Università della Svizzera italiana

Faculty of Informatics

Image and Video Processing (Spring 2023)

Assignment 1: Point Operations

March 17, 2023

1 Point Operations [20 points]

Point operations are the simplest kind of image processing transforms. They modify pixel values independently of their location within the image or neighboring values. They are used for simple contrast and brightness enhancements, color adjustments, matting and compression, among other things. They are also notably used to adapt the dynamic range of images to the screen that is used to display them, a process known as *tone mapping*. In this assignment, you will code and apply some of the most common point operators, as well as answer several theoretical questions about them. Optionally, you will be able to get creative with your own pictures and play around with the transformations.

In this assignment, we will be working with color images. Remember to use matrix operations whenever possible to obtain efficient code and take full advantage of MATLAB.

1.1 Tone mapping & Linearization [2 points]

Tone mapping is usually the last step of the digital processing pipeline, and its most basic implementation is through point operators (gamma correction). RAW images usually have a much larger dynamic range than the capabilities of most displaying devices, so we use tone mapping to adapt pixel intensities to the limits of the device that will display them. Tone mapping algorithms are designed to adjust the contrast and brightness of HDR images so that detail in highlights and shadows is preserved when displayed in LDR.

Display devices like CRTs, LCDs or OLED devices cannot, or are not designed to, achieve a linear conversion from voltage or intensity to light emitted. In practice, this means that doubling the value of a pixel would not result in double the light being emitted. In order to solve this issue, each pixel value is mapped to an exponential function of the type $f(x) = x^{\gamma}$, with an encoding gamma of typical value $\frac{1}{2.2}$. This type of tone mapping is called gamma correction.

The final product of any digital camera pipeline is a compressed non-linear image, normally in JPEG format, where tone mapping has already been applied. However, this means intensities have been mapped with an exponential function. Using linear color for image processing is important because it ensures that the mathematical operations performed on the image data are consistent with the physics of light and the way that images are captured and displayed, as light transport is indeed linear.

Implement a basic MATLAB code to load the *Ferrari* JPEG image into a matrix and invert the function above to linearize the image back. Assume that the value used for encoding the image was $\gamma = \frac{1}{2.2}$. You can use the off-the-shelf MATLAB function *imread()* to load your images. Try applying other per-pixel functions to increase brightness (linear multiplications) and contrast enhancement (exponential functions) and showcase your creations.

1.2 Color correction [4 points]

Pixel transformations can be used for correcting colors in images by applying different transformations separately to red, green, and blue channels. Implement a simple white balance method to correct color in an input image such that the objects that should be gray appear gray in the image. Since light is a multiplicative

factor for the recorded intensities of the image, for this exercise, consider defining linear pixel transformation where you modify *gain* depending on the channel and leave the bias equal to zero. In order words, your task is to define per-channel gain, which results in color-corrected images. We ask you to find the gain using two strategies:

1. **Pixel-based correction:** Manually choose a pixel in the input image, which should be gray. Compute per-channel gain such that all components of this pixel (red, green, blue) are the same values and are equal to the average red, green, and blue values in the input image. You may use *ginput()* to pick pixels from an open figure.

2. **Gray-world assumption:** Implement a similar method, but instead of choosing one pixel, consider an average color in the image. This method uses so-called *gray-world assumptions*, which states that on average, the world is gray.

We provide an input image, *white_balance_input.jpg*. The examples of the results are demonstrated below. Note that your results may be different. The difference may come, for example, from choosing a slightly different pixel for color correction. When choosing the pixel try to choose one that is gray, not too bright, and not too dark. For the report, please experiment with doing the transformations directly on the values of the input

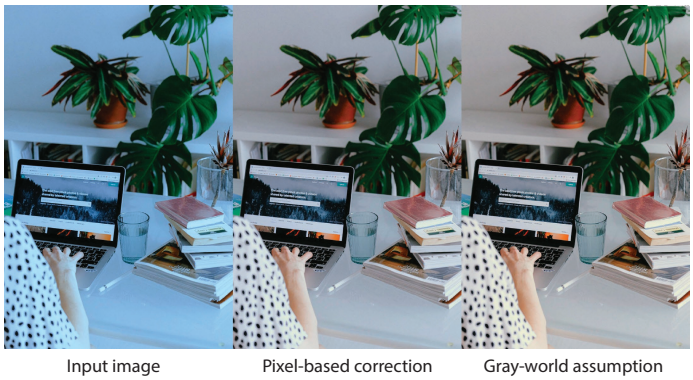

Figure 1: Expected results from the different white balance methods to implement

image and the intensities, i.e., after applying inverse gamma correction (assume $\gamma = 2.2$). Which correction should be more correct?

