Image Processing - HW2

Delivery date: Wednesday, April 13

In this assignment you are asked to write an image editor. You can imagine an image as a two-dimensional array of pixels, the size of the image itself. For example, an image of 712x512 pixels corresponds to a two-dimensional array of pixels of 712 columns and 512 rows (we usually refer to the width first and then the height of the image).

The methods and manufacturers specified in this paper are the minimum. You can also specify additional methods/constructors if you deem it necessary.

The ce326.hw2.Image interface

An image editing program allows you to apply some image processing procedures. This work defines the interface ce326.hw2.Image which defines the following methods of changing the image.

1. **public void grayscale():** converts the image to black and white.
2. **public void doublesize():** doubles the image size.
3. **public void halfsize():** doubles the image size.
4. **public void rotateClockwise():** rotates the image by90 degrees clockwise.

The RGB type image

In the RGB image type, each pixel depicts the brightness values for the three primary colors red, green and blue, (also known as RGB from the initials of the words red, green, blue). We consider brightness values to be natural numbers from 0 to MAX_ COLOR (typically MAX_ COLOR is 255).

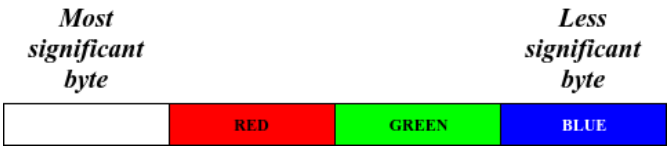
Examples of RGB values are as follows:

1. One Red pixel has RGB values of 255, 0, 0 (maximum brightness value for red and zero for the rest).
2. One yellow pixel results from RGB values 255, 255, 0 (yellow results from mixing red with green color)
3. OneWhitepixel results from the RGB values 255, 255, 255 (maximum brightness for all colors wounds)
4. One pixel results from RGB values 0, 0, 0 (minimum brightness for all colors)

You can try different color combinationshere or by using some image editing program (e.g. gimp).

The ce326.hw2.RGBPixel class

The RGBPixel class describes a pixel that contains the luminance information for the three primary colorsRed,Greenandblue. The information of the three colors can be stored in the following two alternative ways:



- in oneprivate field of type int. An example of such storage is given in the figure above.

- in three type fieldsbytes. To achieve this, you need to add and remove the value 128 every time you want to read or write the color luminance value of a pixel. More specifically, the 0 to 255 scale should be normalized to the -128 to 127 scale.

    - Before storing the information of a color in ayou subtract 128 pixels.
    - When reading the value of onepixel to make the appropriate calculations you first store the information in a variable of type short and then add 128.

The RGBPixel class has the following constructors:

- **public RGBPixel(short red, short green, short blue)**: Creates a pixel based on red, green, blue values.
- **public RGBPixel(RGBPixel pixel):**Creates an object that is precise copy of the pixel.

- **public RGBPixel(YUVPixel pixel):**Creates oneRGBPixel from a YUVPixel (explained below).

and them**public**methods:

- **short getRed():**returns the value of the red color.
- **short getGreen()**: returns the value of the green color.
- **short getBlue()**: returns the value of the blue color.
- **void setRed(short red):**sets the value of the red color.
- **void setGreen(short green):**sets the value of the green color.
- **void setBlue(short blue)**: sets the blue color value.
- **int getRGB():**Returns an integer containing the3 RGB colors as described above.

- **void setRGB(int value):**Sets the values of the three colors based on the integervalue. The integer variable value contains the 3 RGB colors.
- **final void setRGB(short red, short green, short blue):**Sets the values of the three colors based on the values of the variables**red, green, blue**.
- **String toString():**Returns an alphanumeric in the form “ **RGB** ”, where**R**its price red,**G**the price of green and**B**the value of blue color.

The ce326.hw2.RGBImage class

The RGBImage classimplements the interface**ce326.hw2.Image** . Image pixels are composed of objects of the RGBPixel class. The class should have the following constructors.

