Waterpixels Generation

Chems Eddine Benaziza

Supervised by Christophe Kervazo

March 1, 2024

Abstract

Superpixels serve as a crucial preprocessing step in image segmentation techniques. This paper elucidates the implementation pipeline for generating water pixels through the watershed transform, inspired by prior work on water pixels [2]. The document comprehensively details the entire implementation process. Moreover, in the conclusion, alternative methods are proposed to enhance the computational efficiency of the calculations.

1 Introduction

1.1 Superpixels

Definition

SUPERPIXELS represent coherent regions obtained through a low-level segmentation process applied to an image. These regions serve as fundamental building blocks for more advanced analysis, including tasks such as object detection, segmentation, and classification.

In the context of this paper, our investigation focuses on waterpixels as a method for generating superpixels. Waterpixels offer an effective approach to achieve this segmentation and enhance the subsequent analysis of images.

Recognizing that superpixels play a pivotal role as tools for downstream applications, our objective is to ensure that the generation process is **not overly time-consuming** or **memory-intensive**. Therefore, we carefully consider the time and space complexity associated with superpixel generation.

1.2 Properties

The superpixels we want to generate should have the following properties:

1. **Homogeneity**: pixels of a given SP should present similar colors or gray levels
2. **Connected partition**: each SP is made of a single connected component and SPs constitute a partition of the image
3. **Adherence to object boundaries**: object boundaries should be included in the superpixels boundaries
4. **Regularity**: SPs should form a regular pattern on the image. This property is often desirable as it makes the SP more convenient to use for subsequent analysis steps.

Remark One of the main criteria of a good SP generation algorithm is the complexity of the method, because superpixels are tools for further analysis, it should not take too long nor too much memory.

Goal: The main goal of this paper is thus to follow steps in the reference [2] paper and show how we could implement them in Python. A detailed Jupyter notebook comes with this paper.

1.3 Making Superpixels

To get the pixels we have two main steps:

1 Choosing the seeds

2 Building superpixels

In the first step, a set of seeds is chosen, which are typically spaced regularly over the image plane and which can be either regions or single pixels.

Type A: Image independent, the center of a regular grid

Type B: Image dependent, they depend on the content, it takes time to find the good seeds

Type C: Image independent initially, iteratively refined to take into account the image

1.4 Building Superpixels

Two main methods: There are two main methods in superpixels generation.

Shortest path : these methods are based on region growing: they start from a set of seeds (points or regions) and successively extend them by incorporating pixels in their neighborhood according to a usual image-dependent cost function until every pixel of the image plane has been assigned to exactly one superpixel. This process may or may not be iterated.

Shortest Distance Methods : These are iterative procedures inspired by the field of unsupervised learning, where at each iteration step, seeds (such as centroids) are calculated from the previous partition, and pixels are then reassigned to the closest seed (like for example the k-means approach).

Although methods inspired by general clustering methods (type 2) seem appealing at first sight, they could lead to nonconnected superpixels which is undesirable. For type one solutions, based on region growing, implement a path-type distance, where the distance between two pixels does not only depend on value and position of the pixels themselves, but on values and positions along the path connecting them. Type 1 methods imply connected superpixels regions, for which the number of seeds is exactly the number of

1.5 Superpixels and Watershed

1 Good adherence to object boundaries when computed on the image gradient

2 It allows control of the number and spatial arrangement of the resulting regions through the choice of markers

3 Connectivity and no post-process is required

4 Linear with the number of pixels in the image

2 Implementation

2.1 Steps to Make Superpixels

Here are the steps detailed in the reference article, that assure a good time and space complexity.

