# *Introduction to Vision and Robotics:*

# *Computer Vision*

## Image segmentation

Vittorio Ferrari

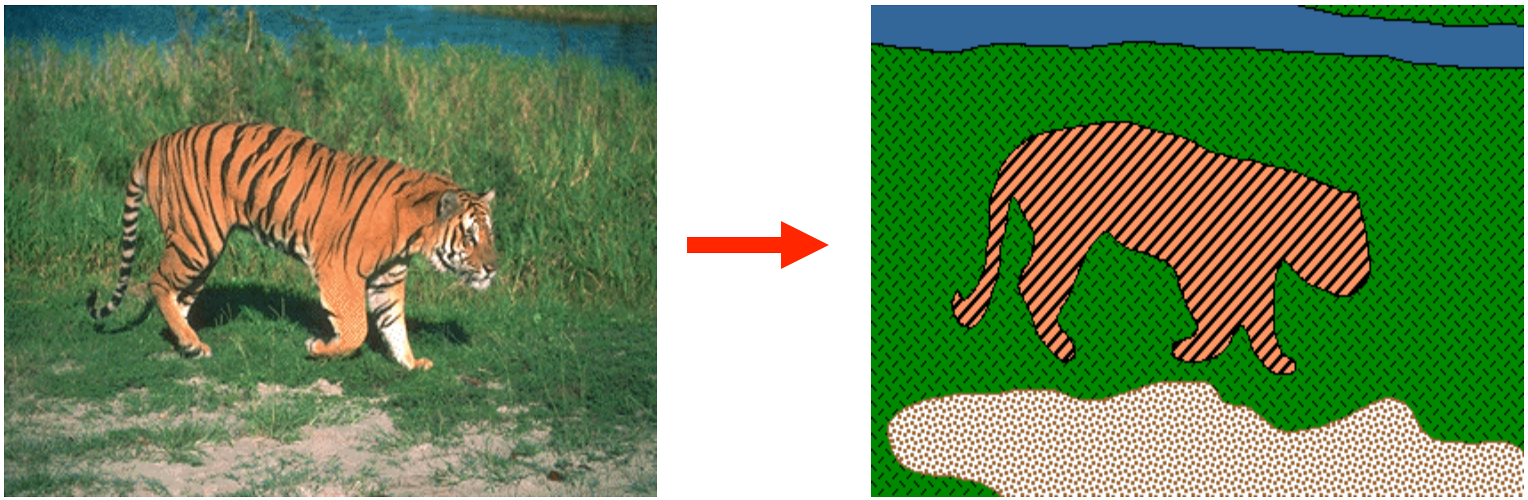
Many slides in this lecture are due to other authors; they are credited on the bottom right

# Topics of This Lecture

- Problem definition and goals
- Greylevel segmentation by thresholding
- Background removal
- Canny edge detection
- Segmentation into multiple regions with mean-shift

# Image Segmentation

- **Goal: identify groups of pixels that go together**

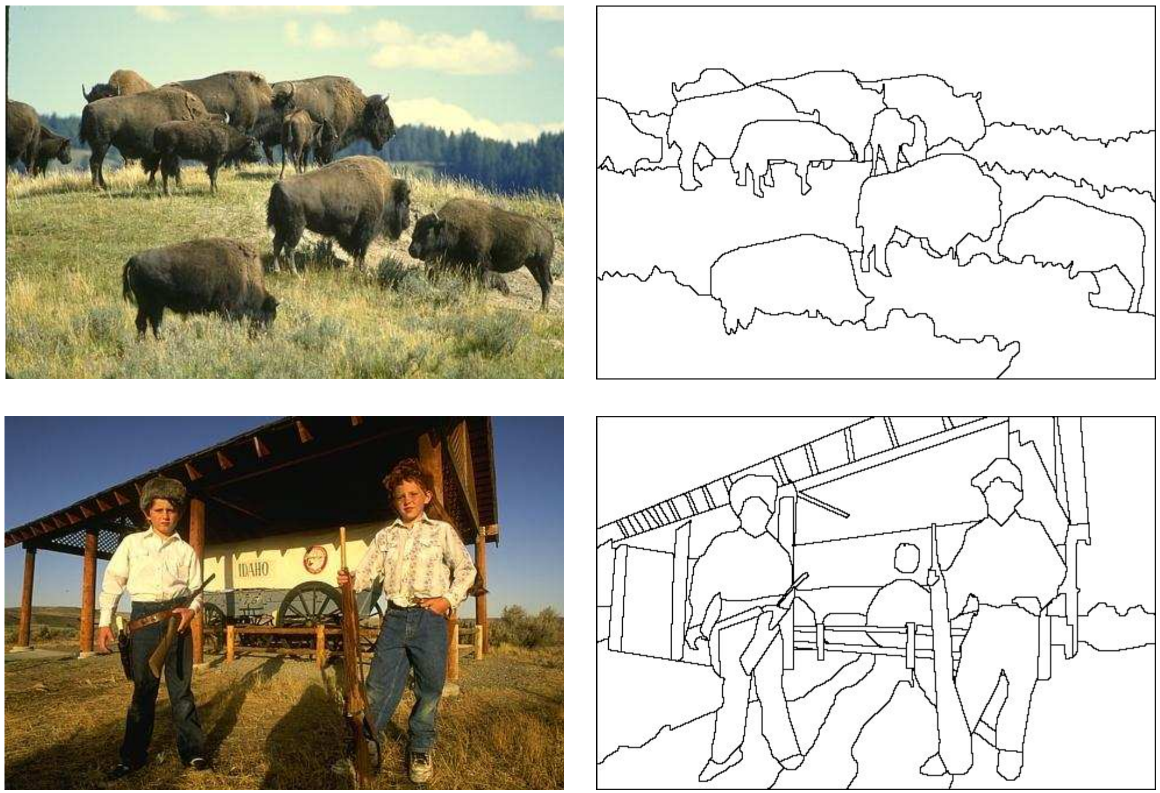


Slide credit: Steve Seitz, Kristen Grauman

# The Goals of Segmentation

- Separate image into objects

**Image** | **Human segmentation**



Slide credit: Svetlana Lazebnik

# Topics of This Lecture

- Problem definition and goals
- Greylevel segmentation by thresholding
- Background removal
- Canny edge detection
- Segmentation into multiple regions with mean-shift

# Isolating flat parts

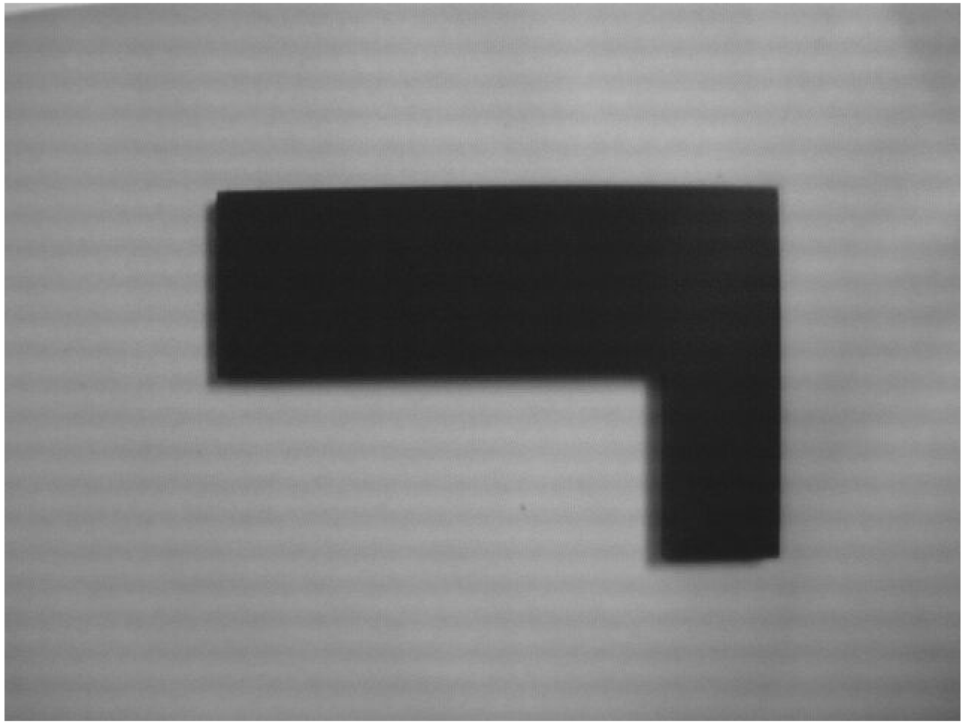
Isolate parts, then characterise later

Assume

- Dark part
- Light background
- Reasonably uniform illumination $->$ distinguishable parts

Slide credit: Bob Fisher

# Given this image, how might we label pixels as object and background?



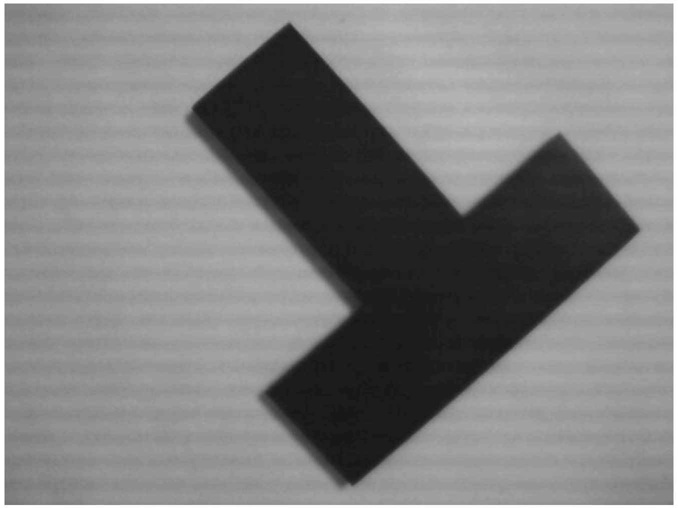
Slide credit: Bob Fisher

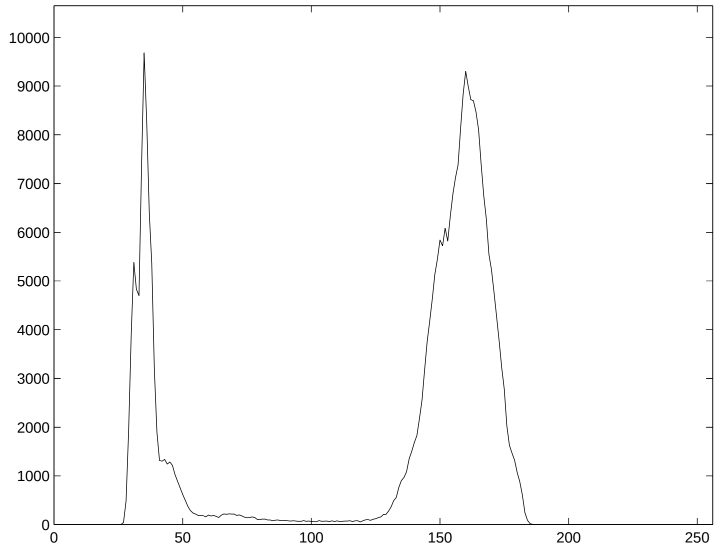
# Thresholding Introduction

Key technique: thresholding

Assume pixel values are separable

Part and typical distribution






Spread: not quite uniform illumination + part color variations + sensor noise

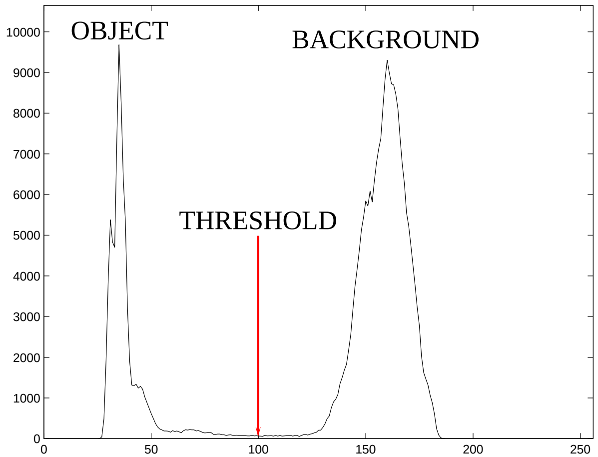
Slide credit: Bob Fisher

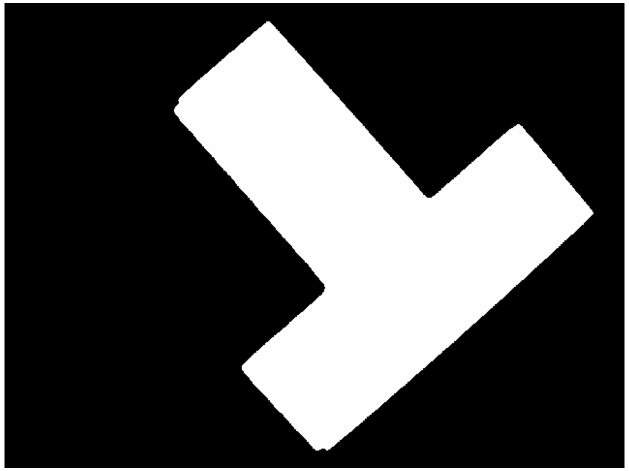
# Thresholding

Thresholding: central technique

```
for row = 1 : height
 for col = 1 : width
  if value(row,col) < ThreshHigh % inside high bnd
  % & value(row,col) > ThreshLow % optional low bnd
      output(row,col) = 1;
  else
      output(row,col) = 0;
  end
```

Slide credit: Bob Fisher

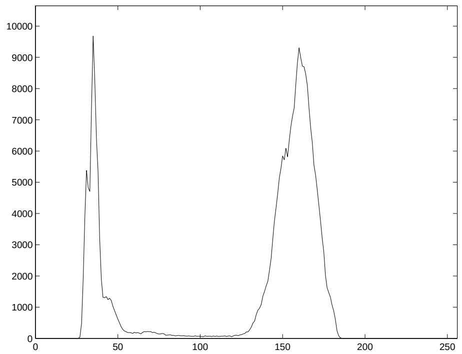


Slide credit: Bob Fisher

# Threshold Selection

Exploit bimodal distribution




But:

- Distributions broad and some overlap $->$ misclassified pixels
- Shadows dark so might be classified with object
- Distribution has more than 2 peaks

So: smooth histogram to improve shape for selection

Slide credit: Bob Fisher

# Convolution

General purpose image (and signal) processing function

Computed by a weighted sum of image data and a fixed mask

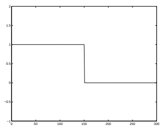
Linear operator: conv(a*B,C) = a*conv(B,C)

Used in different processes: noise removal, smoothing, feature detection, differentiation, ...

