UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

# COS791 - Image Processing

# Assignment 2 - Multilevel Thresholding

**Reuben Jooste**
**Student number:** u21457060
**Email:** u21457060@tuks.co.za

## 1 Introduction

In this assignment we implemented two algorithms for multilevel thresholding: **Simulated Annealing (SA)** and **Variable Neighbourhood Search (VNS)**. Furthermore, we were required to evaluate and compare these two algorithms using the **Otsu** multilevel thresholding and the **Kapur** multilevel thresholding objective functions for threshold levels *2, 3, 4, and 5.*

Additionally, we evaluated the quality of the best performing solutions using two **quality metrics**: *Peak Signal Noise Ratio (PSNR)* and *Structural Similarity (SSIM).* In this assignment we present our results in a tabular format, highlighting the evaluation scores of each algorithm's best solution for a given threshold level.

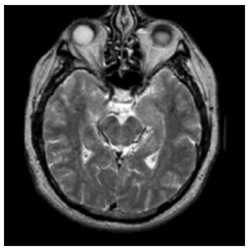
## 2 Experimental Setup

In this section we highlight on how we preprocessed our dataset, the quality metrics used, our objective functions, and how we implemented our SA and VNS algorithms. We used a **global seed value of 99.**

### 2.1 Dataset

Our dataset consists of **5 coloured images** meaning all the images have 3 dimensions. Due to the nature of the algorithms and quality metrics used in this assignment, we had to convert these images to **Gray scale** (i.e. such that the images only have 2 dimensions).

We used the open source Python library, **OpenCV**, to convert the coloured images to gray scale. Figure 1 shows an example of an image converted to gray scale. The difference is not that obvious but by analysing the number of dimensions of the input image we were able to identify that it has a dimension of 3.


(a) Original Coloured Image: T62.jpg

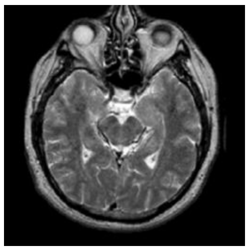

(b) Converted Gray Image: T62.jpg

Figure 1: Original Image vs Gray scaled image

### 2.2 Quality Metrics

As mentioned, we used additional metrics to measure the quality of the best solution. Below we briefly discuss these two metrics.

- **Peak Signal Noise Ratio**
  The PSNR metric measures the ratio between the quality of the image's representation i.e. the power of the noise within the image and the maximum possible pixel value i.e. the power of a signal.
- **Structural Similarity**
  The SSIM metric, compares the similarity between two images (i.e. the target image and a generated image) in terms of the changes in their structural information.

For both of these metrics, we used the open source library, **skimage.metrics**[1].

[1]The link to the Skimage library: `https://scikit-image.org/docs/stable/api/skimage.metrics.html`

## 2.3 Objective Functions

We were required to evaluate the algorithms on two different objective functions: Otsu multilevel thresholding and Kapur multilevel thresholding. Below we briefly discuss how these functions were implemented.

- **Otsu Multilevel Thresholding:**
  This method segments the input (gray) image into multiple regions and seeks to find the optimal threshold values that **maximises** the variance between the regions. The number of threshold values which the algorithm needs to find is determined by the level of thresholding that is applied e.g. if level=2 threhsolding is applied two thresold values will be produced.

  Our objective function works by taking in the gray image, its histogram, the cumulative histogram, the cumulative distribution function (CDF), and the number of thresholds (N) to produce. Using the number of thresholds, the function initialises a sorted list of N randomly generated thresholds where each value is between 0 and 255.

  The function then computes the variance between the regions using the histogram, cumulative histogram, CDF, and list of randomly initialised thresholds. A single region's variance is calculated as follows:

  1. We first calculate the weight of the region by calculating the **sum of pixel counts** within the current region.
  2. Using the calculated pixel counts, we calculate the CDF of the region by calculating the sum of the pixel intensities times their pixel counts.
  3. We then compute the average pixel intensity values for the current region.
  4. Finally, we calculate the variance using the formula of Equation 1. This then contributes to the total variance.

  $$T_{\text{variance}} = \sum W \cdot M^2 \tag{1}$$

  ,where $W$ refers to the total region weight, and $M$ refers to the total squared region mean. We repeat this process for each combination of threshold values such that we can find the optimal threshold combination that maximises the variance between the different regions, thereby segmenting the image effectively.

- **Kapur Multilevel Thresholding:**
  This objective function works in a similar fashion to the Otsu thresholding method. However, instead of trying to segment the image into different regions using different threshold values to maximise the variance, this method tries to **maximise the total entropy of the regions**.

  Again, our objective function takes in the gray-scaled image, its calculated histogram, the cumulative histogram, the CDF, and finally the number of thresholds to use. The function then starts by randomly initialising a **sorted list** of N threshold values between the range 0 and 255.

