|  |  |  |  |
| --- | --- | --- | --- |
| السكشن | القسم | رقم الجلوس | الاسم |
| 5 | SC | 20191700793 | يوسف نادر ميشيل صبحي |
| 6 | CS | 20191700793 | يوسف ناصر صابر بيلاطس |
| 4 | SC | 20191700460 | كيرلس نبيل منير فهمي |

Team ID: T077

Image Quantization

2021 2022

# 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.

# Graph

## Why graph?

The graph is an essential input for constructing the MST. Constructing a graph will be in two steps first one is to get the unique pixel from the image in "Graph" class. Second step is to calculate the weight (distance) between each unique pixel. This step is exhaustive process as it will take time and storage also it will take time to index it in the process of MST, so we decide to not store it and only calculate the weight (distance) that we need in MST.

## Graph construction

Graph will be constructed in "Graph" class. It consists of

* public int distinctColors
* public RGBPixel [,] ImageMatrix
* public RGBPixel [] UniqueColors
* public Graph (RGBPixel [,] ImageMatrix)
* private void GetUniqueColors ()

### Public int distinctColors

An integer variable which is initialized with zero. It refers to the number of distinct colors in the image.

### public RGBPixel [,] ImageMatrix

Array 2D that stores the image pixels as RGBPixel type.

### public RGBPixel [] UniqueColors

Array 2D that will be used to store only the unique pixels from ImageMatrix array.

### public Graph (RGBPixel [,] ImageMatrix)

A constructor that is used to initialize the ImageMatrix with the one in the parameter list of the constructor.

Calling "GetUniqueColors ()" function is the second usage for the constructor which get fill UniqueColors matrix once the user instantiate an object from the class to avoid using empty matrix.

### Private void GetUniqueColors ()

The function aims to find the unique pixel from the image by initializing an 3D Boolean array which is initially false and loop on the image pixels and mark the pixel not visited with true.

Text

Description automatically generated

Figure 1: GetUniqueColors()

## Graph complexity

The complexity of Graph class related to GetUniqueColors () function. So, the overall complexity

# Minimum spanning tree

## Why MST?

Minimum spanning tree is the way that connects all the vertices together, without any cycles and with the minimum possible total edge weight. Achieving this by using Prim algorithm.

## Prim construction

Prim algorithm is used to find the MST and it produce parent array, value array and minimumSpanningTreeCost

MST construction will be in "Prim" class, and it consists of

* private readonly Graph
* public int [] parent
* public double [] values
* public Prim (Graph graph)
* private int selectMinimumVertex (double [] values, bool [] setMST, int numberOfVertex)
* public double MST ()

### private readonly Graph graph

MST will use this variable to access the distinct color array for calculating the size of distinct colors and calculating the distances as this is the second step in calculating the map this way optimize the space and time complexity.

### public int [] parent

This array will store the parent of each vertex as the value will be the source and the index will be destination.

A picture containing text

Description automatically generatedExample: In sample test case one

A picture containing text, clock, different

Description automatically generated

Figure : Graph

Figure : parent array

### public double [] values

This array will store the value (weight) of each edge mapped to parent array

A picture containing text, clock

Description automatically generated

Figure : value array

### public Prim (Graph graph)

A constructor that initializes the graph to prepare it for MST to use it.

### private int selectMinimumVertex (double [] values, bool [] setMST, int numberOfVertex)

This function helps the MST to choose what vertex to visit next as it iterates on all vertices. This choice is based on if statement as it should be not visited, and its value is the smallest in the graph.

Finally, it returns integer number that indicates the index of the minimum weighted vertex

### public double MST ()

The main objective of this function is to calculate the minimumSpanningTreeCost and update value and parent arrays by the end of the function to form an MST.

The function starts with initializing the arrays with the default values, then iterates on (V – 1) "Number of edges" then use selectMinimumVertex () function to choose the minimum weighted vertex and start to relax the graph and update each iteration.

In graph , where V is the number of vertices and E is the number of edges so when we extract the tree

, where

A picture containing graphical user interface

Description automatically generated

Figure 5: Example 1 for color palette

## MST Complexity

### selectMinimumVertrc Complexity

Graphical user interface, text

Description automatically generated

Figure 6: SelectMinimumVertex function

The function complexity depends on the number of distinct colors (V), Complexity

### MST Complexity

Text

Description automatically generated

Text

Description automatically generated

The function complexity depends on the nested loop that iterates on the edges then search for the minimum weighted node for all vertices Complexity =

# Palette generation

## Why palette?

A picture containing text, clock

Description automatically generated As quantization is process of reduction number of colors in the image so we should know what colors to use and what colors to replace. This will happen by the helping of the color palette which mapped the similar color to their representative in the color palette.

Figure : MST

## Palette construction

"Cluster" class will be used to construct the palette. Process of constructing is based on clustering algorithm which aims to map each vertex (pixel or color) to a cluster which all members of that cluster have comparable properties then choose a representative for each cluster and put it in color palette finally replace each vertex in the image to its representative from the color palette.

The algorithm assumes that all vertices are initially in one cluster then removing (k – 1) most expensive weight from the connected MST and then apply BFS algorithm on the vertices to form k different trees each tree node is begin from the point of removal

* int numberOfClusters
* Graph graph
* Prim prim
* public Cluster (Prim prim, Graph graph, int K)
* private RGBPixel [,] GeneratePallete ()
* public RGBPixel [,] GetQuantizedImage ()

### int numberOfClusters

Number of desire clusters. Taken as an input from the "MainForm".

### Graph graph

Graph variable allows us to use UniqueColors array to map the colors.

### Prim prim

Prim variable allows us to parent and value arrays that stores the MST.

### public Cluster (Prim prim, Graph graph, int K)

A constructor that initializes graph, prim, and number of clusters once the object is instantiated to avoid accessing any empty items.

### private RGBPixel [,] GeneratePallete ()

"GeneratePalette" function is responsible for producing "K" clusters as it takes the "MST" from the prim and remove the most (K – 1) expensive weighted edges and then make sub trees.

### public RGBPixel [,] GetQuantizedImage ()

" GetQuantizedImage" function to map each pixel in "ImageMatrix" to its representative color in map (color palette).

Return the ImageMatrix so that we can show it in the form.

## Palette Complexity

### GeneratePalette

Text

Description automatically generated

The function complexity depends on

Text

Description automatically generated

Text

Description automatically generated

### GetQuantizedImage Complexity

Text

Description automatically generated

The function complexity depends on the size of the picture as we loop on the picture and replace every color by its representative from color palette (map) Complexity