**Develop a mean-shift segmentation algorithm for color images (Comaniciu and Meer 2002).**

1. Convert your image to L*a*b* space, or keep the original RGB colors, and augment them with the pixel (x, y) locations.
2. For every pixel (L, a, b, x, y), compute the weighted mean of its neighbors using either a unit ball (Epanechnikov kernel) or finite-radius Gaussian, or some other kernel of your choosing. Weight the color and spatial scales differently, e.g., using values of (hs, hr, M) = (16, 19, 40) as shown in Figure 5.18.
3. Replace the current value with this weighted mean and iterate until either the motion is below a threshold or a finite number of steps has been taken.
4. Cluster all final values (modes) that are within a threshold, i.e., find the connected components. Since each pixel is associated with a final mean-shift (mode) value, this results in an image segmentation, i.e., each pixel is labeled with its final component.
5. (Optional) Use a random subset of the pixels as starting points and find which component each unlabeled pixel belongs to, either by finding its nearest neighbor or by iterating the mean shift until it finds a neighboring track of mean-shift values. Describe the data structures you use to make this efficient.
6. (Optional) Mean shift divides the kernel density function estimate by the local weighting to obtain a step size that is guaranteed to converge but may be slow. Use an alternative step size estimation algorithm from the optimization literature to see if you can make the algorithm converge faster.