Segmentation

Segmentation

- Image segmentation is a long standing problem in vision
  - We want to divide an images into regions
  - The image within a region should be uniform in some way
  - Adjacent regions should be different in some way

- Segmentation is a case of clustering
  - Given a set of entities (eg pixels) we want to divide them into groups (eg regions)
  - Elements of each group should be

  similar
  - Elements of different groups should be different

Segmentation

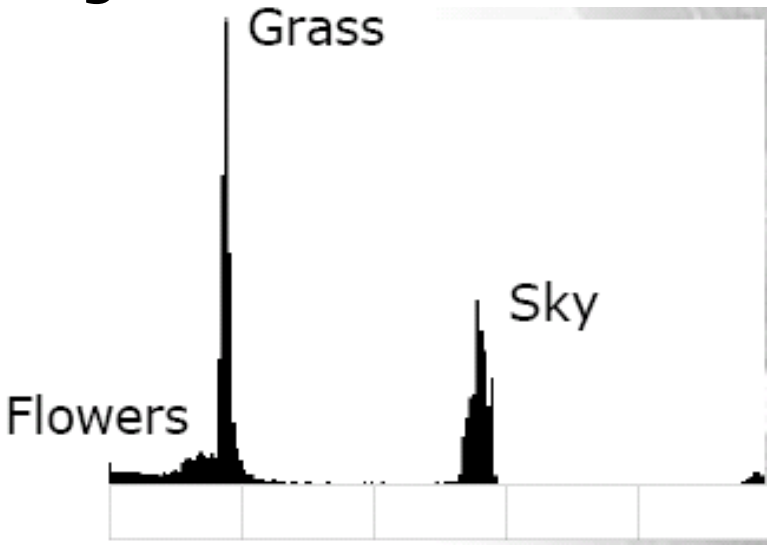
Image segmentation is a hard problem

- There is no 'correct solution – it depends very much on interpretation
- It is usually based on low-level cues (colour, texture, etc), but is trying to find high level things (objects)

Segmentation

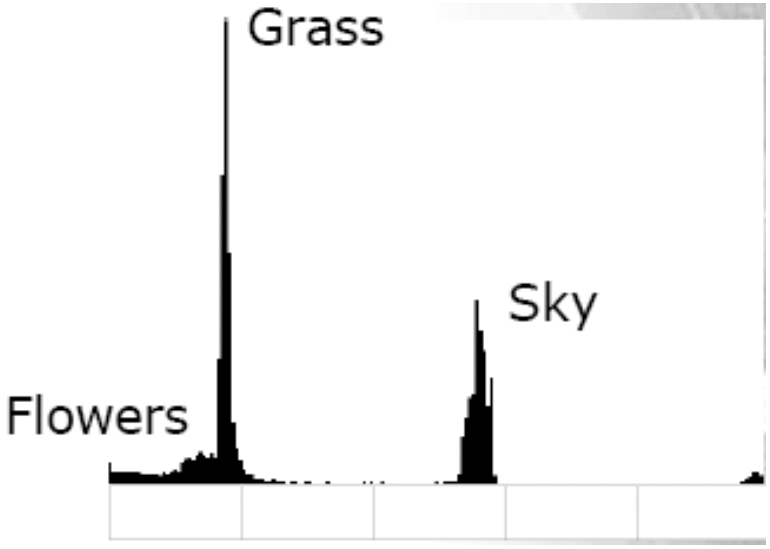
We find a histogram of some value we want to segment on

- Often intensity, or a colour metric
- We can use hue for the tree image since we have the green grass, flowers, and blue sky



Segmentation

- Assign a label based on pixel similarity to grass or sky



Questions: goo.gl/K61te5

Clustering/Segmentation

- Clustering
  - To apply *k*-means to the image
  - We start with three hue estimates – say 0, 120, 240 which are red, green and blue
  - We cluster the pixels with *k*-means, being careful to remember that hue is a cyclic quantity

Clustering/Segmentation

Clustering, such as with *k*-means

- Finds groups of pixels with similar properties
- Doesn't guarantee that these groups form continuous areas in the image
- Even if it does the edges of these areas tend to be uneven

Region-Based Segmentation

We want smooth regions in the image

- We still want the pixels in each region to be similar, and those in adjacent regions to be different
- One way to do this is to work with regions rather than pixels

Region growing

- Start with a small 'seed' and expand by adding similar pixels
- Split and merge
  - Splitting divides regions that are inconsistent
  - Merging combines adjacent regions that are consistent

Region Growing

Region growing starts with a small patch of seed pixels

- Compute statistics about the region
- Check neighbours to see if they can be added
- Recompute the statistics

This procedure repeats until the region stops growing

- Simple example: We compute the mean grey level of the pixels in the region
- Neighbours are added if their grey level is near the average

Region Growing




Split and Merge - Split

We start by taking the whole image to be one region

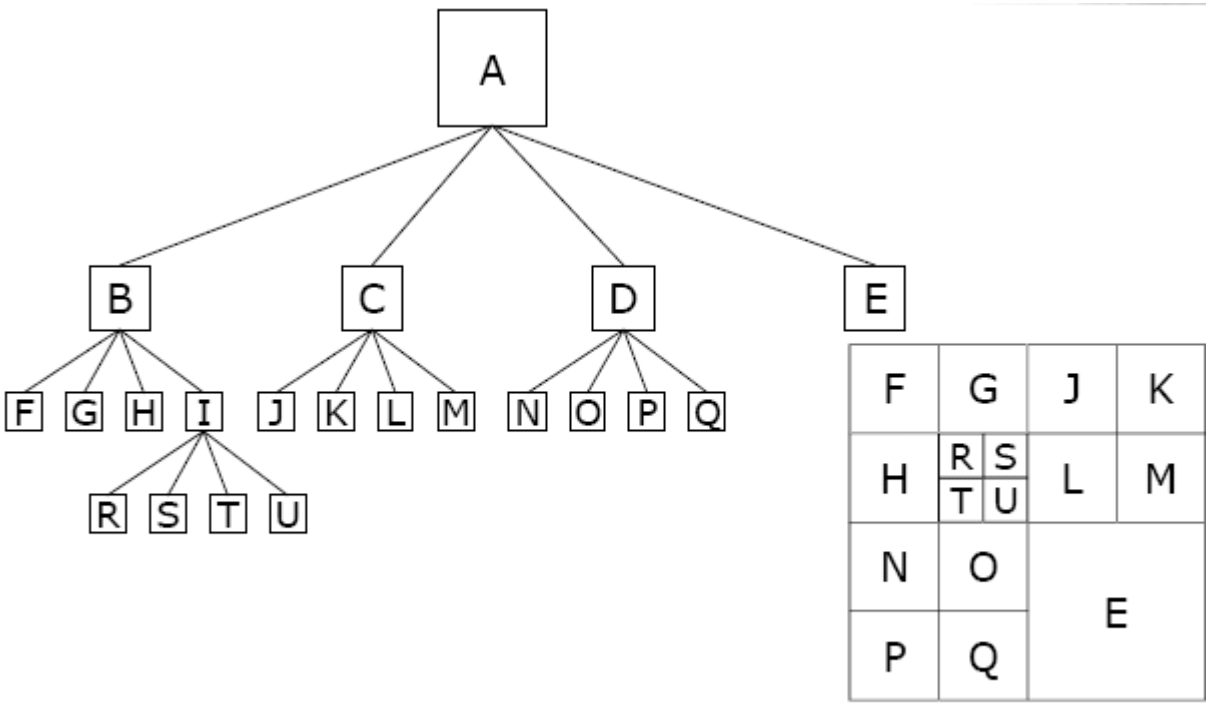
- We compute some measure of internal similarity
- If this indicates there is too much variety, we divide the region
- Repeat until no more splits, or we reach a minimum region size

Some details are needed

- How to we measure similarity – standard deviations are commonly used
- How do we determine whether to split or not
  - thresholding is easy
- How do we split regions – quadtrees are a common method

Split and Merge - Split

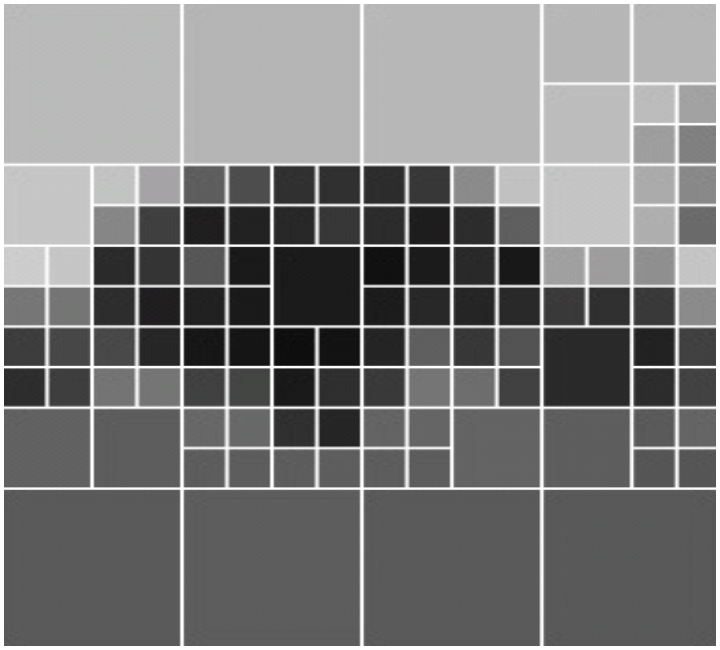
- Quadtrees



Split and Merge - Split

We'll use the tree image again

- Splitting based on intensity (could use something else)
- Splitting based on standard deviation, with a threshold of 25
- Split using a quadtree with a maximum of 5 levels



Split and Merge - Merge

Splitting gives us

- Regions that are small, consistent, or both
- Rather too many regions, as adjacent ones may be very similar
- We can now combine adjacent regions to make bigger ones

Merging

- We merge two regions if they are adjacent and similar
- Need a measure of similarity – can compare their mean grey level, or use statistical tests
- Repeat the merging until you can do no more

Split and Merge - Merge

- We consider merging adjacent regions
- Two regions are merged if their mean grey levels differ by less than 25
- This leads to less regularly shaped regions, but they are larger and still consistent

Split and Merge

Define a measure of internal region dissimilarity (e.g. entropy of colour histogram), a similarity threshold and minimum region size.

1. Start by taking the whole image to be one region.
2. Compute internal similarity.
3. Divide the region in 4 equal parts if the measure exceeds the threshold (indicates there is too much variety).
4. Repeat steps 2 and 3 for each part until there are no more splits, or we reach a minimum region size.
5. Compute internal similarity for each pair of adjacent regions at the bottom level of the tree.
6. Merge the regions if their similarity is higher than the threshold.
7. Repeat steps 5 and 6 until no merge is possible.

Normalised Cuts

Split and merge uses a regular division of the image

- This leads to a blocky final segmentation
- The regular division is what leads to an over segmentation by splitting, it forces e.g 4 sub-regions to be created, fewer or more may be better

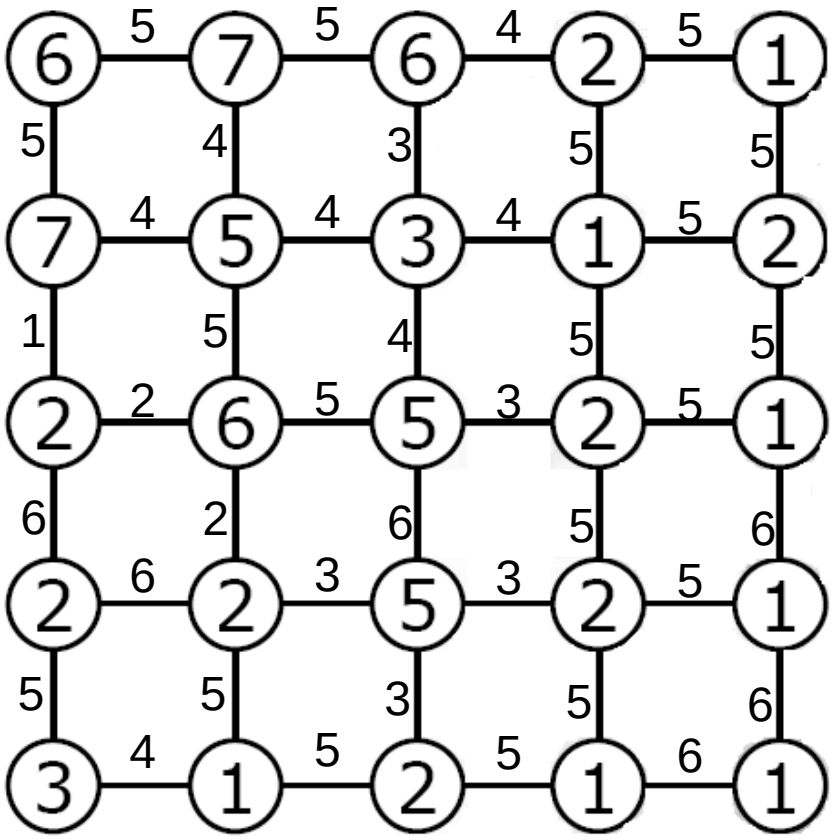
One alternative is normalised cuts

- It aims to do the splitting so that the division is optimal
- This gives better definition around the edges
- It (should) remove the need to merge regions after splitting

Normalised Cuts

Is based on graph theory

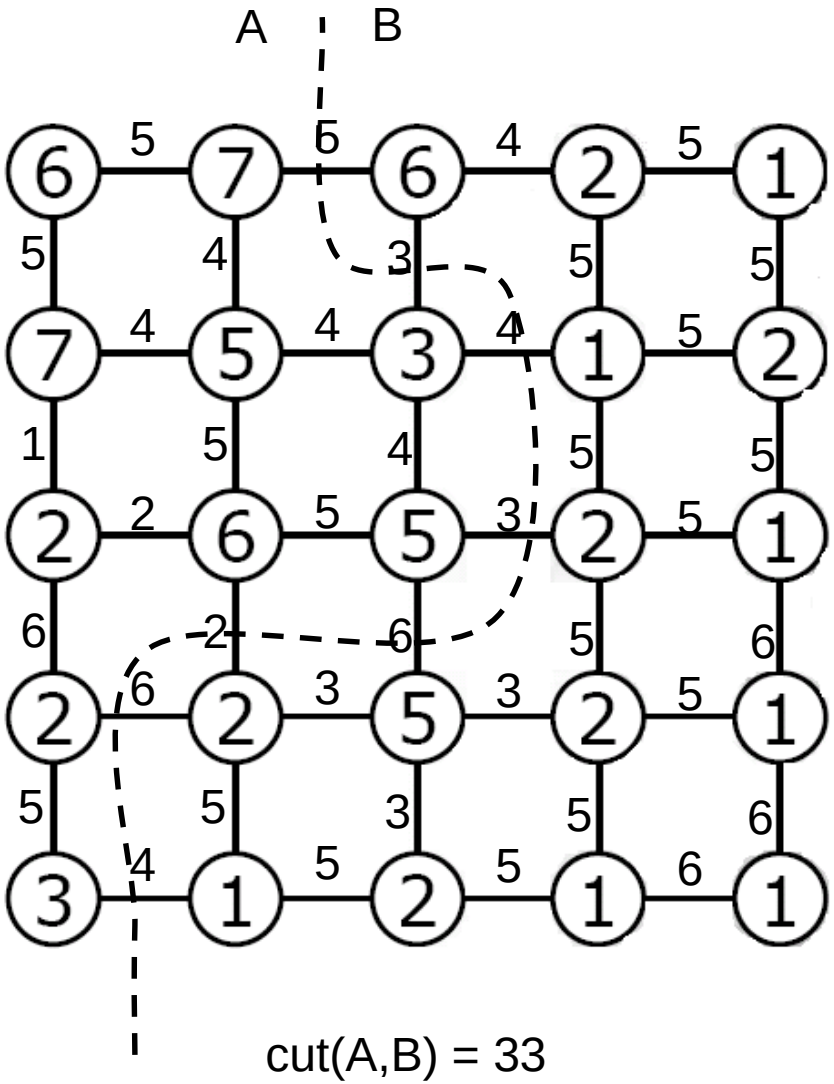
- Graphs are used to represent images
- Each pixel is a vertex in the graph, *vi*
- Edges, *e=(vi,vj)* link adjacent pixels, and are given weights so that if *vi* and *vj* are similar *w(vi, vj)* is large



Graph Cuts

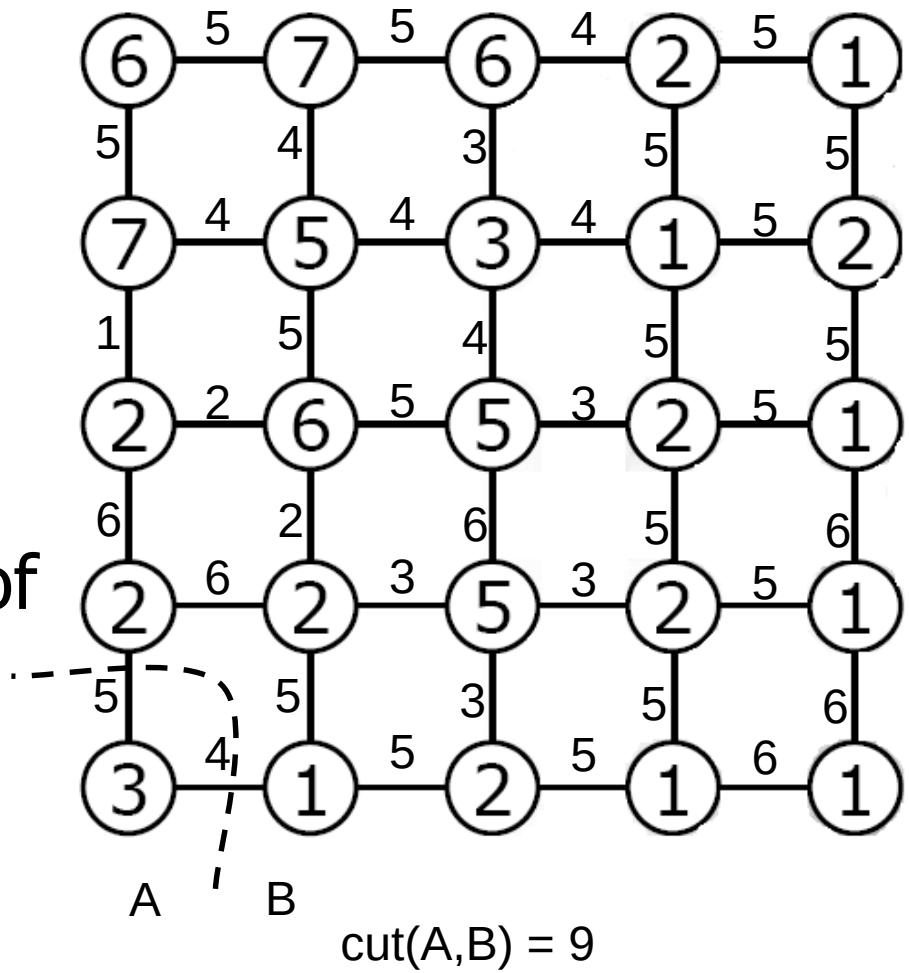
- A *cut* divides the vertices of a graph into 2 sets, *A* and *B*
- The weight of the cut is the sum of the edges between vertices in *A* and *B*

$$cut(A,B) = \sum_{u \in A,\, v \in B} w(u,v)$$



Cuts and Segmentation

- We divide on cuts to segment an image
- We want the cuts to have a low weight
- This means that they don't separate similar pixels
- However, the weight of a cut depends on its length, so short cuts have low weights