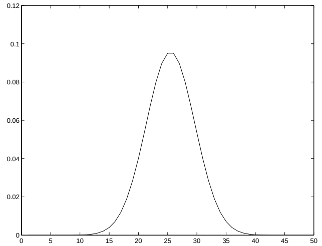
Slide credit: Bob Fisher

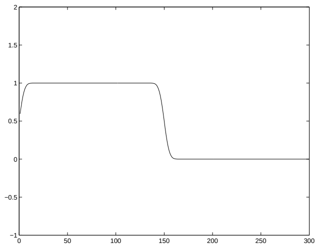
# Convolution in 1D

$$Output(x) = \sum_{i=-N}^{N} weight(i) * input(x - i)$$

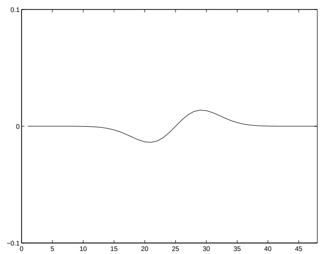
Input: 

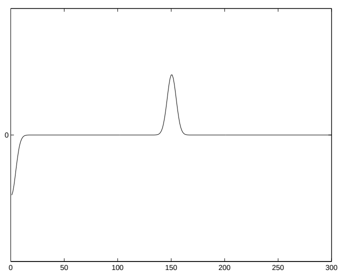
Gaussian Mask and Output:





Derivative of Gaussian Mask and Output:





Slide credit: Bob Fisher

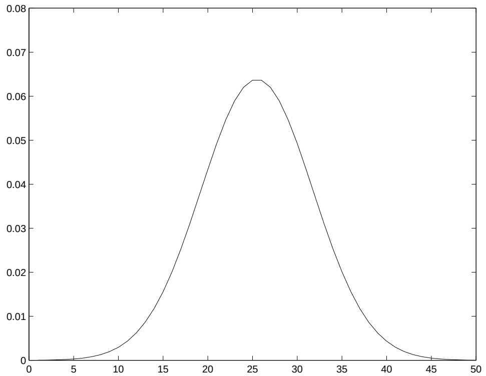
# Histogram Smoothing for threshold selection

Histogram Smoothing (in `findthresh.m`)

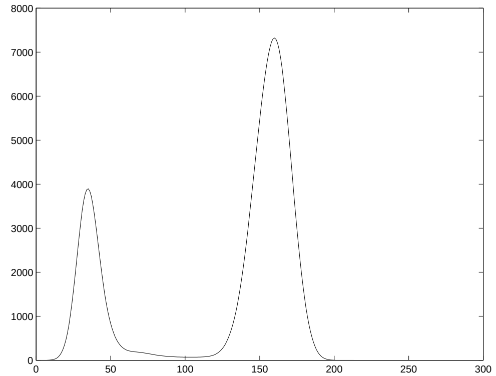
Convolve with a Gaussian smoothing window

```
filterlen = 50;                         % filter length
thefilter = gausswin(filterlen,sizeparam); % size=4
thefilter = thefilter/sum(thefilter); % unit norm
tmp2=conv(thefilter,thehist); % makes longer output
% select corresponding portion
offset = floor((filterlen+1)/2);
tmp1=tmp2(offset:len+offset-1);
```

Slide credit: Bob Fisher




FILTER SHAPE



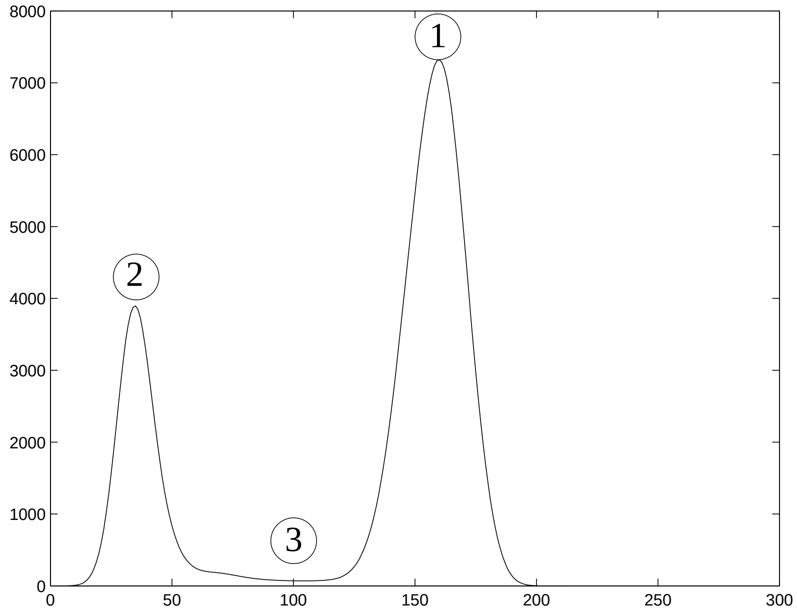

SMOOTHED HISTOGRAM

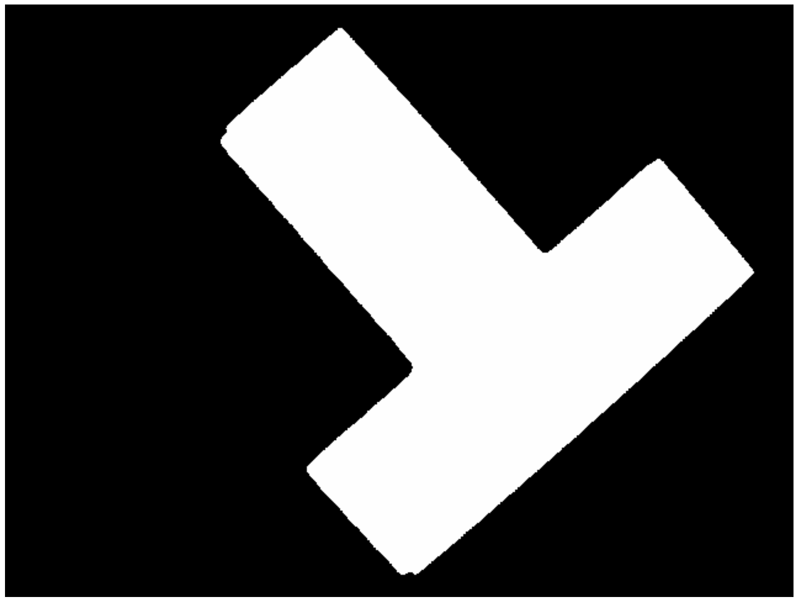
Slide credit: Bob Fisher

# Threshold Selection

Assume 2 big peaks, brighter background is higher:

1. Find biggest peak (background)
2. Find next biggest peak in darker direction
3. Find lowest point in trough between peaks



Slide credit: Bob Fisher

# Peak Pick Code

Omit special cases for ends of array and closing ‘end’s.

```
peak = find(tmp1 == max(tmp1));   % find largest peak

% find highest peak to left
xmaxl = -1;
for i = 2 : peak-1
  if tmp1(i-1) < tmp1(i) & tmp1(i) >= tmp1(i+1) ...
  & tmp1(i)>xmaxl
      xmaxl = tmp1(i);
      pkl = i;
```

Slide credit: Bob Fisher

```
% find deepest valley between peaks
xminl = max(tmp1)+1;
for i = pkl+1 : peak-1
  if tmp1(i-1) > tmp1(i) & tmp1(i) <= tmp1(i+1) ...
  & tmp1(i)<xminl
      xminl = tmp1(i);
      thresh = i;
```

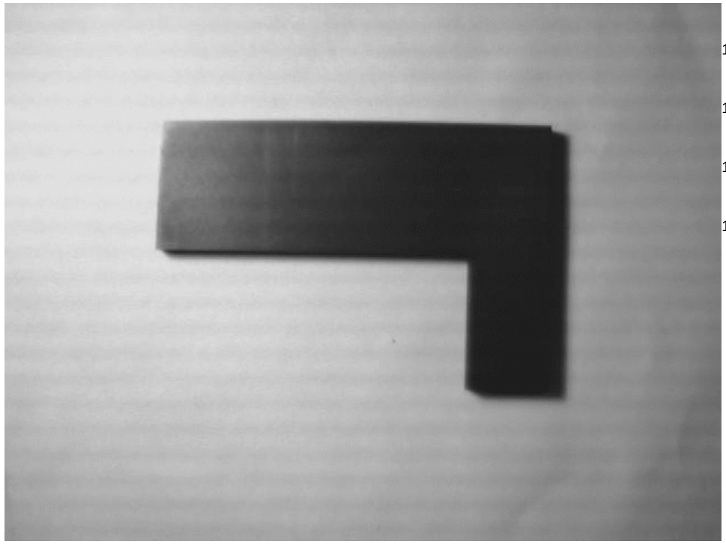
Slide credit: Bob Fisher

# Adaptive Thresholding

What if varying and unknown background? Can select threshold locally

At each pixel, use a different threshold calculated from an NxN window (N=100)

Use: threshold = mean(window) - Constant (eg. 12)



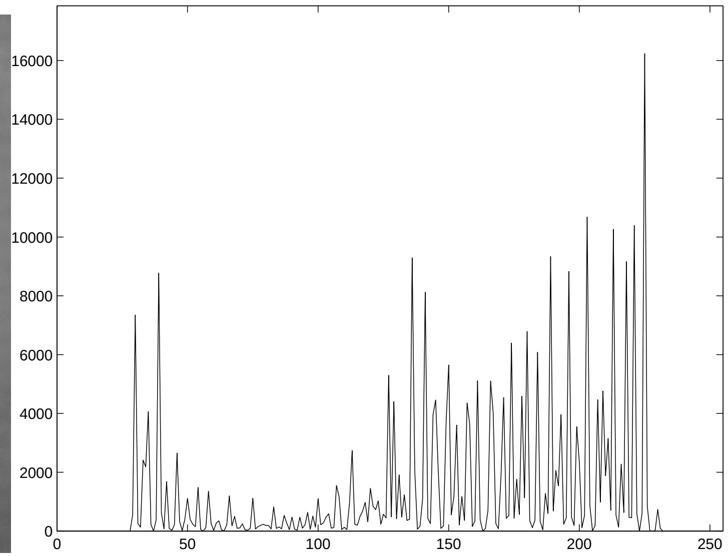



Image with intensity gradient

Histogram

Slide credit: Bob Fisher

# Adaptive Thresholding Code

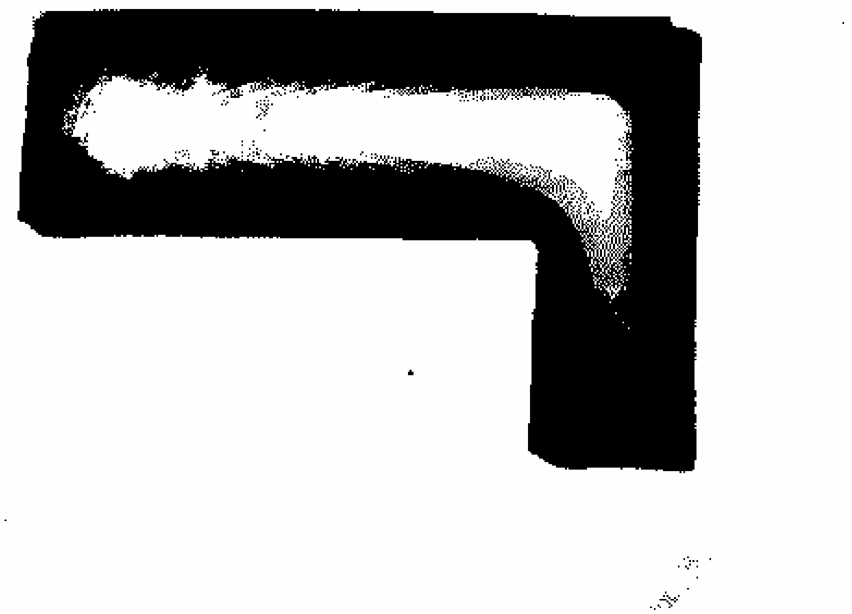
```
N = 100;
[H,W] = size(inimage);
outimage = zeros(H,W);
N2 = floor(N/2);
for i = 1+N2 : H-N2
  for j = 1+N2 : W-N2
    % extract subimage
    subimage = inimage(i-N2:i+N2,j-N2:j+N2);
    threshold = mean(mean(subimage)) - 12;
    if inimage(i,j) < threshold
     outimage(i,j) = 1;
    else
     outimage(i,j) = 0;
```

Slide credit: Bob Fisher

```
        end
    end
end
```

Slide credit: Bob Fisher

# Adaptive Thresholding Results



Selection has included shadow at bottom and right

Slide credit: Bob Fisher

# Background Removal

If known but spatially varying illumination

Reflectance: percentage of input illumination reflected. A function of the light source, viewer and surface colors and positions.

**Recall:**

background(r,c) = illumination(r,c)*bg_reflectance(r,c)

object(r,c) = illumination(r,c)*obj_reflectance(r,c)

Slide credit: Bob Fisher

Divide to remove illumination:

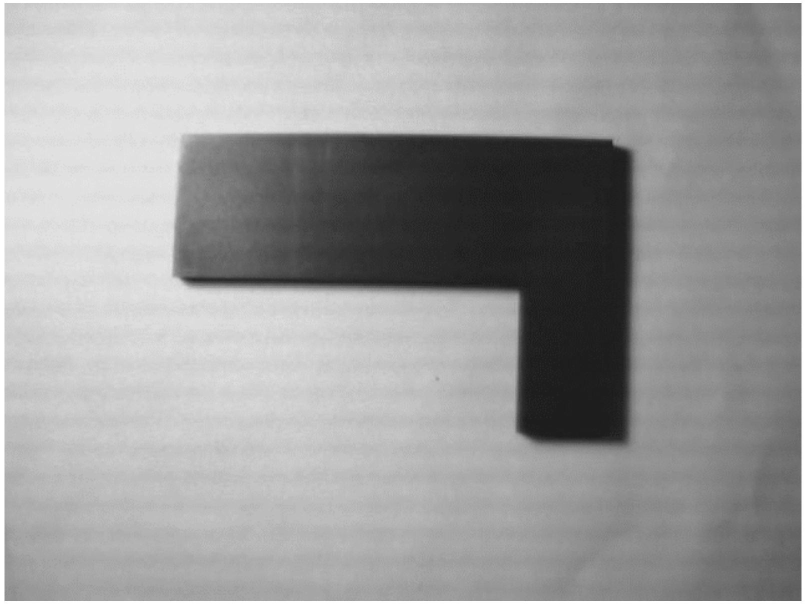
unknown(r,c)/background(r,c) =

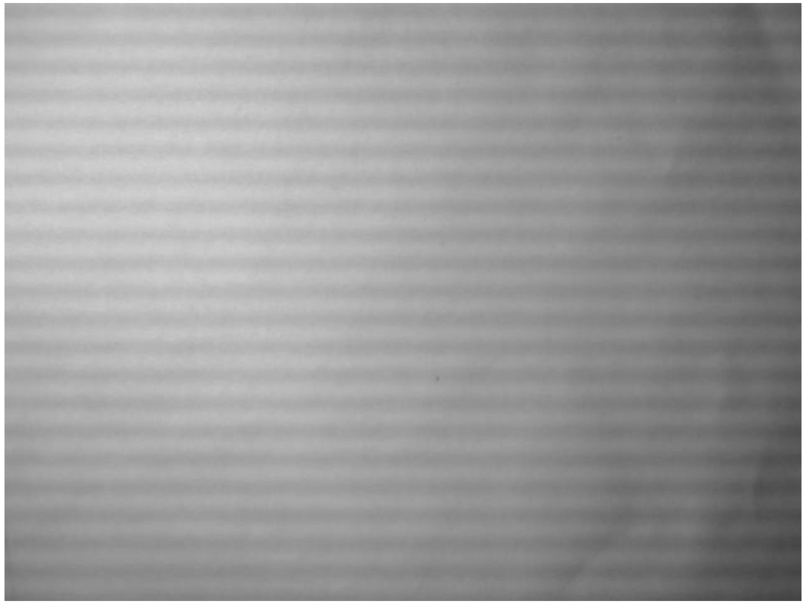
| | |
|---|---|
| 1 | if unknown = background |
| <<1 | if unknown = dark object |

