Image Segmentation

Shree K. Nayar

Monograph: FPCV-5-2

Module: Reconstruction II

Series: First Principles of Computer Vision

Computer Science, Columbia University

May, 2025

FPCV Channel

FPCV Website

Image segmentation is a fundamental and important problem in computer vision. We are given an image and want to partition this image into regions, or segments. We want these segments to be meaningful in some sense so that they prove helpful in solving the vision problem at hand.

Image Segmentation

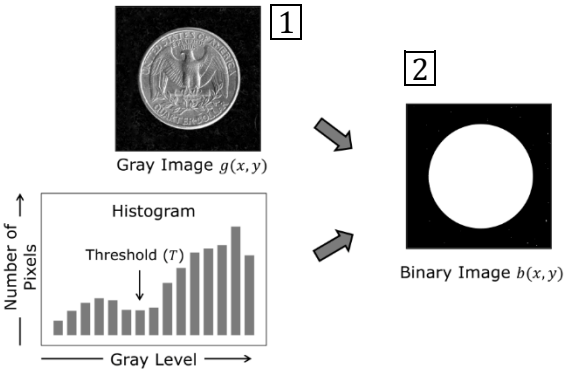
Shree K. Nayar
Columbia University

Topic: Object Tracking, Module: Perception
First Principles of Computer Vision

1

We have already seen some simple approaches to image segmentation. In the context of binary images, we saw how we can segment a simple image, such as this one 1, with an object on a uniform background. We can compute a histogram of the image, find an appropriate threshold, and then threshold the image to produce a binary image 2. This binary image is the segmentation we are looking for in this case.

Binary Segmentation



REVIEW 2

We also talked about active contours or "snakes." We can initialize a contour (dotted) as shown here, and then automatically deform it to latch onto the boundary of the object. Once it has found the boundary, the image has been segmented. In this setting, the user guides the process, in that, they must initialize the active contour. This is a very useful tool in many different domains but does not solve the general segmentation problem, which must be fully automatic and handle complex images.

Active Contours

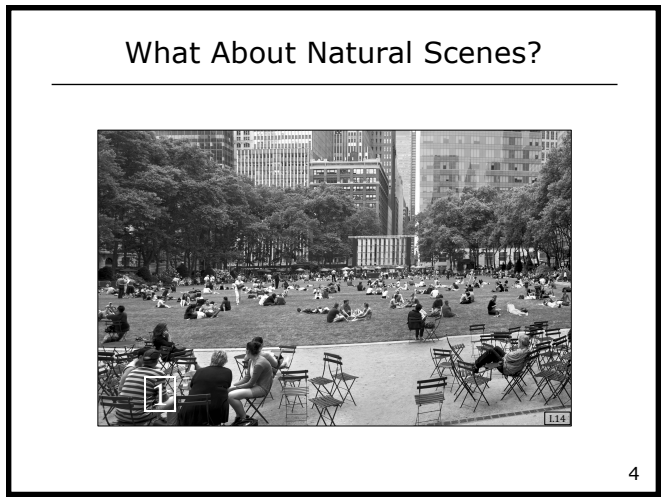
Given: Approximate boundary (contour) around the object.

Active Contours: Evolves (move) the contour to fit exact object boundary.

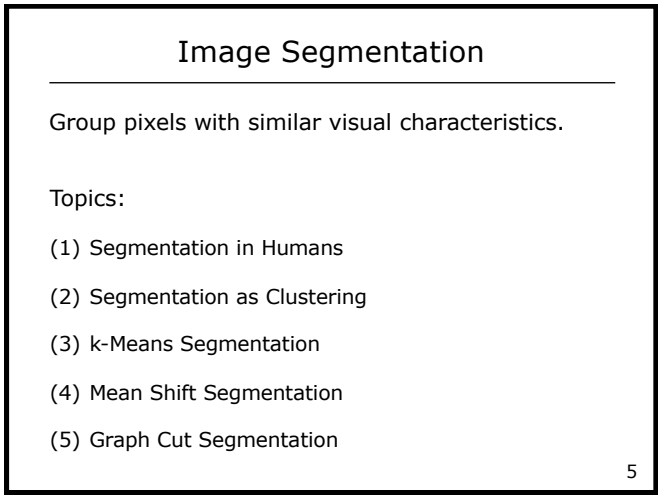
Image

REVIEW [Kass 1987] 3

Segmentation is, in general, a hard problem. Consider this complex image. How does one segment it into meaningful regions? If we look at, for instance, this man sitting [1], is the cap that he is wearing a part of him or a separate object? The answer really depends on the task that we are trying to solve. In short, the quality of a segmentation depends on the context and the application.



In this lecture, we are going to develop some simple methods for image segmentation. Our approach is going to be to group pixels together in the image that have similar visual attributes, or characteristics. First, we will look at how we, humans, seem to perform segmentation. What kinds of principles do we use to guide ourselves during the process of segmentation? Interestingly, if we take a single natural image and ask two people to come up with a meaningful segmentation of the image, we may get two dramatically different results. In other words, segmentation is highly subjective. That is because it is an ill-defined problem.



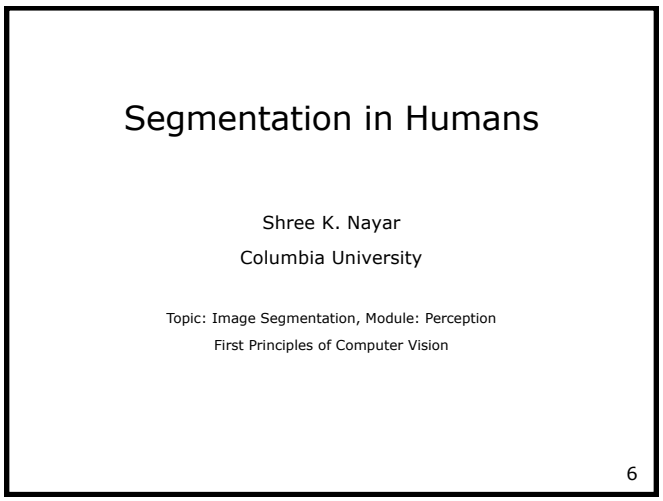
We are going to take a simple approach and say that segmentation is grouping together pixels that have similar visual attributes. In other words, we treat it as a clustering problem. At each pixel in the image, we have a feature vector that describes the pixel, which can be seen as a point in a high-dimensional feature space. In this space, we are going to cluster points together, and each cluster represents a meaningful segment.

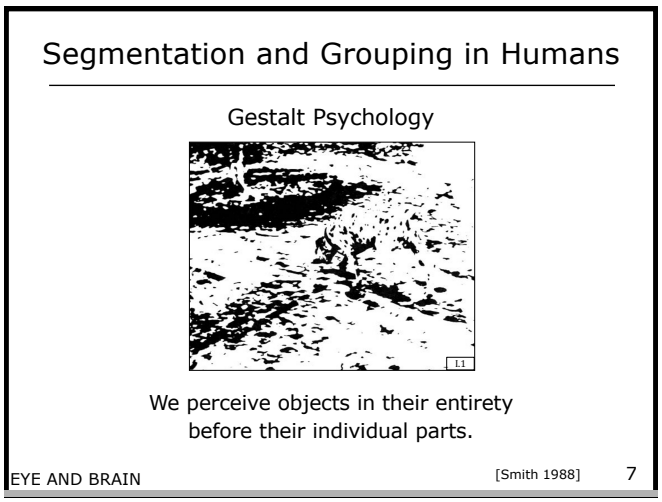
The next question is what clustering algorithm should be used. The first algorithm we will discuss is called k-means segmentation. In this case, we specify the number of regions (k) that we are looking. The algorithm then finds k clusters in feature space, which correspond to the k segments.

A more sophisticated and reliable algorithm is called “mean shift.” The advantage of mean shift over k-means is that we do not need to define the number of segments that we are looking for. Mean shift is

based on hill climbing. It is able to find how many hills exist in a distribution in feature space, and assign each image pixel to one of those hills. In a sense, it performs clustering without knowing the number of segments in the image.

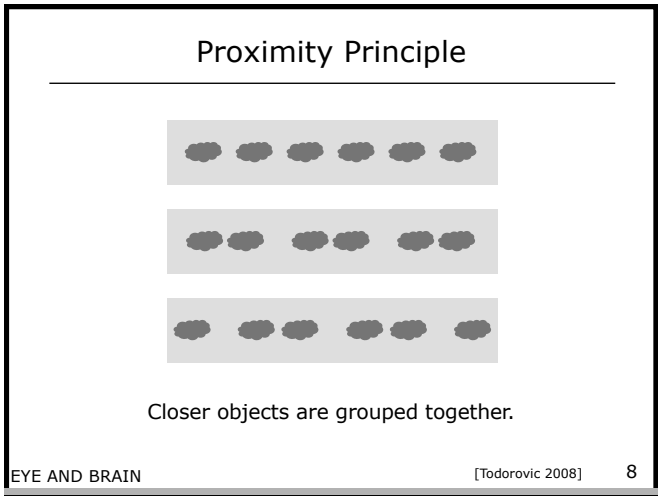
Finally, we describe a graph-based approach to segmentation. This is called the “graph cut” method. We first represent the image as a graph, where the pixels are the vertices and the similarities between pixels are weights of edges between the vertices. Then, we can find the segments of the image by finding cuts in the graph using a cost function.