Normalised Cuts

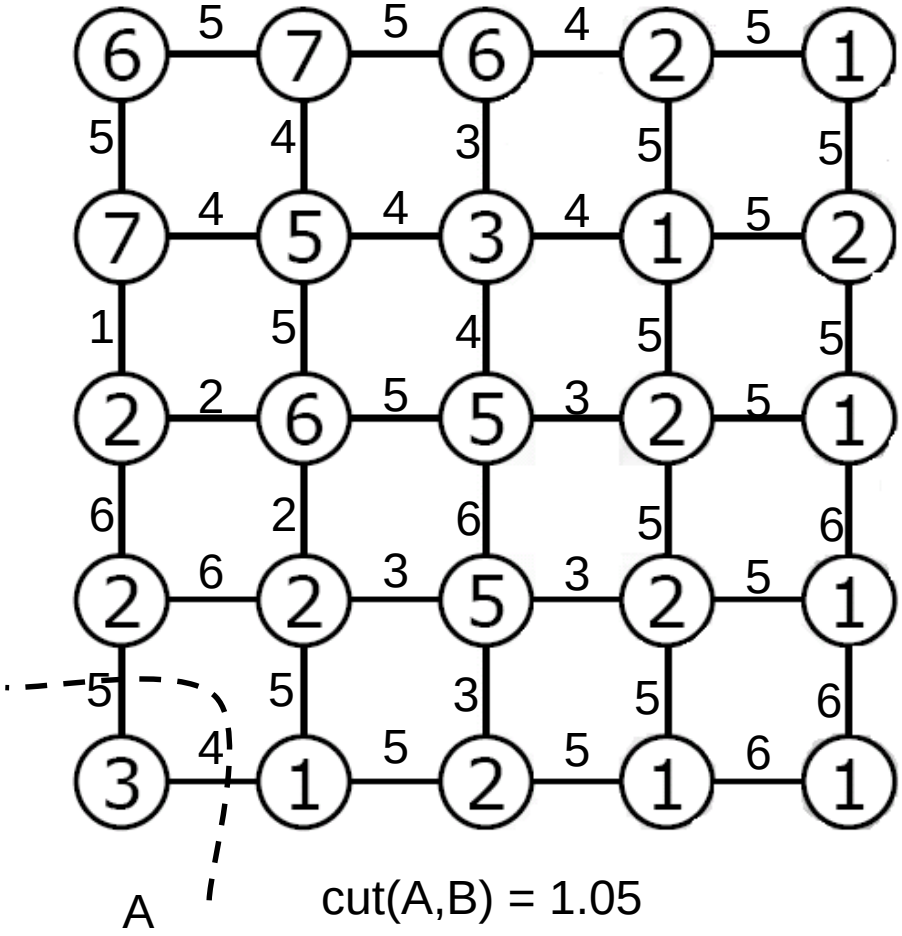
- A normalised cut removes this bias
- It also takes into account the total weights of edges in the two sets, called the association where V is the set of all vertices in the graph

$$assoc(A,V) = \sum_{u \in A,\, v \in V} w(u,v)$$

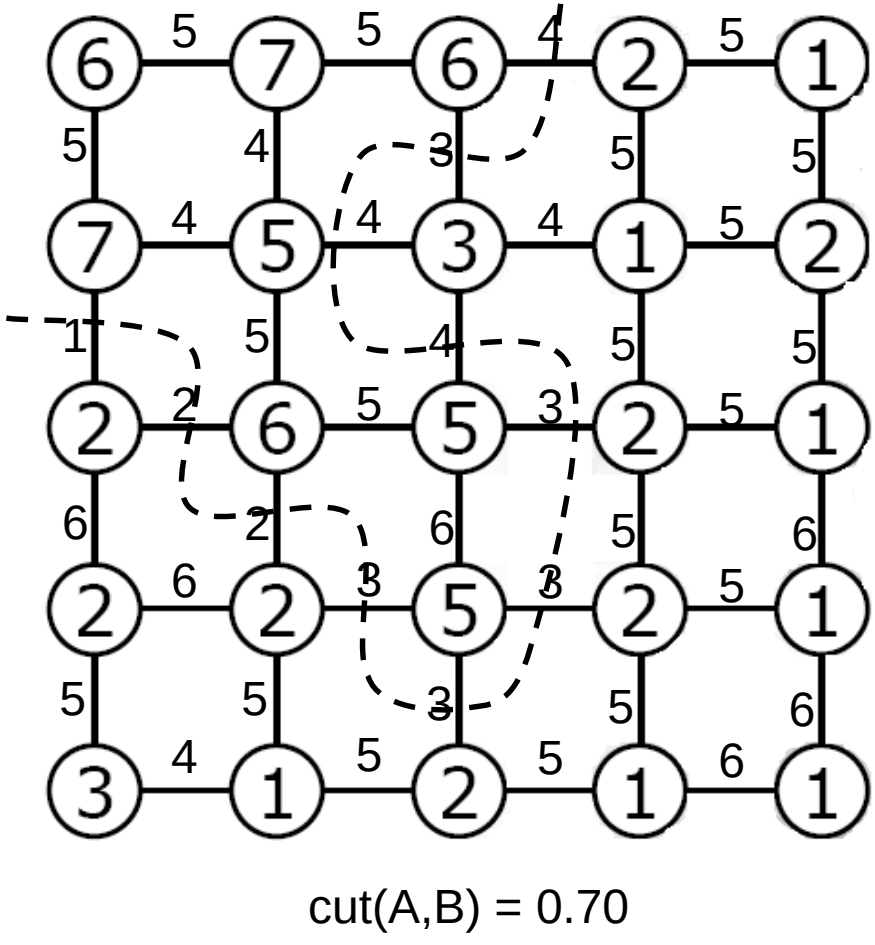
A split is now made along the normalised cut with the smallest weight, given by

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

Normalised Cuts



cut(A,B) = 1.05



cut(A,B) = 0.70

$$assoc(A,V) = \sum_{u \in A,\, v \in V} w(u,v)$$

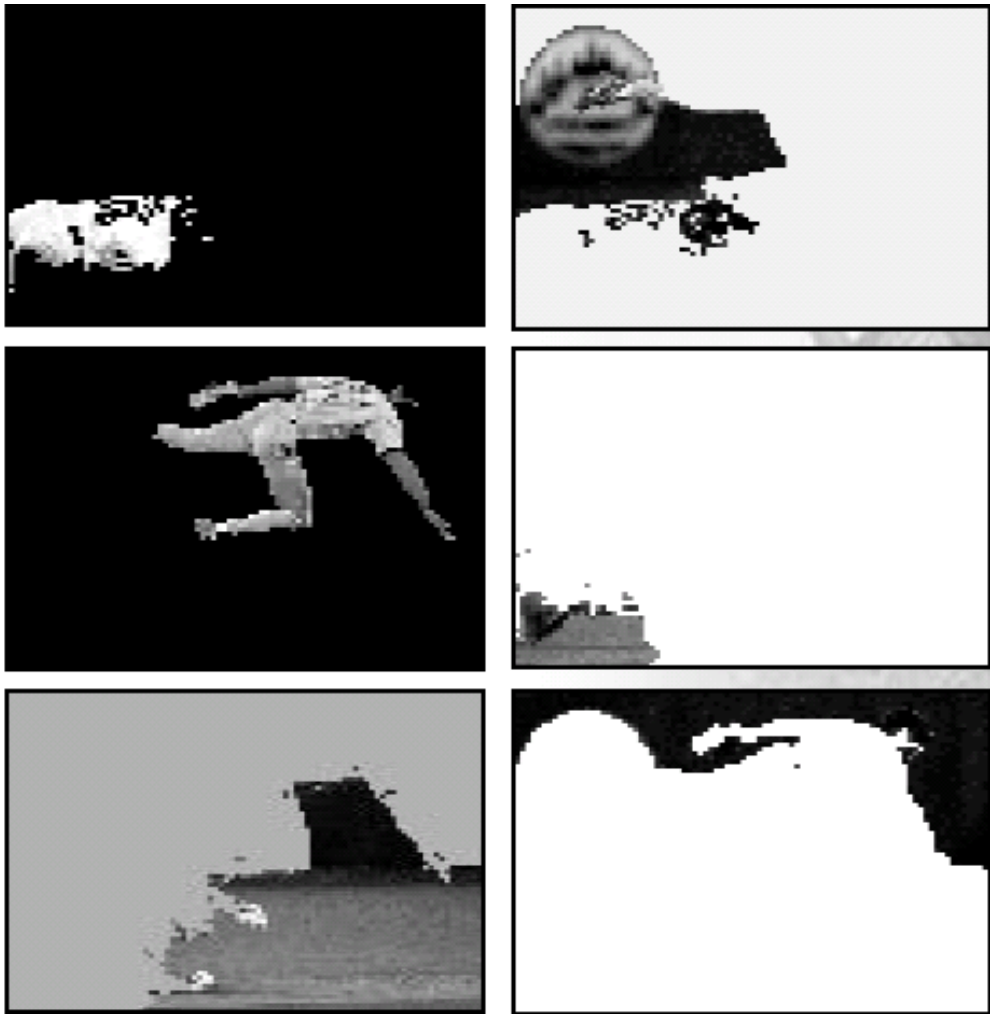
$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

Questions: goo.gl/K61te5

Computing Normalised Cuts

- We want to find a minimal cut
- It turns out that this is NP-complete so we need $O(2^n)$ time to solve it in general n here is the number of edges – about 3 billion for a 320x240 image, so 2^n is very large indeed
- An approximate solution is found
- This is based on a branch of maths called spectral graph theory

Normalised Cuts Example

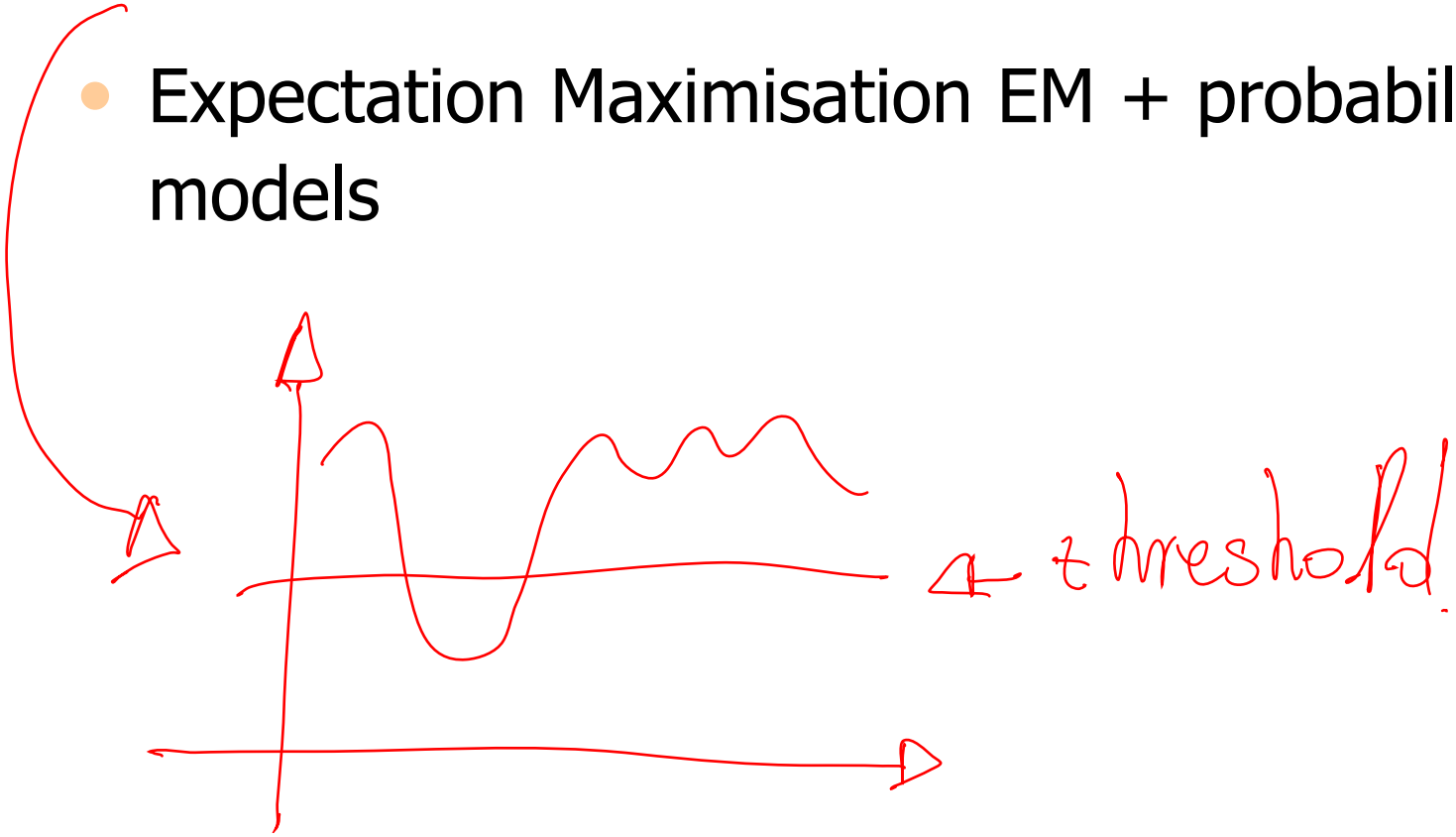


CNN segmentation

Questions: goo.gl/K61te5

Other Segmentation methods

- Watershed
- Expectation Maximisation EM + probabilistic models

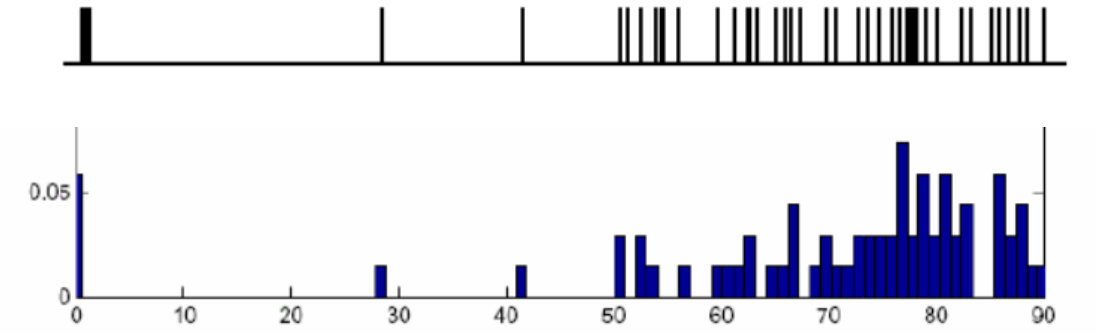


mean-shift

The mean-shift algorithm is an efficient approach to segmentation or tracking of objects whose appearance is defined by histograms.

Intro to Parzen Estimation
(Aka Kernel Density Estimation)

Mathematical model of how histograms are formed
Assume continuous data points

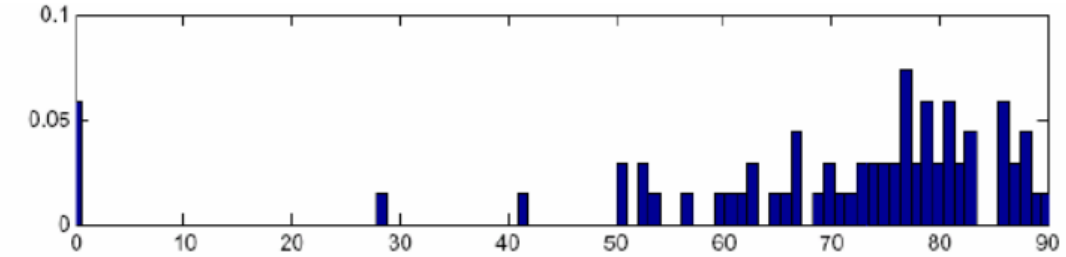


Convolve with box filter of width w (e.g. [1 1 1])
Take samples of result, with spacing w
Resulting value at point u represents count of data points falling in range u-w/2 to u+w/2

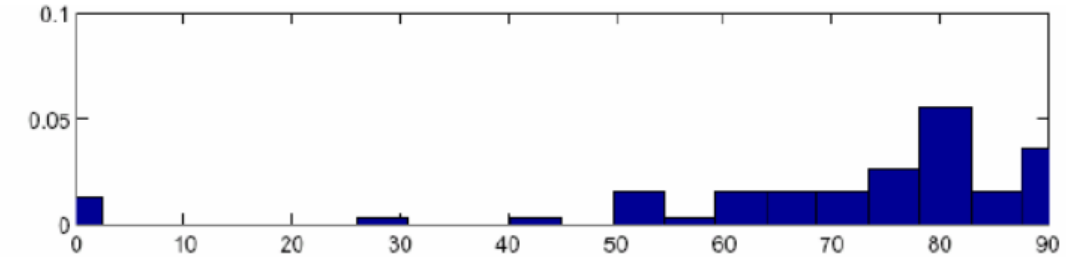
Example Histograms

Box filter

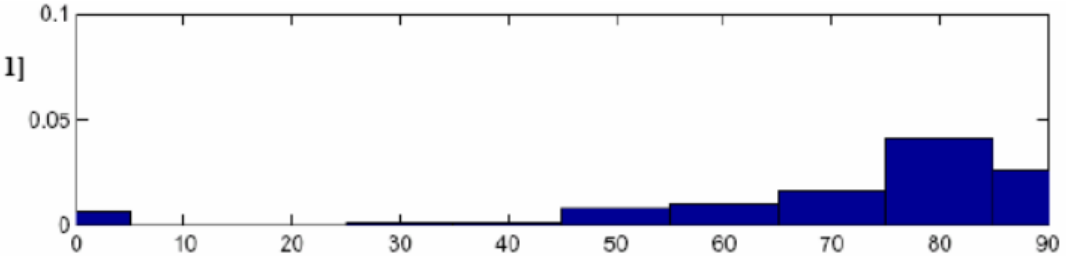
[1 1 1]



[1 1 1 1 1 1 1]



[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]



Bandwidth h controls the radius of influence of each data point.

