

Image Processing

By Riham Muneer

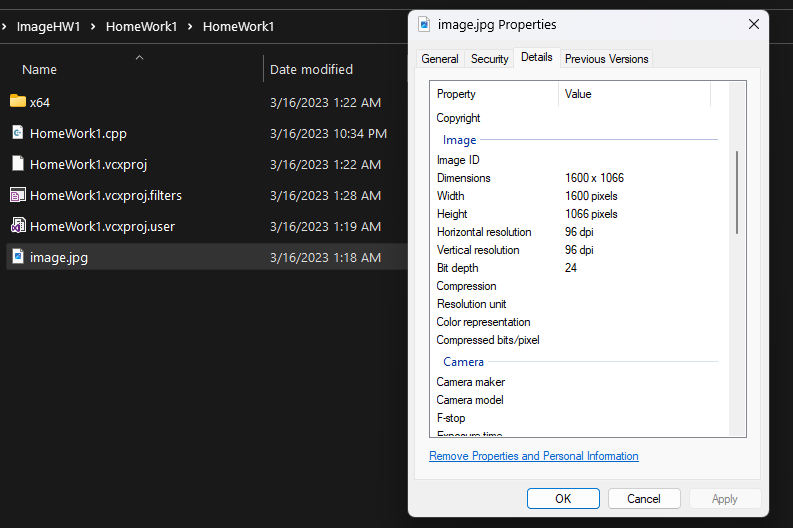
Homework1 - report

Dr. Anas Toma

|  |
| --- |
| The color input image |



The original size of this image

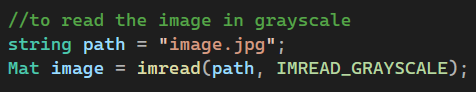


|  |
| --- |
| The gray-scale image |

* Transfer the BRG image into g ray-scale image

We can do that in two ways:

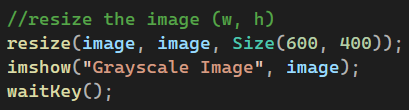
* Read it as a grayscale image directly using “imread” method (which I used).



* Read it as it is using “imread” method then transfer it into gray-scale using

cv :: cvtColor(source, destination, conversion code) method.

* After that, I resized it into (600 X 400) pixels, and show it using “imshow” method.



* The result

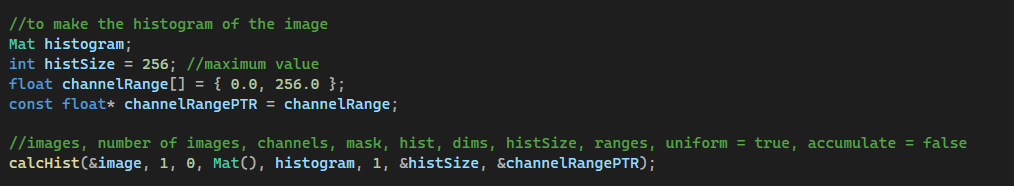


|  |
| --- |
| Original Gray-scale image’s histogram |

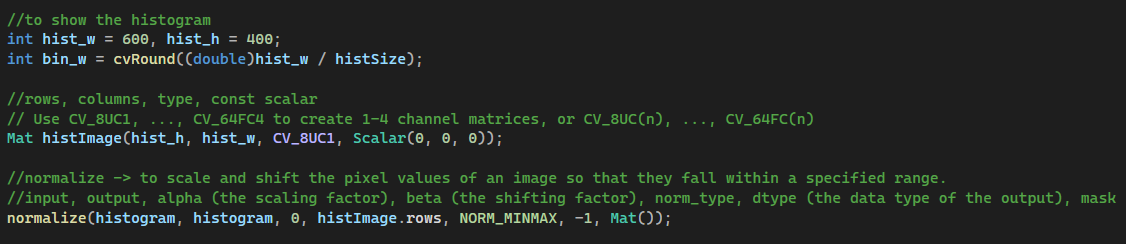
**Histogram:** is a representation of frequency distribution of pixel intensities in the image.

* Use calcHist (images, #images, #channels, mask, histogram, dimensions, histogram size, ranges)

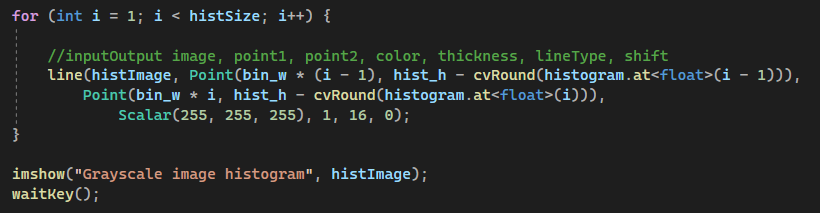
to calculate the histogram of the gray-scale image.

