Maulana Azad National Institute of Technology

(An Institute of National Importance)
Bhopal – 462003 (India)

Department of Computer Science & Engineering

SEMINAR PROJECT REPORT (CSE-435)

Image Segmentation Enhancements & Optimizations
A Stochastic Approach

Submitted in partial fulfillment for the degree of Bachelor of Technology of Maulana Azad National Institute of Technology, Bhopal

Jishan Shaikh	161112013	jishanshaikh9893@gmail.com
Ankit Chouhan	161112051	ankitchouhan.dws97@gmail.com
Sudhanshu Ranjan	161112046	ranjansudha3042@gmail.com

Supervisor
Dr. Jyoti Bharti
Dept of CSE, MANIT Bhopal

Session 2019-20

Contents

Abstract

This project proposes a novel technique for image segmentation providing customized parameters. Predefined customized parameters makes the technique usable for supervised segmentation, but the results of unsupervised segmentation can be used as an input to the method to get combined results. Image segmentation is very important in visual detection and visual recognition such as facial recognition, object detection, etc. It is quite adaptive to all the supervised segmentation techniques, and also flexible to the unsupervised techniques. Major features of the technique includes easy customization, neutrality towards nature of the segmentation technique, open-source nature, and uniformity of segmentation procedure. The core idea of technique lies in random probabilistic distribution (stochastic model), in which we created 2 matrices of size 8 x 8; one for highly bright image and one for a highly dim image. Matrices were constructed on the basis of random nature of image and we can also use other matrices also (customization). Those initial matrices were then weighted to an score of 10 and -10 (Plus for brightness and Minus for dim image) for effective manipulation and operations. The smallest image that can be processed using this technique is 8 x 8, in that case only one iteration is necessary to calculate the resulting segmented matrix.

If the size of image is greater than 8 x 8, matrices act as filters and calculate filtered value for each row-column element for the image. Since there are two matrices, we calculate 2 filtered results and then average out them for best results of highly bright image and highly dim image. We've tested some images with supervised techniques such as thresholding technique as well as some of unsupervised techniques such as otsu, li, local, etc. We've also demonstrated the technique on some advanced techniques such as snake based contours and Felzenszwalb; and that gives excellent results from a human perspective. We've also implemented this technique for a clustering technique – linear iterative clustering that also give very promising results.

Index terms: *Image segmentation, enhancement, optimization*

Categories: *Computer Vision, Digital Image Processing, Optimization Techniques.*

Section 1
Introduction

"You are not known to me. Would you like to introduce yourself?"
- Unknown

1.1 Image Segmentation

Image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as image objects). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. [Wiki]

1.2 Objectives

- To gain the technical knowledge and experience of image manipulation techniques
- A clear idea of working title; incorporating traditional accent but with some modern software engineering principles, practices, and standards
- Improving team skills under fully supervision (Project management including risk analysis and Soft-skills)
- To present a final result implementation, its analysis, and evaluation with proper documentation/project report
- To present the work in form of a publication ready research paper
- To learn concepts of computer vision to apply them to web browsers
- To enhance team work and individual responsibilities by following Software Engineering principles and standards

1.3 Scope

- Tool/framework can be used theoretically for any image segmentation usage
- With a few feasible modifications, constructed tool can be used for image recognition (objects) as well as face recognition from a data set.
- Research oriented nature of project will definitely leads to better technological developments
- Maintenance phase could compromise with further modifications and enhancements

1.4 Motivation

- Learning and applying the conceptual and fundamental knowledge to a work based on a vast technology i.e. Computer Vision and DIP.
- Contribution to industrial-research community
- To have practical “hands on” experience with Python and OpenCV.
- Enhancing team and soft skills of members, with research
- Learning by doing; modern technologies and their challenges

1.5 Output and deliverable(s)

- Project report (with other documents including synopsis, plan document, design document, etc.).
- Seminar or presentation based on the topic.
- A clean white paper on the novel technique. Other research papers may include comparison parameters of various techniques, etc.
- **Proposition of new model with more advanced architecture applicable to various predefined tools such as Tensorflow and Pytorch.**

1.6 Problem Statement

Image segmentation usually concerns dividing an image into multiple segments of some semantic usage for a human. Per se, if an image having description of nature, rocks, sun, etc. then segmented image should clearly distinguish the elements based on intensity of pixels after converting it to an gray-scale image. Various segmentation techniques are available e.g. supervised thresholding and unsupervised techniques such as otsu, li, and local. Each technique has their own pros and cons, the major one is enhancement perspective; none of the technique by default do not normalize the intensity of pixels.

We proposed a new technique of image segmentation with enhancement and optimized stochastic approach for usage of random markov model (Brain inspired approach).

▪▪▪▪▪

Section 2
Literature Review and Survey

"What else a textbook can even say about topic out of its coverage?"
- Unknown

2.1 Image Segmentation Research

Image segmentation is one of the most explored field in the subject of computer vision and digital image processing. Many techniques have been discovered for segmentation which are widely used in inter-domain applications. Each of them have their own pros and cons.

Some segmentation techniques survey and comparisons are presented in [24] and [62]. The idea of a hidden Markov model in Image segmentation is inspired from [2]. The randomization concept was first given by [41]. Some techniques such as linear iterative method is given in [3]. Contour based segmentation is popularized by [31], [32]. Solving the problem of segmentation using genetic algorithms was first studied by [59]. An automata based approach was used in [65].

Image segmentation has been exclusively studied in [23] and [24] for their Phd work and dissertation.

The concept of hidden Markov model with randomized matrices has never been explored anywhere in image segmentation problem. We've used it with stochastic distribution over all the tested images. The implemented model is somewhat capable of demonstration of human behavior (NNN – Natural Neural Networks) in internal visualization in senses analysis using Random Markov Model.

2.2 Literature Review

(Literature review from https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic3.htm)

There are two types of image segmentation techniques:

- Supervised Segmentation (Pixel thresholding techniques)
- Unsupervised Segmentation (Including otsu, li, and local)

A survey of segmentation techniques is presented in [].

Segmentation partitions an image into distinct regions containing each pixels with similar attributes. To be meaningful and useful for image analysis and interpretation, the regions should strongly relate to depicted objects or features of interest. Meaningful segmentation is the first step from low-level image processing transforming a gray scale or color image into one or more other images to high-level image description in terms of features, objects, and scenes. The success of image analysis depends on reliability of segmentation, but an accurate partitioning of an image is generally a very challenging problem.

Segmentation techniques are either *contextual* or *non-contextual*. The latter take no account of spatial relationships between features in an image and group pixels together on the basis of some global attribute e. g. gray level or color. Contextual techniques additionally exploit these relationships, e. g. group together pixels with similar gray levels and close spatial locations.

Non-contextual thresholding

Thresholding is the simplest non-contextual segmentation technique. With a single threshold, it transforms a gray-scale or color image into a binary image considered as a binary region map. The binary map contains two possibly disjoint regions, one of them containing pixels with input data values smaller than a threshold and another relating to the input values that are at or above the threshold. The former and latter regions are usually labelled with zero (0) and non-zero (1) labels, respectively. The segmentation depends on image property being thresholded and on how the threshold is chosen.

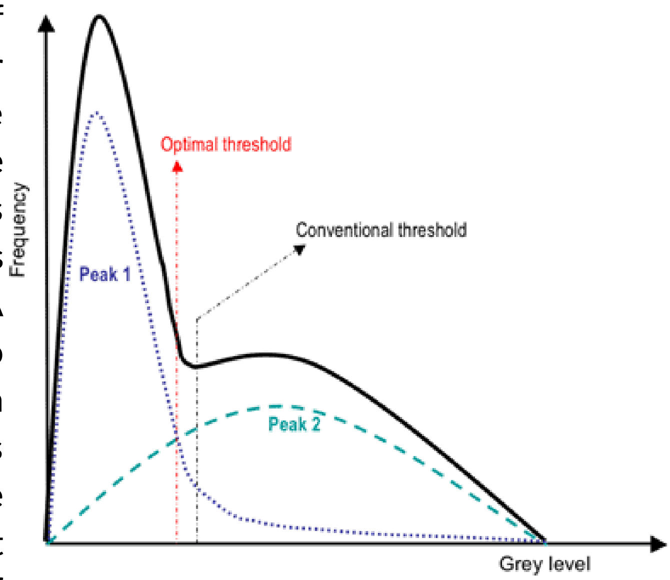
Generally, the non-contextual thresholding may involve two or more thresholds as well as produce more than two types of regions such that ranges of input image signals related to each region type are separated with thresholds. The question of thresholding is how to automatically determine the threshold value.

Simple thresholding

The most common image property to threshold is pixel grey level: $g(x,y) = 0$ if $f(x,y) < T$ and $g(x,y) = 1$ if $f(x,y) \geq T$, where T is the threshold. Using two thresholds, $T_1 < T_1$, a range of grey levels related to region 1 can be defined: $g(x,y)= 0$ if $f(x,y) < T_1$ OR $f(x,y) > T_2$ and $g(x,y) = 1$ if $T_1 \leq f(x,y) \leq T_2$.

The main problems are whether it is possible and, if yes, how to choose an adequate threshold or a number of thresholds to separate one or more desired objects from their background. In many practical cases the simple thresholding is unable to segment objects of interest, as shown in the above images.

A general approach to thresholding is based on assumption that images are *multi-modal*, that is, different objects of interest relate to distinct peaks (or modes) of the 1 D signal histogram. The thresholds have to optimally separate these peaks in spite of typical overlaps between the signal ranges corresponding to individual peaks. A threshold in the valley between two overlapping peaks separates their main bodies but inevitably detects or rejects falsely some pixels with intermediate signals. The optimal threshold that minimizes the expected numbers of false detections and rejections may not coincide with the lowest point in the valley between two overlapping peaks. (See figure)



Adaptive thresholding

Since the threshold separates the background from the object, the adaptive separation may take account of empirical probability distributions of object (e.g. dark) and background (bright) pixels. Such a threshold has to equalise two kinds of expected errors: of assigning a background pixel to the object and of assigning an object pixel to the background. More complex adaptive thresholding techniques use a spatially varying threshold to compensate for local spatial context effects (such a spatially varying threshold can be thought as a background normalization).

A simple iterative adaptation of the threshold is based on successive refinement of the estimated peak positions. It assumes that (*i*) each peak coincides with the mean grey level for all pixels that relate to that peak and (*ii*) the pixel probability decreases monotonically on the absolute difference between the pixel and peak values both for an object and background peak. The classification of the object and background pixels is done at each iteration j by using the threshold T_j found at previous iteration.