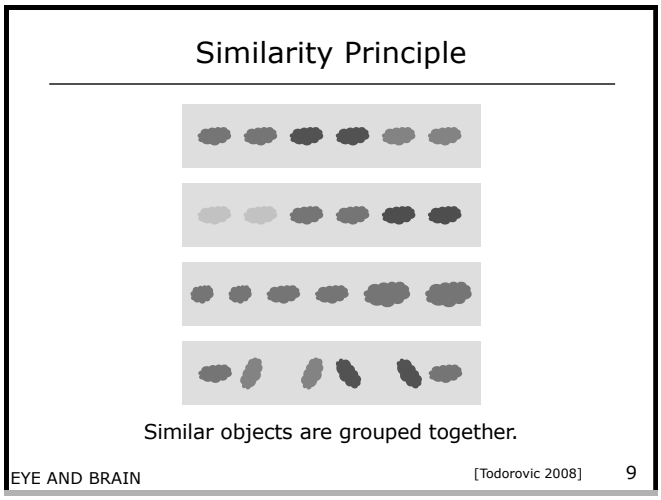


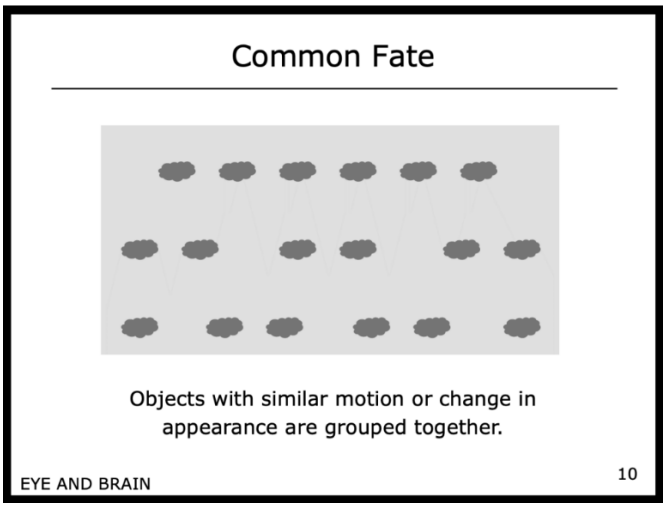
We know that segmentation is a hard problem. How do we humans accomplish it? If we stare at the image on the right, we will ultimately perceive a dog close to the middle of the image. Once we perceive the dog, we are then able to identify its various parts: the feet, the tail, the head, and so on. This is referred to as the Gestalt theory of visual perception. Gestalt is German for form or shape, and the theory was developed by a group of German scientists. This theory implies that we perceive an object as a group, and then identify its individual parts, or subgroups.

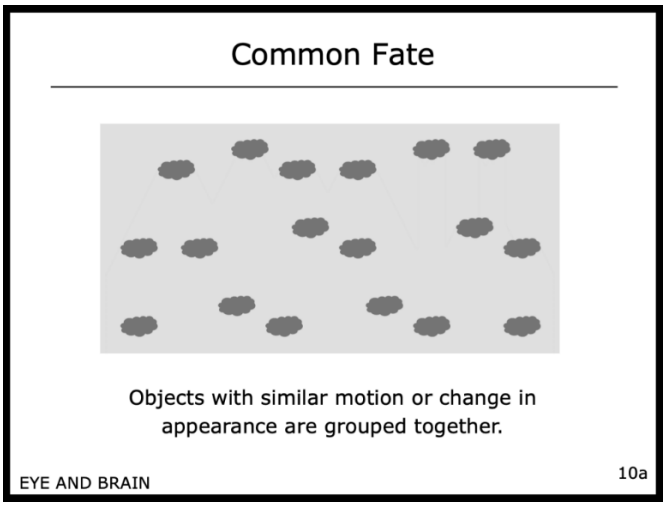
Todorovic has proposed a collection of rules that humans use to perform segmentation. Let us start with the notion of proximity. If we look at the first row, we can see the entire collection of six elements as being a group. If we change the distances between the elements, as in the second row, we still see all the elements as one group, but now we see the emergence of three subgroups. In the case of the arrangement in the third row we perceive four subgroups. Proximity therefore plays an important role in the perception of subgroups.



Next, we have the principle of similarity. In this case, objects with the same appearance tend to be grouped together. In each of the first three images, we see one group with three subgroups, where each subgroup has elements with similar brightness or size. In the fourth image, we have similarity and proximity competing with each other. In this case, proximity appears to win as the three subgroups (pairs of elements) we perceive are based on the distance between elements and not their brightnesses or orientations.

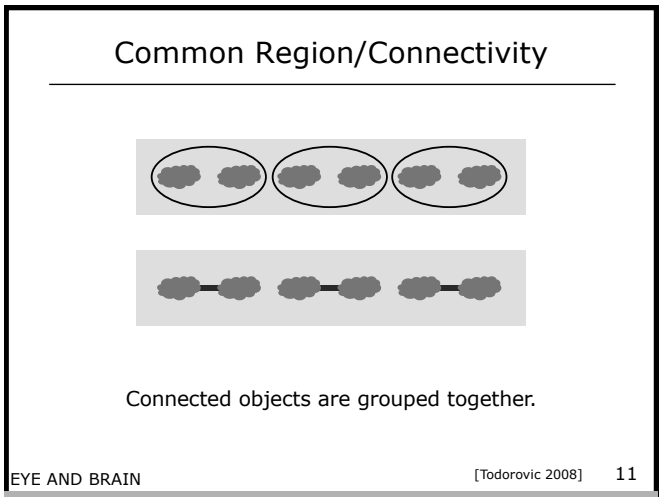




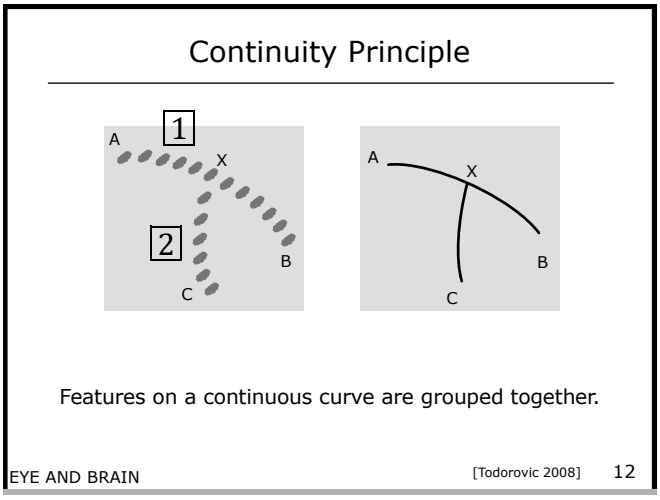


Next, we have the principle of common fate. On the left we see several elements with different distances between them. If any set of these elements move together (see online video lecture), we refer to them as having the same fate, and perceive them as belonging to the same subgroup. This is true even when the moving elements are not close to each other.

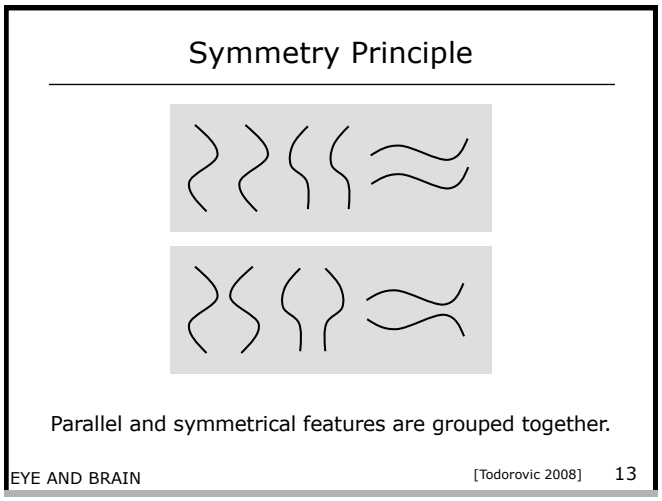
There is also the notion of a common region or connectivity. In the image on the top, the ellipses drawn around the pairs of elements make each pair appear as a subgroup, even though the individual elements are uniformly spaced. In the bottom image, the elements are connected by links, which makes each pair of connected elements appear as a subgroup, even though the elements are uniformly spaced.



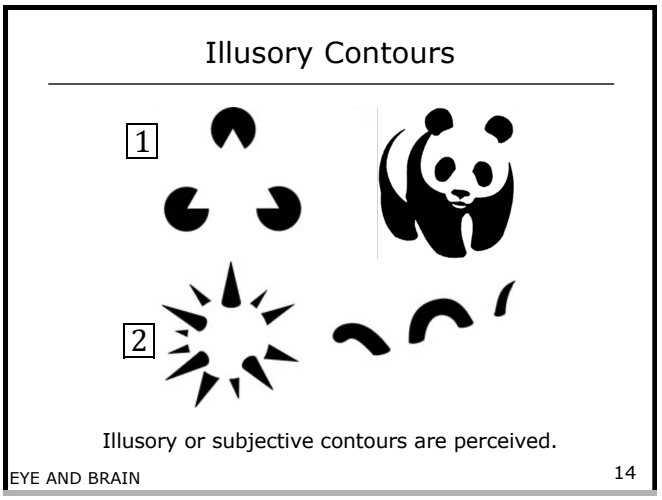
Shown here is the continuity principle, which is related to the alignment or the orientation between the elements. In this example, the group of elements 1 appear to lie on a single smooth curve and hence are grouped together as a single subgroup. The same is true for the group of elements 2. In short, features that lie on a continuous curve tend to be grouped together as a subgroup.