****

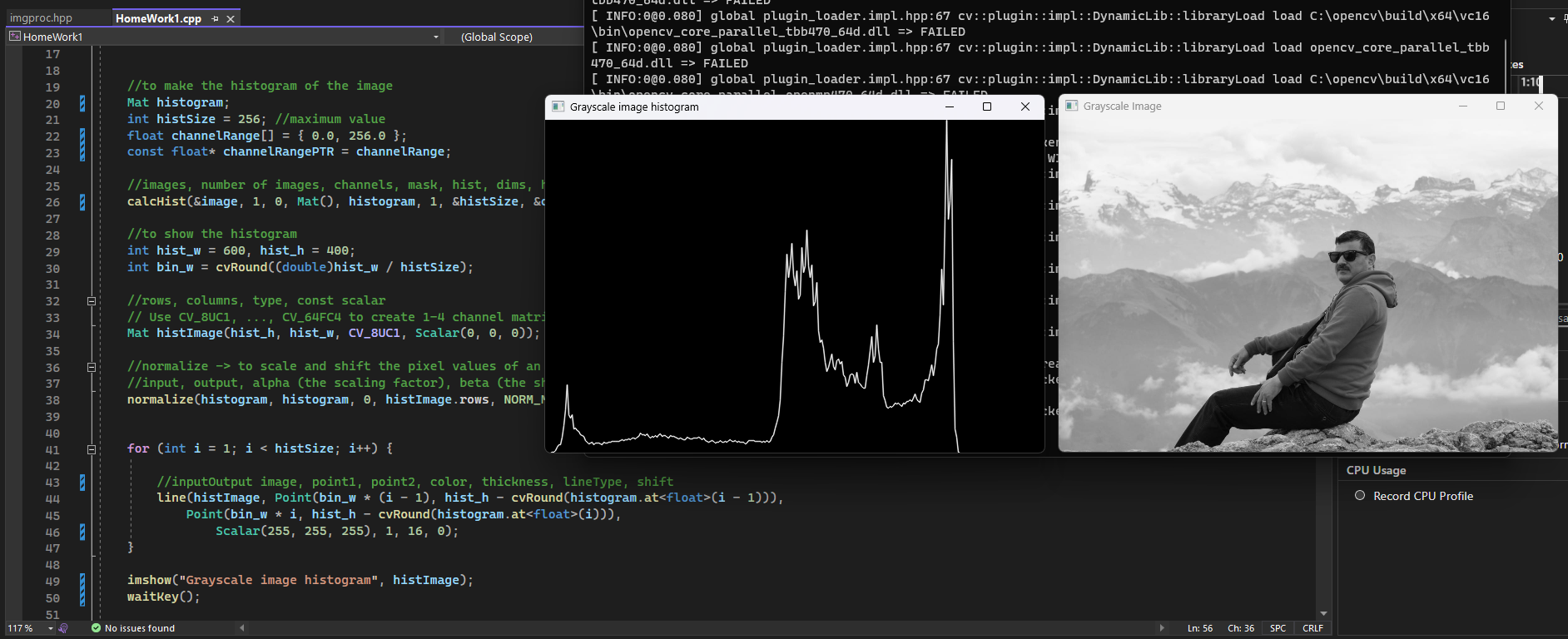
* **&image**: The gray-scale image
* **1:** The number of images
* **0:** The number of channels, (gray-scale image need only a single channel)
* **Mat():** The mask to apply to the image (in our case we don’t need a mask)
* **histogram:** The resultant histogram
* **1:** 1D histogram
* **&histSize:** The number of bins on the x-axis
* **&channelRangePTR:** The range of these bins
* Use normalize(input, output, alpha-the scaling factor, beta-the shifting factor, normalization type constant, desired output data type, mask) to normalize the histogram.