1. **public RGBImage():***Default*constructor without functionality. Required by constructor of the inherited class PPMImage.
2. **public RGBImage(int width, int height, int colordepth)**: Creates an RGB image with width dimensions**width**and height**height**and maximum brightness value**colordepth**.
3. **public RGBImage(RGBImage copyImg)**: Creates an RGB image from another RGB image. The new image is a copy of the original (copy constructor).
4. **public RGBImage(YUVImage YUVImg)**: Creates an RGB image from a YUV image. The YUV image will be explained next.

In addition to the above constructors the RGB class should have the following methods

- **int getWidth():**Returns an integer corresponding to the width value of the image.
- **int getHeight():**Returns an integer corresponding to the height value of the image.
- **int getColorDepth():**Returns an integer corresponding to the depth value image color.
- **RGBPixel getPixel(int row, int col):**returns the type objectPixel corresponding to row, col of the image table.
- **void setPixel(int row, int col, RGBPixel pixel):**sets the objectpixels in row, col of the image array.

Finally, the class has the public integer constant MAX_COLORDEPTH with a value of 255.

Convert the image to black and white (grayscale method)

A grayscale image is an image in which the three RGB values of each pixel have the same value. For example RGB values 0 0 0 represent the color black, values 255 255 255 white and values 128 128 128 the specific shade of gray.

Write a method that converts the image pixels to*grayscale*based on the following formula

Gray = Red * 0.3 + Green * 0.59 + Blue * 0.11

Doubling the image size (doublesize method)

Write a method that doubles the size of the image. To double the image, the brightness of the pixel at the random position is required**row, col**to be copied to the positions

- **2*row, 2*col,**
- **2*row+1, 2*col,**
- **2*row, 2*col+1**
- **2*row+1, 2*col+1**

Doubling the size of the image (halfsize method)

Write a method that doubles the size of the image. Image doubling requires the brightness of the pixel at the random location**row, col**to arise from average of pixel brightness at positions

- **2*row, 2*col,**
- **2*row+1, 2*col,**
- **2*row, 2*col+1**
- **2*row+1, 2*col+1**

Rotate clockwise by 90The(rotateClockwise method)

Write a method that rotates the image by 90Theclockwise.

The PPM format for RGB images

In this project you will use color images that are saved to read from a file and save to a file **in text format**and have an ending**.PPM**(PPM format). The format of a PPM file is as follows:

1. It starts with the alphanumeric**P3**.
2. Followed by an integer corresponding to the width of the image (in pixels).
3. Followed by an integer corresponding to the height of the image (in pixels).
4. Next is an integer corresponding to the maximum brightness value of the image.

Points 1-4 make up the header of the file.

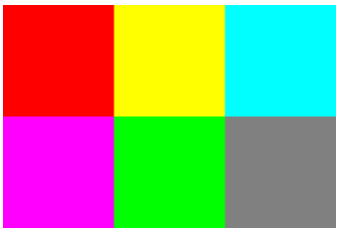
5. Then for each pixel 3 integers are displayed for the three RGB colors Red,Greenand bluein order. The file starts with the information of the top left pixel (we consider the first row of pixels, the top row), followed by the immediately rightmost pixel, until we reach the rightmost element of the top row. The file then continues from the leftmost element of the next row and so on.

For example, if we have an image of size**25x25**of pixels in the file we will have stored **25x25x3**integers corresponding to its pixel information image. It is obvious that the value of each integer must not exceed the maximum brightness value set at the beginning of the file (in point 4).

Any alphanumeric characters within the file are separated from the rest of the information by one or more blank characters,**tab**or newline characters or combinations of the above (whitespace characters).

