A picture containing text

Description automatically generated

**Ain Shams University**

**Faculty of Computer and Information Science**

**Scientific Computing department**

**Ain shams university**

**Faculty of computer and information science**

**Bioinformatics department**

**Project Title**

**Image Quantization**

**By**

|  |  |  |
| --- | --- | --- |
| **Name** | **ID** | **Section** |
| **Nour Mohamed Hussein Kamaly** | **20191700701** | **5** |
| **Nourhan Abdel-Karim Khalaf Abdel-Hafez** | **20191700716** | **5** |
| **Mohammed Nour-Elden Abbas Ismael** | **20191700583** | **4** |
| **Abdul-Rahman Sayed Ali Mohammed** | **20191700339** | **3** |

**Under the supervision of**

**Dr. Ahmed Salah**

**Computer Science Department,**

**Faculty of computer and Information Science**

**Ain Shams University**

A picture containing text, bird, aquatic bird, screenshot

Description automatically generatedGraphical user interface, application

Description automatically generatedGraphical user interface

Description automatically generated

**What is Image Quantization?**

The idea of *color quantization* is to reduce the number of colors in a full resolution digital color image (24 bits per pixel) to a smaller set of representative colors called ***color palette*.** Reduction should be performed so that the quantized image differs as little as possible from the original image. Algorithmic optimization task is to find such a color palette that the overall distortion is minimized.

An example of color quantization is depicted in the following Figure. First, a color palette is found by using clustering algorithm and then the original image values are replaced by their closest values in the palette.

Graphical user interface, text, application

Description automatically generated

**A group of colorful birds

Description automatically generated with medium confidence**

## Main Steps

Color quantization consists of two main steps:

1. ***Palette Generation:*** A palette generation algorithm finds a smaller representative set of colors ***C***= *{c1,c2,c3,…,ck}*  from the *D* distinct colors.
2. ***Quantization:*** by mapping the original colors to the palette colors.

SO, we created five functions in this project to reduce the quantized image as little as possible from the original image and reduce the number of colors in a full-resolution digital color image (24 bits per pixel) to a smaller set of representative colors (***color palette***).

***Let’s know each function, what it does and how.***

**Functions Description:**

* **Get Distinct Colors.**
* **Name: getDistincitColors.**
* **input: ImageMatrix.**
* **output: List of distinct RGB pixels.**
* **Description: Extract distinct color from image matrix.**
* **Overall Complexity: O(N^2)**
* **Minimum Spanning Tree.**
* **Name: mininmumSpanningTree.**
* **input: DistinctColors.**
* **output: Array of struct of MST vertices.**
* **Description: Construction Minimum Spanning Tree.**
* **Overall complexity: O(D).**
* **Clusters Construction.**
* **Name: getKClusters.**
* **Input: array of struct of MST vertices, number of clusters, list of distinct colors.**
* **Output: dictionary composed of each distinct color and the number of clusters it belongs to.**
* **Description: adds colors with minimum edge weight to the same cluster.**
* **Overall complexity: O(K\*D).**
* **Get Cluster’s Representative Color.**
* **Name: getClusterRepresentitive.**
* **Input: dictionary of clusters, list of distinct colors.**
* **Output: dictionary composed of the ID of the cluster and an array of size 3 representing its representative color in red, green, and blue.**
* **Description: loops over the distinct colors and calculates the mean of the colors belonging to the same cluster.**
* **Overall complexity: O(D).**
* **Quantization.**
* **Name: Quantize.**
* **input: ImageMatrix, ClustersColors, Clusters, MapColor.**
* **output: returned Image matrix after reducing the number of colors in a full resolution.**
* **Description: map each color in the image matrix to his representative color in the palette.**
* **Overall complexity: O(D^2).**

### **Intensity**

The intensity of a pixel is expressed within a given range between a minimum and a maximum value [Inclusive], based on the color depth of the pixel.