1.3 Histograms [1 point]

Histograms show the frequency of occurrence of different pixel intensity levels in the image. We can use them to obtain exposure information from an image (assessing whether an image has captured a sufficient amount of light). Color histograms can also be used to manipulate images. For instance, if an image has too much blue, you can adjust the histogram to reduce the amount of blue, by shifting the blue values towards the center of the histogram, which will reduce its intensity in the image.
Implement a histogram plotter using basic MATLAB operations and show the color histogram of the provided image. You will need to obtain the cumulative distribution of pixel values for each color channel and map them into a certain number of bins to display them.

1.4 Histogram Equalization [5 points + 2 Bonus]

Another way of using histograms to manipulate images is through histogram equalization. By redistributing pixel intensities across the image based on its histogram, we can improve its contrast. Answer the following question:

- What is the effect histogram equalization has on compressed images?
- How will the histogram of our image look like after equalization?

There are several ways of performing histogram equalization:

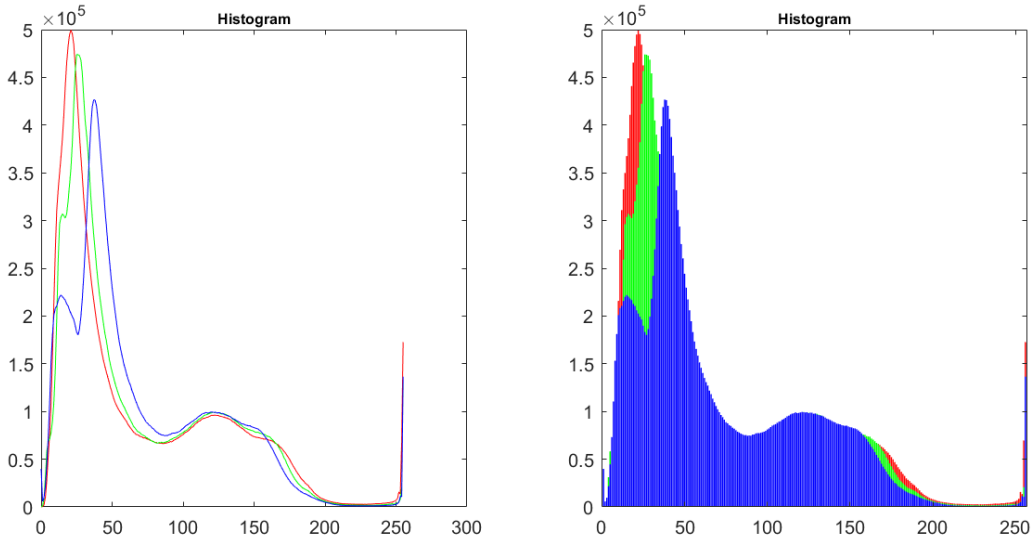

Figure 2: You should get something like this when computing the histogram of the *Ferrari* picture

- **Global** methods consist on calculating the cumulative distribution function of the histogram, normalizing it and then mapping each pixel in the image to their new normalized intensity.
- **Local** methods on the other hand, work on pre-defined windows across the image (think of cropped squares of a fixed size), where a histogram is computed for each window and normalized using the previous method independently.
- **Locally adaptive** methods use a moving window across the image instead of fixed-place boxes, which reduces the artifacts that the purely local method creates. Each pixel value is then equalized according to the histogram of its small surrounding window, repeating the task for every pixel in the image.

Implement global and local histogram equalization methods in MATLAB using basic functions and show the results of applying each to the same image. You can get bonus points for implementing a locally adaptive method based on the previous two. For local methods, you should try with different window sizes.
Interpret these results and argue benefits and problems of each. One thing to note is that running any kind of histogram equalization on each color channel separately will probably alter the colors in the image. Instead, you may use a stock MATLAB function to convert the image to HSV color space and work only on the value axis. This will only alter brightness in the image and conserve color.

