Hybridization of Divide-and-Conquer Technique And Metaheuristic Algorithm For Better Contrast Enhancement In Medical Images

Ali Ait Houssa

LICENCE MIASI

Project Supervised By Pr Nour Eddine Alaa

The Plan

1. General Introduction
2. Representation Of Digital Images
3. Convolution Product And Kernels
4. Image Enhancement
5. Divide And Conquer Method
6. Metaheuristics
7. Used Tools
8. Conclusions And Perspectives

General Introduction

- What's a numerical image ?
- What is a convolution product ?
- What does it mean to enhance an image ?
- How can we use Divide And Conquer algorithm to enhance an image ?
- What are Metaheuristics and how can we use them to enhance images ?
- Could we perform better ?

Representation Of Digital Images: (Grayscale)

Grayscale images are 2D matrices of dimension nxm (width x height) of values ranging from 0 to 255 representing the shades of gray.

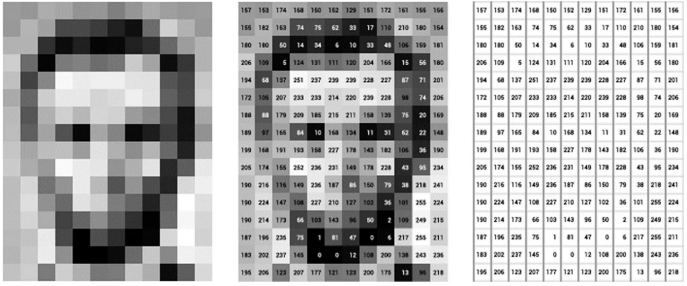


Figure: An Image With Its Values Over It

Representation Of Digital Images: (RGB)

RGB images are represented as 3D matrices (or tensors), where each 2D slice corresponds to one of the three color channels: Red, Green, and Blue. Each pixel in the image is a combination of intensity values from these three channels, determining its final color.

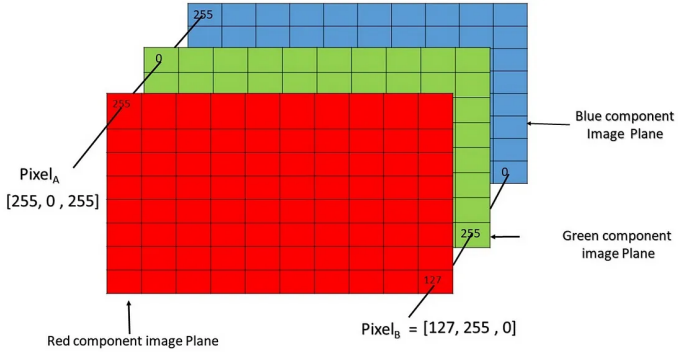


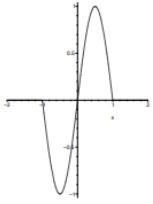
Figure: Representation of RGB images

Convolution Product

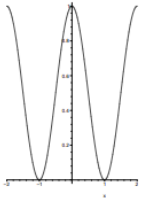
In mathematics , convolution is a mathematical operation on two functions (f and g) that produces a third function ($f * g$). The convolution of two functions f and g is defined as:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau)\, d\tau \tag{1}$$

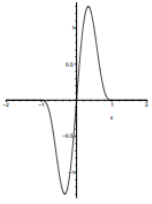
Example:



(d) $f = \chi_{[-1,1]}(x)\sin(\pi x)$.



(e) $w = \cos^2(\frac{\pi x}{2})$



(f) $f * w$

Convolution Product

In image processing, convolution applies filters to enhance features or extract information by sliding a kernel (small squared matrices) over the image, computing the sum of element-wise products.