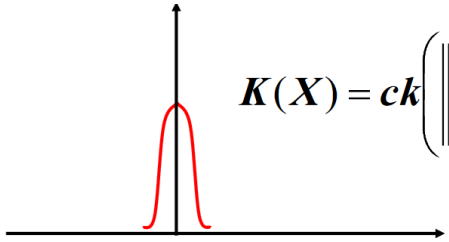
- Too small: Overfits the data points
- Too large: Smoothes out the details of the data

Questions: goo.gl/K61te5

Parzen window

Why Formulate it This Way?

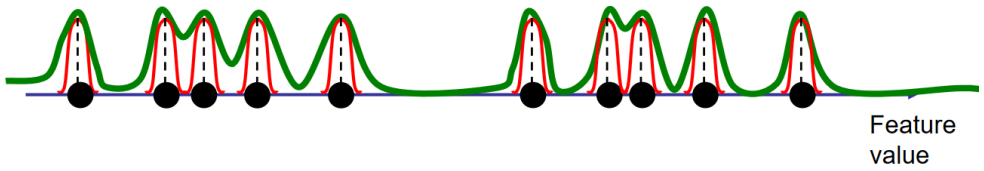
- Generalize from box filter to other filters (for example Gaussian)
- Gaussian acts as a smoothing filter.



$$K(X) = ck\left(\left\|\frac{X}{h}\right\|^2\right)$$

h too small: The pdf overfits the noise in the data → Too many modes

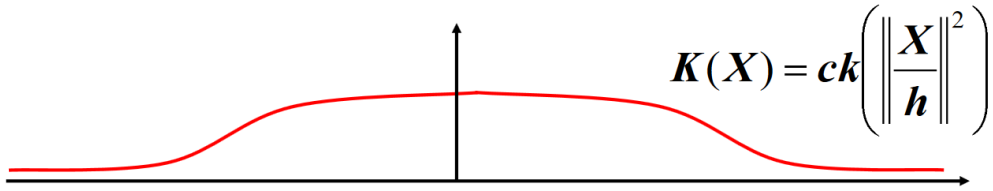
$$f(X) = \sum_i ck\left(\left\|\frac{X - X_i}{h}\right\|^2\right)$$



Parzen window

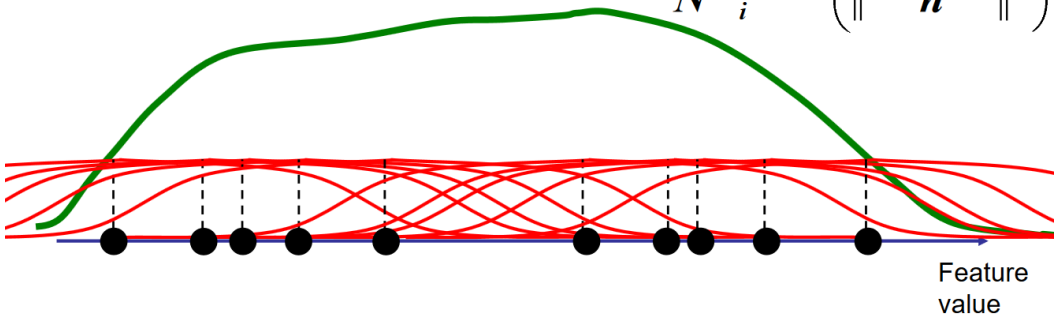
Why Formulate it This Way?

- Generalize from box filter to other filters (for example Gaussian)
- Gaussian acts as a smoothing filter.



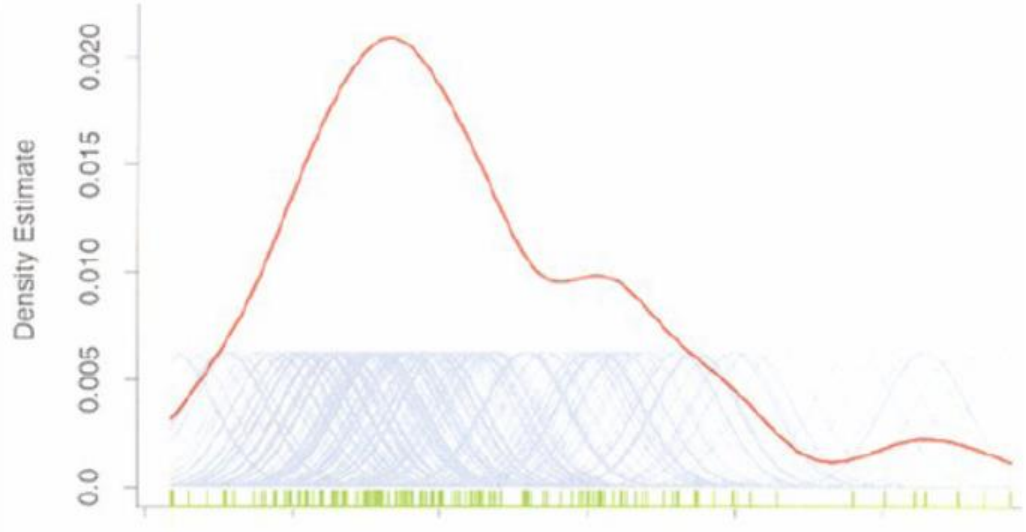
h too large: The details of the initial data are smoothed out
→ Too few modes

$$f(X) = \frac{1}{N}\sum_i ck\left(\left\|\frac{X - X_i}{h}\right\|^2\right)$$



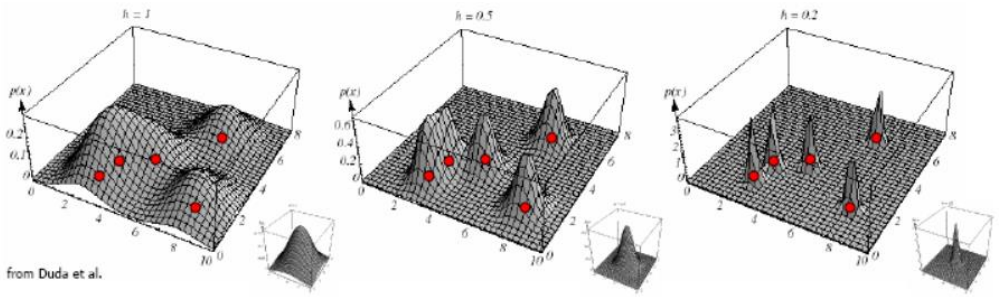
Kernel Density Estimation

- **Parzen windows**: Approximate probability density by estimating local density of points (same idea as a histogram)
  - Convolve points with window/kernel function (e.g., Gaussian) using scale parameter (e.g., sigma)



Density Estimation at Different Scales

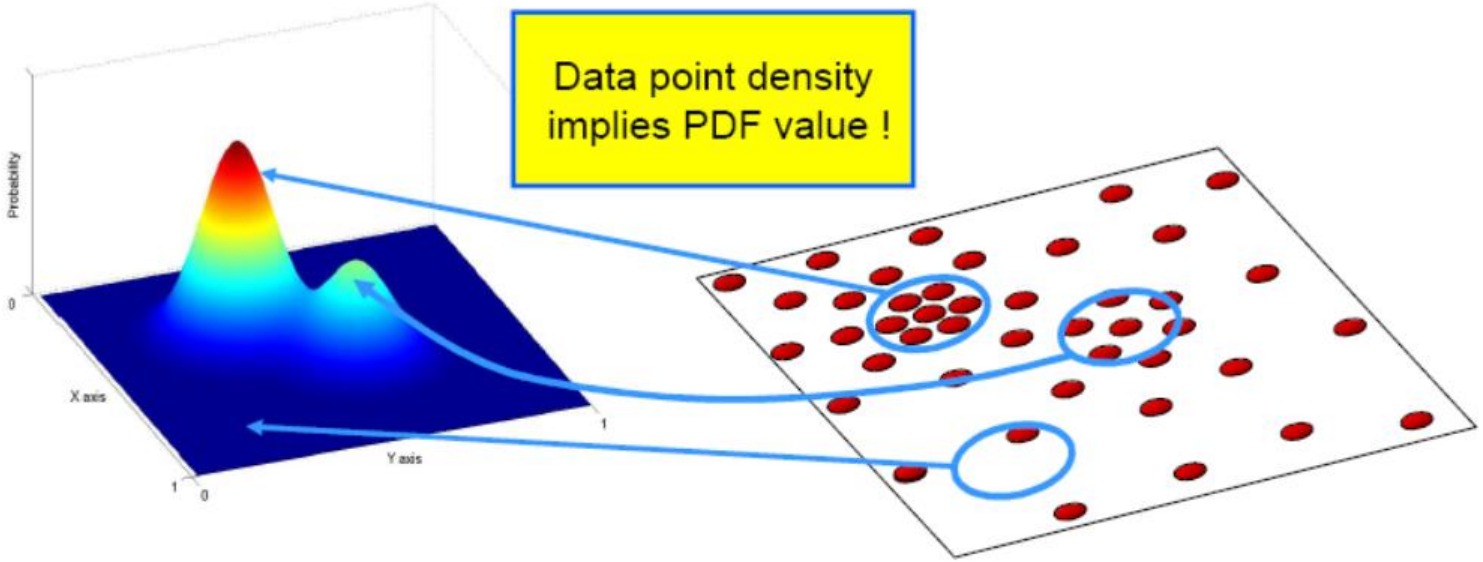
- Example: Density estimates for 5 data points with differently-scaled kernels
- Scale influences accuracy vs. generality (overfitting)



- Unresolved issue: how to pick the scale (sigma value) of the smoothing filter
- Answer for now: a user-settable parameter

Non-Parametric Density Estimation

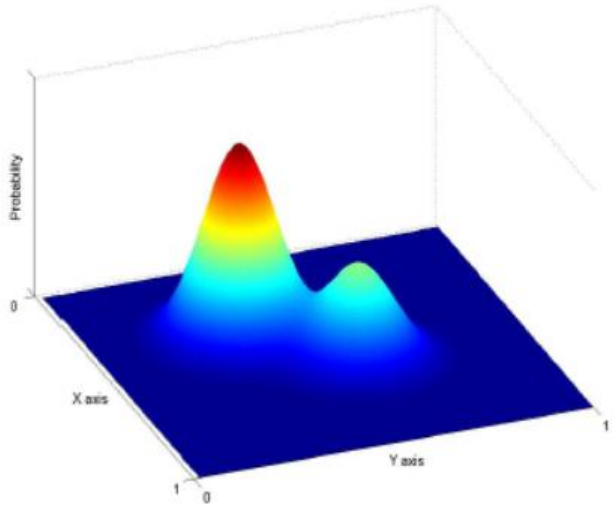
Assumption : The data points are sampled from an underlying PDF



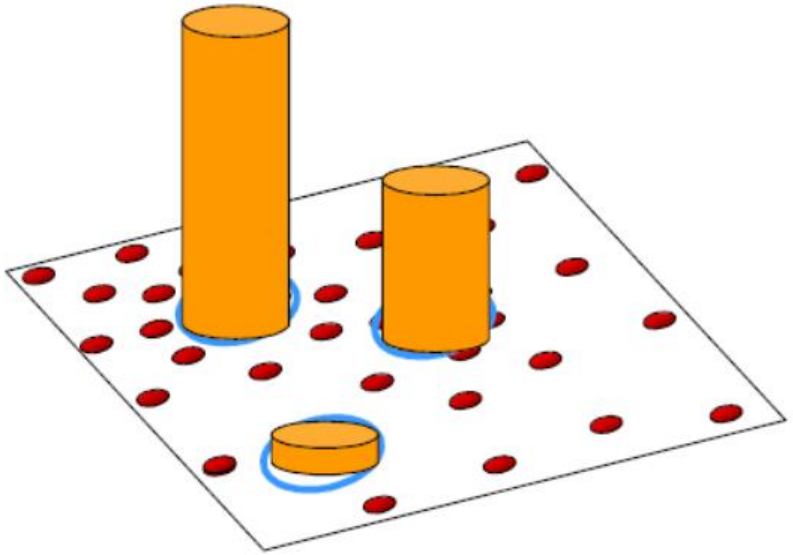
Assumed Underlying PDF

Real Data Samples

Non-Parametric Density Estimation



Assumed Underlying PDF



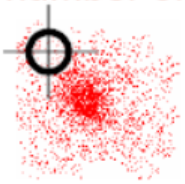
Real Data Samples

Kernel Density Estimation

Parzen Windows - General Framework

$$P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K(\mathbf{x} - \mathbf{x}_i)$$

A function of some finite number of data points $x_1 \ldots x_n$



Data

Kernel Properties:

- Normalized $\int_{R^d} K(\mathbf{x})d\mathbf{x} = 1$
- Symmetric $\int_{R^d} \mathbf{x}K(\mathbf{x})d\mathbf{x} = 0$
- Exponential weight decay $\lim_{\|\mathbf{x}\|\to\infty} \|\mathbf{x}\|^d K(\mathbf{x}) = 0$

$$\int_{R^d} \mathbf{x}\mathbf{x}^T K(\mathbf{x})d\mathbf{x} = c\mathbf{I}$$

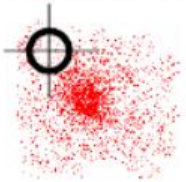
The extent of the kernel is the same along all dimensions

Kernel Density Estimation

Parzen Windows - Function Forms

$$P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K(\mathbf{x} - \mathbf{x}_i)$$

A function of some finite number of data points $x_1 \ldots x_n$



Data

In practice one uses the forms:

$$K(\mathbf{x}) = c\prod_{i=1}^{d} k(x_i) \quad \text{or} \quad K(\mathbf{x}) = ck(\|\mathbf{x}\|)$$

Same function on each dimension

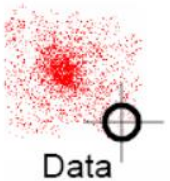
Function of vector length only

Kernel Density Estimation

Various Kernels

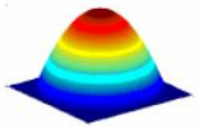
$$P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K(\mathbf{x} - \mathbf{x}_i)$$

A function of some finite number of data points $x_1 \ldots x_n$

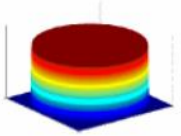


Examples:

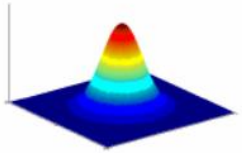
- Epanechnikov Kernel $K_E(\mathbf{x}) = \begin{cases} c\left(1 - \|\mathbf{x}\|^2\right) & \|\mathbf{x}\| \le 1 \\ 0 & \text{otherwise} \end{cases}$



- Uniform Kernel $K_U(\mathbf{x}) = \begin{cases} c & \|\mathbf{x}\| \le 1 \\ 0 & \text{otherwise} \end{cases}$

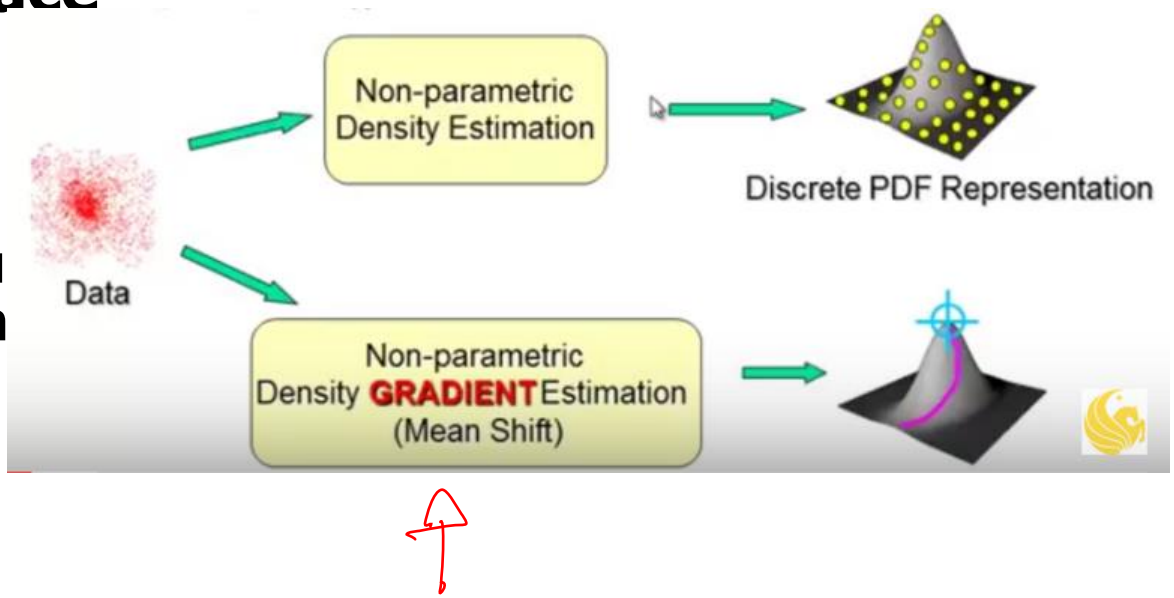


- Normal Kernel $K_N(\mathbf{x}) = c \cdot \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)$

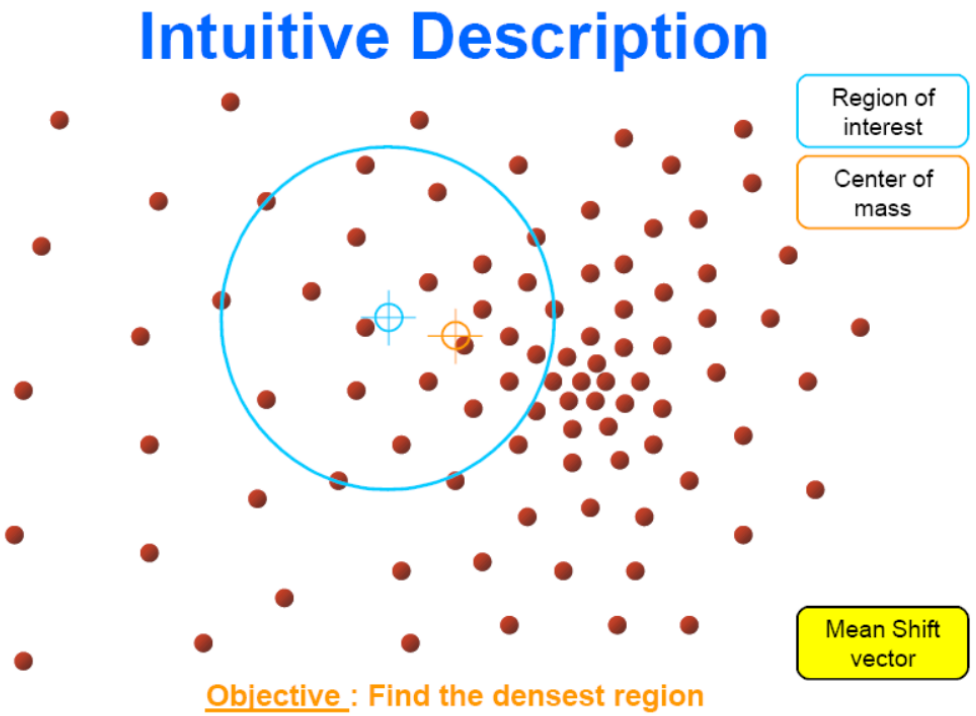


What is Mean Shift ?