  Next, the function iterates through the list of randomly initialised thresholds and for each iteration it computes the entropy between the regions as follows:

  1. The function starts by calculating the value of the histogram by calculating the number of pixels within the current region (i.e. the area between two sequential threshold values).
  2. It then **normalises** the pixel values to a sum of 1.
  3. Next, using the normalised histogram, the function calculates the region entropy using the formula defined by Equation 2.
  4. Finally, the function contributes (adds) the calculated entropy to the total entropy.

  $$E_{\text{region}} = -\sum_{i=1}^{I} p_i \log(p_i + \delta(p_i \leq 0)) \tag{2}$$

  ,where $E$ represents the region entropy, $I$ is the total number of intensity levels in the region, $p_i$ is the normalized probability of the intensity level $i$, $\delta(p_i \leq 0)$ is the *Kronecker delta function*, which equals 1 if $p_i$ is less than or equal to 0, and 0 otherwise. This iterative process is done for each sequential threshold values (i.e. for each region).

It should be noted that these objective functions are not used in a **brute-force** manner. Instead, the SA and VNS algorithms use them to find the optimal combination of threshold values.

## 2.4 Algorithms

As mentioned briefly in the introduction, we will use the simulated annealing (SA) and variable neighbourhood search (VNS) for finding the optimal threshold combinations. Below we thoroughly discuss our implementation for these two algorithms.

- **Simulated Annealing (SA):**
  The SA algorithm takes in the following parameters:
  - ⋆ *objective function*
  - ⋆ *number of iterations*
  - ⋆ *Initial temperature value*: This value progressively decreases until it reaches zero. The idea is to start of with a high temperature and with each iteration move closer to a more optimal solution.
  - ⋆ *Step Size:* This value determine with how much the temperature should decrease with each iteration.
  - ⋆ *Method Name:* Defines the multithresholding method that should be used as our objective function.

  At high temperatures the algorithm is more likely to accept worse solutions, allowing it to explore the solution space more efficiently. However, as the algorithm progresses the temperature starts to cool down causing the algorithm to be more selective. By doing this the algorithm can then focus on selecting solutions that improve the solution quality. The SA algorithm is defined according to the following steps:

  1. The algorithm starts by generating an initial solution with an initial cost. It generates the initial solution by calling the passed in objective function.
  2. We then set the current best solution and current best cost to that of the initial solution, respectively.
  3. Next, the algorithm enters an iterative loop that runs for the defined number of iterations (e.g. 100).
     - 3.1 At the start of each iteration, the temperature updated based on the current iteration and initial temperature.
     - 3.2 Next, we calculate a neighbourhood solution that is closely related to the initial threshold values. The neighbouring solution is calculated by perturbing the current solution.
     - 3.3 We then evaluate the neighbouring solution using the objective function.
     - 3.4 If the neighbouring solution's evaluation is better than that of the current best solution (i.e. has a higher variance or entropy), it is accepted as the new best solution.
     - 3.5 If the neighbouring solution is worse, it is accepted with a probability based on the temperature and the difference in evaluations.
     - 3.6 A **stopping condition** is also used to ensure that there has been an increase in evaluation of at least 1% between two solutions over the last 10 iterations. If there has not been then the algorithm terminates.
  4. Otherwise, once all iterations have completed the algorithm returns the best combination of thresholds with their corresponding evaluation (variance or entropy).

- **Variable Neighbourhood Search (VNS):**
  The VNS algorithm also takes in a number of parameters:
  - ⋆ *objective function*
  - ⋆ *number of iterations*
  - ⋆ *step size*
  - ⋆ *method name*
  - ⋆ *neighbourhood size:* This parameter defines the size of the neighbourhood to explore.

The VNS algorithm search a number of neighbouring solutions around a particular solution, hoping to find a better solution. It starts exploring a small neighbourhood of solutions and gradually increases the size either until a better solution is found or until the maximum neighbourhood size is reached, helping the algorithm avoid getting stuck at local optima. Below we define the steps for our VNS algorithm:

1. The algorithm starts off by generating a random initial solution (i.e. list of thresholds) with its associated cost using the passed in objective function.
2. The current best solution and best cost is then set to that of the initial solution.
3. The algorithm then enters an iterative loop which runs for the number of defined iterations.
   - 3.1 With each iteration, the algorithm generates and evaluates a number of neighbouring solutions. Each neighbouring solution is generated by perturbing the current solution.
   - 3.2 Each neighboring solution's cost is then calculated using the objective function.
   - 3.3 If the current neighbour's evaluation is better, it is accepted and the current best solution becomes the neighbouring solution.
   - 3.4 After all neighbours have been evaluated, the algorithm checks its **stopping condition**. If there has not been an increase of at least 1% between any two of the last 10 solutions, it halts.
4. Once the algorithm reached the maximum number of iterations without triggering the stopping condition, it returns the best solution as well as its evaluation.

