Image Segmentation

CS 4243 Computer Vision & Pattern Recognition

Angela Yao

Recap & Outline

Last Week

- Motivation for studying image edges
- Derivative filters to extract gradients
- Need for smoothing
- Canny edge detector

Today's Lecture

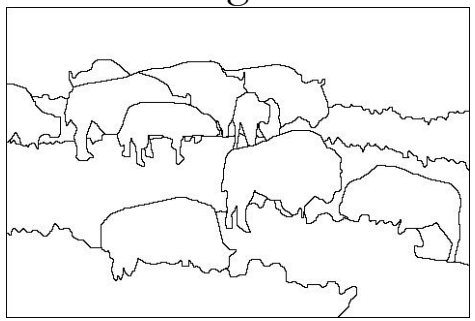
- Goal of segmentation is to separate image into coherent regions
- Treat segmentation as a clustering problem
- K-means clustering for segmentation
  - SLIC super-pixelling
- Mean-shift clustering for segmentation

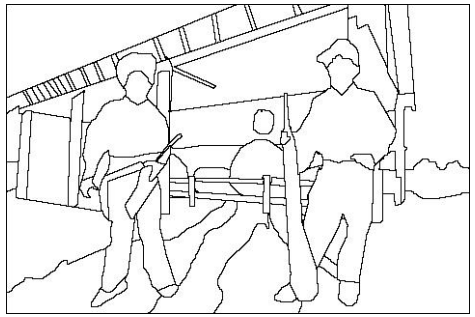
Goals of Segmentation

- Separate image into coherent regions or "objects"
- Group together similar pixels for efficiency of further processing

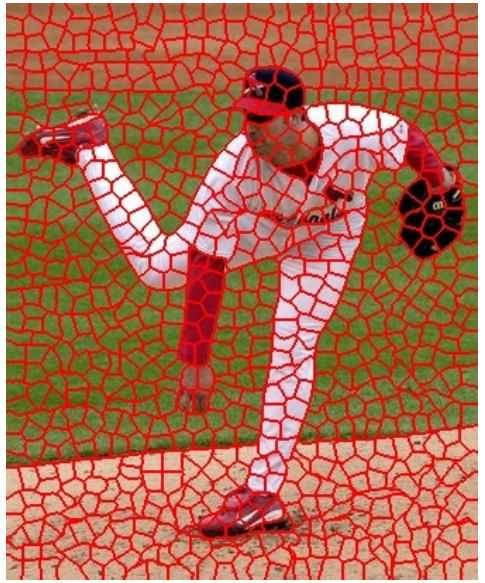
Image

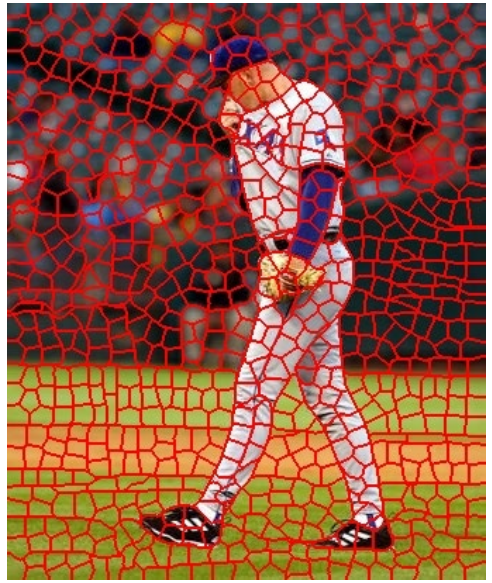
Human segmentation





"superpixels"

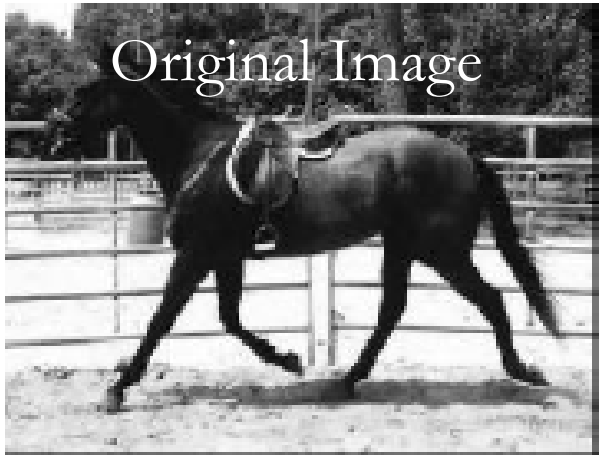




Ren & Malik. [Learning a classification model for segmentation.](#) ICCV 2003.

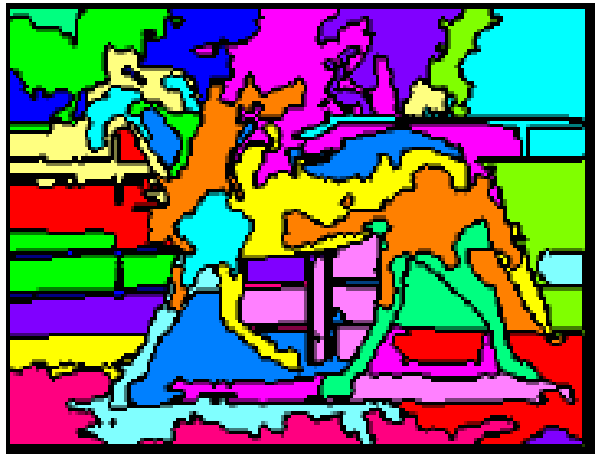
Slide credit: L. Lazebnik

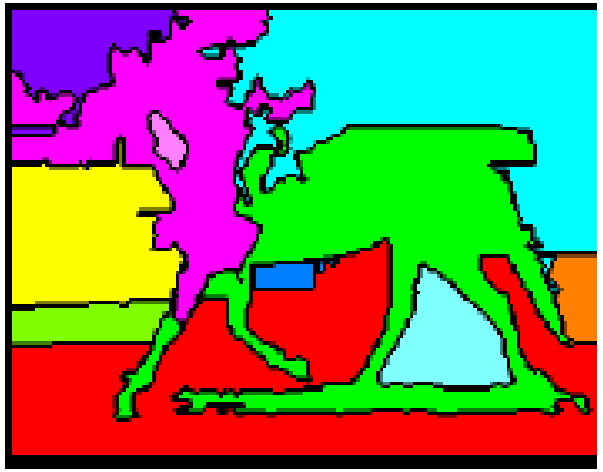
Types of Segmentation

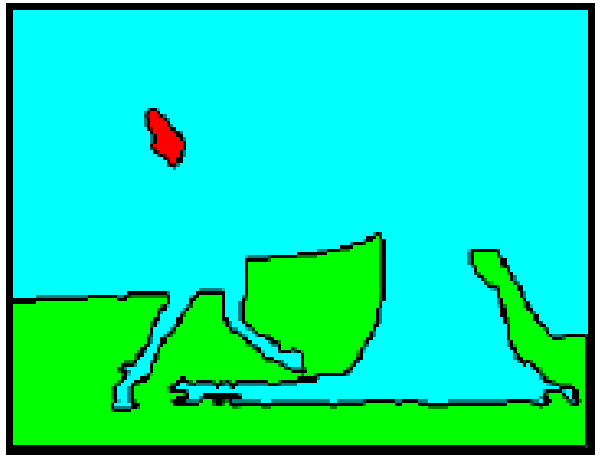


An image may have many different segmentations depending on the algorithm and parameters settings.

The "correct" segmentation depends on the objective and application at hand.