Thus, at iteration j, each grey level $f(x,y)$ is assigned first to the object or background class (region) if $f(x,y) \le T_j$ or $f(x,y) > T_j$, respectively. Then, the new threshold, T_{j+1} =

$0.5(\mu_{j,ob} + \mu_{j,bg})$ where $\mu_{j,ob}$ and $\mu_{j,bg}$ denote the mean grey level at iteration *j* for the found object and background pixels, respectively:

Input : an image histogram $\mathbf{h} = \{h(q) : q = 0,\ldots,255\}$

Initialisation : $j = 0;\quad N = \sum_{q=0}^{255} h(q);\quad T_0 = \frac{1}{N}\sum_{q=0}^{255} qh(q)$

while $T_{j+1} \neq T_j$ **do**

$$\mu_{j,\mathrm{ob}} = \frac{\sum_{q=0}^{T_j} qh(q)}{\sum_{q=0}^{T_j} h(q)}\;;\quad \mu_{j,\mathrm{bg}} = \frac{\sum_{q=T_j+1}^{255} qh(q)}{\sum_{q=T_j+1}^{255} h(q)};\quad T_{j+1} = \frac{\mu_{j,\mathrm{ob}} + \mu_{j,\mathrm{bg}}}{2}$$

end while

Color thresholding

Color segmentation may be more accurate because of more information at the pixel level comparing to gray-scale images. The standard Red-Green-Blue (RGB) color representation has strongly interrelated color components, and a number of other color systems (e. g. HSI Hue-Saturation-Intensity) have been designed in order to exclude redundancy, determine actual object / background colors irrespective of illumination, and obtain more more stable segmentation.

Contextual segmentation: Region growing

Non-contextual thresholding groups pixels with no account of their relative locations in the image plane. **Contextual segmentation** can be more successful in separating individual objects because it accounts for closeness of pixels that belong to an individual object. Two basic approaches to contextual segmentation are based on signal *discontinuity* or *similarity*. Discontinuity-based techniques attempt to find complete boundaries enclosing relatively uniform regions assuming abrupt signal changes across each boundary. Similarity-based techniques attempt to directly create these uniform regions by grouping together connected pixels that satisfy certain similarity criteria. Both the approaches mirror each other, in the sense that a complete boundary splits one region into two.

Pixel connectivity

Pixel connectivity is defined in terms of pixel neighborhood. A normal rectangular sampling pattern producing a finite arithmetic lattice {(x,y): x = 0, 1, ..., X−1; y = 0, 1, ..., Y−1} supporting digital images allows us to define two types of neighborhood surrounding a pixel. A **4-neighborhood** {(x−1,y), (x,y+1), (x+1,y), (x,y−1)} contains only the pixels above, below, to the left and to the right of the central pixel (x,y). An **8-neighborhood** adds to the 4-neighborhood four diagonal neighbors: {(x−1,y−1), (x−1,y), (x−1,y+1), (x,y+1), (x+1,y+1), (x+1,y), (x+1,y−1), (x,y−1)}.

A 4-connected path from a pixel $p1$ to another pixel p_n is defined as the sequence of pixels {$p1$, $p2$, ..., p_n} such that pi+1 is a 4-neighbor of p_i for all i = 1, ..., n−1. The path is 8-connected if pi+1 is an 8-neighbor of p_i. A set of pixels is a 4-connected region if there exists at least one 4-connected path between any pair of pixels from that set. The 8-connected region has at least one 8-connected path between any pair of pixels from that set.

Region similarity

The uniformity or non-uniformity of pixels to form a connected region is represented by **auniformity predicate**, i.e. a logical statement, or condition being true if pixels in the regions are similar with respect to some property (color, grey level, edge strength, etc). A common predicate restricts signal variations over a neighborhood: the predicate *P(R)*, where *R* denotes a connected region, is TRUE if $|f(x,y) - f(x+\xi, y+\eta)| \le \Delta$ and FALSE otherwise (here, (x,y) and $(x+\xi,y+\eta)$ are the coordinates of neighboring pixels in region *R*. This predicate does not restrict the gray level variation within a region because small changes in signal values can accumulate over the region.

Intra-region signal variations can be restricted with a similar predicate: *P(R)* = TRUE if |*f(x,y)* − *&mu*$_R$| ≤ &Delta and FALSE otherwise where (x,y) is a pixel from the region *R* and μ_R is the mean value of signals *f(x,y)* over the entire region *R*.

Generally, a "good" complete segmentation must satisfy the following criteria:

1. All pixels have to be assigned to regions.
2. Each pixel has to belong to a single region only.
3. Each region is a connected set of pixels.
4. Each region has to be uniform with respect to a given predicate.
5. Any merged pair of adjacent regions has to be non-uniform.

Split-and-merge segmentation

The top-down **split-and-merge** algorithm considers initially the entire image to be a single region and then iteratively splits each region into sub-regions or merges adjacent regions until all regions become uniform or until the desired number of regions have been established.

A common splitting strategy for a square image is to divide it recursively into smaller and smaller quadrants until, for any region *R*, the uniformity predicate *P(R)* is TRUE. The strategy builds a top-down *quadtree.*

Apart from these, texture segmentation is also there for highly diverse textures in Gray scale format.

Deep learning techniques have vastly improved the field of image segmentation, some of the common examples have been presented in the references.

▪▪▪▪▪

Section 3
Proposed work

"Don't tell me about yourself. Show me what you are capable of."
- Unknown

3.1 Methodology

The methodology adopted in the project is simple and is easy to implement. **Functional research and Object oriented methodology** constitute a major portion of methodology. Project is divided into individual's tasks. Tasks are then taken as initial problem and solved individually and have to integrate them at last. Objectives are set first and then been tried to fulfill all objectives asserted.

The Project development model used in project development is closely related to hybrid association of waterfall model, Rapid application development (Rapid delivery of project, time to time), Agile methodology (Simplicity and Swift evaluation), Some Extreme Programming practices, and 4th generation tool usage (Usage of CASE tool: LibreOffice Base/Draw, etc.). The involvement of multiple model element in single project leads to higher level of customization, better management of project, better requirements adaptability, and improved rigidity of overall management procedure and development life cycle. Yet on a broad way, this project is closely resembles with **Agile methodologies**.

3.2 Description

This project proposes a novel technique for image segmentation providing customized parameters. Predefined customized parameters makes the technique usable for supervised segmentation, but the results of unsupervised segmentation can be used as an input to the method to get combined results. Image segmentation is very important in visual detection and visual recognition such as facial recognition, object detection, etc. It is quite adaptive to all the supervised segmentation techniques, and also flexible to the unsupervised techniques. Major features of the technique includes easy customization, neutrality towards nature of the segmentation technique, open-source nature, and uniformity of segmentation procedure. The core idea of technique lies in random probabilistic distribution (stochastic model), in which we created 2 matrices of 8x8; one for highly bright image and one for a highly dim image. Matrices were constructed on the basis of random nature of image and we can also use other matrices also (customizable). Those initial matrices were then weighted to an score of 10 and -10 (Plus for brightness and Minus for dim image) for effective manipulation and operations. The smallest image that can be processed using this technique is 8x8, in that case only one iteration is necessary to calculate the resulting segmented matrix.

If the size of image is greater than 8x8, matrices act as filters and calculate filtered value for each row-column element for the image. Since there are two matrices, we calculate 2 filtered results and then average out them for best results of highly bright image and highly dim image. We've tested some images with supervised techniques such as thresholding technique as well as some of unsupervised techniques such as otsu, li, local, etc. We've also demonstrated the technique on some advanced techniques such as snake based contours and Felzenszwalb; and that gives excellent results from a human perspective. We've also implemented this technique for a clustering technique – linear iterative clustering that also give very promising results.

▪▪▪▪▪

Section 4
Tools and Technologies

"Right hammer for a nail is important. Else processing with garbage produces garbage."
- Unknown

4.1 Software and Hardware requirements

Minimum System/Software/Hardware requirements :

- Windows/Linux/Mac OS/Chromium OS.
- Memory requirements: 512 MB (RAM).
  - Recommended 1 GB.
- Secondary memory requirements (Hard disk): 10 GB (ROM).
  - Recommended 256/512 GB
- Latest versions of web browsers - Chrome, Chromium, Firefox, Opera.
- Jupyter Notebook (Can run in Google Colab if libraries installed)
- Python libraries: Scikit-image, numpy, pandas.
- Python 3.8.0 (highly recommended)
- Anaconda Distribution (Navigator) Preferable
- Writer and diagram designing software such as LibreOffice Draw.
- Clock Speed: 866 MHz
- Virtual Memory: 64 bits (minimum).
- Cache Memory: 512 KB, etc.

Resources Usage :

- Operating System: Windows 10/Ubuntu 19.10/Kali Linux 19.0.5
- Memory: 8 GB (RAM)
- Secondary memory: 1 TB (ROM)
- Firefox, Chrome, Chromium, or Opera web browser.
- Microsoft Word 16/ LibreOffice Writer 5.4 (Linux).

Functional requirements for the tool :

- Cached data storage and retrieval.
- Proper User Interface as in accordance with use-case diagrams.
- Customized testing function.

Section 5
Implementation and Coding

"To code is worthless, if it is not decodable."
- Unknown

5.1 main.py (Program to enhance segmentation of images)

```
# -*- coding: utf-8 -*-
"""final_segmentation_opt.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1CU_VehLz2TbVioB5YJqlpS5mQ8qw4R9

# Importing libraries:

**Matplotlib for inclusive use of imread and image_show:**
"""

# Commented out IPython magic to ensure Python compatibility.
# %matplotlib inline

import numpy as np
import matplotlib.pyplot as plt

from skimage import data
from skimage import io
import skimage.filters as filters
import skimage.draw as draw
import skimage.color as color
import skimage.segmentation as seg  # For comparison purposes

import cv2

"""**Importing gray image from sklear.data:**"""

image = data.checkerboard()
# image = data.binary_blobs() or image = data.camera()
plt.imshow(image, cmap = 'gray')

"""**Importing color image from sklearn.data:**"""

image = data.rocket()
# image = data.logo() or image = data.astronaut()
plt.imshow(image)

```

```
plt.imshow(image)

"""**Import a color image using a global URL:**"""

image = io.imread('https://www.gstatic.com/webp/gallery/1.jpg')
plt.imshow(image)

"""**Importing multiple images from local directory:**"""

image_array = io.ImageCollection('../images/*.png:../images/*.jpg')
print('Type: ', type(image_array))

"""**Customized function to show images:**"""

# Returns 2 parameters figuree and axx; the outputs of plt.subplots()
def show_image(image_object, no_of_rows = 1, no_of_cols = 1, cmap = 'gray'):
        # Can change size of subplots from 15x15 to anything else desired.
        figuree, axx = plt.subplots(nrows = no_of_rows, ncols = no_of_cols, figsize = (15, 15))
        axx = imshow(image_object, cmap = 'gray')
        axx.axis('off')
        return figuree, axx

"""**General thresholding:**"""

rock = data.rocket()
plt.imshow(rock);

# Only have pixels greater than 40 intensity value
rock_seg40 = rock>40
# plt.imshow(rock_seg40);

rock_seg80 = rock>80
print(rock_seg80)
# plt.imshow(rock_seg80);

rock_seg120 = rock>120
# plt.imshow(rock_seg120);

"""**Generate circle to create a snake around it:**"""

