Image Reconstruction of Letters Using Algebraic Reconstruction Technique (ART) and the Impact of Filtering Techniques

Emre Tuncer   
Electric and electronic engineering faculty

Ankara University

Ankara, Turkey  
emretuncer396@gmail.com

*Abstract*— The Algebraic Reconstruction Technique (ART) is an iterative method widely used in image reconstruction, particularly in scenarios where the image is reconstructed from projections. This paper demonstrates the implementation of the ART algorithm for reconstructing images of letters represented as 8×8 matrices using MATLAB. The reconstruction process was simulated, and various post-processing techniques, including Laplacian filtering, Gamma correction, and Histogram Equalization, were applied to improve the reconstructed image quality. The results show that the Laplacian filter significantly enhances the image sharpness, while the other filters did not provide satisfactory improvement. The Root Mean Square Error (RMSE) was used to assess the performance of each filter, revealing that the Laplacian filter achieved the best results in comparison to other techniques. The findings emphasize the importance of filter selection in improving the accuracy and clarity of images reconstructed using the ART algorithm.

Keywords—MATLAB, Algebraic Reconstruction Technique, Laplacian filter, Gama correction filtering, Histogram Equalizing filtering.

# Introduction

The Algebraic Reconstruction Technique (ART) is an iterative method widely used for image reconstruction from projections. Originally proposed by Gordon, Bender, and Herman, ART is based on Kaczmarz's method for solving systems of consistent linear equations. The technique operates sequentially, updating the estimated image vector in such a way that each ray-sum equation is satisfied before moving to the next. This sequential correction ensures that the reconstructed image gradually converges to a solution that satisfies all projection data. One iteration of ART is considered complete when all ray-sum equations in the system have been addressed once [1].

In this report, the basic mechanism of ART has been simulated using MATLAB programming language. The primary objective was to reconstruct different letters represented as 8×8 matrices. Each letter was constructed such that its pixels were assigned random intensity values ranging between 100 and 255, while the background pixels were set to 0. This approach allowed for a clear distinction between the letter and its background, simulating a simple yet effective image reconstruction scenario.

Additionally, the effects of various filters on the reconstructed images were explored to evaluate their impact on image quality. These filters were applied to enhance the reconstructed images by reducing noise and improving clarity. The study not only demonstrates the fundamental principles of ART but also highlights its potential for further refinement through post-processing techniques. This combination of ART and filtering offers useful insights into the reconstruction process and its potential applications in image processing and analysis.

# Background and Theory

## ART algorithm explanation

In this section, the ART model used in the simulations is explained. The Algebraic Reconstruction Technique (ART) operates on the principle of iteratively solving a system of linear equations to reconstruct an image from projection data. At the beginning of the reconstruction process, the resulting matrix, which represents the image, is initialized with mean of row, sums, column sums, and diagonal sums. This matrix serves as the starting point for the algorithm, which then attempts to reconstruct the image by iteratively updating the pixel values based on the sensor data.

The sensor data consists of the total sums of pixel intensities along specific rows, columns, and diagonals of the image matrix. The pixel intensities are between 0-255 and grey scale. These sums, often referred to as ray-sum equations, represent the projections of the image as captured by the sensors. The image to be reconstructed consists of a letter, where the pixel values of the letter are random numbers between 100 and 255, while the background pixel intensities are set to 0. The ART algorithm sequentially processes each ray-sum equation, applying corrections to the estimated image matrix in such a way that the updated estimate satisfies the corresponding equation. This process is repeated for all ray-sum equations, and one iteration is considered complete when all equations have been addressed once. Algorithm try to reconstruct image from sensor values that consist of total of rows, columns, and diagonals.

## Mathematical explanation of ART

In this section, the mathematical background of the algorithm is explained using relevant formulas. The reconstruction process is carried out using the following formula, which iteratively updates the pixel values. Each pixel in the image is updated three times during the process: once for the column sensor values, once for the horizontal sensor values, and once for the diagonal sensor values. The following formula used for update pixel from coulomb values:

Here F1 represent the updated value of pixel. F0 is represent the initial value before update, Sc represents sensor data which is the sum of the column of the pixel that wanted to reconstruct. TC is the sum of the column that pixel standing in the reconstructed image before vertical update. And m represents the number of the columns. The next step is calculating values of each pixel for horizontal sensor values. For this purpose, the following formula is used:

Here F1 represent the updated value of pixel F0 is represent the pixel value before horizontal update, SR represents sensor data which is the sum of the row of the pixel that wanted to reconstruct. TR is the sum of the row that pixel standing in the reconstructed image before horizontal update. And n represents the number of the rows. The next step is calculating values of each pixel for diagonal sensor values. For this purpose, the following formula is used:

