Segmenting images and mean shift

D.A. Forsyth

Segmenting images

- Why?
  - To find “chunks” of image that have meaning
    - possibly objects
  - Because pixels are too small to work with individually
    - and most pixels are like their neighbors
  - To compress the image

Example segmentations

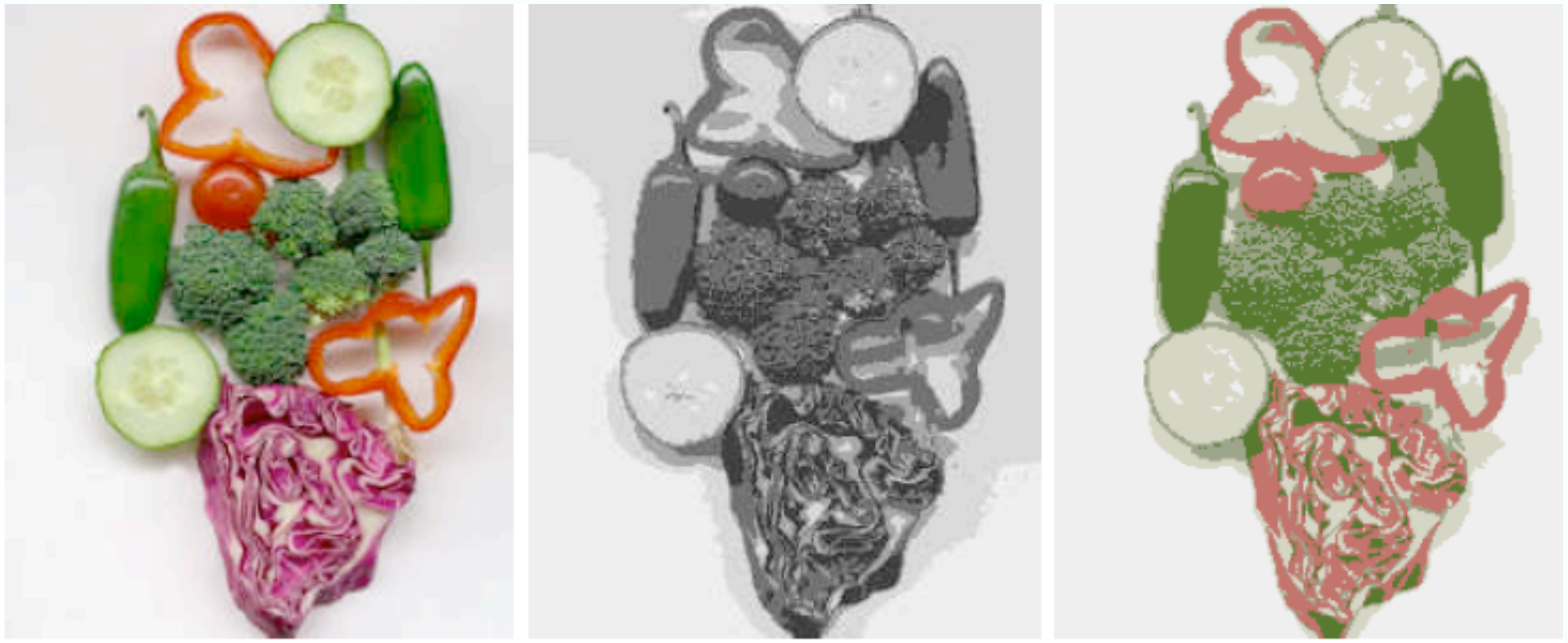


FIGURE 9.17: On the **left**, an image of mixed vegetables, which is segmented using k-means to produce the images at **center** and on the **right**. We have replaced each pixel with the mean value of its cluster; the result is somewhat like an adaptive requantization, as one would expect. In the center, a segmentation obtained using only the intensity information. At the right, a segmentation obtained using color information. Each segmentation assumes five clusters

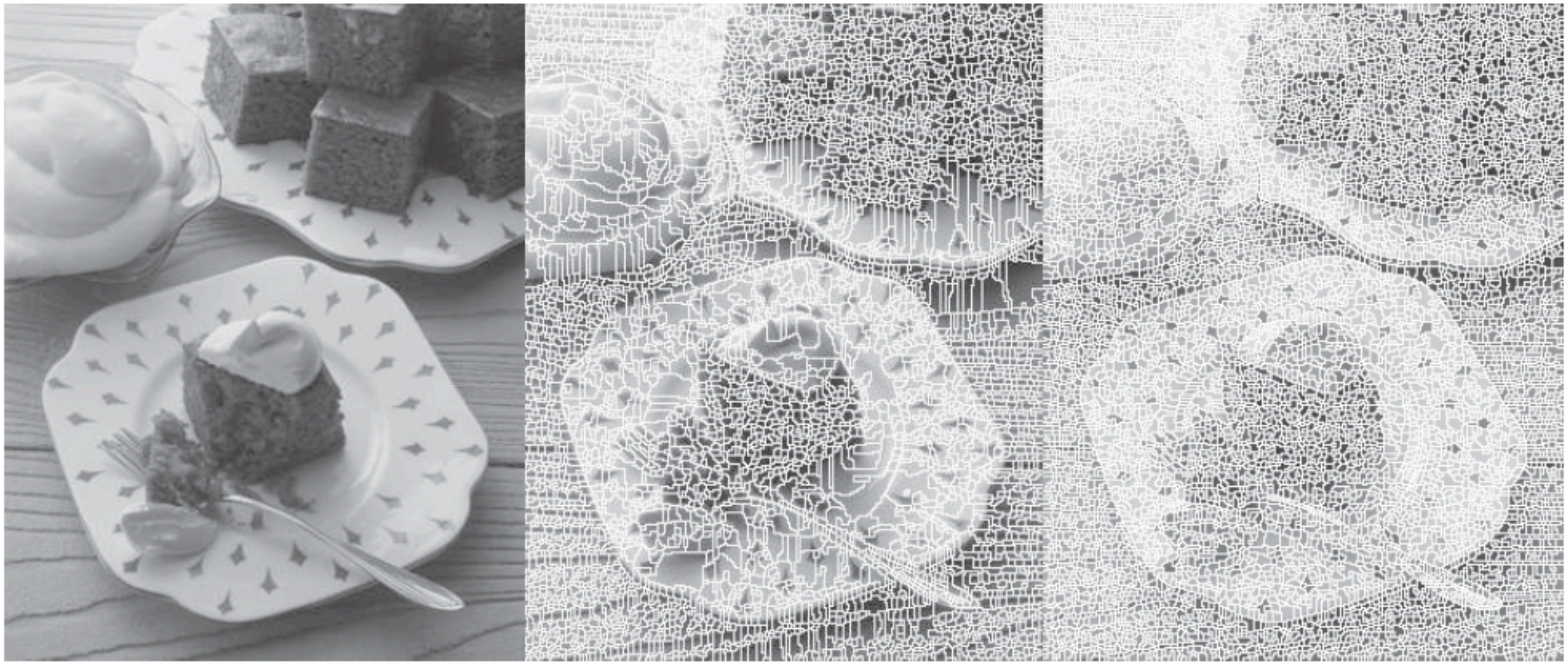


FIGURE 9.16: Segmentation results from the watershed algorithm, applied to an image by Martin Brigdale. **Center:** watershed applied to the image intensity; notice some long superpixels. **Right:** watershed applied to image gradient magnitude; this tends to produce rounder superpixels. *Martin Brigdale © Dorling Kindersley, used with permission.*

