# Image segmentation Using heuristic search algorithms

Parsa Tasbihgou
Prof. Sajedi
University of Tehran, department of
Mathematics and computer science
Fall 2022

# Introduction

In this paper we propose a bio inspired algorithm to find suitable segmentations for images. We have tested our algorithm on “ALL IDB” dataset, containing cell images for acute lymphoblastic leukemia detection.

There are many different methods to measure the quality of a segmentation, and using any of them, image segmentation can be viewed as an optimization problem, where we are trying to find a segmentation maximizing the quality measure. In this paper we have chosen Sum of Squared Errors (SSE) as quality measure, and we try to minimize this error, consequently maximizing Peak Signal to Noise Ration (PSNR).

Viewing segmentation as an optimization problem allows us to define a search space and representation method for each candidate segmentation.
Assuming we want to reduce the number of colors used in an image to C, it means we want to divide the pixels from the original image into C clusters and assign some color to each cluster, so that by recoloring the pixels according to the new clustering and color assignment, the recolored image is as close to the original as possible (maximizing the segmentation quality measure). So we can represent any feasible segmentation by the center of its clusters as an C-tuple where each member is itself a 3-tuple, for each pixel has 3 values for RGB intensities. therefor our search space is all C-tuples of pixel values.

The proposed algorithm consists of three main sections:

1. Image histogram analisys.
2. Particle swarm search
3. Solution refinement by exhaustive search

The base method used in this algorithm is particle swarm optimization, because it

offers good exploration of the search space without diverging too much, and using very small amount of resources and many parameters for specifically tuning the algorithm to our problem. Specially in this problem at the time of initializing the particles we have some good candidate positions for the particles that we extract from the histogram.

## Histogram Analisys

Since the search space of this problem is considerably large and the size of the input is also large meaning the operations that require working with the input i.e calculating fitness of a candidate solution are complex and time consuming, random initialization of the particles is not optimal and in fact produces very poor results.

By analyzing the histogram of the original image, we can find peaks for each color. By combining these peaks a set of candidate starting points is created, that in fact contains decant candidates. How ever it is easy to see that not all combination are suitable candidates since their corresponding peaks could be from different colors. To eliminate bad candidates each pixel of the original image is assigned to the candidate closest to it by euclidian distance, and C candidates with highest number of pixels assigned to them are chosen as most suitable and are used to initialize the particles.

## Particle Swarm Search

Using the candidates from last section we create an initial swarm of particles in the search space, consisting of the C-tuple of the candidates and some points randomly selected around this point. The reason for this assignment is that it is most likely that a very good segmentation can be found close to the group of

pixels with highest recurrence in the histogram.
After this initialization a normal PSO algorithm is used to further explore the neighborhood of the candidate solutions. A key characterization of the PSO algorithm that we used is higher-than-usual coefficient for moving towards the global best particle, this is because we are trying to decrease exploration and increase exploitation of PSO algorithm since the initial particles are most likely very close to a good solution.

## Solution Refinement

After the particle swarm search is finished the particle with lowest SSE (highest PSNR) is chosen and a hill climb search is preformed to find the optimal solution in its neighborhood. The search method that we have used first tries all $2^{3 \cdot C}$ possible directions from a given point and finds the optimum movement vector, then moves the point in the direction of the movement vector as long as the destination point is a better point than the starting point.

This search has shown to be extremely effective, and in 95% of cases considerably improves the solution. Also by implementing this search we have been able to reduce the number of iterations for the PSO search and maintain solution quality, meaning that the required running time to achieve a solution with a certain accuracy is reduced, equivalently it means that in the same running time as before the algorithm can achieve better results

## Implementation

An optimization that has been used to reduce running time of the algorithm is compressing the input image, so that fitness of a candidate solution can be calculated more quickly. To preform this compression, we define a neighborhood

radius and replace the pixel value of the pixel in the center of the neighborhood with the mean pixel value of the entire neighborhood, by doing this we can still get a good estimate of the fitness (or error) for each candidate solution for a fraction of complexity and time of calculating the exact fitness, and since the PSO method only relies on qualitative difference of particles and not their quantitative difference, it is expected that this variation will not effect the quality of the search.