Here F1 represent the updated value of pixel F0 is represent the pixel value before diagonal update, SD represents sensor data which is the sum of the row of the pixel that wanted to reconstruct. TD is the sum of the row that pixel standing in the reconstructed image before diagonal update. And n represents the number of the rows. The next step involves calculating vertical, horizontal, and diagonal updates multiple times. In this experiment, vertical, horizontal, and diagonal updates are performed sequentially for each pixel, repeated 100 times in total. Between each vertical, horizontal, and diagonal calculation, the negative values are set to 0. Otherwise, the reconstructed image does not form as intended.

# Explanation Of Simulation

In this section of the report, the implementation of the Algebraic Reconstruction Technique (ART) algorithm in the MATLAB programming environment is discussed in detail. The explanation includes how the mathematical formulations underlying the algorithm are translated into MATLAB code. Additionally, the methods employed to enhance the reconstruction quality are described, with particular focus on the integration of post-processing filters and algorithms. These enhancements contribute significantly to minimizing the reconstruction error, thereby improving the overall performance of the algorithm.

## Implementation of mathematical formulas

Initially, to simulate the reconstruction process, an image was created that consisted of the letter "E." This was achieved using the following snippet of MATLAB code:

% Create an 8x8 zero matrix

matrix = zeros(8);

% Left vertical line

matrix(2:7, 2) = randi([100, 255], 6, 1);

% Top horizontal line

matrix(2, 3:5) = randi([100, 255], 1, 3);

% Middle horizontal line

matrix(4, 3:4) = randi([100, 255], 1, 2);

% Bottom horizontal line

matrix(7, 3:5) = randi([100, 255], 1, 3);

As observed in the provided code snippet, the pixel intensities of the letter "E" vary between 100 and 255, while the background pixels have an intensity of 0. This specific intensity range was chosen to create a distinct contrast between the foreground (the letter) and the background, facilitating a clear visual distinction during the reconstruction process.

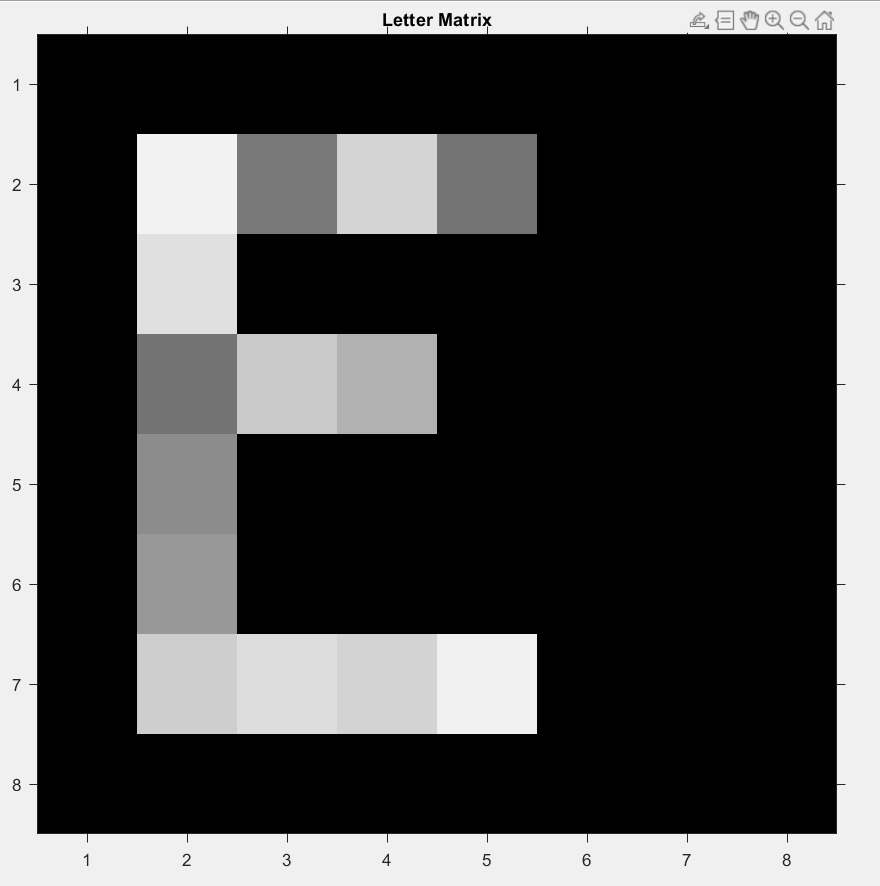


Figure 1 The original letter matrix.