## 2.5 Algorithm Parameter Values

Below we present a table of parameter values that we used for our algorithms.

| Parameter | SA | VNS |
|---|---|---|
| Seed | 99 | 99 |
| Step Size | 10 | 10 |
| Number of Iterations | 1000 | 1000 |
| Initial Temp. | 100 | - |
| Neighbourhood Size | - | 5 |

Table 1: Algorithm Parameter Values

*Section 3: Results and Discussion follows on the next page...*

## 3 Results and Discussion

In this section we present our findings for the two algorithms. Furthermore, we will conduct a comparative analysis between the two algorithms. The tables for each subsection below presents the final threshold values, their corresponding evaluation scores (variance or entropy), and the quality metric scores for both algorithms on a number of threshold levels for both multithresholding objective functions. Additionally, we present the output images based on the best performing algorithm (in terms of the SSIM score) and show the output images for the worse performing algorithm for the corresponding thresholding level. **The general discussion is done in section 3.6.**

### 3.1 T22.jpg Results

| Image | Level | | Otsu | | Kapur | |
|---|---|---|---|---|---|---|
| | | | SA | VNS | SA | VNS |
| T22.jpg | 2 | th | 51, 246 | 51, 252 | 2, 205 | 2, 252 |
| | | values | 217973845.1062 | 218014378.4247 | 5.05173 | 5.06221 |
| | | SSIM | 61.3994% | 61.3928% | 38.9605% | 38.5241% |
| | | PSNR | 19.1972 | 19.1960 | 12.6975 | 12.6881 |
| | 3 | th | 26, 89, 243 | 27, 89, 255 | 2, 173, 255 | 2, 92, 246 |
| | | values | 239674805.5388 | 239814360.0881 | 8.8600 | 8.8554 |
| | | SSIM | 71.2612% | 71.4029% | 42.2896% | 57.0523% |
| | | PSNR | 18.6434 | 18.7272 | 16.4690 | 16.1922 |
| | 4 | th | 17, 59, 113, 251 | 27, 89, 186, 255 | 25, 105, 181, 255 | 2, 85, 173, 253 |
| | | values | 245842496.6058 | 240708890.7962 | 12.2679 | 12.5862 |
| | | SSIM | 77.0215% | 77.5435% | 76.7832% | 62.3822% |
| | | PSNR | 18.7682 | 24.4143 | 24.0922 | 20.4198 |
| | 5 | th | 23, 82, 200, 230, 249 | 11, 42, 76, 113, 255 | 1, 63, 112, 176, 255 | 0, 52, 110, 176, 255 |
| | | values | 240156181.5532 | 247368382.6675 | 15.7733 | 15.7836 |
| | | SSIM | 78.3034% | 76.9036% | 70.1034% | 40.7359% |
| | | PSNR | 23.4514 | 16.8169 | 21.8658 | 16.2797 |

Table 2: Best Results for Image T22

***Abbreviations:*** *th = Thresholds, values = Objective Function values (variance for Otsu, and total entropy for Kapur)*

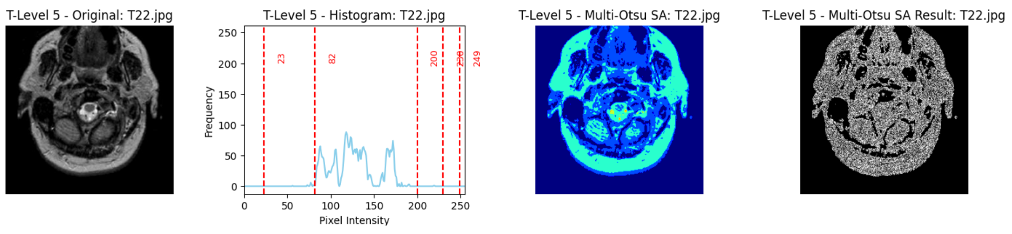

T-Level 5 - Original: T22.jpg
T-Level 5 - Histogram: T22.jpg
Frequency
Pixel Intensity
23
82
200
230
249
T-Level 5 - Multi-Otsu SA: T22.jpg
T-Level 5 - Multi-Otsu SA Result: T22.jpg

Figure 2: Best SA result - Otsu level 5

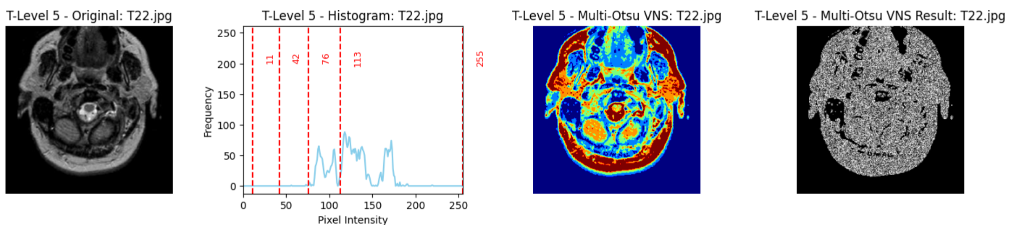

T-Level 5 - Original: T22.jpg
T-Level 5 - Histogram: T22.jpg
Frequency
Pixel Intensity
11
42
76
113
255
T-Level 5 - Multi-Otsu VNS: T22.jpg
T-Level 5 - Multi-Otsu VNS Result: T22.jpg

