Homework 1

Introduction to Image Analysis

Handout: 8th, March 2023
Handin: April 5th, 2023, 13:15

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

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.

Figure 1: Edge maps of shapes

1 Regular Tesselation [1 point] - Theory Question

Two key properties of tesselations, as seen in class, are that when cells are placed on a 2D plane, they (1) do not overall each other and (2) cover the entire space (i.e. no holes). For example, square-shaped cells satisfy these two conditions. For the case of regular tesselation, where each cell is the same size and shape, answer the following questions:

1.1 [0.5 points]

Identify all shapes that satisfy the two conditions above. (Hint: there are 3 cases).

1.2 [0.5 points]

Formally show that no other shape satisfies the above conditions.

2 Chamfer distances [2.5 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.

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(figure 1), and the bottom row the distance maps you calculated.

There is a brute-force approach to calculating 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 do not need to provide clues on boundary conditions.

Chamfering Algorithm - Coding question

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

$$\texttt{distance\_map}(x) = \min_{q \in AL} [L1(x, q) + \texttt{distance\_map}(q)]$$

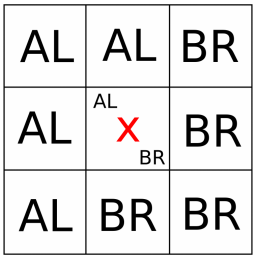

Figure 2: Neighboring pixels. AL: Above or Left, BR: Below or Right, with respect to the central pixel. Note that the central pixel belongs *both to the AL and the BR sets*.

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

$$\texttt{distance\_map}(x) = \min_{q \in BR} [L1(x, q) + \texttt{distance\_map}(q)]$$

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

3 Bilinear interpolation [3 points] - Coding question

We aim to resize an image using bilinear interpolation. 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 to the available channels, e.g. one repeats the same procedure for R, G, B channels separately if an RGB image is given.

In this question, you are asked to implement bilinear interpolation and apply it on some example images.

3.1 [2 points]

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

3.2 [1 point]

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.