To simulate the detector values, the pixel intensities of the letter matrix are summed up across each column, each row, and both diagonals. This process involves calculating the sum of pixel values for each individual row, column, main diagonal, and anti-diagonal of the matrix. These summed values are then used to simulate the sensor's detection of the image. These processes handle by this snippet code:

% Calculate row and column sums

rowSums = sum(matrix, 2); % Row sums

colSums = sum(matrix, 1); % Column sums

% Calculate diagonal sums

% Main diagonal (top-left to bottom-right)

mainDiagSum = sum(diag(matrix));

% Anti-diagonal (top-right to bottom-left)

antiDiagSum = sum(diag(flipud(matrix)));

Initialization of the reconstructed matrix after the ART algorithm sets the result\_matrix to an initial value based on the average of the row sums, column sums, and diagonal sums. This approach replaces the zero matrix with a matrix of ones scaled by the mean of these sums, providing a balanced starting point for the ART algorithm.

% Initialize result matrix for ART

% Instead of a zero matrix, initial matrix considering the %averages of rowSums, colSums, and diagonal sums have used.

result\_matrix = mean([rowSums(:); colSums(:)]) \* ones(8);

The ART iterations are performed over a maximum of 100 iterations. In each iteration, the matrix is updated in three stages. First, the vertical (column) update adjusts each column by comparing the current column sum with the desired column sum, and the difference is distributed across the column. Next, the horizontal (row) update modifies each row similarly, based on the difference between the current and target row sums. Finally, the diagonal updates focus on adjusting the main and anti-diagonals to match their target sums. After each update, any negative values in the matrix are clamped to zero to ensure the values remain non-negative. This process is repeated for all iterations to iteratively refine the matrix. Example snippet of code to calculate pixel intensities from column detector values:

max\_iter = 100; % Maximum number of iterations to perform

for iter = 1:max\_iter

    % 1. Vertical (Column) Update

    for j = 1:8 % Iterate through each column

        for i = 1:8 % Iterate through each row

% Calculate the sum of each column in the result matrix

            result\_matrix\_colSums = sum(result\_matrix, 1);

%Calculate the adjustment needed for each element in the column

        col\_update = (colSums(j) - result\_matrix\_colSums(j)) / 8;

% Apply the update to the element in the matrix

          result\_matrix(i, j) = result\_matrix(i, j) + col\_update;

        end

    end

## Post image procesing algorithm.

After implementing the mathematical formulas in the simulation, some post-processing techniques are necessary to improve the accuracy of the image. The first post-processing technique applied is normalization, and the second sets pixel intensities below 30 to0. The following snippet of code is used for this purpose.

% Normalize to range 0-1

result\_matrix\_normalized = mat2gray(result\_matrix);

 % Convert to range 0-255

result\_matrix\_normalized = uint8(result\_matrix\_normalized \* 255);

% Set values below 30 to 0

result\_matrix\_normalized(result\_matrix\_normalized < 30) = 0;

This code snippet normalizes the ‘result\_matrix’ to a range of 0-1, then converts it to a range of 0-255 by scaling it. Additionally, pixel values below 30 are set to 0 to eliminate low-intensity noise, which is a part of post-image processing to enhance the final output.

