Superpixel segmentation

Student: **Ciurtea Virgil-Octavian**, gr. E\_30434

Supervisor: Asist.prof. Robert VARGA

03-06-2019

# Introduction

* The concept of superpixel was first introduced by Xiaofeng Ren and Jitendra Malik in 2003. Superpixel is a group of connected pixels with similar colors or gray levels. Superpixel segmentation is dividing an image into hundreds of non-overlapping superpixels. Instead of working with just pixels, Ren and Malik use superpixels to do image segmentation.
* There are two major advantages for using superpixels:

1. You can compute features on more meaningful regions;
2. You can reduce the input entities for the subsequent algorithms.

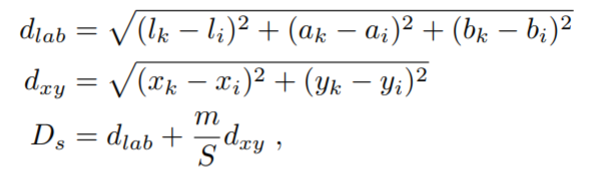
* The Superpixel method is used in Digital Image Processing context, which is the use of computer algorithms to perform image processing on digital images.
* I chose this project because when the algorithm is applied in real life situations, it can save a lot of time performing hard tasks like: finding a cancer cell with a specific shape within millions of red cells, face features recognition, visual place recognition in changing environments, image classification and so on. Moreover, the SLIC algorithm is very simple and is extremely useful.

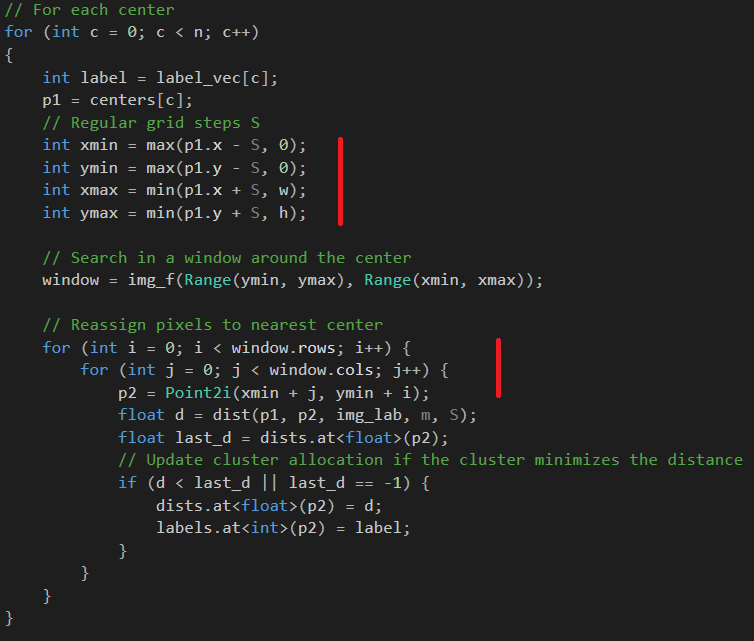
# State of the art

* The method I will be using for this project is the SLIC algorithm, which is one of the most efficient algorithms. There are a lot of other implementation and state of the art solutions presented in the SLIC Superpixels Compared to State-of-the-art Superpixel Methods article, which are split into 2 categories: graph-based and gradient-ascent-based. I will present one method from each of the categories.
  + From graph-based methods, we have NC05, the Normalized cuts algorithm recursively partitions a graph of all pixels in the image using contour and texture cues, globally minimizing a cost function defined on the edges at the partition boundaries. It produces very regular, visually pleasing superpixels. However, the boundary adherence of NC05 is relatively poor and it is the slowest among the methods (particularly for large images), although attempts to speed up the algorithm exist. NC05 has a complexity of O(), where N is the number of pixels.
  + From gradient-ascent-based methods, we have MS02, mean shift, an iterative mode-seeking procedure for locating local maxima of a density function, is applied to find modes in the color or intensity feature space of an image. Pixels that converge to the same mode define the superpixels. MS02 is an older approach, producing irregularly shaped superpixels of non-uniform size. It is O() complex, making it relatively slow and does not offer direct control over the amount, size, or compactness of superpixels.