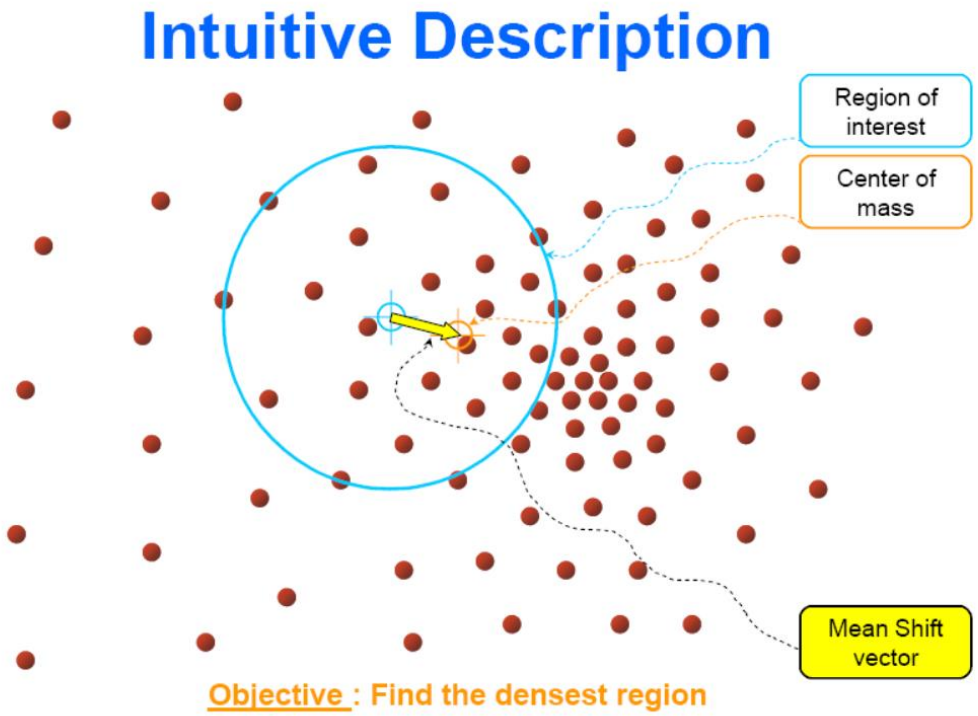
- **A tool for:**
  - **Finding modes in a set of data samples, manifesting an underlying probability density function (PDF) in R^N**
- **PDF in feature space**
  - **Color space**
  - **Scale space**
  - **Actually any featu space you can con**



What is Mean Shift ?

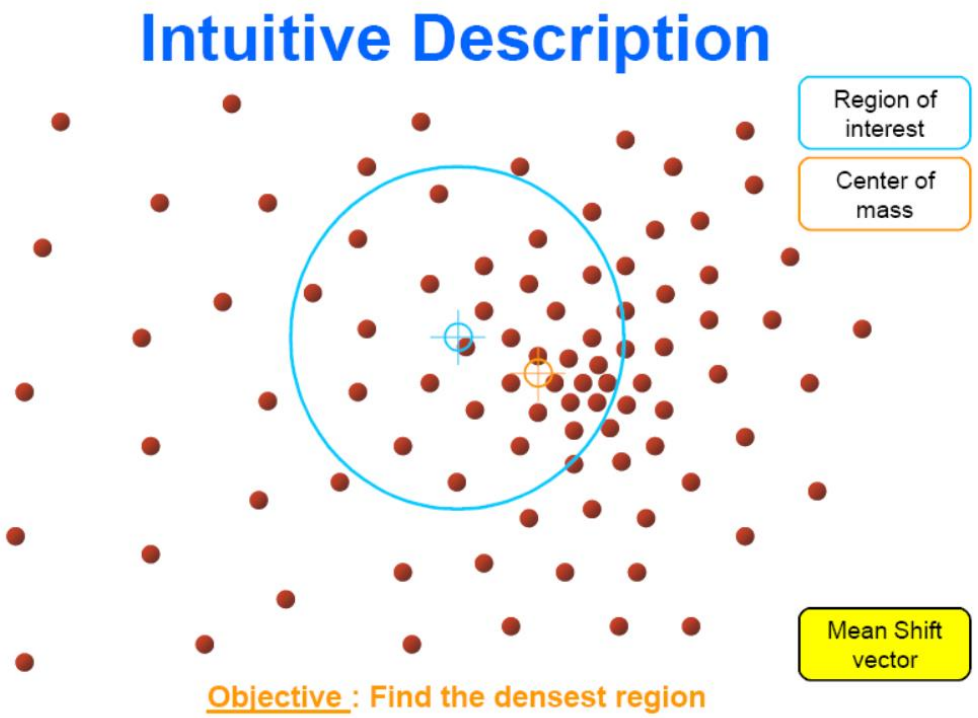


What is Mean Shift ?

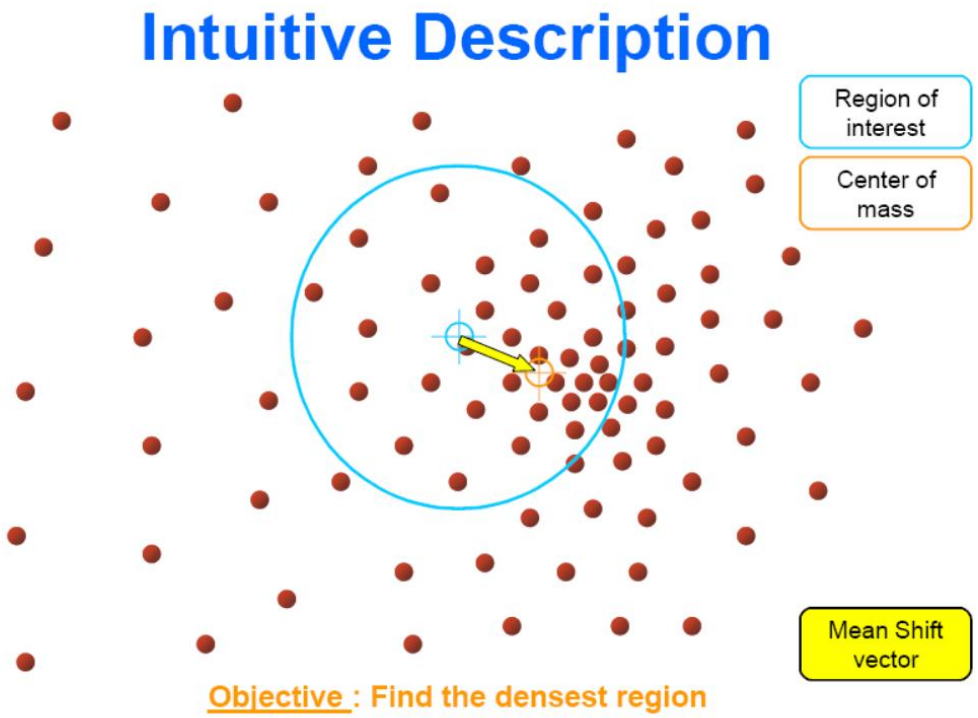


Questions: goo.gl/K61te5

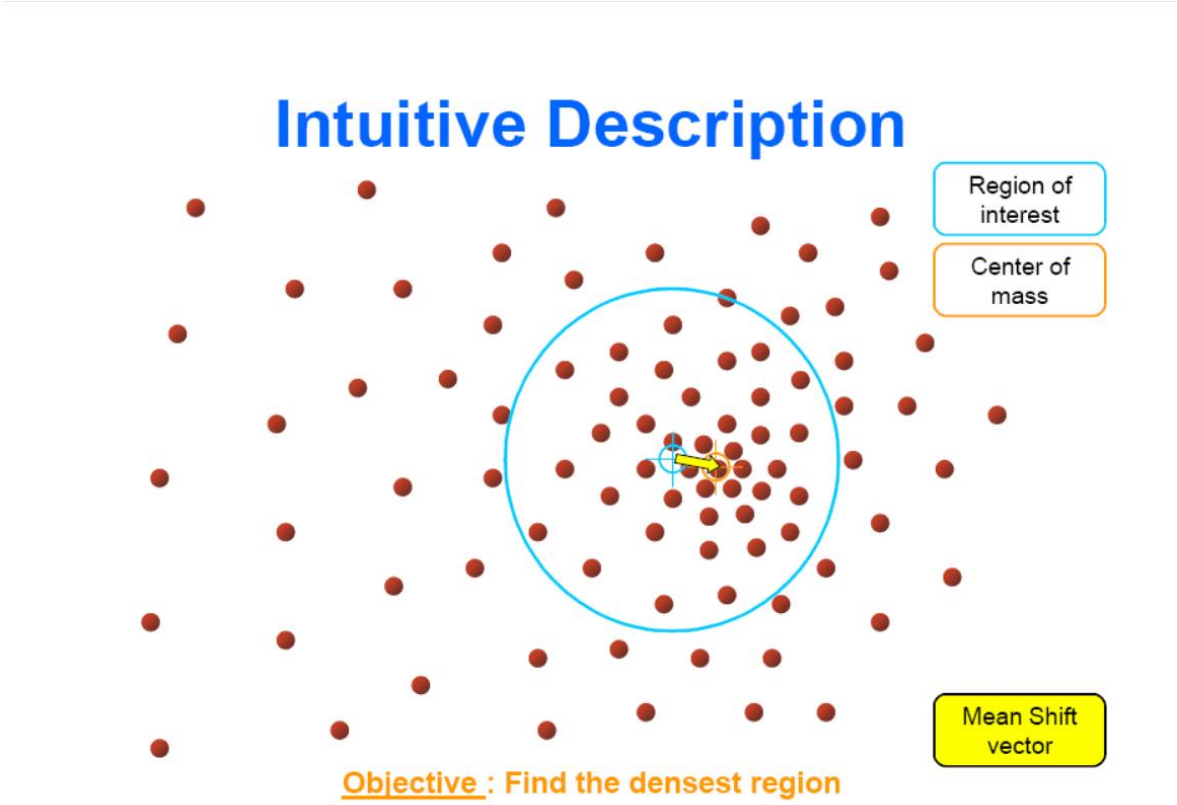
What is Mean Shift ?



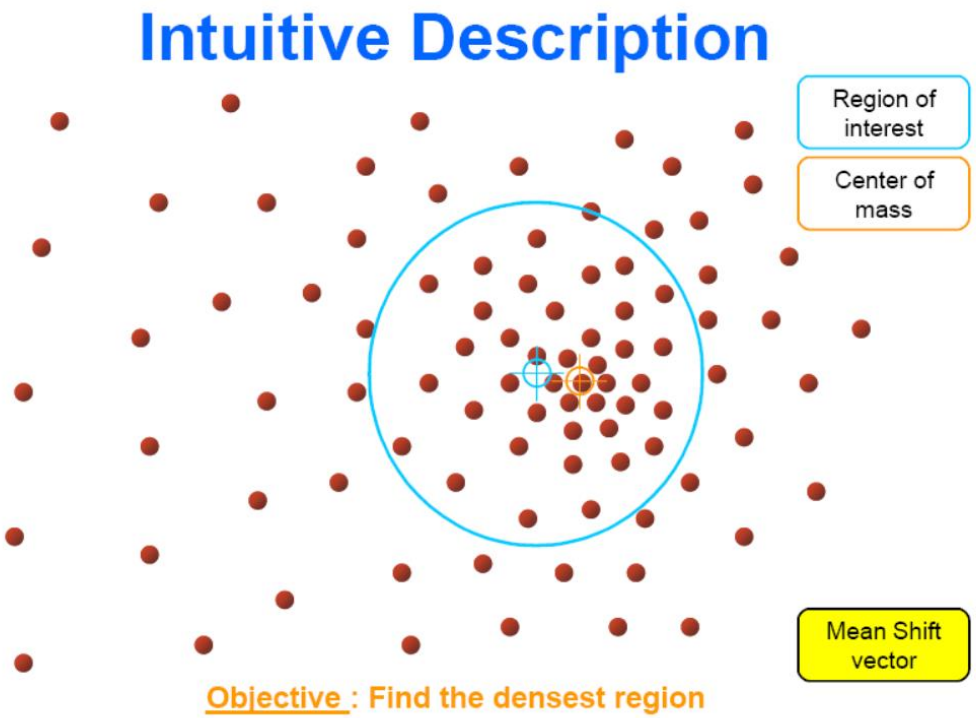
What is Mean Shift ?



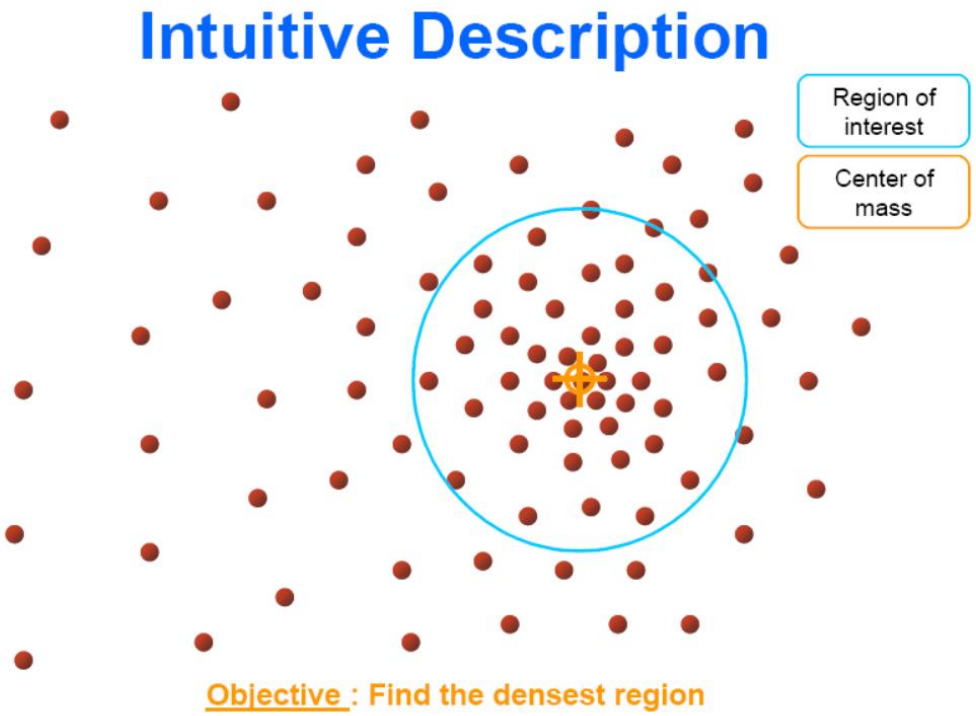
What is Mean Shift ?



What is Mean Shift ?



What is Mean Shift ?