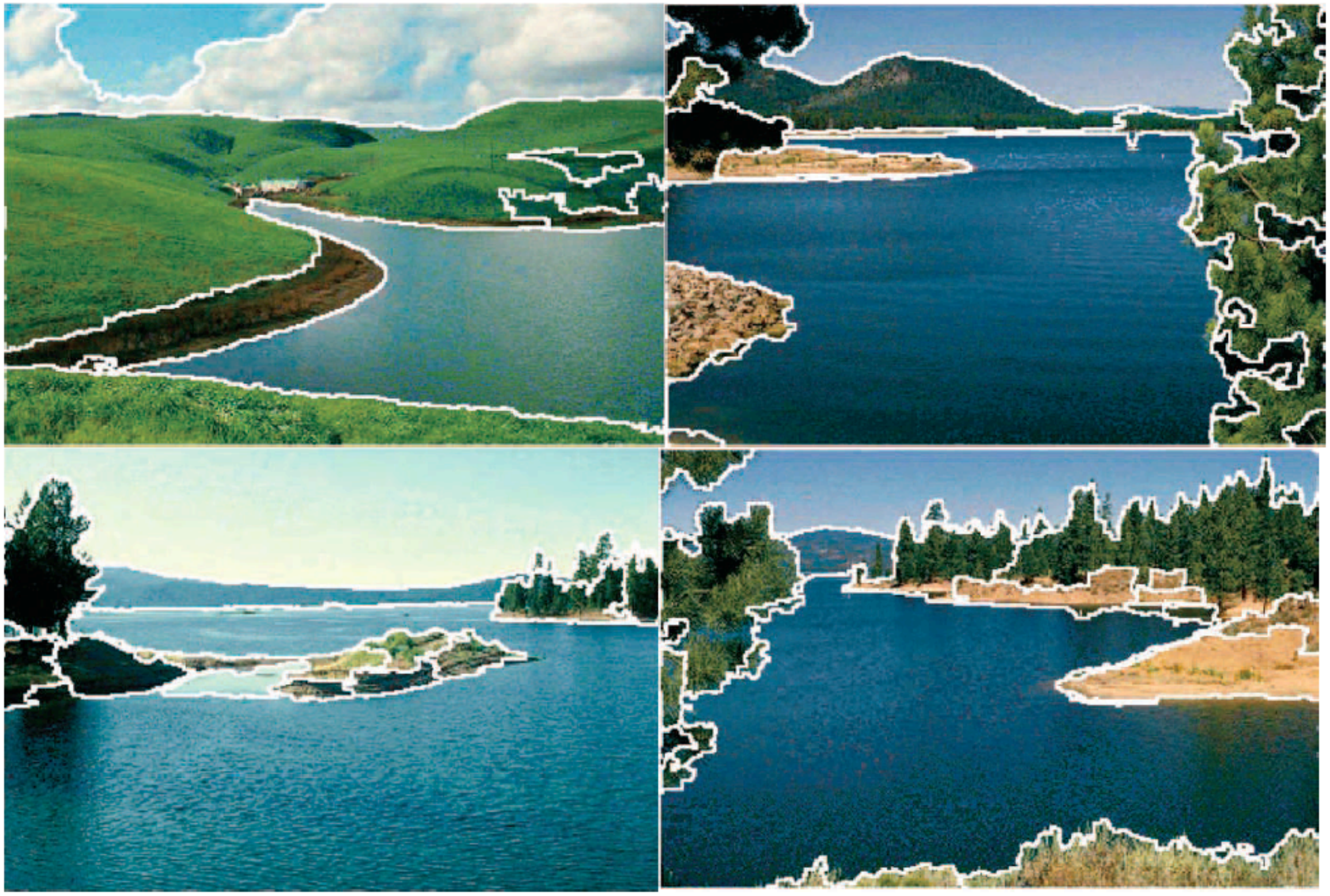


FIGURE 9.21: Segmentations of images obtained using the mean shift algorithm. *This figure was originally published as Figure 10 of "Mean Shift: A Robust Approach Toward Feature Space Analysis," by D. Comaniciu and P. Meer, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002 © IEEE, 2002.*

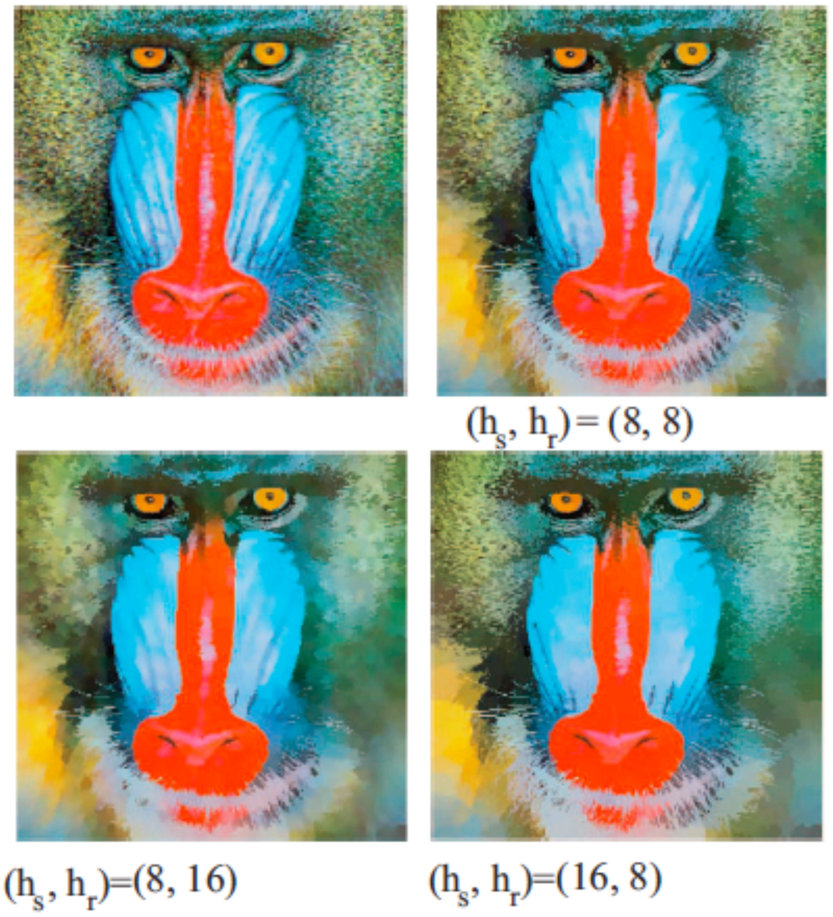


FIGURE 9.20: An image (**top left**) and mean shift modes obtained with different clustering scales for space h_s and appearance h_r. If h_s is small, the method must produce clusters that are relatively small and compact spatially because the kernel function smoothes over a relatively small radius and so will allow many distinct modes. If h_r is small, the clusters are compact in appearance; this means that small h_s and large h_r will produce small, blobby clusters that could span a range of appearances, whereas large h_s and small h_r will tend toward spatially complex and extended clusters with a small range of appearances. Cluster boundaries will try harder to follow level curves of intensity. *This figure was originally published as Figure 5 of "Mean Shift: A Robust Approach Toward Feature Space Analysis," by D. Comaniciu and P. Meer, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002 © IEEE, 2002.*

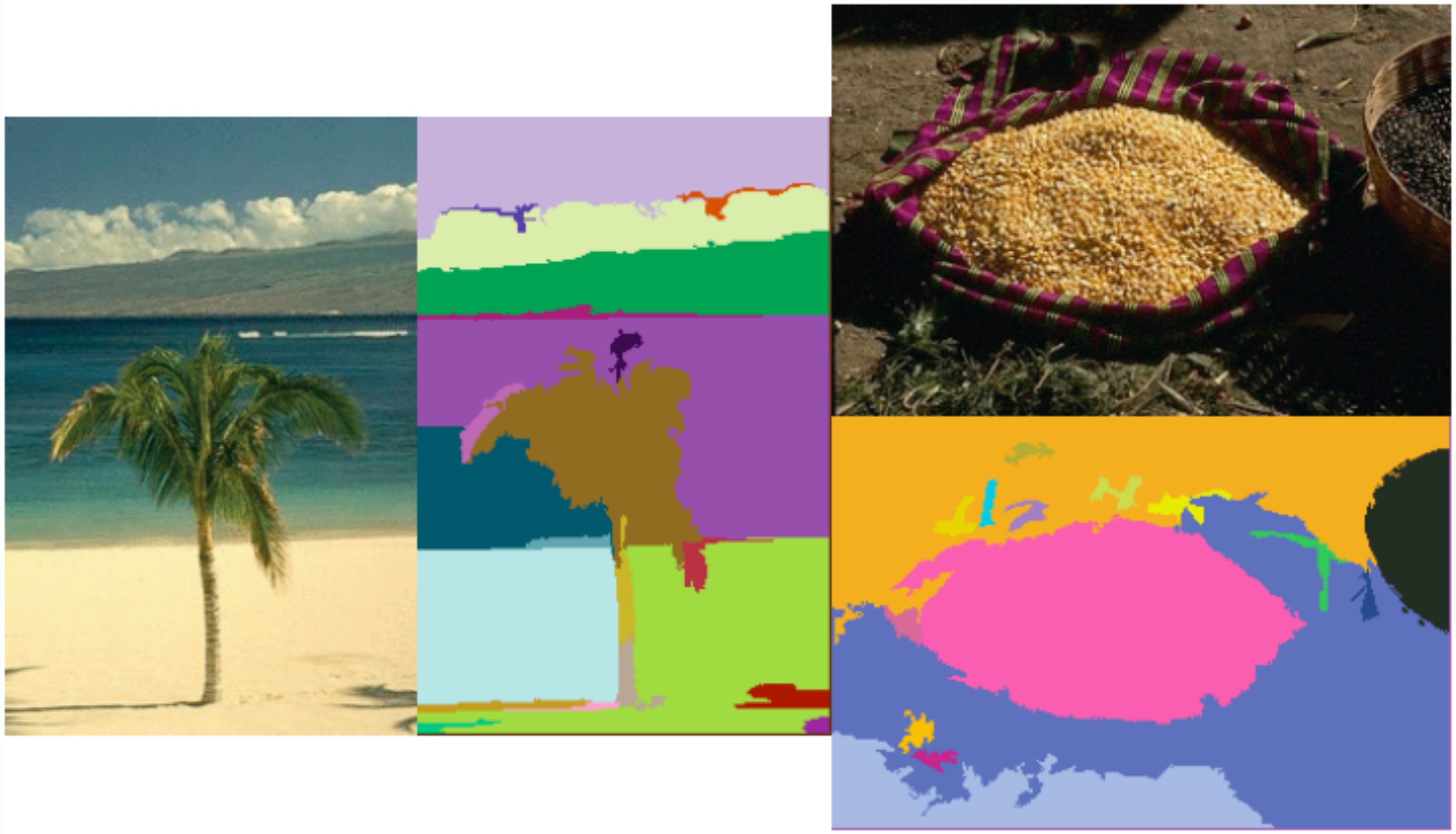


FIGURE 9.22: Images segmented using Algorithm 9.8, shown next to segments. Figures obtained from `http://people.cs.uchicago.edu/~pff/segment/`, by kind permission of Pedro Felzenszwalb.