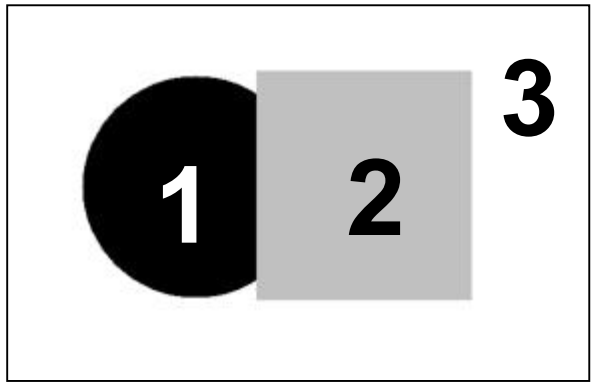




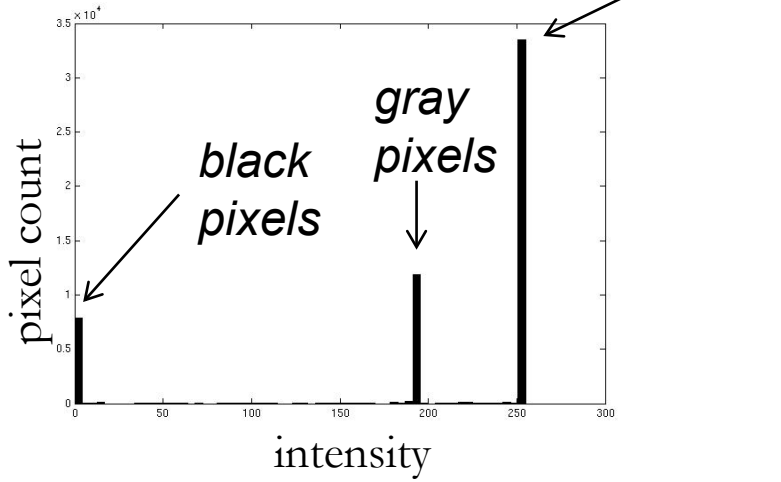
A Toy Example on Segmentation

Label every pixel based on its primary intensity:

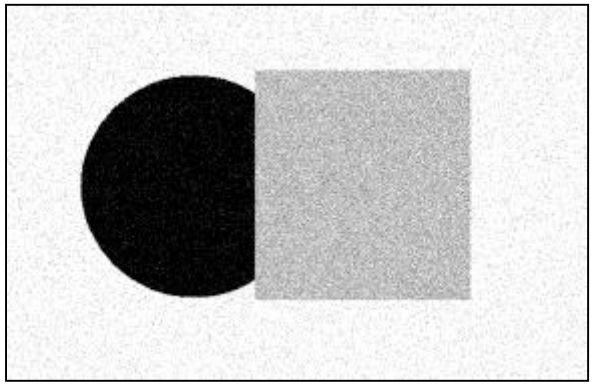
Segmentation based on intensity.



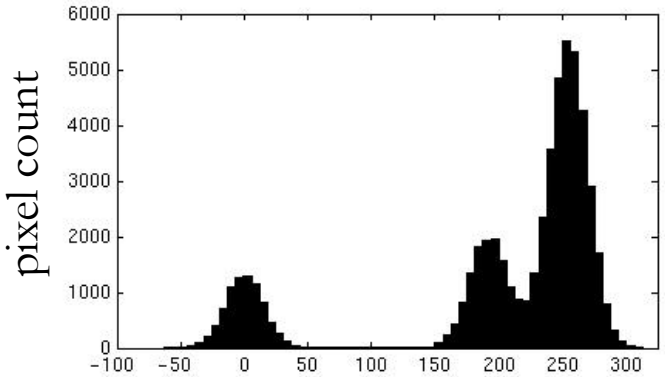
input image



What if the image was not so simple?



input image



Slide Credit: K. Grauman

“Segmentation” = Clustering ?

Clustering: group similar data points together and represent them with a single token

Two Key Challenges:

1. What makes two points similar or different?
   - Representing data (pixels) with features
   - Similarity/distance measure between the features

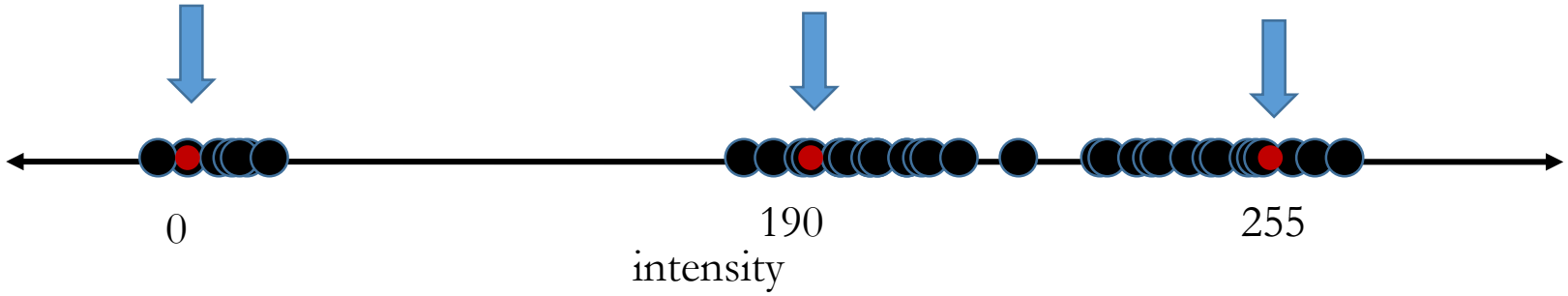
For purposes of our lecture today, we will consider only simple features such as pixel intensity or colour, and use a simple sum of squared differences (SSD).

2. How do we compute an overall grouping from pairwise similarities? (Which clustering algorithm?)
   - K-means: iteratively re-assign points to the nearest cluster center
   - Mean-shift clustering: estimate modes of a probability density function

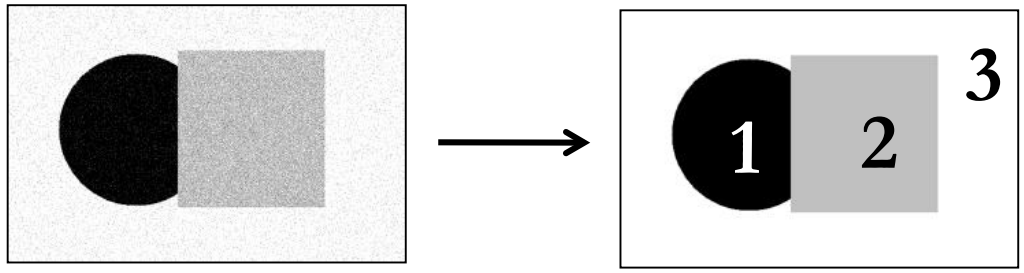
Slide adapted from D. Hoiem

k-Means Clustering

Clustering



- Goal: choose three "centers" as representative intensities, and label every pixel based on the nearest centers
- The best cluster centers minimize squared difference between all points and their nearest cluster center.

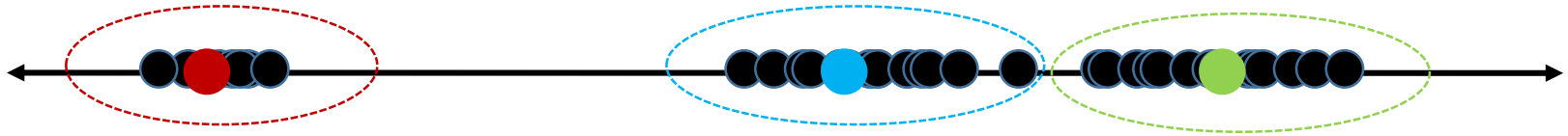


Adapted from K. Grauman

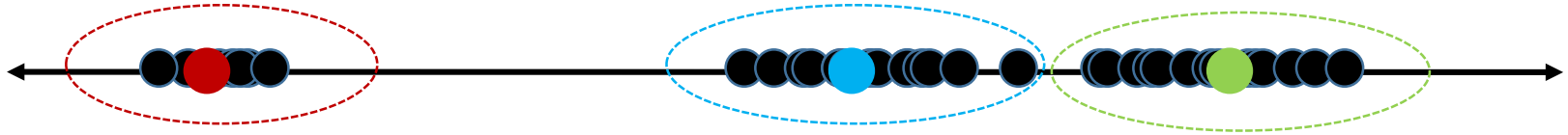
Clustering

With this objective, it is a "chicken and egg" problem:

- If we knew the **cluster centers**, we could allocate points to groups by assigning each to its closest center.



- If we knew the **group memberships**, we could get the centers by computing the mean per group.



Slide Credit: K. Grauman

k-Means Clustering

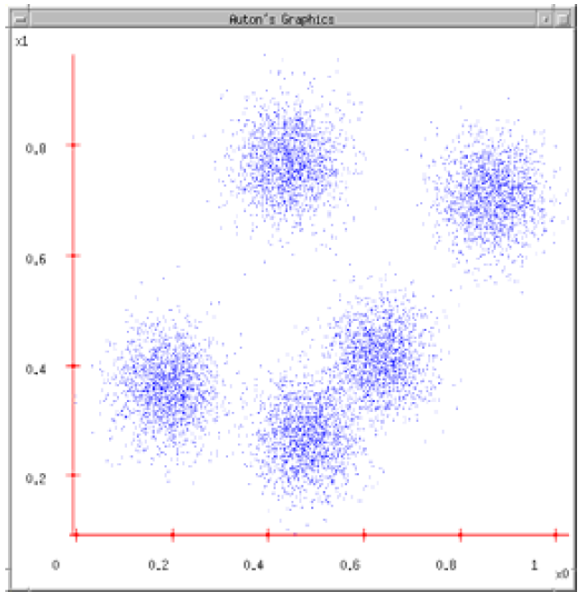
Basic idea: randomly initialize the *k* cluster centers, and iterate between assigning membership and computing cluster centers.