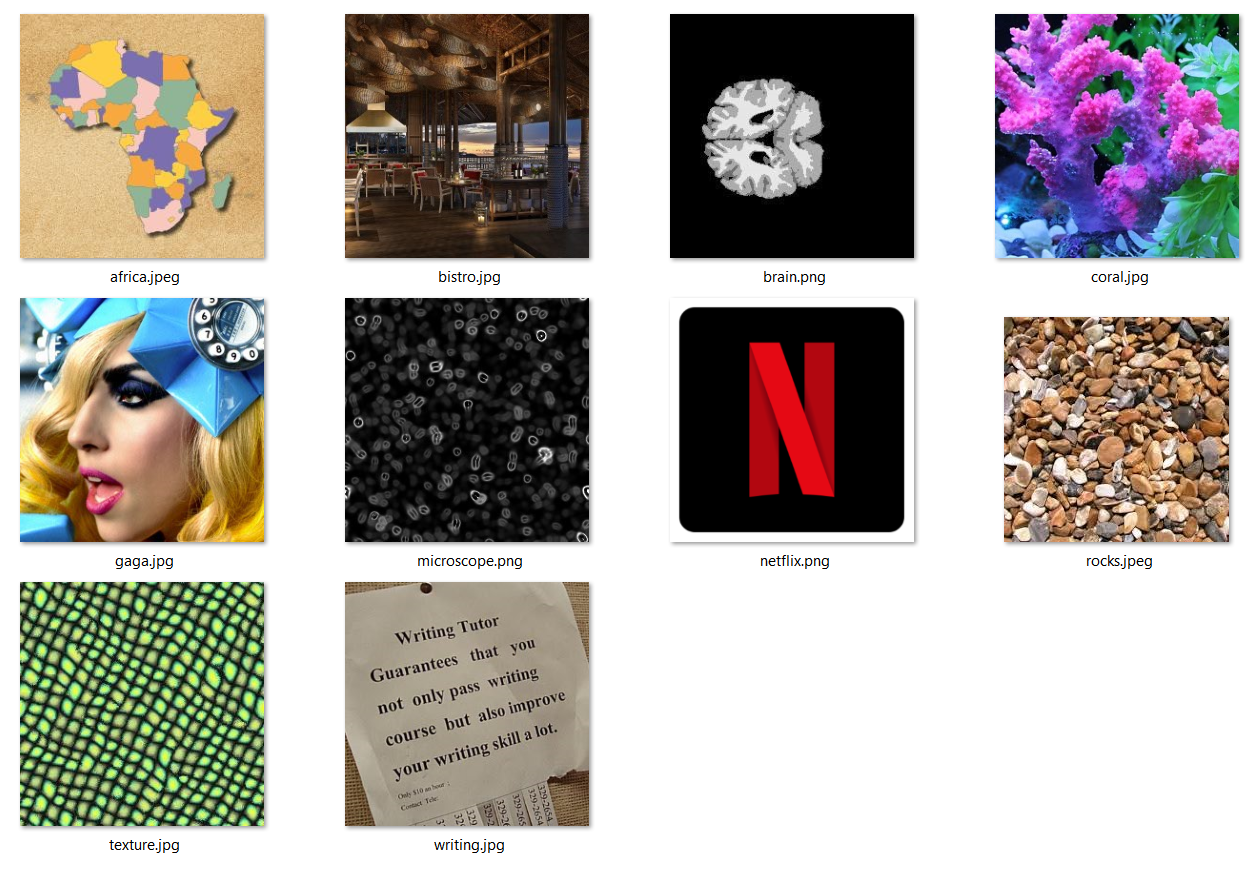
# Proposed method



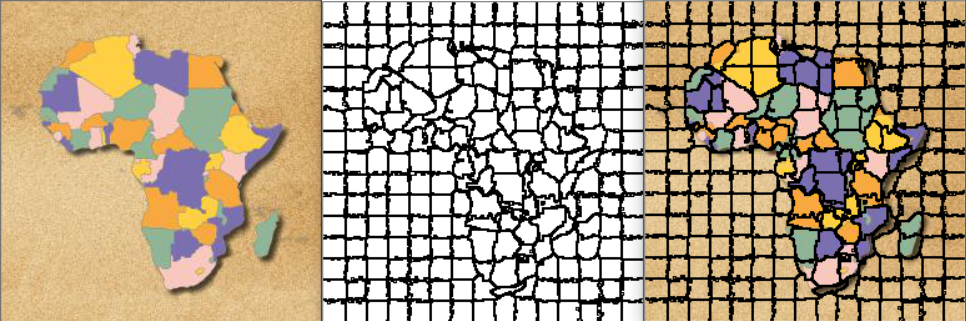
* The approach has 3 steps:
  + - Input parameters step: the user has to choose the desired image, compactness ratio and how many superpixels should the image be segmented into;
    - Calculate superpixels step: this is the main functions that computes the superpixels position and compactness taking into consideration the shape of the image and pixels. The function gets as arguments the image, the compactness ratio, the number of superpixels, and the S step size. Inside, some initialization is done for the float image, lab image, centers, labels, distances and centerStep of the superpixel. Then, we iterate 10 times (as stated in the paper) for each center the following algorithm: using regular grid steps S we search in a window around the pixel’s center and reassign pixels to nearest center. At last, we calculate the distance between 2 pixel points (using the functions stated in the paper) and update cluster allocation if the cluster minimizes the distance;
    - View superpixels step: we take the labels from the previous step, iterate through them and if a label from a pixel is different from the other, we paint that pixel on the image black and show the output images to the user.
* The algorithm used for this project is named SLIC, which comes from Simple Linear Iterative Clustering.
* Suppose that we have:
  + - N (number of image pixels)
    - K (amount of superpixels)
    - (average area of superpixels)
    - (distance between centers)
    - (the superpixel)
    -  (distance between superpixels)
* We initialize cluster centers, repeat 10 times: for each center of superpixel, find similar pixels in step neighborhood and compute the new centers.
* Library used for this project is OpenCV (Open source computer vision), which is a library of programming functions mainly aimed at real-time computer vision.
* Functions from the library used:
  + - convertTo() – to convert the color image to a floating point one;
    - cvtColor() – to convert the float image to a l-a-b colorspace one;
    - ones() – to initialize labels and distance matrices;
    - imshow() – to output images for the user;
    - imwrite() – to write the image into a file;
    - waitKey() – to hold the images so that they don’t disappear unless the user presses a key.
* The most difficult part of the implementation was to decide the size of the window (of the superpixel) with respect to the pixel center. So, I’ve chosen the window size so that the step S is added to the size of the original pixel. The paper specifies that the step should be 2S but I have found some exceptions when doing that, so I’ve let the step S as it is.