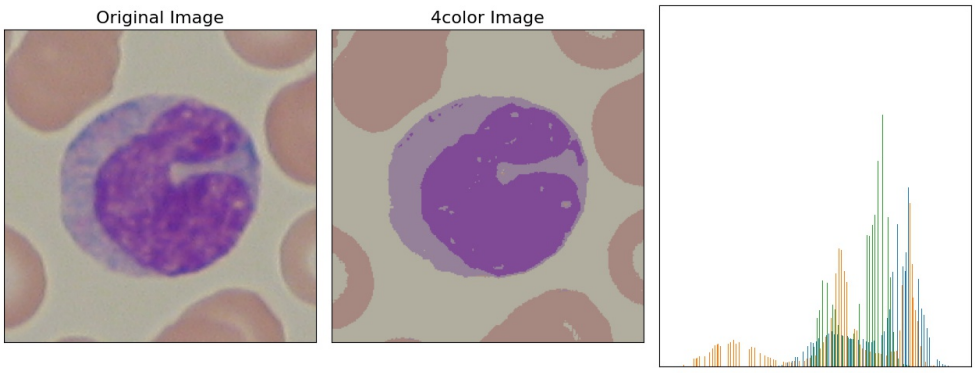
For this implementation of this algorithm we have used python and test have been conducted using cpython to execute the code, each execution takes about 1:20 to 2 minutes depending on the number of PSO iterations and number of clusters (number of colors in the final image). By using pypy to execute the code it is expected that each execution takes 10 to 15 less time to complete, however some of the python modules used in this code are not compatible with pypy. These modules can be replaced and are not used in functionality of the code, they are used to handle input and output. And a c++ implementation could run up to 200 times faster.

## Conclusion

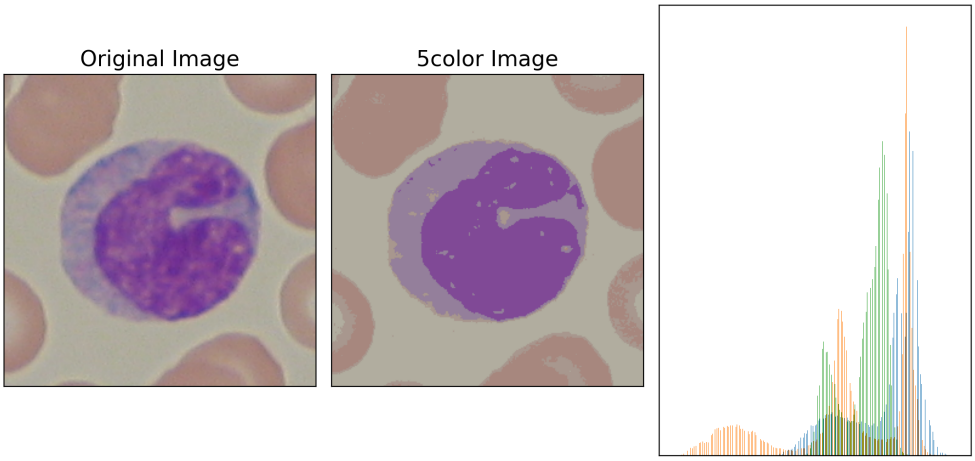
Using the particle swarm optimization method for image segmentation, is very effective and we have been able to achieve good results (in some cases better than the results in Dhal paper). However due to computational complexity of the problem (specially fitness computations) and the limited resources available we have bounded the number of PSO iterations to 30 for our experiments, and only a handful of test are conducted with more iterations (100, 150 and 250) where results has been considerably better, showing this method is promising as a concept.