Pick threshold in [0,1] e.g. 0.6

Slide credit: Bob Fisher

# Background removal results 1



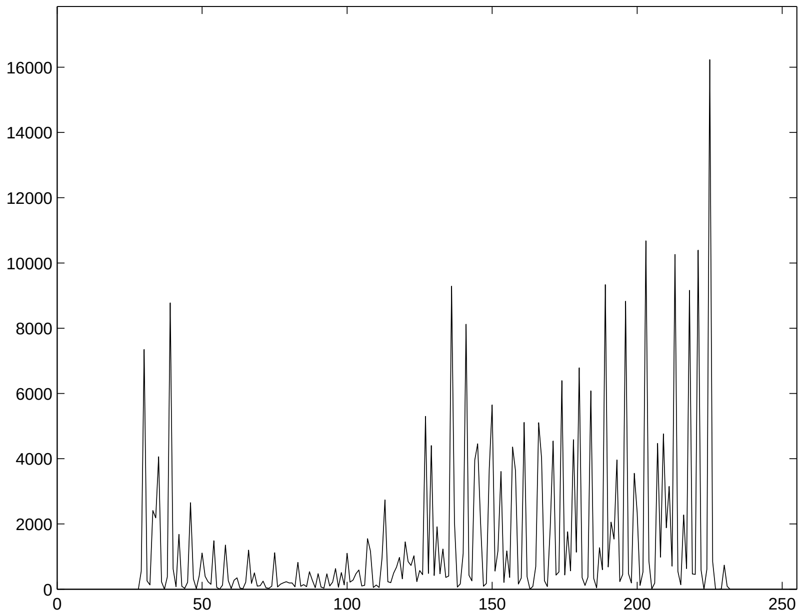


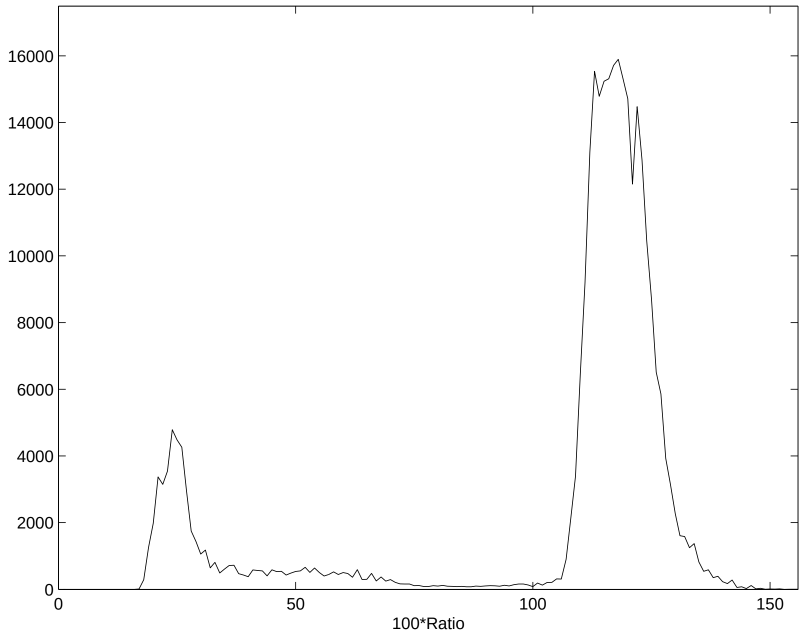
Part

Background

Slide credit: Bob Fisher

# Background removal results 2




Raw histogram

ratio histogram

Slide credit: Bob Fisher

# Background removal results 3

Has also included shadow below and right.

Slide credit: Bob Fisher

# Colour background removal

Before

After

change=open(2,coloror(thresh(35,abs(Before-After))))
(Use HSI instead of RGB to cope with illumination changes?)

Slide credit: Bob Fisher

# Colour background removal

Red change

Green change

ORed change

Opened

Slide credit: Bob Fisher

# Coping with varying lighting

Use normalised RGB:

$$(r, g, b) \rightarrow (\frac{r}{r+g+b}, \frac{g}{r+g+b}, \frac{b}{r+g+b})$$

Double illumination still gives same normalised RGB:

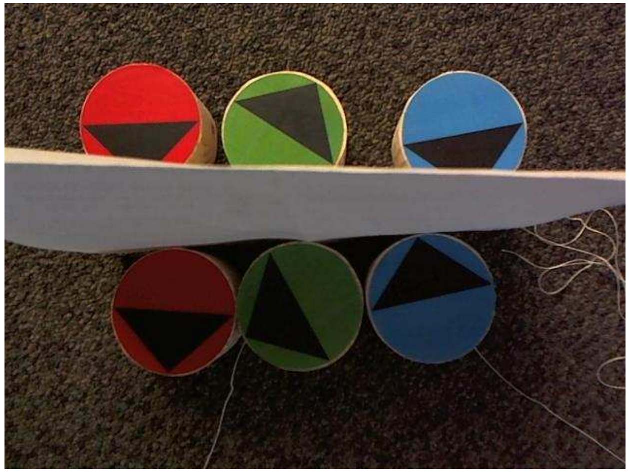
$$(\frac{r}{r+g+b}, \frac{g}{r+g+b}, \frac{b}{r+g+b})$$

$$= (\frac{2r}{2r+2g+2b}, \frac{2g}{2r+2g+2b}, \frac{2b}{2r+2g+2b})$$

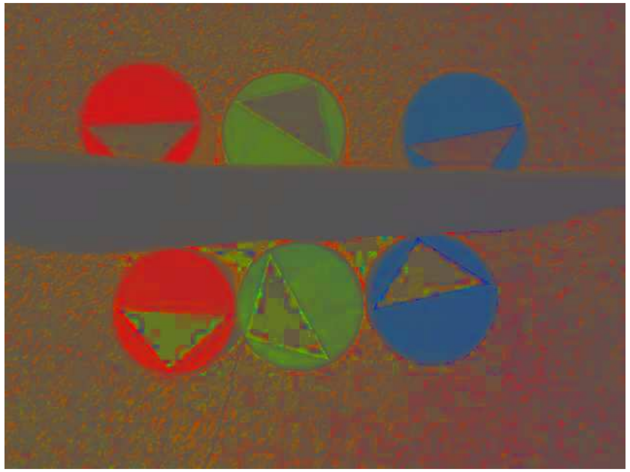
Slide credit: Bob Fisher

# Normalised RGB Example

Original



Normalised



Reduces shadow effects, too.

Slide credit: Bob Fisher

# Topics of This Lecture

- Problem definition and goals
- Greylevel segmentation by thresholding
- Background removal
- Canny edge detection
- Segmentation into multiple regions with mean-shift

# Edge detection

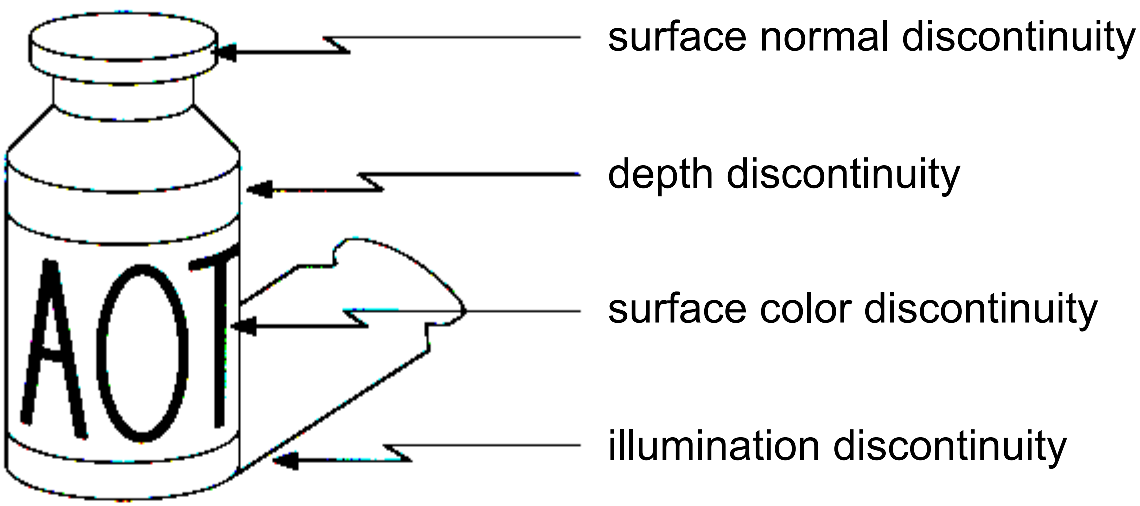
- **Goal:** Identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels

- **Ideal:** artist's line drawing (but artist is also using object-level knowledge)

Source: D. Lowe

# Origin of edges

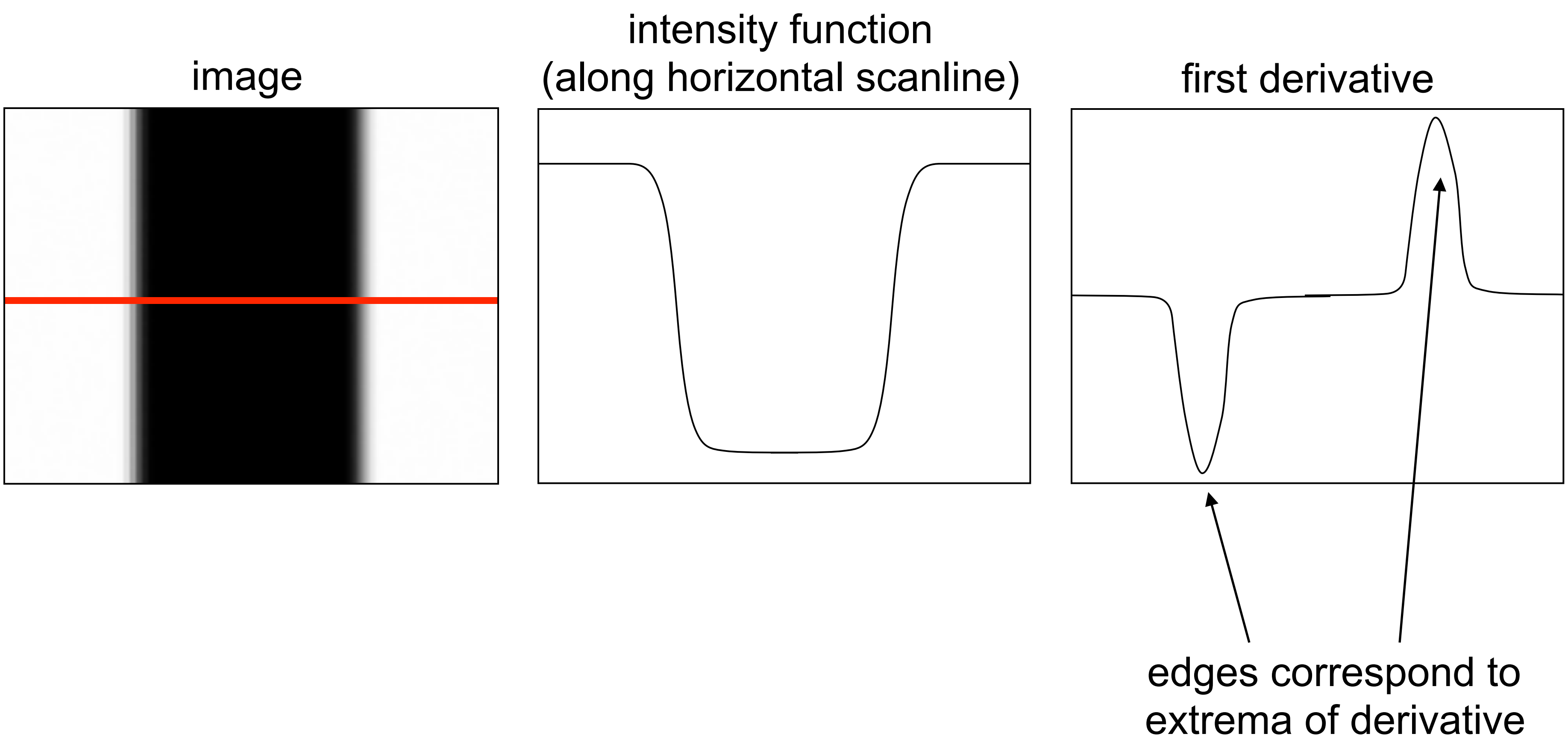
Edges are caused by a variety of factors:




Source: Steve Seitz

# Characterizing edges

- An edge is a place of rapid change in the image intensity function



# Derivatives with convolution

For 2D function f(x,y), the partial derivative is:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

For discrete data, we can approximate using finite differences:

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

How to implement the above? → convolutions!

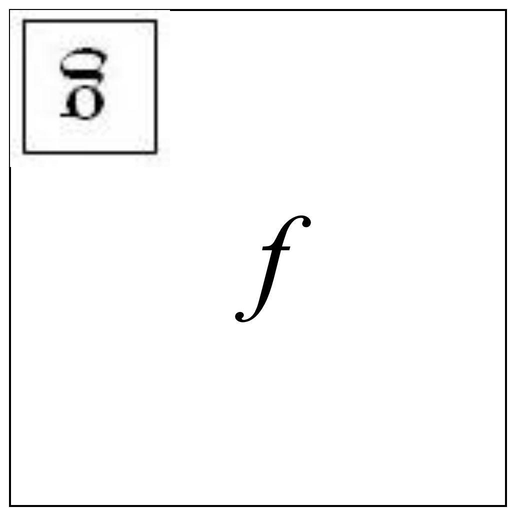
Source: K. Grauman

# Defining 2D convolutions

- Let *f* be the image and *g* be the kernel. The output of convolving *f* with *g* is denoted *f* * *g*.

$$(f * g)[m,n] = \sum_{k,l} f[m-k, n-l]\, g[k,l]$$

Convention: kernel is “flipped”




- MATLAB functions: conv2, filter2, imfilter

Source: F. Durand

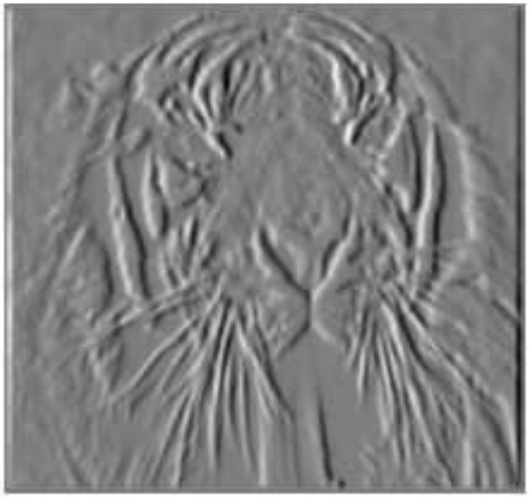
# Key properties

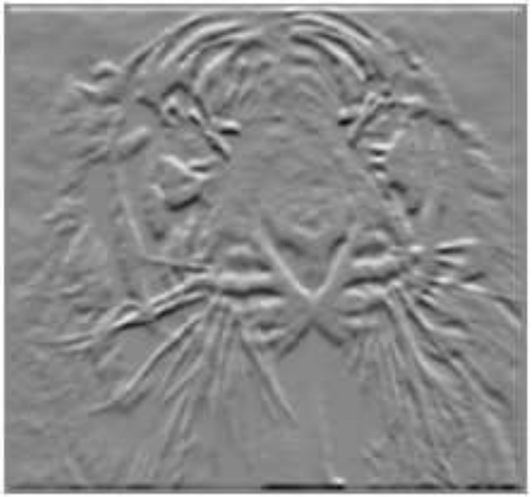
- **Linearity:** filter($f_1 + f_2$) = filter($f_1$) + filter($f_2$)
- **Shift invariance:** same behavior regardless of pixel location: filter(shift($f$)) = shift(filter($f$))
- Theoretical result: any linear shift-invariant operator can be represented as a convolution

# Partial derivatives of an image

$$\frac{\partial f(x,y)}{\partial x}$$

$$\frac{\partial f(x,y)}{\partial y}$$





| | |
|---|---|
| -1 | 1 |

| |
|---|
| -1 |
| 1 |

or

| |
|---|
| 1 |
| -1 |

Which shows changes with respect to x?