Below is an example PPM sized image**2x3**and next to it the image corresponding to this is enlarged. The following content corresponds to the text file**test.ppm**where you supplied ready for your program control.

```
P3
3 2
255
255 0 0  255 255 0  0 255 255
255 0 255  0 255 0  128 128 128
```



A PPM file can be viewed on Linux from any image viewer or editor (e.g. gwenview,gimp). On Windows you can view it viagimp or through the programOpenSeeIt (no installation required). Alternatively, you can view the photos andonline here .

The class ce326.hw2.UnsupportedFileFormatException

The class**UnsupportedFileFormatException**is a descendant of the class **java.lang.Exception**. An exception of this type is thrown when we attempt to read an image file that is of a different type than the one we are trying to read. The class has the following two constructors:

- public UnsupportedFileFormatException()

- public UnsupportedFileFormatException(String msg)

The ce326.hw2.PPMImage class

The PPMImage class inherits the RGBImage class and has the following constructors

- **public PPMImage(java.io.File file):**Creates an object of the receiving class as input the content of the file**file**. In the case that the file file does not exist or exists but cannot read it, it produces one**java.io.FileNotFoundException**, while in the case that the file file is not of type**PPM**then it produces one **ce326.hw2.UnsupportedFileFormatException**.
- **public PPMImage(RGBImage img):**He builds onePPMImage from an RGBImage.
- **public PPMImage(YUVImage img):**He builds onePPMImage from a YUVImage.

Note: You can assume that all the images you will read will have a maximum value brightness 255 .

The class additionally has the following**public**methods:

- **toString():**returns oneString containing the contents of the PPM file.
- **toFile(java.io.File file):**Writes the image to formatPPM inside the file archive. If the file already exists, it deletes the existing content.

The photo stacking process

The stacking process is applied to images that take several seconds to capture. They are usually night shots (landscapes, monuments or images of the night sky from telescopes). Prolonged exposure of an image to light introduces noise, which "blurs" the final result. In order to overcome the problem of introduced noise, the stacking technique is applied, which is summarized as follows:

Several images of the same subject are taken using the same extended exposure time. From these images, a unique image is obtained based on the following algorithm: each of the RGB values of any pixel of the final image is obtained from the average value of the RGB values of the corresponding pixel, of all the images involved.

Illustration: Assuming that thenoise follows a normal distribution , if we take a large enough sample of images the noise tends to disappear, as its mean value is 0.

The ce326.hw2.PPMImageStacker class

The class has the following constructor:

- **public PPMImageStacker(java.io.File dir):**It takes as input a file which must be a directory.
  - ○ If no file with this name exists it produces one **java.io.FileNotFoundException**with parameter the string "**[ERROR] Directory <dirname> does not exist!**",
  - ○ If a file with this name exists but is not a directory it produces one **java.io.FileNotFoundException**with parameter the string "**[ERROR] <dirname> is not a directory!**".
  - ○ If the specified directory exists in thefilesystem, but contains one or more files that are not PPM image files, throws an exception of the type **UnsupportedFileFormatException**.

Note: Is**mandatory**the pictures the class reads**PPMImageStacker**to are stored in a list type structure whose class implements the interface**java.util.List**. The list class can be ready-made (from the package**java.util**).

and the following methods:

- **public void stack()**: Applies the stacking method to the images it read.
- **public PPMImage getStackedImage()**: Returns the image resulting from the stacking process.

Download here the images for the image stacking process

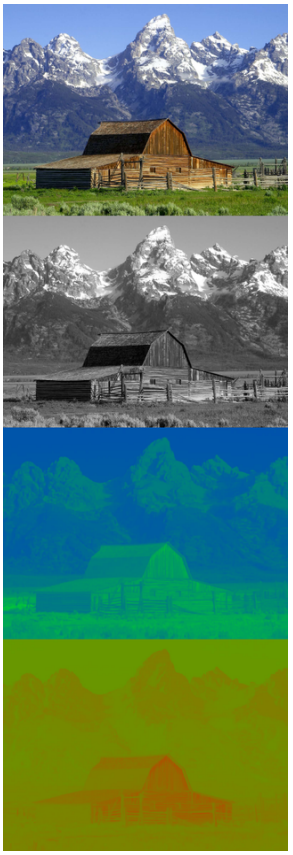
The YUV image

A picture**YUV** is an image of which eachpixel is described by the three integer values Y, U and V. The Y value represents the luminance value for the pixel and the U, V values represent color information.

