

IMAGE SEGMENTATION USING GENETIC ALGORITHM

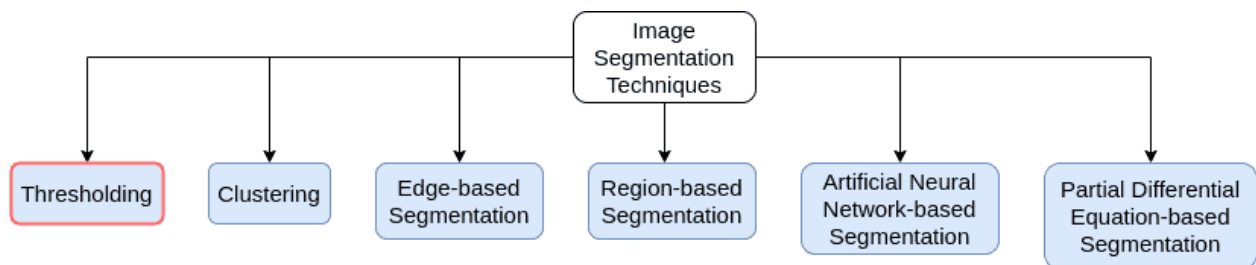
SUBMITTED BY: SANYA MAHAJAN 102103750

Repository: https://github.com/sanya-mahajan/AI_image_segmentation_genetic_Algorithm

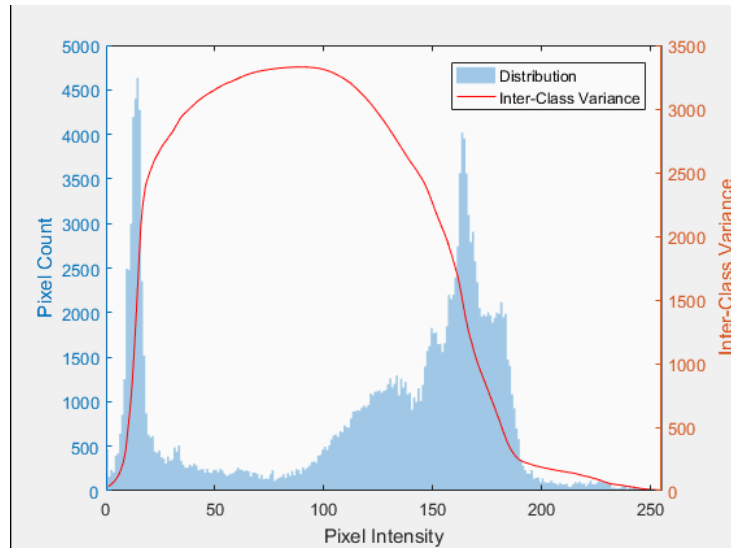
Introduction

Image segmentation refers to the class of algorithms that partition the image into different segments or groups of pixels. In that sense, image thresholding is the simplest kind of image segmentation because it partitions the image into two groups of pixels — white for foreground, and black for background.

The figure below shows different types of segmentation algorithms:




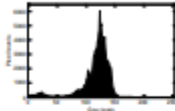

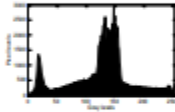
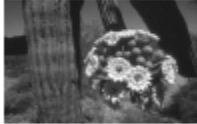
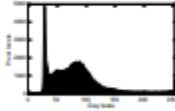

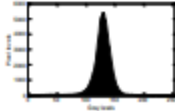
A genetic algorithm is used to search for near optimal solutions when no deterministic method exists or if the deterministic method is computationally complex, GA is a population based algorithm (i.e. it generates multiple solutions each iteration). The number of solutions per iteration is called population size. Each solution is represented as chromosome and each chromosome is built up from genes.



Otsu's method visualization- source: Wikipedia

In this project, the aim is to segment the given image using the genetic algorithm in order to compute the optimal threshold value. Otsu's method is adopted for computing the fitness value which is elaborated upon subsequently. Python libraries such as [OpenCV](#) , Matplotlib are utilized for visualization.

