Image reconstruction with superpixels and simulated annealing

Problem Solving and Bayesian Inference

PEDRO ALVARADO GARCIA

June 2024

Resumen

The project involves reconstructing images from the superpixels of a set of images that do not contain the image being recreated. The SLIC (Simple Linear Iterative Clustering) algorithm is used to obtain the superpixels of the images. The reconstruction is carried out using Simulated Annealing, with the aim of minimizing the mean squared error (or maximizing the structural similarity index measure) between the target image and the reconstructed image.

Figure 1: Set of images from which superpixels were obtained

Figure 2: Image to reconstruct

Figure 3: Image reconstructed from the superpixels of the image set (Figure 1)

1 Introduction

In the field of digital image processing, image reconstruction represents an ongoing challenge, especially when it comes to preserving specific details and the structural integrity of the original images. This project addresses the challenge of reconstructing images using an innovative technique based on the use of superpixels, which are groupings of neighboring pixels in an image that share similar characteristics. The use of superpixels allows for a more compact and efficient representation of visual information, which is crucial for processing large volumes of data efficiently.

The specific problem faced in this context is how to select and arrange these superpixels

from different images so that a new image can be reconstructed that is not only visually pleasing, but is also as faithful as possible to the original structure and appearance.

This project looks at this specific problem using the simulated annealing technique, an optimization method that mimics the metal cooling process to find low energy states in complex problems. Applied to image reconstruction, simulated annealing is used to adjust the position and selection of superpixels, optimizing image reconstruction in terms of visual fidelity and structural coherence.

2 Objectives

General objective
Develop a system for image reconstruction using superpixels extracted from images that are not the target image, using simulated annealing and machine learning techniques.

Specific objectives

1. Implement superpixel selection techniques. Implement a method for superpixel selection and positioning to reconstruct images with high visual precision.
2. Evaluate the efficiency and effectiveness of simulated annealing in image reconstruction. Analyze how simulated annealing handles superpixel placement to minimize reconstruction error and improve visual consistency and detail of the resulting images.

3 Methodology

1. Image preparation and superpixel segmentation
    - **Image selection.** Choose a set of images to extract superpixels. The images in this set should include colors similar to those of the image to be reconstructed, this to guarantee the generalization of the method.
    - **Image segmentation.** The SLIC (Simple Linear Iterative Clustering) algorithm is used to divide each selected image into superpixels. This step is crucial to ensure that each superpixel captures consistent visual features of its corresponding region.
2. Reconstruction with simulated annealing
    - **Algorithm configuration.** Set simulated annealing parameters such as initial temperature, cooling rate, and stopping criterion. These parameters must be adjusted experimentally to optimize the performance of the algorithm.
    - **Superpixel placement optimization.** Apply simulated annealing to progressively select and position superpixels in the reconstructed image. The goal is to minimize the mean square error between the reconstructed image and the target image.
3. Reconstruction evaluation
    - **Image quality measurement.** Evaluate the quality of the reconstructed images using standard metrics such as mean square error (MSE). These metrics will help determine how effectively the algorithm has reconstructed the target image.

- **Visual analysis.** Perform a visual analysis to identify areas of improvement or failures in the reconstruction.

4 Theoretical framework

Superpixels

A superpixel is a grouping of adjacent pixels that share similar characteristics, such as color or texture. This grouping is done to facilitate image processing, since working with superpixels reduces the complexity of the image while maintaining essential information. Superpixels more closely follow natural edges and regions within an image compared to an approach based on individual pixels, helping to improve efficiency and accuracy in computer vision tasks such as segmentation and object recognition.

SLIC (Simple Linear Iterative Clustering)

SLIC is an effective technique for performing image segmentation into superpixels. It provides a way to decompose an image into regions of similar size that are more meaningful and easier to analyze than individual pixels. Unlike other segmentation methods, SLIC is particularly fast and memory efficient, making it suitable for high-resolution images and real-time applications.

