A Simple Image Processing Algorithm Using Graph and Depth First Search  

by Xulong Peng  

for CSC445 Algorithm

Digital image processing is the manipulation and interpretation of digital images with the help of a computer. It is an extremely broad subject and it is used in many fields such as satellite image classification and medical image diagnosis. Digital image processing often involves procedures that can be mathematically complex. Graph is a set of objects, called nodes with pairwise connection between them. Graph traversal is a system procedure to explore a graph by search its nodes and edges. Depth First Search (DSF) is one efficient algorithm for graph exploration. Image is a fully connected graph between pixels; hence graph can be used to solve image classification problems. In this project, a simple image classification algorithm is developed based on the connected-components algorithm using a threshold value. To classify the image, a user input value is used as a threshold. If the absolute difference between the digital values of two pixels is less than the threshold, then they are similar. Otherwise, they are different. The algorithm is applied to several sample images to demonstrate its effectiveness.

1.0 Background

A digital image is a two-dimensional image represented using binary numbers, such as digital photos, satellite images and medical images. Digital image processing is a way to derive information from digital images using computer based algorithms. It can be used for image classification, feature extraction and pattern recognition. Satellite image classification is among the major application areas of digital image processing. Image classification technology has been developed since 1960s [1], with rapid advancements after the United States launched Landsat 1 in 1972[2]. In the earlier days of digital image processing, it was expensive and limited by computer hardware and software. Computer technology has led an era of information revolution in the last a few decades, which has made digital image processing a common practice and much more cost effective.

To understand digital image processing, the format of a digital image needs a brief description. A pixel is the basic unit of an image and a two-dimensional array of pixels represent a one-band image. The horizontal dimension is width and vertical dimension the height of the image. Each pixel is located by two coordinates, \( x \) and \( y \) (Figure 1), with \((0,0)\) at the upper left corner.

![Figure 1. Pixels in an image.](image-url)
Multi-band image is made up of multiple two dimensional array of pixels. It can be presented as a color image, where RGB are rendered using digital numbers from two or more bands. Thus, in a color image, each pixel contains three components and these components make up the pixel’s color. In an 8-bit image, each individual color component has a value range of 0-255. Digital image processing is the manipulation of these components to produce a result that serve a purpose to reveal certain information content, such as grouping the pixels into spectrally similar regions (classification) or detecting linear features (feature extraction). The basic idea in image processing is quite simple. The digital image is fed into a computer one pixel at a time and then the pixels are computed based on some designed algorithm. The computed results are stored in the resultant image.

2.0 Introduction

In this project, a simple image processing method, a segmentation algorithm, is developed to group the pixels into a few relatively homogeneous areas. The segmentation result would reveal a crude classification of the raw image, the interpretation of the raw data. For example, raw Landsat imagery displays land cover, but without any annotation/labeling. A human interpreter would give the scene its meaning for the raw image to make sense to average people. The same is achieved using computer classification algorithms, when the pixels in the imagery are labelled with their land cover identities. Thus, the classification process produces an information product from raw data. However, image classification algorithms can be very complex. This project implements the simplest classification algorithm based on a threshold value. The algorithm compares the digital values of one pixel to its neighboring pixels in the image and group similar pixels into the same region. A graph and its traversal, like Depth First Search (DFS), fit this application very well. Graph is a set of objects, called nodes with pairwise connection between them. The applications of graphs are widespread and diverse. Graph traversal is a system procedure to explore a graph by search its nodes and edges [3]. Depth First Search (DFS) is one efficient algorithm for graph exploration. Image is a fully connected graph between pixels; hence graph can be used to solve problems such as image classification.