Table 1. Test images and the gray scale histogram

Image		Histogram
Jet		
Train		
Flower		
Snake		

Literature Survey

Otsu's approach

The method processes image histogram, segmenting the objects by minimization of the variance on each of the classes. The histogram of such image contains two clearly expressed peaks, which represent different ranges of intensity values.

The core idea is separating the image histogram into two clusters with a threshold defined as a result of Method adopted :maximize the between-class variance using the expression:

$$\sigma_b^2(t) = w_1(t)w_2(t)[\mu_1(t) - \mu_2(t)]^2, \text{ where } \mu_i \text{ is a mean of class } i.$$

To get the total variance we simply need to summarize the within class and between-class variances:

$$\sigma_T^2 = \sigma_w^2(t) + \sigma_b^2(t), \text{ where } \sigma_b^2(t) = w_1(t)w_2(t)[\mu_1(t) - \mu_2(t)]^2.$$

Thus, the general algorithm's pipeline for the between-class variance maximization option can be represented in the following way:

1. calculate the histogram and intensity level probabilities
2. initialize $w_i(0), \mu_i(0)$
3. iterate over possible thresholds: $t = 0, \dots, \text{max_intensity}$
 - update the values of w_i, μ_i , where w_i is a probability and μ_i is a mean of class i
 - calculate the between-class variance value $\sigma_b^2(t)$
4. the final threshold is the maximum $\sigma_b^2(t)$ value

Kapur's Entropy

The entropy of an image is a measure of the randomness or unpredictability of the intensity values in the image. The formula for entropy is given by:

$$H = - \sum p(i) \log_2 p(i)$$

where $p(i)$ is the probability of intensity level i occurring in the image.

The Kapur algorithm works by computing the histogram of the image, which represents the distribution of intensity values. The algorithm then selects a threshold value that maximizes the difference in entropy between the segmented regions while minimizing the entropy of the entire image. The formula for the Kapur criterion is given by:

$$\Delta H = H(W) - \sum_{k=1,2} H(R_k)$$

where ΔH is the difference in entropy between the segmented regions, $H(W)$ is the entropy of the entire image, and $H(R_k)$ is the entropy of each segmented region.

The threshold value is selected by computing the Kapur criterion for each possible threshold value and selecting the threshold that maximizes the criterion. The algorithm then segments the image into two regions based on the selected threshold value.