SLIC segments images into superpixels using an iterative process that begins by selecting uniformly distributed cluster centers, adjusted to avoid color edges. Each pixel is assigned to the nearest cluster center, considering both color similarity and spatial proximity using a combined distance. After mapping, the cluster centers are updated by averaging the pixels in each cluster. This cycle of allocation and updating is repeated until changes in the cluster centers are minimal, indicating convergence.

Local search and hill-climbing search

Local search algorithms start from an initial state and explore neighboring states without keeping a record of the paths traveled or the set of states visited. This implies that they are not exhaustive: they may omit areas of the search space that contain a possible solution. However, they have two main advantages: (1) they require very little memory; and (2) they are capable of finding acceptable solutions in extensive or infinite state spaces where exhaustive algorithms fail. Furthermore, these algorithms are useful in optimization problems where the objective is to identify the optimal state according to an evaluation function.

The hill-climbing search algorithm is the most basic local search algorithm. It keeps track of a current state and at each iteration it moves to the neighboring state with the highest value, that is, it heads in the direction that provides the steepest rise. It ends when it reaches a "peak" where no neighbor has a higher value. Unfortunately, hill-climbing is prone to several common local search problems, some of which are:

- Local maximums. This is probably the most common problem. The algorithm can get "stuck" at a local maximum, which is a point that is better than all of its immediate neighbors, but is not the global optimum (the best possible) of the search space.

- Plateaus: These are flat regions of the search space where all neighbors have a similar value. On a plateau, the algorithm may not have a clear direction of movement, since there are no better neighbors, making it difficult to continue the search toward a better solution.
- Ridges: These are structures in the search space where there is a narrow ascending path flanked by descents in various directions. Ridges can be difficult to navigate, as the algorithm may require precise, non-obvious movements to continue ascending.

Simulated Annealing

An algorithm like hill-climbing that never makes "downhill" moves toward states with lower value (or higher cost) is always vulnerable to getting stuck at a local maximum. Simulated annealing is also a local search algorithm. However, it seeks to attack the common problems that hill-climbing is prone to.

The simulated annealing algorithm is based on the physical process of annealing, which is used in metallurgy to quench or harden materials by heating them to high temperatures and then gradually cooling them. In the context of optimization, this principle is applied to find the lowest point of a cost function, which is comparable to finding the deepest point on an irregular surface. To visualize this, you can imagine the task of letting a ping-pong ball roll on a bumpy surface: if it is allowed to roll without intervention, the ball will settle in the nearest gap, which could be a local minimum.

To prevent the ball from getting trapped in these local minima and instead finding the global minimum, the simulated annealing starts with a high temperature. This high initial temperature is similar to shaking the table strongly at first, allowing the ball to jump out of the local minima. As the algorithm progresses, the "temperature" of the system is gradually reduced. This decrease is analogous to gradually reducing the intensity with which the table is shaken, allowing the ball to have less energy to jump off a minimum and therefore a greater chance of settling into the deeper global minimum.

During the simulated annealing process, each potential move of the solution is evaluated as follows: if the move improves the solution (i.e., reduces the cost function), then it is immediately accepted. If the move makes the solution worse, it can still be accepted, but with a probability that decreases with the magnitude of the worsening and with decreasing temperature. This probability is calculated using the formula $e^{-\Delta E/T}$, where ΔE is the increment in the cost function and T is the current temperature.

The process continues until a predefined termination criterion is reached, which could be a maximum number of iterations, a minimum temperature, or stabilization of the solution in terms of not finding significant improvements. This gradual and controlled approach allows the simulated annealing algorithm to effectively explore the solution space, avoiding getting trapped in local minima and increasing the probability of finding the global minimum.

```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
  current ← problem.INITIAL
  for t = 1 to ∞ do
    T ← schedule(t)
    if T = 0 then return current
    next ← a randomly selected successor of current
    ΔE ← VALUE(current) – VALUE(next)
    if ΔE > 0 then current ← next
    else current ← next only with probability e^{−ΔE/T}
```

Figure 4: Simulated annealing algorithm. The *schedule* determine the temperatura value T as a function of time.

