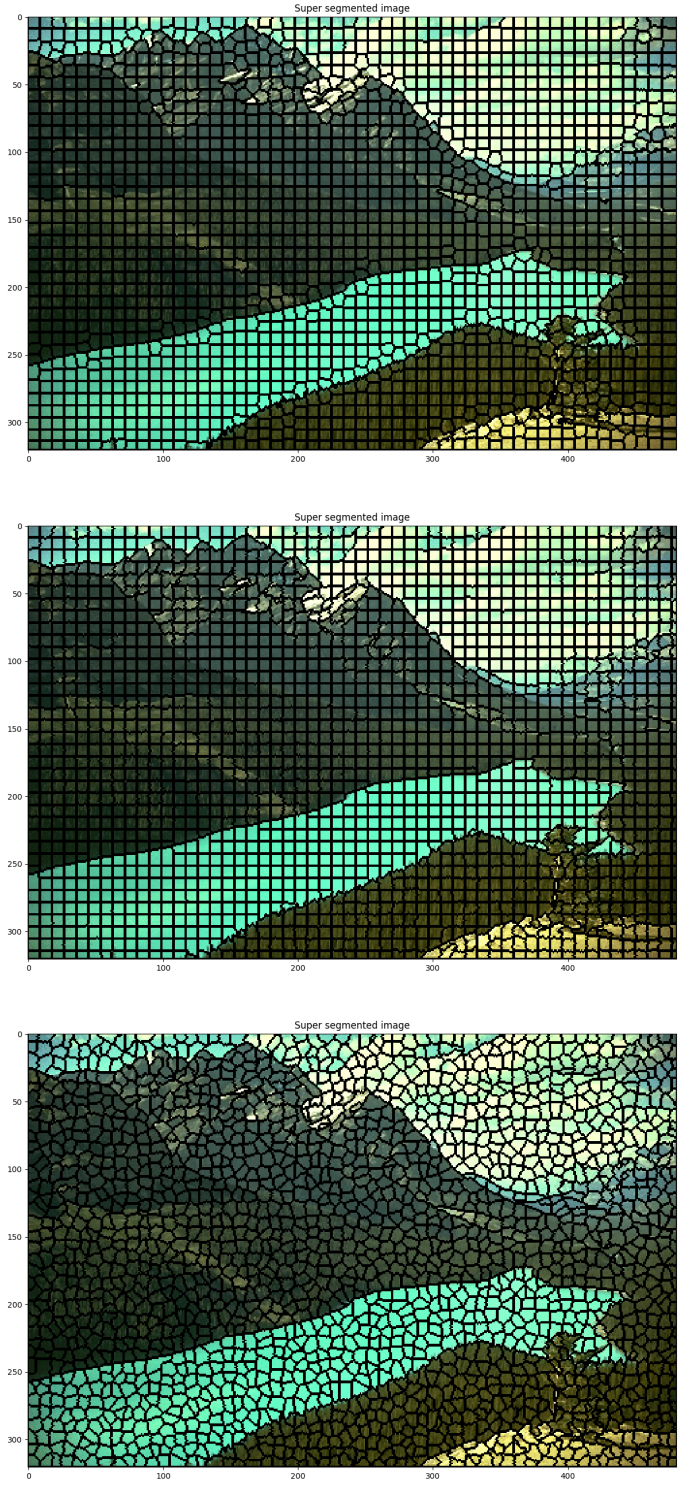

Figure 1: Superpixel partitioning on a 2-dimensional image. Top: with the SLIC superpixel algorithm. Middle: with classical waterpixels. Bottom: with waterpixels and the m-marker recomputed distance map.

[3] N. Boutry, T. Géraud, and Najman L., *A tutorial on well-composedness*, Journal of Mathematical Imaging and Vision, 60 (2018), pp. 443–478. https://doi.org/10.1007/s10851-017-0769-6.