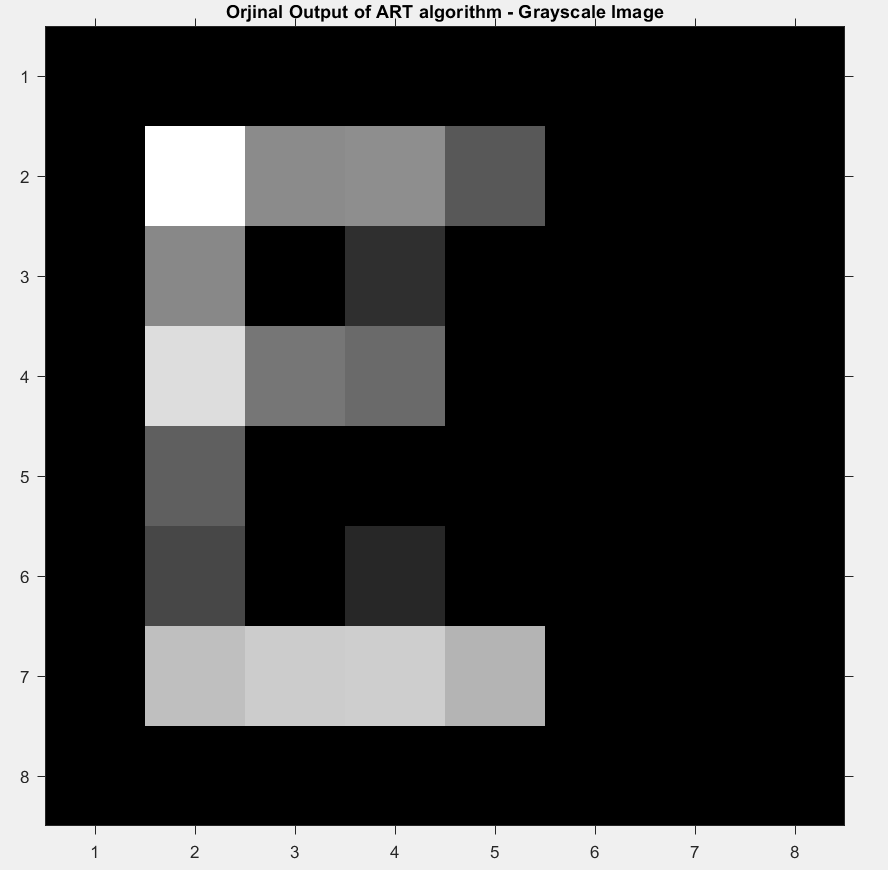


Figure 2 Original output of ART algorithm.

As seen in Figure 2, the output of the ART algorithm does not meet the desired accuracy. Therefore, the effects of different filters have been examined to improve the result.

## Effects of filters

To enhance the output image, several filters were applied. These filters include the Laplacian filter, Gamma correction filter, and Histogram Equalization filter. Each filter was applied individually to the original output of the ART algorithm, and they were not nested together.

Laplacian filter:

Sharpening is applied to the image using the Laplacian filter. First, the Laplacian filter is created with an alpha value of 0.9, which emphasizes sharper edges in the image. The ‘imfilter’ function is then used to apply the Laplacian filter to the normalized result matrix. After applying the filter, the Laplacian image is subtracted from the original normalized image to enhance the sharpness of the edges, resulting in a sharpened image. This method is commonly known as unsharp masking[2], where the high-frequency details (edges) are enhanced by subtracting the blurred (low-frequency) version of the image.

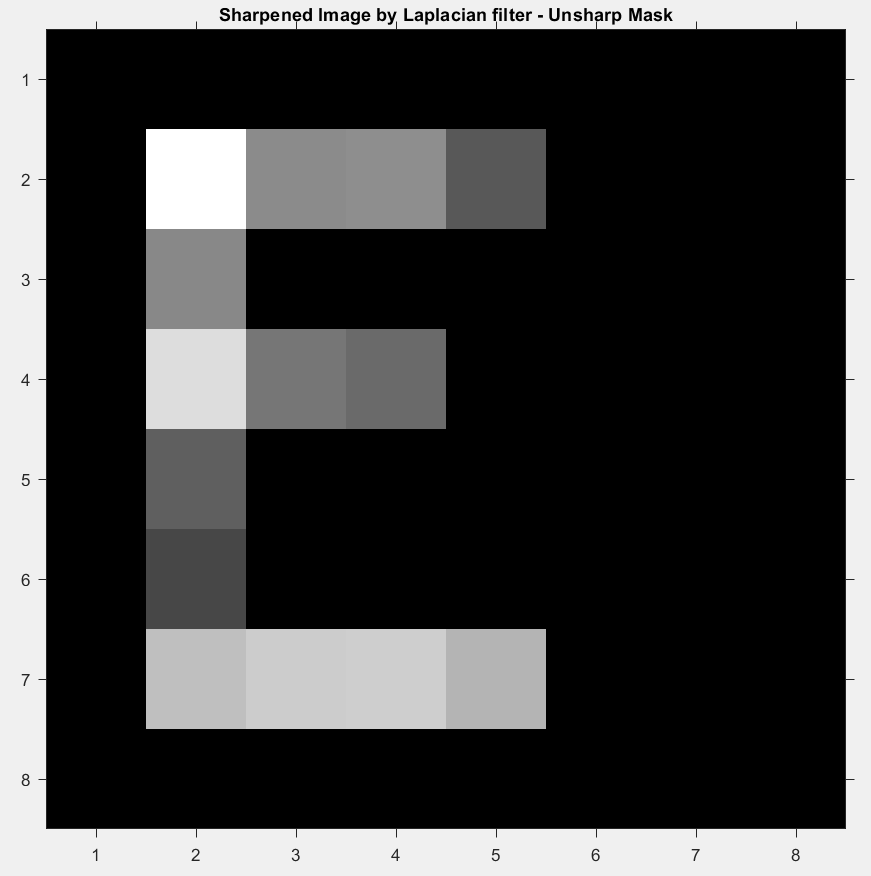


Figure 3 Sharpened Image by Laplacian Filter

The snipped code for create Laplacian filtered image:

% Sharpening with Laplacian Filter

h\_laplacian = fspecial('laplacian', 0.9);  % Laplacian filter, alpha value 0.5: Higher alpha emphasizes sharper edges.

laplacian\_image = imfilter(result\_matrix\_normalized, h\_laplacian, 'same');  % Apply Laplacian filter

sharpened\_image\_laplacian = result\_matrix\_normalized - laplacian\_image;  % Subtract Laplacian from the original image

Gamma Correction filtering:

First, a gamma value of 0.5 is defined, which will affect the image's pixel intensities by making dark areas darker and bright areas brighter. The image matrix (result\_matrix\_normalized) is first converted to a double precision type to ensure precise calculations. Then, the gamma correction is applied by raising the pixel values to the power of the gamma value (0.5). After that, the image is normalized to a range between 0 and 1 using mat2gray. Finally, the result is scaled back to the range 0-255 and converted to uint8 to match the standard image format for display and further processing. This technique enhances the contrast in the image based on the gamma value.

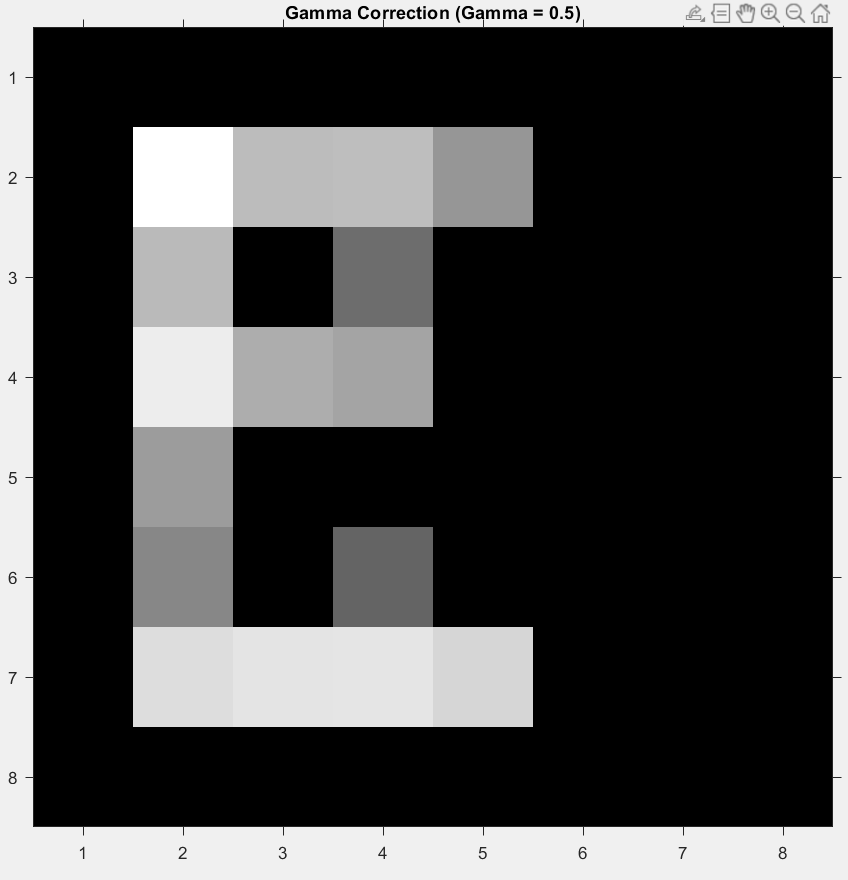


Figure 4 Reconstructed image after gamma correction filter.

The snipped code for create gamma corrected image:

% Contrast Enhancement with Gamma Correction

gamma\_value = 0.5;  % Gamma value

result\_matrix\_gamma = double(result\_matrix\_normalized) .^ gamma\_value;  % Apply gamma correction

result\_matrix\_gamma = mat2gray(result\_matrix\_gamma);  % Normalize to range 0-1

result\_matrix\_gamma = uint8(result\_matrix\_gamma \* 255);  % Convert to range 0-255

Histogram equalization:

The function ‘histeq’ redistributes the intensity values of the image, aiming to spread out the most frequent pixel intensities across the entire available range (0-255). This enhances the contrast, especially in areas of the image with low contrast. The input image, result\_matrix\_normalized, is the image that has been normalized to a range of 0 to 255. After the equalization, the result (result\_matrix\_eq) will have improved contrast, making the details in both dark and light areas more visible.