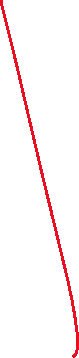


* **histogram:** the input histogram
* **histogram:** the output histogram
* **0:** the scaling factor
* **histImage.rows:** the shifting factor
* **NORM\_MINMAX:** the type of the normalization
* **-1:** it is negative so the output will have the same type as the input.
* **Mat():** The mask
* Draw the histogram using line(inputOutput image, point1, point2, color, thickness, lineType, shift) method and show it



* + - histImage: inputOutput image
    - Point(x, y)
    - Scalar(255, 255, 255): the color – white
    - 1: the thickness of the line
    - 16: the type of the line it can be -1, 4, 8, 16
    - 0: shift
* The result





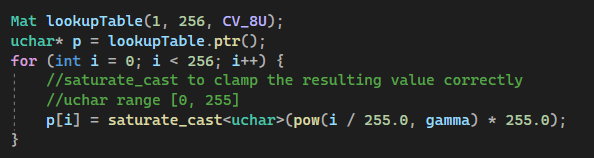
|  |
| --- |
| modify the brightness |

**Modifying the brightness with gamma:**

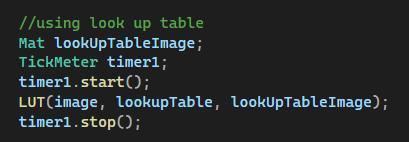
* Generate a random gamma value

****

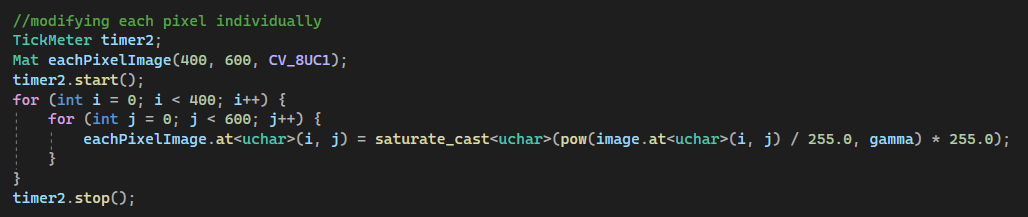
* Prepare a look up table with a palette of 1-256

****

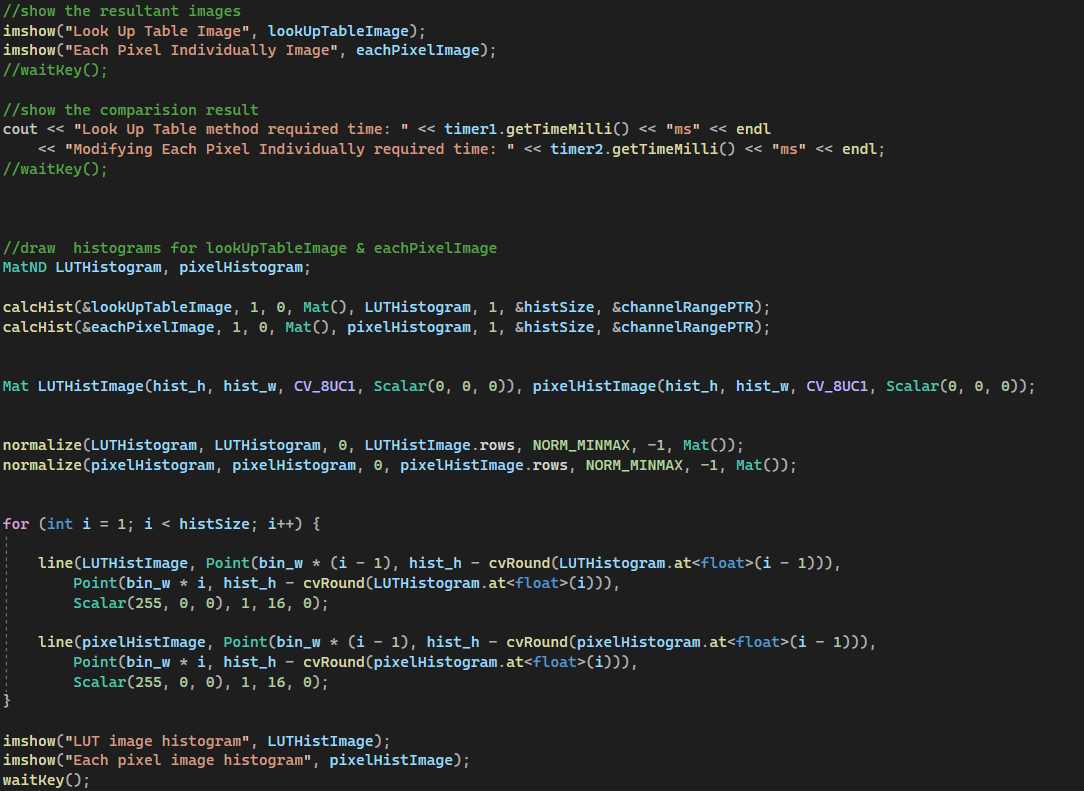
* using “uchar” because it has a range {0, 255}
* using saturate\_cast method to clamp the resultant value correctly
* we divide I by 255 then multiply it by 255 to scale it correctly
* Use the look up table with LUT(input image1, imput image2, output image) method and calculate the execution time of the process using a timer