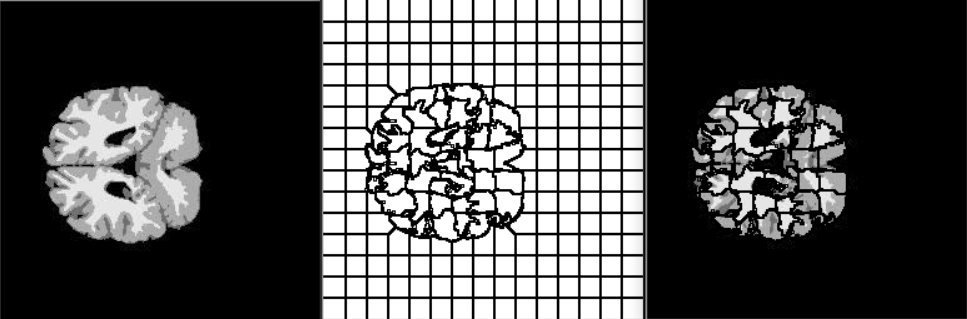
# Experimental results

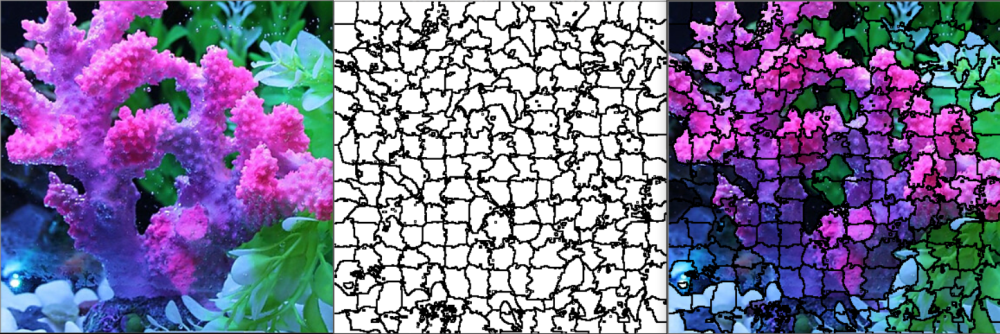


* The dataset is made of 10 images, put up in several categories. Therefore:
  + Textured photos (rocks, texture, microscope), to see how well the segmentation is done and at which dimensions it can go;
  + Text photos (netflix, writing), to see how well the segmentation is done when there are sharp corners;
  + Face photo (gaga), to see how well the segmentation is done in case of face features recognition;
  + “Perturbance” photos (brain, coral), to see how well the segmentation is done when there exists a lot of perturbances in the photos (like segments of the brain and branches of a coral);
  + Environment photo (bistro), to see how well the segmentation is done in case of real life the environments;
  + Map photo (africa), this is more of a test photo, to see how well the segmentation is done in general (covering the borders, contour, etc.).
* Here are the images together with their respective boundaries:

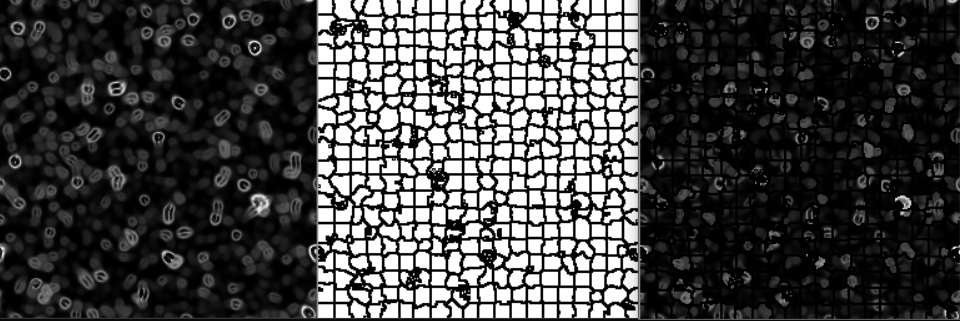


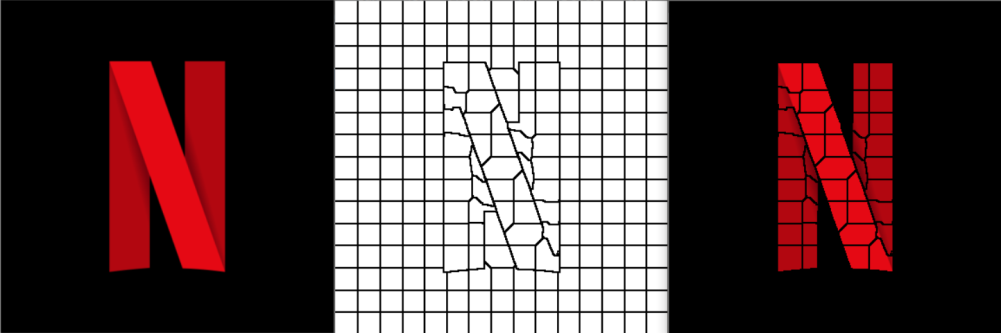


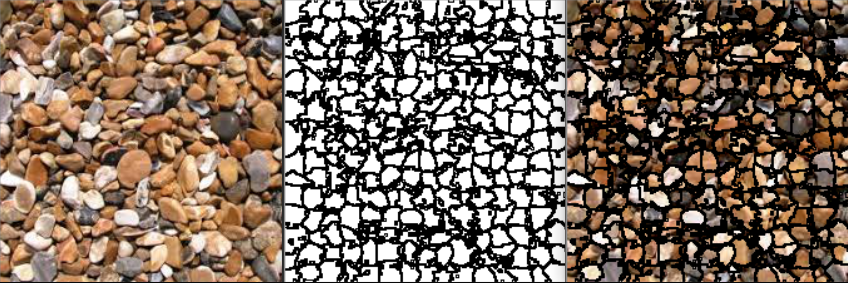


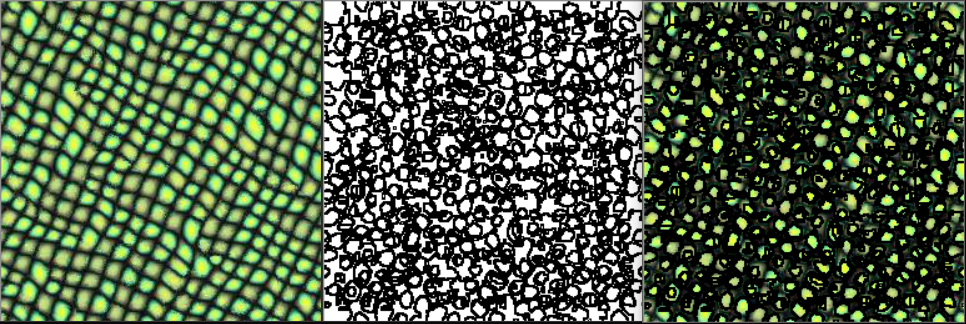






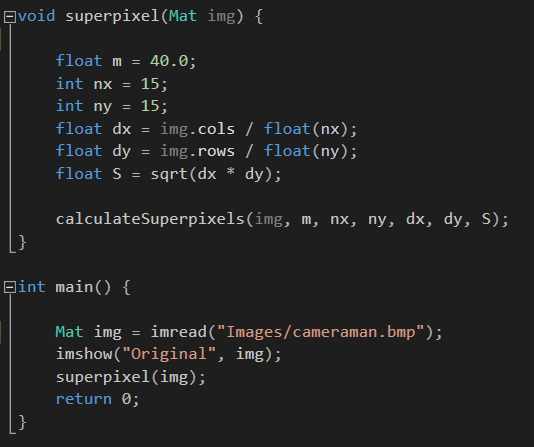








* As a user manual, below is an image with settings that the user needs to do, which is: input in the imread from main() the absolute path to the desired image, input in superpixels(Mat img) the compactness ration m and the number of superpixels on axis x and y which are nx and ny. At last, run the implementation from any C++ IDE.



# Conclusions

* As a wrap-up for my project, I have achieved creating superpixels for any image, of any number and of any compactness ratio. All of the three parameters (image, dimension, compactness) can be choose by the user.
* As accomplishments of the projects objectives, I have successfully implemented the superpixel segmentation of a color image using the SLIC algorithm.
* The pro-s of the projects are: easy implementation with fast results, linear time complexity, parameters that can be modified by the user and convenient output images (original, boundaries, original merged with boundaries).
* The limitations are that I haven’t computed the residual error E and I haven’t enforced connectivity, as the original paper states.
* As future developments, I could create a GUI so that the user can choose the image from a directory (so that he doesn’t need to specify the absolute path) and to choose the parameters in a more intuitive way. As algorithm developments, I could compute the residual error and implement a function that would enforce connectivity of the stray labels.

# Bibliography

<http://www.kev-smith.com/papers/SLIC_Superpixels.pdf>

<http://www.kev-smith.com/papers/SMITH_TPAMI12.pdf>

<https://infoscience.epfl.ch/record/149300>

<https://www.pyimagesearch.com/2014/07/28/a-slic-superpixel-tutorial-using-python/>