Figure 3: Corresponding VNS result - Otsu level 5

## 3.2 T32.jpg Results

| **Image** | **Level** | | **Otsu** | | **Kapur** | |
|---|---|---|---|---|---|---|
| | | | **SA** | **VNS** | **SA** | **VNS** |
| **T22.jpg** | 2 | th | 53, 255 | 53, 255 | 3, 240 | 3, 255 |
| | | values | 255399613.3883 | 255399613.3883 | 5.11668 | 5.12199 |
| | | SSIM | 64.2028% | 64.2028% | 41.1463% | 40.9216% |
| | | PSNR | 19.5885 | 19.5885 | 13.5455 | 13.5355 |
| | 3 | th | 38, 110, 255 | 39, 115, 255 | 2, 103, 220 | 2, 107, 255 |
| | | values | 273798437.8483 | 273983970.5988 | 8.90321 | 8.94869 |
| | | SSIM | 74.8196% | 74.8839% | 58.8706% | 58.3480% |
| | | PSNR | 22.0867 | 22.5507 | 17.1110 | 17.2924 |
| | 4 | th | 16, 67, 115, 255 | 34, 103, 215, 255 | 2, 61, 125, 245 | 2, 61, 120, 255 |
| | | values | 282375985.6621 | 274856600.2247 | 12.2150 | 12.2306 |
| | | SSIM | 78.8970% | 77.6939% | 67.8851% | 67.9005% |
| | | PSNR | 19.5673 | 23.8447 | 18.4174 | 18.1973 |
| | 5 | th | 11, 42, 72, 123, 255 | 12, 35, 70, 120, 255 | 5, 48, 89, 130, 217 | 2, 95, 173, 209, 255 |
| | | values | 283739464.0298 | 283973003.2776 | 14.9799 | 15.7006 |
| | | SSIM | 77.7462% | 76.6939% | 77.2636% | 64.5099% |
| | | PSNR | 16.9277 | 16.3859 | 18.5509 | 20.7255 |

Table 3: Best Results for Image T32

***Abbreviations:*** *th = Thresholds, values = Objective Function values (variance for Otsu, and total entropy for Kapur)*

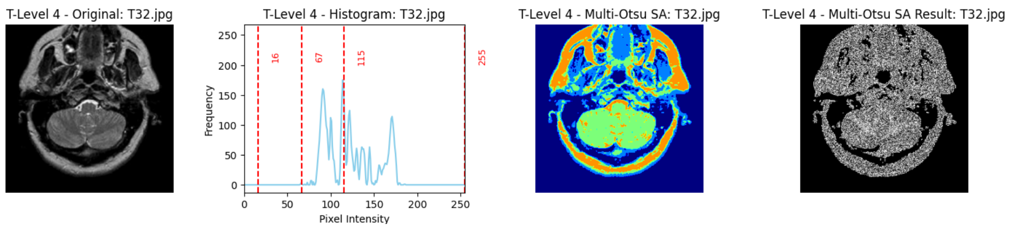

T-Level 4 - Original: T32.jpg
T-Level 4 - Histogram: T32.jpg
16
67
115
255
Frequency
Pixel Intensity
T-Level 4 - Multi-Otsu SA: T32.jpg
T-Level 4 - Multi-Otsu SA Result: T32.jpg

Figure 4: Best SA result - Otsu level 4

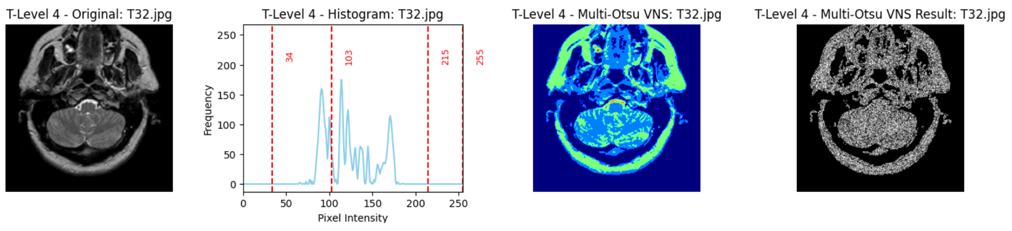

T-Level 4 - Original: T32.jpg
T-Level 4 - Histogram: T32.jpg
34
103
215
255
Frequency
Pixel Intensity
T-Level 4 - Multi-Otsu VNS: T32.jpg
T-Level 4 - Multi-Otsu VNS Result: T32.jpg