Example applications

- Label images by:
  - segment regions
    - compute region description
    - vector quantize region descriptions
  - now have a translation problem
    - solve
  - Label new image with translation table

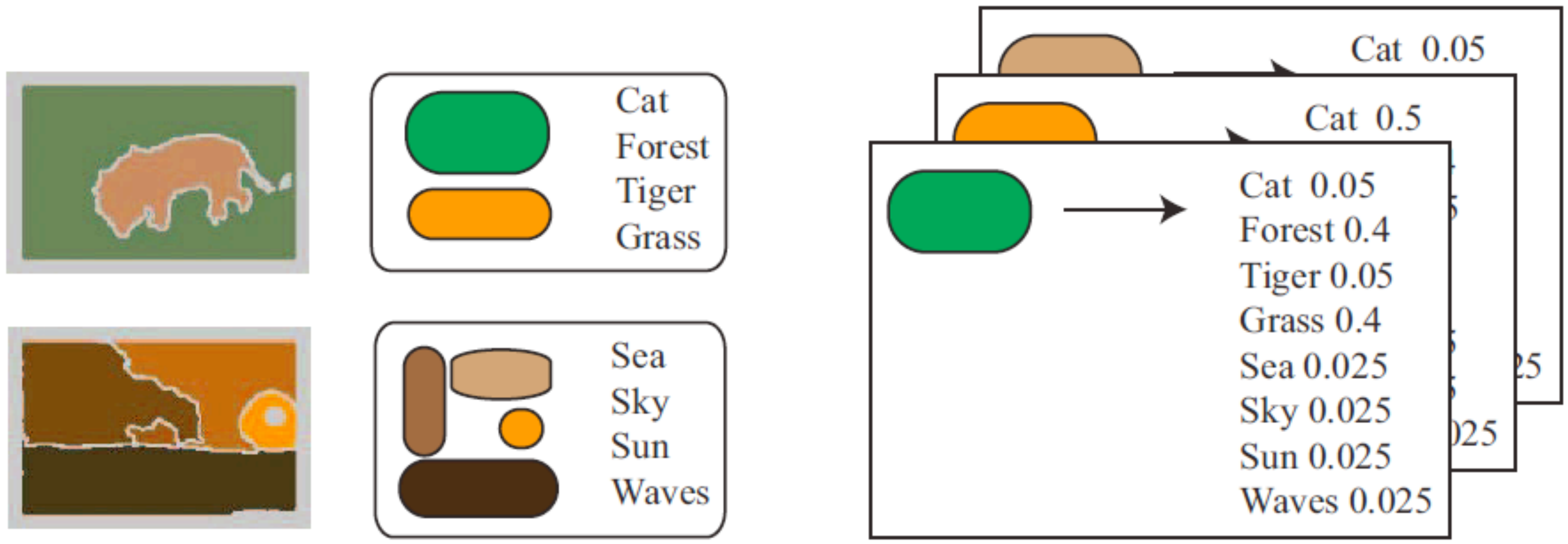


FIGURE 21.13: Duygulu *et al.* (2002) generate annotations for images by segmenting the image (**left**) and then allowing each sufficiently large segment to generate a tag. Segments generate tags using a lexicon (**right**), a table of conditional probabilities for each tag given a segment. They learn this lexicon by abstracting each annotated image as a bag of segments and tags (**center**). If we had a large number of such bags, and knew which tag corresponded to which segment, then building the lexicon just involves counting; similarly, if we knew the lexicon, we could estimate which tag corresponded to which segment in each bag. This suggests using an EM method to estimate the lexicon. *This figure was originally published as Figure 1 of "Object Recognition as Machine Translation: Learning a lexicon for a fixed image vocabulary," by P. Duygulu, K. Barnard, N. deFreitas, and D. Forsyth, Proc. European Conference on Computer Vision. Springer Lecture Notes in Computer Science, Volume 2353, 2002 © Springer, 2002.*

Superpixels

- “Chunks” of image larger than a pixel
    - coherent interior
- Can do shape reasoning
- For example, people are made out of long thin superpixels

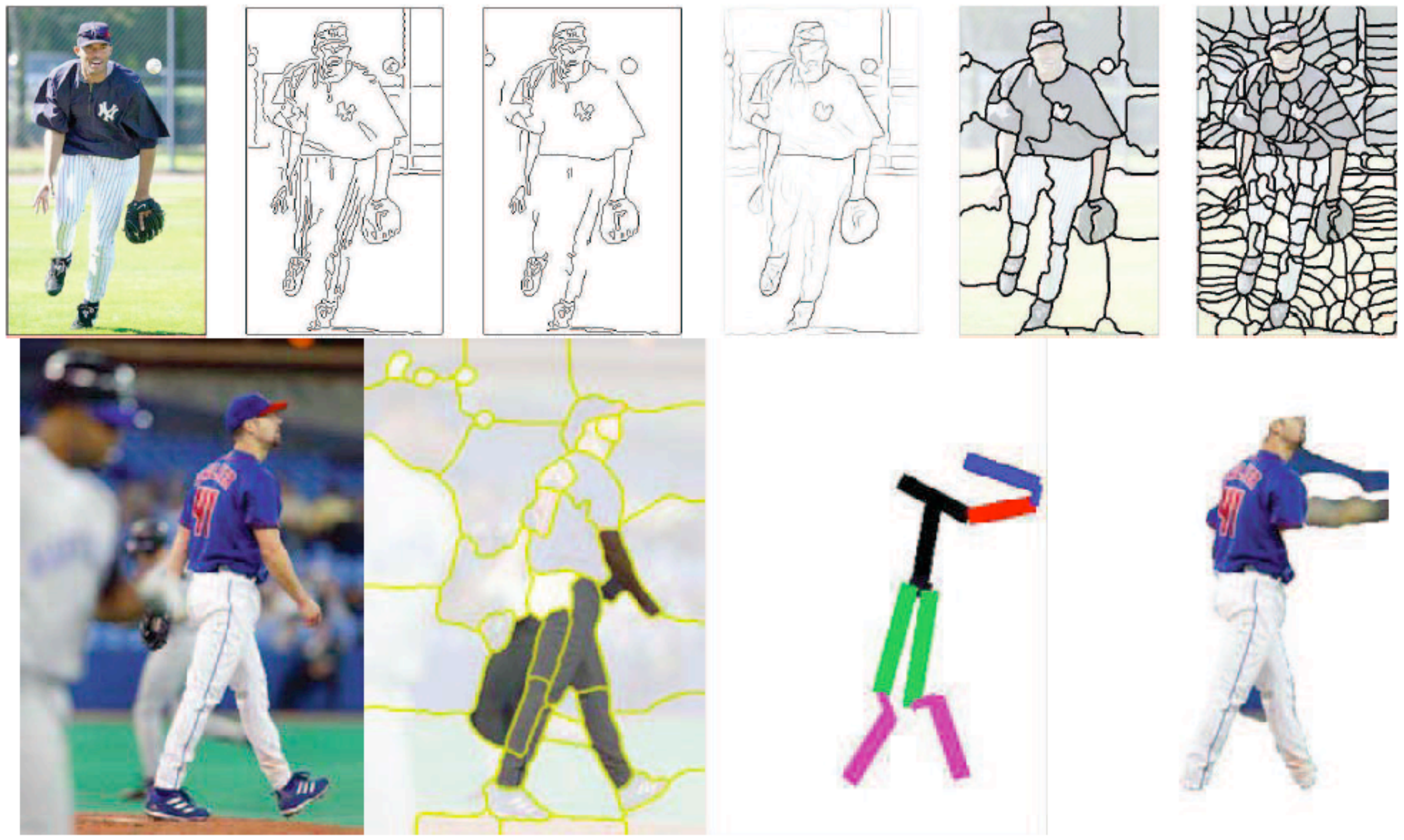


FIGURE 9.14: Superpixels often can expose structure in images that other representations conceal. Human body segments tend to appear as long, thin segments. In the **top** row, an image together with three different edge maps (the edge detector of Section 5.2.1, with two scales of smoothing, and the P_b of Section 17.1.3) and superpixels computed at two "scales" (in this case, the number of superpixels was constrained). Notice that the coarser superpixels tend to expose limb segments in a straightforward way. On the **bottom** row, another image, its superpixels, and two versions of the body layout inferred from the superpixel representation. *This figure was originally published as Figure 3 and part of Figure 10 of "Recovering human body configurations: Combining Segmentation and Recognition," by G. Mori, X. Ren, A. Efros, and J. Malik, Proc. IEEE CVPR, 2004 © IEEE, 2004.*

