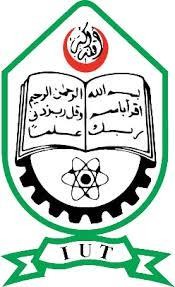
**DIGITAL SIGNAL PROCESSING LAB**

Project



**Solving the Puzzle; A MATLAB Approach to Image Reconstruction**

**Group Information**

Group No. : 4

Section : A1

Semester : Winter (7th)

Department : Electrical and Electronic Engineering (EEE) Institution : Islamic University of Technology (IUT)

**Group Members**

|  |  |
| --- | --- |
| **Student ID** | **Name** |
| **200021137** | Khandoker Muntaseer Shams |
| **200021141** | Mustafid Bin Mostafa |
| **200021147** | Monem Shahriar Saikat |
| **200021149** | Md. Bakibullah Sakib |
| **200021151** | Modou Kabirr Faal |

**2. Project Problem Statement:**

The goal of this project is to correctly determine the proper spatial arrangement of the provided gray scale puzzle pieces in order to reconstruct the original color image. The challenge is to put those pieces in the right slots of a 4x4 grid to get back to the original image

**3. Solution Abstract**

This project aims to use MATLAB to solve an image reconstruction puzzle using image processing methods. We have a colored image and a set of colorful puzzle pieces.

**Step 1:** Get the images load

**Step 2**: Segment the original image

**Step 3**: Calculate similarity metrics between the puzzle pieces with Mean Squared Error (MSE)

**Step 4**: Recreate the final gray image Both the matrix output and visual output of reconstructed image are shown.

**4. Detailed Methodology:**

The process works as follows:

**Loading the Image Data**:

MATLAB loads the colorful original image and 16 grayscale puzzle pieces.

**Preprocessing the Images:**

Grid of the original image segmented into 4x4 grayscale blocks for comparison.

Scaling and rotating puzzle segments to fit reference blocks.

**Algo 3: Feature Extraction and Matching**

Observed Mean Squared Error (MSE) between newly displaced puzzle pieces and nearest reference blocks.

**Optimal Placement Calculation:**

Finding the best fit for each piece of a 4x4 puzzle.

**Image Reconstruction:**

Putting together the puzzle pieces in their recognized positions to produce the final reconstructed grayscale image.

**Displaying Results:**

4x4 matrix with names of each of the puzzle pieces and final image.

**5. Code**

|  |
| --- |
| clc;  clear all;  close all;  % Load the original color image and grayscale puzzle pieces  originalImage = imread('noisy\_colorful\_image.jpg'); % replace with the actual filename of the original image  imshow(originalImage);  % Cropping image pieces to reduce the white borders  for k=1:16  i = imread(sprintf('piece%d.jpg', k));  gImg = rgb2gray(i);  th = 210;  mask = gImg < th;  [row, col] = find(mask);  trow = min(row);  brow = max(row);  lcol = min(col);  rcol = max(col);  cropped = i(trow:brow, lcol:rcol, :);  imwrite(cropped, sprintf('new\_piece%d.jpg', k));  end  puzzleFolder = 'puzzleimages'; % folder containing the 16 puzzle pieces  numPieces = 16;  gridSize = 4; % 4x4 grid  % Convert the original image to grayscale  grayOriginal = rgb2gray(originalImage);  % Get the dimensions of the original grayscale image  [rows, cols] = size(grayOriginal);  % Calculate integer dimensions for each block  blockRows = floor(rows / gridSize);  blockCols = floor(cols / gridSize);  % Initialize a cell array to store 4x4 blocks of the original image  originalBlocks = cell(gridSize, gridSize);  % Split the original image into 4x4 blocks  for i = 1:gridSize  for j = 1:gridSize  % Calculate the row and column ranges for each block  rowRange = (i-1)\*blockRows + 1 : min(i\*blockRows, rows); % Ensure we stay within bounds  colRange = (j-1)\*blockCols + 1 : min(j\*blockCols, cols); % Ensure we stay within bounds    % Extract the block and store it in the cell array  originalBlocks{i, j} = grayOriginal(rowRange, colRange);  end  end  % Load each puzzle piece and store it in an array  puzzlePieces = cell(numPieces, 1);  for k = 1:numPieces  pieceFilename = fullfile(puzzleFolder, sprintf('new\_piece%d.jpg', k)); % replace with the actual filenames of the pieces  puzzlePieces{k} = imread(pieceFilename);  end  % Initialize a matrix to store the final arrangement of puzzle pieces  finalArrangement = zeros(gridSize, gridSize);  % Initialize an array to keep track of used puzzle pieces  usedPieces = false(numPieces, 1); % false means the piece hasn't been used yet  % Loop through each block of the original image and find the best matching puzzle piece  for i = 1:gridSize  for j = 1:gridSize  bestMatch = inf;  bestPiece = 0;    % Compare each unused puzzle piece to the current block  for k = 1:numPieces  if ~usedPieces(k) % Only consider pieces that haven’t been used  piece = puzzlePieces{k};  resizedPiece = imresize(piece, [blockRows, blockCols]); % Resize to match the block size    % Convert the puzzle piece to grayscale if it is RGB  if size(resizedPiece, 3) == 3  resizedPiece = rgb2gray(resizedPiece);  end    % Calculate the similarity measure  diff = sum((double(originalBlocks{i, j}) - double(resizedPiece)).^2, 'all');    if diff < bestMatch  bestMatch = diff;  bestPiece = k;  end  end  end    % Assign the best matching piece to the current position  finalArrangement(i, j) = bestPiece;  usedPieces(bestPiece) = true; % Mark this piece as used  end  end  % Display the final arrangement as a 4x4 matrix of piece numbers  disp('Final arrangement of puzzle pieces (as matrix):');  disp(finalArrangement);  % Initialize the reconstructed image as grayscale  reconstructedImage = uint8(zeros(rows, cols));  for i = 1:gridSize  for j = 1:gridSize  % Define the range for each block in the reconstructed image  rowRange = (i-1)\*blockRows + 1 : i\*blockRows;  colRange = (j-1)\*blockCols + 1 : j\*blockCols;    % Resize the puzzle piece to the required dimensions  piece = imresize(puzzlePieces{finalArrangement(i, j)}, [blockRows, blockCols]);    % Convert the puzzle piece to grayscale if it is RGB  if size(piece, 3) == 3  piece = rgb2gray(piece);  end    % Assign the grayscale piece to the appropriate location in the reconstructed image  reconstructedImage(rowRange, colRange) = piece;  end  end  % Display the reconstructed grayscale image  figure;  imshow(reconstructedImage);  title('Reconstructed Grayscale Image'); |