Figure 5: Corresponding VNS result - Otsu level 4

## 3.3 T42.jpg Results

| Image | Level | | Otsu | | Kapur | |
|---|---|---|---|---|---|---|
| | | | SA | VNS | SA | VNS |
| T42.jpg | 2 | th | 55, 255 | 55, 255 | 3, 255 | 3, 255 |
| | | values | 310784191.30 | 310784191.30 | 5.21 | 5.21 |
| | | SSIM | 64.40% | 64.40% | 39.16% | 39.16% |
| | | PSNR | 19.03 | 19.03 | 13.67 | 13.67 |
| | 3 | th | 41, 121, 255 | 40, 116, 255 | 4, 178, 255 | 3, 169, 255 |
| | | values | 336097673.54 | 336230108.47 | 9.16 | 9.17 |
| | | SSIM | 75.67% | 75.75% | 48.58% | 47.54% |
| | | PSNR | 22.99 | 22.72 | 17.23 | 17.22 |
| | 4 | th | 21, 63, 125, 255 | 19, 61, 125, 255 | 4, 103, 181, 255 | 3, 106, 181, 255 |
| | | values | 343256705.05 | 343256667.89 | 12.89 | 12.89 |
| | | SSIM | 79.38% | 79.03% | 68.02% | 65.13% |
| | | PSNR | 20.00 | 19.77 | 21.11 | 20.65 |
| | 5 | th | 18, 68, 121, 193, 255 | 37, 112, 187, 239, 255 | 4, 68, 130, 179, 247 | 4, 62, 117, 183, 255 |
| | | values | 345782834.27 | 339229524.66 | 16.04 | 16.14 |
| | | SSIM | 84.65% | 75.09% | 76.13% | 76.84% |
| | | PSNR | 25.93 | 19.64 | 23.91 | 23.32 |

Table 4: Best Results for Image T42

***Abbreviations:*** *th = Thresholds, values = Objective Function values (variance for Otsu, and total entropy for Kapur)*

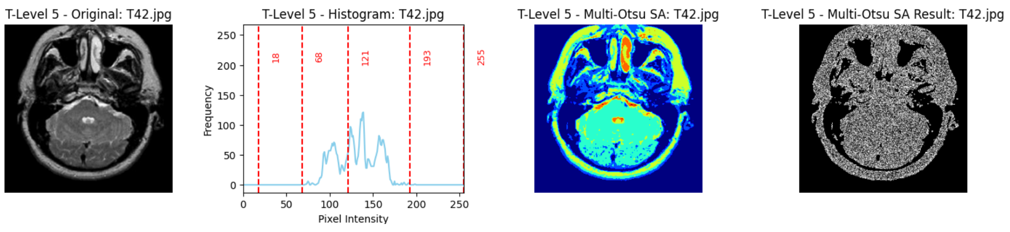

T-Level 5 - Original: T42.jpg
T-Level 5 - Histogram: T42.jpg
Frequency
Pixel Intensity
18
68
121
193
255
T-Level 5 - Multi-Otsu SA: T42.jpg
T-Level 5 - Multi-Otsu SA Result: T42.jpg

Figure 6: Best SA result - Otsu level 5

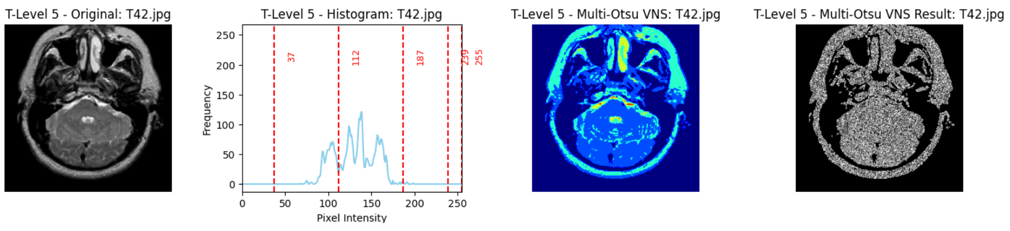

T-Level 5 - Original: T42.jpg
T-Level 5 - Histogram: T42.jpg
Frequency
Pixel Intensity
37
112
187
239
255
T-Level 5 - Multi-Otsu VNS: T42.jpg
T-Level 5 - Multi-Otsu VNS Result: T42.jpg

