Data structure and algorithm analysis project -- Intelligent Scissors

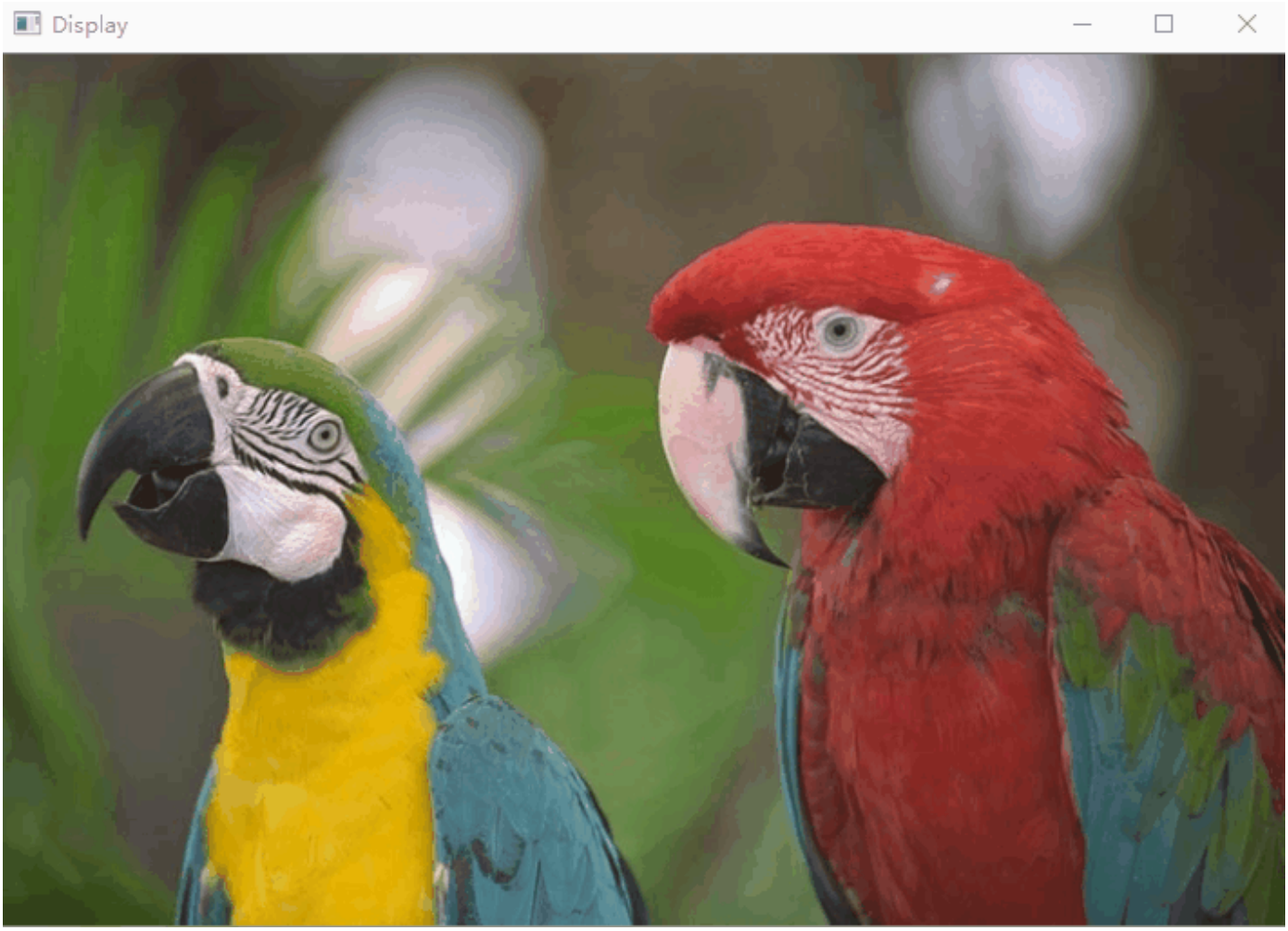
Introduction

Image segmentation is simply breaking down an image into segments. This process is beneficial as it can help us identify objects in an image and can also make image analysis easier.

For successful image segmentation, it is essential that the objects within an image are correctly selected. Selection of an image depends on two factors, object identification and then its boundary selection. Boundary selection; therefore, has to be accurate so that the extraction process is accurate.