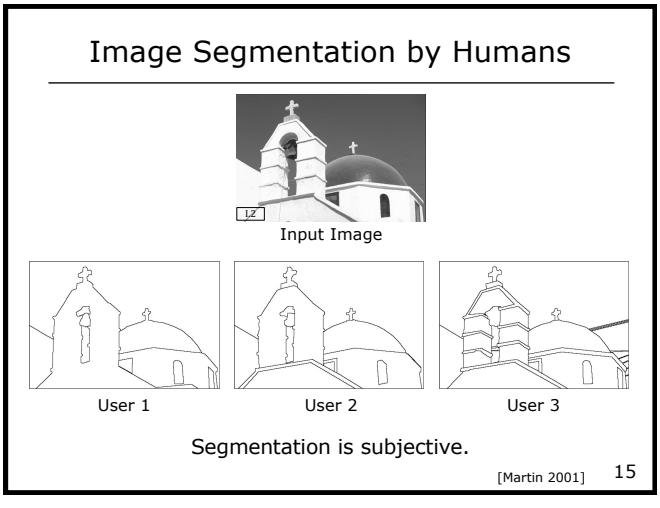
Next, we have the symmetry principle. Symmetry is a very strong cue because it is very unlikely in a scene for two things to be perfectly aligned in some respect and not belong to the same group. In each of the two images seen here we see three subgroups, where the elements in each subgroup are perfectly symmetric. The symmetry is translational in the top image and reflectional in the bottom one.



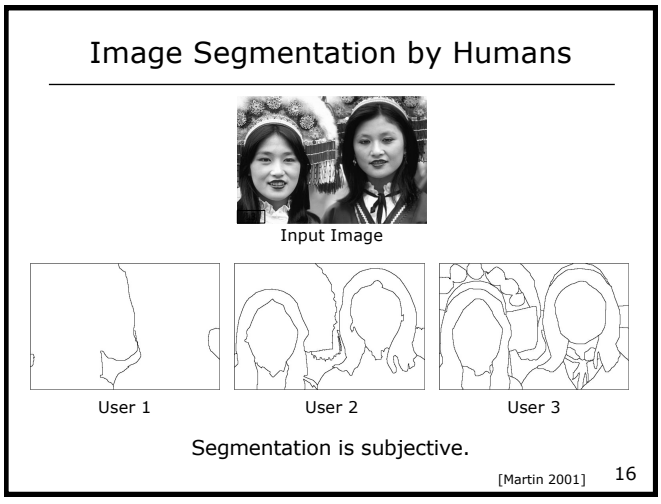
Finally, there are illusions related to segmentation. In our first lecture, we discussed the Kanizsa triangle 1, where we perceive a bright triangle in between the three "pac-man" like fragmented discs. We know that the attributes of the pixels inside and outside the triangle are identical, and yet we perceive the triangle. That is because the discs are perfectly aligned and the triangle is a very familiar shape to us. Similarly, in the case of 2, we perceive a sphere. The perception of such illusory, or subjective, contours makes segmentation an even more complex task for us humans.



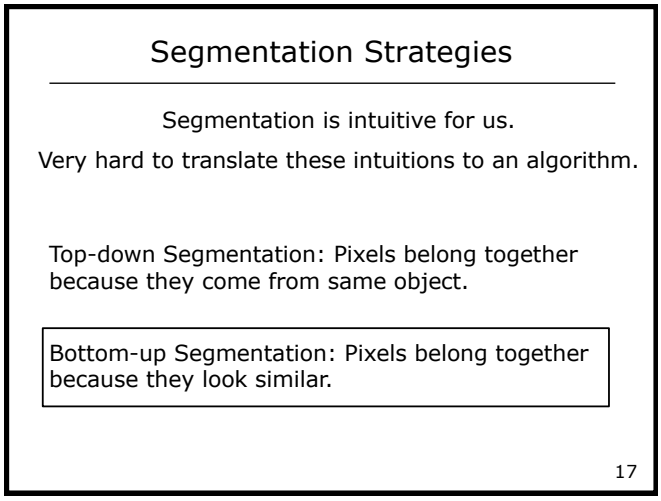
The final point worth making regarding image segmentation by humans is that it is highly subjective for. In this work done by Martin et al., the image on the top was given to three different people who were asked to come up with meaningful segments. The first two answers look quite similar, but the third has many more small regions that were deemed to be meaningful.



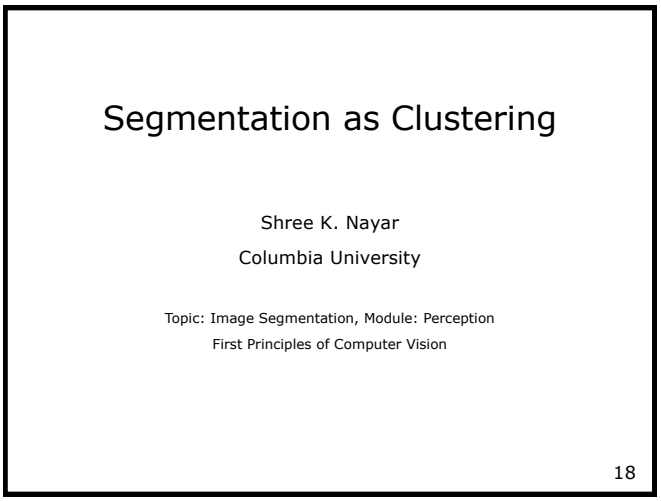
In this example, one person simply sees the image as having two segments (left), another sees the hair and face of each person as being additional segments (middle), and the third person sees the decorative pieces in the background as being separate segments as well. These experiments demonstrate that image segmentation is highly subjective.



We have seen that we humans use our intuitions to segment images. Since we do not fully understand these intuitions, and we don't necessarily agree on them, it is hard to translate them into a single algorithm. We must therefore develop an algorithm from scratch. In doing so, we could use one of two approaches. The first is what we will call top-down segmentation, which is more in line with the Gestalt approach where pixels belong together because they come from the same object. In this approach we need to first identify the object and then look for its parts. The second approach is bottom-up segmentation. In this case, pixels belong to the same group because they look similar. Since this approach is easier to define, it is the one we will take.



We know that an image is an array of pixels, and at each pixel we have some visual attributes or characteristics. We can view these attributes as a feature vector and map the image to a distribution of points in a high-dimensional feature space. Our goal is to find clusters in this feature space, where each cluster corresponds to image pixels that belong to the same segment. In short, segmentation can be posed as a clustering problem.



Let us take a look at some of the visual characteristics that we can either directly measure or compute at a pixel. In terms of measurements, we have the brightness and color (red, green, and blue) of the pixel. We also have its spatial coordinates, x and y. We could also have depth, either measured using an active illumination method such as time of flight, or computed using stereo or depth from defocus. We also know how to compute optical flow (motion) at each pixel. In addition, a descriptor of the local texture can be computed from the image. In the coming years, we expect cameras to also be able to sense material properties at each pixel. All of the measured or computed attributes at each pixel can be seen as a feature vector.

Visual Characteristics of Pixels

Visual similarity can be based on:

- Brightness
- Color
- Position
- Depth
- Motion
- Texture
- Material

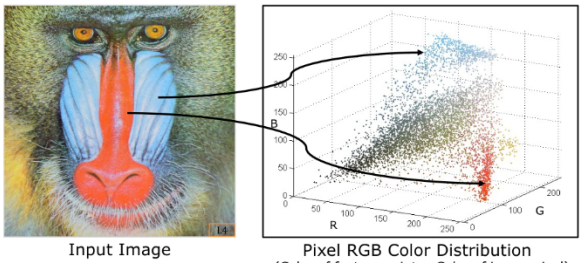
⋮

19

We can therefore map each pixel to a point in a high-dimensional feature space. In the example shown here, the feature vector is simply the color of the pixel. To this feature vector, we could add other parameters such as the x and y coordinates of the pixel and the depth of the corresponding scene point. Irrespective of the attributes used, the image can be mapped to a distribution in a Euclidean space, which is essentially a generalization of the familiar three-dimensional Cartesian space to higher dimensions.

Pixels in Euclidean Space

Euclidean Space: Generalization of 3D Cartesian space to higher dimensions.



Input Image

Pixel RGB Color Distribution
(Color of feature point ≡ Color of image pixel)

Pixels as feature vector: $[R, G, B, x, y, d, \ldots]$
(Color, Location, Depth,...)

20

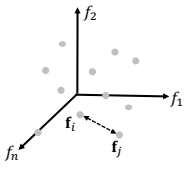
Now, to perform segmentation, what we need is a definition for the similarity between pixels. Let i and j be two pixels and let f_i and f_j be their feature vectors, which are simply two points in feature space. One way to measure the similarity S between these two points is by computing the L^2 distance them. The smaller the value of S, the greater the similarity between the pixels.

Pixel Similarity

Let i and j be two pixels whose features are $\mathbf{f}_i$ and $\mathbf{f}_j$.