****

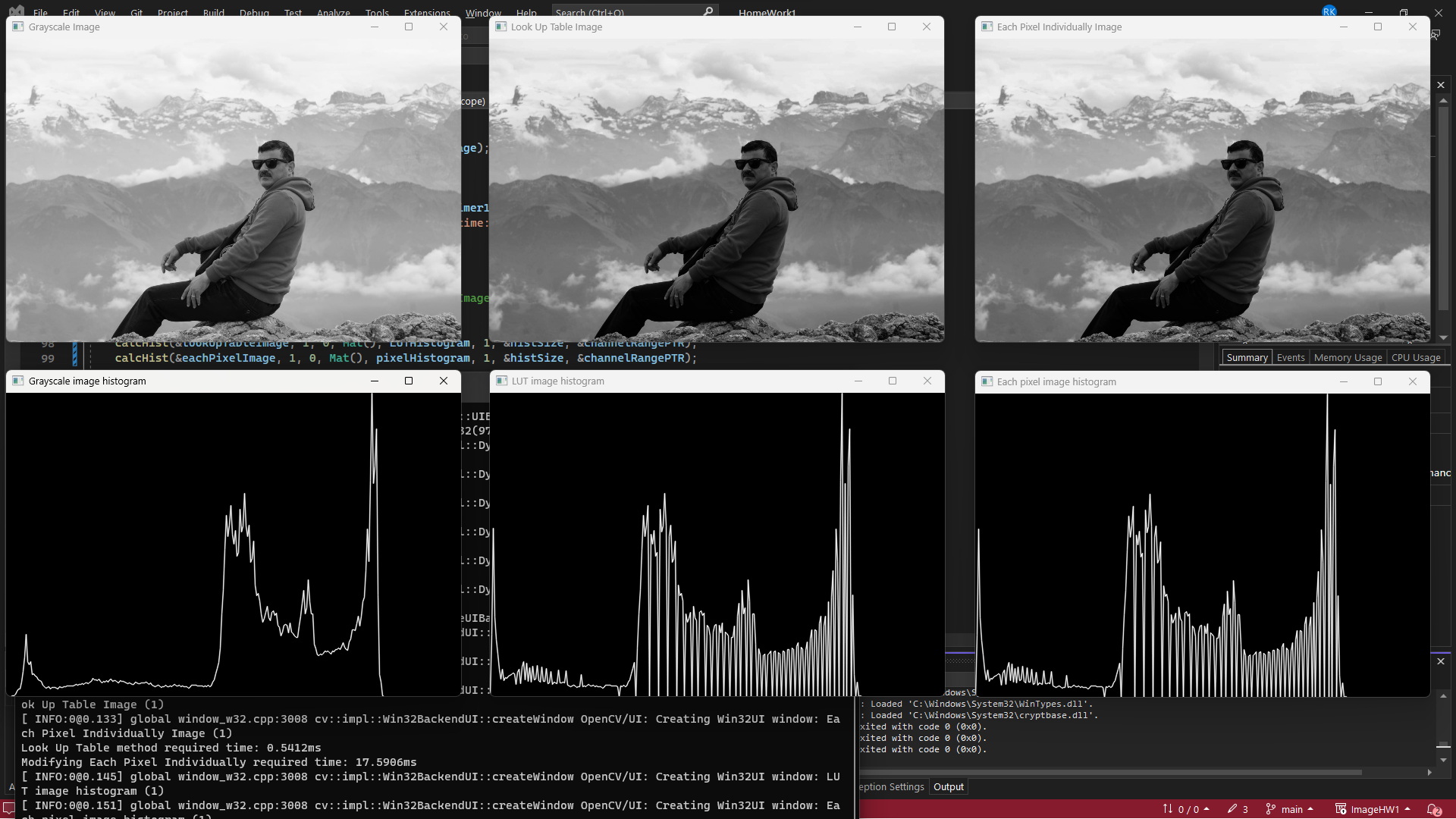
* Modify the brightness for each pixel individually and calculate the execution time of the process using another timer

****

* Show the difference between the execution time of the two methods and generate the histograms.

****

|  |
| --- |
| The final result |

****