In the adjacent image, an example of a color image is given and how it is distinguished in the parameters Y, U and V, which follow in order immediately after the top image.

Notice that the Y image (2nd image in the row) is black and white as it only describes the brightness of each pixel.

* image sourceWikipedia



Convert between RGB to YUV and vice versa

It is possible to convert a pixel from RGB to YUV and vice versa. The functions to convert an RGB pixel to YUV are given below:

Y = ((66 * R + 129 * G + 25 * B + 128) >> 8) + 16 U =
((-38 * R - 74 * G + 112 * B + 128) >> 8) + 128 V =
((112 * R - 94 * G - 18 * B + 128) >> 8) + 128

Accordingly, the functions to convert a YUV pixel to RGB are:

C = Y - 16
D = U - 128
E = V - 128

R = clip((298 * C + 409 * E + 128) >> 8)
G = clip((298 * C - 100 * D - 208 * E + 128) >> 8) B
= clip((298 * C + 516 * D + 128) >> 8)

The function**clip**does the following: if its input value is negative it does0, while if it is greater than 255 it makes it 255.

Note: During conversionYUV to RGB you can safely assume that the maximum brightness value of the RGB image is 255.

You can**download a spreadsheet here** which converts RGB values to YUV and vice versa.

The YUV file format

YUV files have an extension**.yuv**. The format of a YUV file is as follows:

1. It starts with the alphanumeric**YUV3**.
2. Followed by an alphanumeric representing an integer corresponding to the width of the image (in pixels).
3. Followed by an alphanumeric representing an integer corresponding to the height of the image (in pixels).

Points 1-4 make up the header of the file.

4. Then for each pixel 3 alphanumeric (integers) are displayed for the three Y, U, V values of the pixel. The file starts with the information of the top left pixel (we consider the first row of pixels, the top row), followed by the immediately rightmost pixel, until we reach the rightmost element of the top row. The file then continues from the leftmost element of the next row and so on.

The YUV format is defined in the context of the work and you cannot see it with the help of an image viewer.

Examples of YUV images can be found here.

The ce326.hw2.YUVPixel class

The YUVPixel class represents the values of a pixel in the YUV system. The information of the three YUV parameters for each pixel can be saved

- in oneprivate type field**int**.
- or in threeprivate type fields**short**.

The class has the following constructors:

- **public YUVPixel(short Y, short U, short V)**: Creates an object of the class YUVPixel based on Y, U, V values.
- **public YUVPixel(YUVPixel pixel):**Creates an object that is precise copy of the pixel.

- **public YUVPixel(RGBPixel pixel):**Creates an objectYUVPixel from an RGBPixel object.

and them**public**methods:

- **short getY():**returns the value of the parameterY.
- **short getU()**: returns the value of the U parameter.
- **short getV()**: returns the value of the V parameter.
- **void setY(short Y):**sets the value of the parameterY.
- **void setU(short U):**sets the value of the parameterU.
- **void setV(short V)**: sets the value of the V parameter.

The ce326.hw2.YUVImage class

The YUVImage class represents an image in the YUV system. Image pixels are composed of objects of the YUVPixel class. The class should have the following constructors:

1. **public YUVImage(int width, int height)**: Creates an object of type YUVImage with dimensions **width x height**. The values for the Y,U,V parameters are defined as follows:**Y=16, U=128, V=128**.
2. **public YUVImage(YUVImage copyImg)**: Creates an object of type YUVImage from another object of type YUVImage. The new image is a copy of the original (copy constructor).
3. **public YUVImage(RGBImage RGBImg)**: Creates an object of type YUVImage from an object of type RGBImage.
4. **public YUVImage(java.io.File file)**: Creates an object of type YUVImage whose information it reads from the file**file**. The file encoding is YUV.
5. In the event that
    a. the filefile does not exist produces one**java.io.FileNotFoundException**.
    b. the filefile is of no type**YUV**then it produces one **ce326.hw2.UnsupportedFileFormatException**.