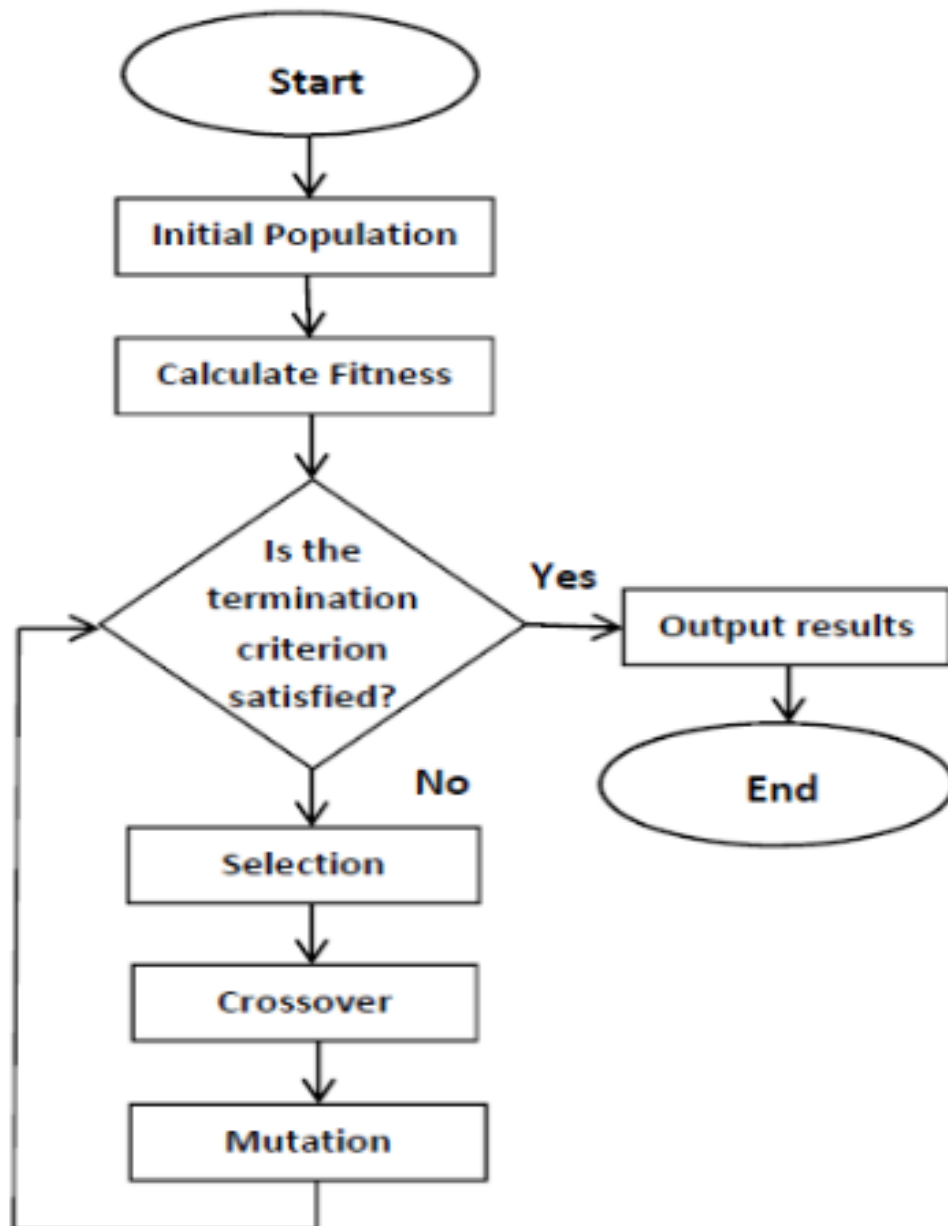
The Kapur algorithm can be applied iteratively to segment an image into multiple regions. In each iteration, the algorithm selects a threshold value that maximizes the difference in entropy between the segmented regions and segments the image into two regions based on the threshold. The process is repeated until the desired number of regions is obtained.

Algorithm

1. Load the image and convert it to grayscale using OpenCV's `cvtColor` function.
2. Compute the histogram of the grayscale image using NumPy's histogram function.
3. Design fitness function that takes a threshold and the histogram as inputs and computes the between-class variance for that threshold. It returns the variance as the fitness score.
4. Define the genetic algorithm function. The genetic algorithm takes the histogram as input and has the following parameters:
 - `population_size`: The number of individuals in each generation of the genetic algorithm.
 - `num_generations`: The number of generations to run the genetic algorithm for.
 - `mutation_rate`: The probability that a mutation will occur during the reproduction phase of the genetic algorithm.
5. The genetic algorithm initializes a population of `population_size` individuals. Each individual is a random threshold value between 0 and 255.
6. For each generation, the genetic algorithm evaluates the fitness of each individual using the fitness function.
7. The fittest individuals from the current generation are selected as parents for the next generation. The number of fittest individuals selected is set to be 20% of the total population size.
8. Offspring are generated by randomly selecting two parents from the fittest individuals and taking the average of their threshold values. The offspring replace the original population.
9. Each offspring has a chance of being mutated by changing its threshold value to a random value between 0 and 255 with probability `mutation_rate`.
10. After `num_generations` generations, the threshold value of the fittest individual is returned as the optimal threshold.
11. Threshold the grayscale image using OpenCV's threshold function with the optimal threshold value and display the resulting binary image.

Depending on the image and the desired output, different values for `population_size`, `num_generations`, and `mutation_rate` may be needed to find the optimal threshold.