$\mathcal{L}^2$ Distance between $\mathbf{f}_i$ and $\mathbf{f}_j$:

$$S(\mathbf{f}_i, \mathbf{f}_j) = \sqrt{\sum_k (f_{ik} - f_{jk})^2}$$

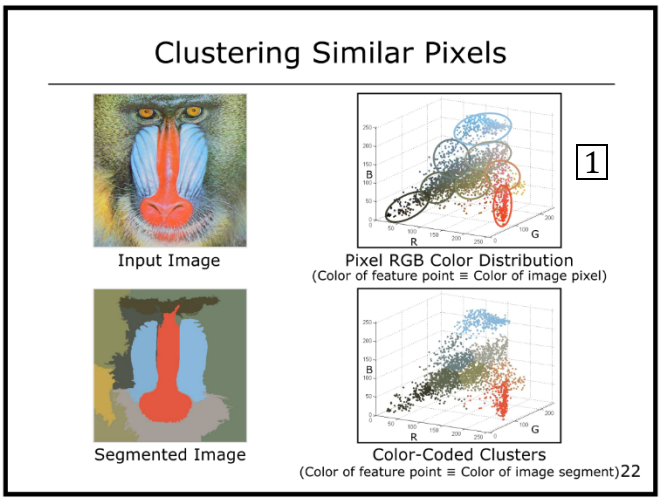


$[R, G, B, x, y, \ldots]$-space

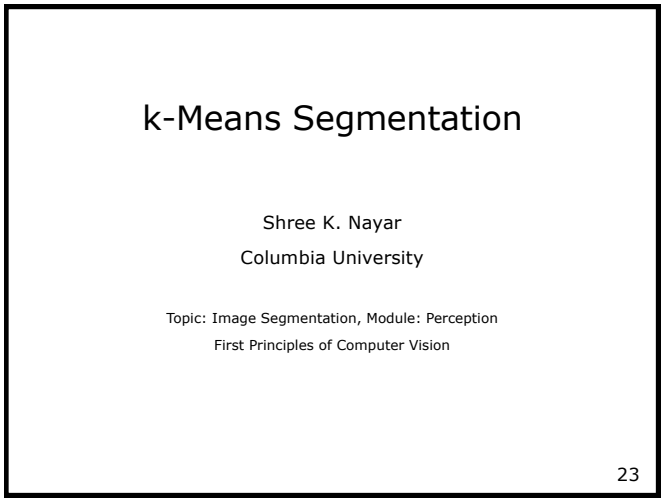
Smaller the Distance, Greater the Similarity

21

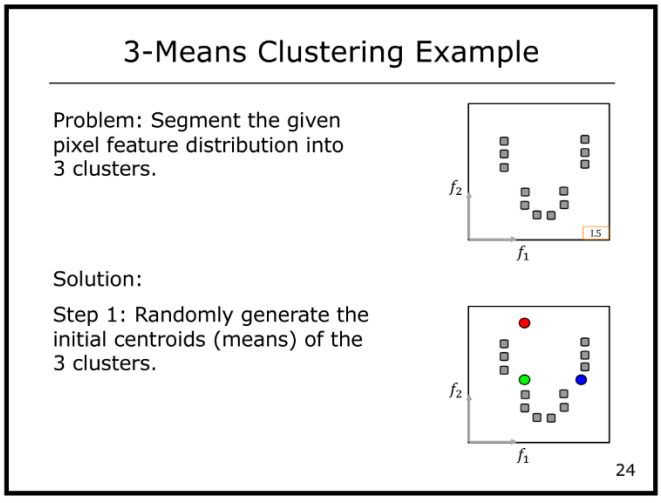
We are now able to view segmentation as essentially a clustering problem. Here we see an input image and its color distribution. We want to use our similarity metric to group points together into clusters. If we are able to develop such a clustering algorithm, we may get the 8 clusters shown by the ellipses in [1]. Here, all the points that belong to a cluster are given the same color, which is the average of all the colors in the cluster. By coloring each pixel with its cluster color, we get the segmented image shown at the bottom. This is the kind of result that we are looking for.



Our first algorithm for clustering is called k-means. The idea here is straightforward. We are looking for k segments in the image, or k clusters in features space. Each point is assigned to the cluster whose mean is closest to it.



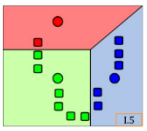
Let us first look at a simple case, 3-means clustering. For simplicity, we will assume that our feature space is two-dimensional, which is shown at the top-right. We randomly pick three points in the distributions and assume them to be the initial means (centroids) of the three clusters. These three initial means are shown as red, green and blur dots in the bottom-right.



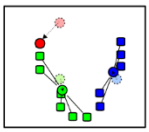
We create three clusters by assigning each feature point to the nearest mean. Next, we recompute the mean of each of the three clusters. We repeat the above steps (2 and 3) until the three means converge, i.e., they do not change appreciably. At that point, we can say that we have found our three clusters.

3-Means Clustering Example

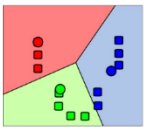
Step 2: Create 3 clusters by assigning each feature point to the nearest mean.



Step 3: Recompute the mean of each cluster.



Step 4: Repeat steps 2 and 3 until convergence.



25

The above method can be generalized to k means, i.e., k segments. We are given an image with N pixels, where each pixel has a feature vector. We randomly pick k points in features space as our initial means (m_1 through m_k). For each pixel x_j, we find the mean m_i that is nearest to its feature vector f_j, and assign it to i (step 2). Once this is done for all pixels, we recompute the mean for each cluster (step 3). If the locations of all the k means change by less than some threshold, we stop. Otherwise, we go back to step 2.

k-Means Clustering

Given: Image with N pixels and number of clusters k.

Task: Find the k clusters.

Clustering:

1: Pick k points randomly as the initial centroids (means) $\{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_k\}$ of the k clusters in feature space.

2: For each pixel $\mathbf{x}_j$ find nearest cluster mean $\mathbf{m}_i$ to pixel's feature $\mathbf{f}_j$ and assign pixel to cluster i.

3: Recompute mean for each cluster using its assigned pixels.

4: If changes in all k means is less than a threshold ε, stop. Else go to step 2.

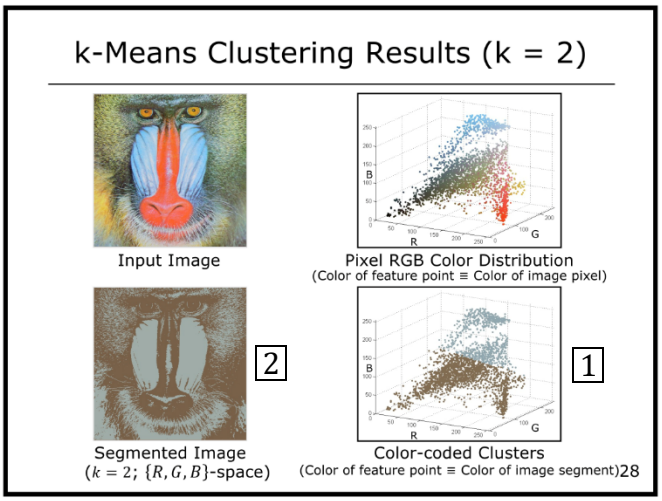
[Lloyd 1957, MacQueen 1967] 26

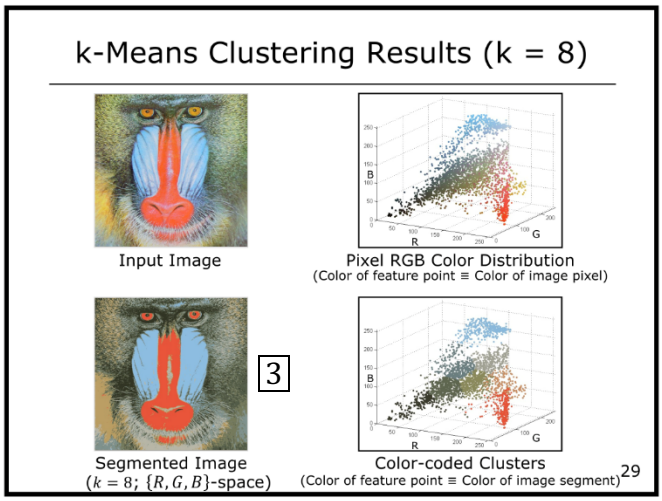
For initialization, we simply used k points randomly picked from our distribution. If any of these points are too close to each other, we can randomly resample the distribution until all the points are well separated. Another approach is to pick the initial means such that they are uniformly distributed within the range of the entire distribution. Yet another approach is to perform k-means clustering on a much smaller subset of the distribution where the subset is randomly picked. Then, the k means computed for this subset are used as initial means for the entire distribution.

k-Means Initialization Methods

- Method 1: Select k random feature points as initial centroids. If two points are very close, resample.
- Method 2: Select k uniformly distributed means within the range of the distribution.
- Method 3: Perform k-means clustering on a subset of pixels and use the result as the initial means.