1. Given *K*, randomly initialize the cluster centers, c_1, ..., c_K
2. Given cluster centers, determine points in each cluster
   - For each point p_i, find the closest c_j. Put p_i into cluster j
3. Given points in each cluster, solve for c_j
   - Set c_j to be the mean of points in cluster j
4. If c_j have changed (up to some threshold), repeat Steps 2, 3.

Adapted from S. Seitz

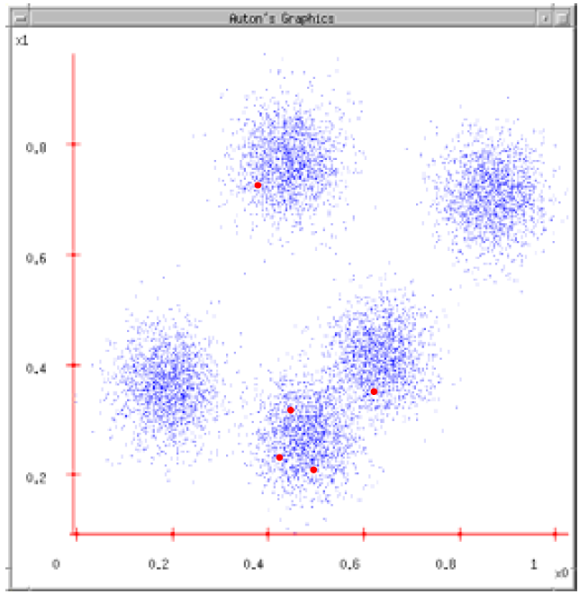
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*



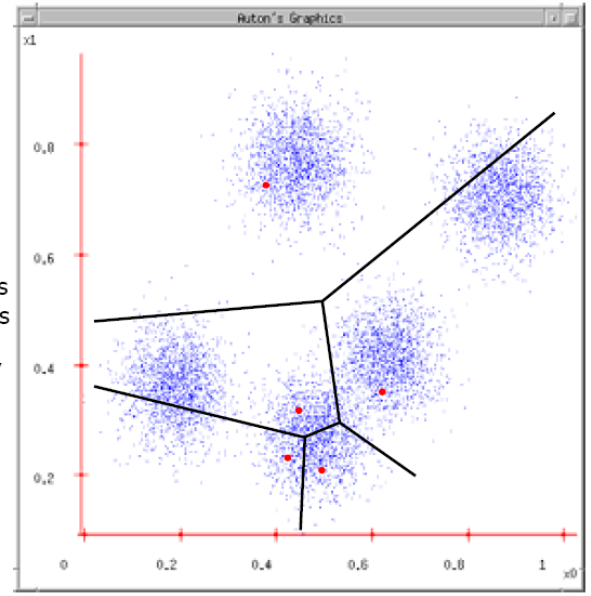
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations



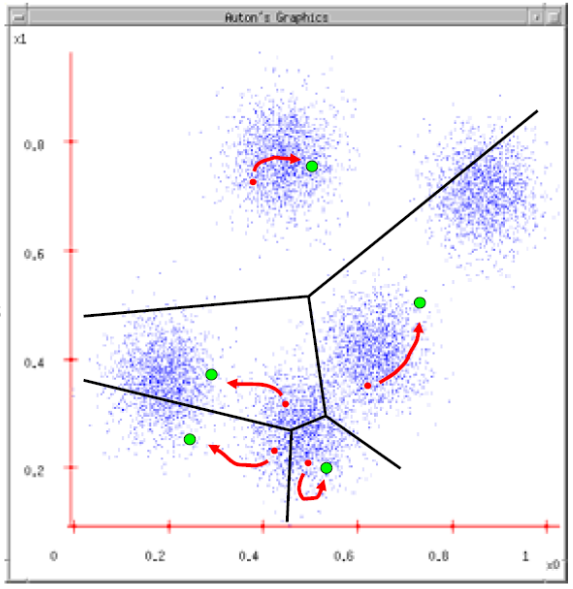
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to. (Thus each Center "owns" a set of datapoints)



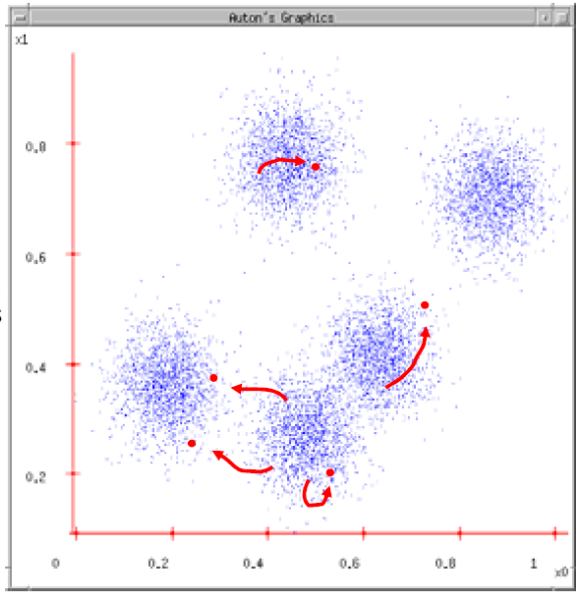
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns



K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns...
5. ...and jumps there
6. ...Repeat until terminated!



The Math of k-Means Clustering

Input: Given a set of data points P, number of clusters k

1. Randomly pick k points from P as the centers (means) $c_j, j = 1, \ldots, k$
2. Iterate (until max-iterations or c_j no longer changes up to some threshold)
   - Assign each point to nearest center: $y_i = \arg\min_j \|p_i - c_j\|^2$

     y_i: cluster id of point i
   - Re-estimate each center as mean of points assigned to it: $c_j = \frac{\sum_{i:y_i=j} p_i}{\sum_{i:y_i=j} 1}$

     $\sum_{i:y_i=j} 1$: points i belonging to cluster j

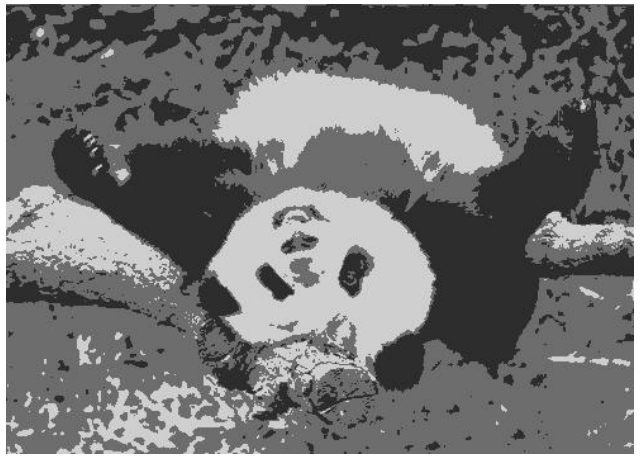
- Minimization of an objective function: $\min_{c,y} \sum_i ||p_i - c_{y_i}||^2$
- Every iteration of k-means takes a downward step
  - Fixes c and sets y to minimize objective;
  - Fixes y and sets c to minimize objective
- Always converges, but likely to a local minimum

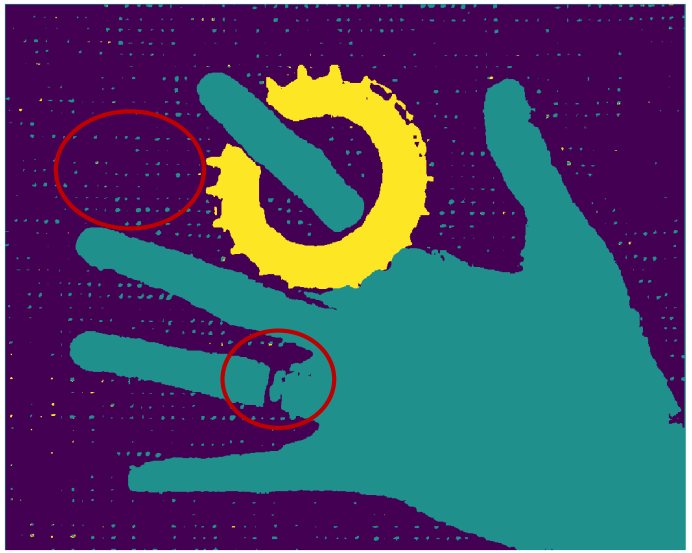
Slide adapted from K. Grauman

K-Means on Images

K=2

K=3





Essentially a non-uniform quantization of the image intensities.

Note the spatial discontinuities!

Adapted from K. Grauman

Feature Selection for Clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **intensity** similarity.

