A Study of Classic Image Segmentation Methods

Bruno Luiz Dias Alves de Castro
Artificial Intelligence and Cybersecurity (AIC) - E4
ESIEE Paris
Paris, France
bruno.castro@edu.esiee.fr

***Abstract*—This article summarises the work developed by the author *Bruno Castro* during his 3 month research project at ESIEE Paris.**
The work developed involves using classic image segmentation algorithms such as Felzenszwalb, Salience Map, Watershed and Random walks to classify their behavior regarding the hierarchical "quality" of it's results, when applied in the same image for different scales.

***Index Terms*—mathematical morphology, image segmentation, felzenszwalb, salience map, IFT, watershed, random walks**

I. Introduction

Image segmentation is a very well known and extensively studied topic in computer science. Scientists have been studied ways to segment images even before computers could display them. A good segmentation of an image leads us to better classifications, better object-background separation, etc.

A. Hierarchy in image segmentation

One other aspect that can be observed in different segmentation algorithms, is their behavior when the number of regions (or segments) increase. Applying the same algorithm twice in the same image with different scale parameters may result in a different segmentation. We say a result is "highly hierarchical" when the borders in the lower scaled image have a good alignment with the borders of the high scale image.
This doesn't directly impact the quality, a non-hierarchical result doesn't imply in bad segmentation, but in some applications hierarchy may be desirable.
For this study, two classic algorithms of the mathematical morphology field of image research were chosen, being them: Felzenszwalb and Watershed.

II. The segmentation methods

In this section, we present a quick overview of the methods used in their implementations.

A. Felzenszwalb's Algorithm

Representing a given image as a graph, one approach is tuning every pixel in a node, and every edge as the dissimilarity between 2 neighboring pixels. Felzenszwalb takes this approach in his algorithm. [1]

We start with a graph containing all pixels in our image, but no edges. If we say that every pixel is it's on component, our image starts with n * m components. It's possible two connect two components together, and form a new one. We use a value K to help in the decision of adding a edge to our graph, or not.

Now, ordering every edge of the image graph, we add this edge to our segmentation graph if it's value is higher than the weakest edge in the two components it connects, adjusted by a value K/size(component).

A demo of this algorithm's results is available in Fig. 1.

1) The implementation chosen: For this research project, the implementation chosen for the Felzenszwalb's Algorithm is the one present with the skimage library, for Python.

2) Binary search: Sadly, even though K is inversely proportional to the number of segments in the result, there is no known way of precisely setting a value K that gives us the exact number of segments. Because approximating a certain number os segments was necessary for this project, a binary search for a "good" value for K was implemented.

The way it works is simple: it starts with a small number of K, and run the algorithm. If the number of segments we got is higher than the target one, we increase K by a factor of 2, otherwise we decrease it by a factor of 0.75.

The method was implemented recursively, and has a maximum number of iterations, to prevent the algorithm to run indeterminately, as an exact number of segmentation may not be possible. We got good approximations in all the cases we tested.

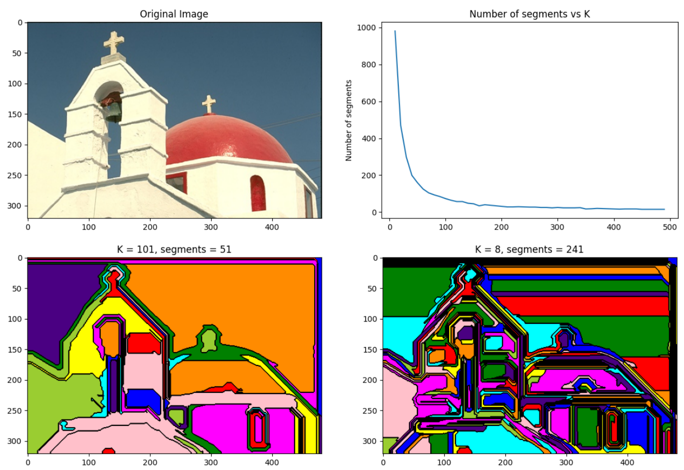


Fig. 1. Felzenszwalb result demo

B. Watershed

An image can be interpreted as a topological surface, where the intensity of each pixel determines it's "altitude" in said plane. In this interpretation, we can assume points with a high intensity corresponds to peaks in our surface, and the opposite is true for low intensity "valleys". [3]

One important field in geography is the study of water basins, how they form and their boundaries. The watershed algorithm brings us a similar approach in the study of these topological images. If we fill our surface with water, how does it behaves? When does two water bodies meet?

