Leyla YAYLADERE
5 January 2020

REPORT OF ASSIGNMENT 3

1. Problem Description

We should write a program which includes at least two static methods in addition to our main method and also, we are not allowed to use statements that we haven't learned in class. Our program should make some calculations on 3D pixel arrays of some images. In order to make this, we should turn a ppm file into a 3D array. In the first part, we will write this 3D array to another ppm file. In the second part, we will turn the image to black and white version by calculating color-channel based average. Third, we will implement convolution operation using a 2D array filter on the image. In the end, we will control the values of neighboring pixels by using recursion, in order to modify these pixels to be equal if they are in the same given range. This process is called color quantization.

2. Problem Solution

I wrote 6 static methods in addition to my main method. One of them (fileIntoArray) is in order to read a ppm image file into 3D array. I implemented this method only once in the beginning and I turned a ppm image taken as argument2 into 3D array. Another method named arrayToFile writes the 3D array to another ppm file and requested file name is taken as a parameter. I implemented this method 4 times, once for each mode, because required name for ppm file is different in different modes. Another method called blackAndWhite calculates color-based channel average for each pixel and write this value to all color channels of a pixel. Another method named filterIntoArray reads a filter file into 2D array. Another method called convolution applies convolution with filter given as a parameter to each color channel of the image separately. My last method called quantization check the values of the neighboring pixels to see if they are within a given range and modify these pixels to be equal if they are in the same range. In order to control the values of the neighboring pixels of each pixel, I used recursive method.

I handled with modes by using if statements in my main method.

3 . Implementation

```
package leyla;

import java.util.*;

import java.io.*;

public class LY2018400216 {

	public static void main (String[] args) throws FileNotFoundException
{

		int mode = Integer.parseInt(args[0]);
```

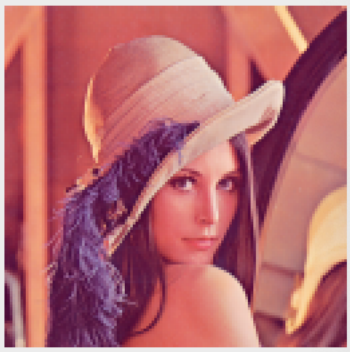
```
            String inputFile = args[1];
            File input = new File(inputFile);
            Scanner input1 = new Scanner(input);
            input1.nextLine();
            int column = input1.nextInt();
            int row = input1.nextInt();
            int range = input1.nextInt();

            int[][][] arr = fileIntoArray(input);

            if(mode==0) {
            arrayToFile(arr , "output.ppm", range);
            }

            if(mode==1) {
            arr = blackAndWhite(arr);
            arrayToFile(arr,"black-and-white.ppm", range);
            }

            if(mode==2) {
            String inputFile2 = args[2];
            File filter = new File(inputFile2);
            int[][] filterArr = filterIntoArray(filter);
            arr = blackAndWhite(convolution(arr, filterArr, range));
            arrayToFile(arr, "convolution.ppm", range);
            }

            if(mode==3) {
            int rangeQuantization = Integer.parseInt(args[2]);
            int[][][] control = new int [arr.length][arr[0].length][3];
            for(int z=0; z<3; z++) {
                  for(int x=0; x<arr.length; x++) {
                        for(int y=0; y<arr[0].length; y++) {
                              arr = quantization(arr, control,
rangeQuantization, x, y, z, x, y, z);
                        }
                  }
            }
            arrayToFile(arr, "quantized.ppm", range);
            }

      }

      public static int[][][] fileIntoArray (File inputFile) throws
FileNotFoundException { // this method read a ppm image file into 3D array

            Scanner input2 = new Scanner(inputFile);
            input2.nextLine();
            int column = input2.nextInt();
            int row = input2.nextInt();
            int range = input2.nextInt();
            int[][][] arr = new int[row][column][3];

            for(int i=0; i<row; i++) {
                  for(int k=0; k<column; k++) {
                        for(int j=0; j<3; j++) {
                              arr[i][k][j] = input2.nextInt();
                        }
                  }
            }
            return arr;
      }
```

```java
	public static void arrayToFile (int[][][] arr , String name, int range) throws FileNotFoundException { // this method write the 3D array to another ppm file, requested file name is taken as parameter
		PrintStream output = new PrintStream(new File(name));
		int row = arr.length;
		int column = arr[0].length;
		output.println("P3");
		output.println(column + " " + row);
		output.println(range);
		for(int i=0; i<row; i++) {
			for(int k=0; k<column; k++) {
				for(int j=0; j<3; j++) {
					output.print(arr[i][k][j]+ " ");
				}
				output.print("\t");
			}
			output.print("\n");
		}
	}

	public static int[][][] blackAndWhite (int[][][] arr) {	// this method calculate color-based channel average and write this value to all color channels
		int row = arr.length;
		int column = arr[0].length;
		int average = 0;
		for(int i=0; i<row; i++) {
			for(int k=0; k<column; k++) {
				for(int j=0; j<3; j++) {
					average+=arr[i][k][j];
					if(j==2) {
						average /= 3;
						arr[i][k][0]= average;
						arr[i][k][1]= average;
						arr[i][k][2]= average;
						average = 0;
					}
				}
			}
		}
		return arr;
	}

	public static int[][] filterIntoArray (File filter) throws FileNotFoundException { // this method read a filter file into 2D array
		Scanner input1 = new Scanner(filter);
		String s = input1.nextLine();
		s = s.substring(0,s.indexOf("x"));
		int length = Integer.parseInt(s);
		int[][] arr = new int [length][length];
		for(int i=0; i<length; i++) {
			for(int k=0; k<length; k++) {
				arr[i][k] = input1.nextInt();
			}
		}
		return arr;
	}

	public static int[][][] convolution (int[][][] ppm, int[][] filter, int range) { // this method will apply convolution with filter given as a parameter to each color channel of the image separately
		int lengthOfFilter = filter.length;
		int horizontalLengthOfPpm = ppm[0].length;
```