3.0 Method

To implement this algorithm, a threshold value $t$ is provided by the user. Based on this value $t$, pixels are calculated. Two pixels are similar if the absolute difference between each component (color band) is less than threshold $t$. For example, the red, green and blue color components of pixel $p$ are $R_p$, $G_p$, $B_p$. Then pixels $p$ and $q$ are similar if $|R_p - R_q| < t$ and $|G_p - G_q| < t$ and $|B_p - B_q| < t$. The algorithm used in this project works as follows. An undirected graph is used to represent the image and each image pixel is a graph vertex and there is an edge between any two similar neighboring pixels (Figure 2). If there is a path of similar pixels that starts at $p$ and ends at $q$, then these pixels are in the same region (class or segment). To compute these segments, a graph is to be constructed first. Then all neighboring pixels which are similar are connected by edges. Next, compute connected components on this graph using Depth First Search (DFS) for this connected component computation. Each connected component is considered as the same segment of the image. DFS is useful for performing a number of computations on graphs, such as finding a path from one vertex to another, determining whether or not a graph is connected and computing a spanning tree of a connected graph etc. Both recursive and non-recursive algorithms can be used for DFS computation. Even recursive algorithm is very efficient as it makes code very simple; however a non-recursive algorithm for DFS is used for this project because it is an image graph.
application. Since images are large, the recursive algorithm may run out of memory. Instead, it uses an iterator method to perform DSF. This method returns an iterator over the names of vertices visited and uses a stack to implement the non-recursive algorithm. For each newly visited vertex, it puts all (unmarked) adjacent vertices on the stack. The top of the stack is the next vertex which will be visited according to the DSF order. The pseudo-code is below.

**Algorithm Iterator DFS (G, v):**
Create a Linked list dfs, a stack and a Linked list visitlist .
Push v into stack
Add v to visitlist
While loop until stack is empty
  Pop v from stack
  Add v to dfs list
  While loop to go through iterator
    If visitlist does not contain v, push v into stack
    Otherwise pop from stack.

Finally, after segments are computed, the average pixel value of the segment is assigned to the entire segment. The result is saved as the output image. Since it uses average values, the output image is smoothed out after applying this algorithm. The average color in each segment is computed by averaging RGB bands separately over all pixels in the same segment.

The project is implemented by using Java programming language. All classes used in this project are listed below:

a. **Pixel class**
   Represents a pixel and has methods:
   Public pixel (int x, int y) //constructor
   Public int getX() and getY() //return x and y coordinates of a pixel

b. **ImageReader class**
   Used to read/write to an image and has methods:
   getHeight(), getWidth(), getRed(), getGreen(), getBlue(), setColor() and saveimage().

c. **Vertex class**
   Represents a vertex of the graph.

d. **ImageGraph class**
   Represents a graph constructed from an image and it also implements ImageGraphInterface.
   It has methods:
   ImageGraph() //constructor
   addVertex(), boolean isAdjacent(), addEdge(), Iterator Adjacent,
   getNumVertices(), Iterator DFS(), Iterator ConnectedComponents().

e. **GraphException class**
   For the exception of the graph

f. **ImageClassification class**
   Has the main() method to run the program
4.0 Results
Six small sample images are collected for testing: Google icon, Chrysanthemum, Tulips, Penguins, colored squares and colored board. The colored squares uses thresholds 100/150 and colored board uses threshold 30, 40, 48, 49, 100 for comparison and others use 10 as threshold value. The outputs include two parts. The first part is the saved resultant image after the input image is processed. The other part is the printing message which includes the number of connected components, the number of pixels in each segment and the average color value in each segment. In terms of running time, DSF algorithm in this project is an efficient method for traversing. DSF is called exactly once on each vertex. Thus, it is $O(\text{number of vertices visited})$ and not worse than $O(\text{number of vertices + number of edges})$. Figures 3-8 shows the original and output images side-by-side, and the printing message. All outputs are attached in the end of this paper.

![Google icon](image1)

(a) original  
(b) output  
Figure 3. Google icon (threshold: 10)

![Chrysanthemum](image2)

(a) Original  
(b) output  
Figure 4. Chrysanthemum (threshold: 10)

![Tulips](image3)

(a) Original  
(b) Output  
Figure 5. Tulips (threshold: 10)
Figure 6. Penguins (threshold: 10)

(a) Original          (b) Output

Figure 7. Colored Squares (threshold: 100/150)

(a) Original  (b) output(threshold: 100) (c) output(threshold: 150)

(c) Printing message (threshold: 100).