```

```
"""**Generate circle to create a snake around it:**"""

def generate_circle(res, center, radius):
        rads = np.linspace(0, 2*np.pi, res)

        c = center[1] + radius*np.cos(rads)
        r = center[0] + radius*np.sin(rads)

        return np.array([c, r]).T

"""# Optimization in unsupervised segmentation:"""

text = data.page()
image_show(text)

text_threshold = filters.threshold_otsu(text)

image_show(text > text_threshold);

text_threshold = filters.threshold_li(text)

image_show(text > text_threshold);

text_threshold = filters.threshold_local(text,block_size=51, offset=10)
image_show(text > text_threshold);

"""# Optimization in snake based contour segmentation:"""

points = generate_circle(200, [160, 120], 60)[:-1]
fig, ax = image_show(image_gray)
ax.plot(points[:, 0], points[:, 1], '--r', lw=3)

"""**Generating snake based on circle:**"""

import skimage.segmentation as seg
snake = seg.active_contour(image_gray, points)
fig, ax = image_show(image)
ax.plot(points[:, 0], points[:, 1], '--r', lw=3)
ax.plot(snake[:, 0], snake[:, 1], '-b', lw=3);
```

```
ax.plot(snake[:, 0], snake[:, 1], '-b', lw=3);

snake = seg.active_contour(image_gray, points,alpha=0.06,beta=0.3)
fig, ax = image_show(image)
ax.plot(points[:, 0], points[:, 1], '--r', lw=3)
ax.plot(snake[:, 0], snake[:, 1], '-b', lw=3);

"""# Optimization  in Linear Iterative Clustering:"""

imagee = io.imread('https://www.gstatic.com/webp/gallery/1.jpg')
image_slice = seg.slic(imagee, n_segments = 255)

# label2rgb replaces each discrete label with the average interior color
image_show(color.label2rgb(image_slic, image, kind='avg'));

"""# Optimization in Felzenszwalb Segmentation:"""

image_felzenszwalb = seg.felzenszwalb(image)
image_show(image_felzenszwalb);

np.unique(image_felzenszwalb).size

image_felzenszwalb_colored = color.label2rgb(image_felzenszwalb, image, kind='avg')
image_show(image_felzenszwalb_colored);

"""**Utility function for a generalized optimization:**"""

def image_show(image_object):
    # Filter matrix for an 8x8 image with mostly positive intensity values
    single_88_positive = {
        {0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0},
        {5.0,  5.0,  5.0,  5.0,  5.0,  5.0,  5.0,  5.0},
        {1.0,  1.0,  2.0,  3.0,  3.0,  2.0,  1.0,  1.0},
        {0.5,  0.5,  1.0,  2.5,  2.5,  1.0,  0.5,  0.5},
        {0.0,  0.0,  0.0,  2.0,  2.0,  0.0,  0.0,  0.0},
        {0.5, -0.5, -1.0,  0.0,  0.0, -1.0, -0.5,  0.5},
        {0.5,  1.0, 1.0,  -2.0, -2.0,  1.0,  1.0,  0.5},
        {0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0}
    };

```

```
    # Filter for absorbing edges and curves using negative intensity values
    single_88_negative = {
        { -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0},
        { -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0},
        { -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0},
        { -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0},
        { -2.0, -3.0, -3.0, -4.0, -4.0, -3.0, -3.0, -2.0},
        { -1.0, -2.0, -2.0, -2.0, -2.0, -2.0, -2.0, -1.0},
        {  2.0,  2.0,  0.0,  0.0,  0.0,  0.0,  2.0,  2.0 },
        {  2.0,  3.0,  1.0,  0.0,  0.0,  1.0,  3.0,  2.0 }
    };

    image_object = filters.normalize(alpha = 0.01, beta = 0.1, scale = 'absolute')
    image_object_gray = color.rgb2gray(image_object)
    absolute_best_segmented_image = NULL
    no_of_segments_in_abs_best = 1
    iterations_till_now_for_t = 0

    for(t in range(25, max(image_object_gray))):
        try:
            new_segmented_image = seg.inverse_gaussian_gradient(image_object_gray)
            selected_threshold = t
            iterations_till_now_for_s = 0
            for(s in range(1, 255)):
                single_88_positive_image = new_segmented_image.filter(single_88_positive)
                single_88_negative_image = new_segmented_image.filter(single_88_negative)
                new_segmented_image = min(single_88_positive_image, single_88_negative_image)
                new_segmented_image = seg.join_segmentations(image_object_gray, new_segmented_image)
                if(quality(new_segmented_image) > quality(absolute_best_segmented_image)):
                    absolute_best_segmented_image = new_segmented_image
                    no_of_segments_in_abs_best = s
                    iterations_till_now_for_s += 1
            iterations_till_now_for_t += 1
        except:
            print("Error: Finding optimized image is not possible for given image.")
        finally:
            print("No runtime error till now, internally.")

    return absolute_best_segmented_image, no_of_segments_in_abs_best

```

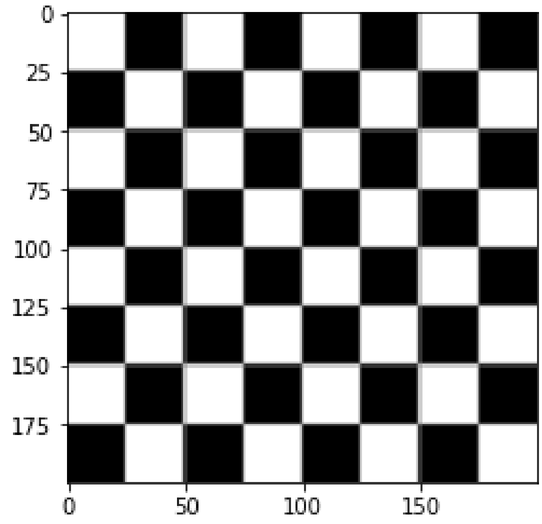
Section 6
Results

"In some contexts, results are more important than complete work done"
- Unknown

6.1 Notebook Results

```
[ ]  image = data.checkerboard()
     # image = data.binary_blobs() or image = data.camera()
     plt.imshow(image, cmap = 'gray')
```

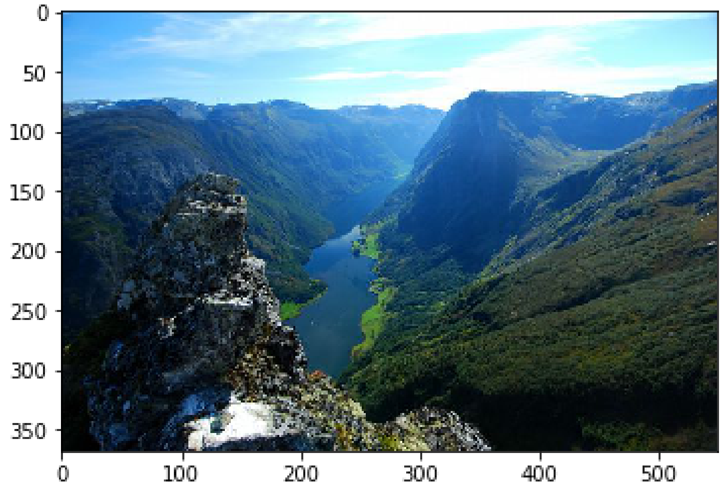
```
<matplotlib.image.AxesImage at 0x7f815865b908>
```



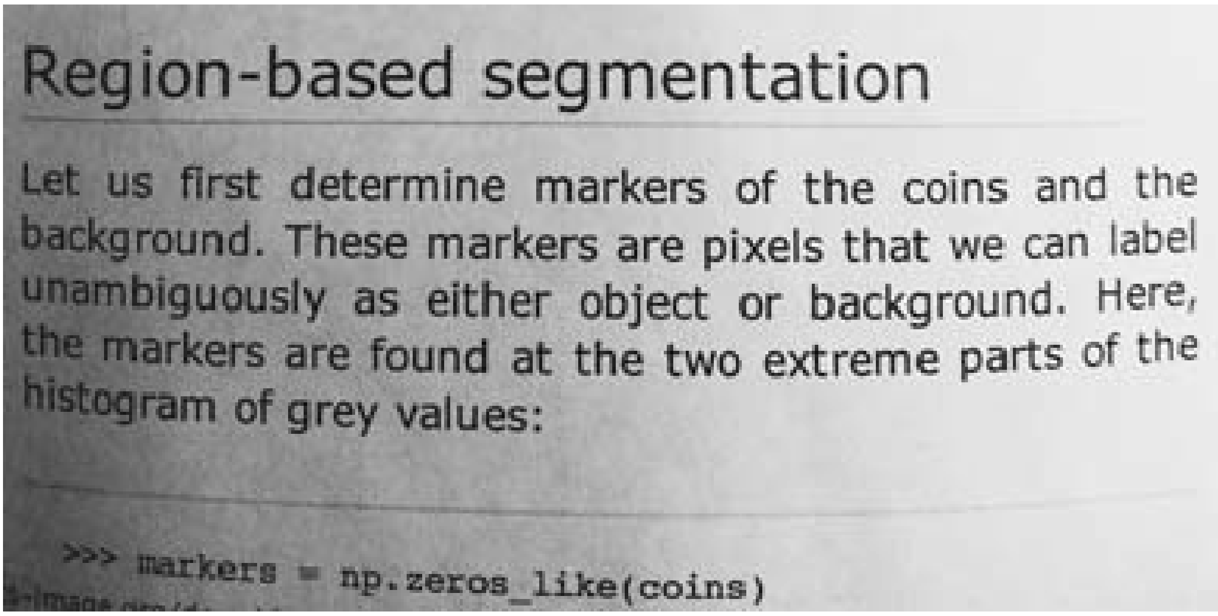
Demonstration on a simple chessboard

```
[ ]  1  image = io.imread('https://www.gstatic.com/webp/gallery/1.jpg')
     2  plt.imshow(image)
```

```
<matplotlib.image.AxesImage at 0x7f8155d84128>
```