Interactive segmentation

- Cut out a region from an image
  - for example, so you can move things around in a picture
    - compose other pictures from pieces (collages)
    - remove inconvenient people
- Major practical application
- In video, known as “rotoscoping”
  - remove a figure from a background
  - usually done with a green screen
  - BUT: hair, etc create problems
- Key idea:
  - foreground/background models and 0/1 labels

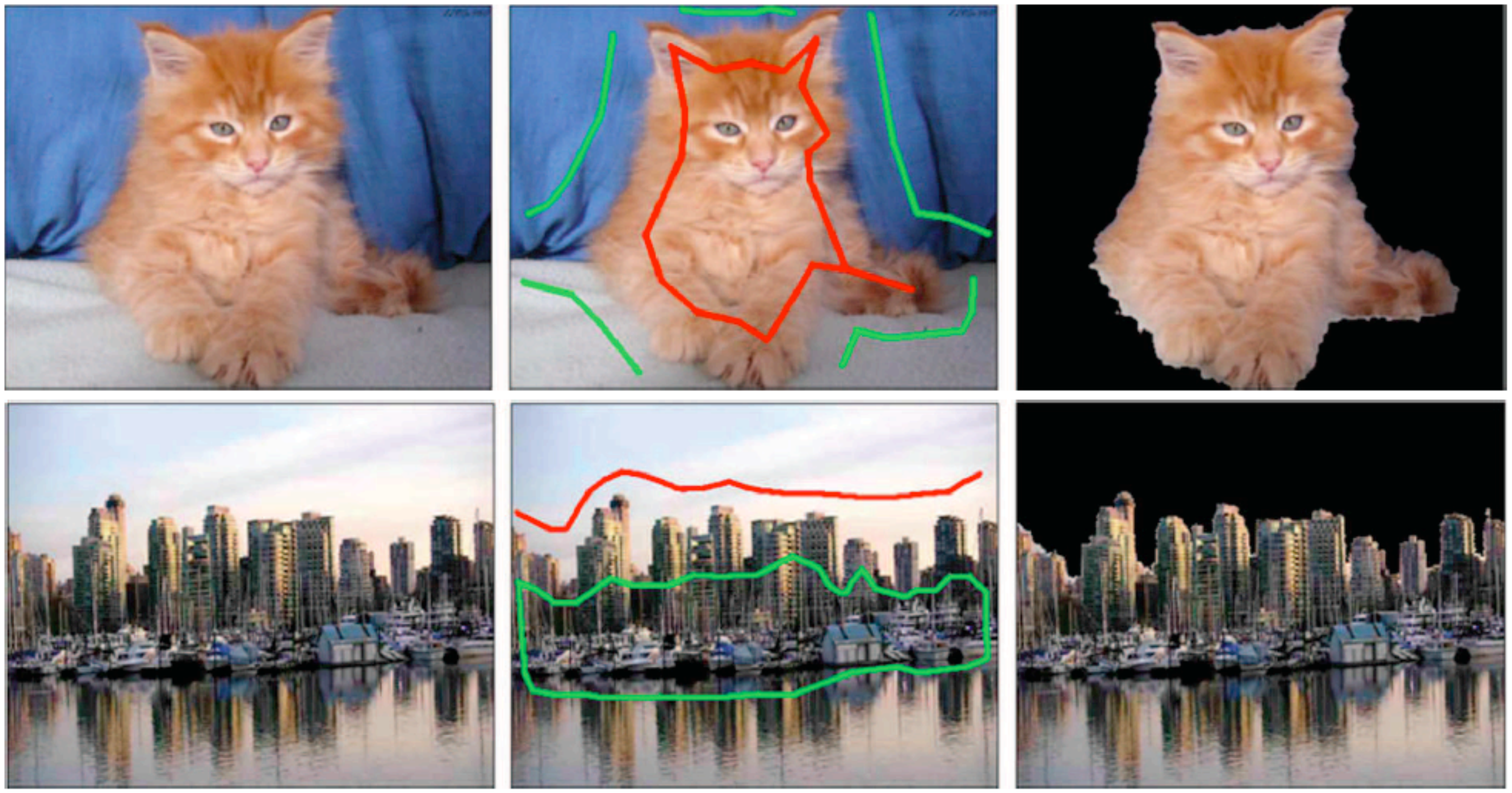


FIGURE 9.11: A user who wants to cut an object out of an image (**left**) could mark some foreground pixels and some background pixels (**center**), then use an interactive segmentation method to get the cut out components on the **right**. The method produces a model of foreground and background pixel appearance from the marked pixels, then uses this information to decide a figure ground segmentation. *This figure was originally published as Figure 9 of "Interactive Image Segmentation via Adaptive Weighted Distances," by Protiere and Sapiro, IEEE Transactions on Image Processing, 2007 © IEEE, 2007.*

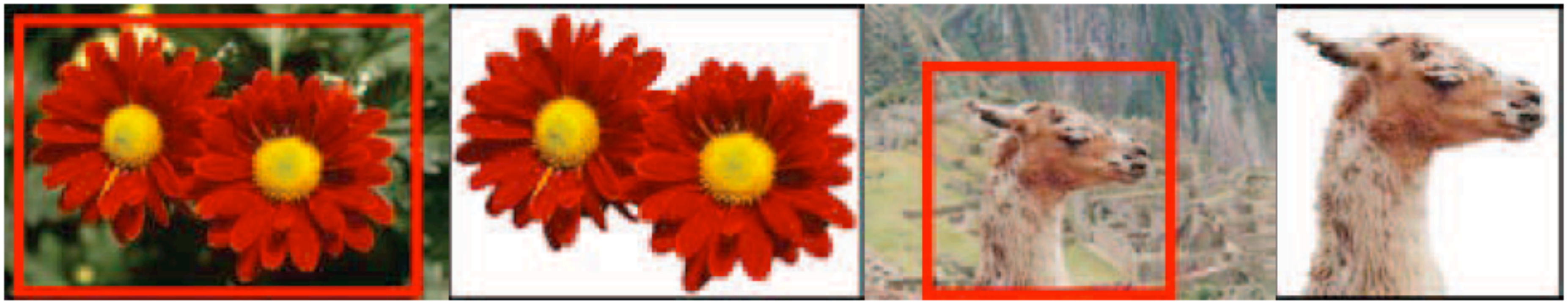


FIGURE 9.12: In a grabcut interface for interactive segmentation, a user marks a box around the object of interest; foreground and background models are then inferred by a clustering method, and the object is segmented. If this segmentation isn't satisfactory, the user has the option of painting foreground and background strokes on pixels to help guide the model. *This figure was originally published as Figure 1 of "GrabCut Interactive Foreground Extraction using Iterated Graph Cuts" by C. Rother, V. Kolmogorov, and A. Blake, Proc. ACM SIGGRAPH, 2004 © ACM, 2004.*

Matting

- Pixels very often are a weighted mixture of colors
    - ie p= $\alpha \boldsymbol{f} + (1 - \alpha)\boldsymbol{b}$
    - we see p, must estimate f, b and alpha
- Why?
    - hair, fluff, etc. on boundary
- How?
    - assume
        - alpha is smooth
        - f, b are
            - near constant
            - from foreground (resp. background) model

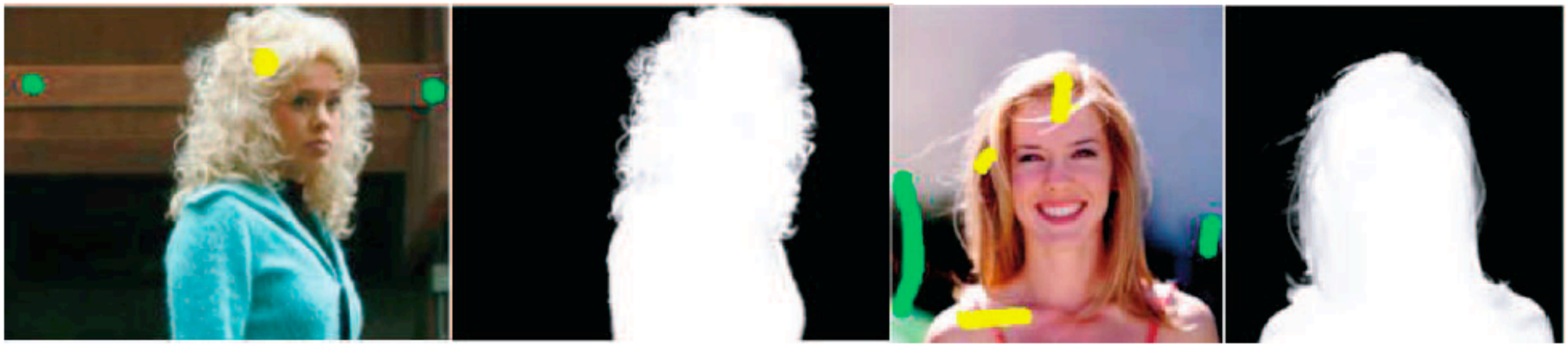


FIGURE 9.13: Matting methods produce a real-valued mask (rather than a foreground-background mask) to try and compensate for effects in hair, at occluding boundaries, and so on, where some pixels consist of an average of foreground and background values. The matte is bright for foreground pixels and dark for background pixels; for some pixels in the hair, it is gray, meaning that when the foreground is transferred to a new image, these pixels should become a weighted sum of foreground and background. The gray value indicates the weight. *This figure was originally published as Figure 6 of "Spectral Matting," by A. Levin, A. Rav-Acha, and D. Lischinski, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2008 © IEEE, 2008.*