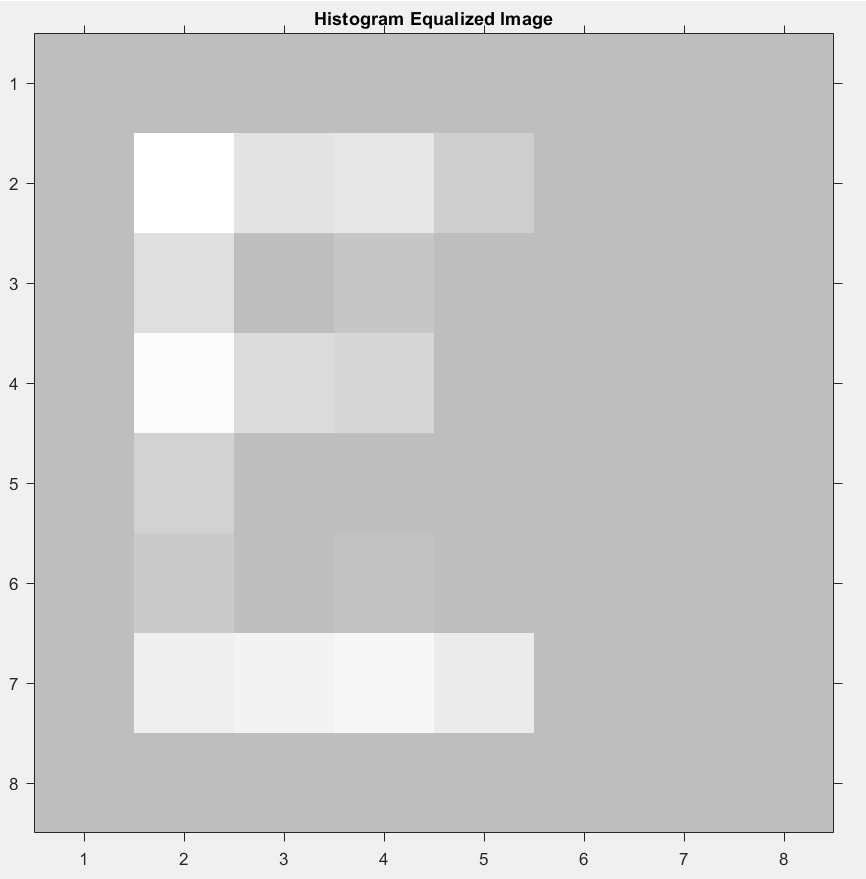


Figure 5 5Reconstructed image after Histogram equalizing filter.

The snippet of code to create a histogram equalization filter:

% Contrast Enhancement with Histogram Equalization

 % Apply histogram equalization

result\_matrix\_eq = histeq(result\_matrix\_normalized);

# Results

In this section, the effects of the Algebraic Reconstruction Technique (ART) algorithm and the applied filters on the reconstruction of different letters are explored. The goal is to show that the algorithm works on a variety of letter forms. In addition to the letter "E," two other letters, "A" and "I," were selected for reconstruction. This allows for a broader assessment of the algorithm's performance across different shapes. By applying the same ART method and post-processing filters, the results demonstrate the algorithm's ability to handle different letters effectively. To evaluate and compare the results after applying the filters, the Root Mean Square Error (RMSE) method was used. The RMSE result can be seen below, providing a quantitative assessment of the performance of each filter.