Figure 7: Corresponding VNS result - Otsu level 5

## 3.4 T52.jpg Results

| **Image** | **Level** | | **Otsu** | | **Kapur** | |
|---|---|---|---|---|---|---|
| | | | **SA** | **VNS** | **SA** | **VNS** |
| **T52.jpg** | 2 | th | [52, 255] | [52, 255] | [2, 218] | [2, 255] |
| | | values | 329435678.52 | 329435678.52 | 5.08 | 5.10 |
| | | SSIM | 62.36% | 62.36% | 38.64% | 37.96% |
| | | PSNR | 19.02 | 19.02 | 14.24 | 14.21 |
| | 3 | th | [38, 109, 255] | [41, 115, 255] | [4, 178, 254] | [2, 120, 255] |
| | | values | 347523817.96 | 347573478.87 | 9.02 | 8.91 |
| | | SSIM | 76.80% | 76.68% | 47.61% | 60.44% |
| | | PSNR | 21.60 | 22.47 | 17.98 | 18.64 |
| | 4 | th | [16, 62, 125, 255] | [18, 60, 121, 255] | [1, 93, 176, 255] | [1, 104, 180, 255] |
| | | values | 353281102.93 | 353778447.67 | 12.67 | 12.69 |
| | | SSIM | 78.65% | 78.93% | 64.98% | 64.58% |
| | | PSNR | 19.16 | 18.80 | 20.79 | 20.54 |
| | 5 | th | [11, 51, 106, 171, 255] | [19, 63, 121, 185, 255] | [2, 39, 85, 133, 202] | [2, 65, 125, 183, 255] |
| | | values | 354740655.71 | 355140597.99 | 14.94 | 16.02 |
| | | SSIM | 82.91% | 84.15% | 72.70% | 73.58% |
| | | PSNR | 22.40 | 25.28 | 17.27 | 23.29 |

Table 5: Best Results for Image T52

***Abbreviations:*** *th = Thresholds, values = Objective Function values (variance for Otsu, and total entropy for Kapur)*

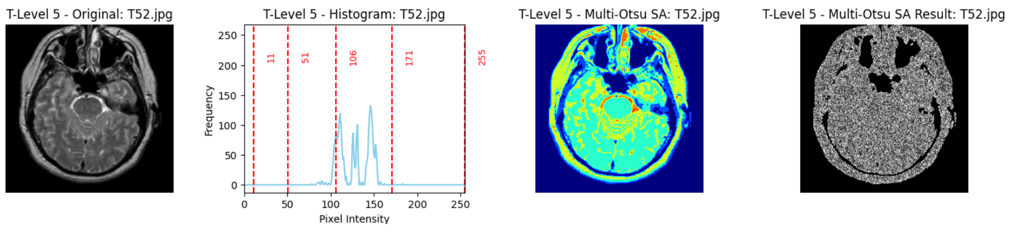

T-Level 5 - Original: T52.jpg
T-Level 5 - Histogram: T52.jpg
11
51
106
171
255
Frequency
Pixel Intensity
T-Level 5 - Multi-Otsu SA: T52.jpg
T-Level 5 - Multi-Otsu SA Result: T52.jpg

Figure 8: Corresponding SA result - Otsu level 5

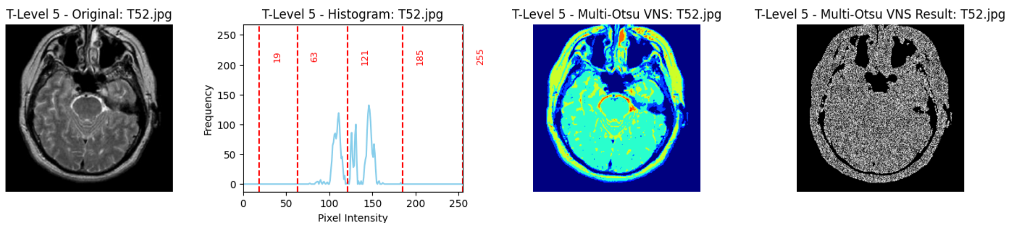

T-Level 5 - Original: T52.jpg
T-Level 5 - Histogram: T52.jpg
19
63
121
185
255
Frequency
Pixel Intensity
T-Level 5 - Multi-Otsu VNS: T52.jpg
T-Level 5 - Multi-Otsu VNS Result: T52.jpg

Figure 9: Best VNS result - Otsu level 5

## 3.5 T62.jpg Results

| Image | Level | | Otsu | | Kapur | |
|---|---|---|---|---|---|---|
| | | | **SA** | **VNS** | **SA** | **VNS** |
| **T62.jpg** | 2 | th | [56, 255] | [55, 255] | [3, 255] | [1, 255] |
| | | values | 389775605.23 | 389786726.10 | 5.19 | 5.19 |
| | | SSIM | 61.38% | 61.35% | 41.34% | 38.32% |
| | | PSNR | 18.62 | 18.60 | 14.79 | 14.17 |
| | 3 | th | [41, 120, 255] | [43, 120, 255] | [0, 174, 255] | [1, 174, 255] |
| | | values | 417335812.57 | 417366034.00 | 9.18 | 9.18 |
| | | SSIM | 75.78% | 75.63% | 17.51% | 46.18% |
| | | PSNR | 22.66 | 22.67 | 12.25 | 17.22 |
| | 4 | th | [54, 182, 175, 255] | [39, 103, 164, 255] | [0, 125, 192, 252] | [0, 100, 181, 255] |
| | | values | 401303903.21 | 422174615.85 | 12.72 | 12.84 |
| | | SSIM | 61.93% | 80.79% | 33.00% | 39.42% |
| | | PSNR | 17.01 | 24.03 | 14.64 | 15.18 |
| | 5 | th | [0, 53, 59, 124, 255] | [21, 60, 113, 183, 255] | [1, 100, 179, 255, 148] | [2, 66, 126, 182, 255] |
| | | values | 810117336.93 | 426503611.91 | 12.84 | 16.19 |
| | | SSIM | 37.07% | 83.68% | 65.24% | 74.79% |
| | | PSNR | 9.87 | 24.17 | 17.92 | 23.55 |