In addition, it has the following**public**methods:

- **toString():**returns onejava.lang.String containing the contents of the file in YUV format.
- **toFile(java.io.File file):**Writes the image in YUV format into the file**file**. If the file already exists, it deletes the existing content.
- **equalize():**Balances the image using the balance algorithm histogram listed below.

Histogram Balancing

What is a histogram of an image

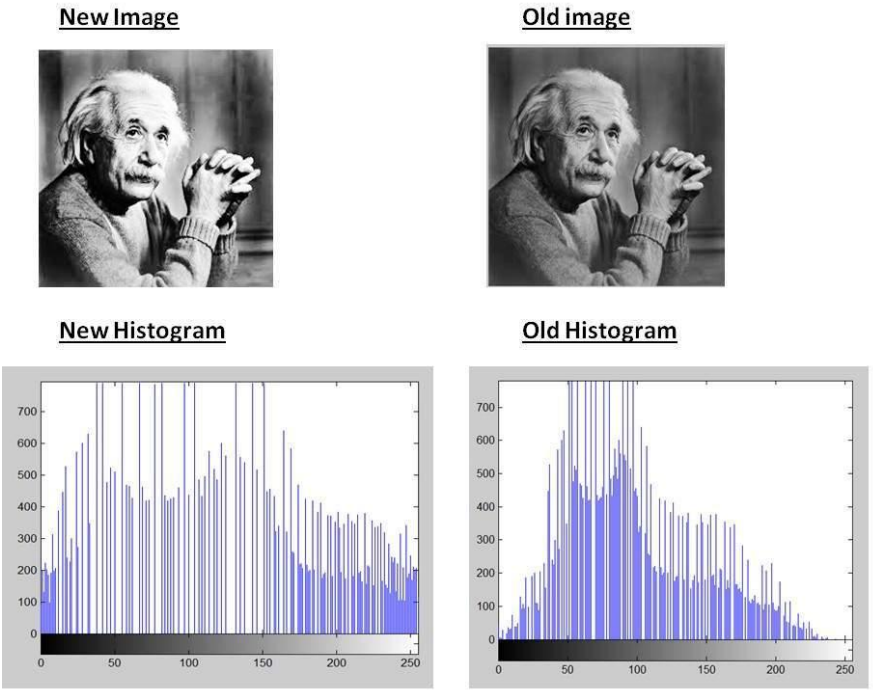
To understand the concept of a histogram, suppose you have a black and white image, that is, an image where the RGB values for each pixel are the same. The histogram is created if in an array of range 256 (from 0 to MAX_COLOR → 255), we place in the**f**the number of image pixels with brightness value**f**. The histogram is the distribution of the brightness of the image pixels.

Histogram of color images

In RGB-type images the histogram is difficult to define, as all three RGB values contribute to brightness. For this reason we prefer converting the image to YUV and calculating the histogram on the Y parameter which is the measure of the brightness of the image.

Histogram Balancing

Often an image appears dark and certain parts of it are hard to see. Image sharpness can be significantly improved if we increase the range of image brightness values. The figure below is indicative.



The process of balancing the histogram is as follows:

1. Based on the histogram, we calculate the probability distribution of the brightness of the image pixels.
2. Based on step 1 we calculate the cumulative probability distribution, i.e. the probability that a pixel has a brightness value less than or equal to the X value. The value is stored in the X position of the cumulative probability table.