Sample image for segmentation

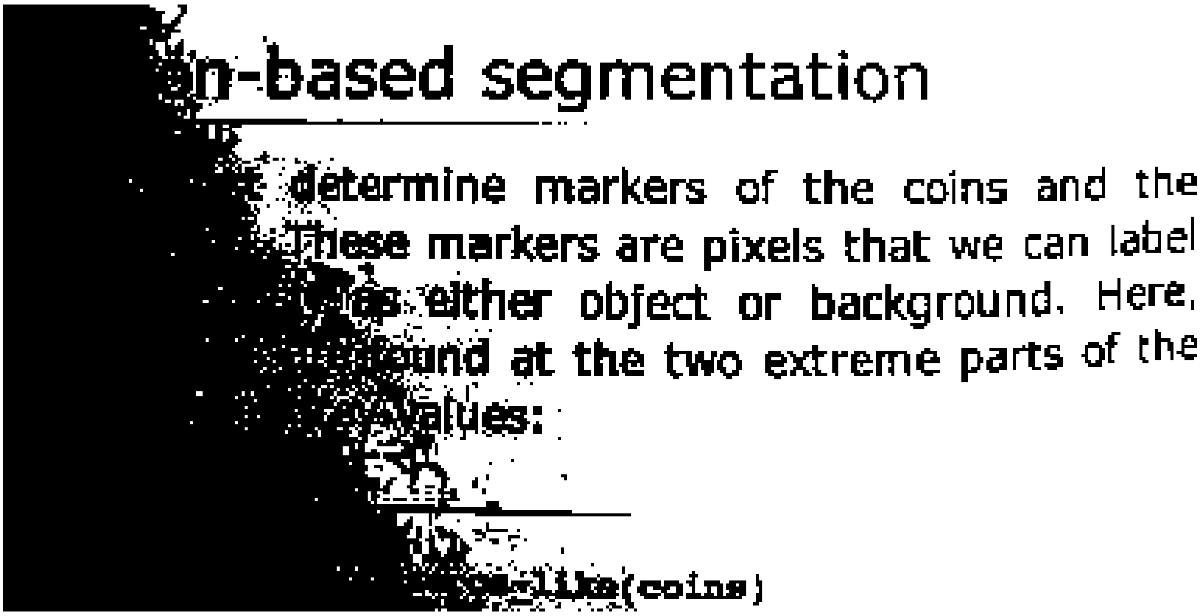


Region-based segmentation

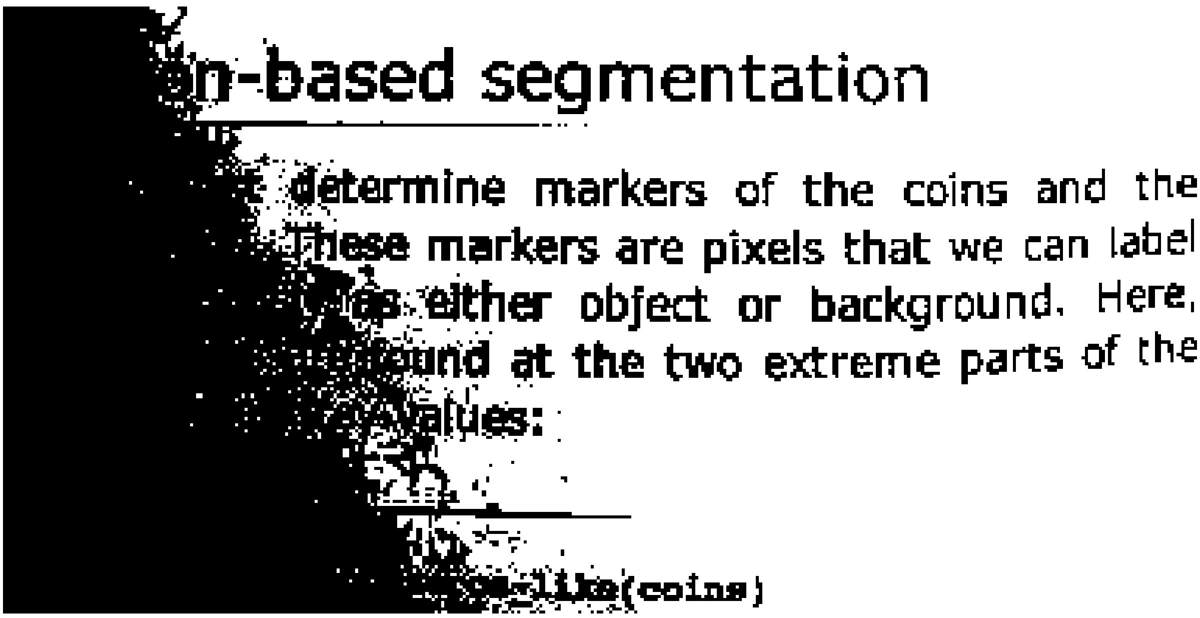
Let us first determine markers of the coins and the background. These markers are pixels that we can label unambiguously as either object or background. Here, the markers are found at the two extreme parts of the histogram of grey values:

```
>>> markers = np.zeros_like(coins)
```

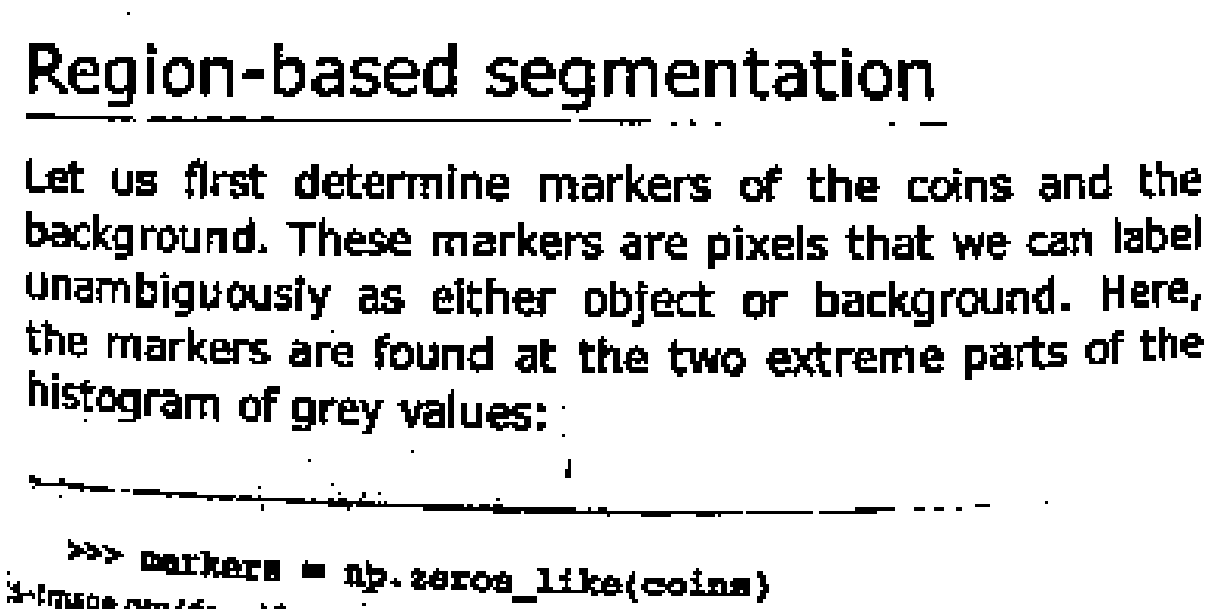
Sample text image for testing



Applied on unsupervised otsu.



Applied on unsupervised li.

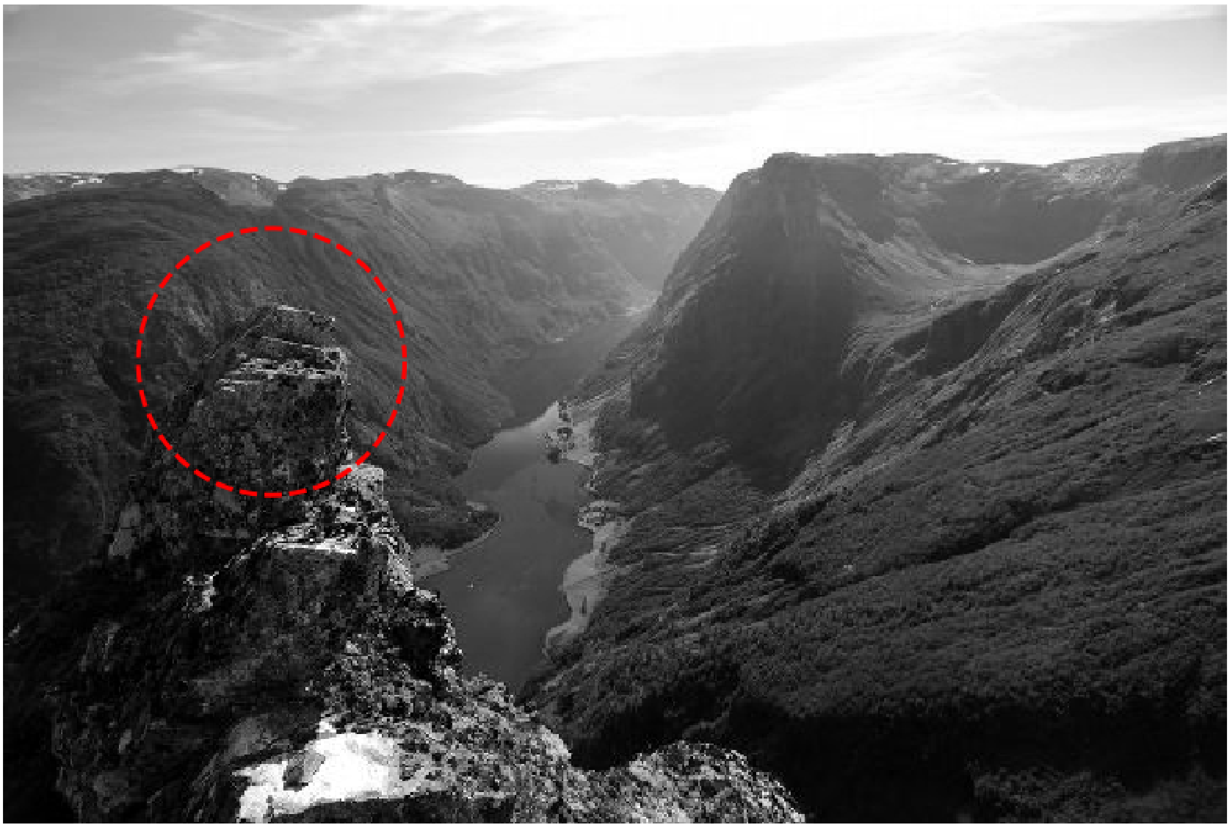


Region-based segmentation

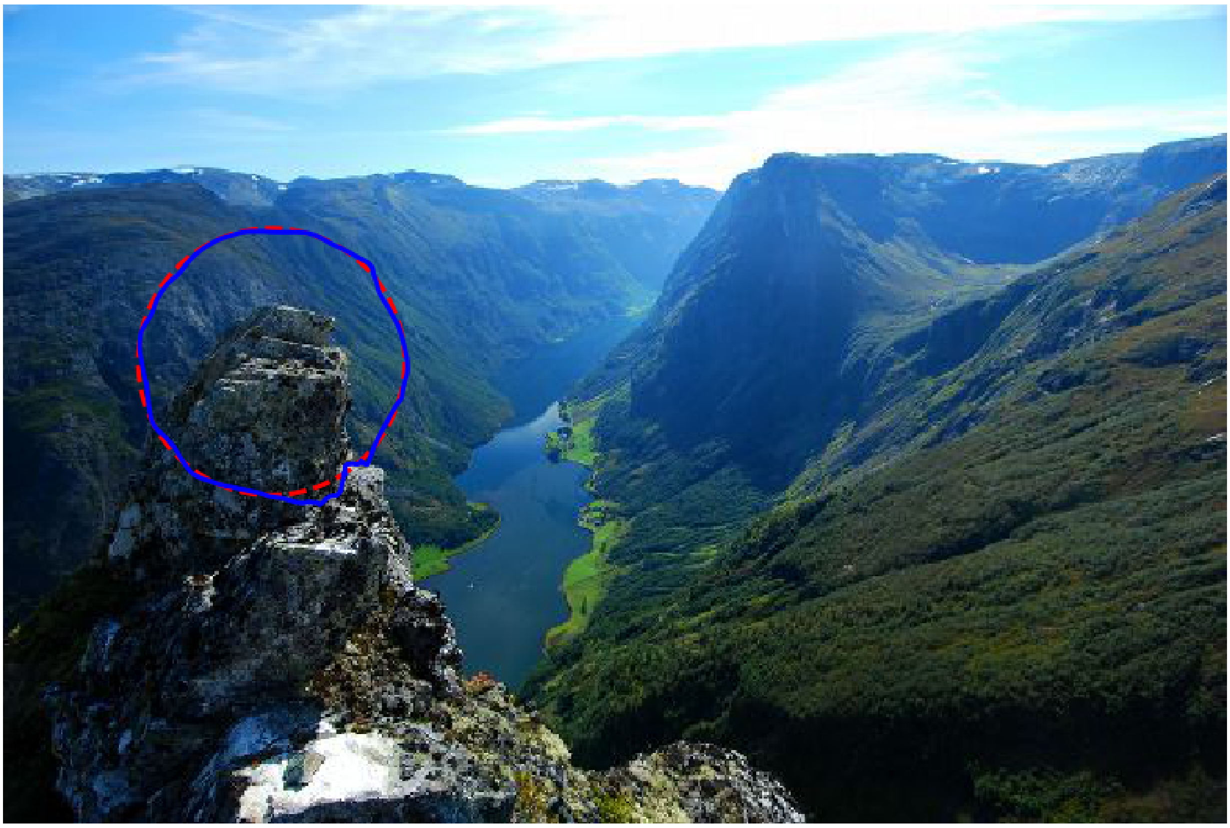
Let us first determine markers of the coins and the background. These markers are pixels that we can label unambiguously as either object or background. Here, the markers are found at the two extreme parts of the histogram of grey values:

```
>>> markers = np.zeros_like(coins)
```