**6. Result Analysis**

The reconstructed image effectively illustrates the proper configuration of the puzzle pieces. The 4x4 matrix showcases how the pieces are positioned according to their filenames, with the visual depiction closely mirroring the original picture. Key observations include the following:

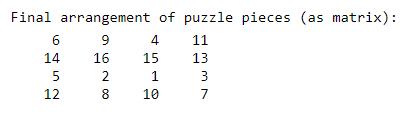
**(1)** The precision of the reconstruction is significantly influenced by the effectiveness of the similarity evaluation techniques employed.

**(2)** Mean Squared Error (MSE) served as the main criterion for aligning the puzzle piece placements.

**(3)** Small modifications, such as rotating or resizing the pieces, enhance the overall outcome.

**(4)** MATLAB's computational efficiency facilitates rapid image processing and reconstruction.

**Spatial arrangement of puzzle blocks:**

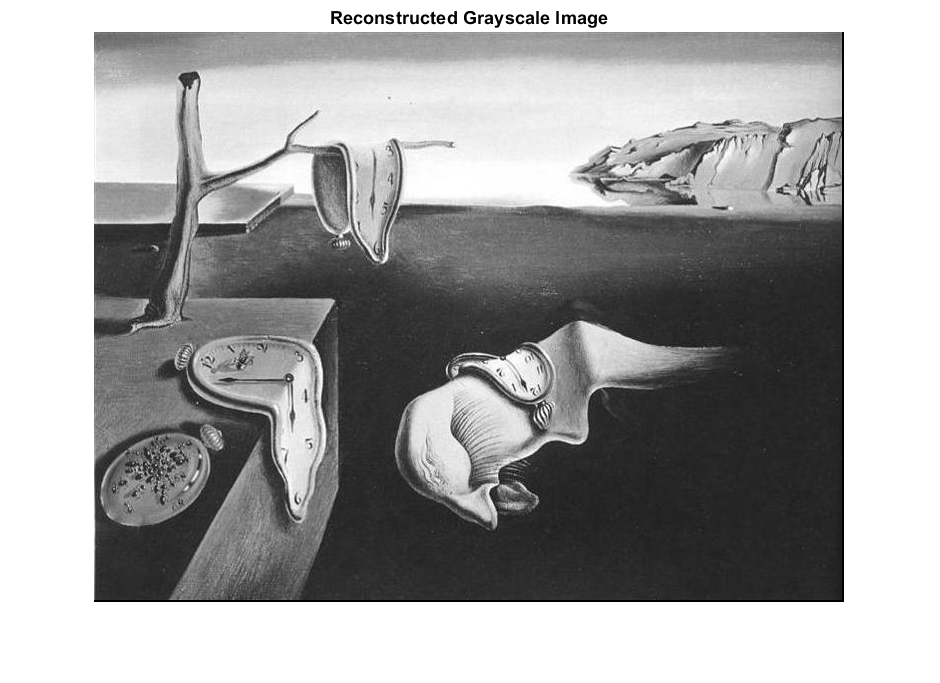


**Original puzzle pieces and cropped pieces:**

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Reconstructed Image:**

****

**7. Conclusion**In summary, this project adeptly executed an image reconstruction solution utilizing MATLAB. The approach cleverly organized puzzle pieces through similarity assessment techniques. Nonetheless, several challenges emerged, notably variations in brightness and orientation, which affected the overall accuracy of the reconstruction. Moving forward, potential enhancements could incorporate advanced feature extraction methods, such as edge detection or deep learning-driven image recognition, to significantly boost both the accuracy and automation of the process.

**8. Team Contribution**

[**Mustafid Bin Mostafa, ID:200021141**]: Crafted the functions responsible for image loading and preprocessing.

[**Monem Shahariar, ID:200021147**]: Cropping and Devised algorithms for assessing similarity between images.

[**Khondaker Muntaseer Shams, ID: 2000211137 & Bakibullah Sakib,**

**ID: 200021149]**: Managed the processes of finding image similarity and reconstruction.

**[Modou Kabirr Faal, ID: 200021151]:** Visualization of the final resultsand documentation.

**Reference:**

1. [**https://www.mathworks.com/matlabcentral/fileexchange/13900-n-puzzle-dynamic-size-and-solver**](https://www.mathworks.com/matlabcentral/fileexchange/13900-n-puzzle-dynamic-size-and-solver)
2. [**https://www.researchgate.net/publication/236130403\_Scientific\_Puzzle\_Solving\_Current\_Techniques\_and\_Applications/citations**](https://www.researchgate.net/publication/236130403_Scientific_Puzzle_Solving_Current_Techniques_and_Applications/citations)