Readings

- Klette 5, 5.2 — Intro to segmentation, mean-Shift Segmentation
- Szeliski 5.3, 5.3.1, 5.3.2 — k-Means, Mean-shift
- Forsyth & Ponce 9.3 — Mean Shift Segmentation (uploaded online to LumiNUS)
- http://www.chioka.in/meanshift-algorithm-for-the-rest-of-us-python/

Summary

Motivation

- Segmentation separates image into coherent regions, sometimes to increase efficiency for further processing e.g. of recognition, other algorithms
- One interpretation of image segmentation is via clustering; each cluster is a segment
- To cluster, we need to consider how to:
  - Represent the points (feature representation)
  - Measure their differences (distance measure)
  - Do the clustering (clustering algorithm)

k-Means Clustering

- Iteratively assigns cluster membership and computing cluster centers
- Pros: simple, fast to compute, guaranteed to converge (to some local minimum)
- Cons: setting k non-trivial, sensitive to initialization, outliers, finds only spherical clusters
- To perform segmentation, represent each pixel as a feature (e.g. greyscale intensity, colour, colour + x,y position) and apply k-means clustering in feature space

SLIC Superpixels

- Superpixels are a group of local pixels that share common characteristics, e.g. intensity
- SLIC uses k-means to find superpixels based on their similarity in intensity and location

Mean-Shift Clustering

- The mean shift algorithm is a mode-seeking algorithm that finds local density maxima
- For each data point, iteratively shift a window towards its local centroid until arrival at some mode (attraction basin)
- Derivation shows that mean-shift performs gradient ascent to find the local maximum on the data distribution via a kernel density estimate
- To perform segmentation, apply mean shift to each point in feature space; assign all data points in an attraction basin to a single cluster
- Parameterized by bandwidth or window size *h*; clustering results vary depending on *h*
- Pros: no assumption on shape, variable number of modes, robust to outliers
- Cons: *h* hard to set, computationally expensive
- Speedups possible by reducing the number of points we run mean-shift over

FAQs

Q: On slide 38 of lecture 5, points within radius r of a particular mode will belong to that mode. Is r necessarily smaller than or equal to the window size to get good clustering?

A: No strict rules -- depends on your kernel and the problem setting.

Q: On slide 20, there is the equation for composite distance. How do you manipulate the equation on the left to get the equation on the right? I attach my attempt below.

$$D = \left[\left(\frac{d_c}{d_{cm}} \right)^2 + \left(\frac{d_s}{d_{sm}} \right)^2 \right]^{1/2} \quad \Rightarrow \quad D = \left[d_c^2 + \left(\frac{d_s}{s} \right)^2 c^2 \right]^{1/2}$$

$$D = \left[\left(\frac{d_c}{d_{cm}} \right)^2 + \left(\frac{d_s}{d_{sm}} \right)^2 \right]^{\frac{1}{2}}$$

$$= \left[\left(\frac{d_c}{c} \right)^2 + \left(\frac{d_s}{s} \right)^2 \times 1 \right]^{\frac{1}{2}}$$

$$= \left[\frac{d_c^2}{c^2} + \frac{d_s^2}{s^2} \times \frac{c^2}{c^2} \right]^{\frac{1}{2}}$$

$$= \frac{1}{c} \left[d_c^2 + \frac{d_s^2}{s^2} c^2 \right]^{\frac{1}{2}}$$

A: You are correct there is this extra 1/c factor. However, because it is set as a constant, it simply scales the entire distance by this amount so we usually ignore it.

*Q: On slide 22, why do we consider 2s * 2s neighbour instead of s * s? Since each centre of a superpixel is s pixels away from each other, isn't s* s neighbours (i.e. 0.5s on top, bottom, left, right) be sufficient? Beyond that, the pixel is closer to other neighbouring centres?*

A: The 0.5s only works well if your superpixels are regular (squares). If the shapes become more irregular this gives more flexibility. Note that this 2s is simply a recommended value trading off between efficiency (i.e. less points given to the clustering algorithm) vs. rate of change over the iterations (limiting to the 0.5s range)

Q: How does changing the kernel shape impact the mean shift algorithm?

A: Consider how the output will differ if you use a square kernel vs. a circular one.

Q: I don't get why we are required to formulate it as such

Use the smallest eigenvalue as the response function

$$R = \min(\lambda_1, \lambda_2)$$

For us, $A = H$ is a 2x2 matrix; we can directly solve for the eigenvalues:

$$\lambda_{\pm} = \frac{1}{2} \left[(h_{11} + h_{22}) \pm \sqrt{4 h_{12} h_{21} + (h_{11} - h_{22})^2} \right]$$

Wouldn't we always pick lambda_- since it is always < lambda_+?

A: Yes, but eigenvalues are rarely associated with the explicit definition which is only for 2x2 matrices. There are other ways of solving for the eigenvalues. Additionally, the theory for the second moment matrix is given in terms of the smaller eigenvalue.