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
- When data are clustered, PDF is large
- Find local maxima in this probability distribution

- Problem:
  - Don't know the distribution, must build a model
    - Place a small window on top of each data point
      - how big? adjust to get best model
      - Add windows, normalize

Data density

- Gaussian model of data density
  - x for data, h for a scale parameter, d is dimension

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

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

$$k(u) = \exp\left(-\tfrac{1}{2}u\right) \qquad C = \frac{(2\pi)^{(-d/2)}}{nh^d}$$

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

Goal: find density maximum

- Move in direction of density gradient

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

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

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

$$y^{(j+1)} = \frac{\sum_i x_i g(\left\|\frac{x_i - y^{(j)}}{h}\right\|^2)}{\sum_i g(\left\|\frac{x_i - y^{(j)}}{h}\right\|^2)}$$

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

$\neq 0$

$$y = \frac{\sum_i x_i g(\left\|\frac{x_i - y}{h}\right\|^2)}{\sum_i g(\left\|\frac{x_i - y}{h}\right\|^2)}$$

Goal: find density maximum

$$\nabla P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \nabla K(\mathbf{x} - \mathbf{x}_i)$$

Give up estimating the PDF !
Estimate **ONLY** the gradient

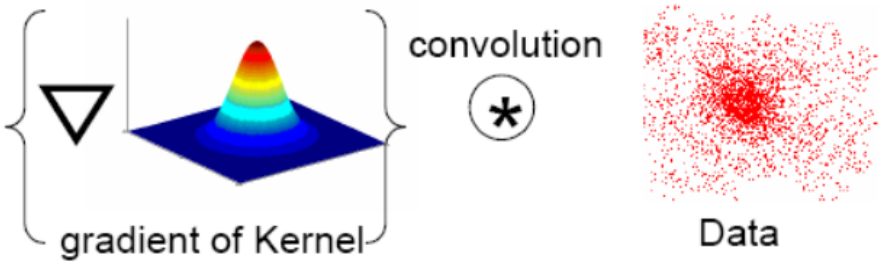
Using the Kernel form:

$$K(\mathbf{x} - \mathbf{x}_i) = ck\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right)$$

Size of window

We get :

$$\nabla P(\mathbf{x}) = \frac{c}{n}\sum_{i=1}^{n} \nabla k_i = \frac{c}{n}\left[\sum_{i=1}^{n} g_i\right] \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g_i}{\sum_{i=1}^{n} g_i} - \mathbf{x}\right]$$

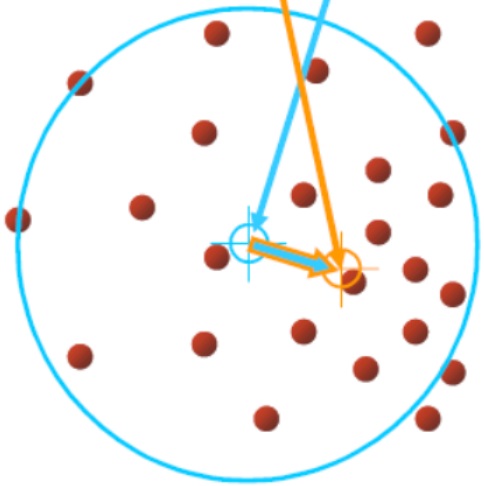


Gradient of superposition of kernels, centered at each data point is equivalent to convolving the data points with gradient of the kernel

Computing The Mean Shift

$$\nabla P(\mathbf{x}) = \frac{c}{n}\sum_{i=1}^{n}\nabla k_i = \frac{c}{n}\left[\sum_{i=1}^{n} g_i\right]\left[\frac{\sum_{i=1}^{n}\mathbf{x}_i g_i}{\sum_{i=1}^{n} g_i} - \mathbf{x}\right]$$

Yet another Kernel density estimation !



Simple Mean Shift procedure:

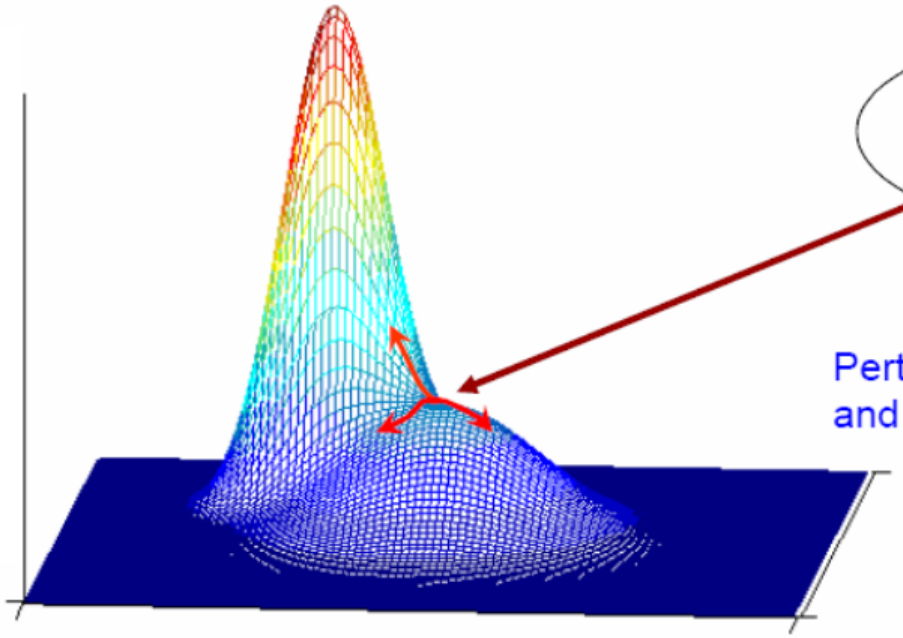
- Compute mean shift vector

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

- Translate the Kernel window by **m(x)**

g(x)

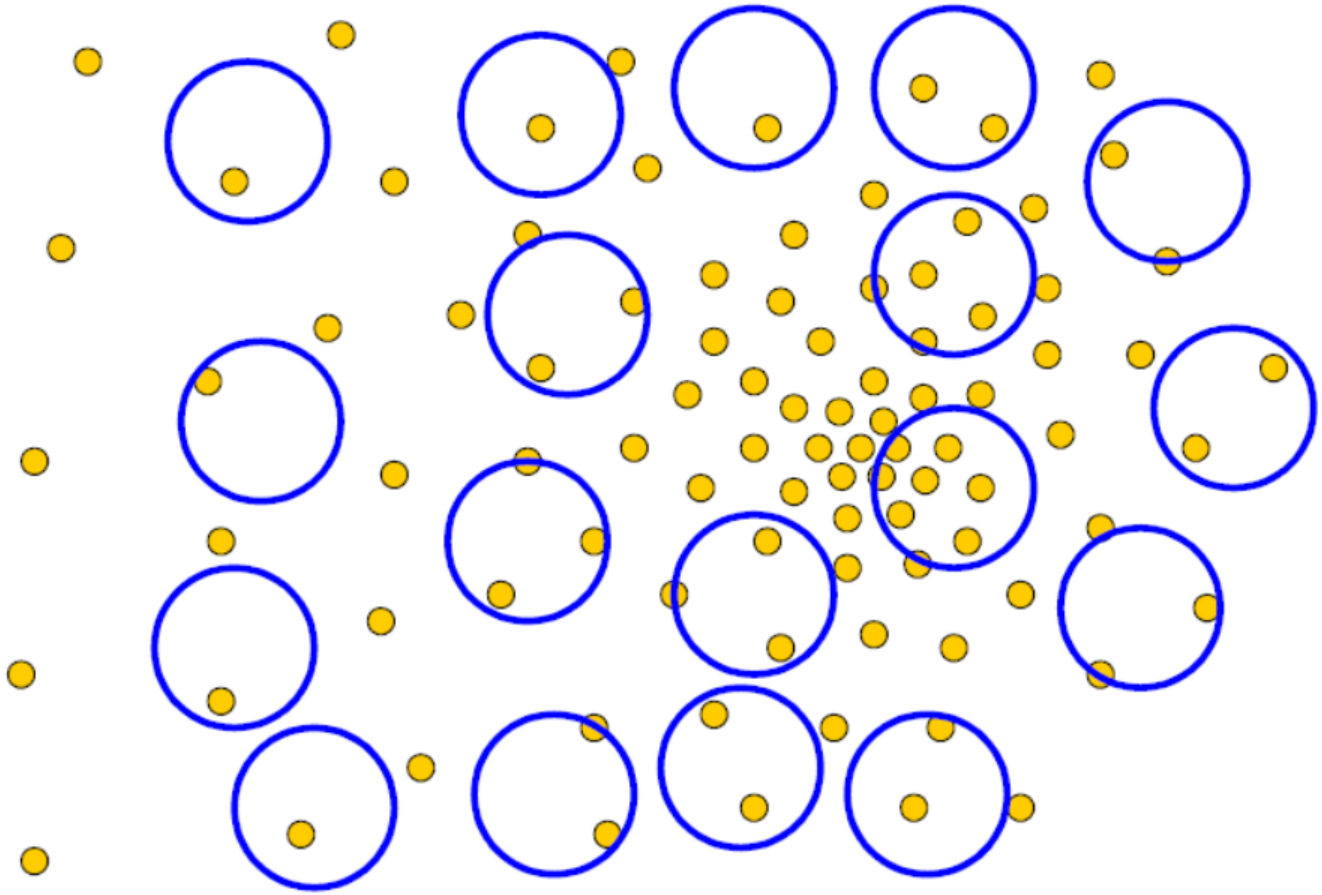
Local maxima



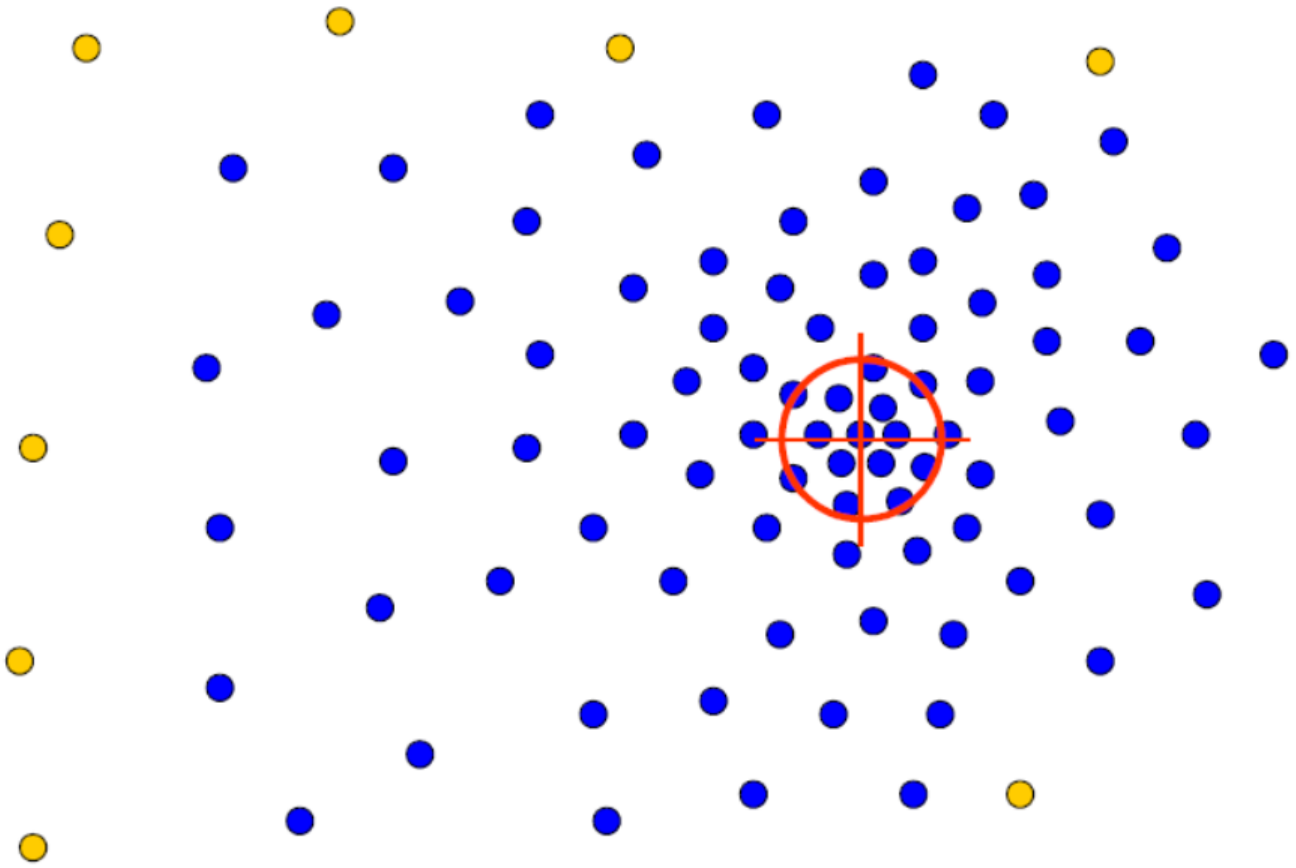
Updated Mean Shift Procedure:

- Find all modes using the Simple Mean Shift Procedure
- Prune modes by perturbing them (find saddle points and plateaus)
- Prune nearby – take highest mode in the window

Random initialisations

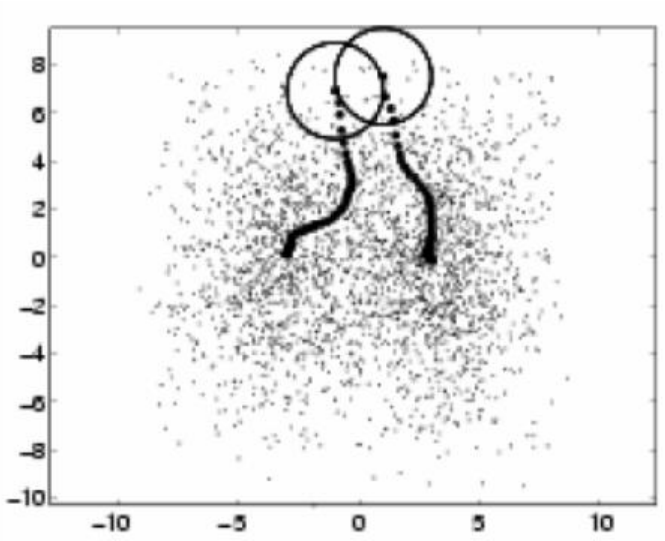


Convergence



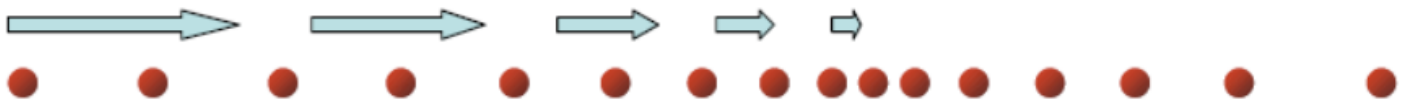
The blue data points were traversed by the windows towards the mode

Gradient ascent to local density max



Window tracks signify the steepest ascent directions

Mean Shift Properties

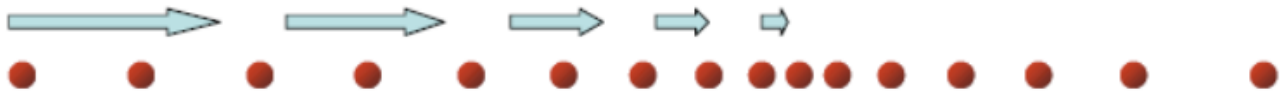


- Automatic convergence speed – the mean shift vector size depends on the gradient itself.
- Near maxima, the steps are small and refined

Adaptive Gradient Ascent

- Convergence is guaranteed for infinitesimal steps only ➔ infinitely convergent, (therefore set a lower bound)
- For Uniform Kernel ( ), convergence is achieved in a finite number of steps
- Normal Kernel ( ) exhibits a smooth trajectory, but is slower than Uniform Kernel ( ).

Mean Shift Strengths & Weaknesses



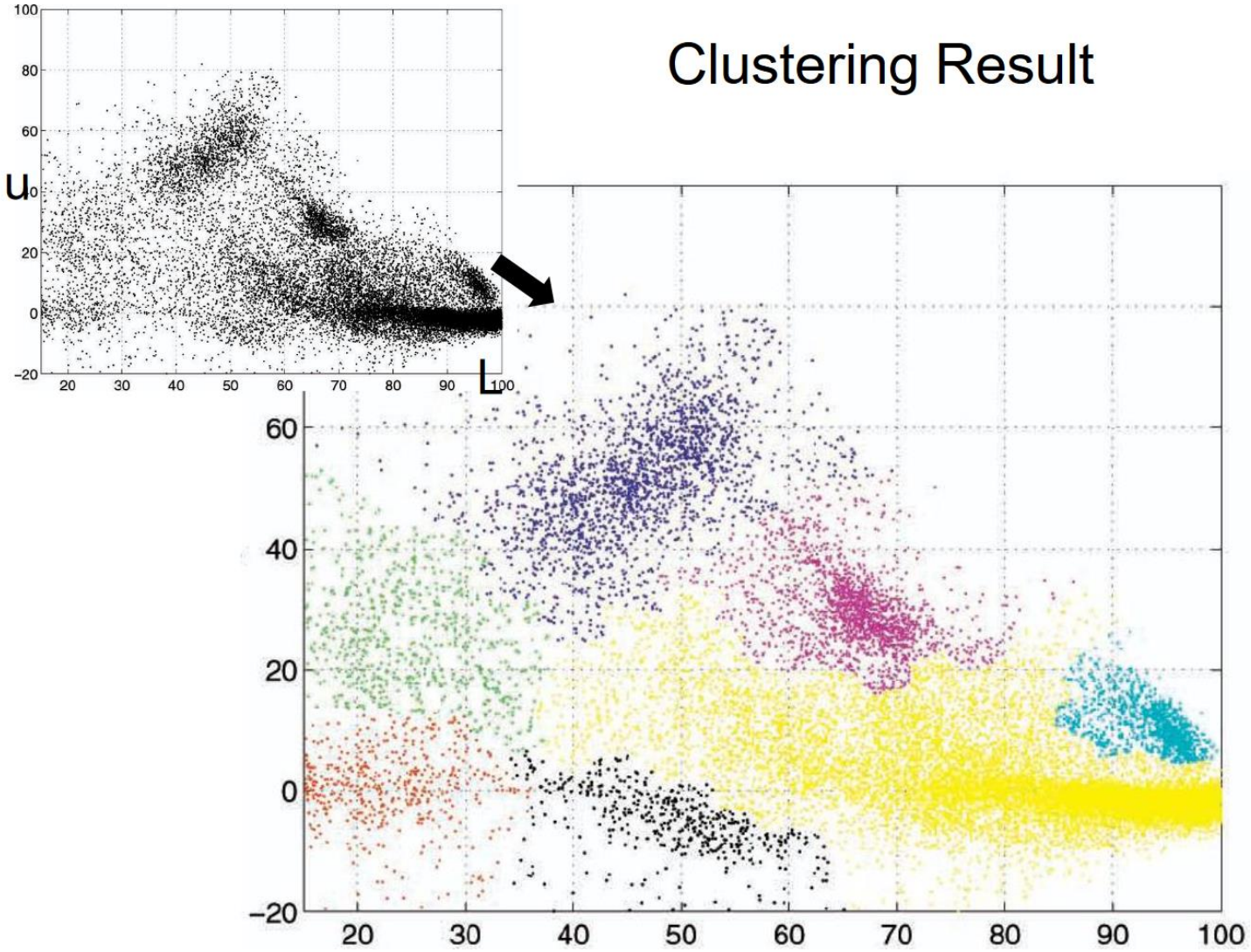
Strengths :