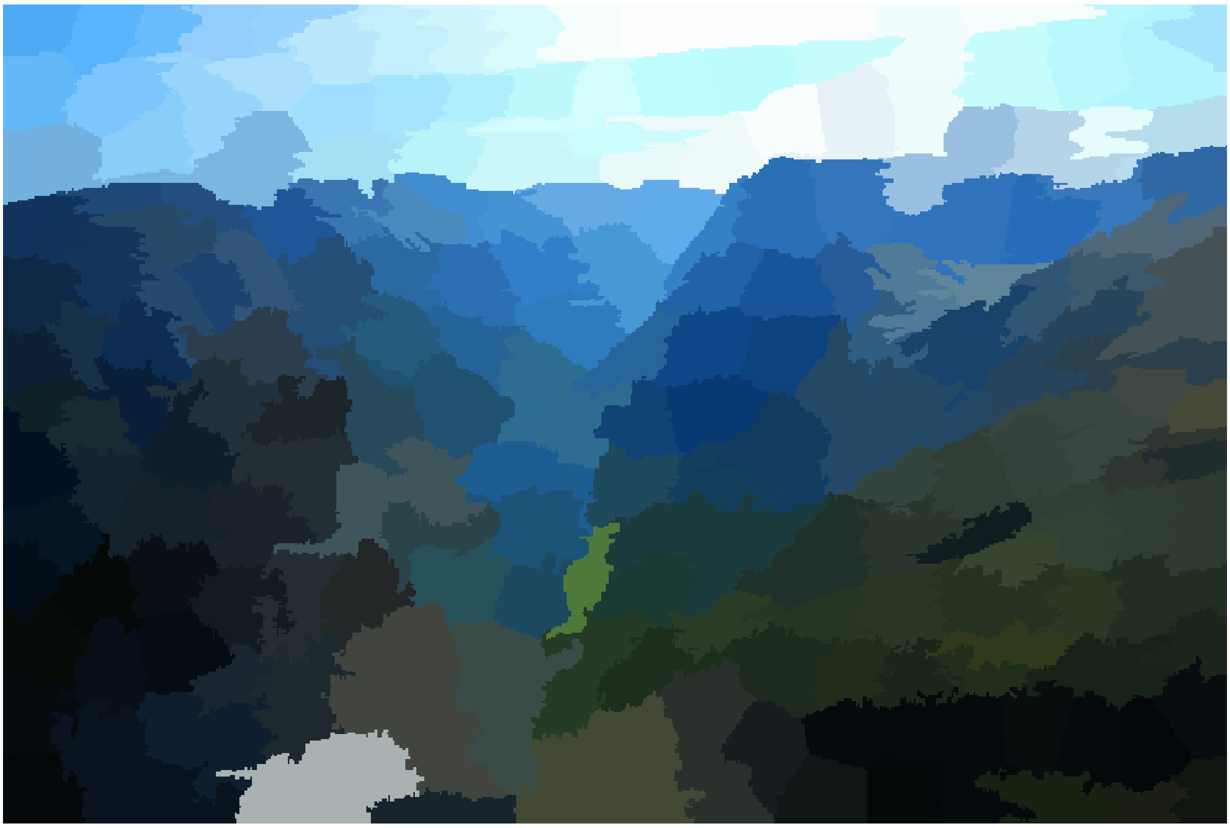
Applied on unsupervised local (block size = 51, and offset = 10).



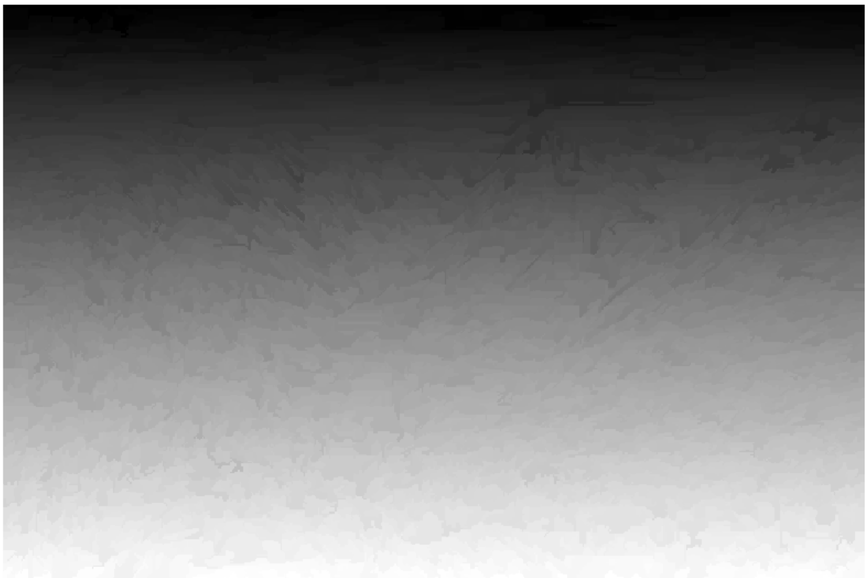
Snake created manually in form of a circle



Applied on a circular snake based contour segmentation (easily converted to RGB)



Applied on Iterative Clustering (Linear)



felzenswalb segmentation for a gray scale image.

Segmented using felzenswalb technique.

Appendix-A (Other Screenshots)

final_segmentation_opt.ipynb
File Edit View Insert Runtime Tools Help Last saved at 7:24 AM
Comment Share
+ Code + Text Connect Editing

Importing libraries:

Matplotlib for inclusive use of imread and image_show:

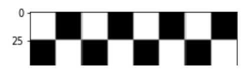
```
%matplotlib inline
```

```
import numpy as np
import matplotlib.pyplot as plt

from skimage import data
from skimage import io
import skimage.filters as filters
import skimage.draw as draw
import skimage.color as color
import skimage.segmentation as seg

import cv2
```

Importing gray image from sklear.data:

```
image = data.checkerboard()
# image = data.binary_blobs() or image = data.camera()
plt.imshow(image, cmap = 'gray')
```

<matplotlib.image.AxesImage at 0x7f815865b908>



final_segmentation_opt.ipynb
File Edit View Insert Runtime Tools Help Last saved at 7:24 AM
Comment Share
+ Code + Text Connect Editing

Importing color image from sklearn.data:

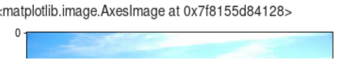
```
image = data.rocket()
# image = data.logo() or image = data.astronaut()
plt.imshow(image)
```

<matplotlib.image.AxesImage at 0x7f8155df2ac8>

Import a color image using a global URL:

```
image = io.imread('https://www.gstatic.com/webp/gallery/1.jpg')
plt.imshow(image)
```

<matplotlib.image.AxesImage at 0x7f8155d84128>



final_segmentation_opt.ipynb
File Edit View Insert Runtime Tools Help Last saved at 7:24 AM
Comment Share
+ Code + Text
Connect Editing

Importing multiple images from local directory:

```
image_array = io.ImageCollection('../images/*.png:../images/*.jpg')
print('Type: ', type(image_array))
```

Type: <class 'skimage.io.collection.ImageCollection'>

Customized function to show images:

```
# Returns 2 parameters figuree and axx; the outputs of plt.subplots()
def show_image(image_object, no_of_rows = 1, no_of_cols = 1, cmap = 'gray'):
        # Can change size of subplots from 15x15 to anything else desired.
        figuree, axx = plt.subplots(nrows = no_of_rows, ncols = no_of_cols, figsize = (15, 15))
        axx = imshow(image_object, cmap = 'gray')
        axx.axis('off')
        return figuree, axx
```

General thresholding:

```
rock = data.rocket()
plt.imshow(rock);

# Only have pixels greater than 40 intensity value
rock_seg40 = rock>40
```

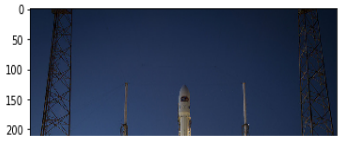
final_segmentation_opt.ipynb
File Edit View Insert Runtime Tools Help Last saved at 7:24 AM
Comment Share
+ Code + Text
Connect Editing

```
plt.imshow(rock);

# Only have pixels greater than 40 intensity value
rock_seg40 = rock>40
# plt.imshow(rock_seg40);

rock_seg80 = rock>80
print(rock_seg80)
# plt.imshow(rock_seg80);

rock_seg120 = rock>120
# plt.imshow(rock_seg120);
```

```
[[[False False False]
  [False False False]
  [False False False]
  ...
  [False False False]
  [False False False]
  [False False False]]

 [[False False False]
  [False False False]
  [False False False]
  ...
  [False False False]
  [False False False]
  [False False False]]

 [[False False False]
  [False False False]
  [False False False]
```

final_segmentation_opt.ipynb
File Edit View Insert Runtime Tools Help Last saved at 7:24 AM
Comment Share
+ Code + Text
Connect Editing

```
[False False False]
[False False False]
...
[ True False False]
[False False False]
[ True False False]]
```



Generate circle to create a snake around it:

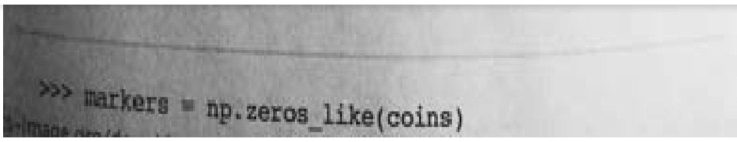
```python
def generate_circle(res, center, radius):
        rads = np.linspace(0, 2*np.pi, res)

        c = center[1] + radius*np.cos(rads)
        r = center[0] + radius*np.sin(rads)

        return np.array([c, r]).T
```

Optimization in unsupervised segmentation:

```python
text = data.page()
image_show(text)
```