In watershed, we fill the valleys with water, and, to prevent two water bodies from merging, we build "barriers". Said "barriers" are the result of our algorithm, and they segment your image.

To control the number of segments generated by watershed, we directly need to control the number of "water sources" we use. For this, the algorithm takes as input a "seeds" file. They are our sources, and every seed will generate one segment in our final image.

A demo of this algorithm's results is available in Fig. 2.

1) Using the salience map algorithm to generate seeds: In this project, we needed a reliable way of generating seeds for our watershed algorithm. There are a multiple ways that could be achieved, such as manually choosing them, or by thresholding the image, but we decided to use another algorithm.

The saliency map algorithm divides an image on "areas of interest", ranking them from 0 to M, where M is the number of areas encountered. Using the implementation by prof. Jean Cousty (sm library), developed in C, we generate a saliency map for every image, select the last N segments of the result, that is, areas labeled (M-N) to M, and use it as input for the watershed algorithm.

2) Watershed implementation: The implementation chose for this project was the library by prof. Falcão, ift-demo, developed in C. It contains a variety of methods developed using the Image Foresting Transform (IFT), one of them being the watershed algorithm. [2]

This implementation takes two inputs: a seeds file (generated using the saliency map) and a gradient file (used to calculate the distance between the pixels in our graph). For this last one we used a gradient implementation available in the same ift-demo library.

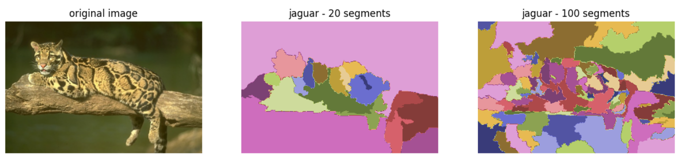


Fig. 2. Watershed result demo

C. Random Walks

The random walks algorithm works in a similar manner to the watershed one. Starting from a set of makers, neighboring pixels are visited accordingly to a equation.

For the project, the implementation available in the skimage lybrary, in Python, was chosen.

1) Using the salience map algorithm to generate markers: In order to execute it, the algorithm require a set of markers to start from. As we did fr the watershed, these markers we also extracted using a saliency map, using the implementation in C by prof. Jean Cousty.

The same seeds used in the watershed algorithm were used as markers in the random walks.

A demo of this algorithm's results is available in Fig. 3.

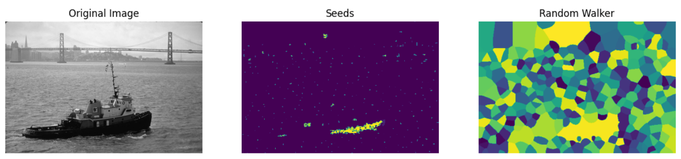


Fig. 3. Random walks result demo

III. Methodology

The main goals of this research project are:

- get to know classic methods of the image segmentation field.
- study how each of them work.
- find a implementation (if it exists).
- compare the results of each of them, regarding the hierarchical "quality".

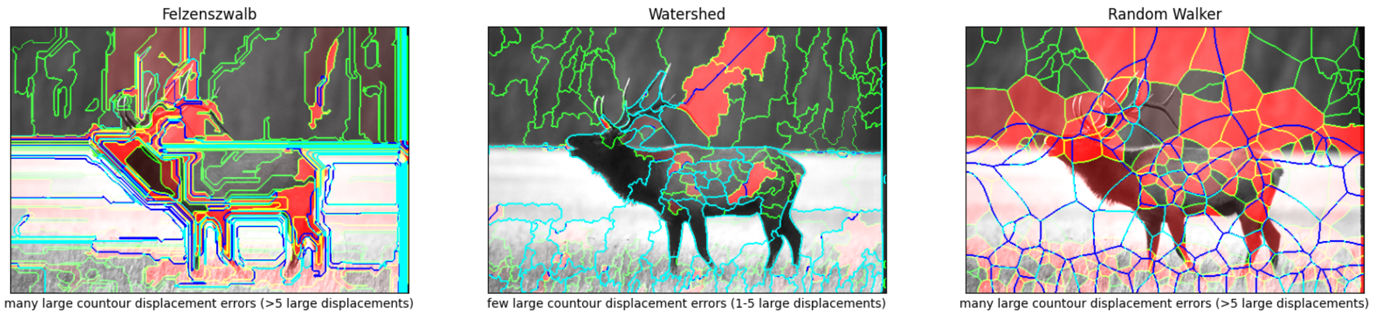
In order to compare the results, we came up with a classification system, explained in the next section.