# Finite difference filters

Other approximations of derivative filters exist:

**Prewitt:** $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$ ; $M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

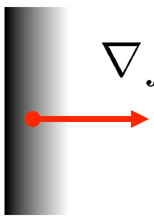
**Sobel:** $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ ; $M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

**Roberts:** $M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$ ; $M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

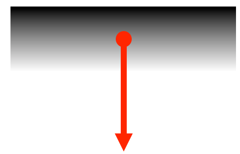
Source: K. Grauman

# Image gradient

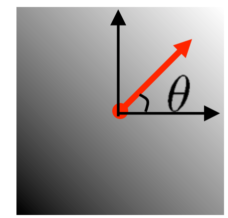
The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$



$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$




$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

The gradient points in the direction of most rapid increase in intensity

- How does this direction relate to the direction of the edge?

The gradient direction is given by $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

The *edge strength* is given by the gradient magnitude

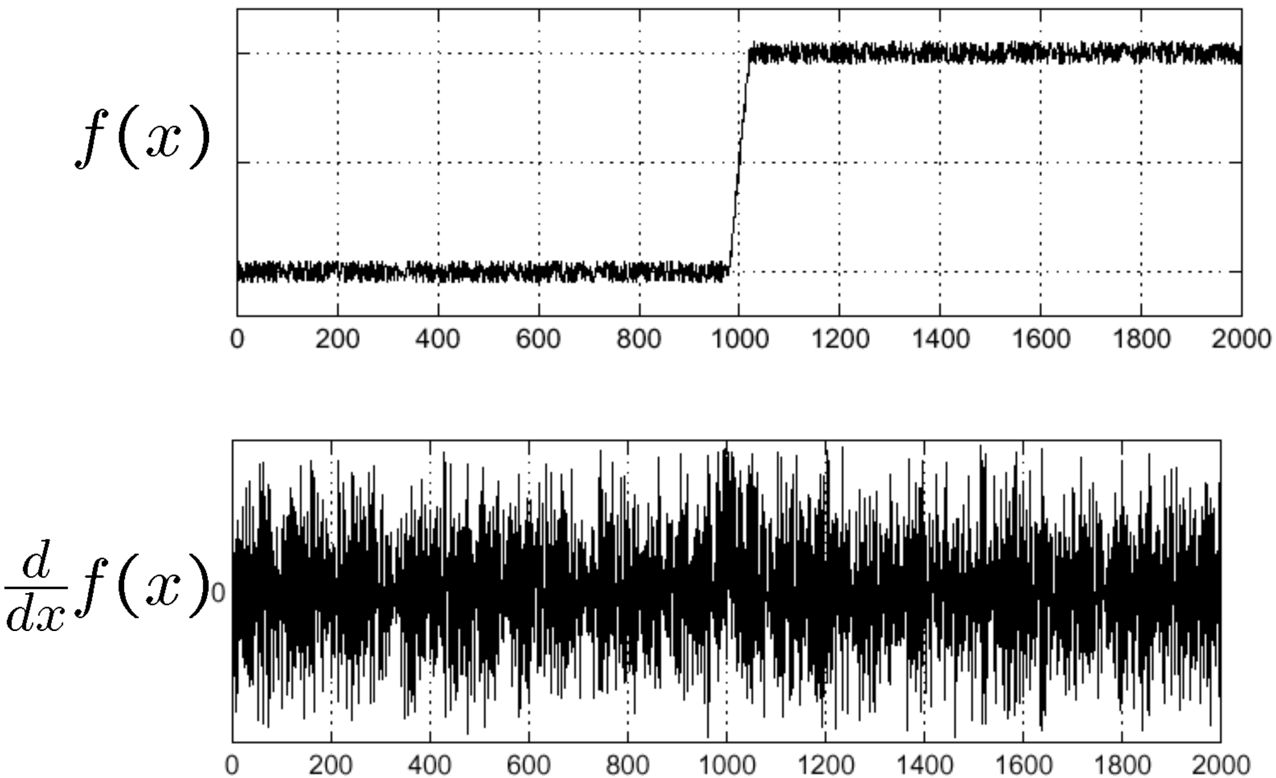
$$||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Source: Steve Seitz

# Effects of noise

## Consider a single row or column of the image

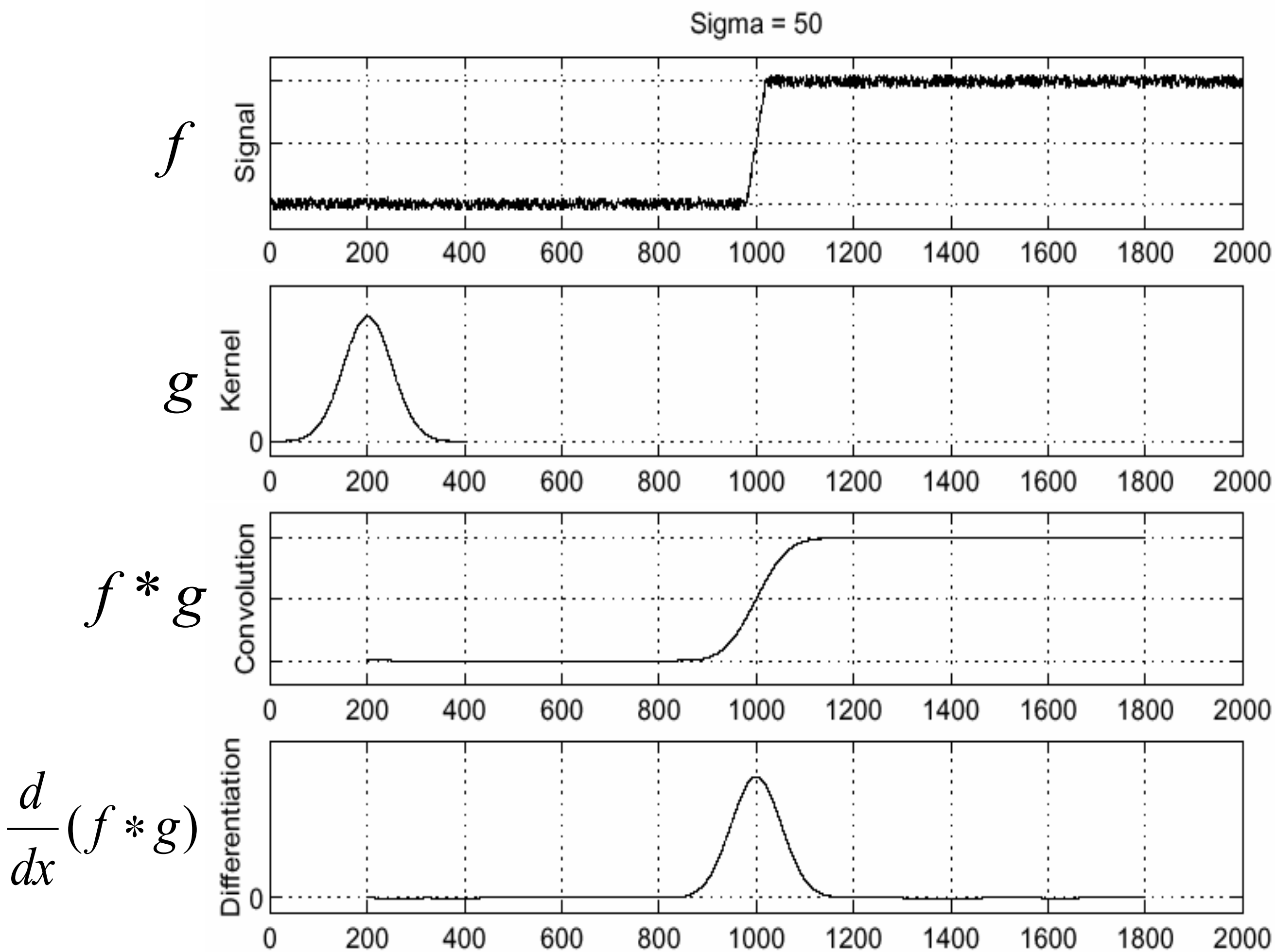
- Plotting intensity as a function of position gives a signal




## Where is the edge?

Source: S. Seitz

# Solution: smooth first




- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

Source: S. Seitz

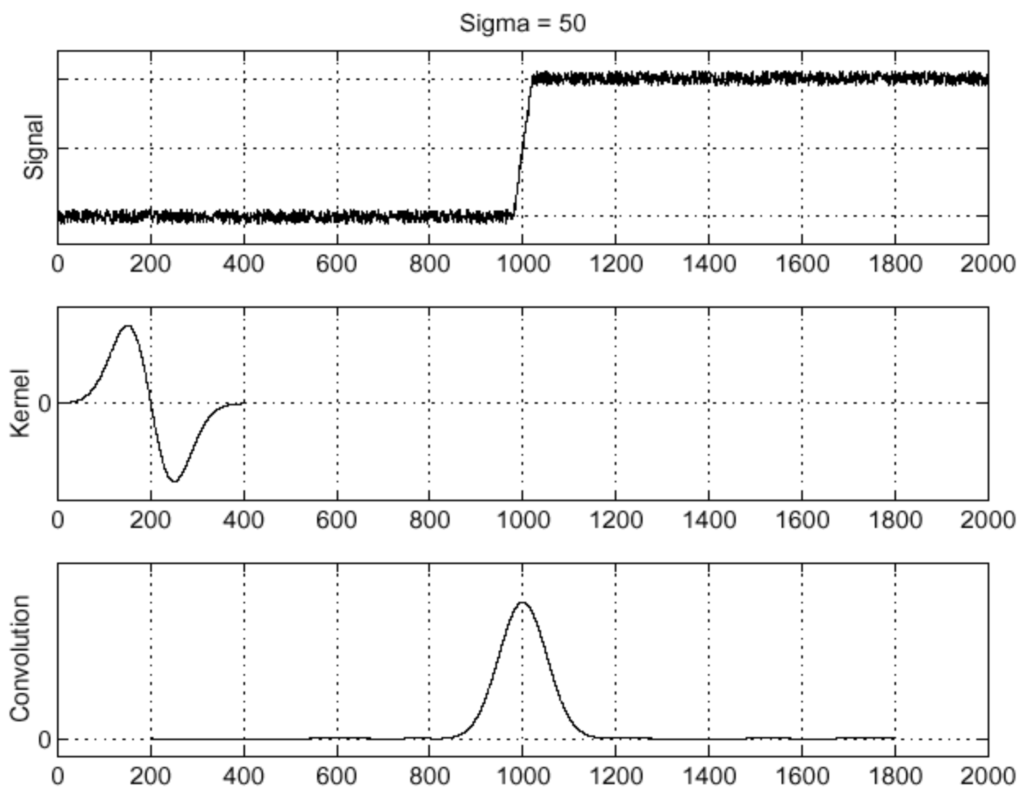
# Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative: $\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$
- This saves us one operation:

$f$

$\frac{d}{dx} g$

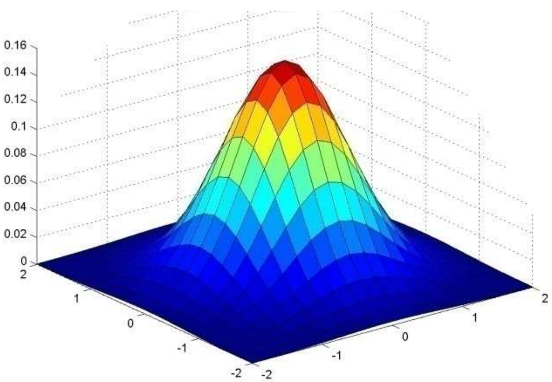
$f * \frac{d}{dx} g$

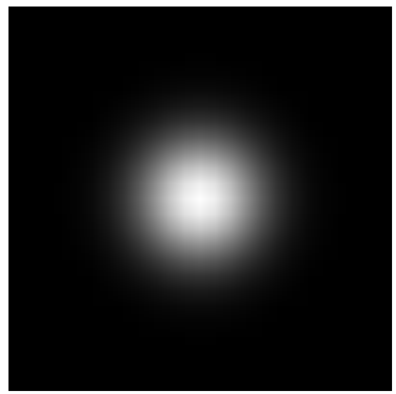



Source: S. Seitz

# Now in 2D: Gaussian Kernel

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$



| | | | | |
|---|---|---|---|---|
| 0.003 | 0.013 | 0.022 | 0.013 | 0.003 |
| 0.013 | 0.059 | 0.097 | 0.059 | 0.013 |
| 0.022 | 0.097 | 0.159 | 0.097 | 0.022 |
| 0.013 | 0.059 | 0.097 | 0.059 | 0.013 |
| 0.003 | 0.013 | 0.022 | 0.013 | 0.003 |

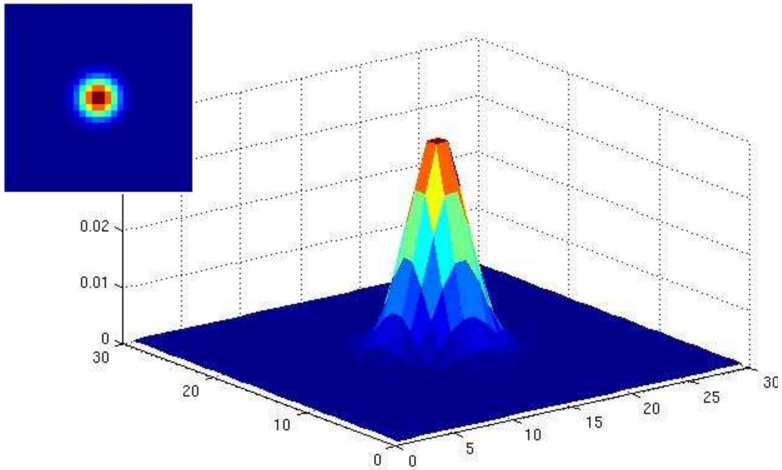
5 x 5, $\sigma$ = 1

- Constant factor at front makes volume sum to 1 (can be ignored when computing the filter values, as we should renormalize weights to sum to 1 in any case)