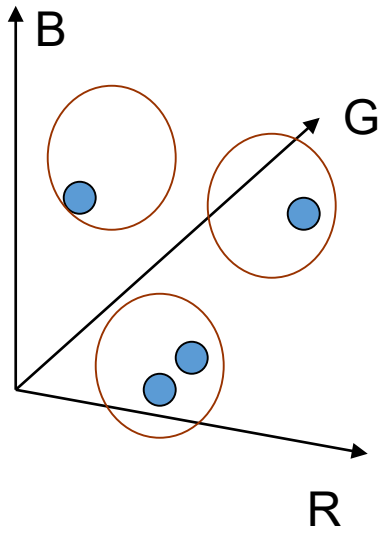
Clusters based on intensity don't have to be spatially coherent.

Slide Credit: K. Grauman

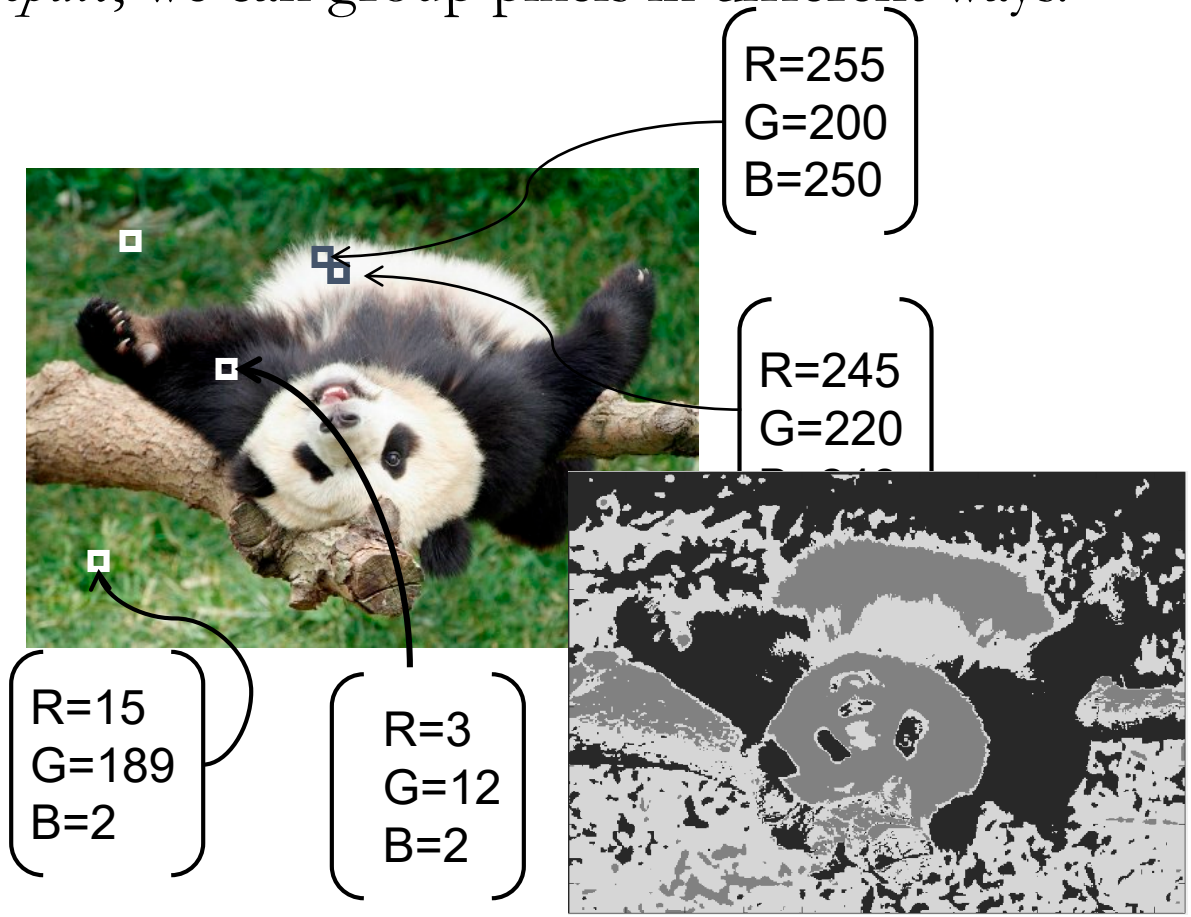
Feature Selection for Clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **color** similarity



Feature space: color value (3-d)

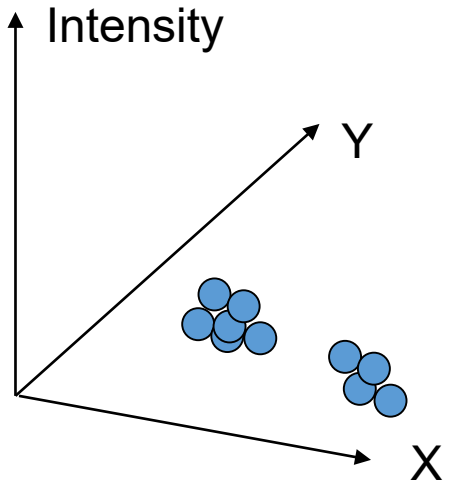


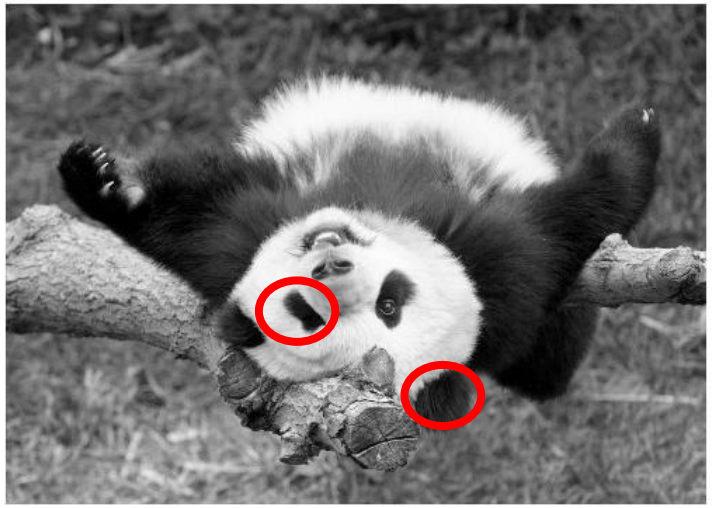
Slide Credit: K. Grauman

Feature Selection for Clustering.

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on
intensity+position similarity





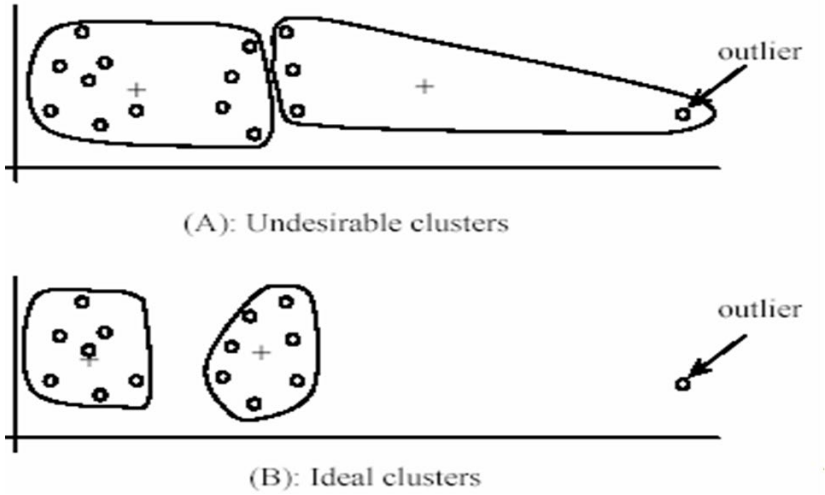
Both regions are black, but if we also include **position (x,y),** then we could group the two into distinct segments; way to encode both similarity & proximity.