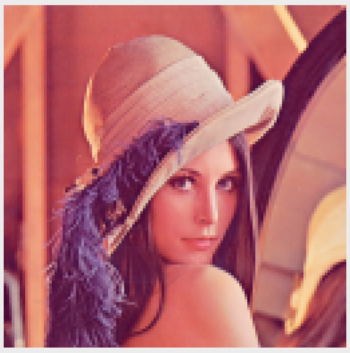
```
            int verticalLengthOfPpm = ppm.length;
            int[][][] newPpm = new int [verticalLengthOfPpm -
(lengthOfFilter-1)][horizontalLengthOfPpm - (lengthOfFilter-1)][3]; // new
ppm array will be smaller because of filter
            int horizontalStart = 0;
            int verticalStart = 0;
            int total = 0;
            for(int j = 0; j < 3; j++) {
                  while(verticalStart+lengthOfFilter-1!
=verticalLengthOfPpm) {
                        while(horizontalStart+lengthOfFilter-1!
=horizontalLengthOfPpm) {
                              for(int k = verticalStart; k < verticalStart
+ lengthOfFilter; k++) {
                                    for(int i = horizontalStart; i <
horizontalStart + lengthOfFilter; i++) {
                                          total += ppm[k][i][j] * filter
[(k-verticalStart)%lengthOfFilter][(i-horizontalStart)%lengthOfFilter];
                                    }
                              }
                              newPpm[verticalStart][horizontalStart][j] =
Math.min(Math.max(total, 0),range); // in order to turn negative values to
zero and turn the values being larger than range into range
                              total = 0;
                              horizontalStart++;
                        }
                        horizontalStart = 0;
                        verticalStart++;
                  }
                  horizontalStart = 0;
                  verticalStart = 0;
            }
            return newPpm;
      }

      public static int[][][] quantization (int[][][] ppm, int[][][]
control, int rangeQuantization, int x, int y, int z, int xStart, int
yStart, int zStart) {   // this method check the values of the neighboring
pixels to see if they are within a given range and modify these pixels to
be equal if they are in the same range
            if(!(ppm[xStart][yStart][zStart]-rangeQuantization<=ppm[x][y]
[z] && ppm[xStart][yStart][zStart]+rangeQuantization>=ppm[x][y][z])) { //
in order to stop process if the value is not in a given range
                  return ppm;
          }
            else {
                  if(control[x][y][z]==0) { // in order to not change the
value if it has changed once
                        ppm[x][y][z] = ppm[xStart][yStart][zStart];
                        control[x][y][z] = 1;
                        if(y < ppm[0].length-1)
                        quantization(ppm, control, rangeQuantization, x,
y+1, z, xStart, yStart, zStart);
                        if(y > 0)
                        quantization(ppm, control, rangeQuantization, x,
y-1, z, xStart, yStart, zStart);
                        if(x < ppm.length-1)
                        quantization(ppm, control, rangeQuantization, x+1,
y, z, xStart, yStart, zStart);
                        if(x > 0)
                        quantization(ppm, control, rangeQuantization, x-1,
y, z, xStart, yStart, zStart);
                        if(z < 2)
```

```
                        quantization(ppm, control, rangeQuantization, x,
y, z+1, xStart, yStart, zStart);
                        if(z > 0)
                        quantization(ppm, control, rangeQuantization, x,
y, z-1, xStart, yStart, zStart);
                    }
            }
            return ppm;
      }
}
```

4. Output Of the Program



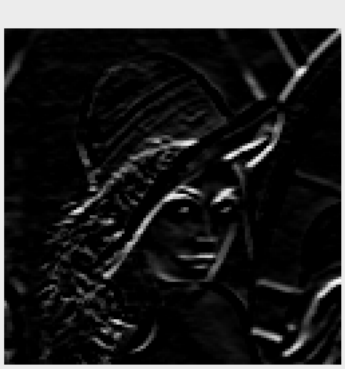
input.ppm

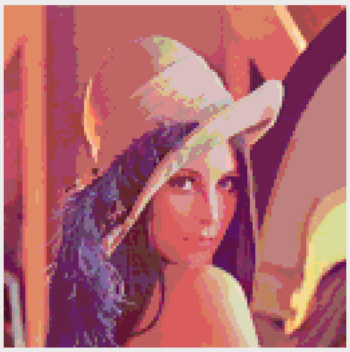


output.ppm

black-and-white.ppm


convolution.ppm


convolution.ppm


quantized.ppm

5. Conclusion

I think that I solved the problem correctly and my code does not contain too much redundant lines. I believe that all assignments given in first semester improve my programming skills but this one satisfies me most. Also, when I make an effort to do assignments, I really feel that my skills progress step by step because the time spent is decreasing and I enjoy more.