- Application independent tool
- Suitable for real data analysis
- Does not assume any prior shape (e.g. elliptical) on data clusters
- Can handle arbitrary feature spaces
- Only ONE parameter to choose
- *h* (window size) has a physical meaning, unlike K-Means

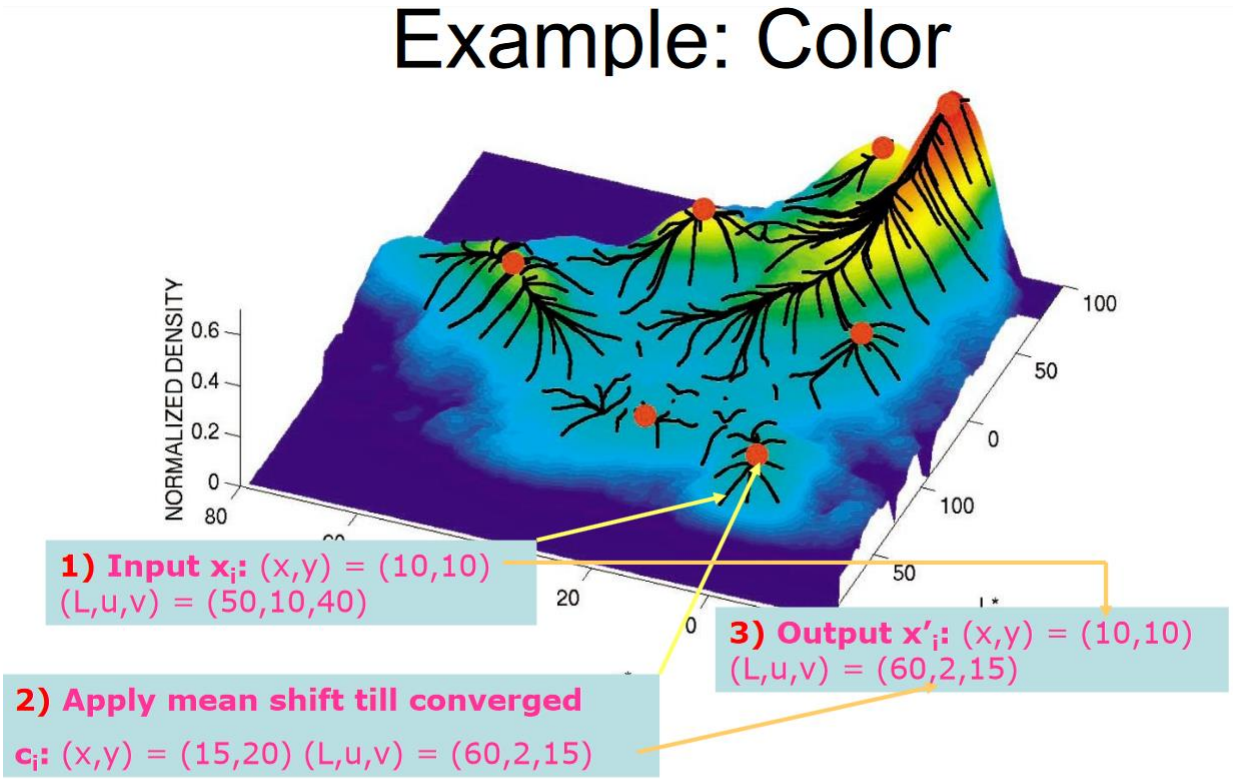
Weaknesses :

- The window size (bandwidth selection) is not trivial
- Inappropriate window size can cause modes to be merged, or generate additional "shallow" modes → Use adaptive window size

Colour based Mean Shift



Colour based Mean Shift



Note: In practice, all points may not converge to the same mode -→ Need an additional (easy) clustering step to group the converged locations

Colour based image segmentation

- 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
  - Each pixel belongs to a segment whose number is indicated by its nearest cluster center
    - Replacing each color by the closest mode
  - Typically too many segments but tend to be much better clustered than pixel representations

Image segmentation

- The "colour" part of the feature can be replaced by other measurements
  - Texture (bank of filter outputs) or other values (multispectral).
  - The dimension p of the feature space increases

- The fundamental operation to compute the kernels is to find the neighbours within some radius (defined by h).
  - Can be very expensive in high dimensional space with lots of points
  - Need efficient data structures for nearest-neighbor search.

Examples

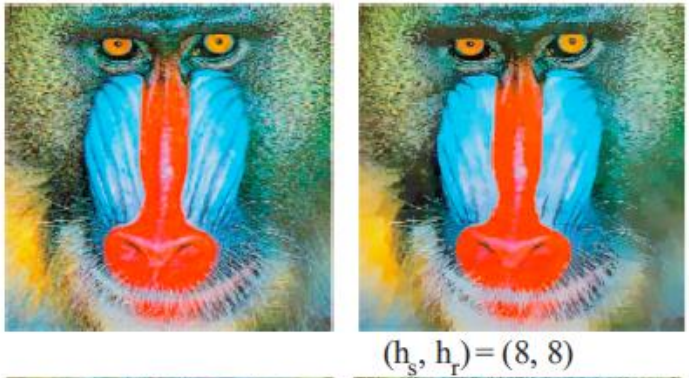


FIGURE 9.20: An image (**top left**) and mean shift modes obtained with different clustering scales for space h_s and appearance h_r. If h_s is small, the method must produce clusters that are relatively small and compact spatially because the kernel function smoothes over a relatively small radius and 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.

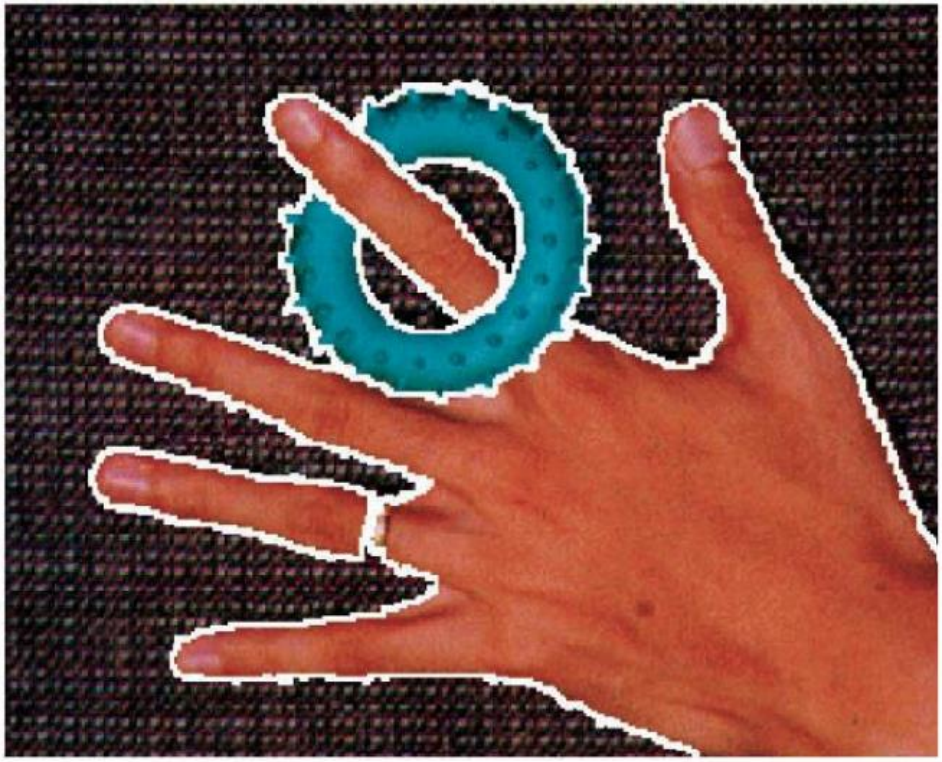
Examples






Questions: goo.gl/K61te5

Examples



Questions: goo.gl/K61te5

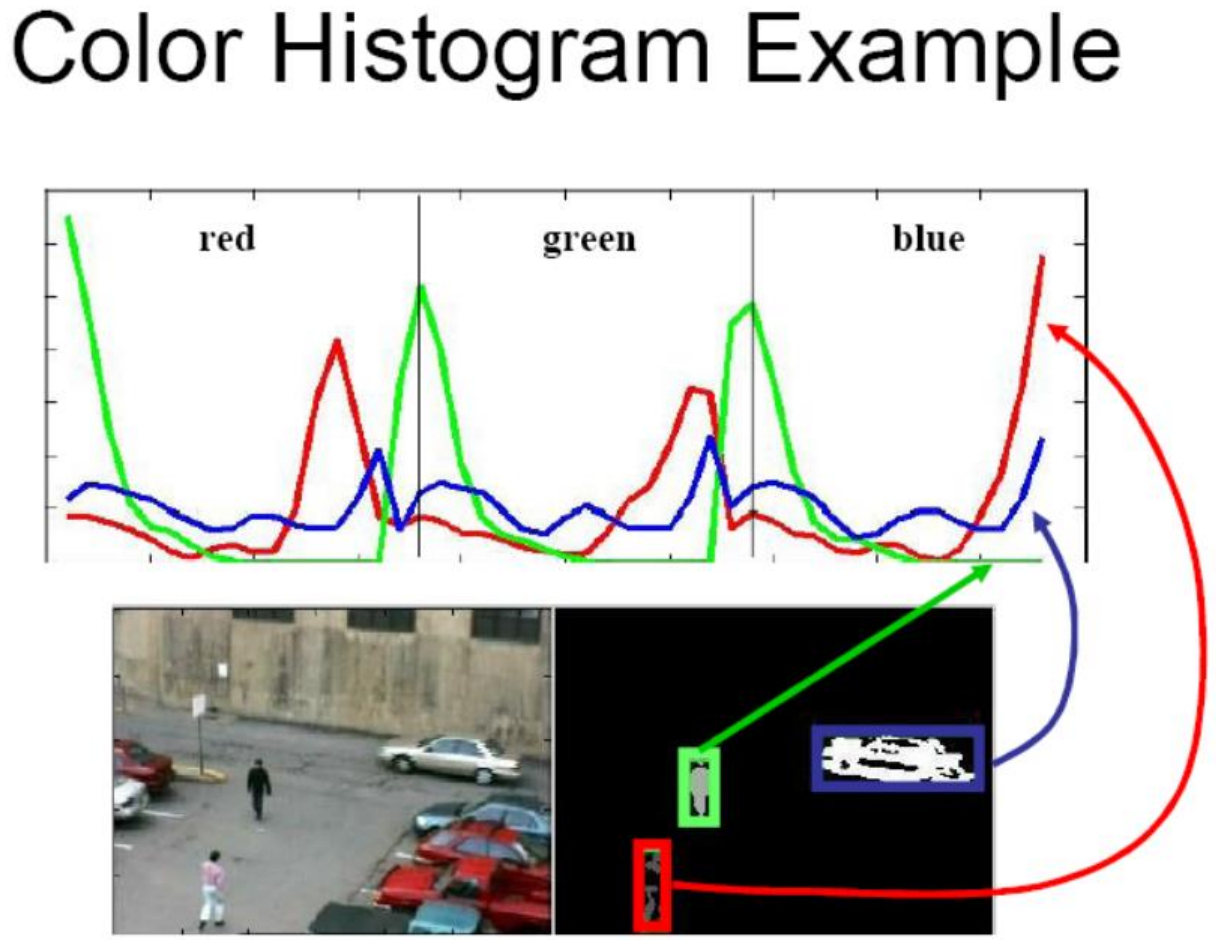
Examples





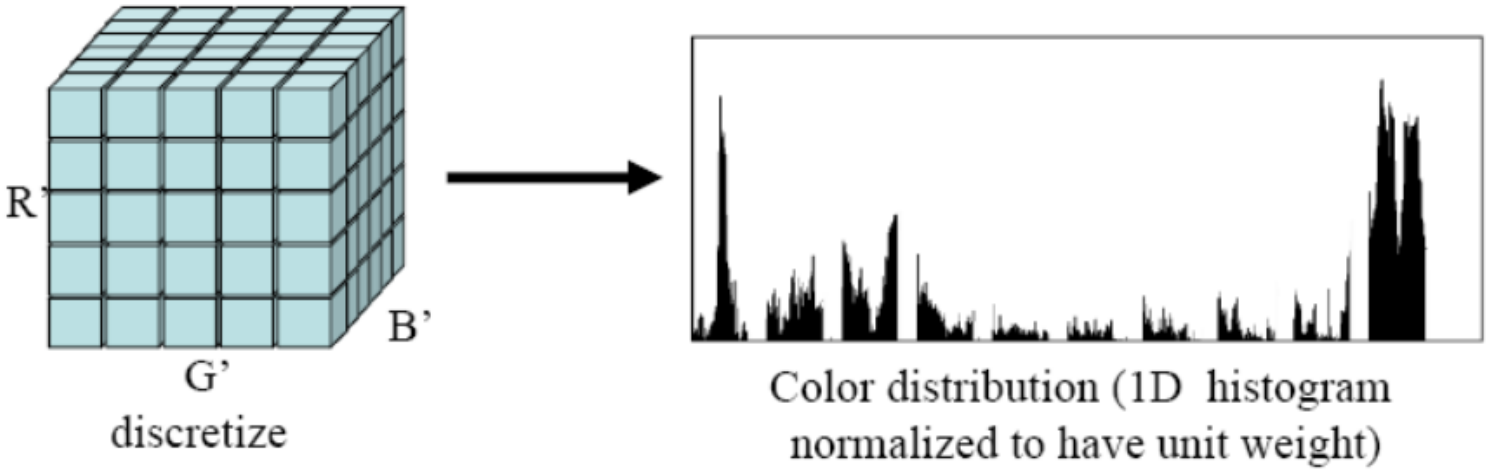

Questions: goo.gl/K61te5

Tracking with Mean Shift



Tracking with Mean Shift

Appearance via Color Histograms



R' = R << (8 - nbits)
G' = G << (8 - nbits)
B' = B << (8-nbits)

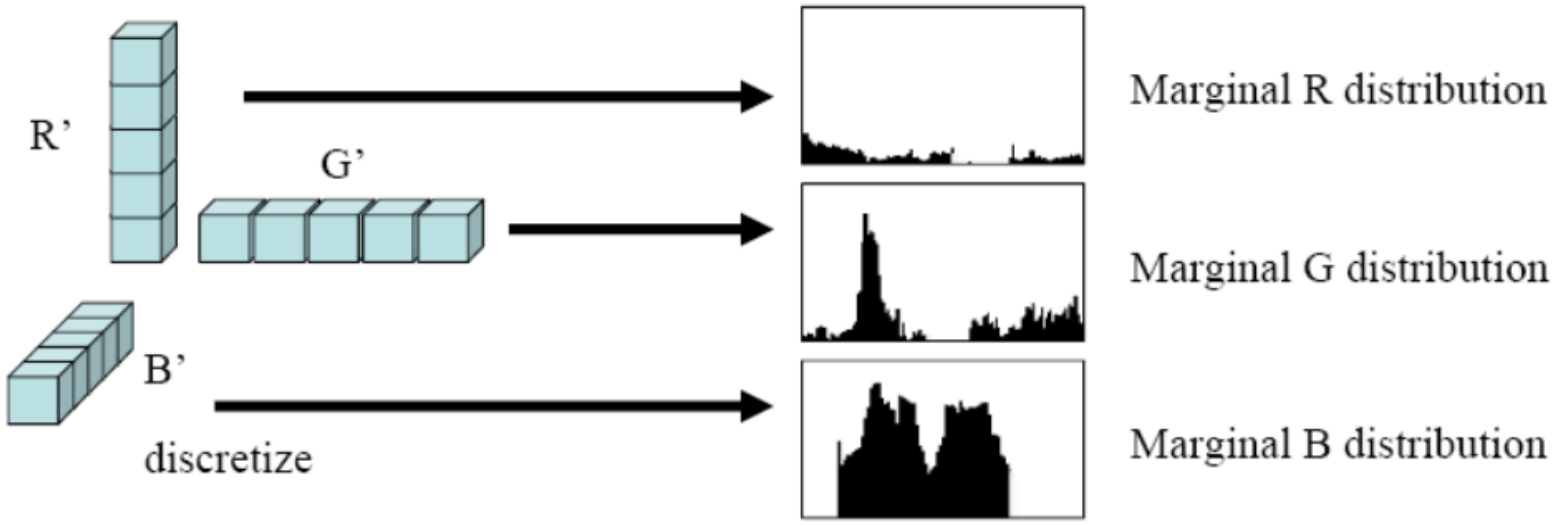
Total histogram size is (2^(8-nbits))^3

example, 4-bit encoding of R,G and B channels yields a histogram of size 16*16*16 = 4096.

Tracking with Mean Shift

Smaller Color Histograms

Histogram information can be much much smaller if we are willing to accept a loss in color resolvability.

