Image Processing: Assignment #1

Submission date: 28\01\2024, 07:59.
Please submit the assignment via Moodle.
For questions regarding the assignment please contact:
Alon Papini (imageprocessinghaifau@gmail.com).

Assignment Instructions:

- Submissions in ***pairs***.
- The code must be written in Python 3.11 or higher.
- The code must be reasonably ***documented***.
- Please submit one single ***.zip*** whose name should be using this format: ID1_ID2_HW1.zip. It should contain ***only***:
  - The python script file where all your code is (you are given TranscribeHistogram.py to work with, so write your code there)
  - The PDF file where your answers will be.

Assignment Topics:

- Sampling
- Wavelength and frequency
- Nyquist theorem
- Quantization
- Image pixel histograms

Good Luck 😊

Problem 1 – Sampling, Quantization (14 points):

The following questions don't have one specific correct answer, so don't worry too much about getting the right answer – what's important is what you learn while trying to answer them.

We'll start with some basic questions to help build some intuition regarding images.

When thinking of image sampling and quantization, we might think about a camera taking a picture of something, or of a computer rendering/generating an image with certain parameters, etc. We call the smallest sampling unit of an image a "pixel".

a) As you may know, cameras take pictures with a certain number of "megapixels", and computers render images in a certain resolution.

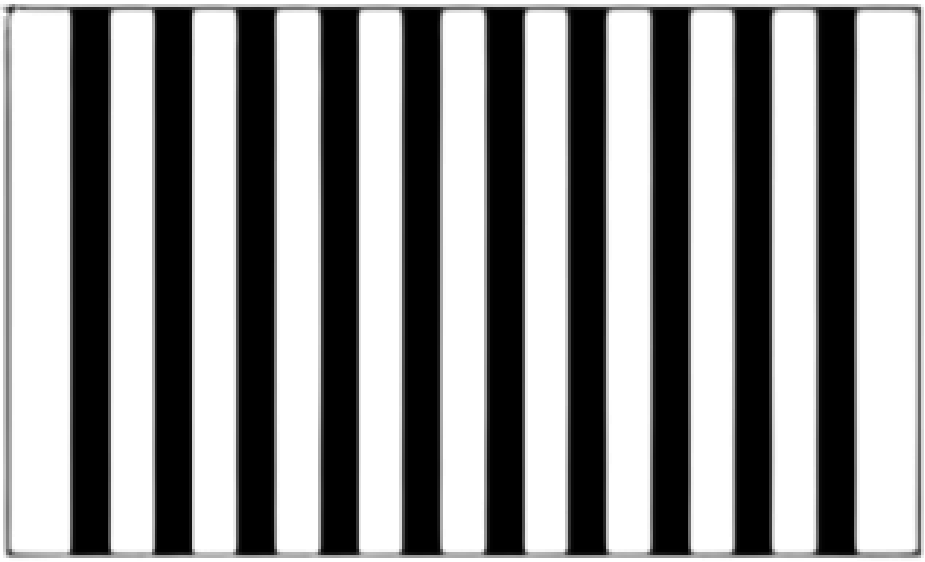
   What do you think are some of the considerations that go into deciding how many megapixels a camera has, or how big the resolution is of a computer-generated image? Try to think of physical, hardware and software related reasons.

b) After an image is sampled, it needs to be quantized before we can actually see it in its final form (remember that computers are mostly digital devices, so analogue values can't just be stored "as-is" in memory).

   What do you think are some of the considerations that go into deciding how strong the quantization of an image is? Try to think back about older computer hardware or less advanced screens.

Problem 2 – Nyquist (16 points):

On Transparent paper you are given the following image:



The width of each column is A, and so is the space between each column. Note that the distance unit is 1cm.

This paper is a sampling grid, which is laid on a continuous image.

You are given the periodic image $I(x, y) = sin(\pi kx)$ where k is an unknown parameter.

For each of two values of A, {0.25, 2}, answer the following:

a) What is the wavelength of this image along the x-axis?

b) What are the k values that using them, we can restore the image I(x,y)?

Hint: Think about exercise 3 in Tutorial 1 and note the differences.

Problem 3 – Histograms, Matching, Quantization (70 points):

Have you ever used the Haifa University app to look at your grades? If so, you might be familiar with how the grade histograms look like. In this question, we'll be using what we've learned so far in the course to read the histograms from the images and print them out in text form. That means we'll need to identify numbers, quantify the histogram bins, and measure the numbers they represent.

Our dataset consists of images of grade histograms and images of the 10 digits. We aim to use those digit images to recognize the digits in the histogram images, get the relative bin heights in pixels and then retrieve the number of students per bin.

For example, the following image's output is: 1,1,1,3,1,2,9,3,3,0.

Note: In this task, all images have the same pixel proportions.
In other words, you can "hardcode" your search areas for any one histogram image you'll work on, and it'll fit for all the others.

For this first homework, you're given a skeleton of a script to start working with called TranscribeHistogram.py. You'll need to do the following steps, and where relevant please include the results in the PDF document that will be part of your submission:

a) Call the function 'read_dir' and display the first image. Make sure the image is in grayscale and is indeed displaying the first image within the folder ‘data’.

b) After seeing that you can properly read and display the histogram images, read the digit images from their directory using read_dir, same as before. We’ll want to find the highest number along the vertical axis of the histogram so we can later determine how many students each bar represents.

c) Implement the function 'compare_hist(src_image, target)'. The function receives the source image (which is a grade histogram) and target, which is an image of a digit. The function should implement the histogram-based pattern matching functionality as seen in lectures, using the EMD between histograms to compare a window to the target. The function should return whether a region was found with EMD < 260.

   - Use np.lib.stride_tricks.sliding_window_view(image, (height, width)) to create the windows. The result is an array with a shape of (hh,ww,height,width), and each window[hh,ww] represents a corresponding window with size (height,width).
   - Use cv2.calcHist([windows[y,x]], [0], None, [256], [0, 256]).flatten() to calculate the window’s histogram (256 length array).
   - Since we will need only the topmost number (e.g 6 in a.jpg), you can search just the region around it without needing to look further down the image.

   With this function you can find if a digit is present in the image.

d) Call the function 'compare_hist' on the first image with the numbers in decreasing order, starting from 9 until 0. The first number which returned True is the number we recognized! For example, if you successfully detected 8, then there’s no need to continue, as no lower digit will also be true (the 260 threshold is sufficient).

   Make sure that all the images in the 'data' directory give the correct result when using compare_hist on them.

e) Our next goal is to find the height of each bar within an image. Before we can do that, we need to be able to distinguish between the bars and the rest of the image. Seeing as the histogram images have only a limited variety of colors in them, we can try quantization to simplify the image!

   Call the 'quantization' function with the first image, display the result and then determine the optimal number of gray levels for this exercise's needs. Explain your choice (in the PDF you'll submit).

   After quantizing an image down to the number of gray levels you'll find is best, you can safely threshold the quantized image to black & white, without being afraid to lose any bar. This will simplify the remaining work.

f) Using the modified images, create a list of 10 items, where each item is the i^{th} bar height. Use 'get_bar_height' which receives the image and bin id (0-9) and returns the height of that bar in pixels.

g) Finally, we will use the topmost number we found earlier and the heights of the bars to calculate the real students' number for each bar with the following formula:

   #students-per-bin = round(max-student-num * bin-height / max-bin-height)

You are now ready to transcribe all given histogram images in the data folder. Here's an example of one line of output your program should print:

'Histogram a.jpg gave 1,2,3,4,5,6,7,8,9'

Please transcribe all images and include a small screenshot of the resulting printed lines.

Remember to follow the submission instructions!