True Color images have intensity from the darkest (0) and lightest (255) of three different color channels, **R**ed, **G**reen, and **B**lue. Each channel has a range from 0 to 255 as shown in Figure below. So we need 8+8+8=24 bits to represent 1 pixel color which means we have 224 = 16,777,216 different colors.

## Image Quantization Algorithm

To Apply the Single-linkage Clustering algorithm on the Image Quantization Problem, we need to:

1. Find the distinct colors *D = {d1, d2, d3 ….dm}* from theinput image. Can be known from the image histogram.

**Get Distinct Colors Function:Table

Description automatically generated**

**Description:**

**We extract the distinct color from the image in this function to reduce the number of colors in a full-resolution digital color image (*24 bits per pixel*) to a smaller set of representative colors called (*distinct colors*).   
This was returned from this function.**

**Steps:**

* **We make a 3D array with size [256,256,256] as three-channel to map colors as RGB and marked each color as visited to prevent it from repeating colors. Complexity: O(1).**
* **Loop (2D) over image matrix pixels and check whether each pixel is visited(marked) or not. Complexity: O(Height \* Width) => O(N^2).**
* **When we marked color as visited over iterations also convert color to (Hexadecimal) to using it after that as integer instead of Struct of (RGBPixels).** **Complexity: O(1).**
* **We created a dictionary called (Mapcolor) to map each color in the distinct color list with its index in it because we are using it in quantization after that. Complexity: O(1).**

**A picture containing text, scoreboard

Description automatically generated**

1. Construct a **fully-connected undirected weighted graph** *G* with

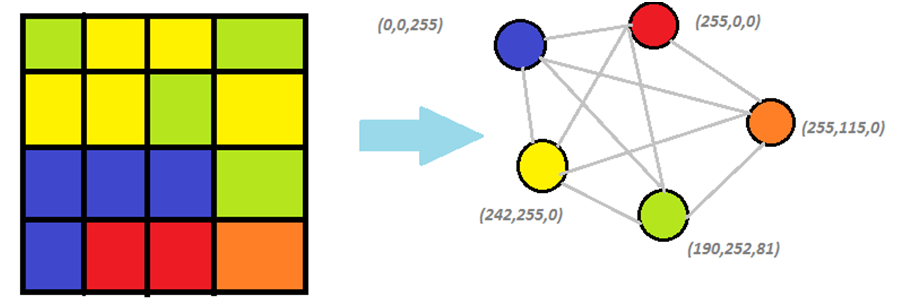
* *D* vertices (number of distinct colors).
* Each pair of vertices is connected by a single edge.
* Edge weight is set as the Euclidean Distance between the RGB values of the 2 vertices.

1. Construct  a [minimum-spanning-tree algorithm](https://en.wikipedia.org/wiki/Minimum_spanning_tree#Algorithms) (a [greedy algorithm](https://en.wikipedia.org/wiki/Greedy_algorithm) in [graph theory](https://en.wikipedia.org/wiki/Graph_theory))

* **Input:** connected undirected weighted graph
* **Output:** a tree that keeps the graph connected with minimum total cost
* **Methodology:** treats the graph as a forest and each node is initially represented as a tree. A tree is connected to another only and only if it has the least cost among all available.
* **Conclusion:** the Minimum Spanning Tree is an implementation of single linkage clustering Strategy that repeats merging distinct points with minimal distances into a single cluster



**4 x 4 image grid**

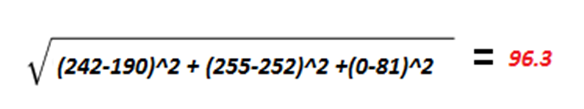
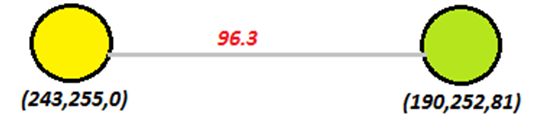


* **To get distance between two colors use *Euclidean Distance:***

The Euclidean Distance between TWO colors is defined as:







**Text

Description automatically generatedMinimum Spanning Tree:**

**Description:**

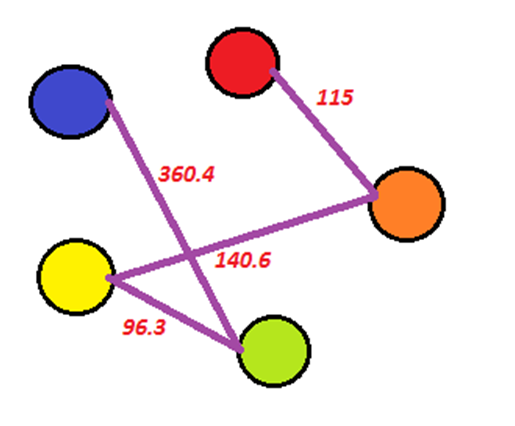
**We should execute two functions here the first is construct fully-connected undirected weighted graph and the second is Minimum spanning tree but we compress them in one function with Complexity Upper bound: O(D^2) and Exact bound:** **Θ(D\*E) to reduce Complexity of program overall.**

**Steps:**

* **Create Struct of Vertex and loop over count of vertices to store all edges in it ( From(parent) : To(child) => Weight(key) ) with initialize some attributes of struct (Key => Max\_value), (parent => -1), (child => index).Complexity: O(D^2)**
* **Initialize randomly any key of vertex with equal zero. Complexity: O(1)**
* **Into the outer while loop we marked vertex as visited and set (minimum edge) with Max\_Value and enter the inner loop and search on minimum weight over edges and update (minimum edge) then start the next iteration in outer loop with child vertex was selected then repeat this cycle till finish all vertices. Complexity: O(D^2) - Θ(D\*E)**

Diagram, schematic

Description automatically generated

****

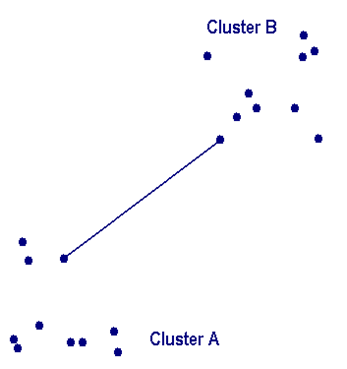
1. Extract the desired number of clusters (K) with maximum distances between each other.
2. Find the representative color of each cluster.

### **Clustering**

### Definition:

**Clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense or another) to each other than to those in other groups (clusters).

1. **Grouping of points** with minimal Distances between them in one cluster 🡪 means (is equivalent to) producing clusters with maximal spacing.



**Maximal Spacing**

1. We can assume the number of groups/clusters.

### Objective

Given a **distance measure** and a **desired number of clusters** 🡺 produce **K clusters** with maximal Spacing which means grouping distinct points with minimal distances into one cluster.

### **Single-linkage Clustering**

With single linkage method (also called nearest neighbor method), the distance between two clusters is the minimum distance between an observation in one cluster and an observation in the other cluster which is defined as the Euclidean Distance.

**Text

Description automatically generated with medium confidenceClusters Construction:**

**Description:**

**We should be converting the minimum spanning tree to clusters and each cluster is with minimal Distances between its vertices and producing maximal spacing with other clusters.**

**Steps:**

* **Create (alledges) to Sort minimum spanning tree with (Fibonacci heap) and create dictionary to store clusters in it as (key => vertex, value => index). Complexity: O(1)**
* **Loop over count of vertices and store (alledgs) to sort this list with Fibonacci heap to use it when construct clusters.  
   Complexity: O(E)**
* **Loop over count of vertices to (enqueue) all edges in queue. Complexity: O(log(E))**
* **Remove from queue specific edge with minimum weight and put source and destination in one cluster with using (Union function). Complexity: O(K\*D)**

**Union:**

**A picture containing chart

Description automatically generated**

**Description:**

**Pick vertex and search about it then change its index and any vertex has same index to destination's index.**

**Graphical user interface, text, application, email

Description automatically generatedGet Cluster’s Representative Color:**

**Description:**

**We compute the average of component RGB (Red, Blue, Green) for each cluster and each cluster has one representative color.**

