Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Vision
and Geometry Lab

ETH Zurich
Computer Science Department

Computer Vision

Lab Assignment - Image Segmentation

Lara Nonino - lnonino@student.ethz.ch

November 2023

Student ID:
22-937-833

Section 1

Mean-Shift Algorithm

For this task, we were asked to implement the *Mean-Shift Algorithm* for image segmentation. Two methods have been used: the first one consists in processing each update considering only one point at a time (naive method), while the second one consists in accelerating the algorithm processing a batch of points for each iteration (accelerated method).

1.1 Implement the distance Function

The function receives the image `X` and a point `x` as parameters. Note that `X` has dimension $(n \times c)$ where n is the number of image pixels and c is the number of channels.

- **Naive implementation**
  Since `x.shape = [c]`, I firstly used `x.unsqueeze(0)` to have `x.shape = [1,c]` and then I computed the distance between `x` and each point of the image using `dist = torch.cdist(x.unsqueeze(0), X)`.
- **Accelerated implementation**
  Since `x.shape = [BATCHSIZE,c]`, I directly computed the distance between `x` and each point of the image using `dist = torch.cdist(x, X)`.

The result `dist` has dimension $(batchsize \times n)$. Note that in the naive implementation, $batchsize = 1$ since we are considering one point at a time.

1.2 Implement the gaussian Function

The function computes the weights of points according to the distance computed by the distance function. It receives the distances tensor and the gaussian bandwidth as inputs.
Since the normalization term of gaussian is a constant factor in each weight, it is possible to simplify it (it would be useless to compute a weighted mean).

$$weight = e^{-\frac{d^2}{2\sigma^2}}$$

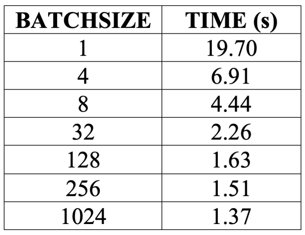
1.3 Implement the update point Function

The function updates the points according to weights computed from the gaussian function. The output is the weighted mean of all points for each channel (`w_mean.shape = [BATCHSIZE, 3]`)

$$w\_mean = \frac{\sum weight_i \cdot x_i}{\sum weight_i}$$

1.4 Accelerating the Naive Implementation

Batch processing consists in computing parallel updates of multiple points (a batch of points) at the same time, therefore it is likely that the computation time decreases as the batch size increases.
However it is not a good idea to have a too large batch since the tensor operations may require too much memory. In this case the computer will not execute pure parallel computations, but will instead combine both iterative and parallel processing. This might lead to a non-decrease of the timing.
The following table shows the timings (in seconds) obtained with different batch sizes.



BATCHSIZE	TIME (s)
1	19.70
4	6.91
8	4.44
32	2.26
128	1.63
256	1.51
1024	1.37

Figure 1.1: Elapsed time for mean-shift