Figure 7. Colored Squares (threshold: 100/150)
Figure 8. Color Board (threshold: 30-100)

The print out messages associated with results above is below:

```plaintext
C:\ImageClassification\src> java ImageClassification c1.jpg c1out150.jpg 150
Segment 1 of size 85234: Red 233 Green 241 Blue 218
Segment 2 of size 3456: Red 1 Green 52 Blue 78
Segment 3 of size 3510: Red 1 Green 68 Blue 102
Segment 4 of size 3456: Red 131 Green 0 Blue 1
Segment 5 of size 3564: Red 2 Green 76 Blue 113
Segment 6 of size 3510: Red 171 Green 0 Blue 1
Segment 7 of size 3456: Red 1 Green 105 Blue 81
Segment 8 of size 3564: Red 189 Green 1 Blue 2
Segment 9 of size 3510: Red 1 Green 155 Blue 117
Number of segments is 9

C:\ImageClassification\src>
```
Segment 1 of size 5013: Red 253 Green 254 Blue 253
Segment 2 of size 2: Red 217 Green 253 Blue 249
Segment 3 of size 1: Red 1 Green 174 Blue 144
Segment 4 of size 3507: Red 1 Green 195 Blue 146
Segment 5 of size 2: Red 37 Green 165 Blue 180
Segment 6 of size 3293: Red 192 Green 192 Blue 2
Segment 7 of size 3456: Red 1 Green 52 Blue 78
Segment 8 of size 1: Red 75 Green 135 Blue 49
Segment 9 of size 3456: Red 154 Green 154 Blue 2
Segment 10 of size 3348: Red 1 Green 68 Blue 123
Segment 11 of size 3456: Red 131 Green 0 Blue 1
Segment 12 of size 63: Red 39 Green 86 Blue 112
Segment 13 of size 3402: Red 172 Green 0 Blue 1
Segment 14 of size 3456: Red 1 Green 196 Blue 81
Segment 15 of size 3339: Red 1 Green 75 Blue 114
Segment 16 of size 3294: Red 190 Green 0 Blue 1
Segment 17 of size 3510: Red 1 Green 156 Blue 117
Segment 18 of size 3456: Red 152 Green 152 Blue 3
Segment 19 of size 1: Red 242 Green 248 Blue 282
Number of segments is 19