1 Computation of the gradient of an image: a morphological gradient

2 Definition of regular cells on the image centered on the vertices of a regular grid; we choose cell centers in the grid

3 Selection of one marker per cell

4 Spatial regularization of the gradient with the help of a distance function

5 Application of the watershed transformation on the regularized gradient defined in step 4 from the markers defined in step 2

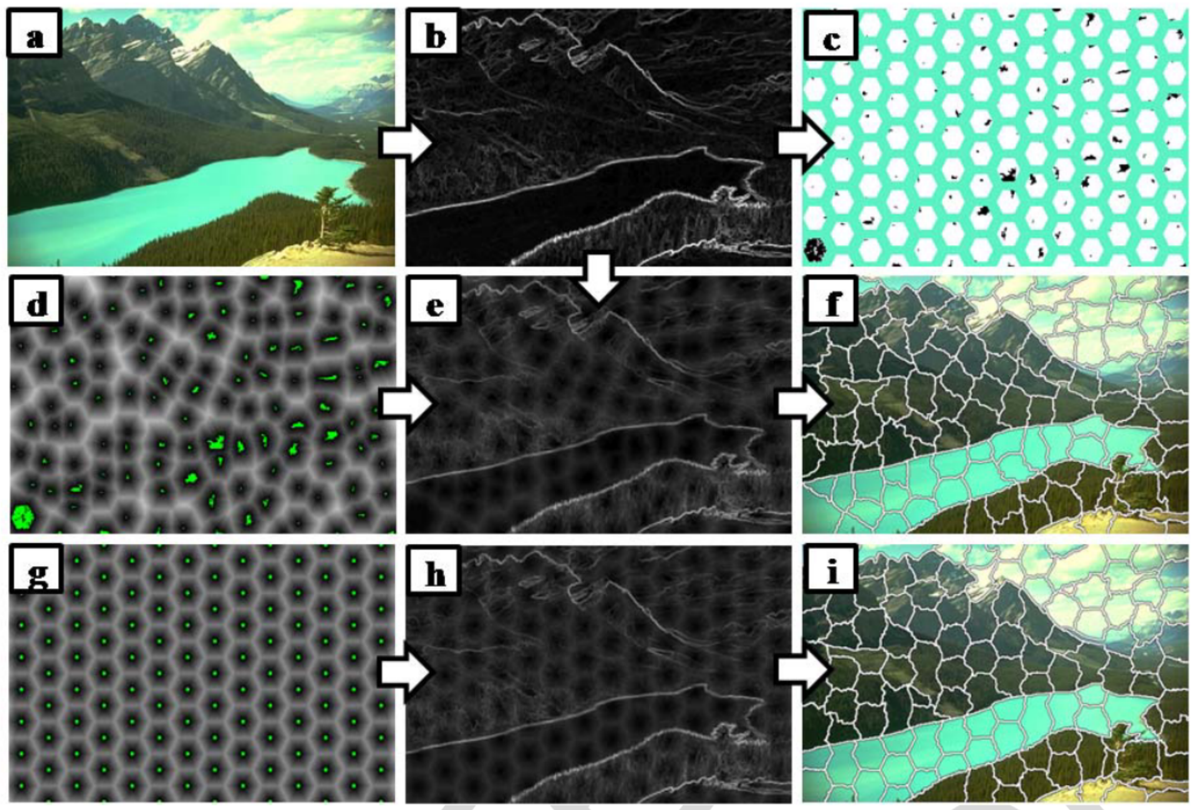

Fig. 2. Illustration of waterpixels generation: (a): original image; (b) corresponding Lab gradient; (c): selected markers within the regular grid of hexagonal cells (step $\sigma = 40$ pixels); (d): distance function to markers; (g): distance function to cell centers; (e) and (h): spatially regularized gradient respectively with distance functions to selected markers (d) and to cell centers (g); (f) and (i): Resulting waterpixels obtained by respectively applying the watershed transformation to (e) and (h), with markers (c).

Figure 1: Image from the reference paper

3 Implentation steps

3.1 Computation of the Gradient of an Image: A Morphological Gradient

Before calculating the gradient, we compute a little smoothing of the image. We can complete this step using a morphological opening and then closing.

$$Image_{smoothed} = Closing(Opening(Image)) \tag{1}$$

After that, we can convert the picture from a three-channel color to a one-gray level channel. After this, we calculate the morphological gradient through a erosion substracted from a dilation of the image.

$$G = dilation(f) - erosion(f) \tag{2}$$

f : Gray scale image
The gradient helps to choose the seeds for the superpixels which will be useful for the coming steps.