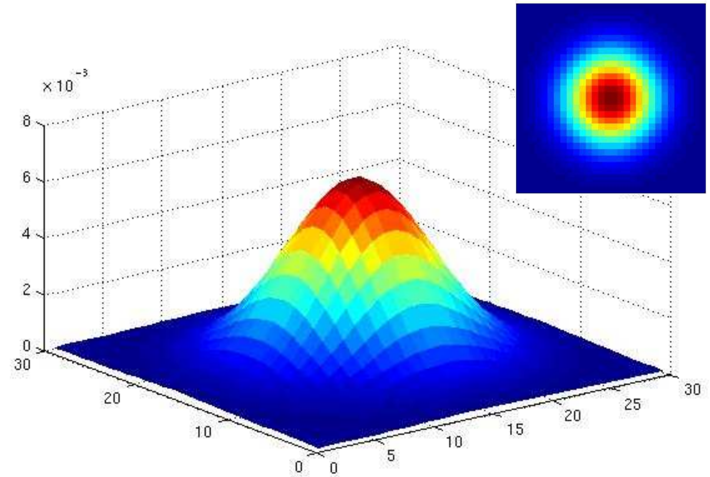
Source: C. Rasmussen

# Now in 2D: Gaussian Kernel

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$



σ = 2 with 30 x 30 kernel



σ = 5 with 30 x 30 kernel

- Standard deviation σ: determines extent of smoothing

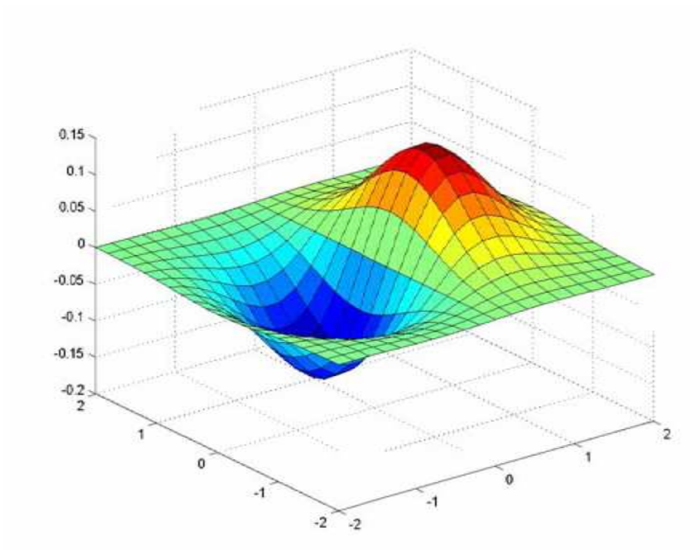
Source: K. Grauman

# Gaussian filters

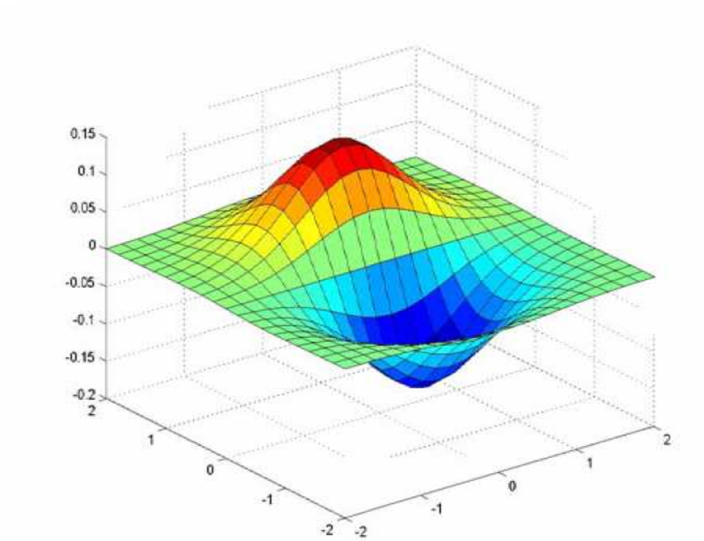
- Remove "high-frequency" components from the image (low-pass filter)
- Convolution with self is another Gaussian
  - So can smooth with small-$\sigma$ kernel, repeat, and get same result as larger-$\sigma$ kernel would have
  - Convolving two times with Gaussian kernel with std. dev. $\sigma$ is same as convolving once with kernel with std. dev. $\sigma\sqrt{2}$
- *Separable* kernel
  - Factors into product of two 1D Gaussians → enable efficient implementations

Source: K. Grauman

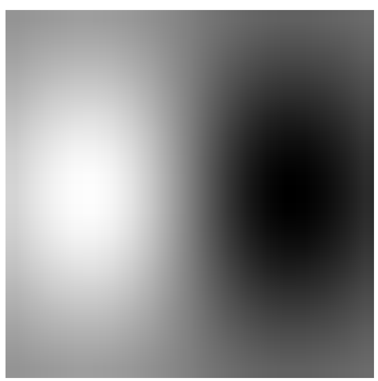
# Derivative of Gaussian filter in 2D

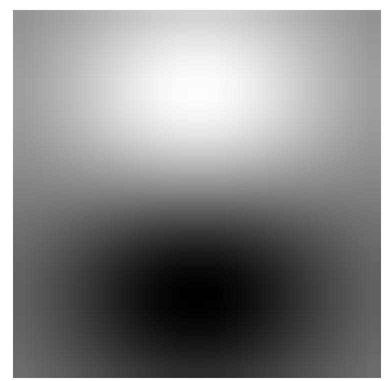


*x*-direction



*y*-direction





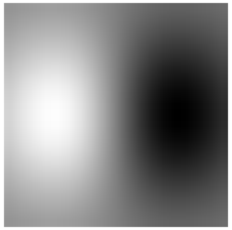
Which one finds horizontal/vertical edges?

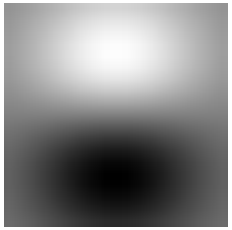
# Review: Smoothing vs. derivative filters

## Smoothing filters

- Gaussian: remove "high-frequency" components; "low-pass" filter
- Can the values of a smoothing filter be negative?
- What should the values sum to?
  - **One:** constant regions are not affected by the filter

## Derivative filters





- Derivatives of Gaussian
- Can the values of a derivative filter be negative?
- What should the values sum to?
  - **Zero:** no response in constant regions
- High absolute value at points of high contrast

# The Canny edge detector

original image

Slide credit: Steve Seitz

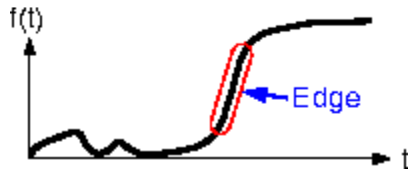
# The Canny edge detector

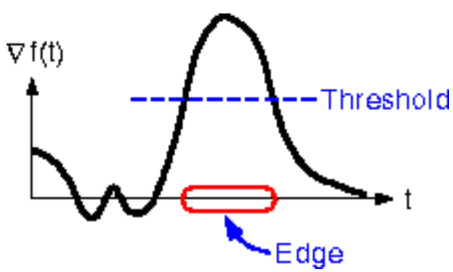
norm of the gradient

# The Canny edge detector

thresholding

# The Canny edge detector

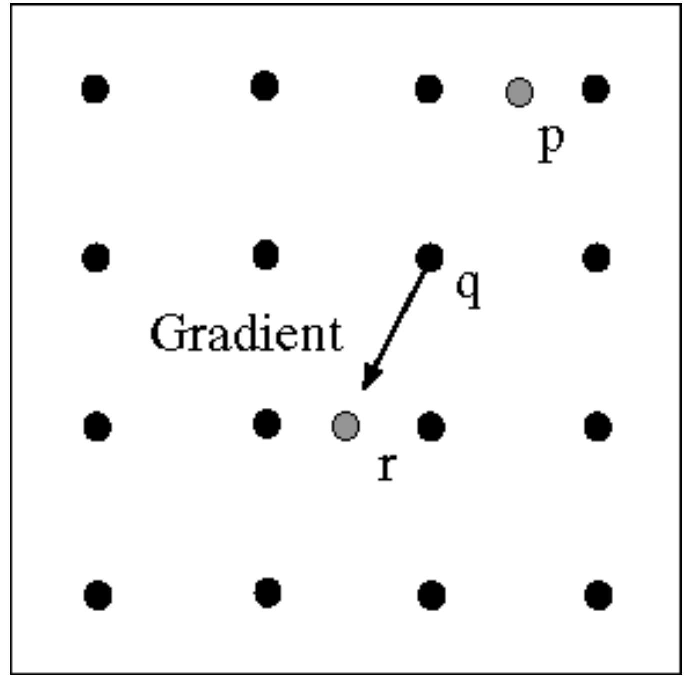



How to turn these thick regions of the gradient into curves?

thresholding

# Non-maximum suppression




Check if pixel is local maximum along gradient direction, select single max across width of the edge

- requires checking interpolated pixels p and r

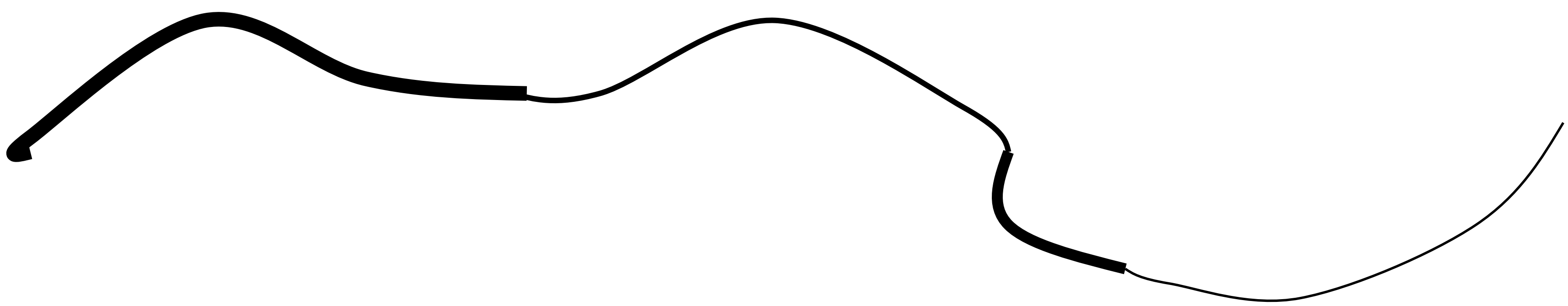
# The Canny edge detector

Problem: pixels along this edge didn't survive the thresholding

thinning
(non-maximum suppression)

# Hysteresis thresholding

Use a high threshold to start edge curves, and a low threshold to continue them.



Source: Steve Seitz

# Hysteresis thresholding

**original image**

**high threshold (strong edges)**

**low threshold (weak edges)**

**hysteresis threshold**

Source: L. Fei-Fei

# Recap: Canny edge detector

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. **Non-maximum suppression**:
   - Thin wide “ridges” down to single pixel width
4. **Linking and thresholding** (**hysteresis**):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

MATLAB: `edge(image, ‘canny’);`

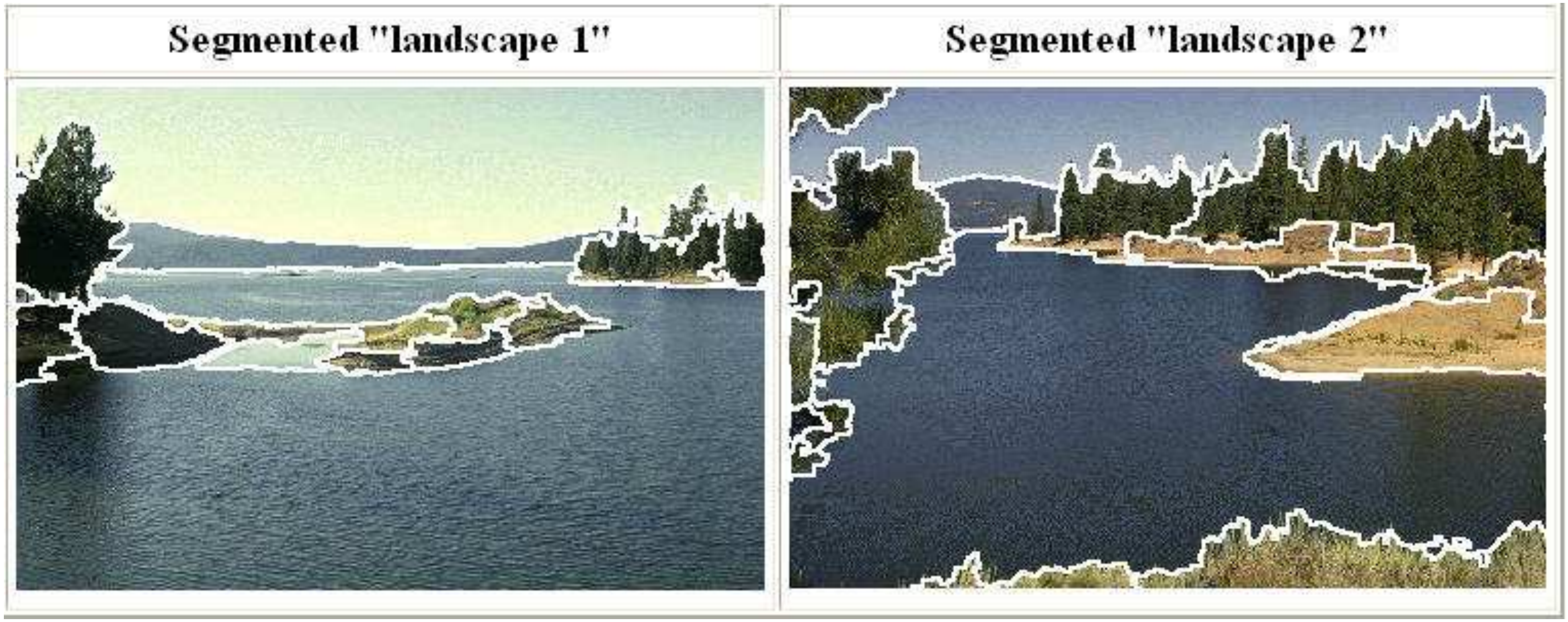
J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

# Topics of This Lecture

- Problem definition and goals
- Greylevel segmentation by thresholding
- Background removal
- Canny edge detection
- Segmentation into multiple regions with mean-shift