Slide Credit: K. Grauman

k-Means Clustering: Pros & Cons

Pros

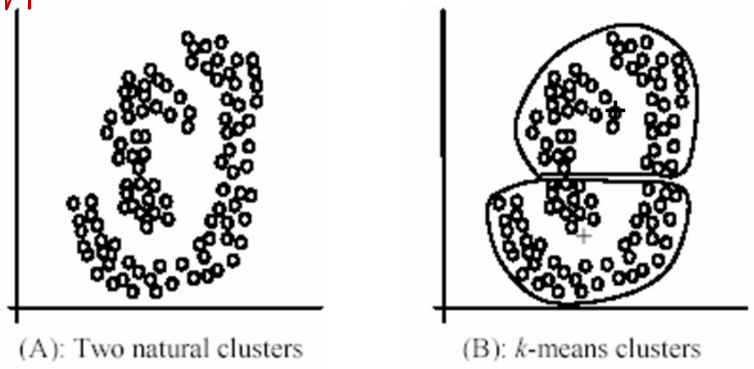
- Simple, fast to compute
- Converges to local minimum of within-cluster squared error



Cons/issues

- Setting k?
- Sensitive to initial centers
- Sensitive to outliers
- Detects spherical clusters
- Assumes means can be computed (efficient, meaningful)

Rectify this with mean-shift



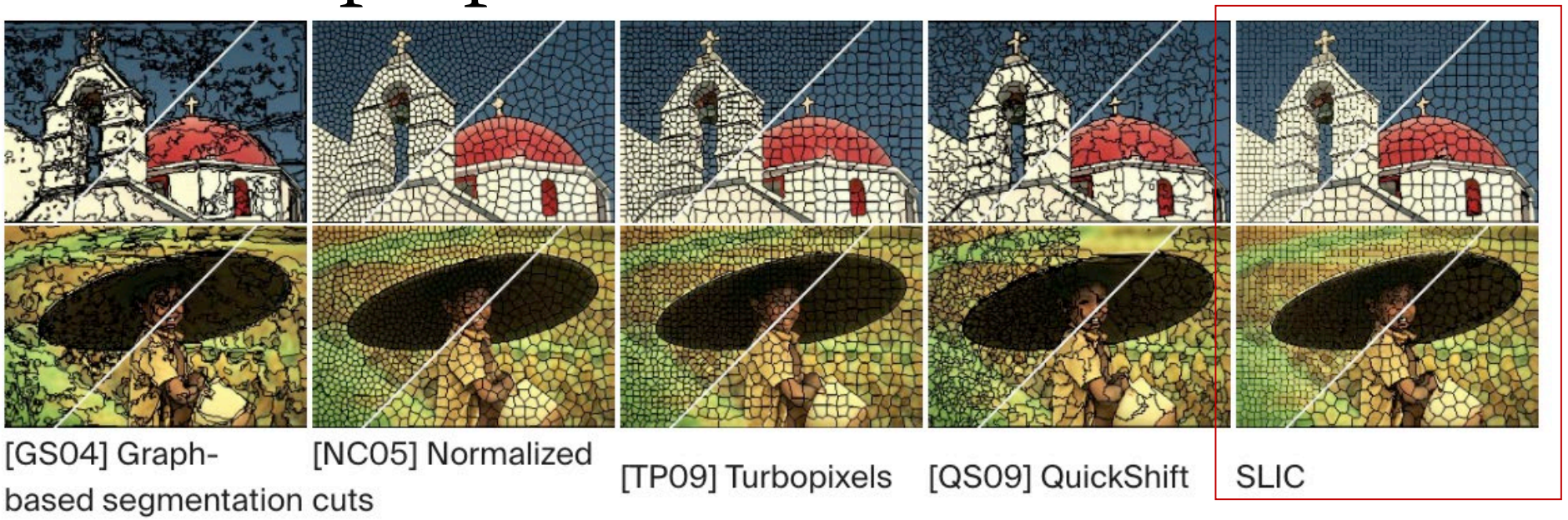
Adapted from K. Grauman

SLIC Superpixels

Simple Linear Iterative Clustering Superpixels

Adapting k-Means Clustering

What is a Superpixel?



- A group of pixels that share common characteristics, e.g. pixel intensity
- Use as inputs to other CV algorithms since it is a convenient and compact representation with perceptual meaning since it member pixels share visual properties
- Some are more regular in shape, others are less; number of superpixels is hyperparameter

https://www.epfl.ch/labs/ivrl/research/slic-superpixels/

Definitions, Features & Distance Measure

n_{tp} ; n_{sp} — Total number of pixels in image; number of superpixels we want

$s = [n_{tp}/n_{sp}]^{1/2}$ — Initial spacing of each superpixel

Features $z = [r, g, b, x, y]$

colour position

Colour Distance $d_c = \left[(r_j - r_i)^2 + (g_j - g_i)^2 + (b_j - b_i)^2\right]^{1/2}$

Grey-scale version $d_c = \left[(l_j - l_i)^2\right]^{1/2}$

Spatial Distance $d_s = \left[(x_j - x_i)^2 + (y_j - y_i)^2\right]^{1/2}$

L2 distance measures

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

scaling factors; set as max expected values of d_c and d_s

$d_{sm} = s = [n_{tp}/n_{sp}]^{1/2}$

Consider as hyperparameter weighing relative importance between colour similarity vs. spatial proximity.

Initialization

1. ***Initialize the algorithm:*** Compute the initial superpixel cluster centers,

$$\mathbf{m}_i = \left[r_i \;\; g_i \;\; b_i \;\; x_i \;\; y_i\right]^T, \;\; i = 1, 2, \ldots, n_{sp}$$

by sampling the image at regular grid steps, s. Move the cluster centers to the lowest gradient position in a 3×3 neighborhood. For each pixel location, p, in the image, set a label $L(p) = -1$ and a distance $d(p) = \infty$.

Different from k-means; rather than initialize randomly, initialize cluster centers on a grid.

Update + Convergence

2. ***Assign samples to cluster centers:*** For each cluster center $\mathbf{m}_i$, $i = 1, 2, \ldots, n_{sp}$, compute the distance, $D_i(p)$ between $\mathbf{m}_i$ and *each* pixel p in a $2s \times 2s$ neighborhood about $\mathbf{m}_i$. Then, for each p and $i = 1, 2, \ldots, n_{sp}$, if $D_i < d(p)$, let $d(p) = D_i$ and $L(p) = i$.

Compute distance only to closest set of cluster centers.

3. ***Update the cluster centers:*** Let C_i denote the set of pixels in the image with label $L(p) = i$. Update $\mathbf{m}_i$:

$$\mathbf{m}_i = \frac{1}{|C_i|} \sum_{\mathbf{z} \in C_i} \mathbf{z} \qquad i = 1, 2, \ldots, n_{sp}$$

where $|C_i|$ is the number of pixels in set C_i.

Same as k-means

4. ***Test for convergence:*** Compute the Euclidean norms of the differences between the mean vectors in the current and previous steps. Compute the residual error, E, as the sum of the n_{sp} norms. If $E < T$, where T a specified nonnegative threshold, go to Step 5. Else, go back to Step 2.

Same as k-means

Post-Processing

5. ***Post-process the superpixel regions:*** Replace all the superpixels in each region, C_i, by their average value, $\mathbf{m}_i$.

Optional, to create the "stained-glass" effect.

Example

$W = 512, \quad H = 384,$

$n_{sp} = 100$

$n_{tp} = 512 \times 384 = 196608$

$s = (196608/100)^{0.5} \approx 44$

$\frac{512}{44} \approx 12, \quad \frac{384}{44} \approx 9$

1. Initialize superpixels as grid of 9x12.
2. For each cluster center, check distance to all pixels within 88 x 88 range; assign pixels to closest (checked) center.
3./4. Update cluster centers, repeat assignment loop until convergence, i.e. no change up to some threshold.
5. Post-process.

Extra Notes

- At its core, is a modification of *k*-means clustering
- No provision made to enforce connectivity of superpixels, so it is possible for isolated pixels to remain after convergence. Can do some post-processing e.g. with connected components to re-assign
- Method equally applicable to other color spaces, e.g. HSV. Vector $\mathbf{z}$ can actually be defined as any real-valued feature, as long as we can define a meaningful distance measure

Mean-Shift Clustering

Mean Shift Algorithm

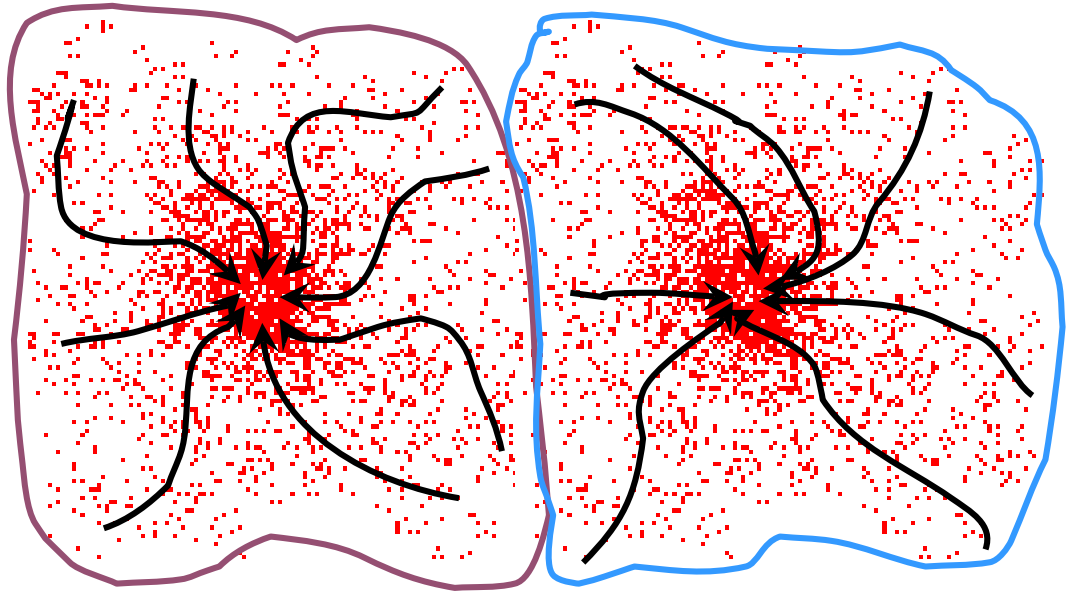
Main Idea: find **modes** or local density maxima in the feature space

For each data point:

1. Define a window around it, compute the centroid
2. Shift the center of the window to the centroid
3. Repeat until the centroid stops moving (convergence).