> The problem that is encountered is that “fully automated segmentation is an unsolvable problem, while manual tracing is inaccurate and laboriously unacceptable…;[thus, we need a tool that] allow[s] objects within digital image to be extracted quickly and accurately using simple gesture motions with a mouse” (Mortensen & Barrett, 1995).

A solution for this problem was proposed by Eric N. Mortensen and William A. Barrett in 1995 which got published under the name “Intelligent Scissors for Image Composition”. The way it works is that the user first clicks on a "seed point" which can be any where in the image. The program then computes a path from the seed point to the mouse cursor that hugs the contours of the image as closely as possible. **This path, called the "live wire", is computed by converting the image into a graph where the pixels in the picture correspond to nodes. Each node is connected by links to its 8 immediate neighbors.** Note that we use the term "link" instead of "edge" of a graph to avoid confusion with edges in the image. Each link has a cost relating to the derivative of the image across that link. The path is computed by finding the minimum cost path in the graph, from the seed point to the mouse position. The path will tend to follow edges in the image instead of crossing them, since the latter is more expensive. The path is represented as a sequence of links in the graph.



Detailed Description

Step 1: Convert Image into Graph

Image is constructed of small boxes, each box is known as a pixel, and each pixel has a number associated with it representative of its intensity value. To simplify the explanation, let's first assume that the image is grayscale instead of color (each pixel has only a scalar intensity, instead of a RGB triple).

1. **Take a image and convert it into its pixel values.**

This image is originally 1000x668 in size.

For our ease of understanding, lets consider this is image is 7x7 in size.

Conversion code

7	4	2	6	1	0	3
5	9	4	8	2	3	1
3	0	1	7	4	5	2
1	8	5	3	6	1	9
4	3	7	1	3	2	8
6	0	5	4	4	1	3
3	2	6	8	1	7	5

Pixel Values

Side note: For the curious ones out there, this is what 7x7 looks in real life:

7x7 in real life

2. **Find Ix and IY for every pixel values using the Kernels (Sobel Kernel):**

- **Sx Kernel (Detecting horizontal changes)**

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

- **Sy Kernel (Detecting vertical changes)**

$$S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

> This means we calculate the value I_x and I_y for each pixel.

3. **Find the value of G for every pixel.**

To calculate the value of G, we will use the formula that calculate differences when working in the Cartesian plane.