Examples of kernels:

1. **Edge Detection Kernel (Sobel Operator - Vertical)**:

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

2. **Emboss Kernel**:

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

Convolution Product

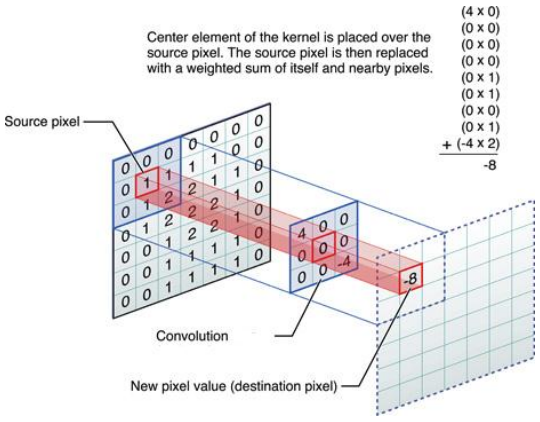


Figure: Illustration Of The Convolutional Product

Convolutional Product

Applying the kernel examples from before on an image:

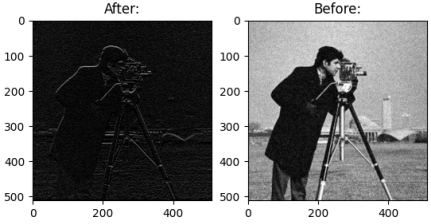


Figure: Sobel Kernel effect on an image.

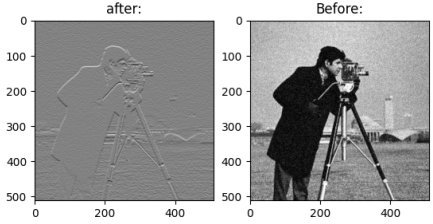


Figure: Emboss Kernel effect on an image.

Image Enhancement

The goal of the Divide And Conquer algorithm on enhancing images is to improve the image quality so that the processed image is better than the original image for further processing. To measure the enhancement we use the Effective Measure Enhancement function (EME) that is expressed as it follows:

$$EME(U, M, N) = \frac{20}{M \times N} \sum_{i=0}^{M} \sum_{j=0}^{N} \log(\frac{I_{maxij}}{I_{minij}}) \tag{2}$$

A higher value of the *EME* function for a given image indicates greater enhancement.

Divide And Conquer Method

In computer science, divide and conquer is an algorithm design paradigm. This approach involves recursively splitting a problem into two or more smaller, similar sub-problems until they become simple enough to solve directly. The solutions to these sub-problems are then combined to form a solution to the original problem.

Divide And Conquer Method

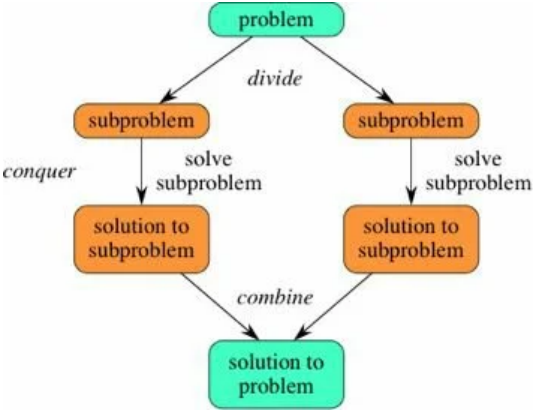


Figure: Illustration Of Merge Sort Algorithm that uses Divide And Conquer method

Divide And Conquer Method

Previous studies in decomposition models have highlighted the distinction between the low-frequency (cartoon) and high-frequency (edges and texture) components of an image. Low-frequency components represent smoothed areas, while high-frequency components contain edges and fine details.

Example:

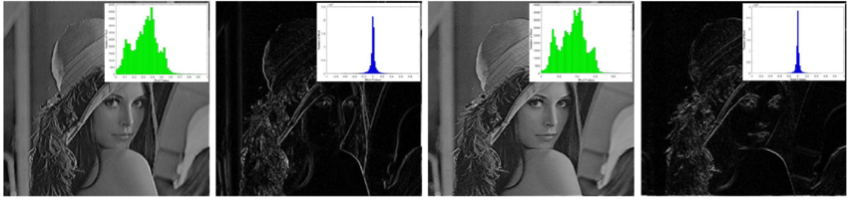


Figure: Illustration Of Image high and low frequencies

Divide And conquer Method

Subsequently, each component is processed separately. To achieve this, an observed image U is decomposed into distinct low and high-frequency components through convolution with some predefined linear filters h_i:

$$h_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \end{bmatrix} \quad h_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix} \quad h_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad h_4 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{3}$$