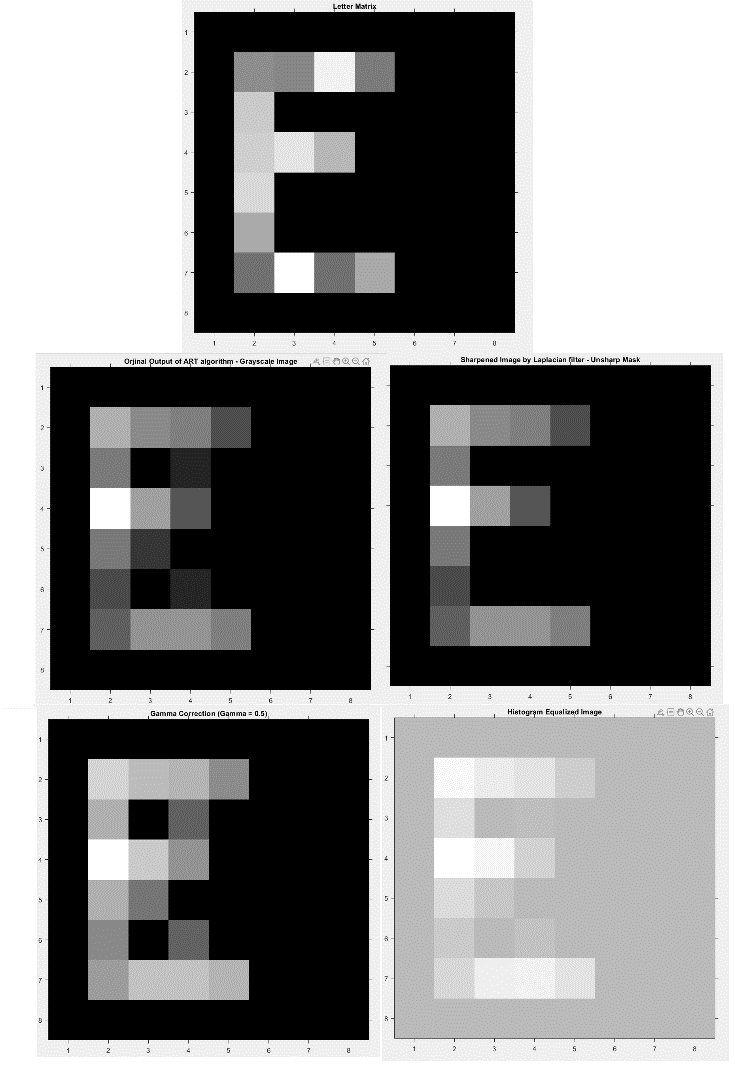


Figure 6 The effect of ART algorithm and filters on E letter.

RMSE between original and ART output: 28.4357.

RMSE between original and Laplacian sharpened image: 27.8983.  
  
RMSE between original and Gamma corrected image: 27.1797.

RMSE between original and Histogram equalized image: 170.4802.

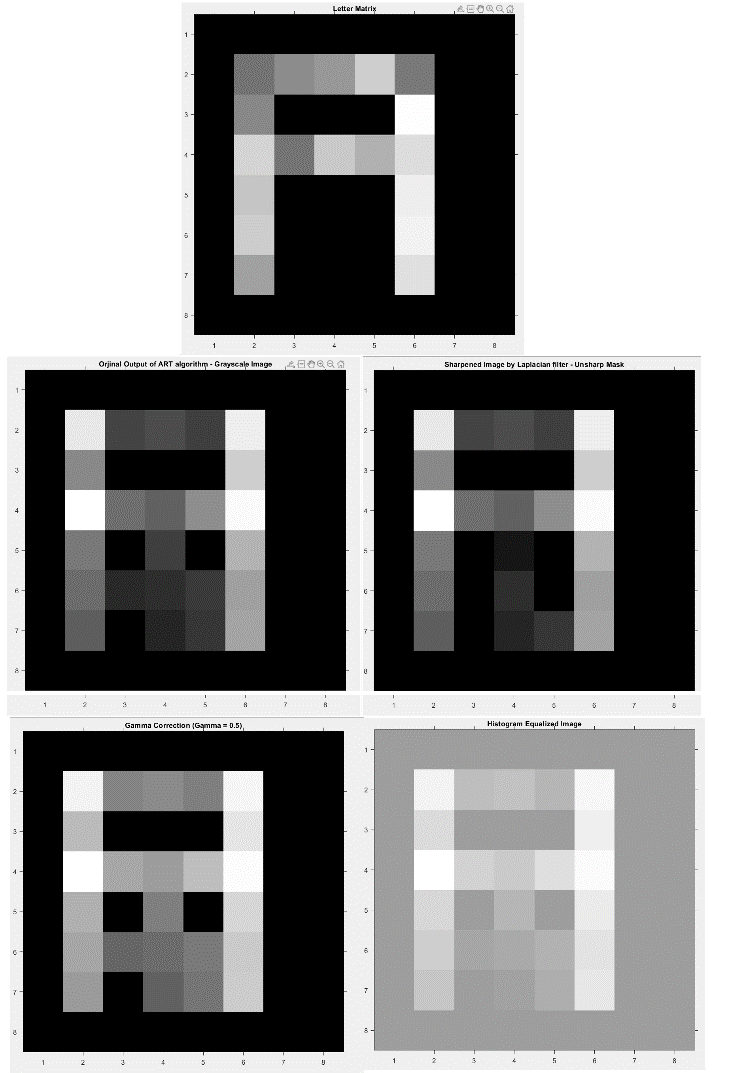


Figure 7 The effect of ART algorithm and filters on A letter.