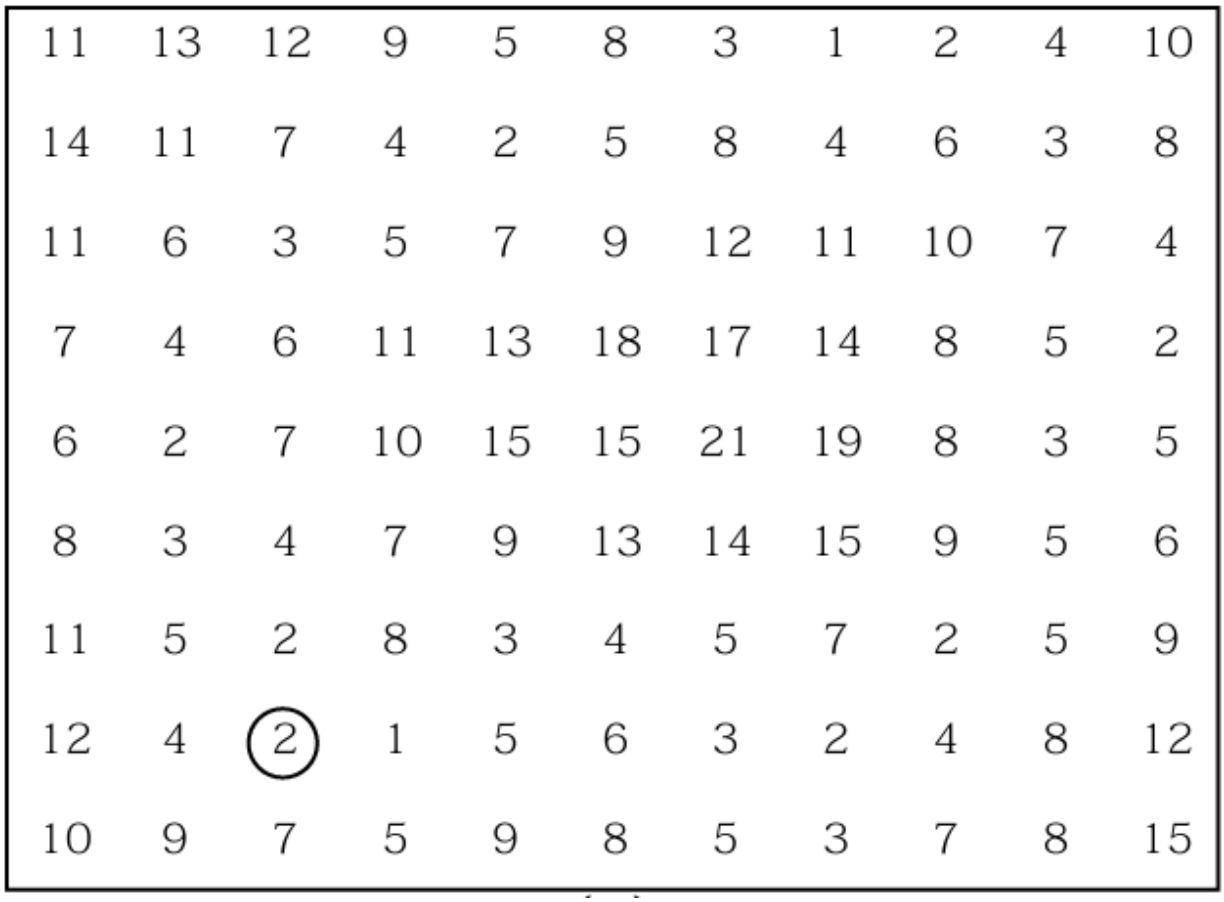
$$G = \sqrt{(I_x^2 + I_y^2)}$$

4. **Find the value of every node (pixel) using**:

$$f_G = \frac{G_{\max} - G}{G_{\max}}$$

Step 2: Computes minimum cost path from seed to all other pixels

1. **Select a seed points**



Assuming your image looks like the picture above after the first part is completed, we start from the circle (start point).

2. **Cost Function**

$$C(x, y) = \frac{1}{1 + G(x, y)}$$

3. **Apply the Dijkstra Algorithm to determine the shortest path**

You now have a complete graph and the cost between nodes. In this step, you will use Dijkstra's algorithm to calculate the distance between nodes in the graph. Furthermore, when the user selects a starting point, you can calculate the shortest path from the starting point to the mouse drop point.

Project Requirement

You need to complete the content mentioned in the first two parts. You will also have to complete a GUI that interacts with the user, which should be able to:

- Upload and display images

- Display the shortest path drawn by your program

We list the detailed indicators and scores here:

- Write the system described above.(60 points)
- Write your report. (10 points)

1. **Write the system decribed above.** First of all, you should make such a system.

- (40 points)Evaluate how precisely the algorithm detects and follows object boundaries. This includes assessing its ability to adhere to edges and produce accurate segmentation results.
- (20 points)The program should be able to run and show the movement of the objects with GUI.

2. **Write your report.**
   Include everything you want us to know about your project. Such as the basic structure of your project, the data structures you have used in the project, the way you optimize the performance, and anything that makes your project different from others.
   Clearly state how to run your program in your report. It's necessary when we test your program.
   You may also include some statistics of your program, such as those mentioned in the reference links.
   However, we have no requirement on the length of the report. Do **NOT** try to make it unnecessarily long.
   Remember to include group members and student numbers. Also remember to upload your report in "**pdf**" format.
3. Extra test
   You may noticed that the above parts add up to 70 points. We will run and test your programs to determine the last 30 points. This part involves **Cursor Snap** and **Path Cooling**.

- **Cursor Snap**(15 points): Implement real-time gradient analysis within the user-defined neighborhood to dynamically adjust the cursor position, ensuring it aligns with significant edge features.
- **Path Cooling**(15 points): Develop a mechanism to monitor path stability by tracking the coalescence of optimal paths. Automatically generate new seed points at the end of stable segments to facilitate seamless boundary tracing.

Other requirements

- There are restrictions on what you can use and cannot use in this project. You are allowed to use algs4 library or LWJGL. No other 3rd party library are allowed.
- Finish the work in group of 3. Less then 3 is allowed but more than 3 isn't allowed. Grouping across lab classes are allowed. Remember to register your group here: https://docs.qq.com/sheet/DQUhSZkVLZGRtU0xv?tab=BB08J2.
- When you upload your work, only one member needs to upload. Upload report pdf file and zip archive of your source code. Remember that when we test your code, we only use your source

code. Any .class file, .jar file, .dll file, .so file will be ignored.

Reference materials

1. Intelligent Scissors for Image Composition

> This serves as the best material for you to understand this algorithm

2. 如何通过缝隙切割图像-Intelligent Scissors