Once the decomposition process of the image U is obtained. V an image candidate is constructed through a linear mapping of the U_i (composed images with h_i kernels) with some weights w_i:

$$V = \sum_{i=1}^{4} w_i U_i \tag{4}$$

Metaheuristics

Metaheuristics are strategies for efficiently solving complex optimization problems. Inspired by natural processes, examples include:

Algorithm	Inspiration	Key Features
Genetic Algorithm	Natural Selection	Population-based, crossover, mutation
Simulated Annealing	Annealing in metallurgy	Cooling schedule, probabilistic acceptance
Ant Colony Optimization	Ant foraging behavior	Pheromone trails, path finding
Particle Swarm Optimization	Social behavior of birds	Swarm intelligence, velocity and position updates

Table: Examples of Metaheuristic Algorithms

Metaheuristics

In this case, we will use the following metaheuristics, which are mostly population based:

- Gravitational Search Algorithm (GSA)
- Flower Pollination Algorithm (FPA)
- Sine Cosine Algorithm (SCA)
- Particle Swarm Optimization (PSO)

Metaheuristics: Gravitational Search Algorithm

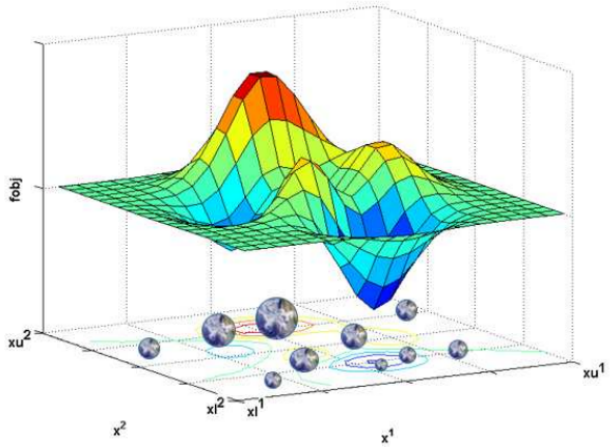


Figure: Mass calculation for objects in a maximization problem. The better the objective function value is, the bigger the mass is.

Metaheuristics: Gravitational Search Algorithm

Gravitational Search Algorithm:

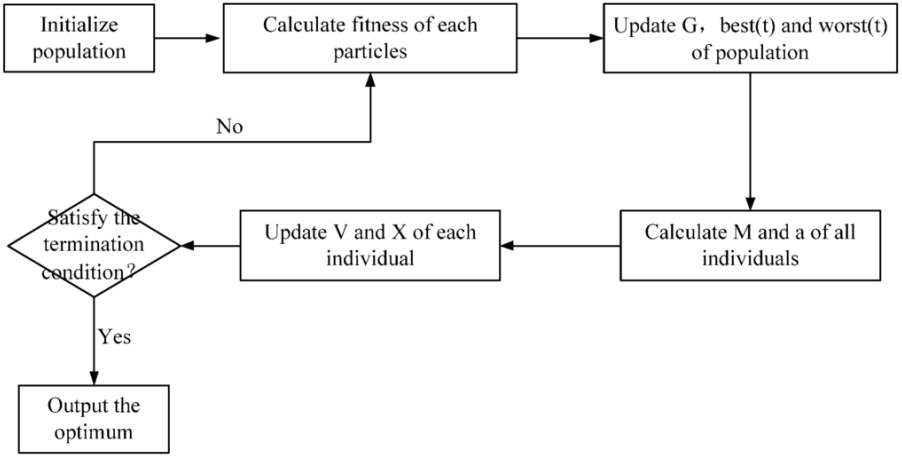


Figure: GSA Flowchart.