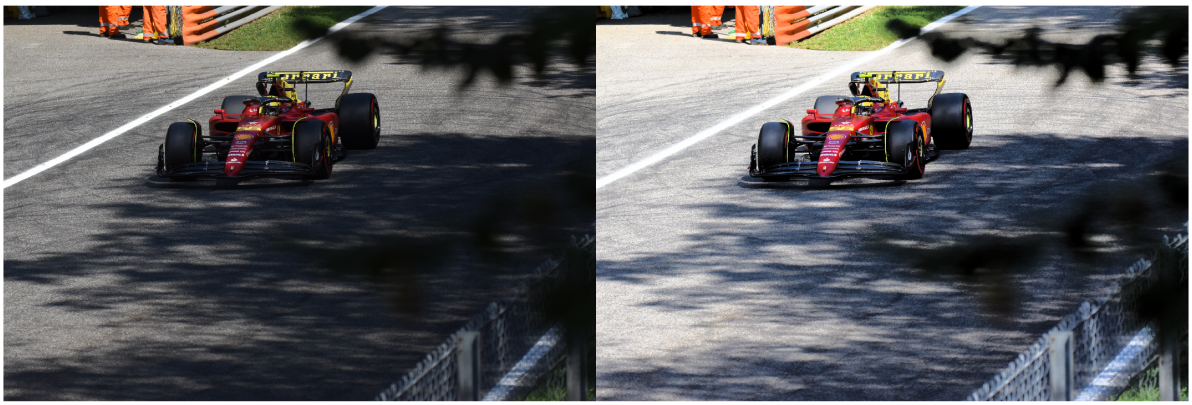

Figure 3: Original (left) vs histogram-equalized image (right). Your results will vary depending on the method, but you should get an overall flatter image with boosted shadows.

1.5 Manual histogram equalization [2 points]

Consider a histogram of an image shown below. Without any calculation, draw a function that corresponds to the approximate pixel transformation that would be applied using standard histogram equalization.

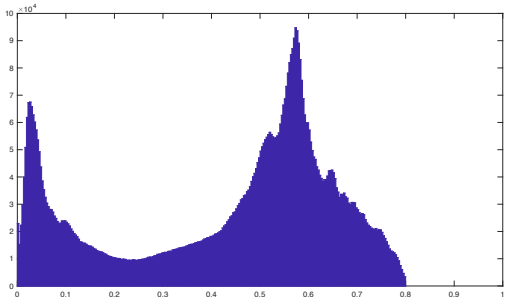

Figure 4: Histogram for Exercise 1.5

1.6 Thresholding & matting [4 points]

Another basic operation in image processing is thresholding and matting. Image pixels are filtered using a threshold which results in a mask or matte. This mask is used to isolate a specific object in the image, so that we can either place it in another image or simply change its background. Although recently, semantic classification of pixels through machine learning or deep learning methods have obsoleted classic techniques such as chroma keys (selecting matte masks according to color backgrounds), it is still a powerful and cheap tool for matting.

Implement a function to filter (threshold) the provided image based on the color background to obtain a mask, which will then be used to replace this background with a different image of your choice.

1.7 How many bits is enough [2 points]

Assume you want to represent an image of a smooth gradient spanning the entire luminance range of your display, which is from 0.5 $\mathrm{cd/m^2}$ to 1000 $\mathrm{cd/m^2}$, without seeing any contouring artifacts. How many bits per pixel do you need to store the grayscale version of the gradient image? In your computations, assume that the Weber law holds and the Weber ratio k equals 0.01. Note this is a bit more conservative assumption than what we used in calculations in the class. Provide your computation.

Figure 5: Original (left) vs thresholded and background-replaced image (right). Your results will vary depending on the method and the range of colors that you set as chroma key. *cat* image sourced from Freepik

2 Bonus: use your own pictures [3 points]

Take your own pictures and process them through your transformations, enhancing them in artistic ways. If you have a tripod or any way to maintain your phone in the same position through several takes, try matting yourself several times into the same take, or changing the background of a picture of yourself. You may or may not use a chroma (color matting background).

Submission

You should submit one ZIP-file via iCorsi containing:

- All your code in MATLAB appropriately commented, the processed pictures that you obtained and your own pictures if you did the Bonus, so that we can reproduce your results.
- A complete PDF report detailing your solution, partial results for each exercise and answers to the theoretical questions poised (Maximum 5 pages + 1 if you choose to do the Bonus).

Grading will be mostly based on the provided PDF report so we encourage clarity and detailed answers. We recommend using LaTeX or Overleaf to write the report. Usage of ChatGPT or any other natural language model is strictly prohibited and will be severely punished.

Solutions must be returned on March 31, 2023 via iCorsi3