# Mean-Shift Segmentation

- An advanced and versatile technique for clustering-based segmentation

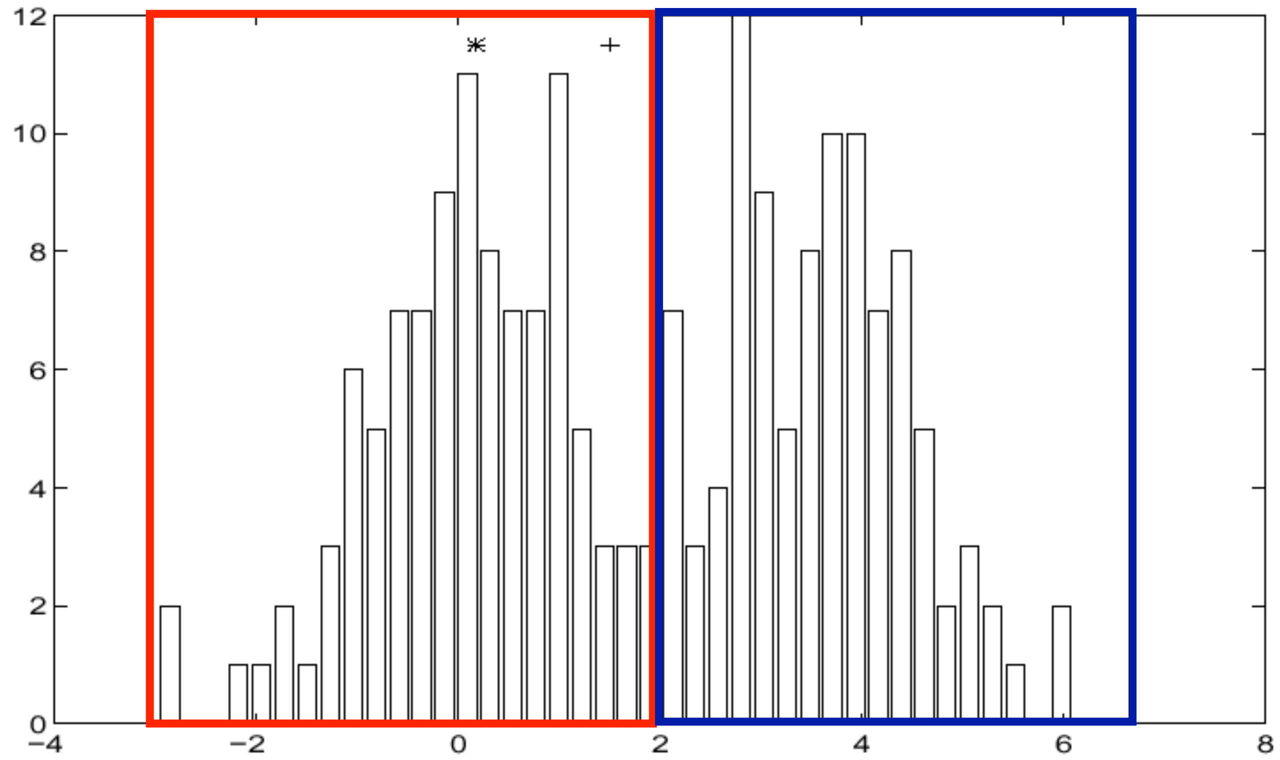



http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

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

Slide credit: Svetlana Lazebnik

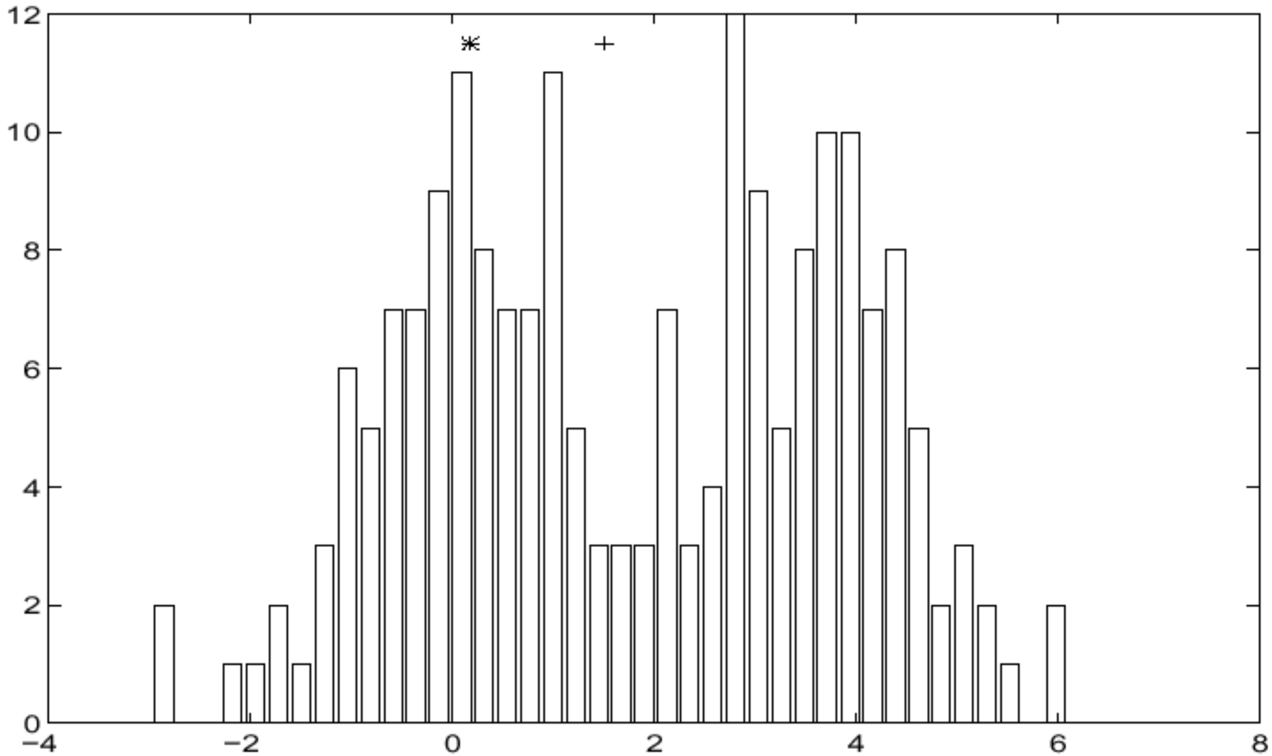
# Finding Modes in a Histogram




- How many modes are there?
  - *Mode* = local maximum of a given distribution
  - Easy to see, hard to compute

Slide adapted from Steve Seitz

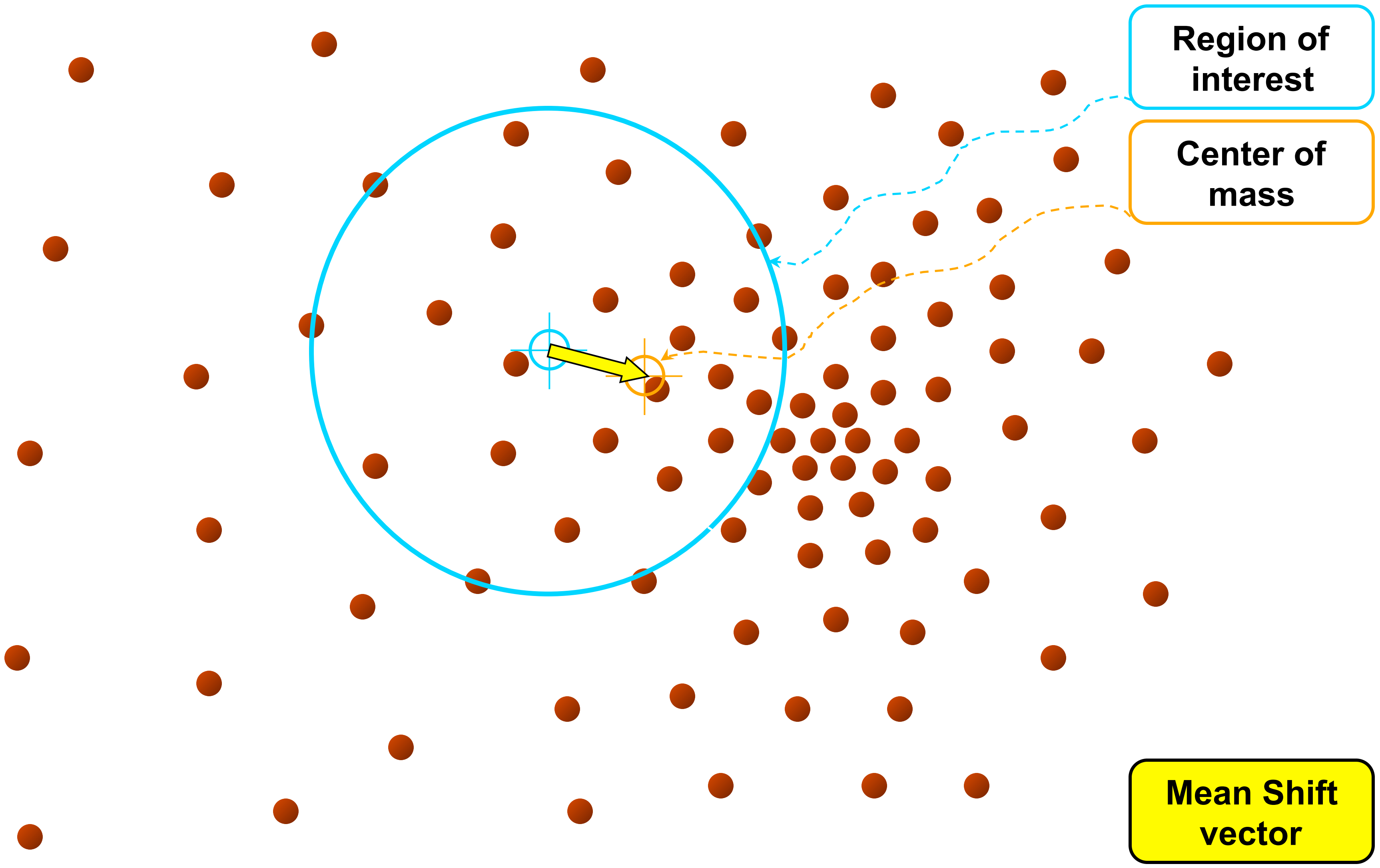
# Mean-Shift Algorithm




- **Iterative Mode Search**
  1. **Initialize random seed center and window W**
  2. **Calculate center of gravity (the "mean") of W:** $\sum_{x \in W} x H(x)$
  3. **Shift the search window to the mean**
  4. **Repeat steps 2+3 until convergence**