# Experimental results

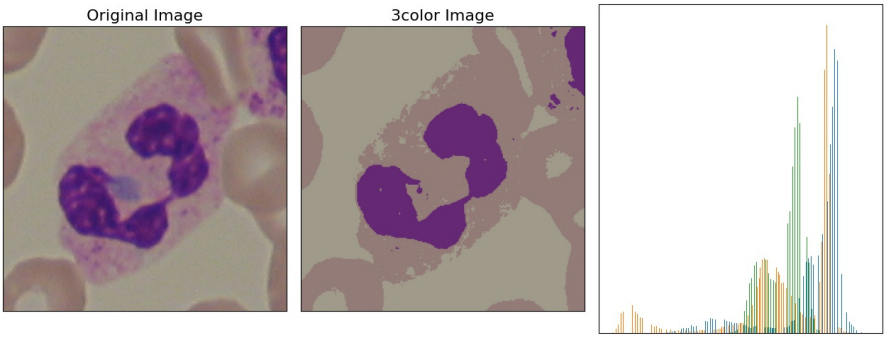
The following are some of the images from the “ALL IDB” dataset that are segmented using this algorithm, and the average fitness and PSNR.




1(a) with 4 colors
SSE: 611859.1443674266
PSNR: 31.961132120662906

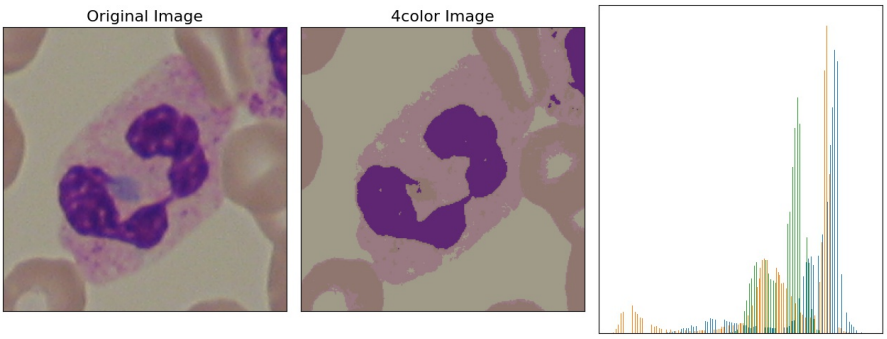



1(a) with 5 colors
Fitness: 517069.47060203686
PSNR: 33.334329826687146



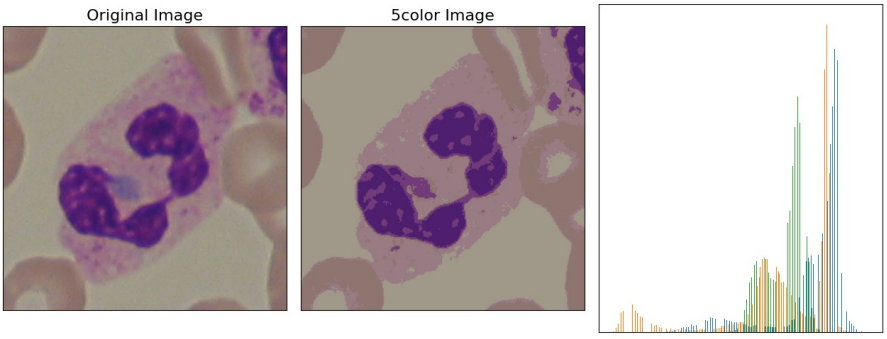

SSE: 812746.9569594868

PSNR: 29.16716376090282




SSE: 658206.8765114485

PSNR: 30.74329596125989




SSE: 591386.9639313993

PSNR: 32.31179126507288

# Average accuracy

| | 3 colors | 4 colors | 5 colors |
|---|---|---|---|
| **SSE** | 729851.7187099 | 29.8732551051627 | 576052.661078807 |
| **PSNR** | 608335.344118058 | 31.6007249482015 | 32.2616271182041 |

The recolored images and accuracy results for all images in the dataset, colored with 3, 4 and 5 colors are presented in the “results” file.
python codes in the folder are used to run multiple processes, “final-code.py” is the final code for this algorithm.