Prepared by

ندى طارق صابر محمد عفيفي

2022170464

يوسف على عبدالعاطى محمد

2021170639

احمد صبحي محروس صقر

2022170542

اسراء محمود عبد المغني احمد

2022170578

مصطفى محمد مصطفى محمد

2022170427

رقيه احمد شعراوي حامد

2022170154

. Graph Representation The image segmentation algorithm treats the image as a graph

where: Each pixel is represented as a node Edges connect neighboring pixels with weights based on color differences

The function BuildGraphWithMST constructs a weighted undirected graph from the image channel (grayscale or color-separated).

Inputs: A 2D byte array channel representing the pixel intensities of a single color channel (red, green, or blue).

Output: A list of Edge structures where each edge connects two neighboring pixels with a weight proportional to their intensity difference.

.Steps:

1.Setup Directions:

It uses 4-connectivity (down, right, and two diagonals) to ensure each pixel is connected to a subset of its neighbors: dx = [0, 1, 1, 1], dy = [1, 0, 1, -1]. This includes: Right (0, 1) Down (1, 0) Down-right (1, 1) Down-left (1, -1)

This approach uses partial 8-connectivity, ensuring each pixel is connected without redundant edges.

2.Iterate Over All Pixels: For each pixel at position (i, j), calculate a unique 1D index: index1 = i * width + j.

3.Check Neighbors:

For each direction (down, right, and diagonals), it checks if the neighboring pixel (ni, nj) is valid (inside bounds).

"The graph uses partial 8-connectivity (right, down, and two diagonals) to build edges between pixels."

4.Compute Edge Weight: The weight of the edge between two connected pixels is the absolute difference between their intensity values: double weight = Math.Abs(val1 - val2);

5.Create Edge:

An Edge object is created using the indices of the current and neighbor pixel and the computed weight. .Sort Edges: All edges are sorted by weight (for Kruskal's algorithm to work efficiently afterward). This prepares the graph so that it can later be used for minimum spanning tree operations or segmentation logic.

Time and Space Complexity Analysis:

Let: H = height of the image W = width of the image N = H * W = total number of pixels

Time Complexity :

1.Loop over all pixels: Outer loop is O(N). 2.Loop over neighbors (fixed 4 directions): Constant-time operations for each pixel → O(4N) = O(N).

3.Edge creation: At most 4 edges per pixel → total edges = O(N).

4.Sorting edges: Using .Sort() on edge list → O(E log E) where E ≈ 4N ⇒ O(N log N). ... Total Time Complexity ... O(N log N)

Space Complexity :

Edge list: Stores up to 4 edges per pixel → O(N). ... Total Space Complexity ... O(N)

Image Segmentation & Visualization

Image Segmentation The image segmentation algorithm builds on the graph constructed for each color channel (Red, Green, Blue) and applies a region merging technique based on edge weights to group similar pixels.

Each channel is segmented independently, then the results are combined to produce final regions.

- Channel Segmentation: SegmentChannel() This function segments a single color channel (2D byte array) using an algorithm similar to Felzenszwalb and Huttenlocher's graph-based image segmentation .

Inputs:- channel: A 2D byte array of a single color channel (e.g., red values for all pixels).-

k: A floating-point constant that controls the segmentation threshold. Output:- An array of integers representing segment labels for each pixel.- Steps:

1. Build Edge List:-

2. Each pixel is a node.-

3. Connected to 4 neighbors.-

4. Edge weights: abs(val1 - val2)

2. Sort Edges:- Sorted ascending by weight.

3. Initialize Disjoint Sets: - Track connected components.

5. Merge Segments:-

6. Merge based on threshold: minInternal = min(internalDiff[rootA] + k / size[rootA], internalDiff[rootB] + k / size[rootB])

7. Time & Space Complexity Analysis

Let:- H = image height-

W = image width

N = H W

Time Complexity:-

Build edges: O(N)-

Sort edges: O(N log N)-

Merge: O(N (N))

Total: O(N log N)

Space Complexity:-

Edge list, structures: O(N)

3. Image Segments Visualization Once segmentation is complete, we visualize the segmented regions by assigning each segment a unique color.

Visualization Function: Filterting() Combines segmented results from all three channels and assigns colors to final merged regions.

Inputs: - ImageMatrix: The original RGB pixel matrix.

k: The segmentation parameter.

Output:-

A new RGB image matrix with distinct colors for each segment.
- Steps:

1. Segment All Channels: redSeg, greenSeg, blueSeg = SegmentChannel(...)

2. Merge by Matching Labels:- Match R, G, B segment labels.

3. Assign Random Colors:- Random RGB per region.

4. Save Region Statistics:- segments.txt with pixel count per segment.

Complexity:

Time Complexity:-

Segment channels: O(N log N)

Merge and color: O(N)

Total: O(N log N)

Space Complexity:- Segment data + map: O(N)