5 Proposal

Before starting the reconstruction process using simulated annealing, superpixels are first extracted from the set of selected images. Below is an example of an image segmented into superpixels using SLIC with 100, 500 and 1000 superpixels.

Figure 5: Image segmented into 100, 500 and 1000 superpixels.

Once the images have been segmented into superpixels, we iterate over all superpixels and store the information for each superpixel in a data structure designed specifically for the superpixel. This data structure, among other things, stores the average color of the superpixel and the size of the superpixel, which will be very useful in the simulated annealing algorithm.

The size of the superpixel is taken by the number of segments of the segmentation to which that superpixel belongs. If a superpixel belongs to the segmentation that was performed with 100 segments, then 100 will be the size of the superpixel. The above allows us to segment the same image into different superpixel sizes to use in the reconstruction. A smaller number of segments implies larger superpixels, while a larger number of segments implies smaller superpixels.

Since you have created this data structure for each superpixel, it is added to a list that will store all the data structures of all the superpixels. This list of "superpixels" is one of the parameters of the simulated annealing.

The simulated annealing algorithm requires an initial state as a parameter. For our par-

ticular problem, the initial state will be an image of the same dimension as the target image and which will be colored a solid color, the average color of the target image.

Figure 6: Target image and initial state.

Now that we have defined what the initial state looks like, we can proceed to define the cooling or temperature change function. (*schedule*).

The cooling function is important because in the initial phases of the algorithm, when the temperature is high, the system is allowed to accept worse solutions with a relatively high probability. This helps prevent the algorithm from getting trapped in local minima at the beginning of the execution. As the temperature decreases, the probability of accepting worse solutions is reduced, and the algorithm begins to perform a more localized and precise exploration.

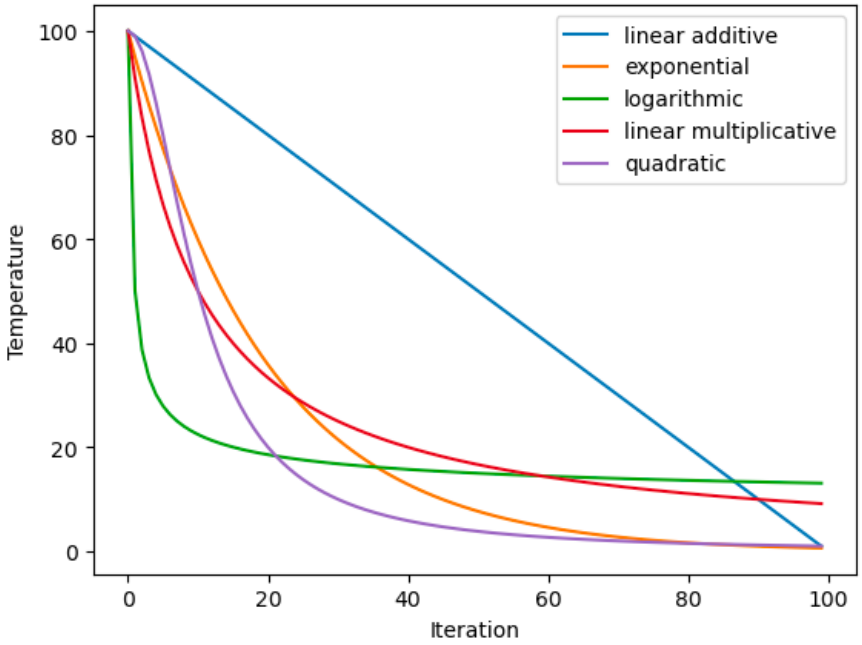

Figure 7: Examples of different types of cooling functions

For our specific problem, we will use an exponential cooling function

$$T_{k+1} = T_k \cdot \alpha$$

where T_k is the temperature in the iteration k, and α is a constant less than 1.