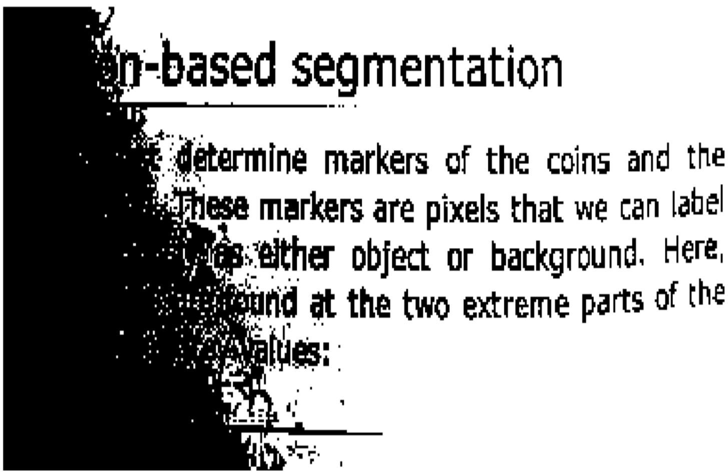
(<Figure size 1008x1008 with 1 Axes>,

final_segmentation_opt.ipynb
File Edit View Insert Runtime Tools Help Last saved at 7:24 AM
Comment Share
+ Code + Text
Connect Editing



>>> markers = np.zeros_like(coins)

```python
text_threshold = filters.threshold_otsu(text)

image_show(text > text_threshold);
```

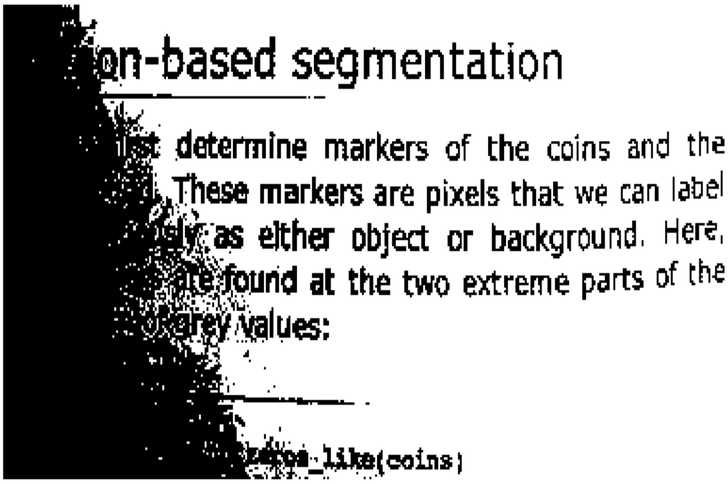


n-based segmentation

determine markers of the coins and the
These markers are pixels that we can label
as either object or background. Here,
round at the two extreme parts of the
values:

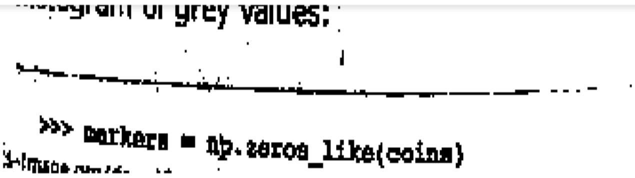
final_segmentation_opt.ipynb
File Edit View Insert Runtime Tools Help Last saved at 7:24 AM
Comment Share
+ Code + Text Connect Editing

```
text_threshold = filters.threshold_li(text)

image_show(text > text_threshold);
```




```
text_threshold = filters.threshold_local(text,block_size=51, offset=10)
image_show(text > text_threshold);
```

final_segmentation_opt.ipynb
File Edit View Insert Runtime Tools Help Last saved at 7:24 AM
Comment Share
+ Code + Text Connect Editing



Optimization in snake based contour segmentation:

```
points = generate_circle(200, [160, 120], 60)[:-1]
fig, ax = image_show(image_gray)
ax.plot(points[:, 0], points[:, 1], '--r', lw=3)
```

[<matplotlib.lines.Line2D at 0x7f131dbbc748>]

final_segmentation_opt.ipynb

File Edit View Insert Runtime Tools Help Last saved at 7:24 AM

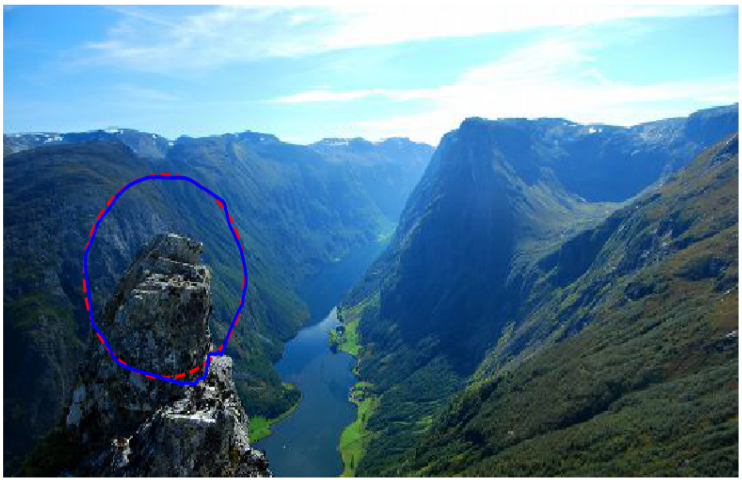
Comment Share

+ Code + Text

Connect Editing

Generating snake based on circle:

```
import skimage.segmentation as seg
snake = seg.active_contour(image_gray, points)
fig, ax = image_show(image)
ax.plot(points[:, 0], points[:, 1], '--r', lw=3)
ax.plot(snake[:, 0], snake[:, 1], '-b', lw=3);
```



