Image Segmentation Report

Nicole Damblon

Computer Vision

1 Mean-Shift Algorithm

The mean shift algorithm finds the modes of an estimated probability density function (KDE). It computes the weighted mean for every data point and updates each point in a copy of the data set with the new mean. Here, the weights are implemented by a gaussian kernel function and are calculated in terms of the distance from each data point to every other point in the set. The updates are spacially bounded by a radius, which is set to infinity here. The weights are calculated in terms of the distance from each data point to every other point in the set. Figure 1 shows the original picture, the expected result and the final mean-shift segmentation.

1. **Calculate distance.** Calculate the euclidian distance between the current point and every point of the data matrix as distance metric

$$distance = \sqrt{x[0] - X[:,0])^2 + (x[1] - X[:,1])^2 + (x[2] - X[:,2])^2} \tag{1}$$

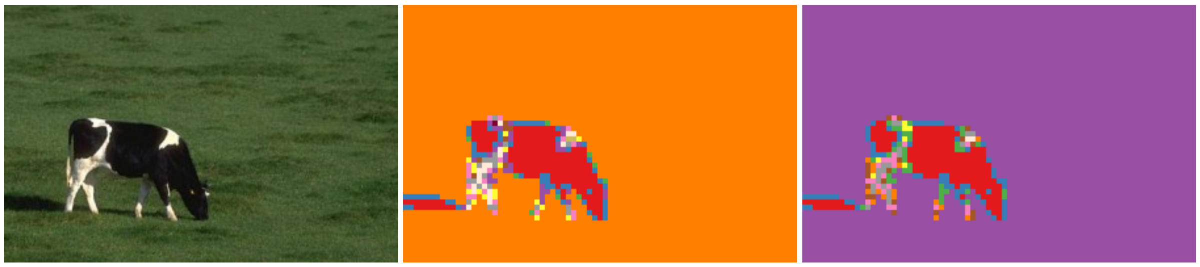
   The batch version uses `torch.cdist` to calculate euclidian distance between all pairs of points.
2. **Calculate gaussian kernel**. A gaussian function without a normalizing term is used to update the points. The outcome of the segmentation is the same when using a gaussian pdf here (because a normalization term that ensures that the weights sum up to 1 would cancel out). The bandwidth parameter controls how fast the weights are decaying with the distance from the mean:

$$\exp\left(-\frac{distance^2}{2 * bandwidth^2}\right) \tag{2}$$

3. **Update each point of the data matrix**. Each point of the input matrix is updated with its weighted mean. The mean is calculated as sum over all points multiplied by the corresponding weight and normalized with the sum of all weights. The function `torch.sum` leverages broadcasting effects while calculating the mean. For the fast version, the denominator needs to be reshaped with the torch function `unsqeeze`.

Timing. The mean-shift algorithm is first implemented in a "slow" version, that means every pixel in the image is updated individually by looping over the image 20 times. An implementation without additional for-loops takes 16.77 seconds to segment the image. On the other hand with updating batches instead of single pixels, the runtime is reduced to 7.8 seconds. Both versions are implemented using the broadcasting semantics of pytorch and numpy.

Note that the difference in color may be caused by different versions of python packages.



(a) Original image. (b) Sample result. (c) Result achieved with "slow" and "fast" mean-shift implementation.

Fig. 1. Image segmentation with mean-shift.