To sum up this step, we take as input a picture with three channels, and as output, we will have a gray scale **gradient image**.

3.2 Definition of Regular Cells on the Image

As each cell is meant to correspond to the generation of a unique water pixel, our method, through the choice of one marker per cell, offers total control over the number of SP, with a strong impact on their size and shape if desired.

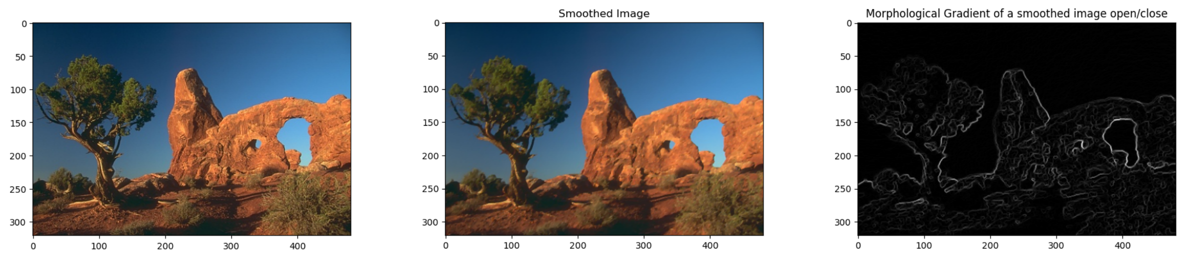

Figure 2: Original - smoothed - gradient of the image

To fix the seed, we compute the minima of the gradient g. When the minima is not unique, it could be a connected component. This can be imagined as a flat surface where all values are similar and represent a minimum. A method to find the gradient minima connected component was developed. But for simplification reasons, we opted for one marker per cell. We first tried using the center of cells. In both cases, either a connected component or a single marker (i.e., the center of the cell), there should be a unique component to ensure regularity and connectivity of the resulting superpixels.

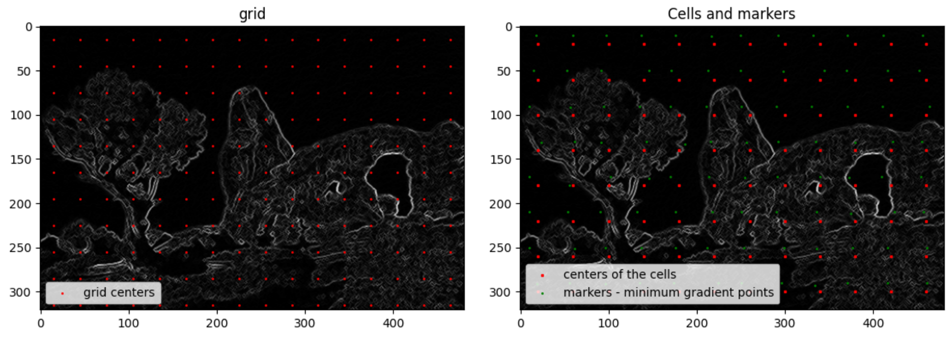

Figure 3: Left : grid centers, Right Grid markers as minimal points of the gradient

3.3 Selection of One Marker per Cell

The selection of markers has enforced the pertinence of future superpixel-boundaries but also the regularity of their pattern (by imposing only one marker per cell). In this paragraph, we discuss the importance of marker selection and its impact on the resulting superpixels.

3.4 Spatial Regularization of the Gradient with the Help of a Distance Function

In this section, we introduce a technique to regulate the gradient in a spatially coherent manner, balancing between boundary adherence and regularity.

Let $Q = \{q_i\}_{1 \leq i \leq N}$ represent a set of N connected components of the image f. For every $p \in D$, where D is the domain of the image, we define a distance function d_Q with respect to Q as follows:

$$\forall p \in d_Q(p) = \frac{2}{\sigma} \min_{i \in [1,N]} d(p, q_i) \tag{3}$$

Here, σ denotes the grid step defined previously. Normalizing by σ ensures that the regularization is independent of the chosen superpixel (SP) size. There are two choices for the markers q_i: either the markers $q_i = M_i$ or the centers for the cells. The former yields better adherence to object boundaries, while the latter produces more regular superpixels.