final_segmentation_opt.ipynb

File Edit View Insert Runtime Tools Help Last saved at 7:24 AM

Comment Share

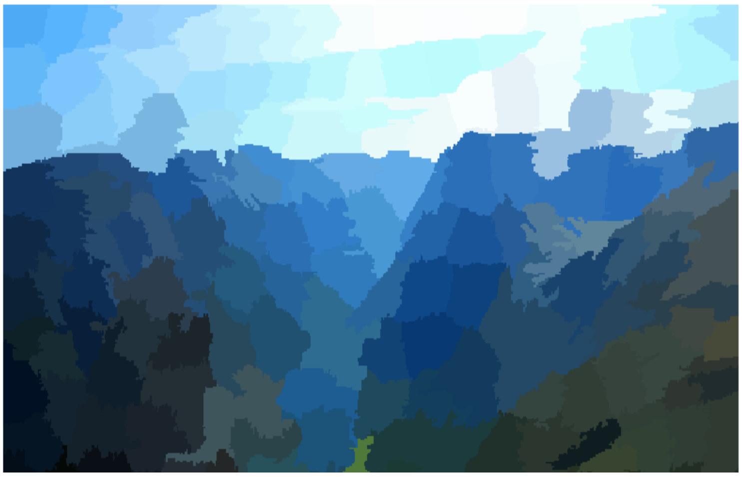
+ Code + Text

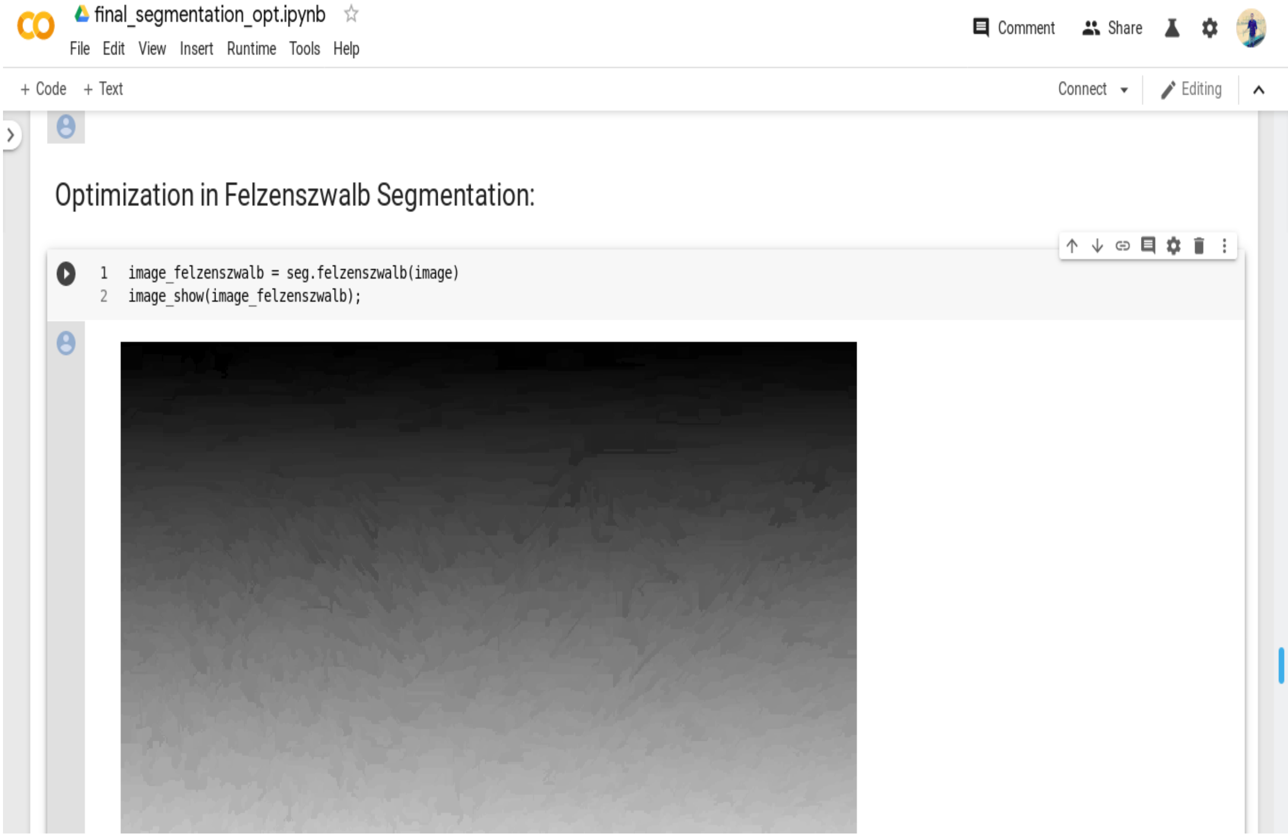
Connect Editing

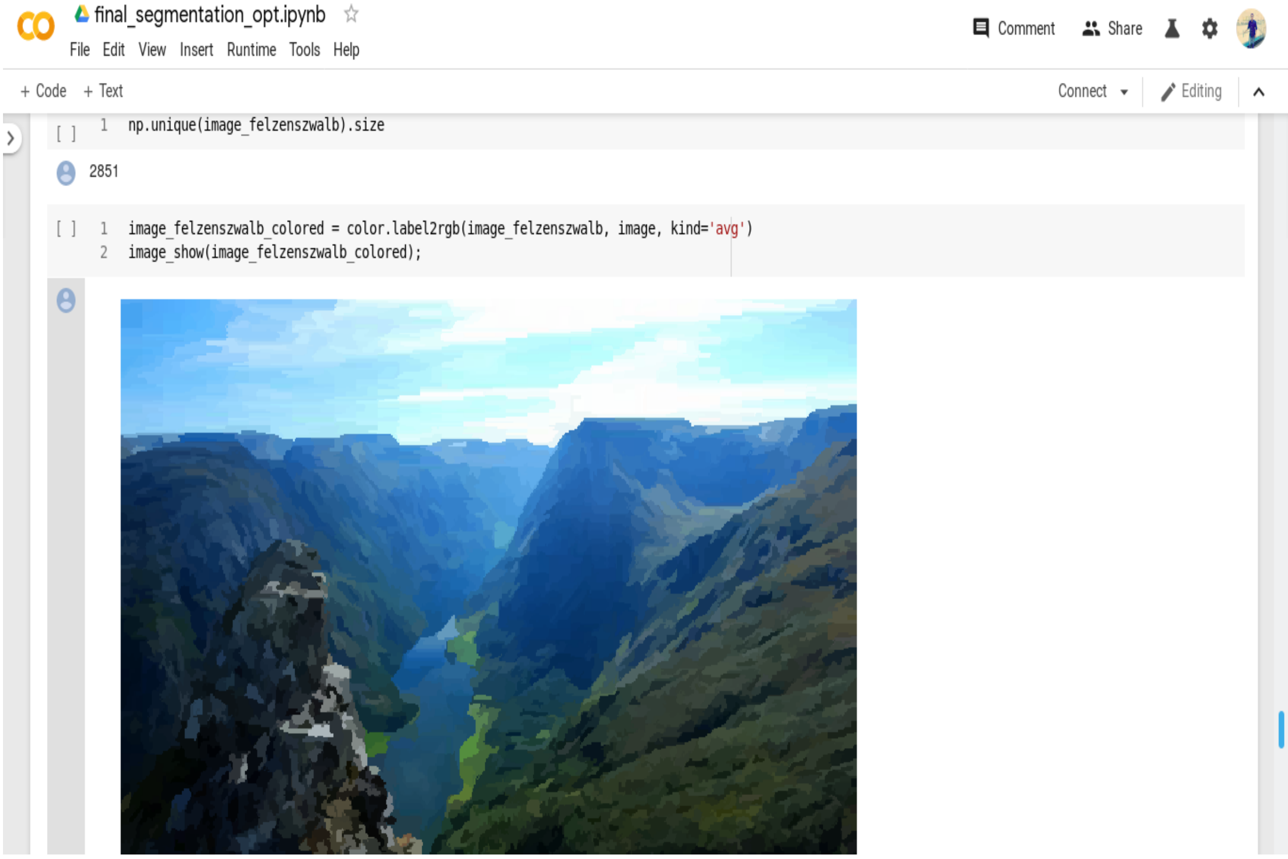
Optimization in Linear Iterative Clustering:

```
imagee = io.imread('https://www.gstatic.com/webp/gallery/1.jpg')
image_slice = seg.slic(imagee, n_segments = 255)
```

```
# label2rgb replaces each discrete label with the average interior color
image_show(color.label2rgb(image_slic, image, kind='avg'));
```




final_segmentation_opt.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
Connect
Editing
Optimization in Felzenszwalb Segmentation:
image_felzenszwalb = seg.felzenszwalb(image)
image_show(image_felzenszwalb);


final_segmentation_opt.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
Connect
Editing
np.unique(image_felzenszwalb).size
2851
image_felzenszwalb_colored = color.label2rgb(image_felzenszwalb, image, kind='avg')
image_show(image_felzenszwalb_colored);

final_segmentation_opt.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text Connect Editing

```
    {0.5, -0.5, -1.0,  0.0,  0.0, -1.0, -0.5,  0.5},
    {0.5,  1.0, 1.0,  -2.0, -2.0,  1.0,  1.0,  0.5},
    {0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0}
};

# Filter for absorbing edges and curves using negative intensity values
single_88_negative = {
    { -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0},
    { -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0},
    { -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0},
    { -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0},
    { -2.0, -3.0, -3.0, -4.0, -4.0, -3.0, -3.0, -2.0},
    { -1.0, -2.0, -2.0, -2.0, -2.0, -2.0, -2.0, -1.0},
    {  2.0,  2.0,  0.0,  0.0,  0.0,  0.0,  2.0,  2.0 },
    {  2.0,  3.0,  1.0,  0.0,  0.0,  1.0,  3.0,  2.0 }
};

image_object = filters.normalize(alpha = 0.01, beta = 0.1, scale = 'absolute')
image_object_gray = color.rgb2gray(image_object)
absolute_best_segmented_image = NULL
no_of_segments_in_abs_best = 1
iterations_till_now_for_t = 0

for(t in range(25, max(image_object_gray))):
    try:
        new_segmented_image = seg.inverse_gaussian_gradient(image_object_gray)
        selected_threshold = t
        iterations_till_now_for_s = 0
        for(s in range(1, 255)):
            single_88_positive_image = new_segmented_image.filter(single_88_positive)
            single_88_negative_image = new_segmented_image.filter(single_88_negative)
            new_segmented_image = min(single_88_positive_image, single_88_negative_image)
            new_segmented_image = seg.join_segmentations(image_object_gray, new_segmented_image)
            if(quality(new_segmented_image) > quality(absolute_best_segmented_image)):
                absolute_best_segmented_image = new_segmented_image
                no_of_segments_in_abs_best = s
                iterations_till_now_for_s += 1
        iterations_till_now_for_t += 1
    except:
        print("Error: Finding optimized image is not possible for given image.")
```

References & Bibliography

“To give credit someone is more difficult than to blame someone.”
- Unknown

[1] Fusion of Image Segmentation Algorithms using Consensus Clustering. Mete Ozay, Fatos T. Yarman Vural, Sanjeev R. Kulkarni, H. Vincent Poor. https://doi.org/10.1109/ICIP.2013.6738834. A version of the manuscript was published in ICIP 2013. Journal ref: 20th IEEE International Conference on Image Processing (ICIP), pp. 4049-4053, Melbourne, VIC, 15-18 Sept. 2013

[2] Combination of Hidden Markov Field and Conjugate Gradient for Brain Image Segmentation. EL-Hachemi Guerrout, Samy Ait-Aoudia, Dominique Michelucci, Ramdane Mahiou.

[3] An efficient iterative thresholding method for image segmentation. Dong Wang, Haohan Li, Xiaoyu Wei, Xiaoping Wang. https://doi.org/10.1016/j.jcp.2017.08.020

[4] A Replica Inference Approach to Unsupervised Multi-Scale Image Segmentation. Dandan Hu, Peter Ronhovde, Zohar Nussinov. https://doi.org/10.1103/PhysRevE.85.016101 Journal ref: Phys. Rev. E 85, 016101 (2012).

[5] Image Segmentation with Multidimensional Refinement Indicators. Hend Ben Ameur, Guy Chavent, Francois Clément, Pierre Weis. Report number: RR-7446, RR-7446. Journal ref: N° RR-7446 (2010).

[6] Optimization of Weighted Curvature for Image Segmentation. Noha El-Zehiry, Leo Grady.

[7] Automatic Spatially-Adaptive Balancing of Energy Terms for Image Segmentation. Josna Rao, Ghassan Hamarneh, Rafeef Abugharbieh. ACM Class: I.4.6.

[8] Image Segmentation by Discounted Cumulative Ranking on Maximal Cliques. Joao Carreira, Adrian Ion, Cristian Sminchisescu. Report number: TR-06-2010.

[9] Semantic Image Segmentation via Deep Parsing Network. Ziwei Liu, Xiaoxiao Li, Ping Luo, Chen Change Loy, Xiaoou Tang To be appered in ICCV 15.

[10] MBIS: Multivariate Bayesian Image Segmentation Tool. Oscar Esteban, Gert Wollny, Sub-rahmanyam Gorthi, Maria-J. Ledesma-Carbayo, Jean- Philippe Thiran, Andres Santos, Meritxell Bach-Cuadra. https://doi.org/10.1016/j.cmpb.2014.03.003

MSC Class: 62P10; 62F15. Journal ref: Comput. Meth. Prog. Bio. 115(2):76-94 (2014).

[11] Fast Planar Correlation Clustering for Image Segmentation. Julian Yarkony, Alexander T. Ihler, Charless C. Fowlkes. This is the extended version of a paper to appear at the 12th European Conference on Computer Vision (ECCV 2012).

[12] Particle methods enable fast and simple approximation of Sobolev gradients in image segmentation. Ivo F. Sbalzarini, Sophie Schneider, Janick Cardinale.

[13] Image segmentation by optimal and hierarchical piecewise constant approximations. M. Kharinov.

[14] Hierarchical pixel clustering for image segmentation. M. Kharinov. submitted to the 12 International Conference on Pattern Recognition and Information Processing May 28-30, 2014, Minsk, Belarus. Journal ref: Proc. of the 12th International Conference on Pattern Recognition and Information Processing (PRIP'2014), May 28-30, 2014, Minsk, Belarus, pp.103-107.

[15] Stable Segmentation of Digital Image. M. Kharinov. Journal ref: Proc. of the 22th Int. Conf. On Comp. Graphics and Vision (Graphicon 2012), Moscow: MSU, 01-05 Oct. 2012, pp. 208-213. (in Russian).

[16] Combinatorial Energy Learning for Image Segmentation. Jeremy Maitin-Shepard, Viren Jain, Michal Januszewski, Peter Li, Pieter Abbeel.

[17] Globally Optimal Joint Image Segmentation and Shape Matching Based on Wasserstein Modes. Bernhard Schmitzer, Christoph Schnörr. https://doi.org/10.1007/s10851-014-0546-8. Accepted by Journal of Mathematical Imaging and Vision, published online. Printed publication pending. MSC Class: 49Q10; 62H35.

[18] Parametric Image Segmentation of Humans with Structural Shape Priors. Alin-Ionut Popa, Cristian Sminchisescu.

[19] Image Segmentation Using Hierarchical Merge Tree. Ting Liu, Mojtaba Seyedhosseini, Tolga Tasdizen. https://doi.org/10.1109/TIP.2016.2592704. Journal ref: IEEE.Trans.Image.Processing 25 (2016) 4596-4607.

[20] lambda-Connectedness Determination for Image Segmentation. Li Chen. 36th Applied Image Pattern Recognition Workshop (AIPR 2007), October 2007, Washington, DC, USA. ACM Class: I.4.6

[21] A Fast Hierarchical Multilevel Image Segmentation Method using Unbiased Estimators 2007. Sreechakra Goparaju, Jayadev Acharya, Ajoy K. Ray, Jaideva C.

Goswami. submitted to "IEEE Transactions on Pattern Analysis and Machine Intelligence".

[22] Image Segmentation using Sparse Subset Selection. Fariba Zohrizadeh, Mohsen Kheirandishfard, Farhad Kamangar. IEEE Winter Conference on Applications of Computer Vision (WACV), 2018.

[23] Image Segmentation Using Subspace Representation and Sparse Decomposition. Shervin Minaee PhD Dissertation, NYU, 2018

[24] Exploring and Exploiting Diversity for Image Segmentation. Payman Yadollahpour PhD Thesis. For overall document size considerations the results in this appendix section have been moved to http://ttic.uchicago.edu/pyadolla/papers/thesis.pdf

[25] Fast, Exact and Multi-Scale Inference for Semantic Image Segmentation with Deep Gaussian CRFs. Siddhartha Chandra, Iasonas Kokkinos. https://github.com/siddharthachandra/gcrf

[26] PT-ResNet: Perspective Transformation-Based Residual Network for Semantic Road Image Segmentation. Rui Fan, Yuan Wang, Lei Qiao, Ruiwen Yao, Peng Han, Weidong Zhang, Ioannis Pitas, Ming Liu. 2019 IEEE International Conference on Imaging Systems and Techniques (IST).