3. We select the maximum brightness value that we want the new image to have. In the YUV system, the maximum brightness value is 235 (try in the spreadsheet to put RGB values 255, 255, 255 and see what the value of the Y parameter is). Select 235 as the maximum brightness value.
4. We multiply the matrix obtained from step 2 (cumulative probability distribution) by the maximum luminance value of step 3 and store the value in an array of integers thus cutting off the decimal part of the number obtained from the product.

5. From step 4, the proposed brightness change is obtained. Assume that the value Y is stored in position X of the array of integers obtained in step 4.
   Based on the above, all pixels with luminance X should be transformed to pixels with luminance Y. By transforming the luminance for all pixels in the image we get the final YUV image which is balanced in terms of its histogram.

Observation: Histogram balancing does not change the colors of the image, only their brightness.

Detailed examples of histogram balancing can be seenhere andhere .

The class ce326.hw2.Histogram

The ce326.hw2.Histogram class creates the histogram of a YUV image and has the appropriate methods for balancing this histogram. The class has the following constructor

public Histogram(YUVImage img): Creates the histogram of a YUV image.

and them**public**methods:

- **public String toString():**Prints the histogram in oneString as follows. Each brightness value occupies one line. The line contains the following:
  a. newline character, and the numeric value of the brightness three decimal places wide (no leading zeros), followed by a period.
  b. left parenthesis, the histogram value four decimal places wide, right parenthesis and a blank tab character.
  c. as many '#' characters as correspond to the thousands of pixels that have the specified brightness value.
  d. as many '$' characters as correspond to the hundreds of pixels that have the specified brightness value.
  e. as many '@' characters as correspond to the tens of pixels that have the specific brightness value.
  f. as many '*' characters as correspond to the units of pixels that have the specific brightness value.

  An extra newline character is printed at the end.

- **public void toFile(File file)**: prints the String of the toString() method to a file.
- **public void equalize():**The method balances the histogram.
- **public short getEqualizedLuminosity(int luminosity)**: Returns the new balanced luminance value corresponding to the original luminance value**luminosity**.

The equalize method of the YUVImage class

The method**equalize**of the classYUVImage creates a new balanced image relative to the original image containing the class object. The original image of the object is replaced by the new one.

Examples of equalized images

You can find examplesinitialsandfinalsimages .

The main program

The ce326.hw2.ImageProcessing class

The class is provided to you ready-made**ce326.hw2.ImageProcessing** which implements a suitable graphic environment through which the user has control over all the functions described above. After each selection, the user sees the new image that results after applying the change to the original image.

The application GUI menu is as follows:

1. File	1.1 Open	1.1.1. PPM File
		1.1.2. YUV File
		1.1.3. Other Format (reads PNG, JPG etc)
	1.2 Save	1.2.1. PPM File
		1.2.2. YUV File
2. Actions	2.1 Grayscale	
	2.2 Increase Size	
	2.3 Decrease Size	
	2.4 Rotate Clockwise	
	2.5 Equalize Histogram	
	2.6 Stacking Algorithm	2.6.2 Select directory

The ce326.hw2.PPMFileFilter and ce326.hw2.YUVFileFilter classes

They are given to you ready-made.

Shipping Instructions

The work will be sent through the autolab platform. There is a maximum limit of 35 free submissions you can make. After this number you are deducted one point for each additional submission.

To submit, follow these steps:

- Compress the contents of the directory**hw2**in which all the files with .java extension of the work, in zip format. The resulting file must have a name **hw2.zip.**

- You are connecting toautolab and select the course ECE326_2022 (S22) and from it the task HW2.
- To submit your work you doclick on the option "I affirm that I have compiled with this course academic integrity policy..." and press submit. Then select the hw2.zip file you created above.

They are provided to youthe input images and an accompanying file which describes the individual tests. In addition, you are also given the classImageProcessing2 which contains the main method of the program and is different from the one given to you (it does not have a graphical interface).