The spatially regularized gradient g_{reg} is defined as:

$$g_{\text{reg}} = g + k \cdot d_Q \tag{4}$$

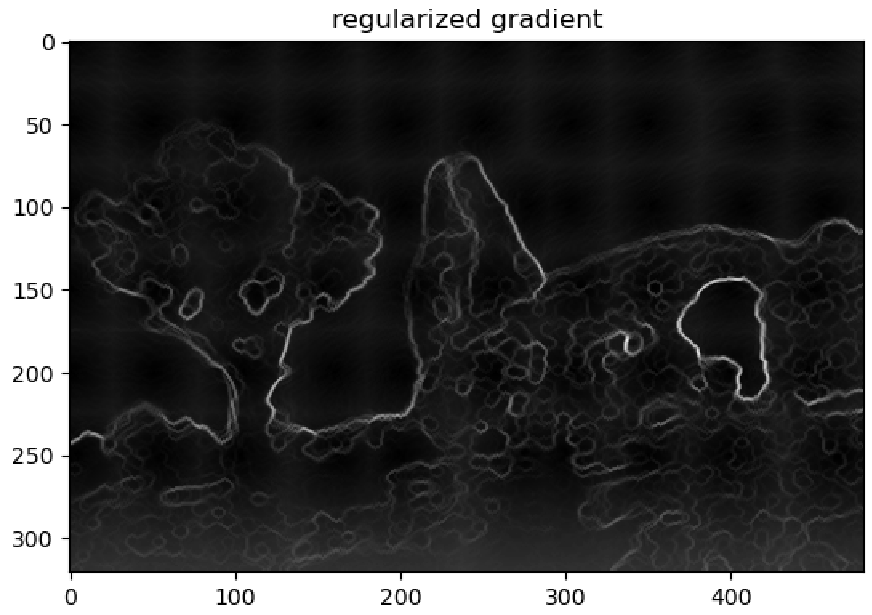

Figure 4: Regularized gradient - smooth grid in the background

Where:

- g represents the gradient of the image f,
- d_Q is the distance function defined above, and
- k is the spatial regularization parameter.

A smaller k implies less regularization. As k approaches infinity, we converge towards the Voronoi tessellation of the set $\{q_i\}_{1\leq i\leq N}$.

To summarize this step, we take the image gradient as input and produce a regularized gradient. In the process, we implement a function that calculates the distance between points in the image domain and a set of markers Q.

3.5 Application of the Watershed Transformation

In the final step of our superpixels generation implementation, we take the regularized gradient defined in step 4 from the markers defined in step 2 and apply the watershed transformation on the spatially regularized gradient g_{reg}, starting the flooding from the markers $\{M_i\}_{1\leq i\leq N}$, so that an image partition $\{s_i\}_{1\leq i\leq N}$ is obtained. The s_i are the resulting waterpixels.

4 Results and Conclusion

Finally, we were able to generate superpixels with a really good adherence to boundaries and regularity. To measure the performance, we shoudl recall the criteria of the superpixel generation that include boundary recall, contour density, and average mismatch factor, as well as computation time. Boundary recall is defined as the percentage of ground-truth contour pixels that fall within strictly less than 3 pixels from superpixel boundaries C, given by:

$$BR = \frac{|\{p \in GT, d(p,C) < 3\}|}{|GT|} \tag{5}$$

where d is the L_1 (or Manhattan) distance. Contour density is defined as the number of superpixel contour pixels divided by the total number of pixels in the image, given by:

$$CD = \frac{1}{2}\frac{|S_c| + |S_b|}{|D|} \tag{6}$$

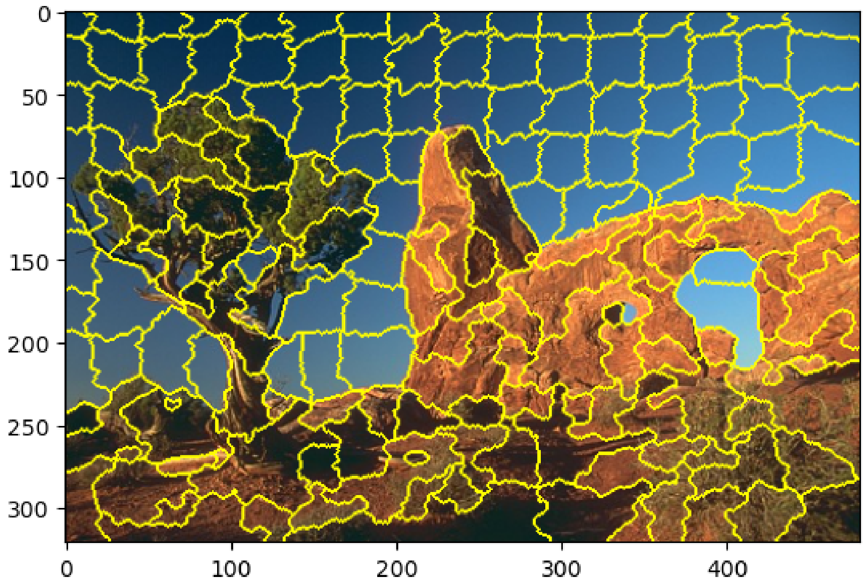

Figure 5: Final with an excellent adherence to boundaries

where S_c is the set of superpixel contour pixels, S_b is the set of one pixel wide image borders, and D is the set of all pixels in the image.

The average mismatch factor is defined as the average shape and size dissimilarity between superpixels, given by:

$$MF = \frac{1}{N} \sum_{i=1}^{N} mf(s_i^*, \hat{s^*})) \tag{7}$$

where s_i^* is the centered version of superpixel s_i, $\hat{s^*}$ is the average centered shape of all superpixels, and mf is the mismatch factor between two sets, defined as:

$$mf(A, B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|} \tag{8}$$

Compactness is sometimes used in superpixels evaluation, but it is a poor measurement for region regularity. The average mismatch factor is more appropriate to evaluate regularity than compactness.

These metrics were used in the reference paper to measure the performance of the proposed method compared to other algorithms like SLIC, but in some cases the waterpixels was faster.

4.1 Takeaway

- The implementation successfully generates superpixels with good adherence to boundaries and regularity.
- The calculation time is around 8 seconds, which can be improved for better computational efficiency using the SMIL [1] python library
- The implementation aligns with the expectations of the reference paper, providing room for improvements in computational efficiency.
- The simplified web app written in Python under the Streamlit framework makes it easy to see and interpret the results.

4.2 Conclusion

In conclusion, this paper presents an implementation pipeline for generating water pixels through the watershed transform, inspired by prior work on water pixels. The document provides a comprehensive detail of the entire implementation process, including the computation of the gradient of an image, definition of regular cells, selection of one marker per cell, spatial regularization of the gradient, and the application of the watershed transformation. The resulting superpixels exhibit good adherence to object boundaries and regularity, making them suitable for further image analysis tasks. However, the calculation time is around 8 seconds, which can be improved. Following recommendations from one of the authors of the reference paper, the use of the SMIL [1] python library is suggested for faster calculations. Future work could focus on optimizing the computational efficiency of the calculations while maintaining the quality of the generated superpixels.

5 Bonus

5.1 Web App

To make testing and parameter tuning easier, we have developed a web app using Streamlit. The app allows the user to upload an image, select the parameters for the algorithm, and visualize the resulting superpixels. The app can be run using the following command:

```
streamlit run app.py
```

5.2 Code and Libraries

The code for this project was written in Python 3.11 and requires the following libraries:

```
python, matplotlib, skimage, numpy, cv2
```

To run the code, an environment with these libraries installed is required. We recommend using Conda as an environment manager.

References

[1] Matthieu Faessel and Michel Bilodeau. Smil: Simple morphological image library. *Séminaire Performance et Généricité, LRDE*, 2014.

[2] Vaïa Machairas, Matthieu Faessel, David Cárdenas-Peña, Théodore Chabardes, Thomas Walter, and Etienne Decencière. Waterpixels. *IEEE Transactions on Image Processing*, 24(11):3707–3716, 2015.