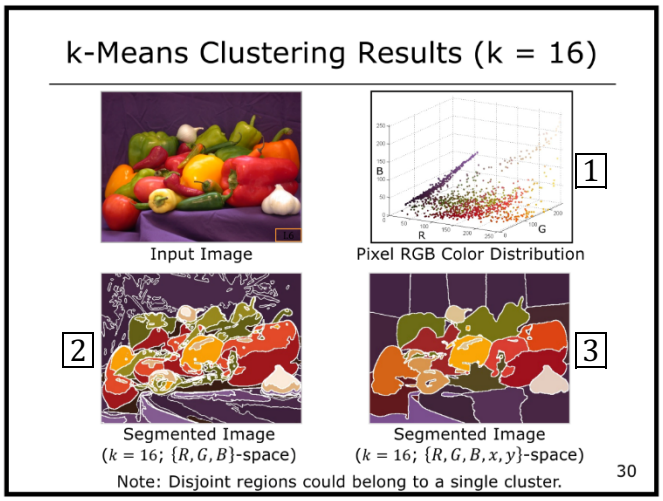
27





Let us now look at some results produced by k-means clustering. On the left is the mandrill image and our goal is to segment it into just two regions, k =2. Our feature space is just the color (RGB) space. The bottom-right plot shows the two computed clusters, each displayed with its average color [1]. The pixels in the image are recolored such that each pixel has the average color of the cluster it belongs to [2]. This segmentation is clearly not very useful given the complexity of the original image. If we increase the number of clusters to 8, we get the result shown in the right slide [3], which is a more meaningful segmentation of the image.

Here is another image, of a variety of peppers. When we use the distribution in the three-dimensional color space shown in [1], and set k =16, we get the segmentation shown in [2]. We see that this segmentation is not very useful as each cluster maps to many disconnected segments in the image. This is because we have given no consideration to the location of each pixel in the image, but rather just its color. By adding the spatial coordinates (x and y) of each pixel to the feature vector, we have a five-dimensional vector. With this change, we are encouraging pixels that are close together to belong to the same segment and pixels that are far apart to not belong to the same segment. The segmentation result in this case is shown in [3], and is clearly a more useful description of the scene.



To summarize, k-means is a simple and efficient clustering algorithm. It does have the drawback that one needs to pick the number of clusters, k. In the case of complex scenes, the results can also be sensitive to initial means chosen. For instance, if outliers of the distribution are chosen for some of the means, the result could include some insignificant image segments. Even so, k-means clustering is popular and widely used for segmentation of relatively simple images.

k-Means Clustering: Comments

- Simple and reasonably fast
- Need to pick the number of clusters k
- Sensitive to initialization
- Sensitive to outliers

Can we do better?

31

We know that k-means has two drawbacks; we need to specify the number of segments k, and it can be sensitive to the initialization of the k means. The mean shift we discuss now mitigates these two problems.

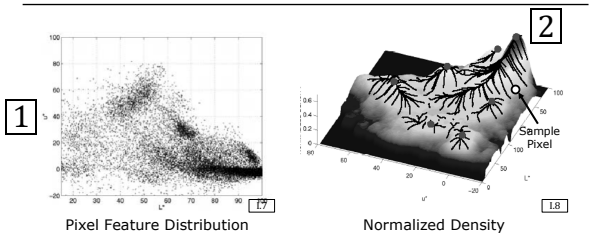
Mean Shift Segmentation

Shree K. Nayar
Columbia University

Topic: Image Segmentation, Module: Perception
First Principles of Computer Vision

32

The Concept of Mean Shift

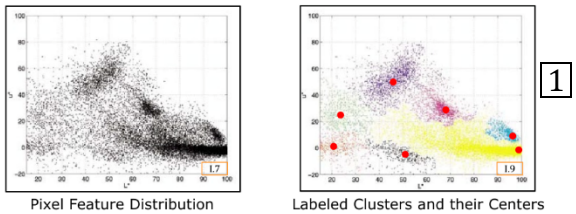


Pixel Feature Distribution Normalized Density

- Each hill represents a cluster.
- Peak (mode) of hill represents "center" of cluster.
- Each pixel climbs the steepest hill within its neighborhood.
- Pixel assigned to the hill (cluster) it climbs.

33

The Concept of Mean Shift

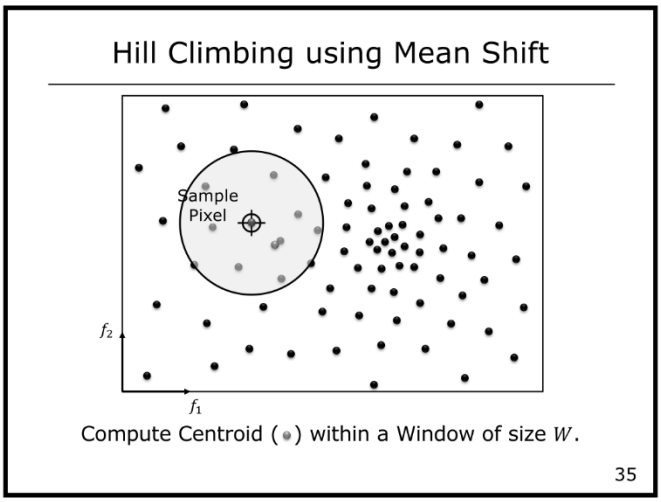


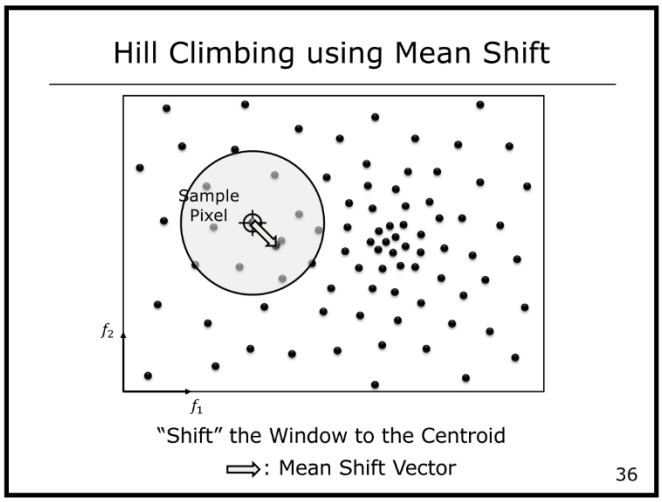
Pixel Feature Distribution Labeled Clusters and their Centers

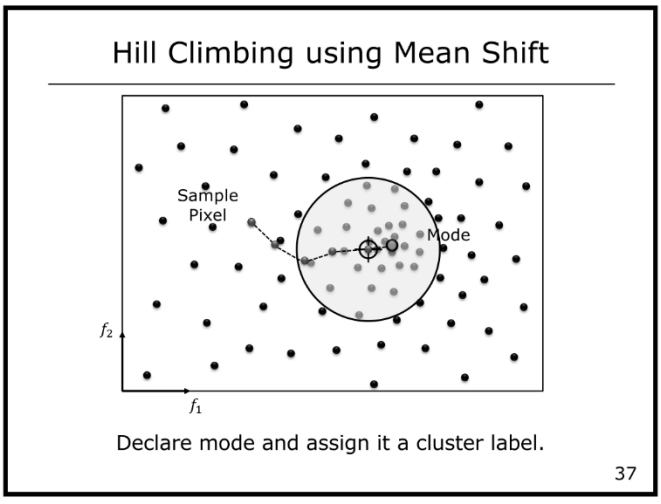
- Each hill represents a cluster.
- Peak (mode) of hill represents "center" of cluster.
- Each pixel climbs the steepest hill within its neighborhood.
- Pixel assigned to the hill (cluster) it climbs.

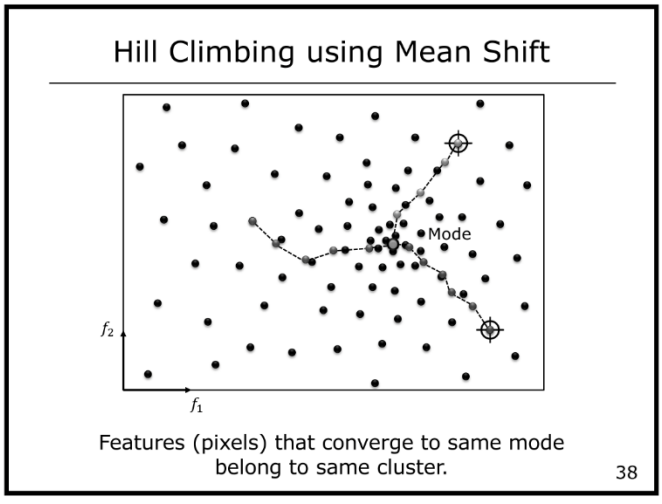
34

Show on the left of slide 33 is a pixel feature distribution [1]. For ease of visualization, we use a two-dimensional distribution. We can see that this distribution has varying density. We can represent the distribution as a normalized density function as shown on the right [2]. The first two axes correspond to the pixel features, and the third (vertical) axis tells us how dense the distribution is. We can view this density function as having several hills, where each hill corresponds to a cluster (segment). For each pixel in the image, we would therefore like to find the hill it belongs to. Let us consider the sample pixel shown in [2]. Our goal is to find the hill that would be the steepest for the pixel to climb. Once we have found this hill, the pixel is labeled as belonging to the cluster the hill represents. This approach to segmentation is called mean shift. In slide 34, we see each point in feature space with the color of the hill it climbs [1]. In this example, the number of hills found is seven and hence the image is deemed to have seven segments. Note that with this approach we do not need to specify the number of segments in the image.