Table 6: Best Results for Image T62

***Abbreviations:*** *th = Thresholds, values = Objective Function values (variance for Otsu, and total entropy for Kapur)*

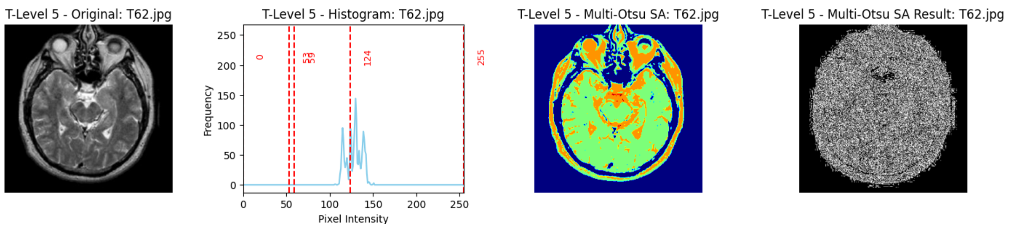

T-Level 5 - Original: T62.jpg
T-Level 5 - Histogram: T62.jpg
Frequency
Pixel Intensity
0
53
59
124
255
T-Level 5 - Multi-Otsu SA: T62.jpg
T-Level 5 - Multi-Otsu SA Result: T62.jpg

Figure 10: Corresponding SA result - Otsu level 5

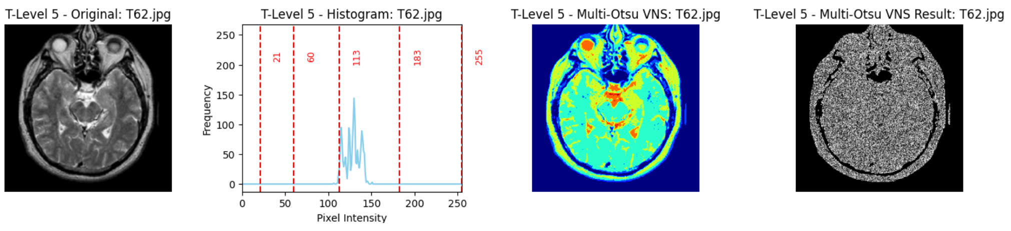

T-Level 5 - Original: T62.jpg
T-Level 5 - Histogram: T62.jpg
Frequency
Pixel Intensity
21
60
113
183
255
T-Level 5 - Multi-Otsu VNS: T62.jpg
T-Level 5 - Multi-Otsu VNS Result: T62.jpg

Figure 11: Best VNS result - Otsu level 5

## 3.6 General Discussion on all results

We will now present a general discussion for all the input images' results.

### 3.6.1 Image T22.jpg

From Table 2 we observe that overall both SA and VNS performed better (in terms of the SSIM score) using the Otsu multilevel thresholding compared to using the Kapur multilevelthresholding. This suggests that the Otsu thresholding technique is more effective in preserving the strucutral similarity of the image. In terms of PSNR, the metric scores are more varied making it difficult to identify which thresholding technique is best to use. Furthermore, we observe that both the SA and VNS algorithms (with any thresholding technique applied) performed better as the threshold levels increase. This indicates that by using more thresholds, the thresholding techniques are able to segment the image more accurately.

Finally, we observe that the SA algorithm achieved the highest SSIM score of **78.3034%** using the Otsu multilevel thresholding method. The VNS algorithm, with the same thresholding applied, achieved a SSIM score of **76.9036%** which suggests that the SA is able to explore the solution space more thoroughly through its probabilistic acceptance mechanism. Figures 2 and 3 shows the output images for the SA and VNS algorithms with Otsu thresholding (level = 5) applied. From the histogram we observe the SA segmented the image more accurately due to the thresholds values used. When we compare the masked images (which was used to calculate the SSIM score and PSNR score) of the two algorithms, we observe that the SA algorithm produced a more accurate result thus its SSIM score and PSNR score are better than that of the VNS algorithm.