Image segmentation with k-means

- Represent each pixel in the image with a vector
  - intensity
  - color
  - color and location
  - color, location, other stuff (texture)
- Choose distance weights
  - ie is color more important than location?
- Apply k-means
- Pixels belong to the segment corresponding to centers
- Different representations yield different segmentations

This is the key step

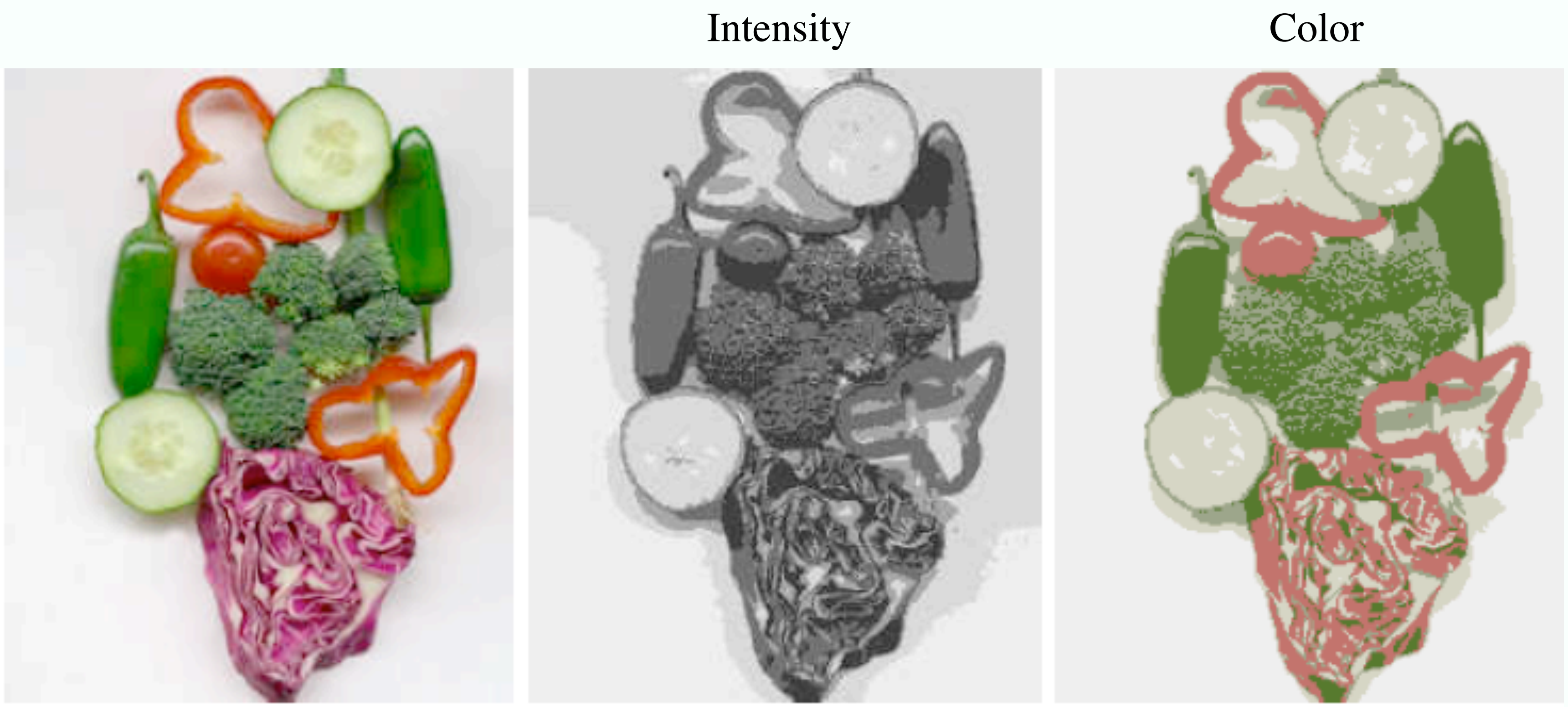


FIGURE 9.17: On the **left**, an image of mixed vegetables, which is segmented using k-means to produce the images at **center** and on the **right**. We have replaced each pixel with the mean value of its cluster; the result is somewhat like an adaptive requantization, as one would expect. In the center, a segmentation obtained using only the intensity information. At the right, a segmentation obtained using color information. Each segmentation assumes five clusters.

Color, k=11

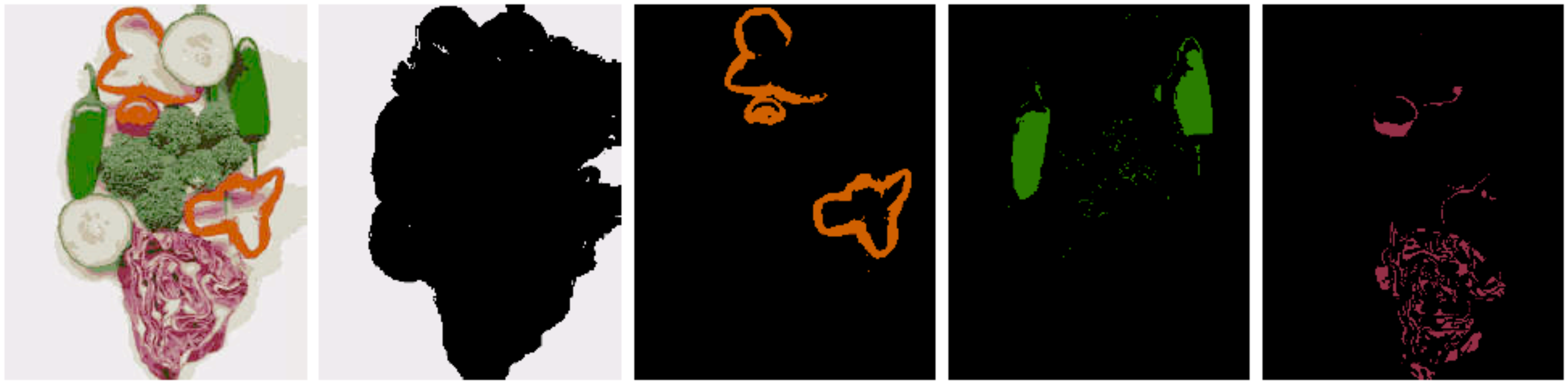


FIGURE 9.18: Here we show the image of vegetables segmented with k-means, assuming a set of 11 components. The **left** figure shows all segments shown together, with the mean value in place of the original image values. The other figures show four of the segments. Note that this approach leads to a set of segments that are not necessarily connected. For this image, some segments are actually quite closely associated with objects, but one segment may represent many objects (the peppers); others are largely meaningless. The absence of a texture measure creates serious difficulties, as the many different segments resulting from the slice of red cabbage indicate.

Color and position, k=11

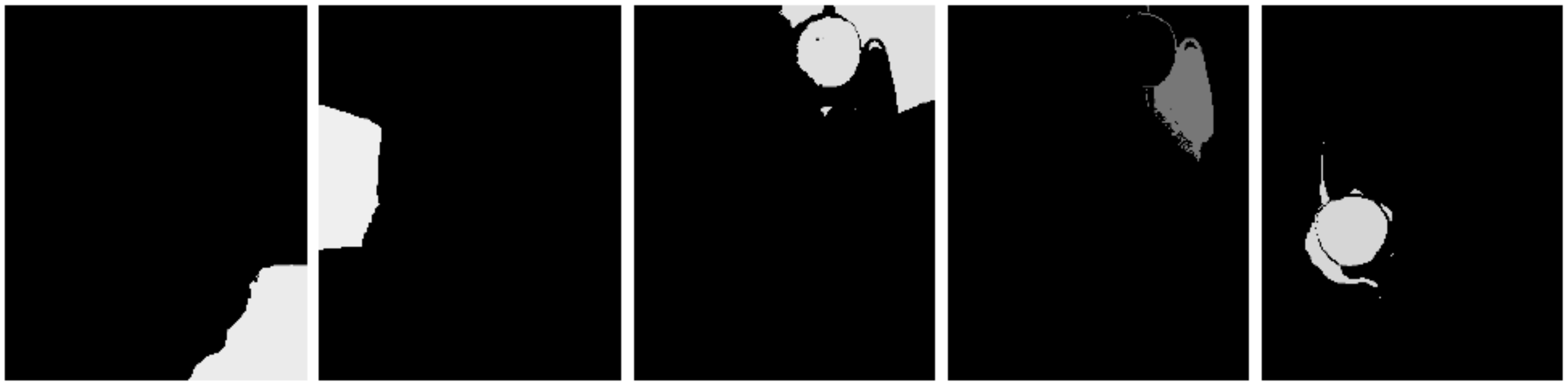


FIGURE 9.19: Five of the segments obtained by segmenting the image of vegetables with a k-means segmenter that uses position as part of the feature vector describing a pixel, now using 20 segments rather than 11. Note that the large background regions that should be coherent have been broken up because points got too far from the center. The individual peppers are now better separated, but the red cabbage is still broken up because there is no texture measure.