Segment 1 of size 5016: Red 253 Green 254 Blue 253
Segment 2 of size 3510: Red 1 Green 195 Blue 146
Segment 3 of size 3294: Red 192 Green 192 Blue 2
Segment 4 of size 3456: Red 1 Green 52 Blue 78
Segment 5 of size 3456: Red 154 Green 154 Blue 2
Segment 6 of size 3348: Red 1 Green 68 Blue 123
Segment 7 of size 3456: Red 131 Green 0 Blue 1
Segment 8 of size 3402: Red 1 Green 76 Blue 114
Segment 9 of size 3402: Red 172 Green 0 Blue 1
Segment 10 of size 3456: Red 1 Green 196 Blue 81
Segment 11 of size 3294: Red 190 Green 0 Blue 1
Segment 12 of size 3510: Red 1 Green 156 Blue 117
Segment 13 of size 3456: Red 152 Green 152 Blue 3
Number of segments is 13
<table>
<thead>
<tr>
<th>Segment 1 of size</th>
<th>Red 253</th>
<th>Green 254</th>
<th>Blue 253</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segment 2 of size</td>
<td>Red 1</td>
<td>Green 195</td>
<td>Blue 146</td>
</tr>
<tr>
<td>Segment 3 of size</td>
<td>Red 192</td>
<td>Green 192</td>
<td>Blue 2</td>
</tr>
<tr>
<td>Segment 4 of size</td>
<td>Red 1</td>
<td>Green 52</td>
<td>Blue 78</td>
</tr>
<tr>
<td>Segment 5 of size</td>
<td>Red 154</td>
<td>Green 154</td>
<td>Blue 2</td>
</tr>
<tr>
<td>Segment 6 of size</td>
<td>Red 1</td>
<td>Green 68</td>
<td>Blue 103</td>
</tr>
<tr>
<td>Segment 7 of size</td>
<td>Red 131</td>
<td>Green 0</td>
<td>Blue 1</td>
</tr>
<tr>
<td>Segment 8 of size</td>
<td>Red 1</td>
<td>Green 76</td>
<td>Blue 114</td>
</tr>
<tr>
<td>Segment 9 of size</td>
<td>Red 172</td>
<td>Green 0</td>
<td>Blue 1</td>
</tr>
<tr>
<td>Segment 10 of size</td>
<td>Red 1</td>
<td>Green 186</td>
<td>Blue 81</td>
</tr>
<tr>
<td>Segment 11 of size</td>
<td>Red 190</td>
<td>Green 0</td>
<td>Blue 1</td>
</tr>
<tr>
<td>Segment 12 of size</td>
<td>Red 1</td>
<td>Green 156</td>
<td>Blue 117</td>
</tr>
<tr>
<td>Segment 13 of size</td>
<td>Red 152</td>
<td>Green 152</td>
<td>Blue 3</td>
</tr>
</tbody>
</table>

Number of segments is 13

<table>
<thead>
<tr>
<th>Segment 1 of size</th>
<th>Red 253</th>
<th>Green 254</th>
<th>Blue 253</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segment 2 of size</td>
<td>Red 1</td>
<td>Green 195</td>
<td>Blue 146</td>
</tr>
<tr>
<td>Segment 3 of size</td>
<td>Red 192</td>
<td>Green 192</td>
<td>Blue 2</td>
</tr>
<tr>
<td>Segment 4 of size</td>
<td>Red 1</td>
<td>Green 52</td>
<td>Blue 78</td>
</tr>
<tr>
<td>Segment 5 of size</td>
<td>Red 154</td>
<td>Green 154</td>
<td>Blue 2</td>
</tr>
<tr>
<td>Segment 6 of size</td>
<td>Red 1</td>
<td>Green 68</td>
<td>Blue 103</td>
</tr>
<tr>
<td>Segment 7 of size</td>
<td>Red 131</td>
<td>Green 0</td>
<td>Blue 1</td>
</tr>
<tr>
<td>Segment 8 of size</td>
<td>Red 1</td>
<td>Green 76</td>
<td>Blue 114</td>
</tr>
<tr>
<td>Segment 9 of size</td>
<td>Red 172</td>
<td>Green 0</td>
<td>Blue 1</td>
</tr>
<tr>
<td>Segment 10 of size</td>
<td>Red 1</td>
<td>Green 186</td>
<td>Blue 81</td>
</tr>
<tr>
<td>Segment 11 of size</td>
<td>Red 190</td>
<td>Green 0</td>
<td>Blue 1</td>
</tr>
<tr>
<td>Segment 12 of size</td>
<td>Red 1</td>
<td>Green 156</td>
<td>Blue 117</td>
</tr>
<tr>
<td>Segment 13 of size</td>
<td>Red 152</td>
<td>Green 152</td>
<td>Blue 3</td>
</tr>
</tbody>
</table>

Number of segments is 12
5.0 Conclusions

In this project, the algorithms developed are efficient for the image segment computation and results based on the algorithm are satisfactory. We can draw the following conclusions:

a. Graph is very useful for image processing;

b. Non-recursive DFS is an alternate algorithm if memory is a concern;

c. Threshold is very important and should be chosen carefully as it affects the output;

d. Connected components algorithms used in this project are useful for digital imagery classification and it can be developed further for complex image processing.

References