### 3.6.2 Image T32.jpg

From Table 3 we analyse that the Otsu method, once again, outperformed the Kapur method across all thresholding levels in terms of the SSIM scores. We notice that the SSIM scores for thresholding levels 4 and 5 are close in range with one another for the Otsu thresholding method (for both algorithms). This indicates that using a higher number of thresholds does not always segment the input image more accurately. In terms of the PSNR scores, we notice variation between the Otsu and Kapur methods achieving better scores. This indicates that if we wanted to make a choice between the best thresholding method, it would depend on the specific image and the balance between peak signal and noise reduction.

Furthermore, we analyse that once again the SA algorithm achieved the best performance (in terms of the SSIM score) using the Otsu thresholding method with 4 threshold values. From Figure 4 and 5 we observe the SA algorithm found more optimal threshold values that segmented the image more accurately compared to that of the VNS algorithm. This then means that the SA's approach of using the acceptance probability helps to explore the solution space more effectively. However, there are cases in terms of the PSNR score where the VNS algorithm achieved better results which indicates that its search strategy can be more efficient for certain threshold levels.

### 3.6.3 Image T42.jpg

From Table 4 we notice that the Otsu threshold method performed much better in terms of the SSIM scores than the Kapur threshold method. Although, the Kapur method maintained roughly 76% structural similarity on threshold level 5, the Otsu method achieved this performance on threshold level 3. This indicates to us that the Otsu threshold method is able to achieve the same performance as the Kapur method by using fewer threshold values to segment the image.

Across all threshold levels, the SA algorithm performed slightly better than the VNS algorithm. However, the is a major jump in performance for the SA algorithm and a drop in performance for the VNS algorithm for threshold level 5 using the Otsu method. This indicates that the VNS algorithm might struggle to explore the solution space as the threshold levels increase. In constrast, as the threshold levels increase, the SA algorithm appears to explore the solution space more effectively, leading to better results. Furthermore, we notice that the VNS algorithm using the Otsu, level 5 thresholding method achieved slighly worse results than when we applied the Kapur level 5 threshold method.

Finally, comparing the output images of the best solution for the SA algorithm and VNS algorithm with level 5 Otsu thresholding applied, we notice that the histogram for the SA algorithm is segmented more accurately than that of the VNS algorithm. We further notice that the thresholded image of the SA algorithm shows the different segments more clearly.

### 3.6.4 Image T52.jpg

Table 5 shows similar results to that of Image T42.jpg. However, we notice that the VNS algorithm with level 5 Otsu thresholding applied, achieved a slightly better performance for this image. This indicates that the VNS algorithm is able to effectively explore the solution space even though it does not use an acceptance probability like the SA algorithm uses.

From the output images, the VNS algorithm was able to find a more optimal combination of threshold values compared to the SA algorithm when using the Otsu level 5 thresholding method. The thresholded images for both algorithms seem very similar. However, due to segmenting the image more accurately using more optimal threshold values, the VNS algorithm's output shows the segments in the thresholded image more accurately.

### 3.6.5 Image 62.jpg

Table 6 shows similar results to that of Table 5. However, we immediately observe the poor performance of the SA algorithm when using level 5 Otsu thresholding. The algorithm only achieved a **37.07%** for the SSIM score. When analysing the masked output image for this result in Figure 10, it is clear that the algorithm failed to maintain the structure of the original image. This bad performance could be due to the SA algorithm getting stuck at local optima or by **converging too quickly**.

On the other hand, the VNS algorithm once again performed the best, achieving a SSIM score of $\approx$ **84%**. This indicates that the algorithm was able to maintain the overall structure of the original image. When comparing the output images of the best solution for the VNS algorithm with the corresponding output of the SA algorithm, we clearly observe the difference in segmentation performance when analysing the histograms. The VNS algorithm with level 5 Otsu thresholding applied, was able to segment the image more accurately due to finding a more optimal threshold value combination than that of the SA algorithm.

We observe the segmentation performance by analysing the thresholded images of the two algorithms. From the SA algorithm's output, it appears as if there are only two segments highlighted in the image which is due to the histogram image almost only being segmented into two distinct parts. In contrast, the VNS algorithm's output clearly shows the image being segmented into distinct parts as we observe the different segment colours.

## 4 Conclusion

In this assignment[2] we implemented two optimization algorithms: simulated annealing (SA) and variable neighbourhood search (VNS). These two algorithms were used to find an optimal combination of threshold values that would accurately segment the input images. We compared the two algorithm's performance by using two multilevel thresholding techniques (Otsu and Kapur) across five threshold levels (2, 3, 4, and 5) and evaluated the quality of the solutions using the structural similarity (SSIM) and peak signal noise ratio (PSNR) metrics.

Overall, the Otsu multilevel thresholding method produced more optimal thresholding results. We also observed that the SA algorithm performed better on 4 of the 5 input images. To further enhance the multithresholding performance, we can try to implement a hybrid approach in the future that would combine the strenghts of both the SA and VNS algorithms to help find more optimal results from the solution space.

[2]Link to github repo: `https://github.com/JsteReubsSoftware/COS791-Assignment2`