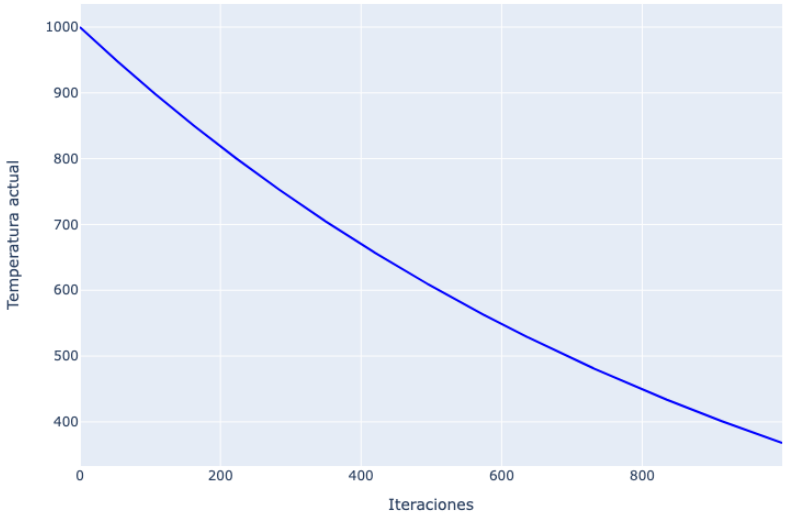

Figure 8: Exponential cooling function. With initial temperature of 1000 and cooling factor 0.999

By carrying out different experiments, it was concluded that it was better to use a cooling factor closer to 1 ($\alpha = 0.999$). This is with the purpose of making the temperature decrease slowly and thus prevent the algorithm from being trapped in local minima.

As for the initial temperature, a number is chosen quite close or similar to the maximum number of iterations. If a fairly accurate reconstruction is desired, then it is necessary to use a greater number of iterations which will result in the use of a greater number of superpixels. The choice of this high initial temperature is made with the purpose of avoiding premature global minima and to facilitate global exploration.

In each iteration there will be a current state (*current*). The next current state has to be a randomly selected successor to *current*. What does that successor to *current* look like? Basically, that successor would be *current* plus a superpixel placed at a random position in the image.

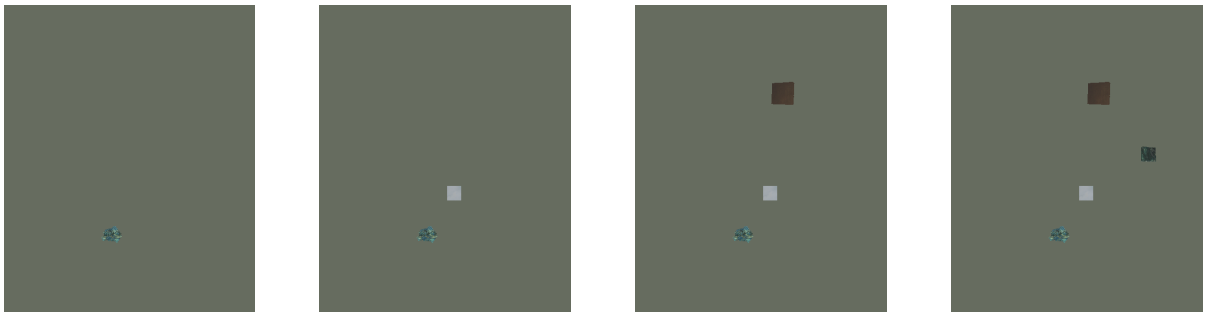

Figure 9: Chain of successors *current*.

The most basic approach to selecting a random successor is to pick a random superpixel and place it at a random position in the image. But this basic approach has a problem: the potential number of superpixels we can choose from can be relatively high. And not all superpixels in the set of selected images will have the same usefulness or importance in the reconstruction. So how can we improve the selection of a successor while maintaining the randomness component?