RMSE between original and ART output: 43.5793.

RMSE between original and Laplacian sharpened image: 42.5738.  
  
RMSE between original and Gamma corrected image: 42.594.

RMSE between original and Histogram equalized image: 146.646.

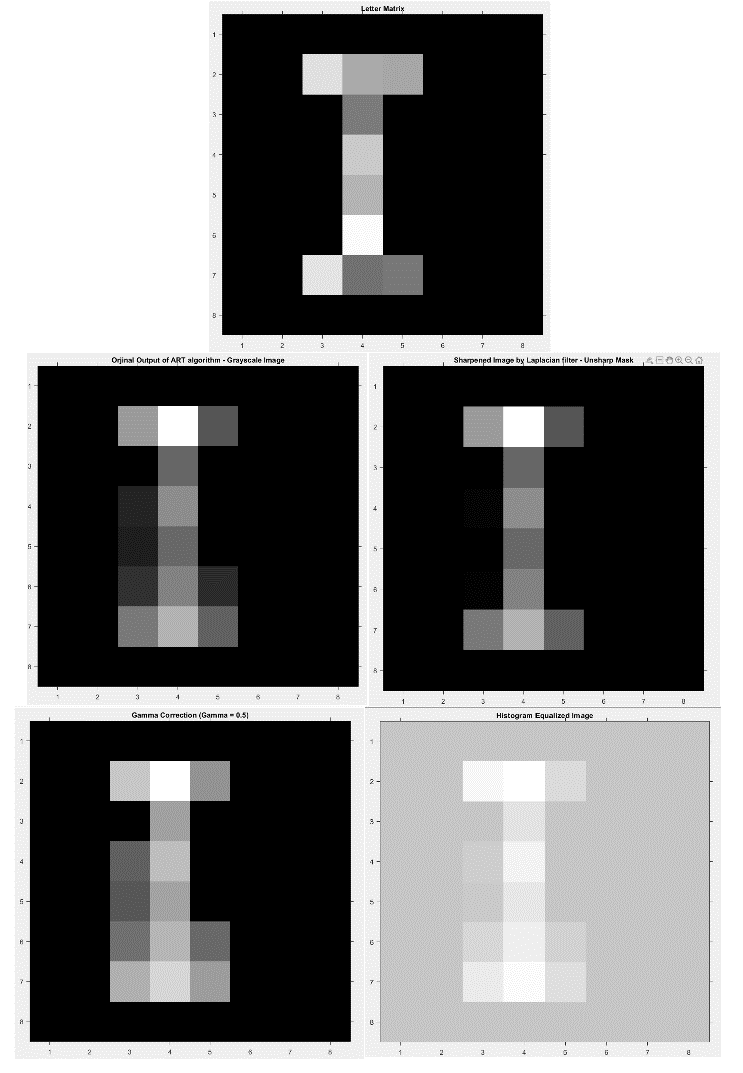


Figure 8 The effect of ART algorithm and filters on I letter.

RMSE between original and ART output: 31.5111.

RMSE between original and Laplacian sharpened image: 30.531.  
  
RMSE between original and Gamma corrected image: 32.9574.

RMSE between original and Histogram equalized image: 184.9479.

# Conculution

Due to the random nature of the initial letter matrix, the performance of the ART algorithm can vary. However, despite this variability, the results show that, in general, applying the Laplacian filter significantly improves the outcome. The enhancement achieved through the Laplacian filter is usually sufficient, ensuring that the final output meets the desired criteria for most cases. This suggests that, while the initial randomness may influence the ART algorithm's performance, the Laplacian filter effectively refines the results, leading to a more consistent and satisfactory success rate. On the other hand, the other filters used in the simulation did not provide a sufficient success rate.

##### References

[1] R. Gordon, R. Bender, and G. T. Herman, ‘Algebraic Reconstruction Techniques (ART) for three-dimensional electron microscopy and X-ray photography’, *J. Theor. Biol.*, vol. 29, no. 3, pp. 471–481, Dec. 1970, doi: 10.1016/0022-5193(70)90109-8.

[2] M. A. Badamchizadeh and A. Aghagolzadeh, ‘Comparative Study of Unsharp Masking Methods for Image Enhancement’, in *Third International Conference on Image and Graphics (ICIG’04)*, Hong Kong, China: IEEE, 2004, pp. 27–30. doi: 10.1109/ICIG.2004.50.