Homework 1

Introduction to Signal and Image Processing

Handout: March 4th, 2020
Handin: March 18th, 2020, 15:15

Instructions

Your hand-in will consist of a `.zip` archive named `hw1_firstName_lastName.zip` containing the following:

- Python source files named `hw1_exY_firstName_lastName.py`, where Y is the exercise number.
- All necessary files to run the above.
- Your report named `hw1_firstName_lastName.pdf`.

Your archive must be uploaded on ILIAS before the deadline.

Code

Code templates are provided to help you get started in solving the exercises. In the typical case, you will fill-in the missing function bodies and code blocks. Note that you may also make your own code from scratch.

IMPORTANT: In general, if you are not able to produce a functional code (i.e. your script crashes), comment out the concerned part, and if necessary give a short analysis in your report. Scripts that do not run will be penalized.

Report [4 points]

Along with the code, you are asked to provide a short report. Comment on all the questions on the report (both theory and coding), show your results and briefly explain what you did to obtain them. If you encountered problems and did not manage to finish the exercise, explain here what you did. On ILIAS you will find a LaTeX template to get started. Note that the use of LaTeX is NOT mandatory.

1 Fourier Transform [1 point] - Theory question

For the following continuous functions $f(x)$, sketch the magnitude spectrum $|F(u)|$, i.e. the absolute value of their Fourier Transform (sketches by hand are sufficient):

- The impulse $f_1(x) = 1/2 \cdot \delta(x)$,
- the function $f_2(x) = 2 \cdot cos(2\pi u_a x) + cos(2\pi u_b x)$, with the frequencies $u_b > u_a > 0$.

2 Lloyd-Max quantization [3 points] - Theory Question

Suppose we have K intervals in the range of possible intensity values defined by levels $z_1, z_2, ..., z_{K+1}$. In each interval $[z_k, z_{k+1}]$, we wish to have a constant intensity value $q_k \in \{0, ..., 255\}$. Given that our intensity data follows the probabilistic distribution $p(z)$, such that $z \in \mathbb{R}$ and $p(z) \geq 0$, the Lloyd-Max method attempts to find the values z_k with $k = 1, ..., K+1$ and q_k with $k = 1, ..., K$ by minimizing the following error

$$\varepsilon = \sum_{k=1}^{K} \int_{z_k}^{z_{k+1}} (z - q_k)^2 p(z) \mathrm{d}z \tag{1}$$

2.1 [1 point]

Show that

$$z_k = \frac{q_{k-1} + q_k}{2} \tag{2}$$

by minimizing equation 1 with respect to z_k. Show and justify all your steps.

2.2 [1 point]

Show that

$$q_k = \frac{\int_{z_k}^{z_{k+1}} z \cdot p(z) \mathrm{d}z}{\int_{z_k}^{z_{k+1}} p(z) \mathrm{d}z} \tag{3}$$

by minimizing equation 1 with respect to q_k. Show and justify all your steps.

2.3 [1 point]

Suppose we have a signal with amplitudes uniformly distributed between $0 \leq z \leq 1$ (i.e., $p(z) = 1$ on this interval). We want to represent the signal with 2 intensity values q_k, and therefore 3 boundaries z_k (where $z_1 = 0$ and $z_3 = 1$). Find the solution for q_1, q_2 and z_2 for the first 2 iterations using the Lloyd-Max method (calculations by hand). Use $q_1 = 0.3$ and $q_2 = 0.8$ to initialize the first iteration. To which values do you think will the algorithm converge to?

3 Chamfer distance maps [9 points] - Coding question

For this question, you will create distance maps of edges. You are provided with a set of binary edge maps of simple shapes, found in `shapes`. The provided edge maps are also shown in figure 1.

Figure 1: Edge maps of shapes

For each one of these shapes, you are asked to produce a corresponding, same size image containing the L1 norm of each pixel to its closest edge. Your script should show a figure with two rows and four columns, with the top row having the edge maps (see Figure 1), and the bottom row the distance maps you calculated.

There is a brute-force approach to calculate those distance maps, by calculating, for every pixel, the L1 norm to all the edges and keeping the minimum value. That would lead to a complexity of $\mathcal{O}(n^2)$, where n is the number of pixels.

There is also a less computationally demanding way, with a complexity of $\mathcal{O}(n)$. This algorithm is described below.

You may choose any approach you prefer for your solution. In case you use the optimal approach, you need to provide clues on boundary conditions.

Chamfering Algorithm

1. Create an array `distance_map` of same shape as `edge_map`.
2. Initialize `distance_map` with 0 corresponding to edges of `edge_map`, and ∞ otherwise.
3. Pass through the image row by row, from top to bottom and left to right. For each central pixel x (figure 2), set

$$\texttt{distance\_map}(x) = \min_{q \in AL} [\mathrm{L}_1(x, q) + \texttt{distance\_map}(q)].$$

4. Pass through the image row by row, from bottom to top and right to left. For each central pixel x (figure 2), set

$$\texttt{distance\_map}(x) = \min_{q \in BR} [\mathrm{L}_1(x, q) + \texttt{distance\_map}(q)].$$

5. The array `distance_map` now holds a chamfer of the edges.

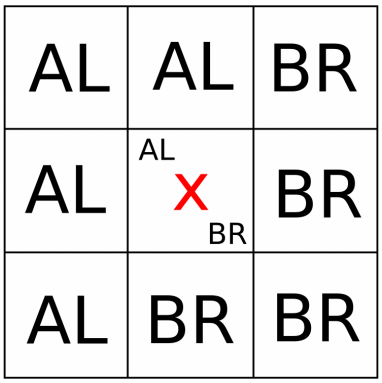

Figure 2: Neighboring pixels for the chamfering algorithm. AL: Above or Left, BR: Below or Right, with respect to the central pixel. Note that the central pixel x belongs both to the AL and the BR sets.

4 Bilinear interpolation [8 points] - Coding question

In this exercise you are asked to implement bilinear interpolation and apply it to resize some example images. Despite its name, bilinear interpolation is a non-linear interpolation method that computes the values of pixels located at new image coordinates in the resized image plane. At its core, this method applies two sequential linear interpolations on the image – first scales the image row-wise, then scales again the previously row-wise scaled image column-wise. This procedure is separately applied on the available channels, e.g. one repeats the same procedure for R, G, B channels separately if an RGB image is given.

4.1 Linear interpolation [4 points]

4.1.1 [2 points]

Implement linear interpolation. Given a set of `(y_vals, x_val)` (signal, support) and new locations `x_new` calculate `y_new`.

4.1.2 [2 points]

Implement rescaling of a 1-D signal using interpolation. Given a signal and a scaling factor, produce a rescaled one dimensional signal. Demonstrate results for a 1-D signal of your choice.

4.2 Bilinear interpolation [4 points]

4.2.1 [2 points]

Implement rescaling of a 2-D signal (e.g. a grayscale image) using interpolation. Given an image and a scaling factor, produce a rescaled image. You can either implement this from scratch, or internally use the 1-D interpolation. Demonstrate results for an image of your choice (please make sure it is grayscale so it only has 1 channel).

4.2.2 [2 points]

Extend the above for a 3 channel RGB image.