Clustering with Mean Shift

- Attraction basin: the region in feature space for which all trajectories of centroids lead to the same mode
- Cluster: all data points in the attraction basin of a mode

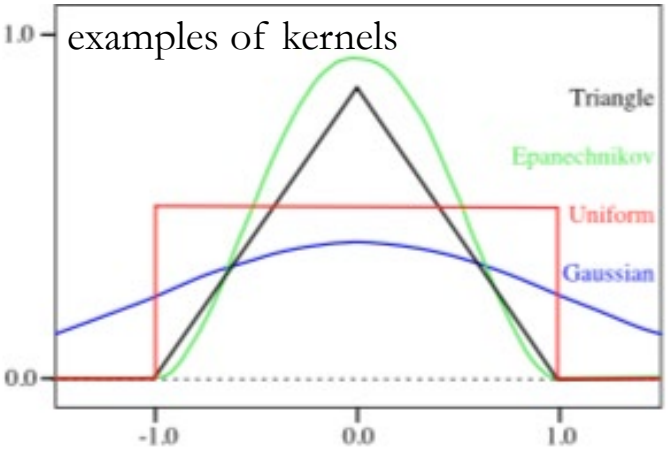


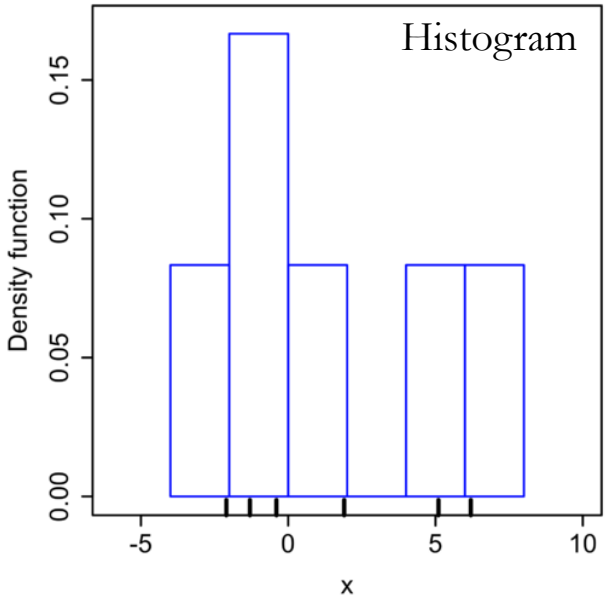
Slide Credit: Y. Ukrainitz & B. Sarel

Kernel Density Estimation

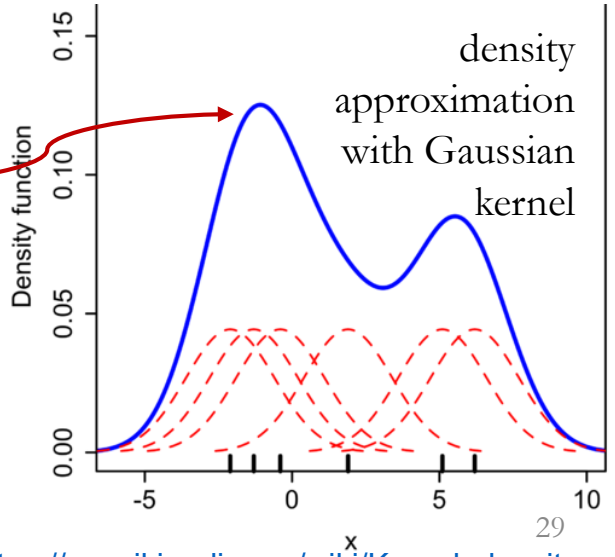
Sample	1	2	3	4	5	6
Value	-2.1	-1.3	-0.4	1.9	5.1	6.2

- Assume that every single data point is a sample drawn from some distribution with an unknown density function *f*
- Kernel density estimation tries to estimate the shape of the function *f*. The estimated shape depends on the kernel used and the kernel bandwidth (window size).
- Kernels can be uniform, Gaussian, …





Mean shift tries to find the local maxima in this density.



Images from https://en.wikipedia.org/wiki/Kernel_density_estimation

The Maths of Mean Shift (1)

Data is d-dimensional; so density function is also d-dimensional.

Given n data points $\mathbf{x}_i \in \mathbb{R}^d$, the multivariate kernel density estimate using a radially symmetric kernel[1] (e.g., Epanechnikov and Gaussian kernels), $K(\mathbf{x})$, is given by,

$$\hat{f}_K = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right), \quad (1)$$

Approximated density is a summation of the kernels centered on each point $\mathbf{x}_i$.

where h (termed the *bandwidth* parameter) defines the radius of kernel. The radially symmetric kernel is defined as,

$$K(\mathbf{x}) = c_k k(\|\mathbf{x}\|^2), \quad (2)$$

where c_k represents a normalization constant.

https://courses.csail.mit.edu/6.869/handouts/PAMIMeanshift.pdf

Comaniciu & Meer, 2002

The Maths of Mean Shift (2)

Taking the gradient ("derivative") of: $\hat{f}_K = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right)$

$$\nabla \hat{f}(\mathbf{x}) = \underbrace{\frac{2c_{k,d}}{nh^{d+2}} \left[\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right)\right]}_{\text{term 1}} \underbrace{\left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)} - \mathbf{x}\right]}_{\text{term 2}}, \qquad (3)$$

We want this to be equal to 0

where $g(x) = -k'(x)$ denotes the derivative of the selected kernel profile.

Proportional to density estimate at x; Unlikely to be 0

Only option is for this term to be 0.

$$x = \frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}$$

But how do we solve this? Incrementally!

https://courses.csail.mit.edu/6.869/handouts/PAMIMeanshift.pdf

Comaniciu & Meer, 2002

Mean-Shift Procedure

We want $x = \frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}$

Because $\nabla f(\mathbf{x}) = 0.$

For each point $\mathbf{x}_j$, where j = 1…n

1. Initialize density window $\mathbf{x}$: $\mathbf{x} = \mathbf{x}_j$
2. Computer mean shift vector $\mathbf{m}$:

Mean shift vector is a function of the bandwidth h and kernel G. These are expressed as subscripts to keep notation tidy.

n total data points; indexed by i

$$\mathbf{m}_{h,G}(\mathbf{x}) = \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)} - \mathbf{x}\right]$$

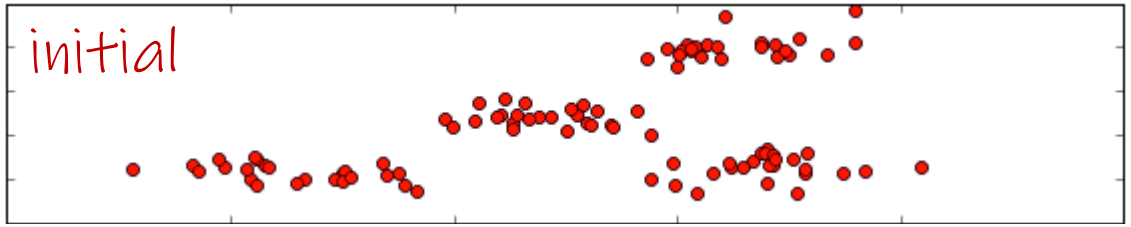
3. Shift density window and update: $\mathbf{x}' = \mathbf{x} + \mathbf{m}(\mathbf{x})$, $\mathbf{x} = \mathbf{x}'$

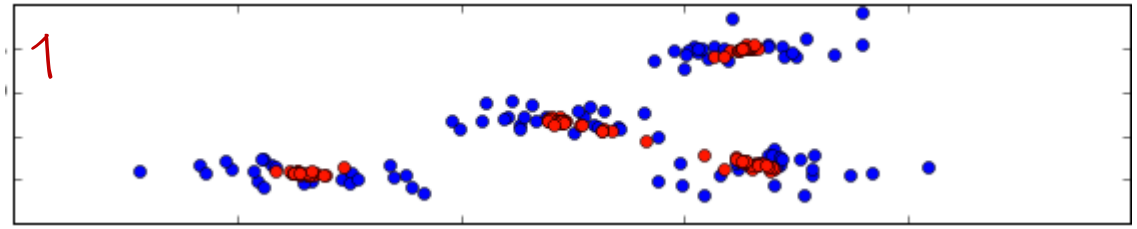
4. Iterate steps 2 and 3 until convergence.