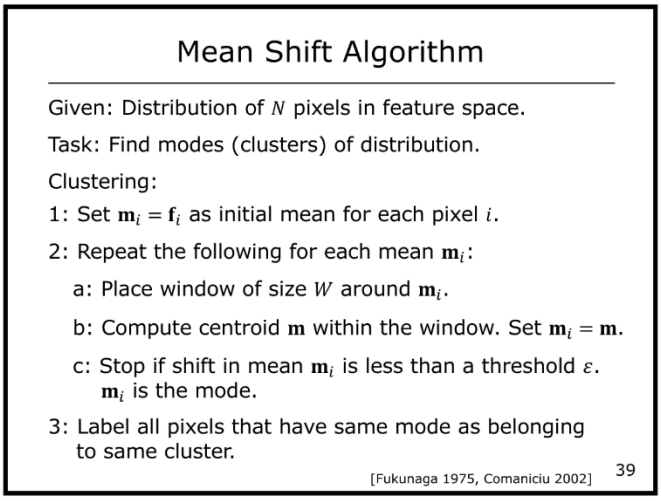




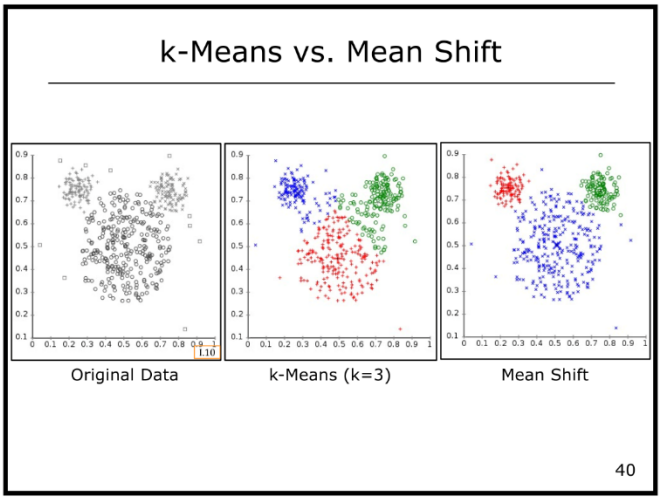
Let us now look at how mean shift is implemented. Once again, we consider a two-dimensional feature space, noting that the algorithm is applicable to a feature space of any dimensionality. In slide 35, we

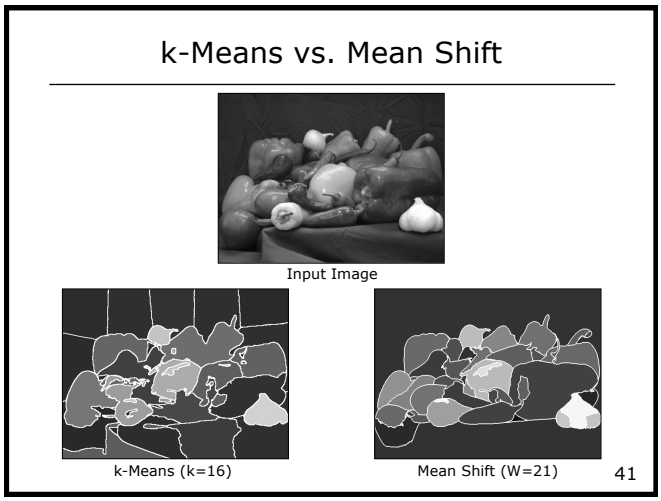
show the feature of a sample pixel and we wish to find the hill it belongs to. We take a circular window around the pixel of radius W and find the mean (centroid) of all the points within the window. This mean could be a simple mean or a weighted mean where points closer to the center of the window are given higher weights. Using a weighted mean reduces the impact of points that are farther from the center of the window. Next, we move the center of our window to the computed mean, as shown in slide 36. Now we repeat the process: compute a new mean using all the points inside the window and move the center of the window to that location. Since the mean of the window shifts with each iteration the method is called mean shift. Eventually, for the sample pixel, we end up climbing the steepest hill close to it, as shown in slide 37. Once the peak (mode) of the hill has been reached, the mean will not shift anymore, and the pixel is labeled as belonging to the hill. This process is applied to every point in feature space, as shown in slide 38. All the pixels whose features climb the same hill belong to the same image segment.

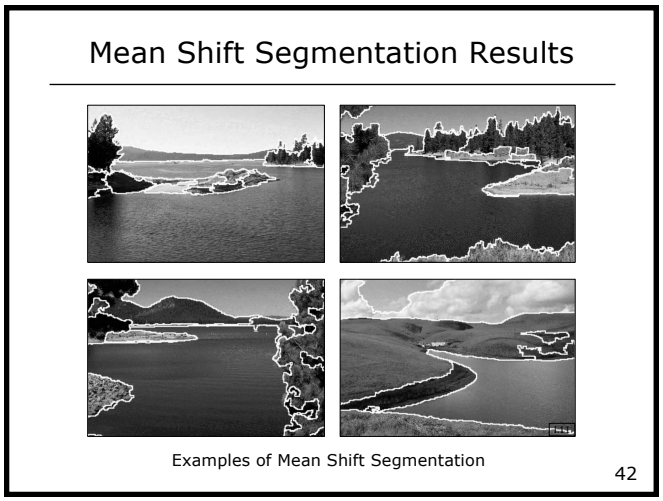
The mean shift algorithm can be summarized as follows. Given a distribution of N pixels in feature space, our goal is to find the clusters of the distribution. We begin by setting the mean value of the feature of each pixel i to be its feature, $\mathbf{m_i} = \mathbf{f_i}$. Then, we repeat the following for each mean $\mathbf{m_i}$. We place a window of size W around $\mathbf{m_i}$, compute the mean $\mathbf{m}$ using all the features within the window, and then set $\mathbf{m_i}$ to the new mean $\mathbf{m}$. If the shift in the mean is less than some threshold, then $\mathbf{m_i}$ is a mode. We label all pixels that have the same mode as belonging to the same cluster.



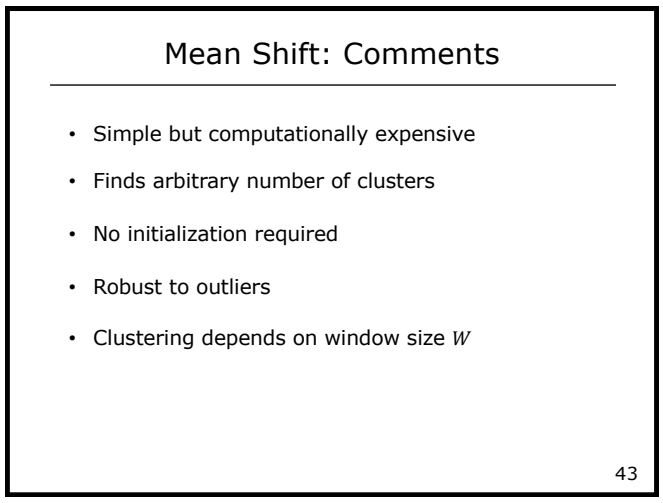
Here, on the left, we see a distribution with three clusters. Each point is marked as a circle, plus, or cross based on the cluster it belongs to, as this information is given to us and can be used as ground truth while evaluating the performance of a clustering algorithm. If we apply the k-means algorithm (slide 26) using k =3, we get the three colored clusters in the middle plot. Comparing this result with our ground truth, we see that several of the points are misclassified. The right plot shows the result of applying the mean shift algorithm, which is very close to the ground truth.







Let us take a look at how the mean shift algorithm performs on real images. On the left, we compare results for the pepper image obtained using k-means and mean shift. In the case of k-means, we set k =16 and used a five-dimensional feature space as in slide 30. While this result is fine, it includes several instances of disjoint image regions being assigned to the same segment. In the case of mean shift with a window radius of 21, we get the cleaner segmentation on the right. Note that in the case of mean shift we do not need to specify the number of segments k, but we do need to pick the window radius. Slides 42 shows results for more challenging scenes. While these are not perfect segmentations, they can be seen to be reasonable ones.



We conclude with a few comments related to mean shift. It is a simple method, but is computationally expensive as means need to be repeatedly computed for each pixel during the hill climbing process. On the upside, we are able to find an arbitrary number of segments, and the means do not need to be initialized as in the case of k-means. It also has the advantage that outliers in the feature space do not create new clusters, but rather end up being assigned to a meaningful cluster. The only parameter that one needs to choose in the case of mean shift is the window radius W. It should be noted that this needs to be done with care as, for a complex image, the result could be sensitive to W.

Another approach to the image segmentation is to represent the image as a graph and find cuts in the graph. Let us first look at how we can represent an image as a graph.

Graph Based Segmentation

Shree K. Nayar
Columbia University

Topic: Image Segmentation, Module: Perception
First Principles of Computer Vision

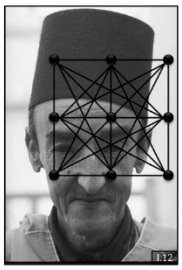
44

Consider the image shown here. We can define every pixel in the image as a vertex of a graph, and have edges between pairs of pixels that are close to each other. We denote this graph as *G* with vertices *V* and edges *E*. Each edge is weighted by the affinity, or similarity, between its two vertices, i.e., pixels. This notion of affinity is important to the way graph-based segmentation works.