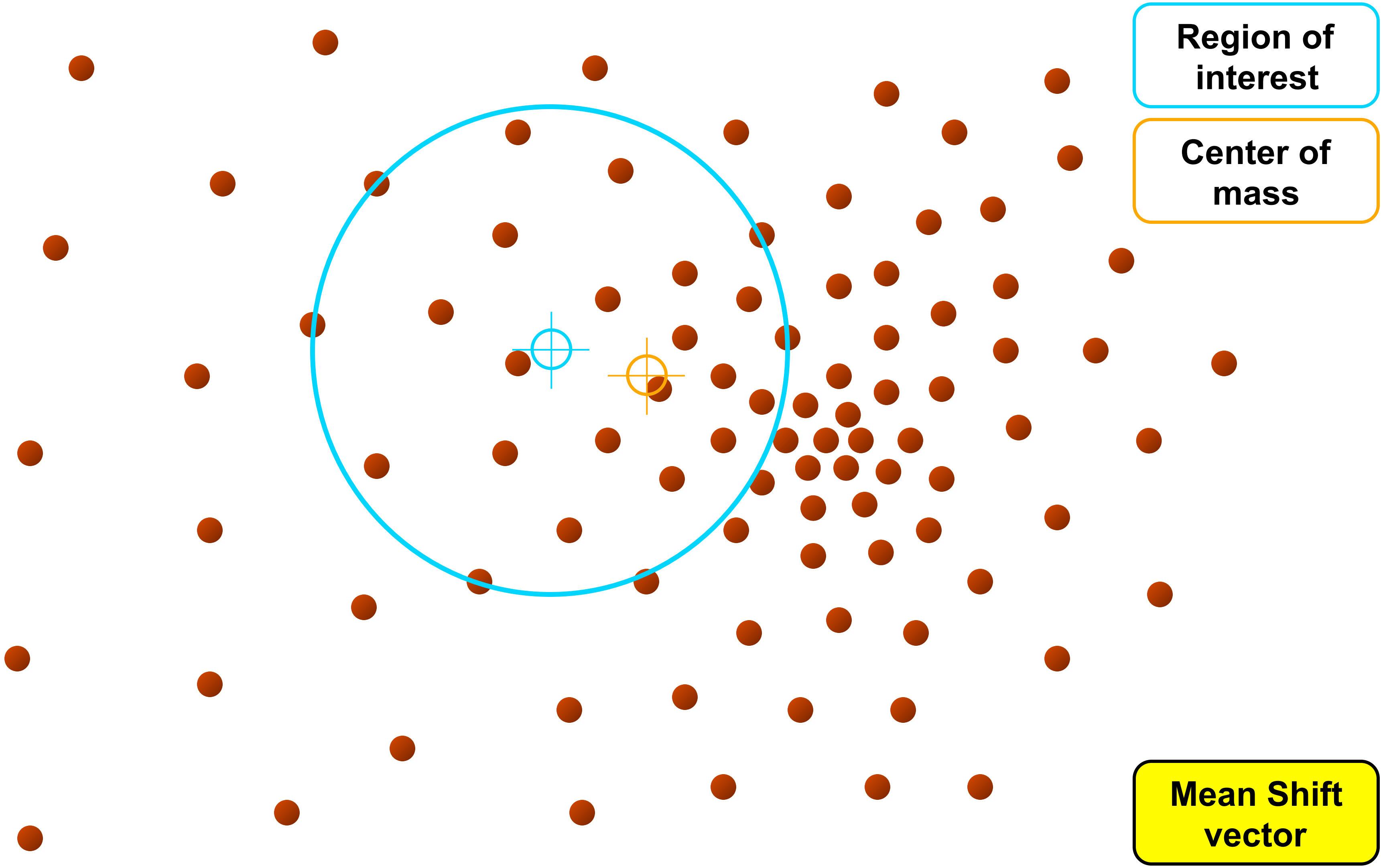
Slide adapted from Steve Seitz

# Mean-Shift



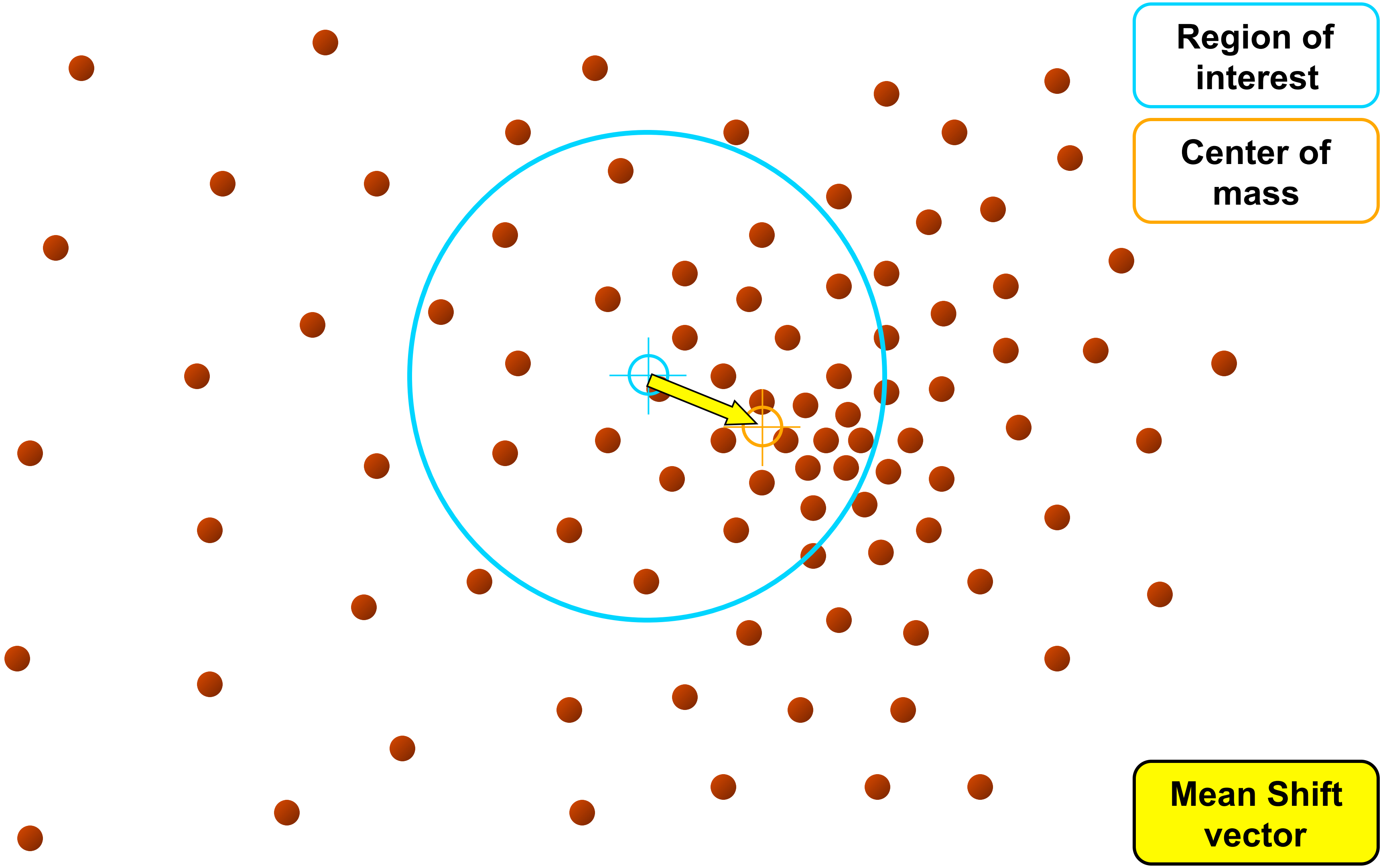

Slide by Y. Ukrainitz & B. Sarel

# Mean-Shift



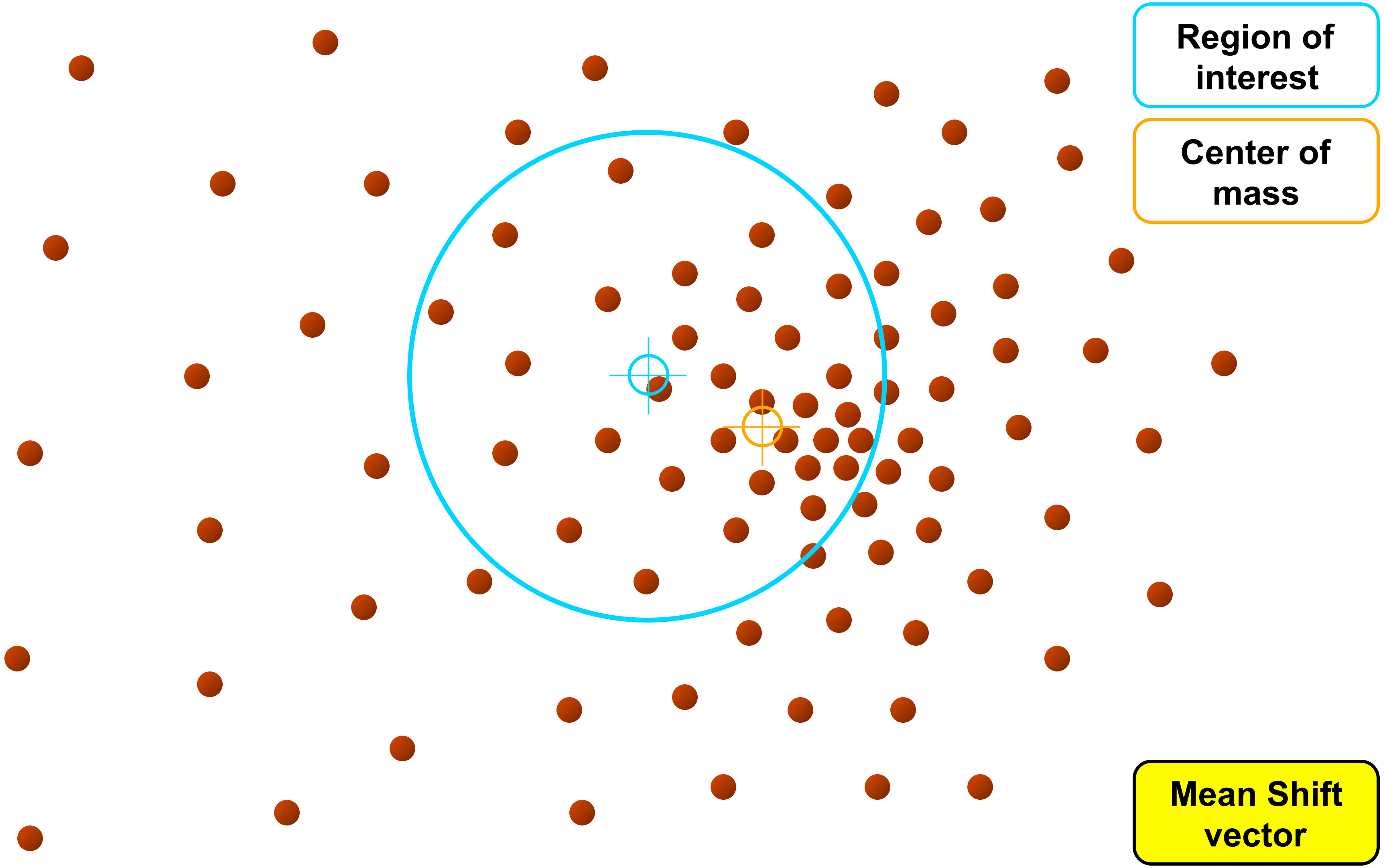

Slide by Y. Ukrainitz & B. Sarel

# Mean-Shift



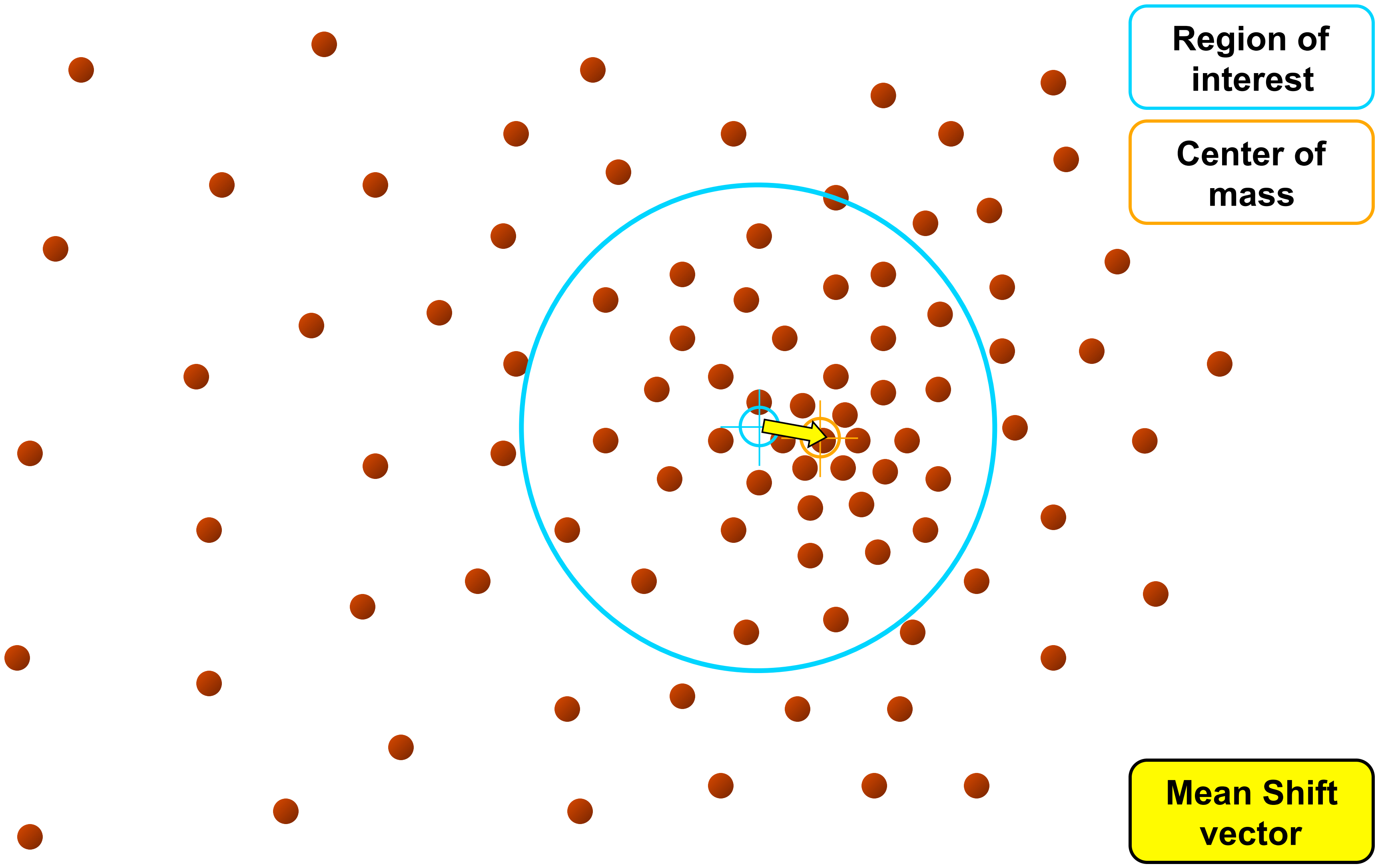

Slide by Y. Ukrainitz & B. Sarel

# Mean-Shift



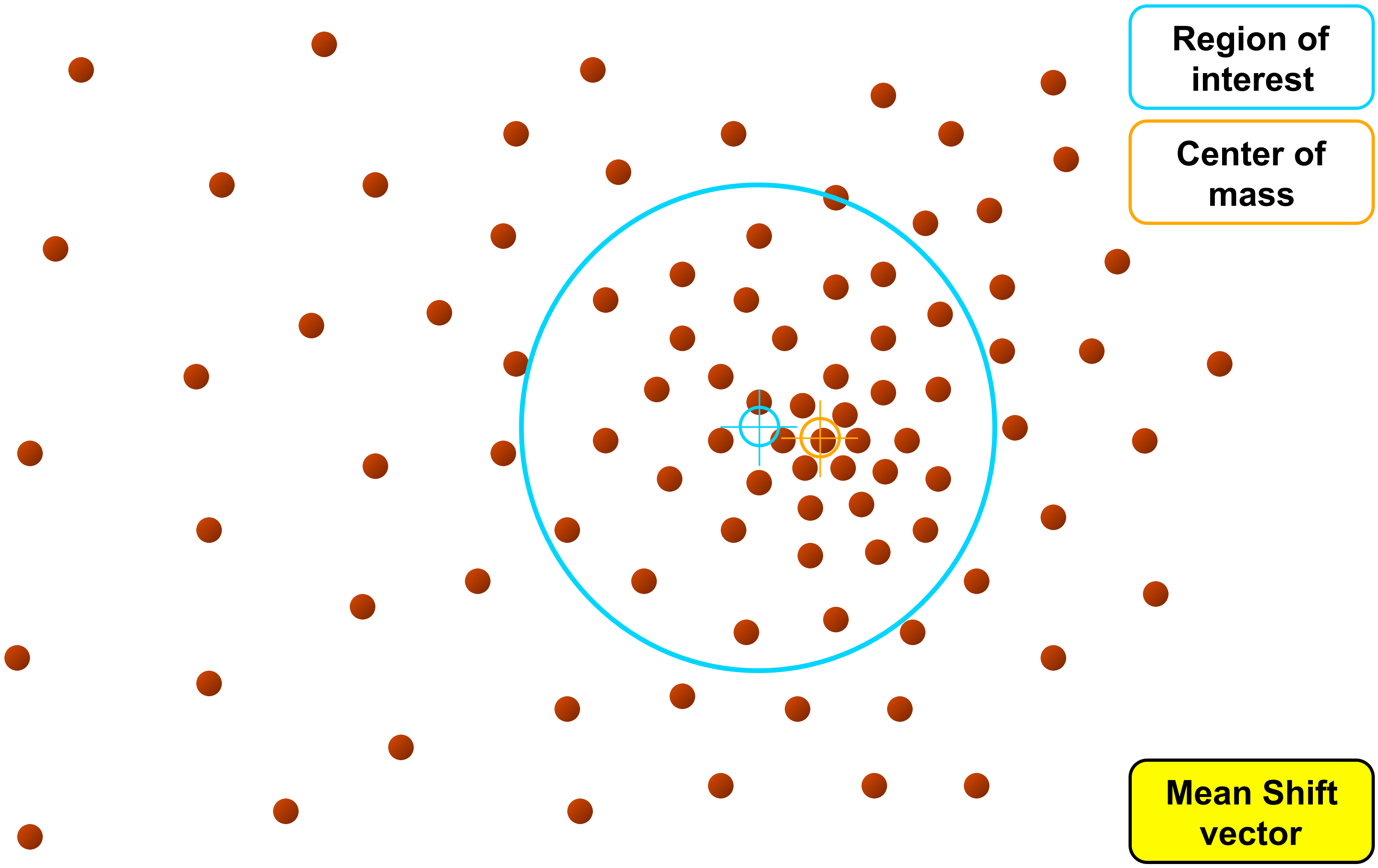

Slide by Y. Ukrainitz & B. Sarel

# Mean-Shift



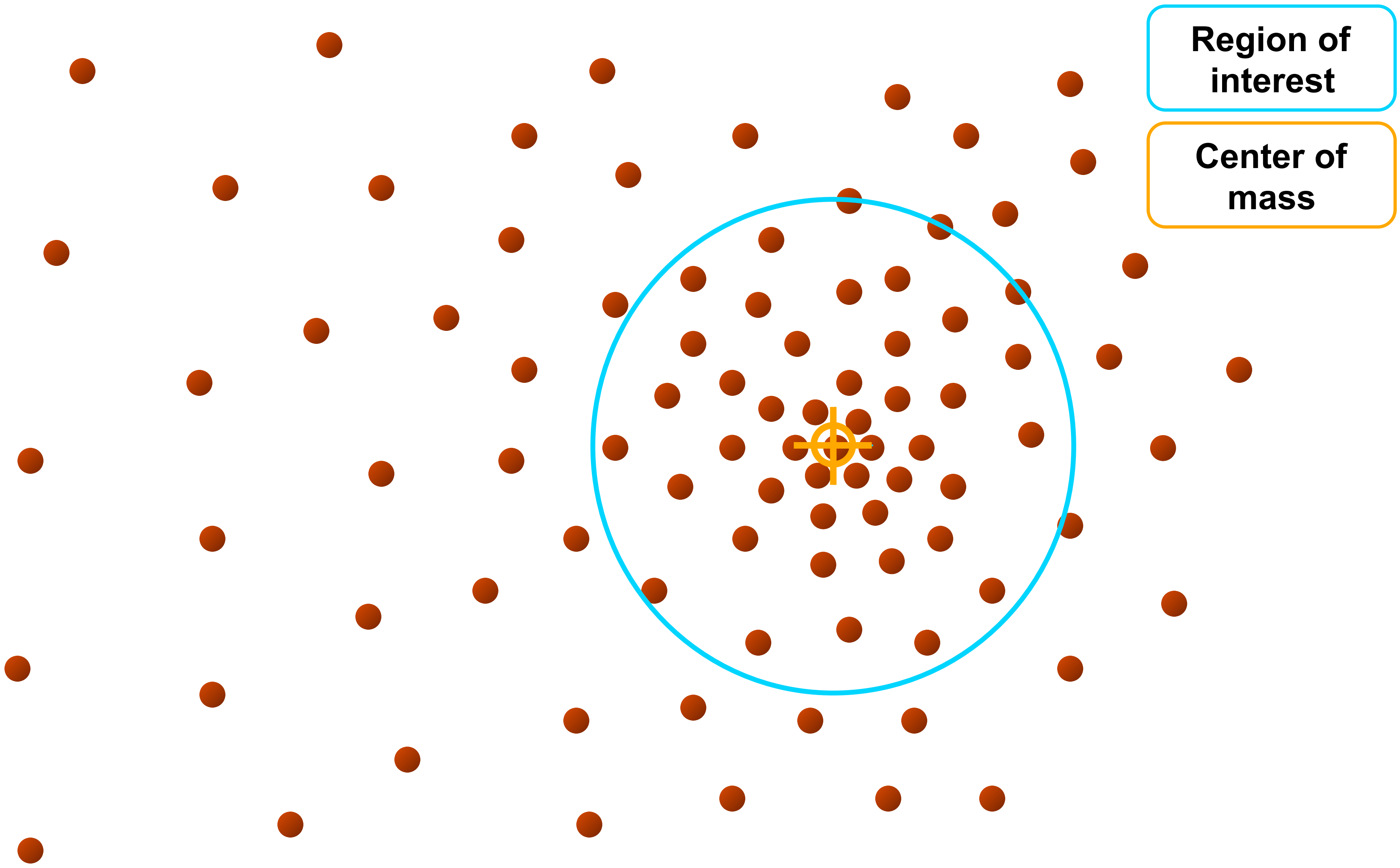

Slide by Y. Ukrainitz & B. Sarel

# Mean-Shift



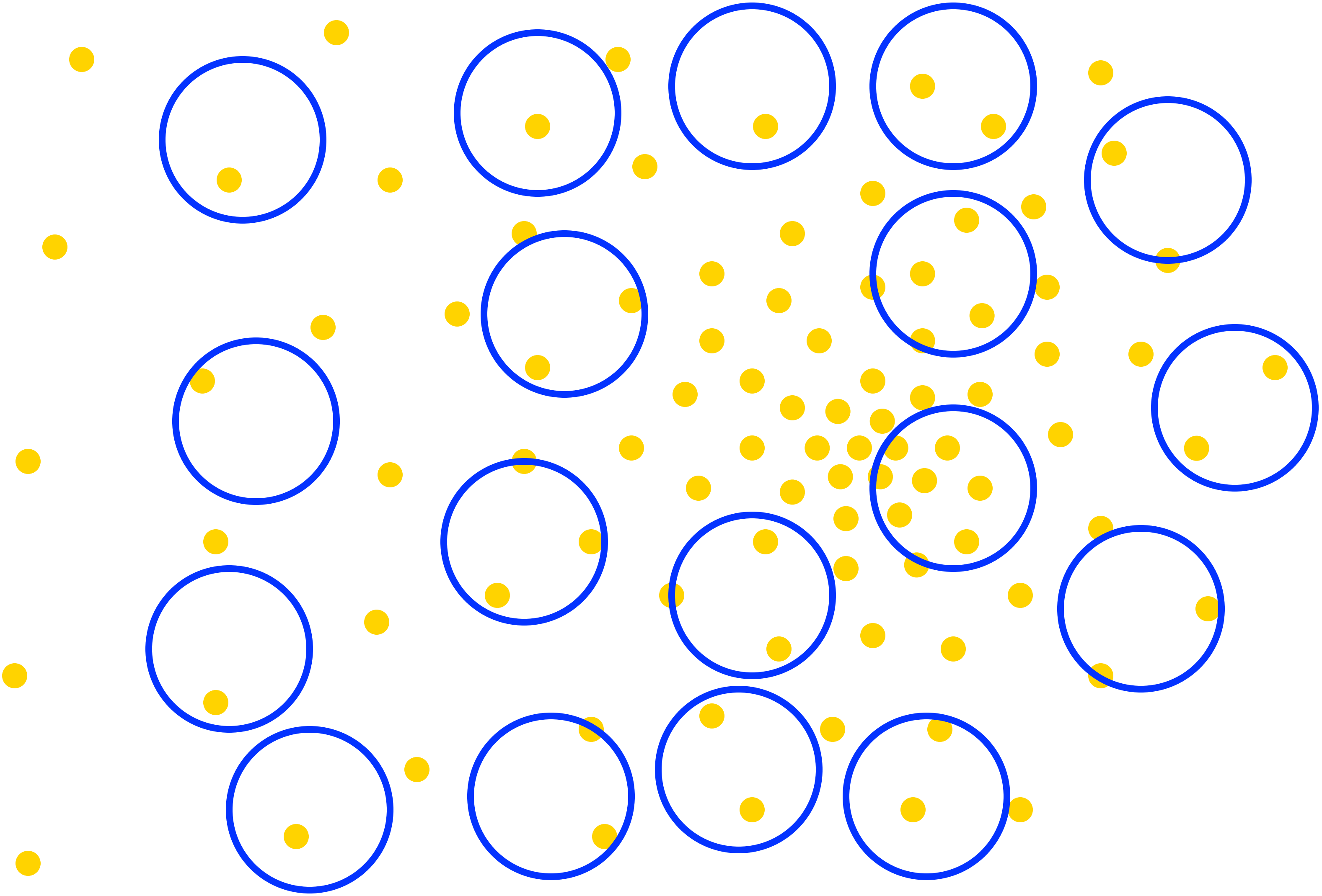

Slide by Y. Ukrainitz & B. Sarel

# Mean-Shift




Slide by Y. Ukrainitz & B. Sarel

# Real Modality Analysis

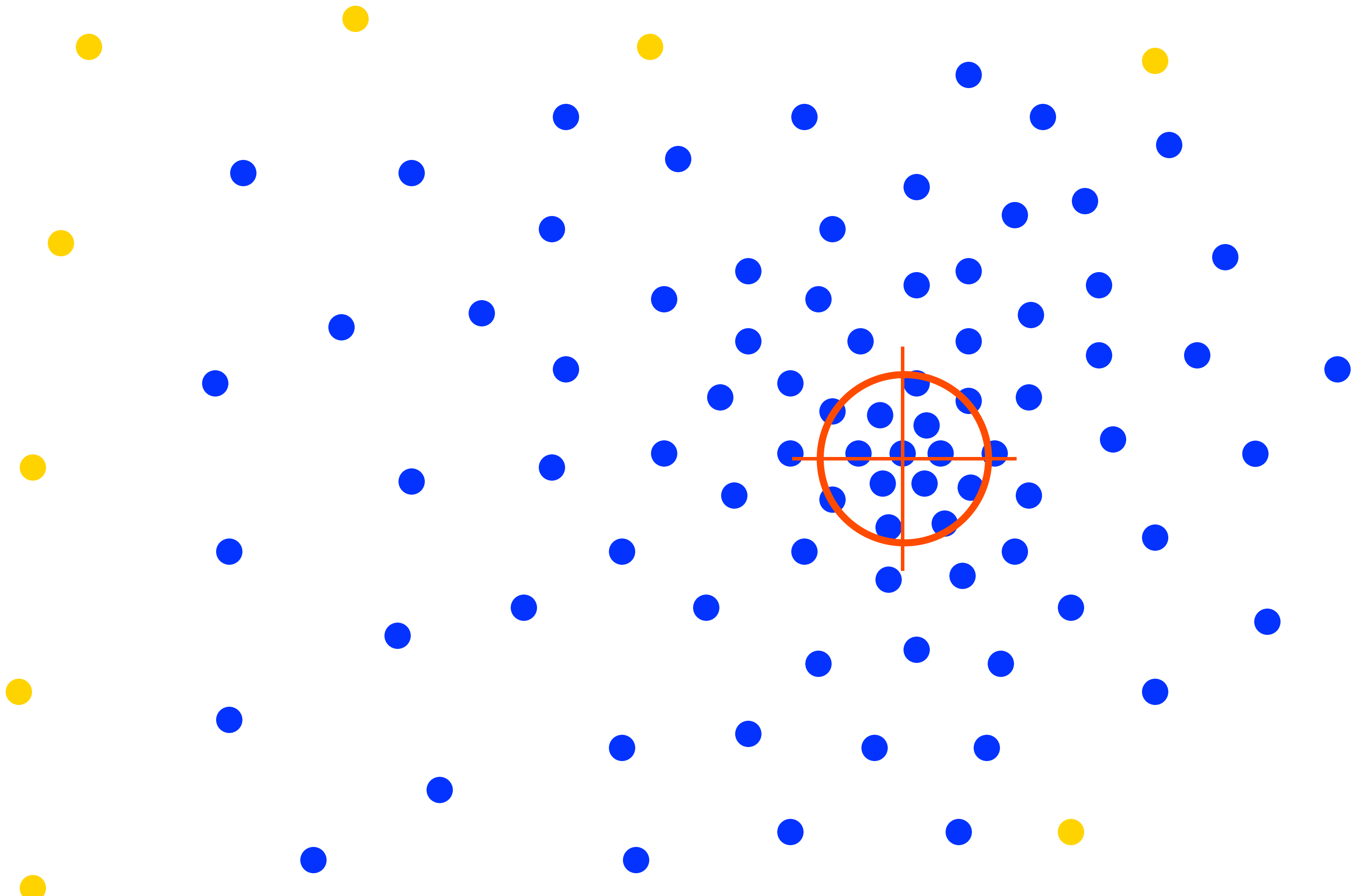


Tessellate the space with windows

Run the procedure in parallel

Slide by Y. Ukrainitz & B. Sarel

# Real Modality Analysis

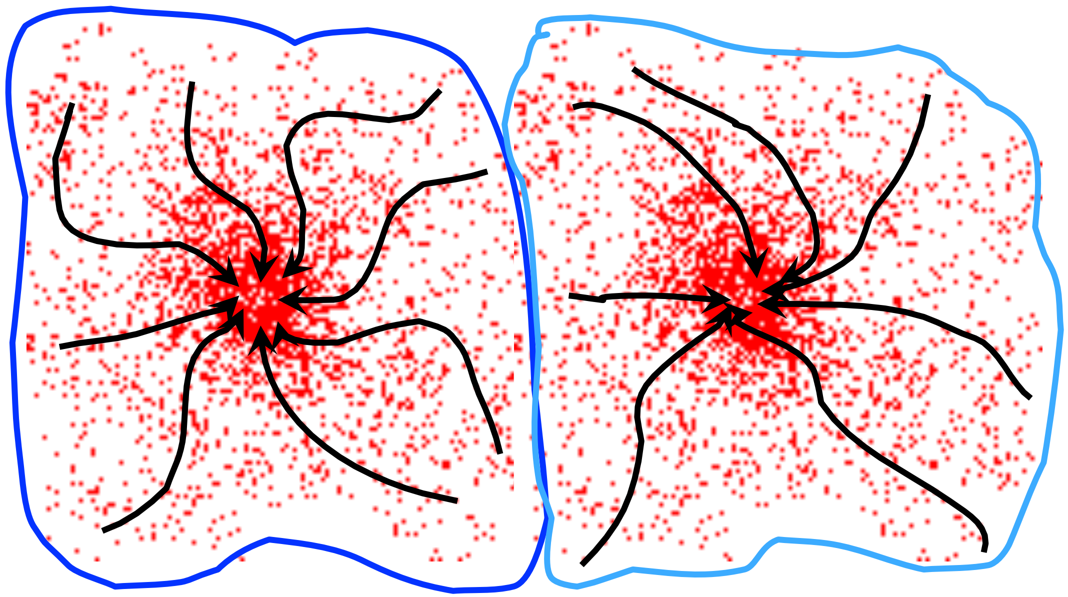


The blue data points were traversed by the windows towards the mode.

Slide by Y. Ukrainitz & B. Sarel

# Mean-Shift Clustering

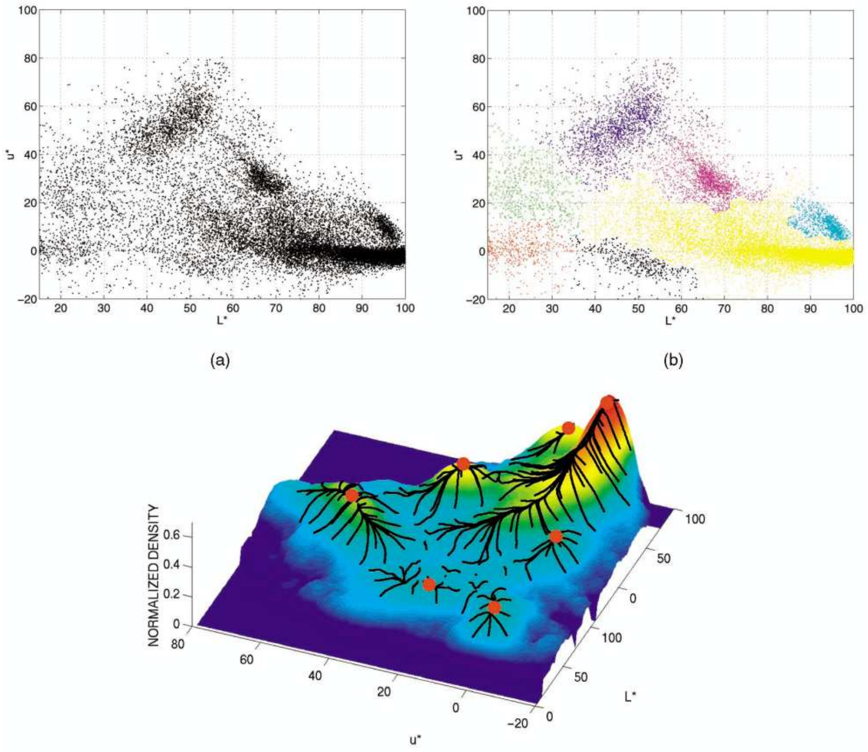
- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode



Slide by Y. Ukrainitz & B. Sarel

# Mean-Shift Clustering/Segmentation

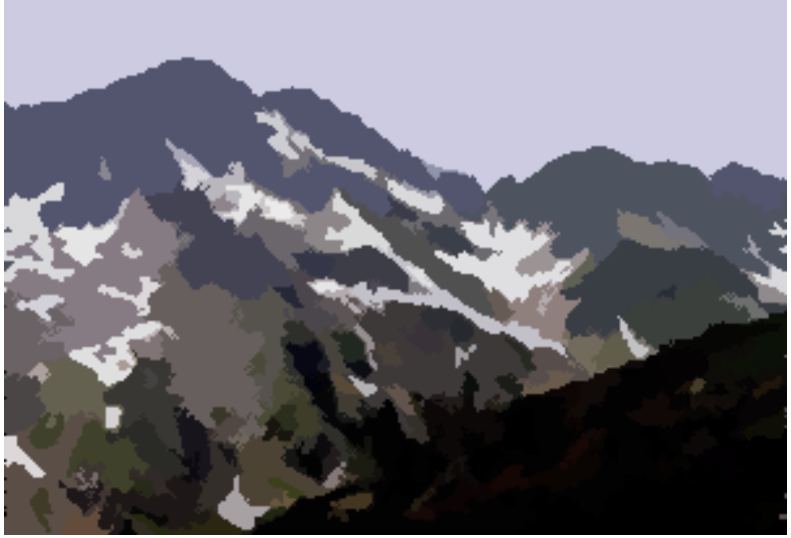
- Choose features (color, gradients, texture, etc)
- Initialize windows at individual pixel locations
- Start mean-shift from each window until convergence
- Merge windows that end up near the same “peak” or mode




Slide adapted from Svetlana Lazebnik

# Mean-Shift Segmentation Results



http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

Slide credit: Svetlana Lazebnik

# More Results

Slide credit: Svetlana Lazebnik

# Summary Mean-Shift

- **Pros**
  - General, application-independent tool
  - Model-free, does not assume any prior shape (spherical, elliptical, etc.) on data clusters
  - Just a single parameter (window size h)
    - h has a physical meaning (unlike k-means) == scale of clustering
  - Finds variable number of modes given the same h
  - Robust to outliers

- **Cons**
  - Output depends on window size h
  - Window size (bandwidth) selection is not trivial
  - Computationally rather expensive
  - Does not scale well with dimension of feature space

Slide adapted from Svetlana Lazebnik