Let's remember that for each superpixel we have a data structure that, among other things, stores the average color of the superpixel and the size of the superpixel.

How is superpixel size information used? Basically what we want to do is that in the first iterations choose the largest superpixels and then, as the iterations go by, choose smaller superpixels. Let's say each image in the set of selected images will be segmented into 100, 500 and 1000 super pixels. The size of the superpixels will be given by these 3 numbers: 100, 500 and 1000. We can group all the superpixels in each of these groups. And we will do the following:

- We will give each group an importance, let's say that the importance of group 100 is 1, of 500 is 2, and of 1000 is 3. The sum of all the importances is 6.
- Let's say the maximum number of iterations is 3000.

Lo que queremos hacer con esta información es determinar el número de iteraciones durante las cual estaremos seleccionando superpíxeles de cada grupo. Y eso lo hacemos dividiendo la importancia de cada grupo sobre la suma de las importancias y multiplicando por el número máximo de iteraciones.

- Iterations for size 100 $= \frac{1}{6} \cdot 3000 = 500$ iterations

- Iterations for size 500 = $\frac{2}{6} \cdot 3000 = 1000$ iterations
- Iterations for size 1000 = $\frac{3}{6} \cdot 3000 = 1500$ iterations

So, for the first 500 iterations we will only be selecting superpixels of size 100, the 1000 iterations after that we will select size 500, and the last 1500 will be size 1000. This is done so that the largest details of the image are reconstructed first. and then move on to focus on more specific details.

Let's also remember that each superpixel data structure stores the average color of the superpixel. So when selecting a successor, we would first define a random position of the image where the superpixel will be placed. Subsequently, we would obtain the color of the target image at that position. Then, the Euclidean distance between the color of the target image pixel at the random position and the average color of each of the superpixels in the current size group is calculated. Finally, the superpixel that will be chosen will be the one that has the smallest distance from the color of the pixel of the target image.

In this way, we obtain a random successor that attacks areas of color and detail more directly. Additionally, what is done is to place the superpixels with a certain opacity or transparency. This is done because without transparency the results of previous iterations could be buried in the results of new iterations. Then, in order not to completely lose the results of previous iterations, the superpixels are placed with a certain transparency (50%).

Finally, the function with which we will measure the value of each successor or state will be the mean square error (MSE):

$$\text{MSE} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} [I(i,j) - K(i,j)]^2$$

where:

- m and n are the dimensions of the images.
- $I(i,j)$ is the value of the pixel (i,j) of the target image.
- $K(i,j)$ is the value of the pixel (i,j) of the reconstructed image.

Now that we have defined the system and simulated annealing components, it is just a matter of running simulated annealing to start the reconstruction process.

6 Results and discussion

We will present the results of the project using an example of reconstruction.

Figure 10: Set of selected images. Each image was segmented into 500, 1000, 5000 and 10000 superpixels.

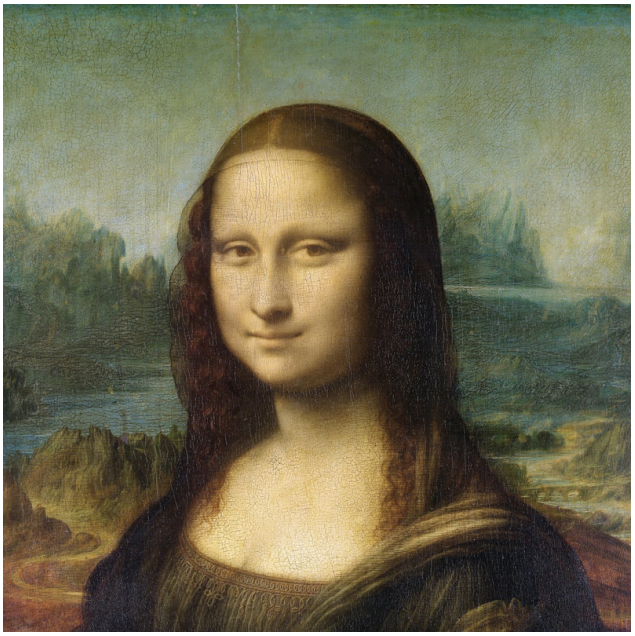

Figure 11: Image to be reconstructured.