Graph Based Segmentation

Images as Graphs:

- A vertex for each pixel.
- An edge between each pair of pixels.
- Graph Notation: $G = (V, E)$ where V and E are the sets of vertices and edges, respectively.
- Each edge is weighted by the affinity or similarity between its two vertices.



Input Image

45

Let us say that we have pixels i and j with features f_i and f_j. We can define the dissimilarity s between the two pixels as the L^2 norm between the features f_i and f_j. Using s, we can define the affinity A between features f_i and f_j as shown in [1]. The smaller the dissimilarity s, the larger the affinity A. The sensitivity of A to s depends on the parameter σ, which must be chosen. The affinity A is the weight $w(i,j)$ associated with the edge.

Measuring Affinity

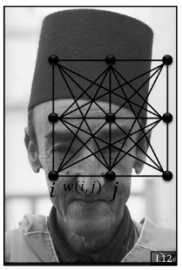
Let i and j be two pixels whose features are $\mathbf{f}_i$ and $\mathbf{f}_j$.

Pixel Dissimilarity:

$$S(\mathbf{f}_i, \mathbf{f}_j) = \sqrt{\left(\sum_k (f_{ik} - f_{jk})^2\right)}$$

Pixel Affinity:

$$w(i,j) = A(\mathbf{f}_i, \mathbf{f}_j) = e^{\left\{\frac{-1}{2\sigma^2} S(\mathbf{f}_i, \mathbf{f}_j)\right\}} \quad [1]$$

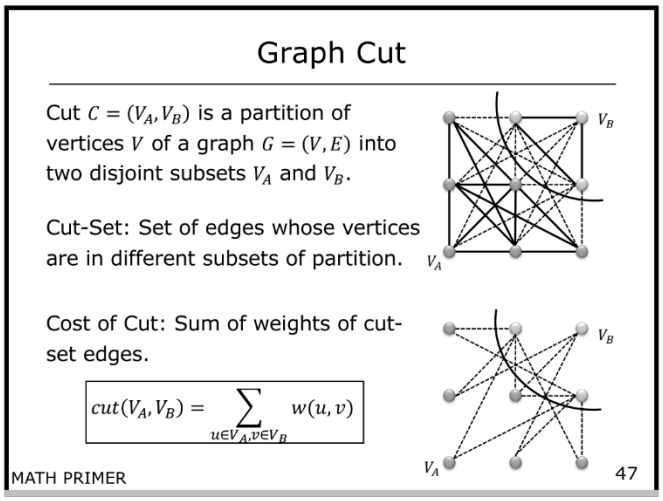


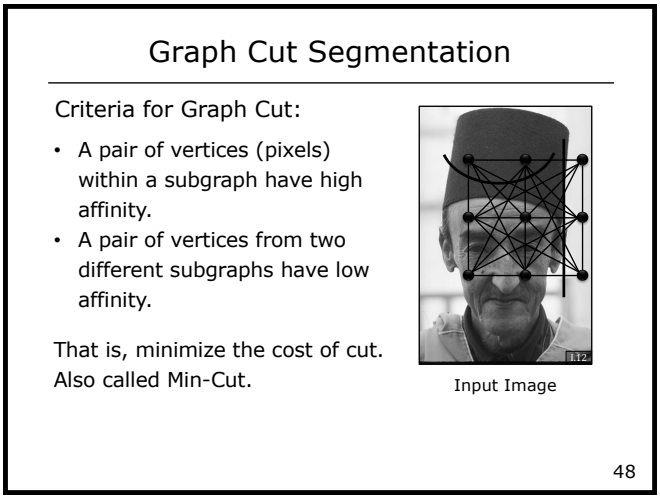
Input Image

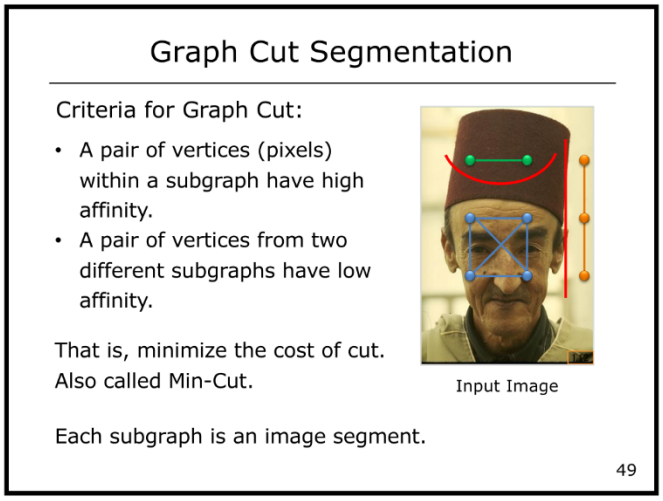
Smaller the Dissimilarity, Larger the Affinity

46

Now, let us define what we mean by a cut. Given a graph *G*, a cut *C* is a partition of the vertices *V* into two disjoint subsets, V_a and V_b. The cut-set is the set of edges whose two vertices belong to different subsets of the partition. The cost of a cut is defined as the sum of the weights of the cut-set edges.

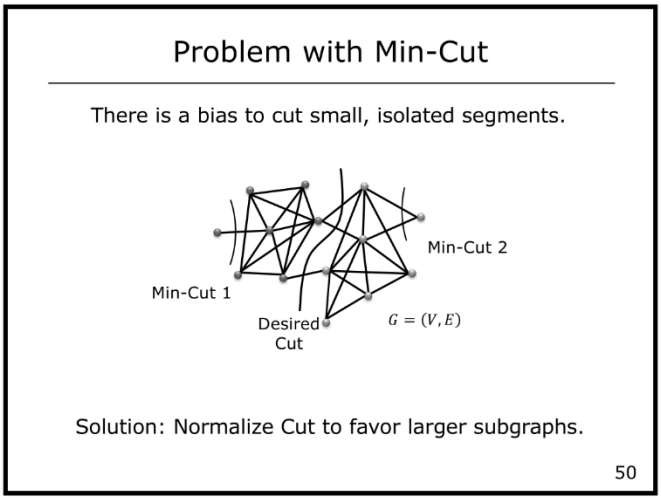




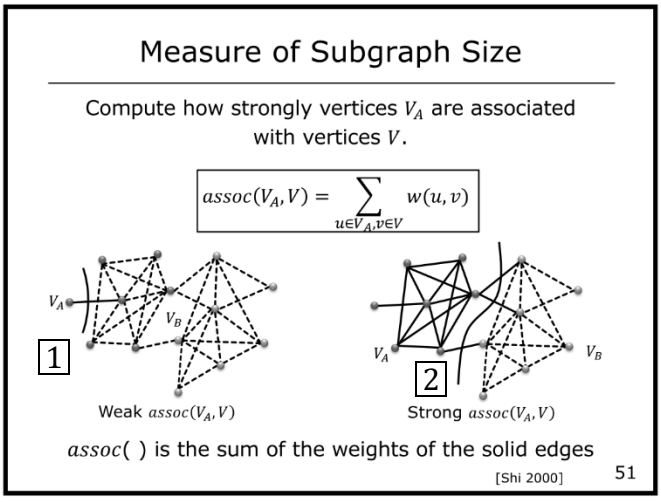


With these definitions in place, we can develop our segmentation algorithm based on the following criteria. The first is that a pair of vertices within a subgraph should have high affinity, i.e., similarity of pixels. Conversely, a pair of vertices from two different subgraphs should have low affinity. We want to find cuts in the graph that have the minimum cost associated with them. An algorithm that finds these cuts is referred to as a min-cut algorithm. In slide 49, we show a plausible result of finding min-cuts that result in three subgraphs, where each subgraph is an image segment.

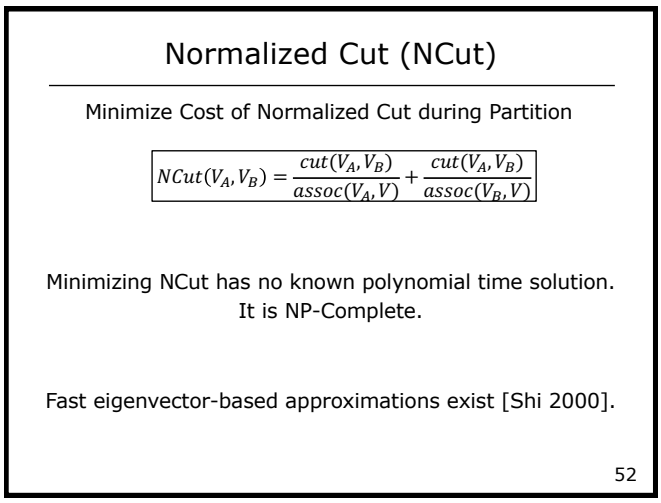
One problem with the min-cut algorithm is that it tends to cut the graph into small, isolated segments. In the example shown here, let us say that the middle edges are relatively weak because they have low affinities. So, we would like to have a cut that goes through these edges, which is shown as the desired cut. However, the cost of cutting the three weak edges of the desired cut happens to be greater than the cost of cutting a single stronger edge denoted by Min-Cut 1, or even the two edges denoted by Min-Cut 2. This suggests that in order to get our desired cut, we need to favor cuts that produce larger subgraphs. Otherwise, we are going to end up with lots of small subgraphs.



