Image Histogram

IMGS-180: Project 7

1 Assignment

In this project, you will be creating functions to calculate the histogram, probability density function (PDF) and cumulative density function (CDF) of the individual channels of an image. Your program will be producing three plots, one for each of these results. For RGB imagery, the color of the plot for each channel should be appropriately colored (red for R, green for G, blue for B). If the image is greyscale (single channel), the information should be displayed in black. The functions to be written are described below.

```
/** Calculates the histogram of an image using brute force calculation
 *
 *  \param[in] image The image to analyze, may either be CV_8UC1 or CV_8UC3
 *  \param[out] histogram Output histogram, will be 1 x 256 for grayscale images
 *                        and 3 x 256 for color images
 *
 *  \return Whether histogram calculation was successful
 */
bool BruteForceHistogram(const cv::Mat& image, cv::Mat& histogram);
```

```
/** Calculates the histogram of an image using OpenCV's function
 *
 *  \param[in] image The image to analyze, may either be CV_8UC1 or CV_8UC3
 *  \param[out] histogram Output histogram, will be 1 x 256 for grayscale images
 *                        and 3 x 256 for color images
 *
 *  \return Whether histogram calculation was successful
 */
bool OpenCVHistogram(const cv::Mat& image, cv::Mat& histogram);
```

```
/** Calculates a probability density function (PDF) from a histogram
 *
 *  \param[in] hist The image histogram to convert. Normalization will occur
 *                  along the rows of the array.
 *  \param[out] pdf Output probability density function
 */
void HistogramToPdf(const cv::Mat& hist, cv::Mat& pdf);
```

```
/** Calculates a cumulative density function (PDF) from a probability
 *  density function (PDF)
 *
```

```
 *  \param[in] pdf The PDF to convert. It is assumed that PDFs are along
 *                 the rows of the array.
 *  \param[out] cdf Output cumulative density function
 */
void PdfToCdf(const cv::Mat& pdf, cv::Mat& cdf);
```

For the given input file "lenna.tif", the output should look similar to:

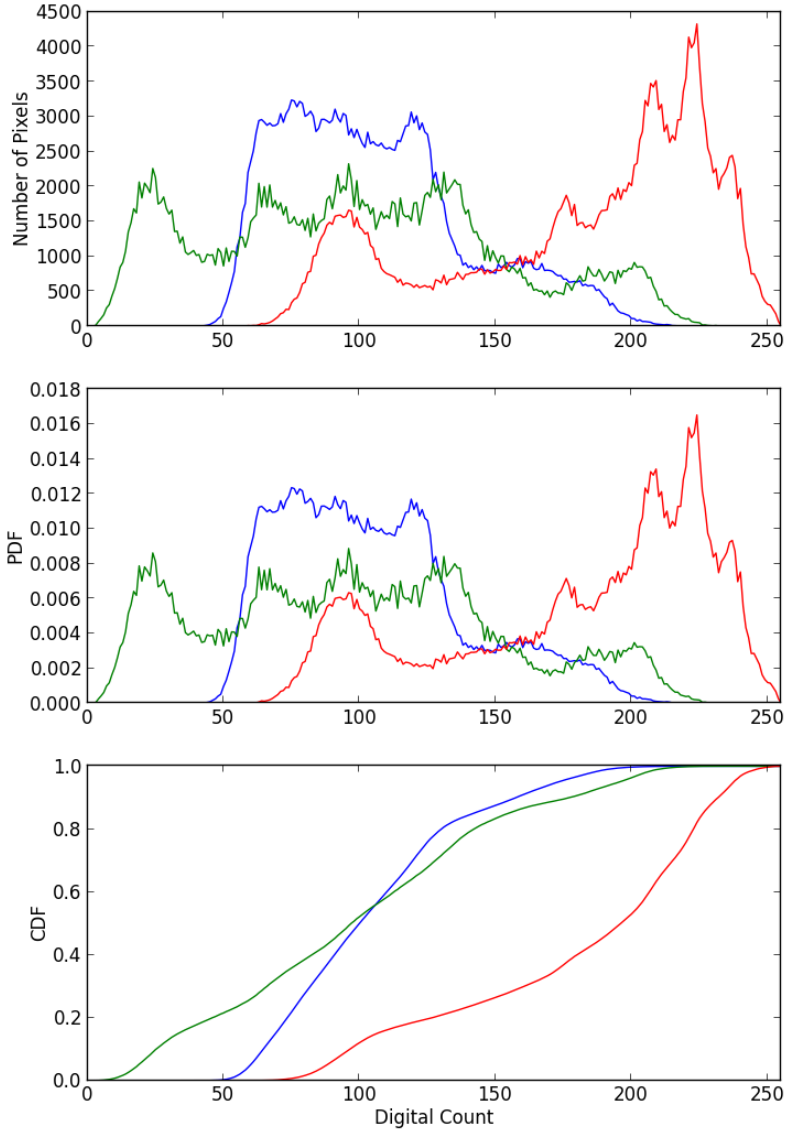

Figure 1: Sample output plots for lenna.tif

For the given input file "crowd.jpg", the output should look similar to:

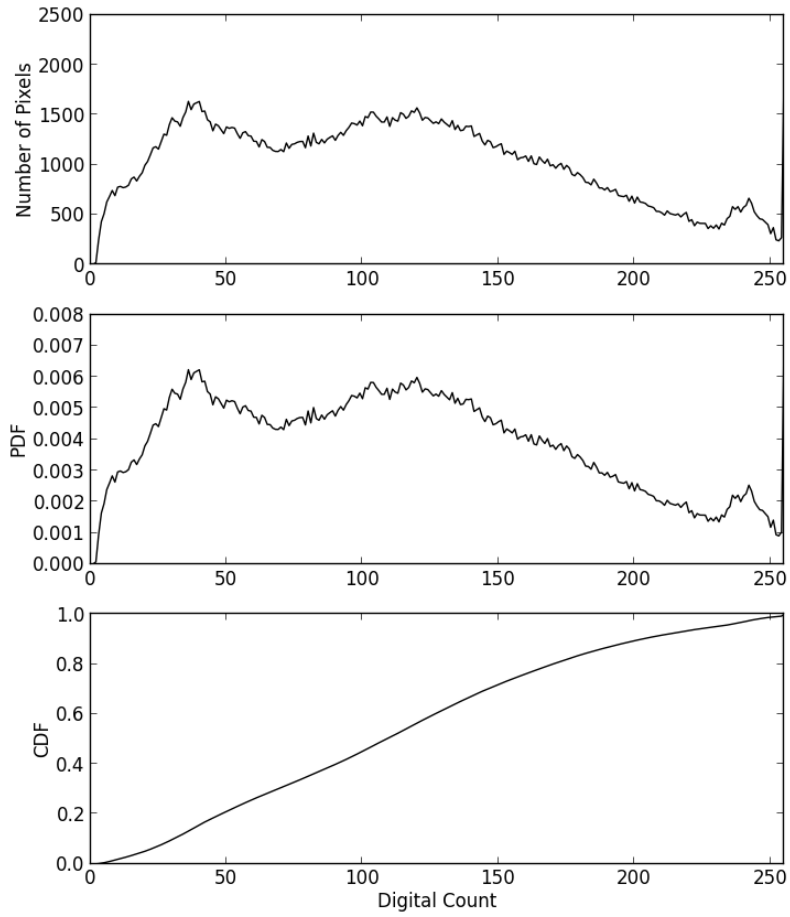

Figure 2: Sample output plots for crowd.jpg

For plotting, you may either use the plotting approach from Project 5 and 6 of piping your output to gnuplot, such as

```
$ ./calculate_histogram lenna.tif OpenCV | gnuplot
```

or you can use the gnuplot-iostream library, which allows you to interface with gnuplot using streams, similar to cout. The documentation for that library may be found at http://stahlke.org/dan/gnuplot-iostream/. If you choose to use gnuplot-iostream, I would recommend either outputting all your data to the stream, as shown in the examples in Project 6, or following Example 2 in their documentation, using `std::vector<pair<double, double>>`.

In addition to the functions listed, you will also be providing the main function for this program. The following command line interface should be used

```
$ ./calculate_histogram filename method
```

where filename is the filename of the image to analyze and method is either "BF" for brute force calculation or "OpenCV" for OpenCV calculation.

2 What to Submit

For this project, you should upload a zip file to myCourses that contains the following:

1. All the code (any .cpp, .h or CMake files)
2. A README file containing instructions on how to build and run your program and generate your plots
3. All of your plots (EPS or PDF files, which gnuplot will produce). Depending on how you generate them, there should be 4 or 12 (2 images, 2 methods, 1 or 3 plots per run).

3 Grading

The grading for this project will be:

- 70% functionality - For producing the requested results
- 30% style - For clear, well written code (good variable names, good formatting, etc.)