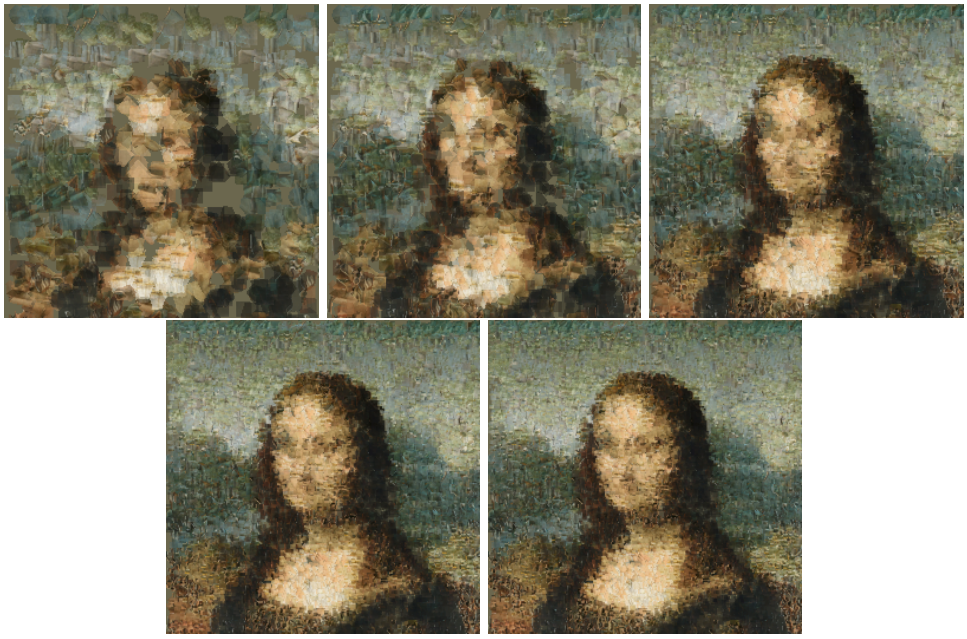

Figure 12: Reconstruction results. Reconstruction of iterations 1000, 2000, 5000, 8000 and 10000.

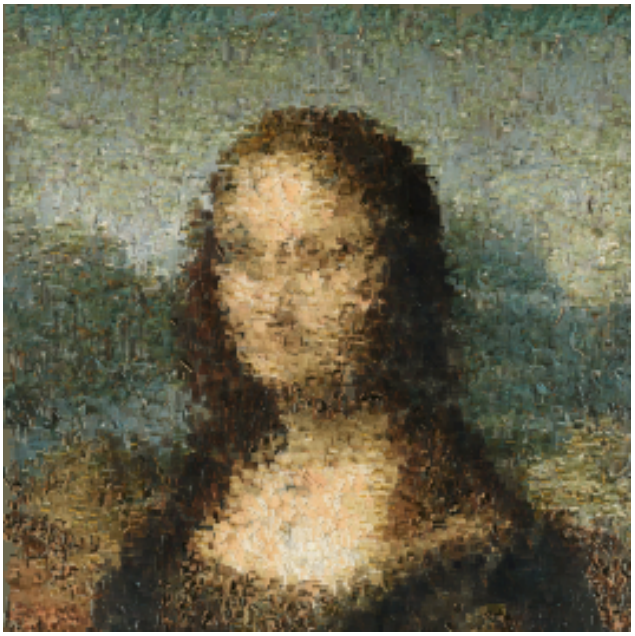

Figure 13: Image reconstructed after 10000 iterations.