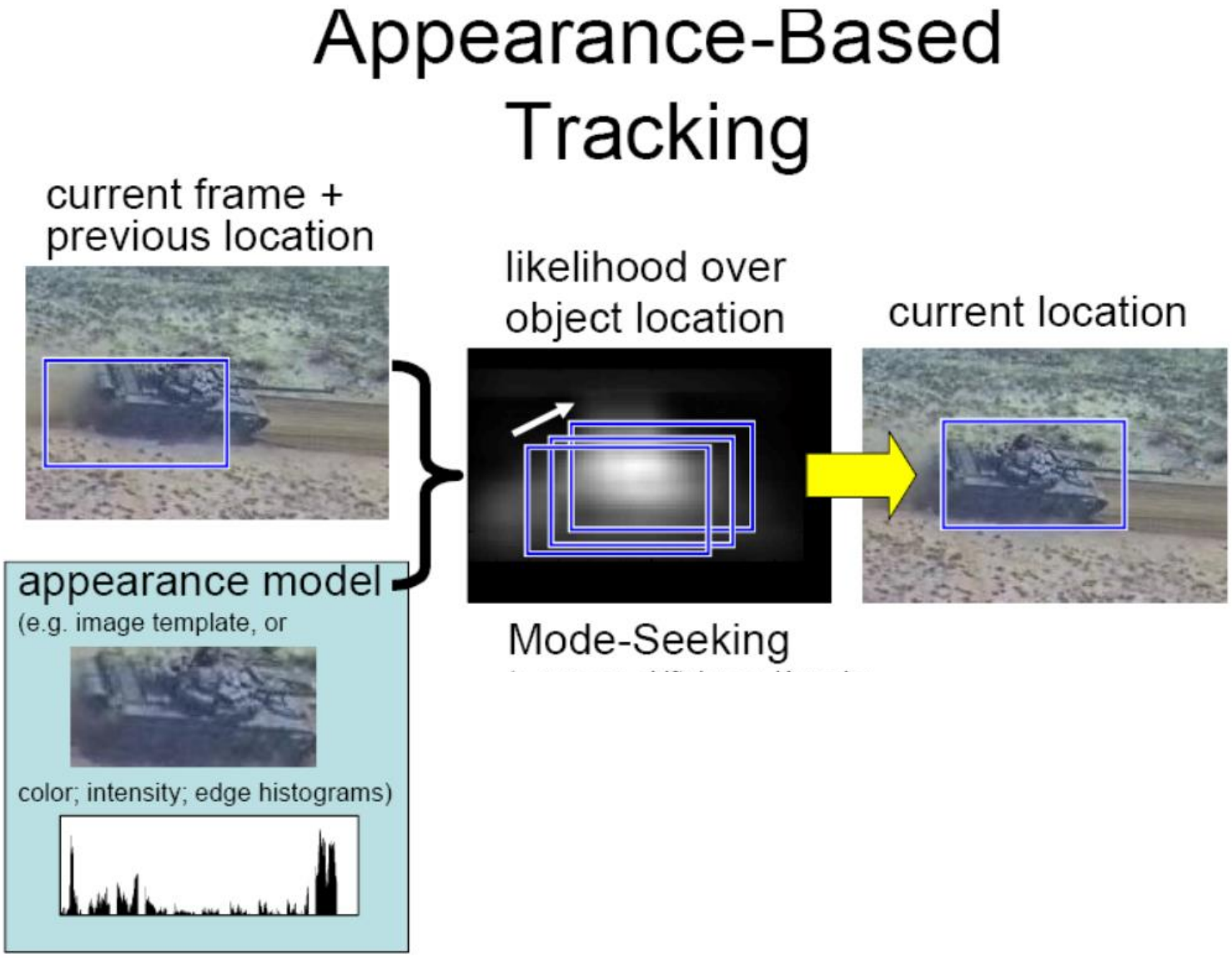


R' = R << (8 - nbits)
G' = G << (8 - nbits)
B' = B << (8-nbits)

Total histogram size is 3*(2^(8-nbits))

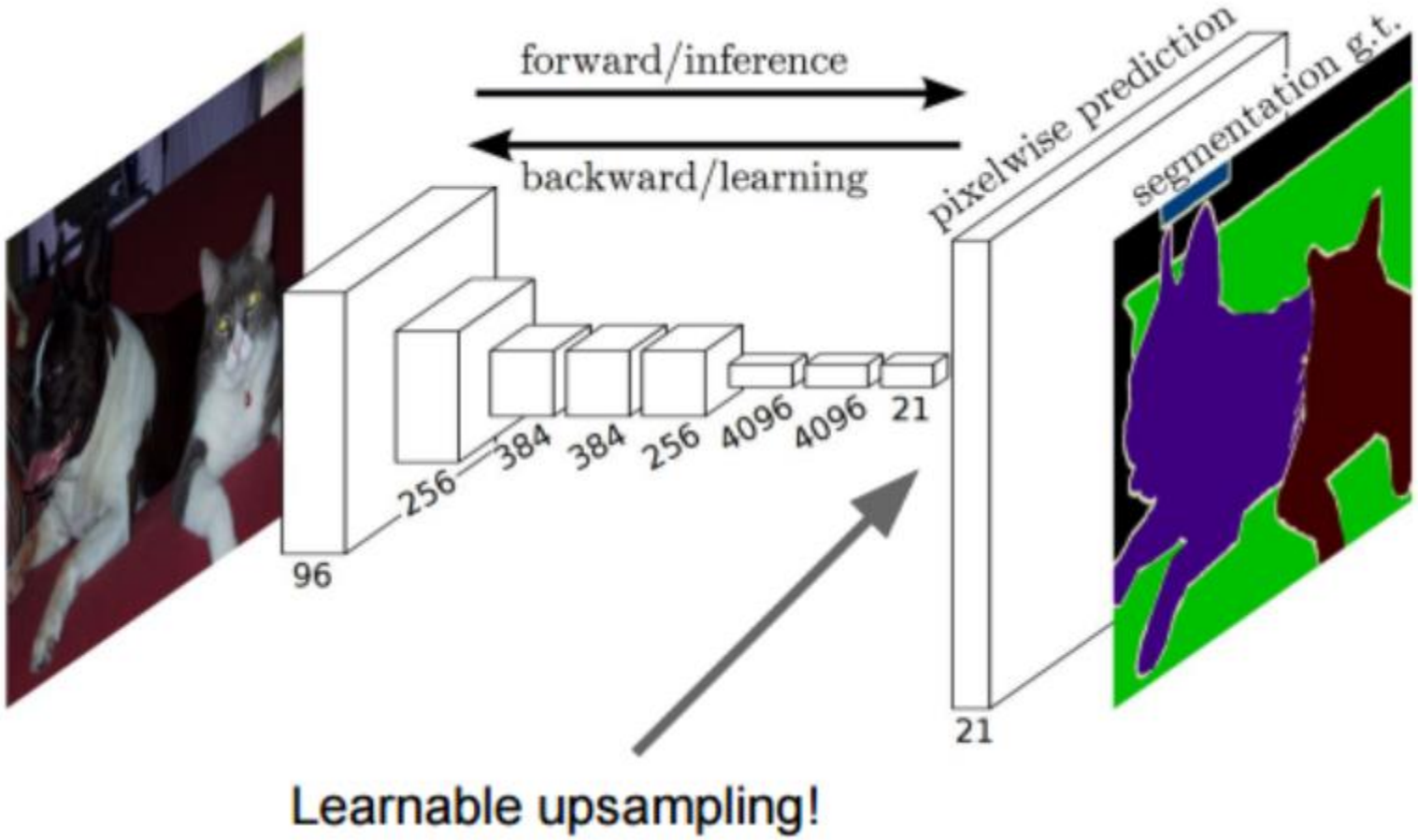
example, 4-bit encoding of R,G and B channels yields a histogram of size 3*16 = 48.