Metaheuristics: Flower Pollination Algorithm

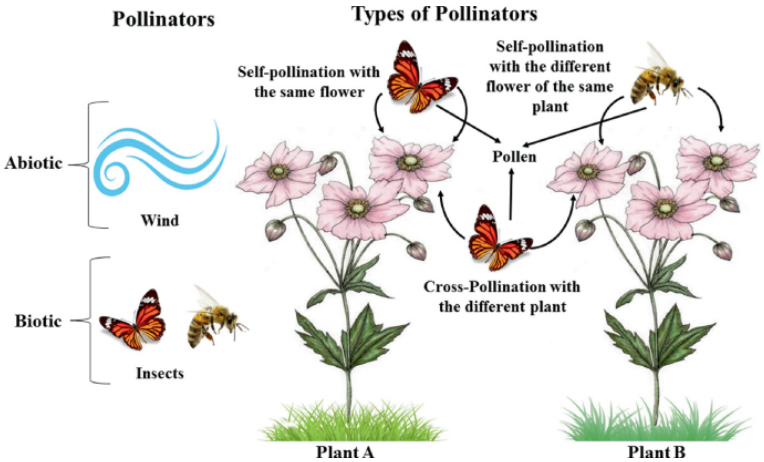


Figure: Flower Pollination.

Metaheuristics: Flower Pollination Algorithm

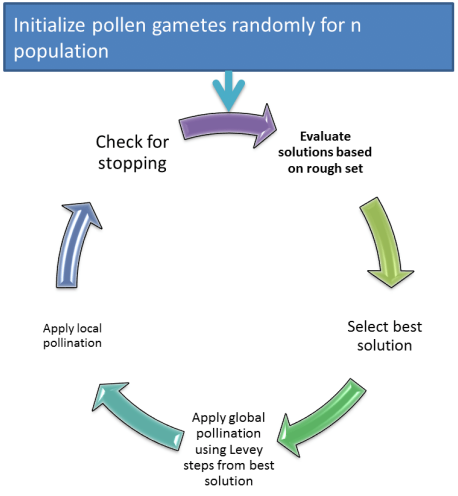


Figure: Flower Pollination Flowchart.

Metaheuristics: Sine Cosine Algorithm

The Sine Cosine Algorithm (SCA) is a population-based optimization method that uses sine and cosine functions to iteratively adjust candidate solutions. It balances exploration and exploitation through randomized updates and is known for its simplicity and effectiveness in solving complex optimization problems. It moves agents of the population based on the following equations:

$$X_i^d(t+1) = X_i^d(t) + r_1(t).sin(r_2).|r_3.P^d(t) - X_i^d(t)| \tag{5}$$

$$X_i^d(t+1) = X_i^d(t) + r_1(t).cos(r_2).|r_3.P^d(t) - X_i^d(t)| \tag{6}$$

Metaheuristics: Sine Cosine Algorithm

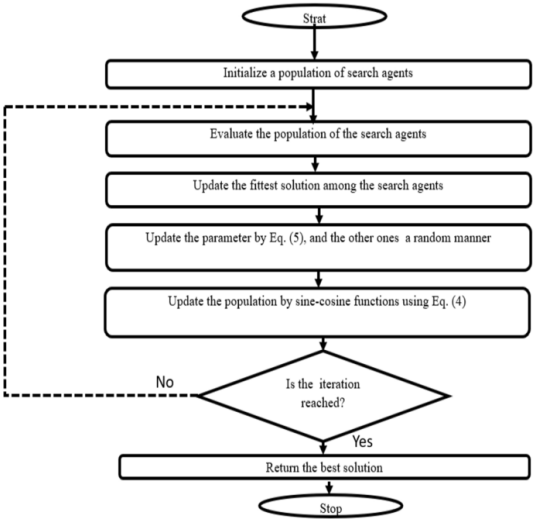


Figure: Sine Cosine Algorithm Flowchart.