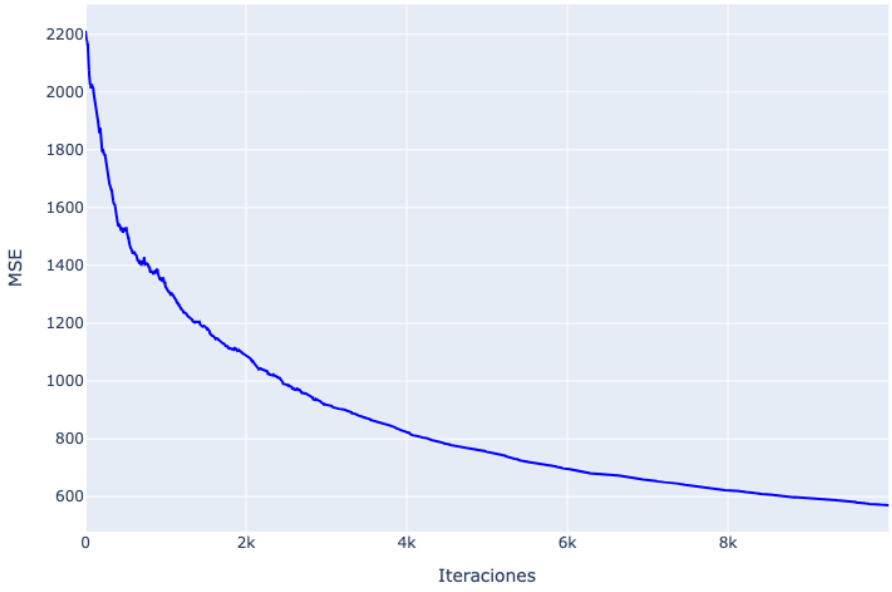

Figure 14: MSE between the reconstruction and the target image across the iterations.

Initially, we can analyze the MSE evolution plot, since it gives us information about how good the reconstruction is and was throughout the iterations. It can be seen that the MSE decreases consistently throughout the iterations. There are a few iterations where simulated annealing accepts worse solutions. This is because the method we developed and implemented to obtain a random successor of *current* is quite effective and functional in terms of the successor tending to be a state with better value, which is very well reflected in the evolution of the MSE.

Also, it should be noted that for each selected image, 4 segmentations of 500, 1000, 5000 and 10000 superpixels were made. The maximum number of iterations was 10,000. The importances and therefore the number of iterations for each group of superpixels were divided as follows:

- Group size 500
  - Importance: 1
  - Number of iterations: $\frac{1}{10} \cdot 10000 = 1000$ iterations
- Group size 1000
  - Importance: 2
  - Number of iterations: $\frac{2}{10} \cdot 10000 = 2000$ iterations
- Group size 5000
  - Importance: 3
  - Number of iterations: $\frac{3}{10} \cdot 10000 = 3000$ iterations

- Group size 10000
  - Importance: 4
  - Number of iterations: $\frac{4}{10} \cdot 10000 = 4000$ iterations

After iteration 3000 there seems to be a consistent improvement where each successor was progressively better. This is because after iteration 3000 the only groups of superpixels left were those of 5000 and 10000. The superpixels of size 5000 and 10000 were quite small. Smaller superpixels allow you to better attack specific details. Attacking specific details in the reconstruction implies an improvement in the MSE.

The evolution of the MSE tells us that there is progress in the reconstruction quality throughout the iterations. But, if we look at the reconstructed image we can deduce certain things. The algorithm is very effective in recreating the large details of the image. However, the algorithm is not as effective at recreating small, fine details such as the eyes, nose and mouth, for example. In fact, we could say that it has certain problems when reconstructing edges. But this is a byproduct of using superpixels and using the MSE as an evaluation function.

The MSE measures the average of the squares of the differences between the pixel values of two images. Essentially, it calculates how much each pixel in one image deviates from the corresponding pixel in another image. Essentially, we could say that MSE is a way to quantify color differences and therefore a simple way to quantify the similarity of two images.