**- Cluster: key => color number, value => cluster number.**

**- Cluster Colors: key => cluster number, value => representative color in   
(red, green, blue)**

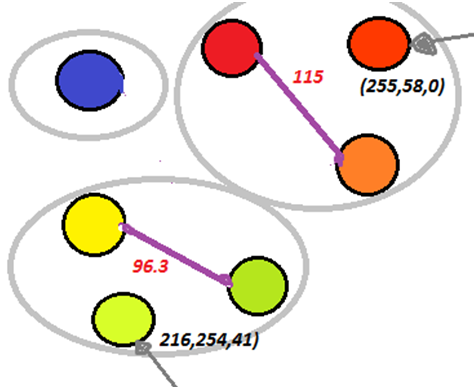
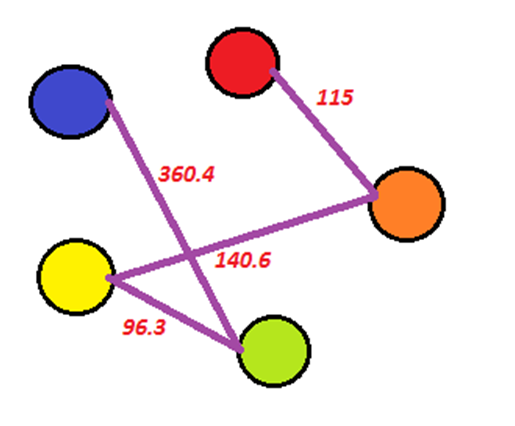
**- Distinct colors: list of RGBPixels that can be accessed by index.**

**Steps:**

* **Loop over count of vertices and each vertex then compute component of RGB (red, blue, green) cluster and add each color to its channel. Complexity: o(D)**
* **Loop over count of clusters to compute the average representative color. Complexity: o(K)**



**MST**



**Clusters (K = 3)**

**Representative color1:**

Average of the RGB of the 2 colors

R=(255+255)/2

G=(0+155)/2

B=(0+0)/2

**Representative color2:**

Average of the RGB of the 2 colors

R=(242+190)/2

G=(255+252)/2

B=(0+81)/2

**Table

Description automatically generatedQuantization:**

**Description:**

**We replace each pixel in image matrix with representative color which exist in the palette of color.**

**Steps:**

* **Loop (2D) over image matrix pixels and replace each pixel with representative color. Complexity: O(Height \* Width) => O(N^2).**
* **When we loop over image also convert color to (Hexadecimal) to map with it in distinct color list to know original color replace it with representative color. Complexity: O(1).**

**Bonus:**

**Automatically Detect Clusters:**

* **Calculate Mean:**
* **Name: calculateMean.**
* **input: alledges.**
* **output: Calculate mean.**
* **Description: Calculate mean of all edges.**
* **Overall Complexity: O(E)**
* **Calculate Standard deviation:**
* **Name: calculateStandardDeviation.**
* **input: alledges.**
* **output: Array of struct of MST vertices.**
* **Description: Calculate Standard Deviation of all edges.**
* **Overall Complexity: O(E)**
* **K-Clusters Detection:**
* **Name: KClustersDetection.**
* **input: Mean, Standard deviation of all edges.**
* **output: Number of detected clusters.**
* **Description: Detect clusters.**
* **Overall Complexity: O(E^2)**

**Description:**

**Clustering the distinct color points into disjoint groups without knowing the predefined number of clusters using the following criteria:**

1. **Computes the standard deviation of the edges in minimum spanning tree and store it**
2. **Choose an edge that leads to max standard deviation reduction once it is removed to obtain a set of two disjoint sub-trees.**
3. **Compute standard deviation reduction in current iteration.**
4. **Repeat steps (2) and (3) until convergence 🡪 the difference between standard deviation reduction in current iteration and previous iteration is very small < 0.0001**

**Complexity: O(N^2)**

**Automatically Detect Clusters:**

**A picture containing table

Description automatically generated**