Task 2: Histogram [50 points]

You are given a folder (which may contain an arbitrarily deep subfolder tree) containing 24-bit .bmp RGB color image files. Each *filename.bmp* has a matching *filename.txt* (located in the same sub-folder), containing a text line with 4 integer numbers (`left top cropWidth cropHeight`) which describe a cropping window in the corresponding image file. Your task is to compute a grayscale histogram for the cropped part of every image in the folder (no matter where the image is in the sub-folder tree), and aggregate (by summing up all values corresponding to each particular color) those histograms into one, which should be written as the program's output.

Note: When dealing with RGB color images, use the formula `grayscale = 0.2126 * r + 0.7152 * g + 0.0722 * b` for grayscale conversion. When converting to an 8-bit number (representing the color's grayscale value), round up to the nearest integer. If using any "low-level" methods for image loading and/or pixel access, keep in mind that the "default" layout for storing 24-bit RGB images is B0G0R0B1G1R1B2G2R2...

Input

`inputFolderPath outputFolderPath` (provided as command line parameters in that order). The input folder (and its subfolders if any) will contain only .bmp images and their corresponding .txt crop window descriptions. The cropping should include all pixels with coordinates (`left <= x < left + cropWidth` and `top <= y < top + cropHeight`). It is guaranteed that the crop window will be completely contained within the corresponding image.

Output

Aggregated grayscale histogram of all (cropped) images in the folder tree. It should be saved as a text file containing 256 comma-separated values, where *i*-th value should be total number of pixels having greyscale color *i* (*i*=0..255). The output should be saved as `outputFolderPath\GlobalHisto.csv`. **The output folder should be created if it doesn't exist.**

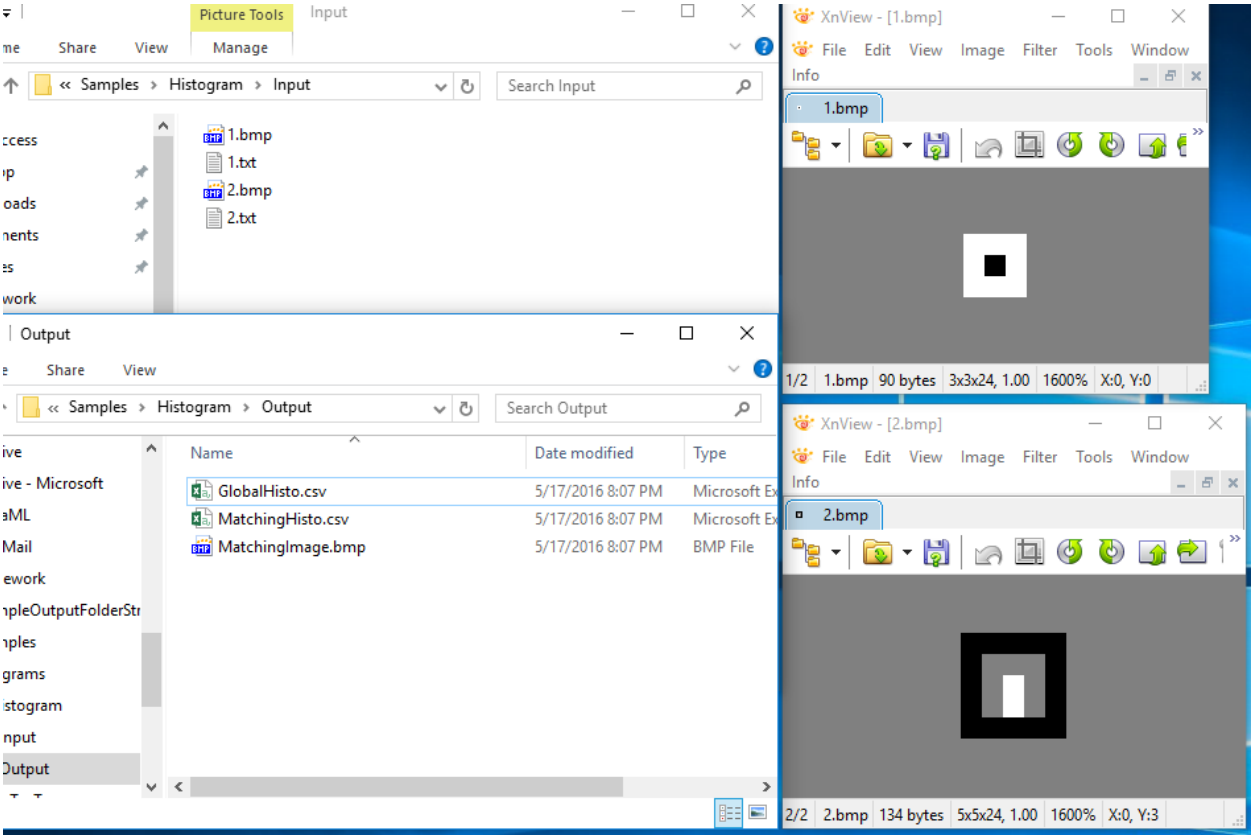
Bonus task

Using the histogram computed in the previous step, generate an arbitrary 100x100 pixel .bmp such that its own grayscale histogram matches the previously computed one in shape (i.e. it generally won't be possible to match the aggregated histogram in absolute pixel counts, but it should be possible to match its distribution). The image should be saved as `outputFolderPath\MatchingImage.bmp`, and its histogram as `outputFolderPath\MatchingHisto.csv`

Samples

In the Samples folder, you will find a sample of the expected input\output. In `Samples\Histogram\Input` you will find one 3x3 image and one 5x5 image. Cropping rectangle corresponding to the first image will start at (and will include) pixel (1,1) and be 1 pixel wide and 1 pixel high (therefore, containing only pixel at (1, 1) which is completely black). Similarly, the other cropping rectangle will contain two white and two gray pixels. Therefore, the resulting aggregate histogram will be filled with zeros, except for values 1, 2 and 2 corresponding to colors 0, 128 and 255 respectively. You can see that histogram in `Samples\Histogram\Output\GlobalHisto.csv`. The `Samples\Histogram\Output\MatchingImage.bmp` is a 100x100 bitmap containing 2000 (thus,

20%) black pixels, 4000 gray and 4000 white pixels. Therefore, its histogram `Samples\Histogram\Output\MatchingHisto.csv` is a scaled version of `Samples\Histogram\Output\GlobalHisto.csv`



Calling

In case you're producing Windows .exe, your program will be called like:

```
Histogram.exe inputFolderPath outputFolderPath
```

In case of Python script

```
Python histogram.py inputFolderPath outputFolderPath
```

In case you write an Octave function, you should have a `Histogram.m` file containing `function Histogram (inputFolder, outputFolder)` (which does the processing and saves the results into the output folder as described).

In case of Java:

```
java Histogram inputFolderPath outputFolderPath
```

or

```
java -jar Histogram.jar inputFolderPath outputFolderPath
```

Input image dimensions will not exceed 1000px in height or width. Time limit is 5 seconds per input image on a 2GHz machine.

Document update history

5/20/2016 7:05 PM: Document created