However, a limitation of MSE is that it does not necessarily capture the quality perceived by the human eye. One similarity metric that does do this is the Structural Similarity Index (SSIM). The SSIM is a more sophisticated measure that attempts to model image quality perception based on human vision. Consider changes in texture, luminosity, and contrast, rather than just direct pixel differences.

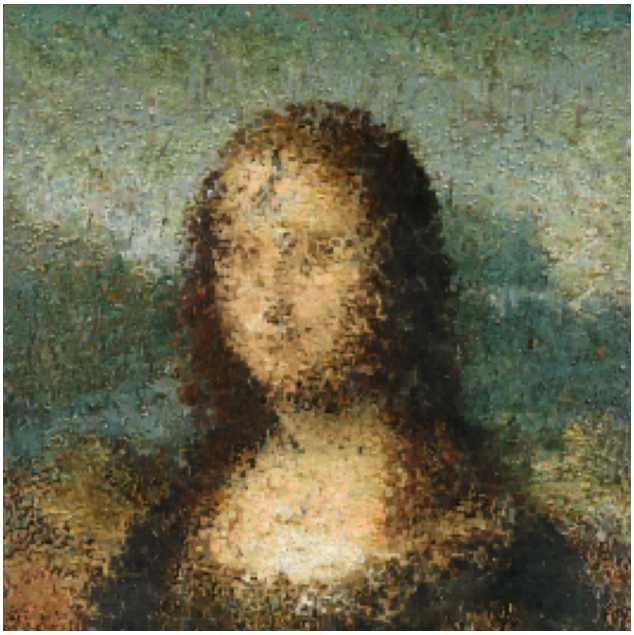

Figure 15: Image reconstructed after 10000 iterations using SSIM as evaluation function.

The use of SSIM was implemented. It presented slightly better results. Visually, its use appears to improve small details and certain edges in the reconstruction. However, the use of SSIM does not represent an incremental improvement over the use of MSE. As an aside, SSIM may be more computationally expensive than MSE. For completeness, the algorithm will give the option to choose which evaluation function to use, this with the purpose of testing different results and making the algorithm more modular.

Finally, it is necessary to remember that the quality of reconstruction will depend on the images selected to extract superpixels (particularly whether these images share similar colors with the target image), the number and group of superpixels that are extracted from each image, the number maximum iterations of the simulated annealing, the initial temperature of the simulated annealing, the cooling factor and the evaluation function to be used (MSE or SSIM). So there are several things that can be modulated to obtain different and better results.

7 Conclussions

In this project, we have explored the feasibility of using superpixels, generated using the SLIC algorithm, and simulated annealing for the image reconstruction task. The results obtained confirm that this technique is effective for reconstructing images, managing to maintain essential visual characteristics with a notable degree of precision. Throughout the development of the project, we have observed that the proposed method provides remarkable adaptability in the manipulation of image resolution and details. However, we have also identified a critical dependence on the initial parameter settings, suggesting that the accuracy of the final result may be highly sensitive to these initial conditions.

In conclusion, the development of this project has clearly demonstrated the effectiveness of combining superpixels with simulated annealing for image reconstruction. The implementation carried out has validated the potential of this technique not only to produce high-quality visual results but also to optimize the arrangement of superpixels, thus maximizing the visual fidelity of the reconstructed images. This approach offers promising prospects for future applications in fields requiring high precision in image reconstruction and enhancement, and opens the door to future improvements that could further expand its applicability and efficiency.

8 Referenced bibliography

- Russell, S. J., & Norvig, P. (2020). Artificial intelligence: A modern approach (4th ed.). Pearson.
- Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., & Süsstrunk, S. (2010). SLIC Superpixels. EPFL Technical Report 149300, School of Computer and Communication Sciences, École Polytechnique Fédérale de Lausanne (EPFL).