[4] J. Cousty, G. Bertrand, M. Couprie, and L. Najman, *Fusion graphs: Merging prop-*

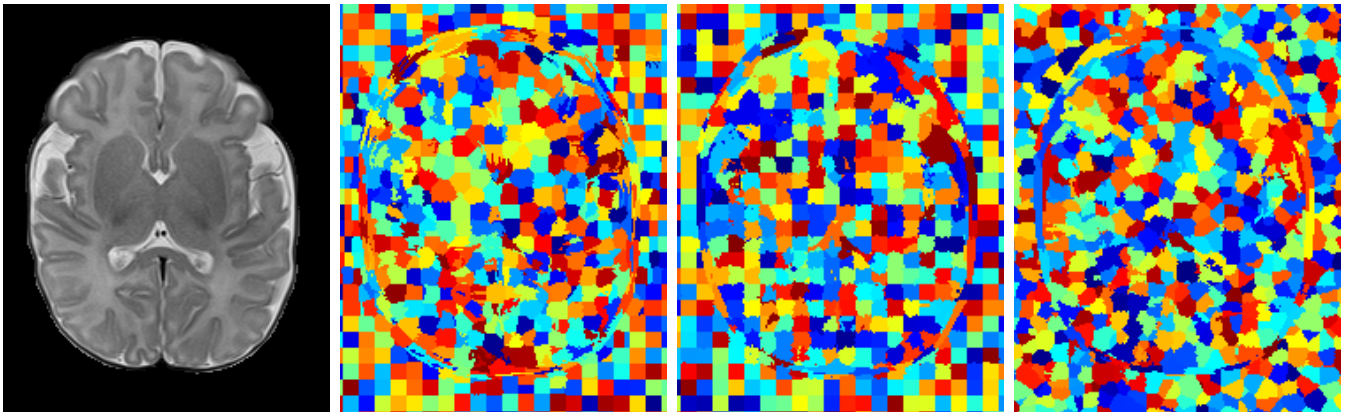

Figure 2: From left to right: input image; SLIC superpixel partitioning; classical watervoxel partitioning; and watervoxel partitioning with the m-marker recomputed distance map. Axial slices of the 3D image volume.

erties and watersheds, Journal of Mathematical Imaging and Vision, 30 (2008), pp. 87–104. https://doi.org/10.1007/s10851-007-0047-0.

[5] T. Y. Kong and A. Rosenfeld, *Digital topology: Introduction and survey*, Computer Vision, Graphics, and Image Processing, 48 (1989), pp. 357–393. https://doi.org/10.1016/0734-189X(89)90147-3.

[6] V. Machairas, E. Decencière, and T. Walter, *Waterpixels: Superpixels based on the watershed transformation*, in International Conference on Image Processing (ICIP), Proceedings, 2014, pp. 4343–4347. https://doi.org/10.1109/ICIP.2014.7025882.

[7] V. Machairas, M. Faessel, D. Cardenas-Pena, T. Chabardès, T. Walter, and E. Decencière, *Waterpixels*, IEEE Transactions on Image Processing, 24 (2015), pp. 3707–3716. https://doi.org/10.1109/TIP.2015.2451011.

[8] D. Martin, C. Fowlkes, D. Tal, and J. Malik, *A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics*, in International Conference on Computer Vision (ICCV), 2001, pp. 416–423. https://doi.org/10.1109/ICCV.2001.937655.

[9] L. Najman and H. Talbot, eds., *Mathematical Morphology: From Theory to Applications*, ISTE / J. Wiley & Sons, 2010. https://doi.org/10.1002/9781118600788.

[10] P. Ngo, N. Passat, Y. Kenmochi, and H. Talbot, *Topology-preserving rigid transformation of 2D digital images*, IEEE Transactions on Image Processing, 23 (2014), pp. 885–897. https://doi.org/10.1109/TIP.2013.2295751.

[11] X. Ren and J. Malik, *Learning a classification model for segmentation*, in International Conference on Computer Vision (ICCV), 2003, pp. 10–17. https://doi.org/10.1109/ICCV.2003.1238308.

[12] R. Strand and G. Borgefors, *Distance transforms for three-dimensional grids with non-cubic voxels*, Computer Vision and Image Understanding, 100 (2005), pp. 294–311. https://doi.org/10.1016/j.cviu.2005.04.006.

[13] L. Vincent and P. Soille, *Watersheds in digital spaces: An efficient algorithm based on immersion simulations*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 13 (1991), pp. 583–598. https://doi.org/10.1109/34.87344.