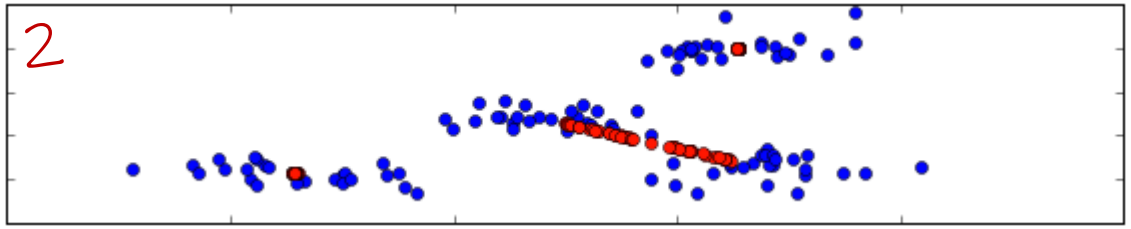
https://courses.csail.mit.edu/6.869/handouts/PAMIMeanshift.pdf

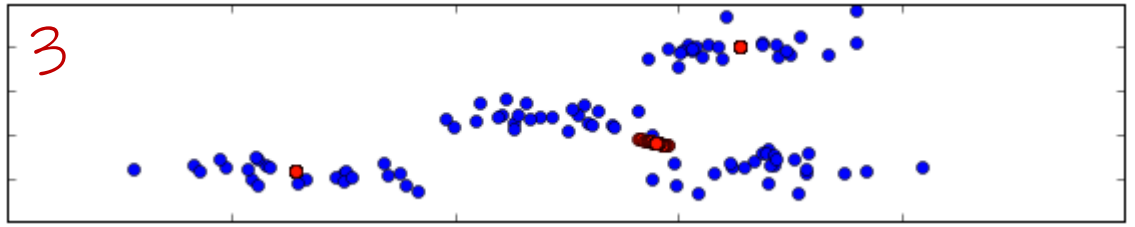
Comaniciu & Meer, 2002

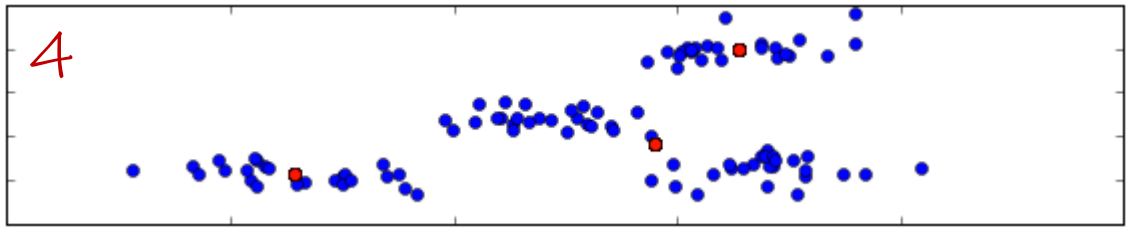
Mean Shift Example

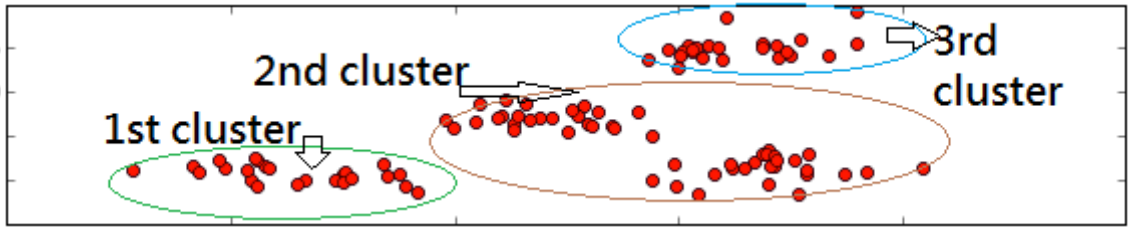












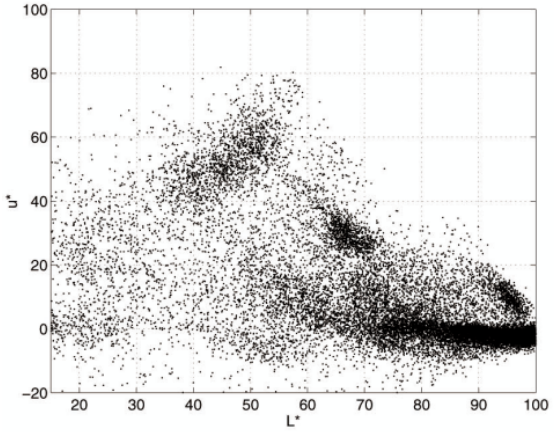
HWQ: The second cluster looks like it should actually be two clusters. How should parameter h be adjusted to encourage this?

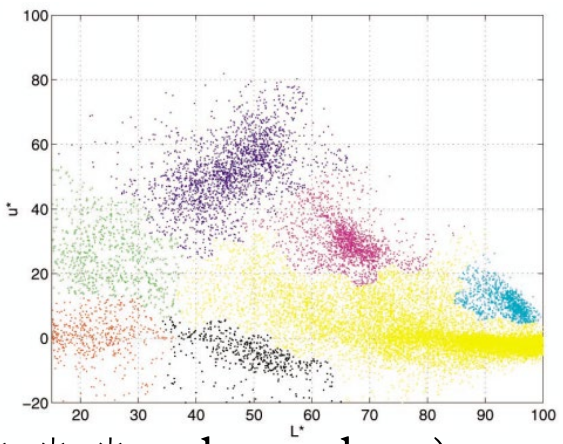
http://www.chioka.in/meanshift-algorithm-for-the-rest-of-us-python/

Segmentation with Mean Shift

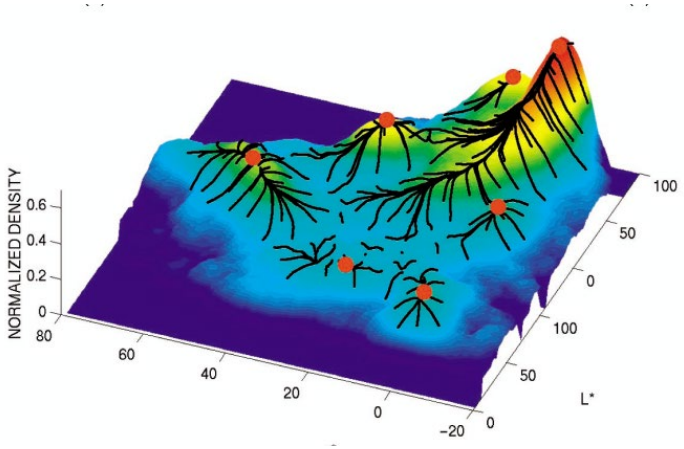
1. Find features (color, gradients, texture, etc) to represent each pixel.
2. Initialize density windows at individual feature points.
3. Perform mean shift for each point until convergence.
4. Merge pixels whose feature points that end up near the same mode or "peak" into the same segment.

Image





Feature space: (L*u*v* color values)



Adapted from K. Grauman

Mean shift segmentation results

https://courses.csail.mit.edu/6.869/handouts/PAMIMeanshift.pdf

Selecting the Bandwidth or Window Size

Original

Increasing bandwidth →

HWQ: what happens to the number of clusters as the window size is increased? Explain your answer.

