CV Lab Assignment Report - Image Segmentation

ETH Zürich, 263-5902-00L Computer Vision HS2022
Prof. Marc Pollefeys, Prof. Siyu Tang, Dr. Fisher Yu

Anisha Mohamed Sahabdeen

1 Mean-Shift Algorithm

Image segmentation can be performed by means of the mean-shift algorithm, a widely used unsupervised learning method. Details of a possible Python implementation of the mead-shift algorithm are provided below.

- `distance` **method.** As a first step, the distance of each of the points of the image with respect to all other points is calculated. This is done via the the `torch` library function `cdist`, which calculates the batched the p-norm distance between each pair of the two collections of row vectors. Here, we consider the Euclidian distance between points, so that p is set to 2.

  ```
  dist = torch.cdist(x, X, p=2).squeeze()
  ```

- `gaussian` **method.** For each of the points in the image, an image weight map should be computed according to the distance from the considered point. For this purpose, a Gaussian kernel is employed to compute the weights of the image points.

  ```
  weights = torch.exp(-dist**2 / (2 * (bandwidth**2))) ,
  ```

  where `bandwidth` is the standard deviation of the Gaussian.

  Note that the normalization constant $\frac{1}{\sigma\sqrt{2\pi}}$ is omitted since it will be simplified in the following operation.

- `update_point` **method.** Finally, each point p is reassigned the value of the weighted mean M_p of all the image points:

  $$M_p = \frac{\sum_{i=1}^{n} w_i(p) x_i}{\sum_{i=1}^{n} w_i(p)} \ .$$

  Here it is clear that the normalization constant in the numerator and denominator of M_p can be simplified, and can be therefore ignored when calculating the weights.

  ```
  weighted_mean = torch.matmul(weight, X) / torch.sum(weight)
  ```

The functions are called in this order for each of the points in the image a number n of times, thus performing n iterations of the mean-shift algorithm.

Figure 1: Test image.

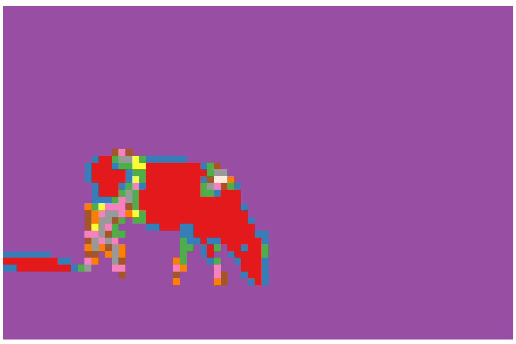

Figure 2: Result of mean-shift on test image.

It is possible to take advantage of tensor broadcasting in order to obtain a vectorized implementation of the algorithm, which results in overall increase in efficiency. As a matter of fact, running the vectorized Python code on the test image (Figure 1) on a M1 CPU instead of the point-wise version resulted in a 86,4% decrease in execution time, going from 8.093 s to 1.093 s, with a batch size of 512 points.