Metaheuristics: Particle Swarm Optimization

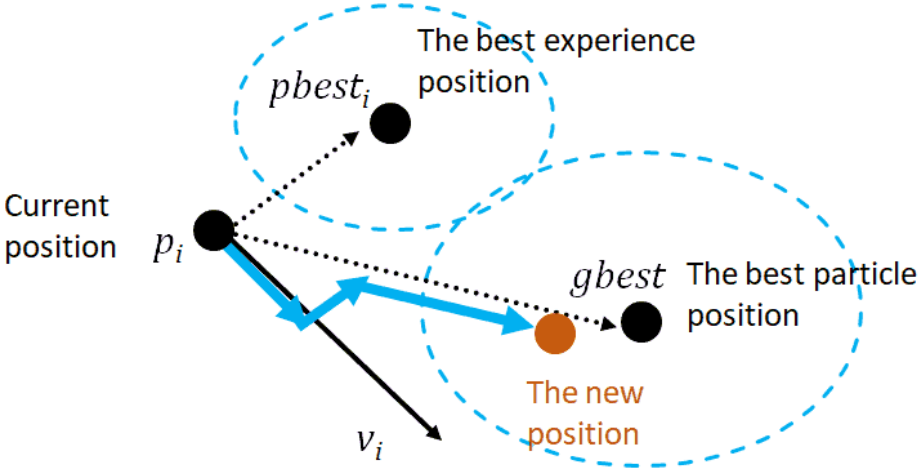


Figure: Particle Swarm Optimisation.

Metaheuristics: Particle Swarm Optimization

laximum value of search scope.

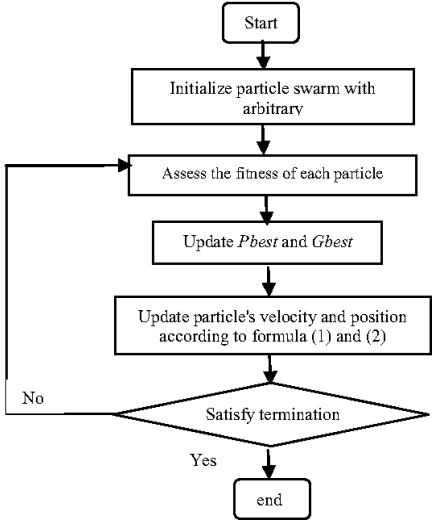
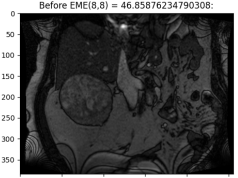
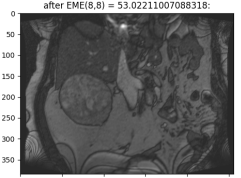


Figure 2. Basic PSO Algorithm Flowchart

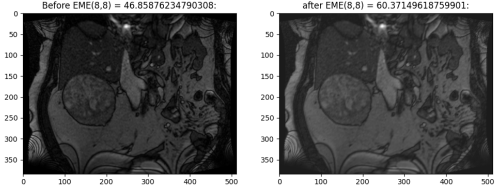
Python Tools Used:

Results of PSO:

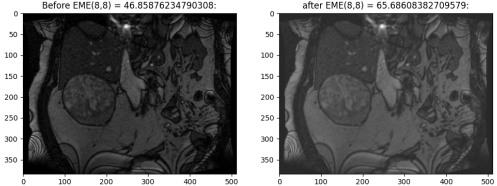




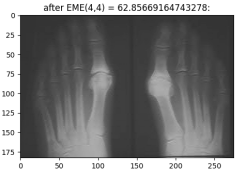
Results of FPA:

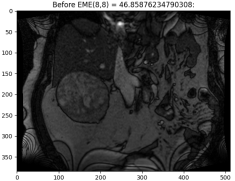


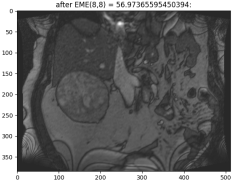
Results of SCA:



Results of GSA:







Conclusions and perspectives

- 1) The Divide and Conquer method is highly effective for image enhancement.
- 2) Metaheuristic algorithms significantly improve image quality.
- 3) Future research could explore new applications in differential equations, digital image processing, and artificial intelligence.


THANK YOU
FOR YOUR
ATTENTION