Interactive segmentation with k-means

- Simple algorithm
    - current alg.s somewhat better than this
- Take all known foreground pixels, do k-means
- Take all known background pixels, do k-means
- For all pixels
    - find closest foreground/background center
    - if closest center is foreground, pixel is foreground
    - if closest center is background, pixel is background
    -

Matting with k-means

- Concept algorithm
    - all matters are shaky right now
    - there are more complicated matters that do better than this
- Find closest foreground, background to pixel
    - but these three are NOT on a straight line
    - fit a line
        - this gives F (point on line closest to foreground)
        - B (point on line closest to background)
        - alpha (mixture weight)

Graph based segmentation

- Build a graph out of image
  - Typically
    - each pixel is a vertex
    - edge between neighboring pixels
    - edges are weighted by similarity of pixels
      - distance between representation vectors (intensity, color, etc.)
- Cut this graph into pieces in various ways

9.4.1 Terminology and Facts for Graphs

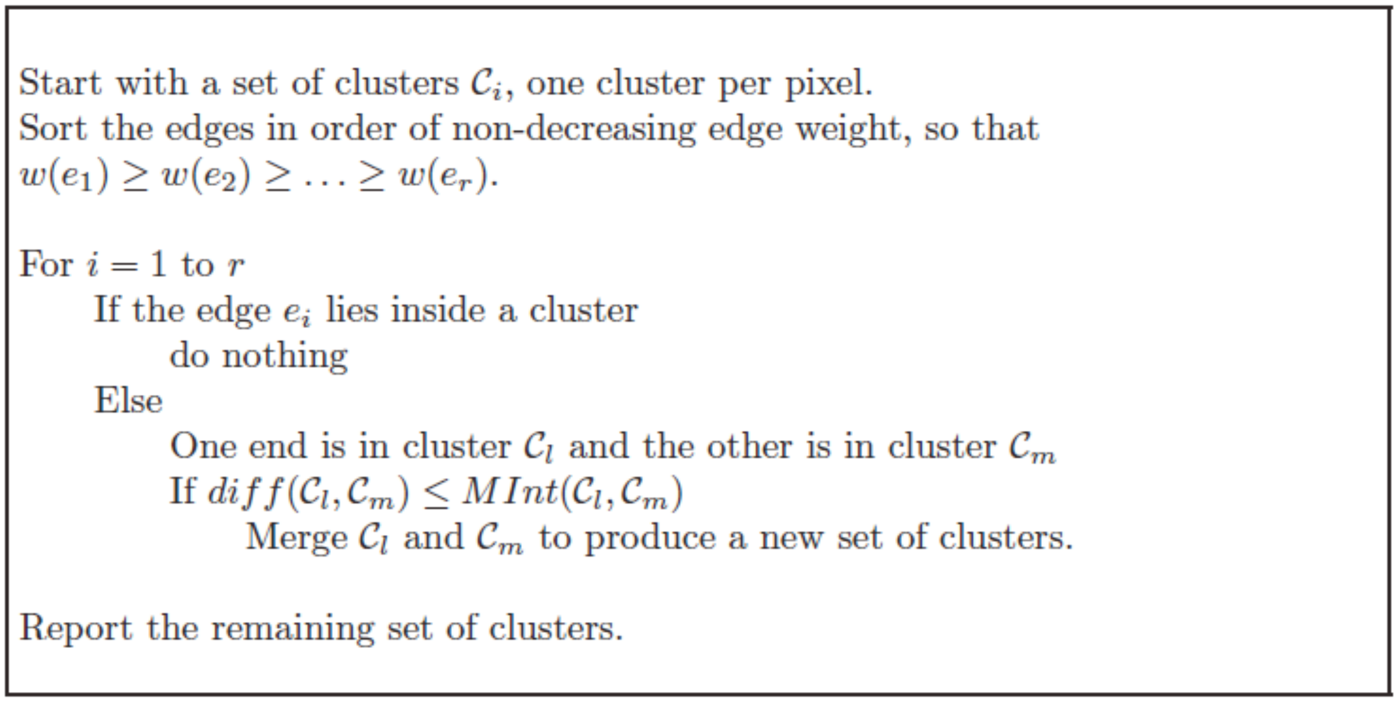
We review terminology here very briefly, as it's quite easy to forget.

- A *graph* is a set of vertices V and edges E that connect various pairs of vertices. A graph can be written $G = \{V, E\}$. Each edge can be represented by a pair of vertices—that is, $E \subset V \times V$. Graphs are often drawn as a set of points with curves connecting the points.
- The *degree* of a vertex is the number of edges incident on that vertex.

- A *directed graph* is one in which edges (a, b) and (b, a) are distinct; such a graph is drawn with arrowheads indicating which direction is intended.
- An *undirected graph* is one in which no distinction is drawn between edges (a, b) and (b, a).
- A *weighted graph* is one in which a weight is associated with each edge.
- Two edges are *consecutive* if they have a vertex in common.
- A *path* is a sequence of consecutive edges.
- A *circuit* is a path which ends at the vertex at which it begins.
- A *self-loop* is an edge that has the same vertex at each end; self-loops don't occur in our applications.
- Two vertices are said to be *connected* when there is a sequence of edges starting at the one and ending at the other; if the graph is directed, then the arrows in this sequence must point the right way.
- A *connected graph* is one where every pair of vertices is connected.
- A *tree* is a connected graph with no circuits.

- Given a connected graph $G = \{V, E\}$, a *spanning tree* is a tree with vertices V and edges a subset of E. By our definition, trees are connected, so a spanning tree is connected.
- Every graph consists of a disjoint set of *connected components*—that is, $G = \{V_1 \cup V_2 \ldots V_n, E_1 \cup E_2 \ldots E_n\}$, where $\{V_i, E_i\}$ are all connected graphs and there is no edge in E that connects an element of V_i with one of V_j for $i \neq j$.
- A *forest* is a graph whose connected components are trees.

Graph based agglomerative clustering



```
Start with a set of clusters $\mathcal{C}_i$, one cluster per pixel.
Sort the edges in order of non-decreasing edge weight, so that
$w(e_1) \geq w(e_2) \geq \ldots \geq w(e_r)$.

For $i = 1$ to $r$
    If the edge $e_i$ lies inside a cluster
        do nothing
    Else
        One end is in cluster $\mathcal{C}_l$ and the other is in cluster $\mathcal{C}_m$
        If $diff(\mathcal{C}_l, \mathcal{C}_m) \leq MInt(\mathcal{C}_l, \mathcal{C}_m)$
            Merge $\mathcal{C}_l$ and $\mathcal{C}_m$ to produce a new set of clusters.

Report the remaining set of clusters.
```

Algorithm 9.8: Agglomerative Clustering with Graphs.

We will start with every pixel forming a cluster, then merge clusters until there is no need to continue. To do this, we need some notion of the distance between two clusters. Each cluster is a component of the graph, formed from all the vertices (pixels) in the cluster, and all the edges that start and end inside the cluster. Then the difference between two components is the minimum weight edge connecting two components. Write $\mathcal{C}_1$, $\mathcal{C}_2$ for the two components, $\mathcal{E}$ for the edges, and $w(v_1, v_2)$ for the weight of the edge joining v_1 and v_2. Then, we have

$$\mathrm{diff}(\mathcal{C}_1, \mathcal{C}_2) = \min_{v_1 \in \mathcal{C}_1, v_2 \in \mathcal{C}_2, (v_1, v_2) \in \mathcal{E}} w(v1, v2).$$

It is also helpful to know how coherent a particular cluster is. This will help us stop clustering. We define the internal difference of a component to be the largest weight in the minimum spanning tree of the component. Write $M(\mathcal{C}) = \{V_{\mathcal{C}}, E_M\}$ for the minimum spanning tree of $\mathcal{C}$. Then, we have

$$\mathrm{int}(\mathcal{C}) = \max_{e \in M(\mathcal{C})} w(e).$$

Comparing the edge weight to the internal difference of the clusters requires some care, because in small clusters the internal distance might be zero (if there is only one vertex), or implausibly small. To deal with this, Felzenszwalb and Huttenlocher (2004) define a function of two clusters, MInt, as

$$\mathrm{MInt}(\mathcal{C}_1, \mathcal{C}_2) = \min(\mathrm{int}(\mathcal{C}_1) + \tau(\mathcal{C}_1), \mathrm{int}(\mathcal{C}_2) + \tau(\mathcal{C}_2))$$

where $\tau(\mathcal{C})$ is a term that biases the internal difference upward for small clusters; Felzenszwalb and Huttenlocher (2004) use $\tau(\mathcal{C}) = k / \mid \mathcal{C} \mid$, for k some constant parameter. This algorithm is notably fast and relatively accurate (Figure 9.22).