- Trial and error ☹
- Sample some points, and use an average distance to the k-nearest neighbours
  - Number of neighbours needs to be large enough to ensure that there is an increase in density within the window as the algorithm progresses

Mean-Shift: Pros & Cons

Pros

- General algorithm for mode-finding (we apply it towards segmentation)
- No prior assumptions on cluster shape (spherical, elliptical, etc.)
- One parameter (window size / bandwidth *h*, which has a physical meaning)
- Finds variable number of modes (vs. pre-specified k in k-means)
- Robust to outliers

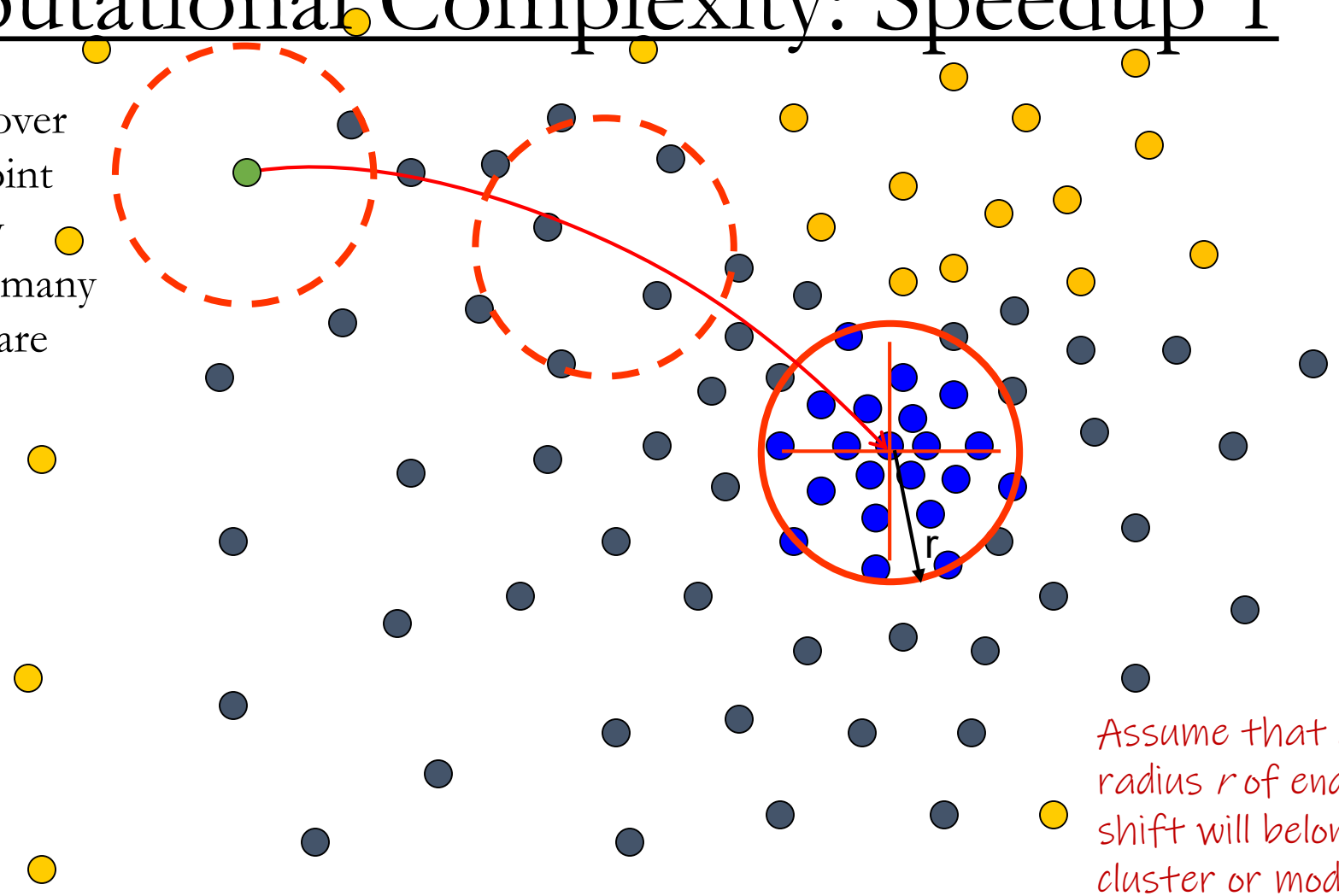
Cons

- Output depends on *h* and selecting *h* is non-trivial
- Computationally expensive and slow to run!
- Scales poorly with feature space dimension

Adapted from S. Lazebnik

Computational Complexity: Speedup 1

Need to loop over every single point and shift many windows. Yet many computations are redundant.

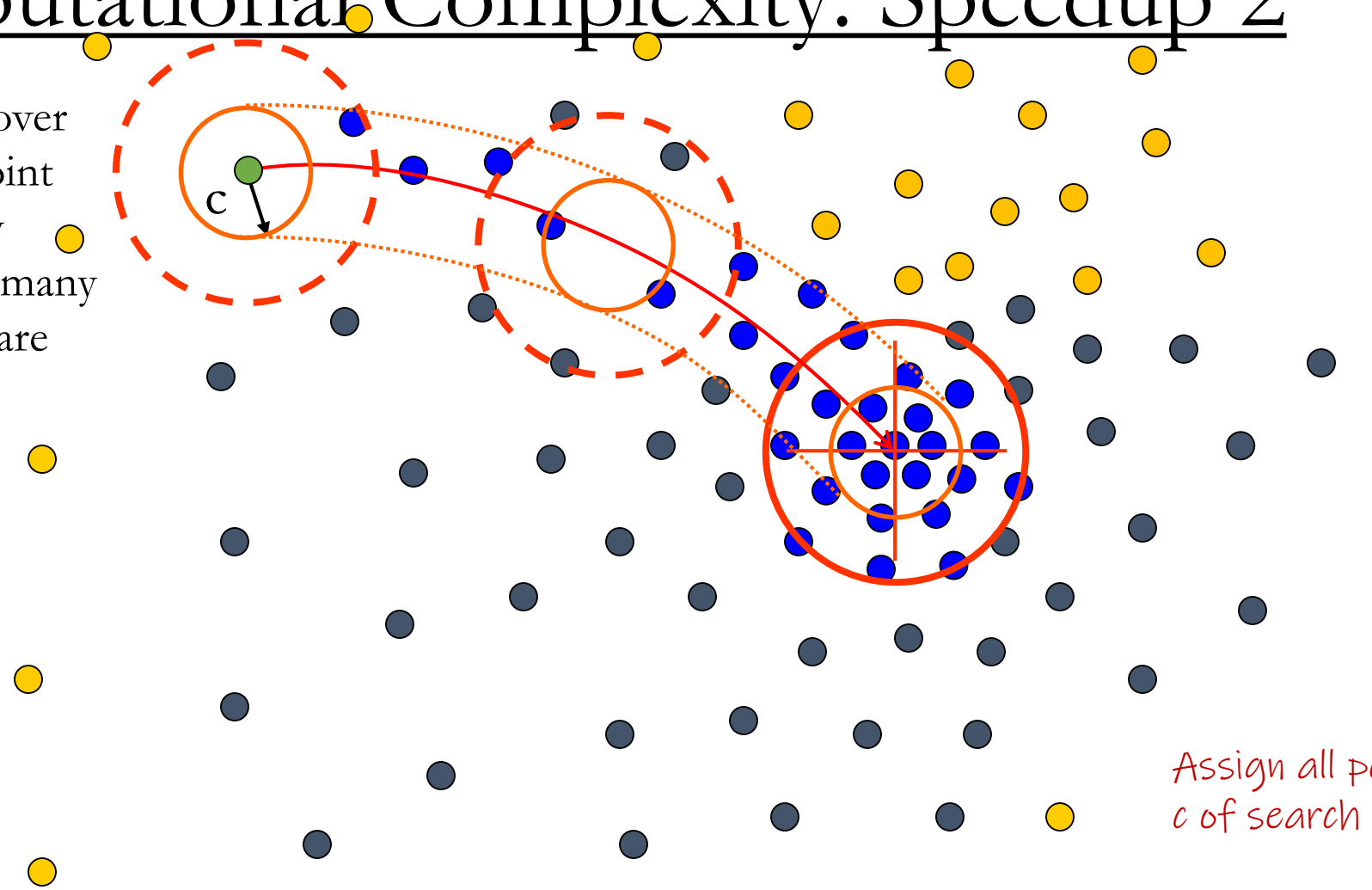


Assume that all points within radius r of end point after mean shift will belong to the same cluster or mode. (discard from list or queue to run mean-shift)

Slide credit: B. Leibe

Computational Complexity: Speedup 2

Need to loop over every single point and shift many windows. Yet many computations are redundant.



Assign all points within radius c of search path to mode.

Slide credit: B. Leibe

Summary

- Segmentation separates image into coherent regions
- Regions can be found via clustering the pixels based on their features
  - Greyscale intensity, colour, colour + (x,y) position
- K-means clustering
  - iteratively assigns membership and cluster centers
- Mean-shift clustering
  - finds modes or local maxima of density in the feature space
  - computationally expensive but model-free alternative to k-means
- Superpixelling
  - SLIC is one superpixelling algorithm built upon k-means clustering