[27] MIScnn: A Framework for Medical Image Segmentation with Convolutional Neural Networks and Deep Learning. Dominik Müller, Frank Kramer. https://github.com/frankkramer-lab/MIScnn

[28] Volume Preserving Image Segmentation with Entropic Regularization Optimal Transport and Its Applications in Deep Learning. Haifeng Li, Jun Liu, Li Cui, Haiyang Huang, Xue-cheng Tai.

[29] Resource Optimized Neural Architecture Search for 3D Medical Image Segmentation. Woong Bae, Seungho Lee, Yeha Lee, Beomhee Park, Minki Chung, Kyu-Hwan JungMICCAI(International Conference on Medical Image Computing and Computer Assisted Intervention). 2019 accepted.

[30] A Semi-Automated Usability Evaluation Framework for Interactive Image Segmentation Systems. Mario Amrehn, Stefan Steidl, Reinier Kortekaas, Maddalena Strumia, Markus Weingarten, Markus Kowarschik, Andreas Maier. Research article at the International Journal of Biomedical Imaging, Hindawi.

[31] Unsupervised Microvascular Image Segmentation Using an Active Contours Mimicking Neural Network. Shir Gur, Lior Wolf, Lior Golgher, Pablo Blinder.

[32] Active Contour Models for Manifold Valued Image Segmentation. Sumukh Bansal, Aditya Tatu.

[33] Multi-Task Attention-Based Semi-Supervised Learning for Medical Image Segmentation. Shuai Chen, Gerda Bortsova, Antonio Garcia-Uceda Juarez, Gijs van Tulder, Marleen de Bruijne. MICCAI 2019.

[34] Self-Adaptive 2D-3D Ensemble of Fully Convolutional Networks for Medical Image Segmentation . Maria G. Baldeon Calisto, Susana K. Lai-Yuen.

[35] Interactive segmentation of medical images through fully convolutional neural networks . Tomas Sakinis, Fausto Milletari, Holger Roth, Panagiotis Kor atis, Petro Kostandy, Kenneth Philbrick, Zeynettin Akkus, Ziyue Xu, Daguang Xu, Bradley J. Erickson.

[36] Learning a sparse database for patch-based medical image segmentation. Moti Freiman, Hannes Nickisch, Holger Schmitt, Pal Maurovich-Horvat, Patrick Donnelly, Mani Vembar, Liran Goshen. https://doi.org/10.1007/978-3-319-67434-6_6 Journal ref: Wu G., Munsell B., Zhan Y., Bai W., Sanroma G., Coupé P. (eds) Patch-Based Techniques in Medical Imaging. Patch-MI 2017. Lecture Notes in Computer Science, vol 10530. Springer, Cham.

[37] Scalable Neural Architecture Search for 3D Medical Image Segmentation Sungwoong Kim, Ildoo Kim, Sungbin Lim, Woonhyuk Baek, Chiheon Kim, Hyungjoo Cho, Boogeon Yoon, Taesup Kim

[38] Multi-scale guided attention for medical image segmentation. Ashish Sinha, Jose Dolz.

[39] The Chan-Vese Model with Elastica and Landmark Constraints for Image Segmentation. Jintao Song, Huizhu Pan, Wuanquan Liu, Zisen Xu, Zhenkuan Pan.

[40] A Novel Euler's ELASTICA based Segmentation Approach for Noisy Images via using the Progressive Hedging Algorithm. Lu Tan, Ling Li, Wanquan Liu, Jie Sun, Min Zhang.

[41] End-to-End Learned Random Walker for Seeded Image Segmentation. Lorenzo Cerrone, Alexander Zeilmann, Fred A. Hamprecht.

[42] Convolutional Random Walk Networks for Semantic Image Segmentation. Gedas Bertasius, Lorenzo Torresani, Stella X. Yu, Jianbo Shi.

[43] Learning of Image Dehazing Models for Segmentation Tasks. Sébastien de Blois, Ihsen Hedhli, Christian Gagné. EUSIPCO 2019

[44] Transformation Consistent Self-ensembling Model for Semi-supervised Medical Image Segmentation. Xiaomeng Li, Lequan Yu, Hao Chen, Chi-Wing Fu, Pheng-Ann Heng.

[45] Annotation-cost Minimization for Medical Image Segmentation using Suggestive Mixed Super- vision Fully Convolutional Networks. Yash Bhalgat, Meet Shah, Suyash Awate. Medical Imaging meets NeurIPS 2018 Workshop.

[46] Foreground Clustering for Joint Segmentation and Localization in Videos and Images. Abhishek SharmaIn Proceedings of NIPS 2018.

[47] Consistent estimation of the max-flow problem: Towards unsupervised image segmentation. Ashif Sikandar Iquebal, Satish Bukkapatnam.

[48] Convexity Shape Constraints for Image Segmentation (minimum cost multicut problem). Loic A. Royer, David L. Richmond, Carsten Rother, Bjoern Andres, Dagmar Kainmueller.

[49] A Time Series Graph Cut Image Segmentation Scheme for Liver Tumors. Laramie Paxton, Yufeng Cao, Kevin R. Vixie, Yuan Wang, Brian Hobbs, Chaan Ng.

[50] UNet++: A Nested U-Net Architecture for Medical Image Segmentation. Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, Jianming Liang. Accepted by 4th Deep Learning in Medical Image Analysis (DLMIA) Workshop.

[51] FPGA based Parallelized Architecture of Efficient Graph based Image Segmentation Algorithm. Roopal Nahar, Akanksha Baranwal, K. Madhava Krishna.

[52] Dominant Sets for "Constrained" Image Segmentation. Eyasu Zemene, Leulseged Tesfaye Alemu, Marcello Pelillo. Accepted at ECCV 2016.

[53] Interactive Image Segmentation Using Constrained Dominant Sets. Eyasu Zemene, Marcello Pelillo.

[54] Structured Learning of Tree Potentials in CRF for Image Segmentation Fayao Liu, Guosheng Lin, Ruizhi Qiao, Chunhua Shen. IEEE Transactions on Neural Networks and Learning Systems.

[55] Camera-trap images segmentation using multi-layer robust principal component analysis. Jhony-Heriberto Giraldo-Zuluaga, Alexander Gomez, Augusto Salazar, Angélica Diaz-Pulido https://doi.org/10.1007/s00371-017-1463-9. This is a pre-print of an article published in The Visual Computer. The nal authenticated version is available online at: https://doi.org/10.1007/s00371-017-1463-9. Journal ref: The Visual Computer, 1-13 (2017).

[56] Systematic study of color spaces and components for the segmentation of sky/cloud images. Soumyabrata Dev, Yee Hui Lee, Stefan Winkler. Published in Proc. IEEE International Conference on Image Processing (ICIP), Oct. 2014.

[57] Consensus Based Medical Image Segmentation Using Semi-Supervised Learning And Graph Cuts. Dwarikanath Mahapatra.

[58] Incorporating prior knowledge in medical image segmentation: a survey. Masoud S. Nosrati, Ghassan Hamarneh.

[59] Image Colour Segmentation by Genetic Algorithms. Vitorino Ramos, Fernando Muge. ACM Class: I.2; I.5 Journal ref: RecPad 2000 - 11th Portuguese Conf. on Pattern Recognition, in Aurelio C. Campilho and A.M. Mendonca (Eds.), ISBN 972-96883-2-5, pp. 125-129, Porto, Portugal, May 11-12, 2000.

[60] Artificial Neoteny in Evolutionary Image Segmentation. Vitorino Ramos. ACM Class: I.2; I.5 Journal ref: SIARP 2000 - 5th IberoAmerican Symp. on Pattern Rec., F. Muge, Moises P. and R. Caldas Pinto (Eds.), ISBN 972-97711-1-1, pp. 69-78, Lisbon, Portugal, 11-13 Sep. 2000.

[61] Multi-Atlas Segmentation of Biomedical Images: A Survey. Juan Eugenio Iglesias, Mert Rory Sabuncu.

[62] A Comparison of Nature Inspired Algorithms for Multi-threshold Image Segmentation. Valentín Osuna-Enciso, Erik Cuevas, Humberto Sossa. Journal ref: Expert Systems with Applications, Volume 40, Issue 4, March 2013, Pages 1213-1219.

[63] Multilevel Threshold Based Gray Scale Image Segmentation using Cuckoo Search. Sourav Samantaa, Nilanjan Dey, Poulami Das, Suvojit Acharjee, Sheli Sinha Chaudhuri ICECIT2012,Anatapur, India.

[64] Image Segmentation using Multi-Threshold technique by Histogram Sampling. Amit Gurung, Sangyal Lama Tamang.

[65] Seeking multi-thresholds for image segmentation with Learning Automata (Read Abstract also) Journal ref: Machine Vision and Applications 22 (5), (2011), pp. 805-818 Comments: 22 Pages.

[66] Mixed Robust/Average Submodular Partitioning: Fast Algorithms, Guarantees, and Applications to Parallel Machine Learning and Multi-Label Image Segmentation. Kai Wei, Rishabh Iyer, Shengjie Wang, Wenruo Bai, Jeff Bilmes.

[67] Identifying Reliable Annotations for Large Scale Image Segmentation Alexander Kolesnikov, Christoph H. Lampert.

[68] SegSALSA-STR: A convex formulation to supervised hyperspectral image segmentation using hidden fields and structure tensor regularization. Filipe Condessa, Jose Bioucas-Dias, Jelena Kovacevic. This paper was submitted to IEEE WHISPERS 2015: 7th Workshop on Hyperspectral Image and Signal Processing: Evolution on Remote Sensing.

[69] Convex Color Image Segmentation with Optimal Transport Distances. Julien Rabin, Nicolas Papadakis.

[70] Semi-supervised Segmentation Fusion of Multi-spectral and Aerial Images. Mete Ozay https://doi.org/10.1109/ICPR.2014.659 Journal ref: Proc. 22nd International Conference on Pattern Recognition, pp. 3839-3844, Stockholm, 2014.

[71] Co-Sparse Textural Similarity for Image Segmentation. Claudia Nieuwenhuis, Daniel Cremers, Simon Hawe, Martin Kleinsteuber.

[72] Q-learning optimization in a multi-agents system for image segmentation. Issam Qa ou, Mohamed Sadgal, Abdelaziz Elfazziki.

[73] Multislice Modularity Optimization in Community Detection and Image Segmentation. Huiyi Hu, Yves van Gennip, Blake Hunter, Mason A. Porter, Andrea L. Bertozzi. To appear in IEEE International Conference on Data Mining PhD forum conference proceedings. https://doi.org/10.1109/ICDMW.2012.72

[74] Iterative graph cuts for image segmentation with a nonlinear statistical shape prior. Joshua C. Chang, Tom Chou. https://doi.org/10.1007/s10851-013-0440-9 . Revision submitted to JMIV (02/24/13).

[75] Non-parametric convolution based image-segmentation of ill-posed objects applying context window approach. Upendra Kumar, Tapobrata Lahiri, Manoj Kumar Pal.

■■■■■