

---

## COMPUTER VISION - LAB 3

---

**Topics:** Image Equalization, Histograms, Filters, Morphological Operators

### *Part 1: Histogram Equalization*

Write a program that:

1. Loads an image (e.g., one of the provided images like “image.jpg” or “countryside.jpg”)
2. Prints the histograms of the image. You have to compute 3 histograms, one for each channel (i.e., R, G and B) with 256 bins and [0, 255] as range. Notice that you need to use the `calcHist()` function separately on the 3 channels. You can use the provided function `showHistogram()` to visualize the data.
3. Equalizes the R,G and B channels by using `cv::equalizeHist()`
4. Shows the equalized image and the histogram of its channels.
5. Is it possible to obtain a better equalization than the one obtained in point 4? Try to perform some experiments using a different color space, e.g. the HSV color space (use the functions `cv::cvtColor()` with the flags `cv::COLOR_BGR2HSV` or `cv::COLOR_HSV2BGR` in your color space conversion code), and equalize only one channel (e.g., see Figure 1). Which one would be better to choose?

**Note:** The HSV color space is very similar to the HSI color space presented in class. As for the HSI color space, H and S stand for Hue and Saturation, respectively, while V stand for Value, that actually is a value of the lightness of the pixel. Roughly speaking, you can think of V as the grey level of the pixel.

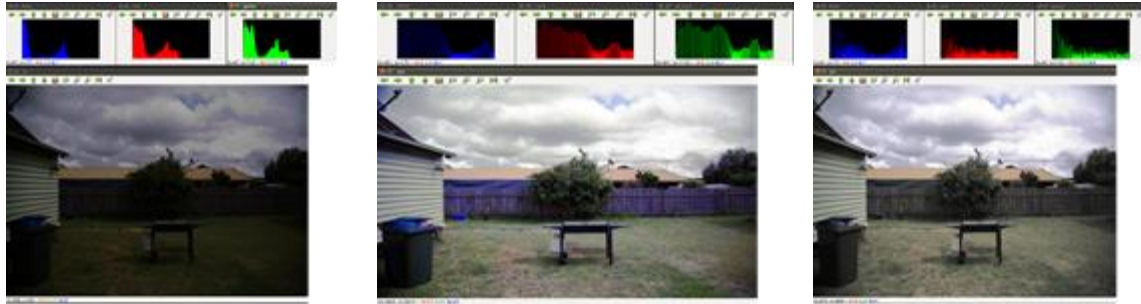


Figure 1: Example of the results of the first part

## Part 2: Image Filtering

Generate a denoised version of the image computed in point 5. You should try different filters and parameter values.

Write a program that shows the result with some trackbars to vary the parameters of each filter (see the example in Figure 2). Table 1 specifies the requested filters to test and the parameters to be controlled with the trackbars.

- In order to generate the trackbars you can use the `cv::createTrackbar()` function.
- In order to pass the image and the parameters to the callback of the trackbar create a class containing the image and the filter parameters
  - See for example the provided class `Filter` and extend and/or modify it creating subclasses for the various filters.

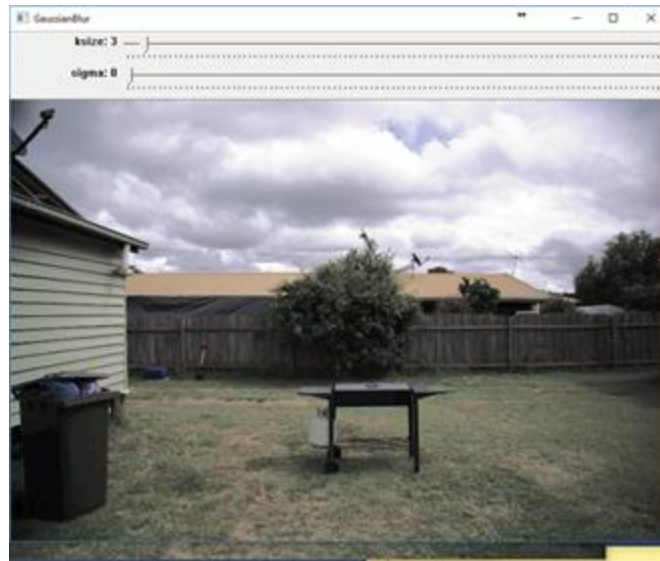


Figure 2: Example of the window with trackbars (2<sup>nd</sup> part)

<code>cv::medianFilter()</code>	<ul style="list-style-type: none"> <li>• <code>kernel_size</code></li> </ul>
<code>cv::GaussianBlur()</code>	<ul style="list-style-type: none"> <li>• <code>kernel_size</code> (keep it square)</li> <li>• <code>sigma_X (=sigma_Y)</code></li> </ul>
<code>cv::bilateralFilter()</code>	<ul style="list-style-type: none"> <li>• <code>kernel_size</code> (trackbar not required, use a fixed value or use the <math>6\sigma_s</math> rule)</li> <li>• <code>sigma_range</code></li> <li>• <code>sigma_space</code></li> </ul>

Table 1: Filters to be implemented and their parameters

### Part 3: Morphological Operators (optional, not required for the homework)

By using one or more morphological operators (in particular the erode and dilate ones, that correspond to min/max filters), remove the electric cables and the handle of the barbecue from the `image.png` image without damaging too much the rest of the image. Recall that the max/min filters correspond to the morphological operators on greyscale images with a square window as structuring element. In OpenCV you can use the dilate/erode morphological filters.

The functions to be used are `cv::erode()`, `cv::dilate()` and `cv::morphologyEx()` with `cv::MOP_OPEN` and `cv::MOP_CLOSE` as operators (in this case, you may try different structuring elements with the function `cv::getStructuringElement()` ).

Which is the filter that provides the best result?

**Reference:**

<http://homepages.inf.ed.ac.uk/rbf/HIPR2/morops.htm>

<https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm>

<https://www.sciencedirect.com/topics/engineering/morphological-operator>