A. The visualisation methods

In order to analyse if a result has a great hierarchical quality, we need to overlay two segmentation, one higher than the other, on top of each other. If a result is completely hierarchical, all borders in the lower segmented image should lay on top of the higher segmented one. This is almost never the case (at least for the methods studied), so we needed a way to classify is the results in different categories.

When analysing a given results, looking at the cells in the higher segmented image, a border from the lower segmentation can have two behaviors: it either aligns with it's border, or it shifts. Knowing this, two methods were developed to help with the visualization.

1) Cell error: The first method, pair cells from the higher segmented image to a cell in the lower one, given the intersection of both. After pairing cells with the highest intersection to each other the "error" of a cell is determined by the area that doesn't intersects it's father.


Felzenszwalb
many large countour displacement errors (>5 large displacements)
Watershed
few large countour displacement errors (1-5 large displacements)
Random Walker
many large countour displacement errors (>5 large displacements)

Summing up, the algorithm works like this:

- for each cell in the higher segmented image, determine it's father in the higher segmentation as being the cell with the highest intersection to it.
- calculate it's error as being the area that don't intersect it's father.
- color each cell in red, proportionally to it's error.

2) Border overlay: This method work by overlaying to segmentation on top of each other. The higher segmentation is given the color green, and the lowest a light blue. When two border lay on top of each other, the light blue color stays on top. If they don't align, a dark blue color appears instead.

The more dark blue color an image has, the more the borders of both segmentation's don't align.

B. Classifying the images

Knowing that, for each cell in the higher segmentation, it's father's borders either align or shifts, we came up with 4 classes to classify the results, being:

- no shifts (all cells aligns with theirs father's).
- only small shifts (no considerable shifts were observed).
- a few large shifts (between 1 and 10 considerable shifts happened).
- many large shifts (more than 10 considerable shifts are present).

After getting the visualisations described in the section above ready for all the sample images in the three studied algorithms, all images were manually classified. The results are discussed next.

IV. Results

At first sight, we could right away tell what algorithms performed the best.

A. Segmentation quality

Analysing only the quality of the segmentation, Felzenszwalb's algorithm, does not deliver good results. It tends to generate straight lines, and over segment some areas.

The watershed combined with the seeds generated by the salience map, is by far the algorithm that performed the best. Areas are in general well segmented, with performance dropping in low frequency areas.

Regarding the random walks algorithm, even though the seeds from the salience map performed great when combined with watershed, the same is not true for this one. It missed most of the contours of the images tested, and the overall segmentation quality is by far the worst among the methods tested.

B. Hierarchical quality

After classifying all the images, some thing were clear. First, in the test set of 25 images used, no algorithm was able to produce a 100% hierarchical result. That means, no image got the ”no shifts” classification.

By far the algorithm that performed the better was watershed with the salience map seeds, with most results falling in the ”only small shifts” or ”a few large shifts” categories.

The Felzenszwalb's Algorithm did present some results in the ”few large shifts” category, but mostly the results fell in the ”many large shifts”.

Random walks performed the worst overall, with all results falling in the ”many large shifts” category.

A table containing a summary for all the classifications for each method is available below.

Class	FS	WS	RW	Total
no shifts	0	0	0	0
only small shifts	0	10	0	10
large shifts	5	12	0	17
many large shifts	20	3	25	48

V. Conclusion

During these 3 months, I've learned a lot of skills that will help me in my professional and academic careers. I had a taste of how to conduct a research project, learn new skills, read articles and develop solutions to real problems.

Having the opportunity to work with specialists in the field was great for my personal growth, and I'm definitely more capable now than I was during the start of this project.

The results I've got even though are now ground-breaking or revolutionary, mean a lot to me as a student. They prove I was able to overcome the challenges I faced, and serve to me as a great introduction to this incredible field.

Hopefully in the future I can continue with my studies, going even further and developing/researching my own solutions.

References

[1] Felzenszwalb, Pedro F., and Daniel P. Huttenlocher. ”Efficient graph-based image segmentation.” International journal of computer vision 59 (2004): 167-181.

[2] Falcao, Alexandre X., Jorge Stolfi, and Roberto de Alencar Lotufo. ”The image foresting transform: Theory, algorithms, and applications.” IEEE transactions on pattern analysis and machine intelligence 26.1 (2004): 19-29.

[3] Belém, Felipe de C., et al. ”Impacts of contour saliency map transformations.”