9/10/2018

Homework 1

Rule 1: no plagiarism (from colleagues or other sources). Detected cases of plagiarism will lead to a significant penalty of your course grade at the end of the semester.

Rule 2: You are asked to write some methods, you can use C/C++/Java/Matlab/Python. You can use readily available software libraries for loading, creating, displaying image files, as well as for getting/setting pixel values and reading image dimensions. Anything else you have to code it on your own!

Rule 3: no late submissions! Even if it is late by one minute, it will be ignored. Learning to plan your schedule according to deadlines is part of your education and an invaluable professional skill.

What to submit: a) the source code of your methods, b) a demo program testing **each** of them, and c) the output image files of your program in png format.

Task 1 (40 points): Code a method that admits as input a grayscale image **f**, two positive doubles **sx** and **sy** greater than 1.0, that represent the scaling factors of f and an angle theta in radians. The method should return a new image object containing f scaled horizontally by a factor of sx and vertically by a factor of sy using bilinear interpolation strategy and rotated counter-clockwise by an angle of theta with respect to the image's center (not with respect to the origin!)

Example: let's say `input` represents an image of width 100 pixels and height 50 pixels; in that case the call:

```
Image output = scale(input, 2.0, 3.0, 1.5707);
```

should return an Image object containing the contents of `input`, but scaled to a width of 200 and height of 150 pixels and rotated by approximately 90 degrees (use the image valve.png to test it).

Task 2 (30 points): Create a method that admits as input a grayscale image **f** and applies **adaptive histogram equalization** to it (use the image valve.png to test it).

```
Image output = equalize(input);
```

Task 3 (30 points): You are given the binary image abdomen.png. Code a method that first removes all unwanted artifacts and then counts the numbers of connected components in the foreground (you can use any method you like; bonus points if you implement the union-find based method).

In case you go with Java, just create a new java project and include the jars provided during week 1 as external libraries. In this library an image is represented through the class **vpt.Image**.

Loading an image from your drive:

```
Image img = Load.invoke("path/to/file/myImage.png"); // supports jpg, png
```

Saving an image to your drive:

```
Save.invoke(img, "path/to/save/filename.png");
```

Copy constructor (creates a black image "copy" of the same dimension as "img"):

```
Image copy = img.newInstance(false);
```

Create a new grayscale image of width w, and height h with **int valued pixels in [0,255]**

```
Image newImg = new ByteImage(w, h, 1);
```

Get the width and height of an image object

```
int width = img.getXDim();          int height = img.getYDim();
```

Display an Image object “img”

```
Display2D.invoke(img, “windowTitle”);
```

Reading the pixel intensity of image “img” at **row 300 and column 200 (assuming 0,0 as the origin)**:

```
int p = img.getXYByte(200, 300);
```

Setting the pixel intensity of image “img” at row 300 and column 200 to a new value 128.

```
img.setXYByte(200, 300, 128);        // the new pixel value is of type int
```

Good luck.