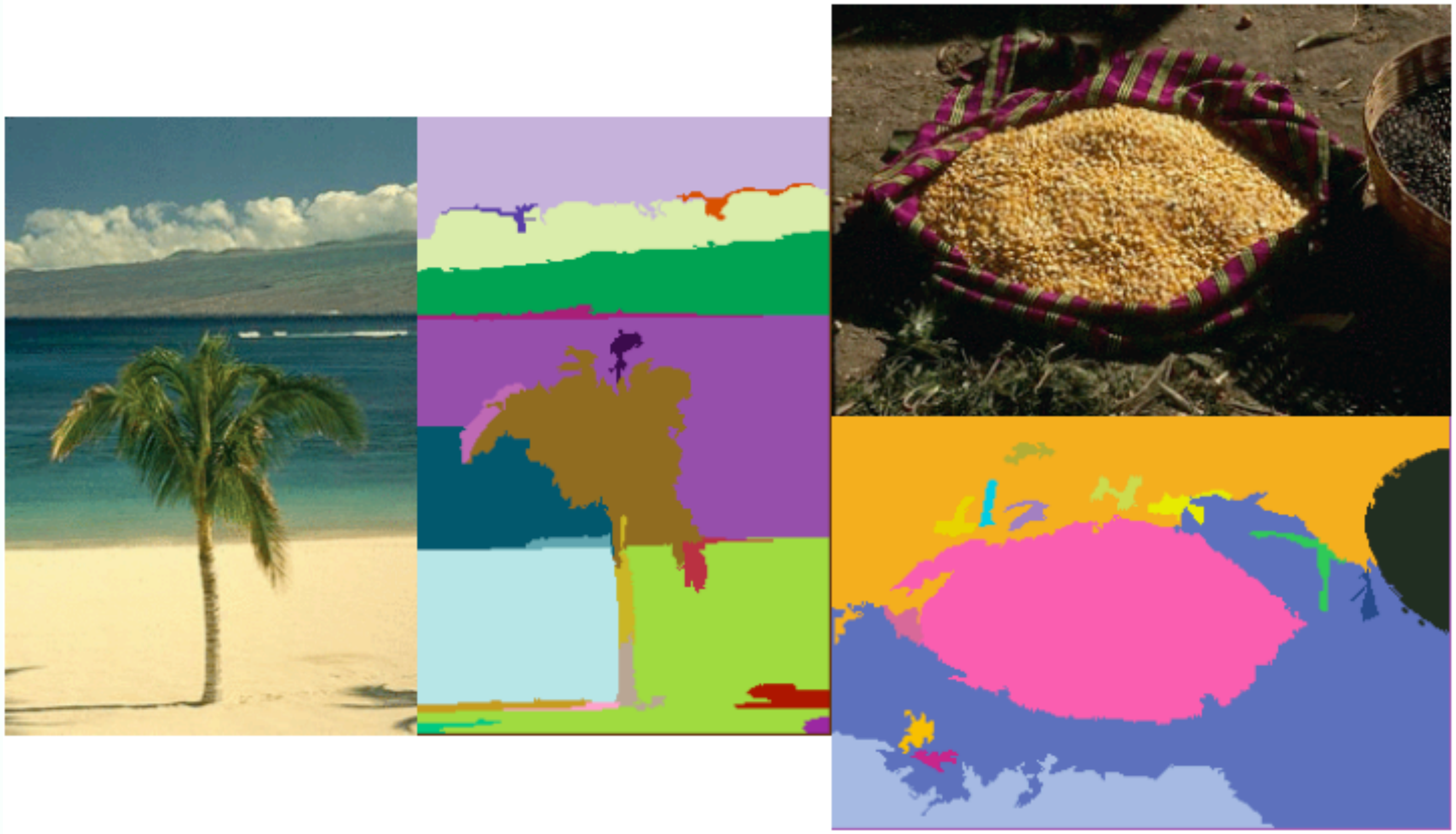


FIGURE 9.22: Images segmented using Algorithm 9.8, shown next to segments. Figures obtained from `http://people.cs.uchicago.edu/~pff/segment/`, by kind permission of Pedro Felzenszwalb.

Mean shift

- Idea:
  - clusters are places where data points tend to be close together
  - assume data are IID samples from probability distribution
    - (independent, identically distributed)
  - find local maxima in this probability distribution
- Problem:
  - don’t know the distribution, must build a model

Building a model of the PDF

- Place a small smooth bump on top of each data point
  - how big? adjust to get best model
- Add bumps, normalize
- When data are clustered, function is big

Small, smooth bumps=kernels

- write x for data, h for a scale parameter, d is dimension
- we'll use a Gaussian bump (there are others)

$$K(x;h) = \frac{(2\pi)^{(-d/2)}}{h^d} \exp\left(-\frac{1}{2}\frac{\|x\|^2}{h}\right)$$

- Model of data density becomes

$$f(x) = \left(\frac{1}{n}\right)\sum_{i=1}^{n} K\left(x_i - x; h\right)$$

(Assume we know h at the moment, we'll get to it)

Small smooth bumps, again

$$K(x;h) = \frac{(2\pi)^{(-d/2)}}{h^d} \exp\left(-\frac{1}{2}\frac{\| x \|^2}{h}\right)$$

$$f(x) = \left(\frac{1}{n}\right)\sum_{i=1}^{n} K\left(x_i - x; h\right)$$

We can simplify notation by writing $k(u) = \exp\left(-\frac{1}{2}u\right)$ (this is called the *kernel profile*) and $C = \frac{(2\pi)^{(-d/2)}}{nh^d}$, so that

$$f(x) = C\sum_{i=1}^{n} k\left(\| \frac{x - x_i}{h} \|^2\right) \tag{9.1}$$

Finding a maximum

- We want to find a maximum in f
  - i.e. gradient = 0

$$\begin{aligned}
\nabla f(\boldsymbol{x}) \mid_{\boldsymbol{x}=\boldsymbol{y}} &= 0 \\
&= C \sum_i \nabla k\left(\left\| \frac{\boldsymbol{x}_i - \boldsymbol{y}}{h} \right\|^2\right) \\
&= C \frac{2}{h} \sum_i [\boldsymbol{x}_i - \boldsymbol{y}] \left[g\left(\left\| \frac{\boldsymbol{x}_i - \boldsymbol{y}}{h} \right\|^2\right) \right] \\
&= C \frac{2}{h} \left[\frac{\sum_i \boldsymbol{x}_i g\left(\left\| \frac{\boldsymbol{x}_i - \boldsymbol{y}}{h} \right\|^2\right)}{\sum_i g\left(\left\| \frac{\boldsymbol{x}_i - \boldsymbol{y}}{h} \right\|^2\right)} - \boldsymbol{y} \right] \times \left[\sum_i g\left(\left\| \frac{\boldsymbol{x}_i - \boldsymbol{y}}{h} \right\|^2\right) \right].
\end{aligned}$$

We expect that $\sum_i g\left(\left\| \frac{\boldsymbol{x}_i - \boldsymbol{y}}{h} \right\|^2\right)$ is nonzero, so that the maximum occurs when

$$\left[\frac{\sum_i \boldsymbol{x}_i g\left(\left\| \frac{\boldsymbol{x}_i - \boldsymbol{y}}{h} \right\|^2\right)}{\sum_i g\left(\left\| \frac{\boldsymbol{x}_i - \boldsymbol{y}}{h} \right\|^2\right)} - \boldsymbol{y} \right] = 0,$$

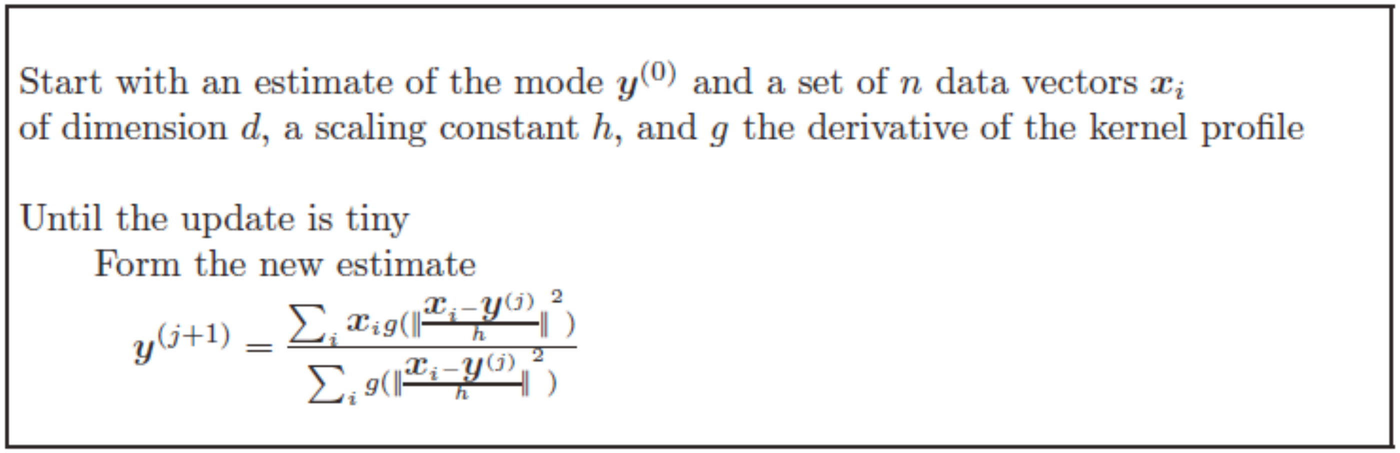
Finding a maximum - II

$$\left[\frac{\sum_i \boldsymbol{x}_i g(\| \frac{\boldsymbol{x}_i - \boldsymbol{y}}{h} \|^2)}{\sum_i g(\| \frac{\boldsymbol{x}_i - \boldsymbol{y}}{h} \|^2)} - \boldsymbol{y}\right] = 0,$$