Tracking with Mean Shift

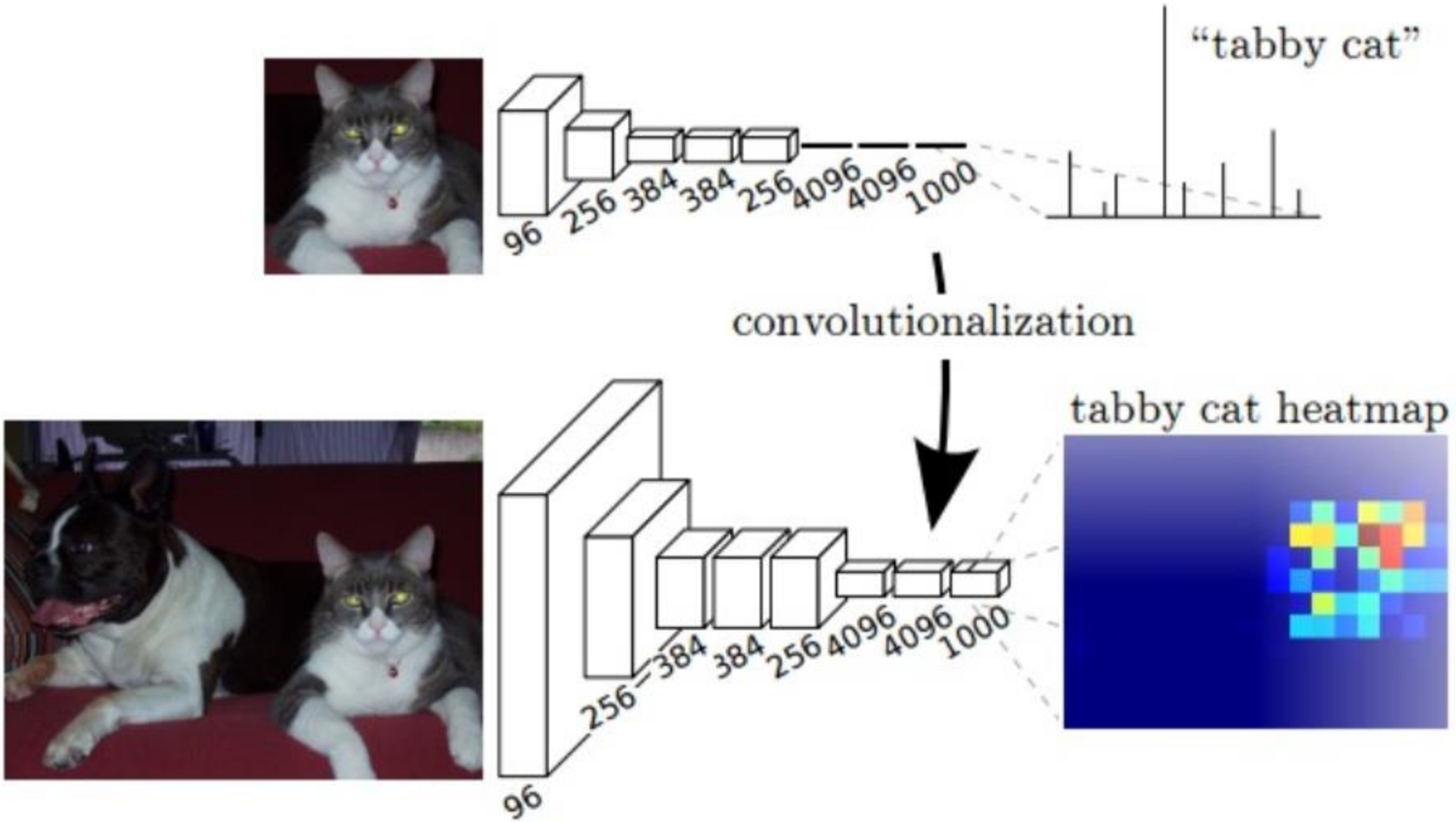


CNN based segmentation

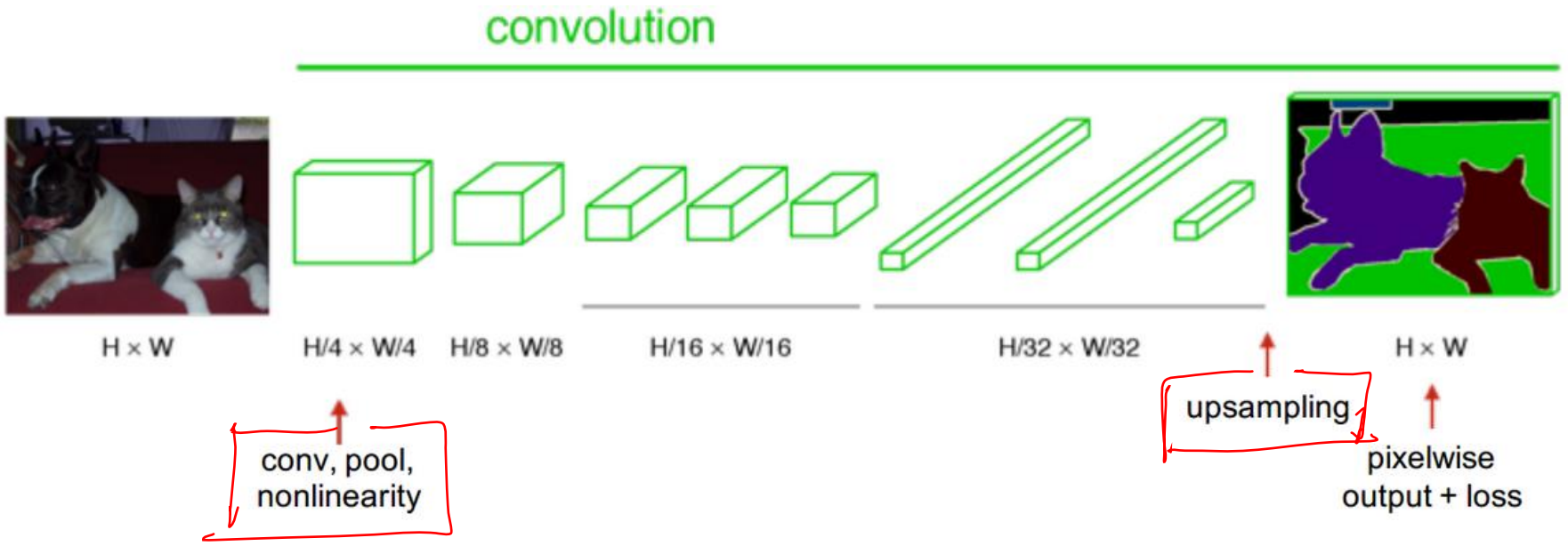
CNN segmentation



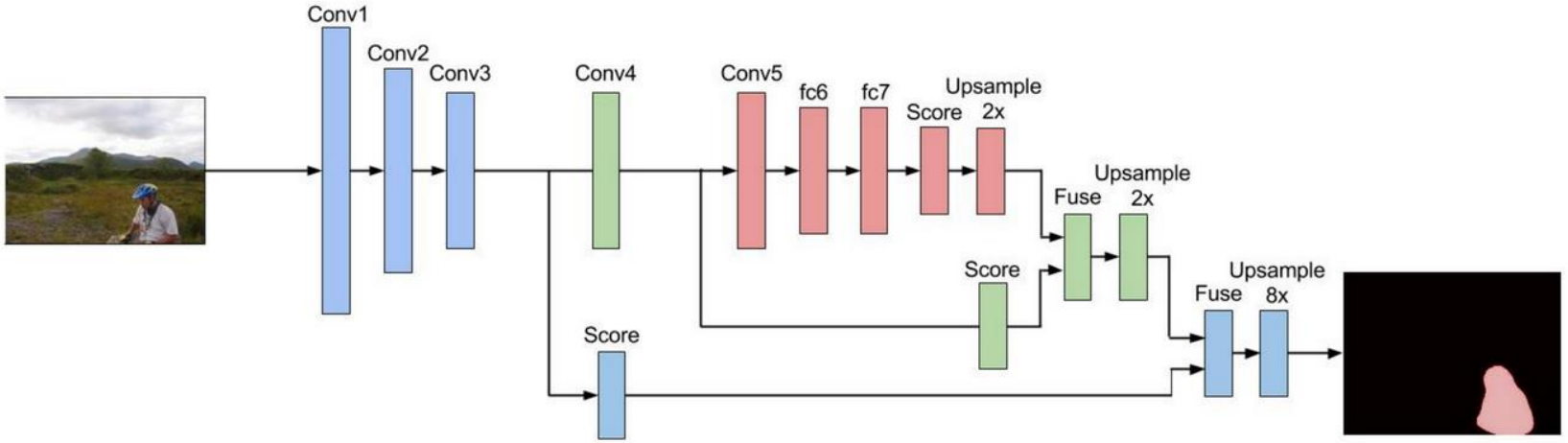
CNN segmentation

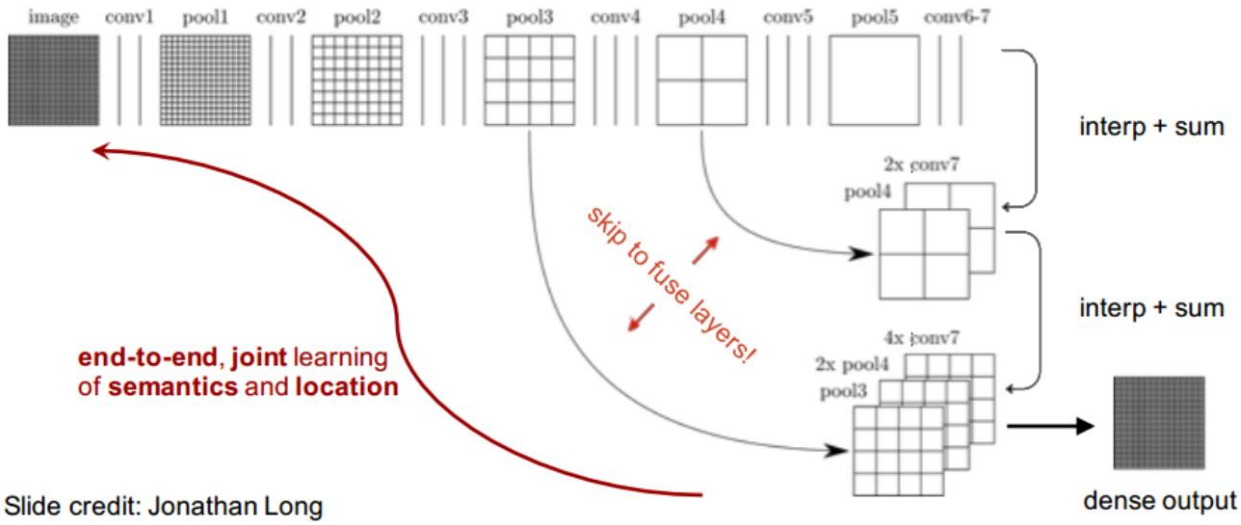


CNN segmentation



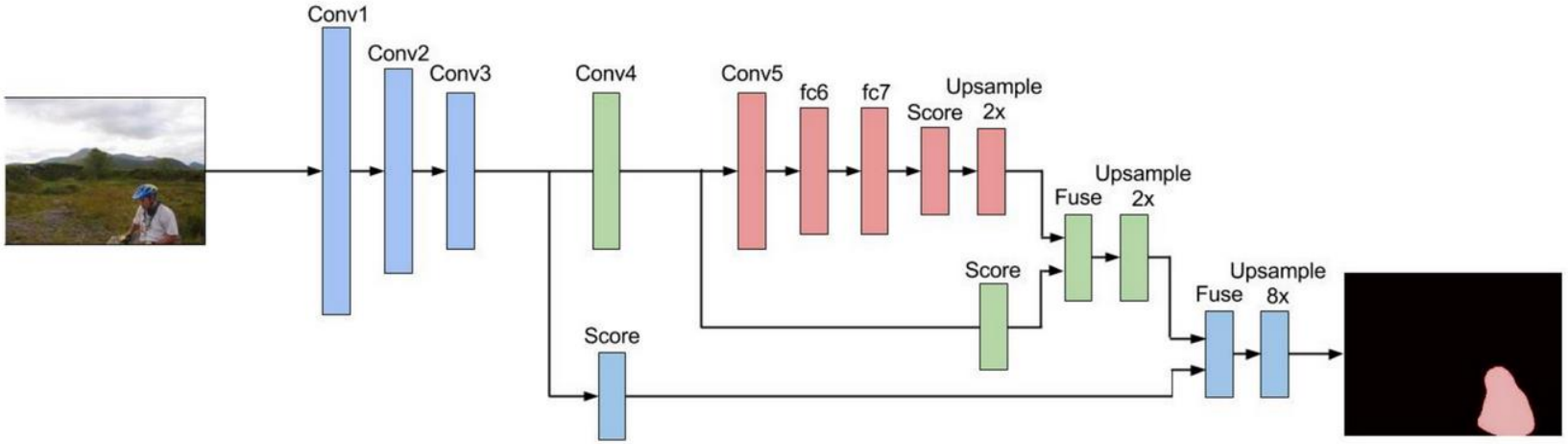
CNN skip connections

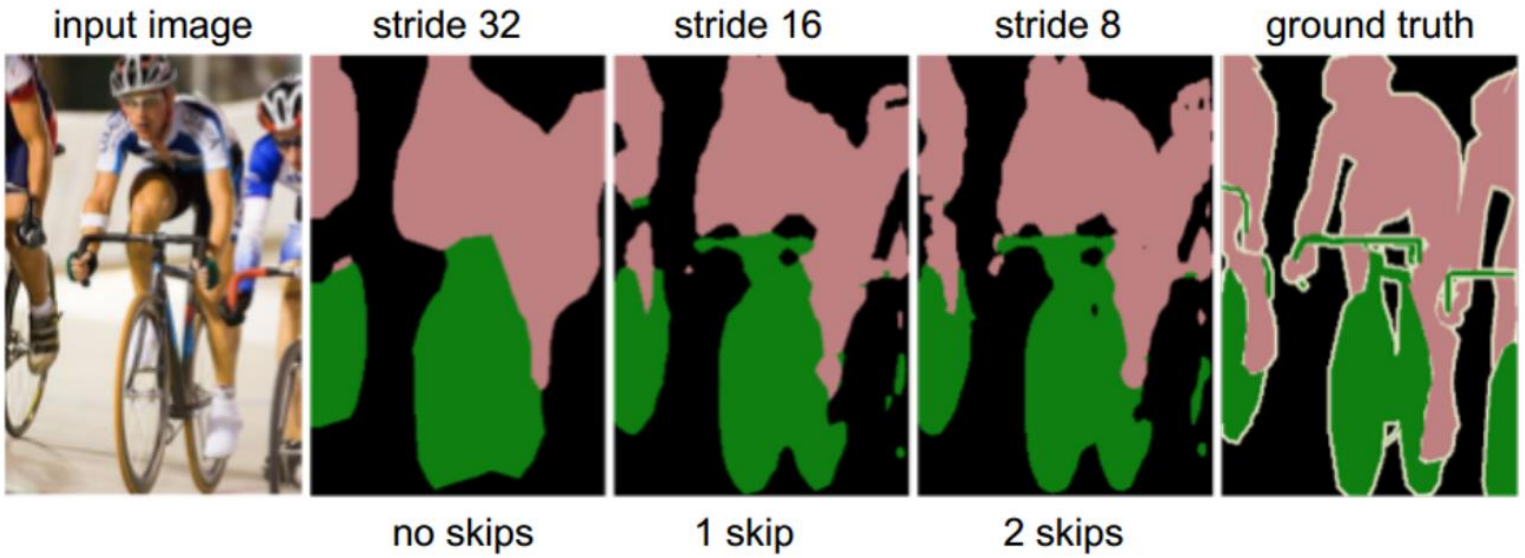




Slide credit: Jonathan Long

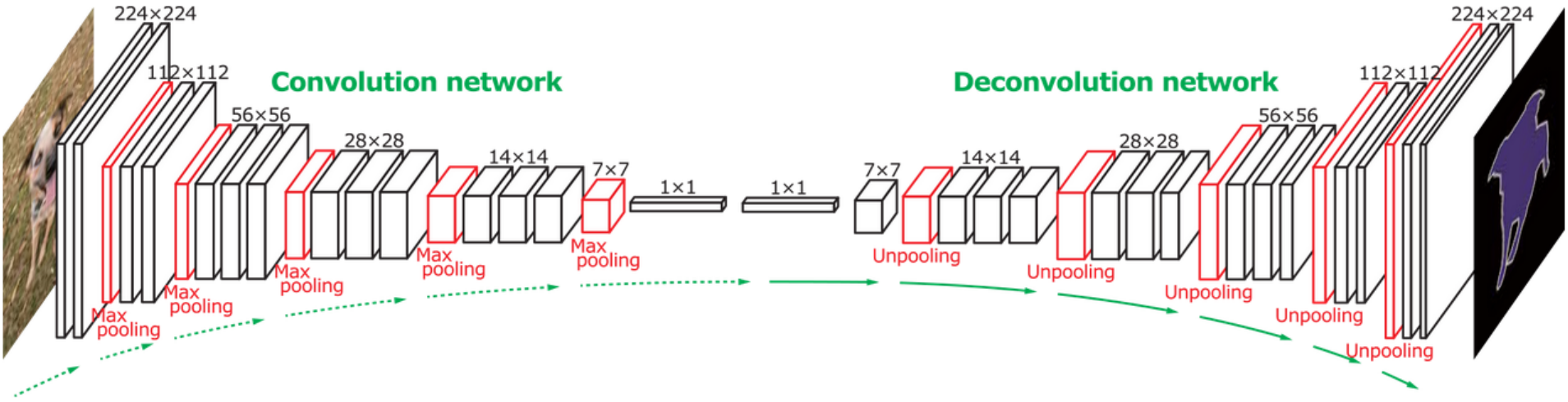
CNN skip connections

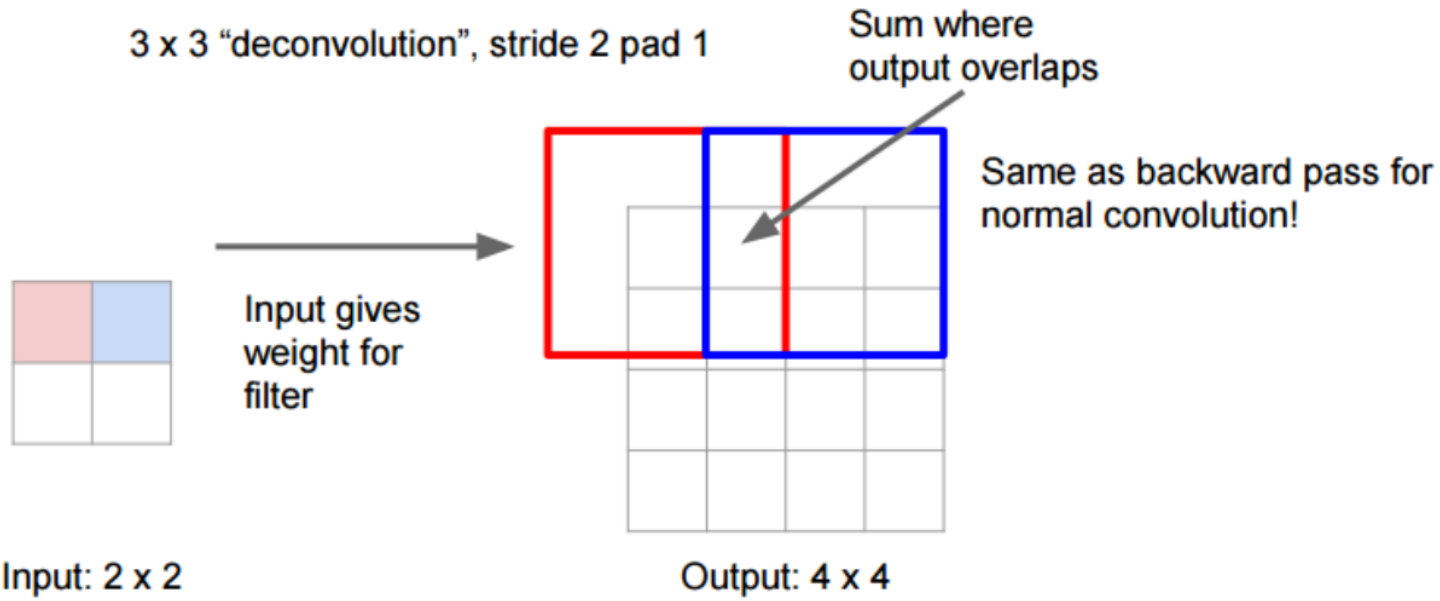




DeconvNet

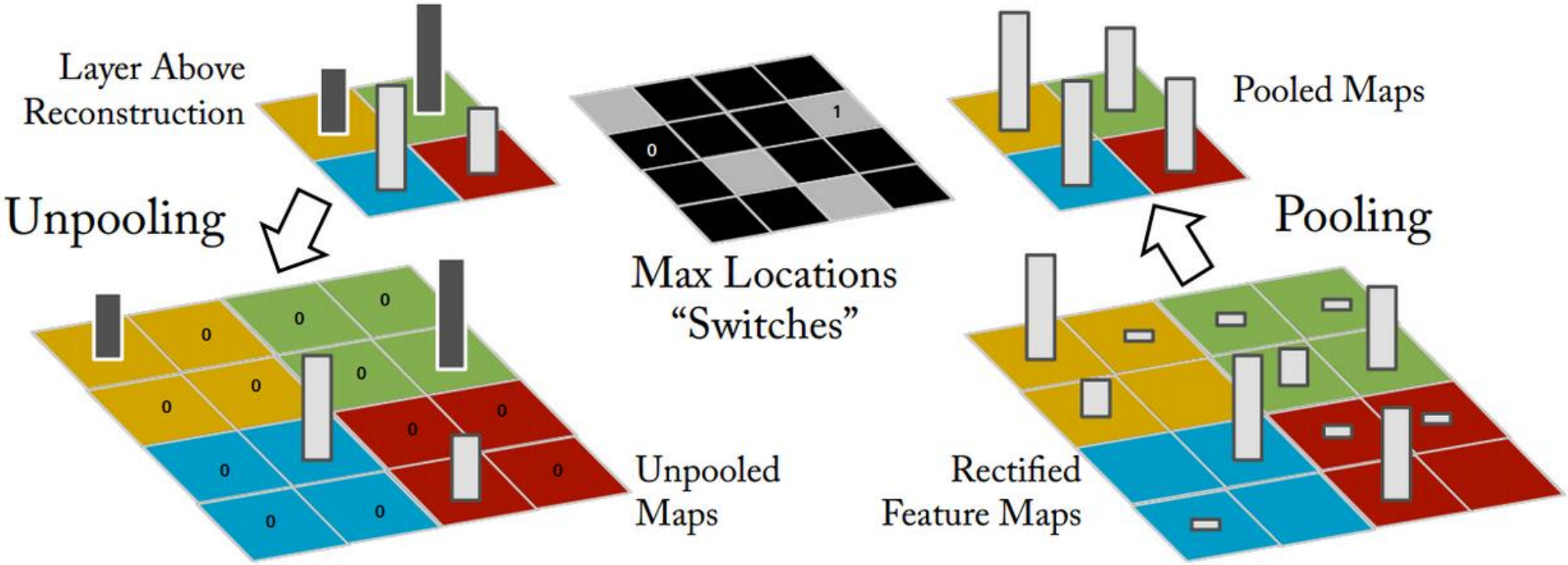
Autoencoders.

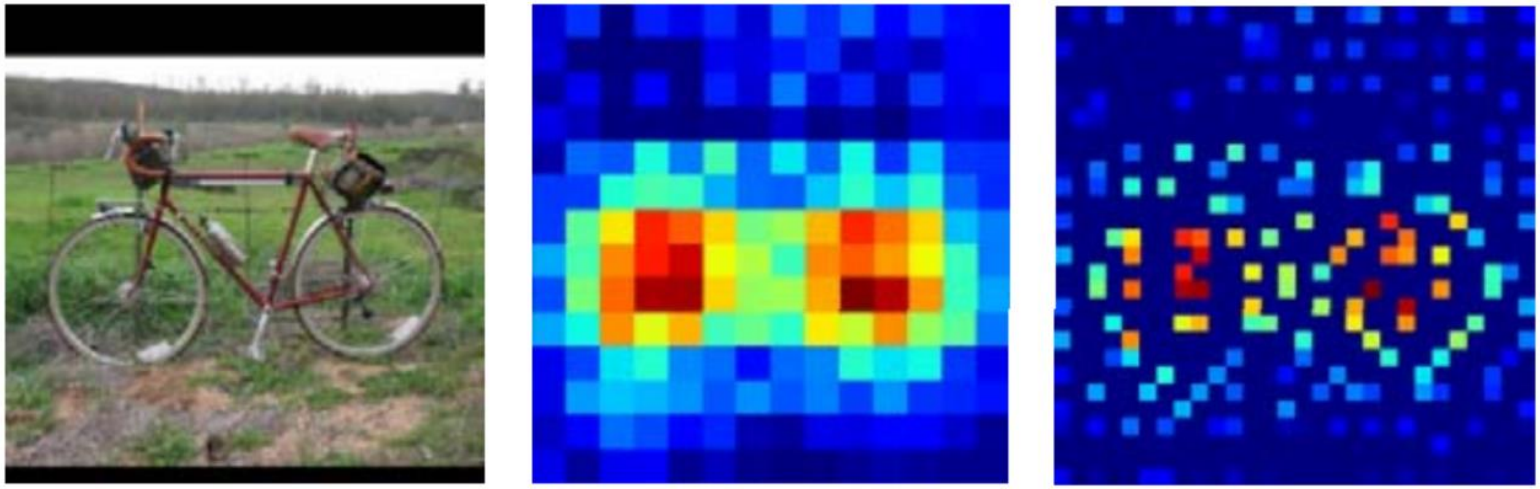




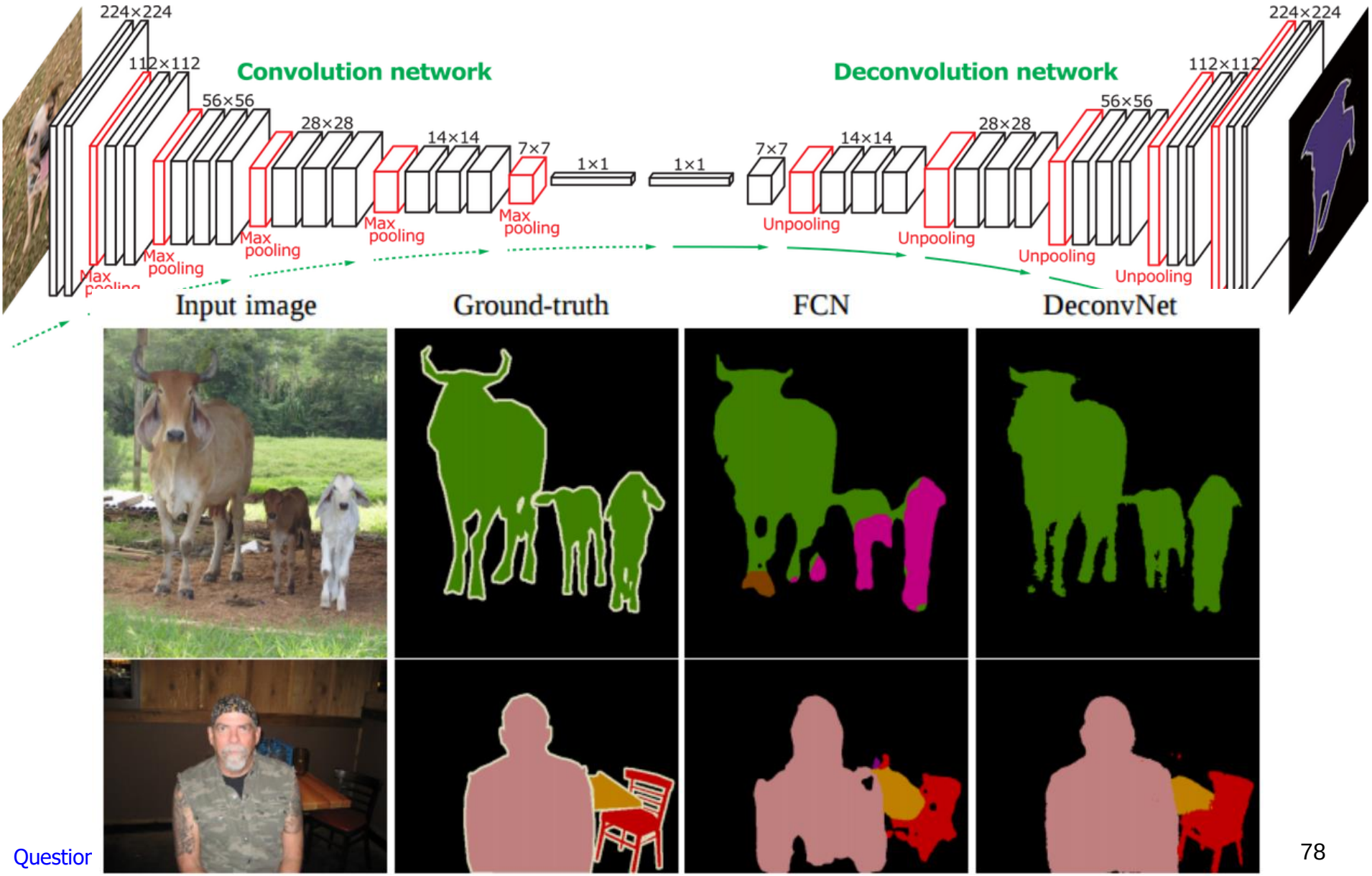
Questions: goo.gl/K61te5

Unpooling


Layer Above Reconstruction
Unpooling
Unpooled Maps
Max Locations "Switches"
Pooled Maps
Pooling
Rectified Feature Maps


Deconv: 14x14
Unpool: 28x28

DeconvNet


224×224
112×112
56×56
28×28
14×14
7×7
1×1
1×1
7×7
14×14
28×28
56×56
112×112
224×224
Convolution network
Deconvolution network
Max pooling
Max pooling
Max pooling
Max pooling
Unpooling
Unpooling
Unpooling
Unpooling
Input image
Ground-truth
FCN
DeconvNet