Therefore, we need to normalize the cost of a cut by the sizes of the subgraphs it produces. That requires us to come up with a measure of the size of a subgraph. One approach is to compute how strongly the vertices of a subgraph V_A are associated with the larger graph *V*. We refer to this as the association between V_A and *V*, which is defined as the sum of the weights between V_A and *V*. We see that the association in the case of the cut shown in 1 is weak compared to the desired cut shown in 2.



That brings us to the notion of a normalized cut, Ncut, which is defined here. It is the cut with cost equal to the sum of the cost of the cut of *V* into V_A and V_B, divided by the association of V_A with *V*, and the cost of the cut of *V* into V_A and V_B, divided by the association of V_B with V. Our goal therefore is to find all the normalized cuts of the graph corresponding to an image. The one issue here is that there is no known polynomial-time algorithm for finding normalized cuts. It is an NP-complete problem, but there are many approximations that have been proposed. One of the popular



approximations was suggested by Shi, which uses fast eigenvector-based approximations. This approach leverages spectral methods, and is widely used for image segmentation.

Here are some image segmentation results produced by the above algorithm. The pixel features used here include the pixel's brightness and its spatial coordinates. These are fairly complex images, and as we can see, the segmentation results are impressive.

NCut Segmentation Results

Segmented Images
Pixel Feature: $\{Brightness, Location\}$

53

Let us summarize what we have covered in terms of image segmentation. First, segmentation is a fundamental problem in computer vision. It is used as a precursor for object detection and recognition. The first method we looked at, the simplest of them all, is k-means segmentation. Next, as an improvement to k-means, we described the mean shift algorithm. Finally, we looked at normalized graph cut segmentation. In this lecture, we have only discussed the most basic versions of these algorithms. Particularly when it comes to mean shift and graph-based algorithms, there have been numerous improvements proposed, which can be found in the literature.

Image Segmentation Summary

- Important for object detection and recognition.
- k-Means Segmentation
- Mean Shift Segmentation
- Normalized Graph Cut Segmentation
- Many variations of mean shift and graph based algorithms have been proposed. [Paris 2007] [Wang 2004] [Tolliver 2006] [Cour 2005] [Felzenszwalb 2004] [Boykov 2004]

54

References and Credits

Shree K. Nayar

Columbia University

Topic: Image Segmentation, Module: Perception

First Principles of Computer Vision

55

References: Textbooks

Computer Vision: Algorithms and Applications
Szeliski, R., Springer

Computer Vision: A Modern Approach
Forsyth, D and Ponce, J., Prentice Hall

56

References: Papers

[Comaniciu 2002] D. Comaniciu and P. Meer. "Mean Shift: A Robust Approach toward Feature Space Analysis". PAMI 2002.

[Fukunaga 1975] K. Fukunaga and L. D. Hostetler. "The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition" . IEEE Transactions on Information Theory 21. 1975.

[Kass 1987] M. Kass, A. Witkin and D. TerzoPoulos. *"Snakes: Active Contour Models", IJCV,* 1987.

[Lloyd 1957] S. P. Lloyd. "Least square quantization in PCM". IEEE Transactions on Information Theory 28. 1957.

[MacQueen 1967] J. B. MacQueen. "Some Methods for classification and Analysis of Multivariate Observations" . Proc of 5th Berkeley Symposium on Mathematical Statistics and Probability. University of California Press. 1967.

[Shi 2000] J. Shi and J. Malik. "Normalized cuts and image segmentation". PAMI 2000.

[Smith 1988] B. Smith. "Foundations of Gestalt Theory". Philosophia Verlag, 1988.

57

References: Papers

[Boykov 2004] Y. Boykov and G. Funka-Lea. "Graph cuts and efficient N-D image segmentation." IJCV 2004.

[Cour 2005] T. Cour, F. Benezit, and J. Shi. "Spectral segmentation with multiscale graph decomposition." CVPR 2005.

[Felzenszwalb 2004] P. F. Felzenszwalb and D. P. Huttenlocher. "Efficient graph-based image segmentation." IJCV 2004.

[Martin 2001] D. Martin, C. Fowlkes, D. Tal and J. Malik. "A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics." ICCV 2001.

[Paris 2007] S. Paris and F. Durand. "A topological approach to hierarchical segmentation using mean shift." CVPR 2007.

[Todorovic 2008] D. Todorovic. "Gestalt principles." Scholarpedia, 3(12):5345.

[Tolliver 2006] D. Tolliver and G. Miller. "Graph partitioning by spectral rounding: Applications in image segmentation and clustering." CVPR 2006.

[Wang 2004] J. Wang, B. Thiesson, Y. Xu and M. Cohen. "Image and video segmentation by anisotropic kernel mean shift." ECCV 2004.

58

Image Credits

I.1 Credited to Ronald C James (Photographer).
I.2 Jitendra Malik. Used with permission.
I.3 Jitendra Malik. Used with permission.
I.4 https://www.mathworks.com/help/matlab/ref/imapprox.html?s_tid=srchtitle. Mathworks.
I.5 https://en.wikipedia.org/wiki/K-means_clustering. Licensed under CC BY-SA 3.0.
I.6 https://www.mathworks.com/help/matlab/ref/rgb2gray.html. Mathworks.
I.7 Dorin Comaniciu and Peter Meer. Used with permission.
I.8 Dorin Comaniciu and Peter Meer. Used with permission.
I.9 Dorin Comaniciu and Peter Meer. Used with permission.
I.10 https://en.wikipedia.org/wiki/File:ClusterAnalysis_Mouse.svg. Public Domain.
I.11 Dorin Comaniciu and Peter Meer. Used with permission.
I.12 Jitendra Malik. Used with permission.
I.13. Timothee Cour. Used with permission.
I.14 Purchased from Shutterstock.com.

59

Acknowledgements: Thanks to Joel Salzman, Kevin Chen and Tracy Cui for their help with transcription, editing and proofreading**.**

References

[Szeliski 2022] Computer Vision: Algorithms and Applications, Szeliski, R., Springer, 2022.

[Forsyth and Ponce 2003] Computer Vision: A Modern Approach, Forsyth, D. and Ponce, J., Prentice Hall, 2003.

[Comaniciu 2002] Mean Shift: A Robust Approach toward Feature Space Analysis, Comaniciu, D. and Meer, P., PAMI, 2002.

[Fukunaga 1975] The Estimation of the Gradient of a Density Function with Applications in Pattern Recognition, K. Fukunaga, K. and Hostetler, L. D., IEEE Transactions on Information Theory, 21. 1975.

[Kass 1987] Snakes: Active Contour Models, Kass, M., Witkin, A., and Terzopoulos, D., IJCV, 1987.

[Lloyd 1957] Least square quantization in PCM, Lloyd, S. P., IEEE Transactions on Information Theory, 28. 1957.

[MacQueen 1967] Some Methods for classification and Analysis of Multivariate Observations, MacQueen, J. B., Proc. of 5th Berkeley Symposium on Mathematical Statistics and Probability, University of California Press. 1967.

[Shi 2000] Normalized cuts and image segmentation, Shi, J. and Malik, J., PAMI, 2000.

[Smith 1988] Foundations of Gestalt Theory, Smith, B., Philosophia Verlag, 1988.

[Boykov 2004] Graph cuts and efficient N-D image segmentation, Boykov, Y. and Funka-Lea, G., IJCV, 2004.

[Cour 2005] Spectral segmentation with multiscale graph decomposition, Cour, T., Benezit, F., and Shi, J., CVPR, 2005.

[Felzenszwalb 2004] Efficient graph-based image segmentation, Felzenszwalb, P. F. and Huttenlocher, D. P., IJCV, 2004.

[Martin 2001] A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics, Martin, D., Fowlkes, C., Tal, D., and Malik, J., ICCV, 2001.

[Paris 2007] A topological approach to hierarchical segmentation using mean shift, Paris, S. and Durand, F., CVPR, 2007.

[Todorovic 2008] Gestalt principles, Todorovic, D., Scholarpedia, 3(12):5345.

[Tolliver 2006] Graph partitioning by spectral rounding: Applications in image segmentation and clustering, Tolliver, D. and Miller, G., CVPR, 2006.

[Wang 2004] Image and video segmentation by anisotropic kernel mean shift, Wang, J., Thiesson, B., Xu, Y., and Cohen, M., ECCV, 2004.

[Nayar 2022D] Binary Images, Nayar, S. K., Monograph FPCV-1-3, First Principles of Computer Vision, Columbia University, New York, March 2022.

[Nayar 2022E] Image Processing I, Nayar, S. K., Monograph FPCV-1-4, First Principles of Computer Vision, Columbia University, New York, March 2022.

[Nayar 2022H] Boundary Detection, Nayar, S. K., Monograph FPCV-2-2, First Principles of Computer Vision, Columbia University, New York, June 2022.

[Nayar 2025F] Depth from Defocus, Nayar, S. K., Monograph FPCV-3-4, First Principles of Computer Vision, Columbia University, New York, March 2025.

[Nayar 2025G] Active Illumination Methods, Nayar, S. K., Monograph FPCV-3-5, First Principles of Computer Vision, Columbia University, New York, March 2025.

[Nayar 2025J] Optical Flow, Nayar, S. K., Monograph FPCV-4-3, First Principles of Computer Vision, Columbia University, New York, April 2025.