Means that:

$$\boldsymbol{y} = \frac{\sum_i \boldsymbol{x}_i g(\| \frac{\boldsymbol{x}_i - \boldsymbol{y}}{h} \|^2)}{\sum_i g(\| \frac{\boldsymbol{x}_i - \boldsymbol{y}}{h} \|^2)}.$$

Finding a maximum - III



```
Start with an estimate of the mode $y^{(0)}$ and a set of $n$ data vectors $x_i$
of dimension $d$, a scaling constant $h$, and $g$ the derivative of the kernel profile

Until the update is tiny
    Form the new estimate
    $y^{(j+1)} = \frac{\sum_i x_i g(\|\frac{x_i - y^{(j)}}{h}\|^2)}{\sum_i g(\|\frac{x_i - y^{(j)}}{h}\|^2)}$
```

Algorithm 9.5: Finding a Mode with Mean Shift.

Segmenting images with mean shift

- I: apply mean shift to pixel representations
  - we expect many, quite tightly clustered, local minima
  - balancing color distance and position distance differently changes results
- II: apply k-means to local minima
  - too many to be segments
  - but tend to be much better clustered than pixel representations
  - pixel belongs to segment whose number is number of local center

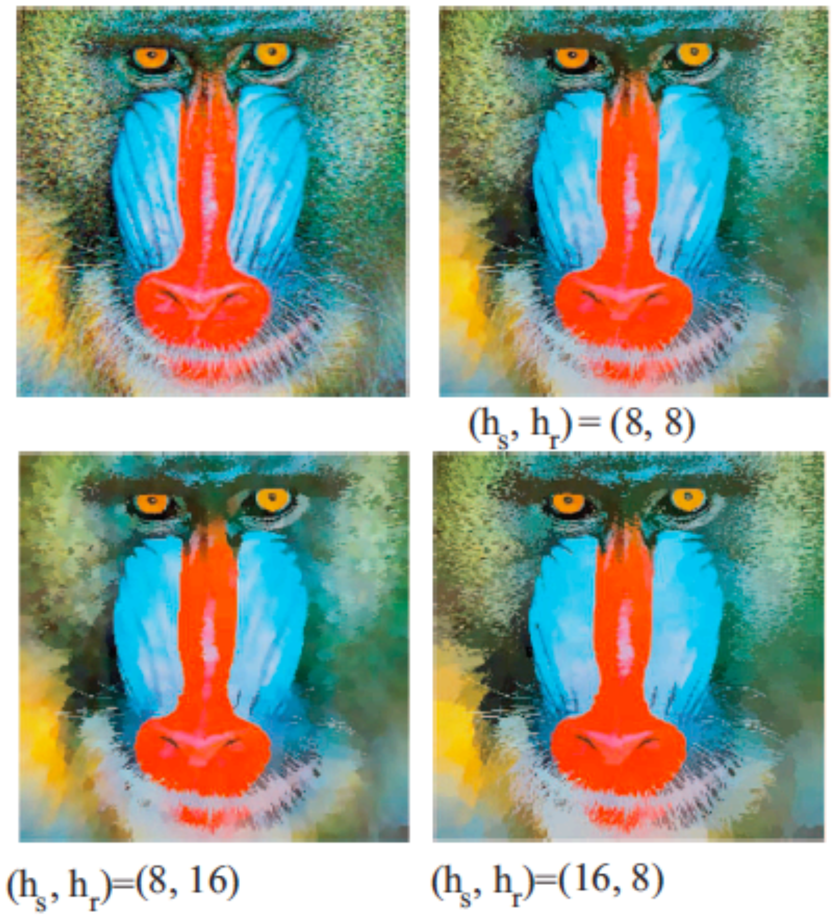


FIGURE 9.20: An image (**top left**) and mean shift modes obtained with different clustering scales for space h_s and appearance h_r. If h_s is small, the method must produce clusters that are relatively small and compact spatially because the kernel function smoothes over a relatively small radius and so will allow many distinct modes. If h_r is small, the clusters are compact in appearance; this means that small h_s and large h_r will produce small, blobby clusters that could span a range of appearances, whereas large h_s and small h_r will tend toward spatially complex and extended clusters with a small range of appearances. Cluster boundaries will try harder to follow level curves of intensity. *This figure was originally published as Figure 5 of "Mean Shift: A Robust Approach Toward Feature Space Analysis," by D. Comaniciu and P. Meer, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002 © IEEE, 2002.*

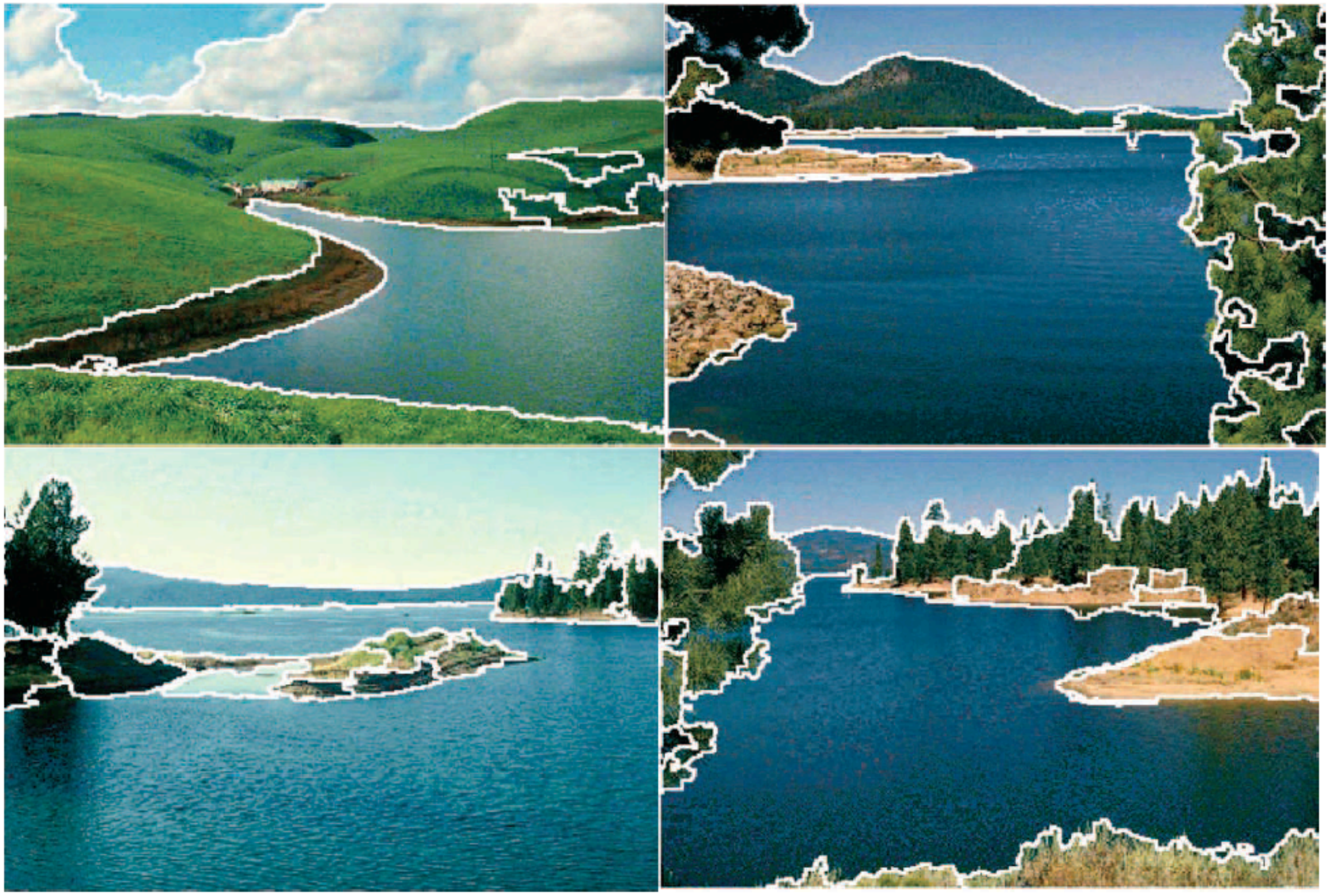


FIGURE 9.21: Segmentations of images obtained using the mean shift algorithm. *This figure was originally published as Figure 10 of "Mean Shift: A Robust Approach Toward Feature Space Analysis," by D. Comaniciu and P. Meer, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002 © IEEE, 2002.*