Flowchart



Results

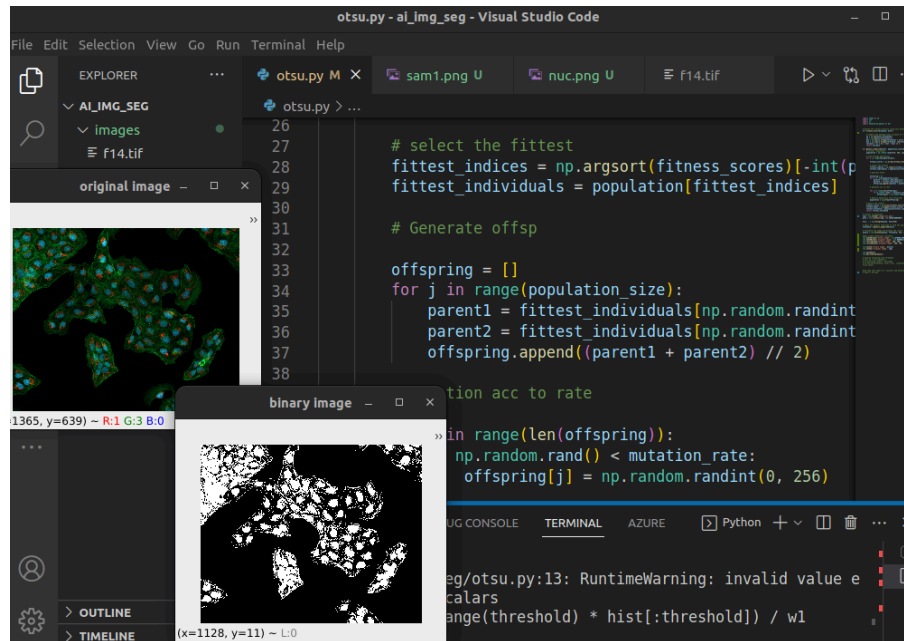


Fig 1: Otsu's method used for single level thresholding

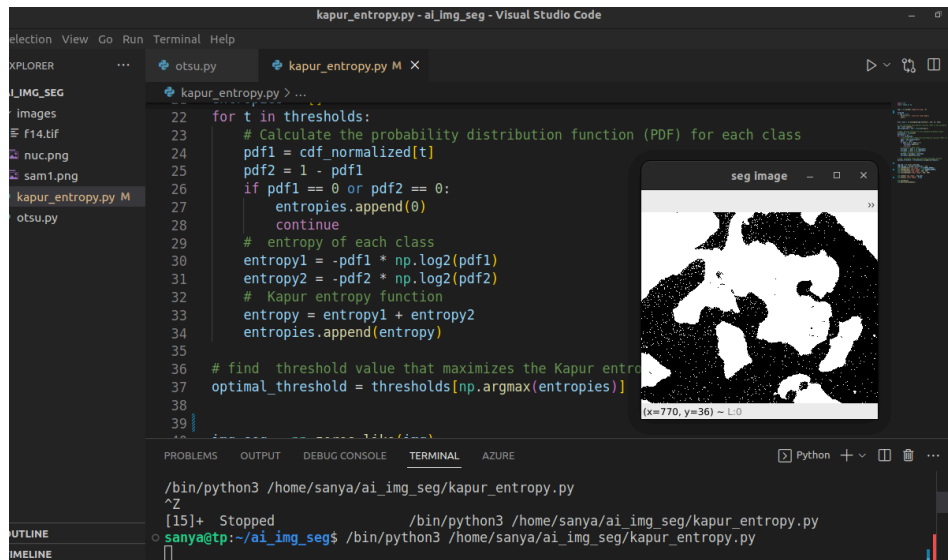


Fig 2: Results of Kapur entropy function

Future Scope

1. Improving the fitness function: While Otsu's method is a widely used technique for thresholding, there may be other methods that can provide better results for certain types of images. Hence, exploring and implementing other techniques can be a future scope.
2. The genetic algorithm used in the code has several parameters such as population size, number of generations, mutation rate, etc. These parameters can be tuned further to improve the performance of the algorithm.
3. Integration with other image processing techniques

Literature References :

http://www.ripublication.com/ijaerdoi/2015/ijaerv10n9_20.pdf

<https://www.hcu.edu.iq/modules/research/res/47655318309301696.pdf>

<https://learnopencv.com/otsu-thresholding-with-opencv/>

https://en.wikipedia.org/wiki/Otsu%27s_method#Python_implementation

<https://www.mdpi.com/1099-4300/21/3/318>

<https://sciresol.s3.us-east-2.amazonaws.com/IJST/Articles/2016/Issue-12/Article51.pdf>