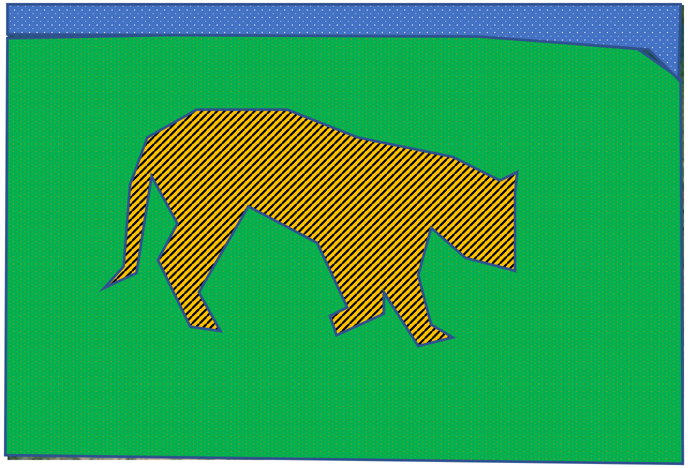
Grouping

What is grouping?

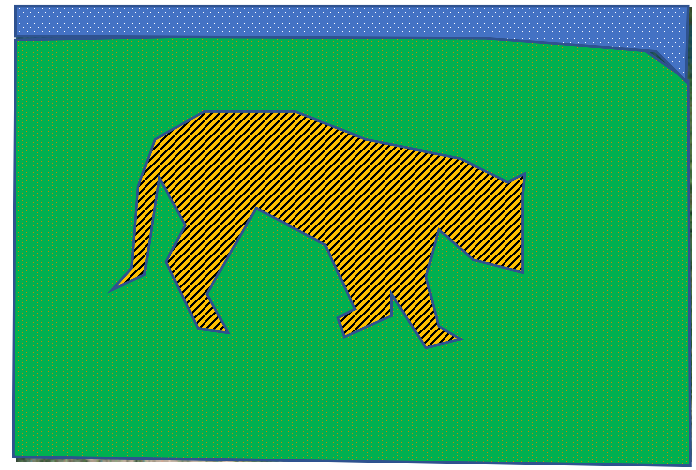


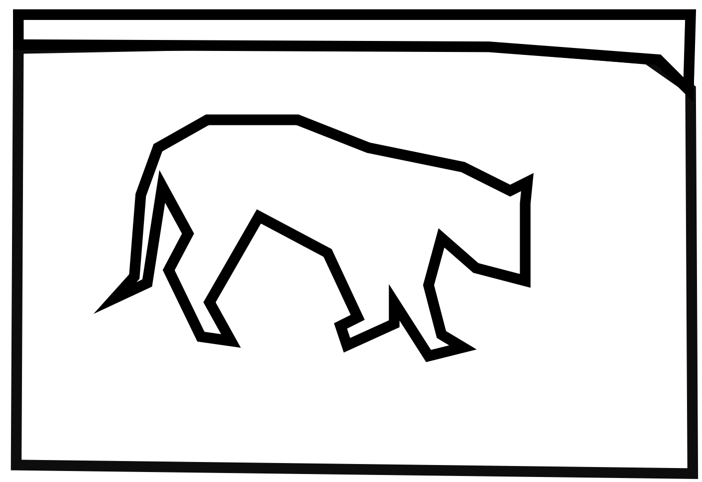
Why grouping?

- Pixels property of sensor, not world
- Reasoning at object level (might) make things easy:
    - objects at consistent depth
    - objects can be recognized
    - objects move as one

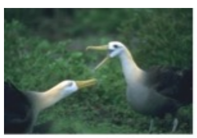
"I stand at the window and see a house, trees, sky. Theoretically I might say there were 327 brightnesses and nuances of colour. Do I have "327"? No. I have sky, house, and trees."
Max Wertheimer

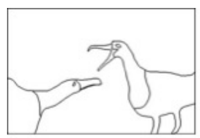
Regions ⟷ Boundaries



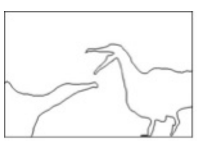


Is grouping well-defined?

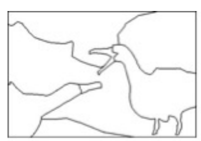




A



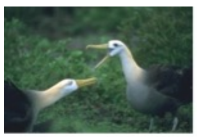
B

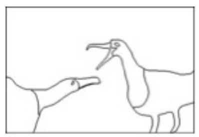


C

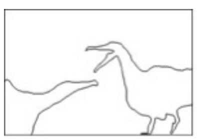
- Depends on purpose
  - Object parts
  - Background segmentation

Is grouping well-defined?

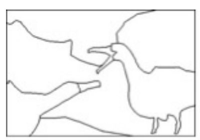




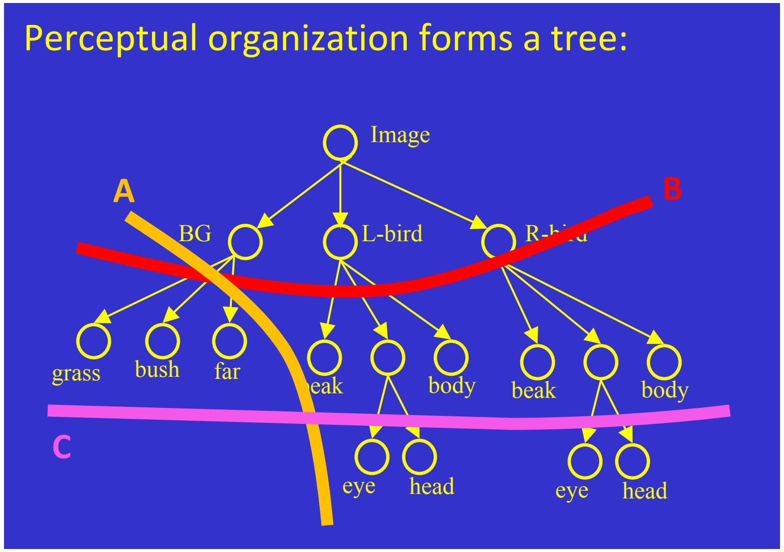
A



B



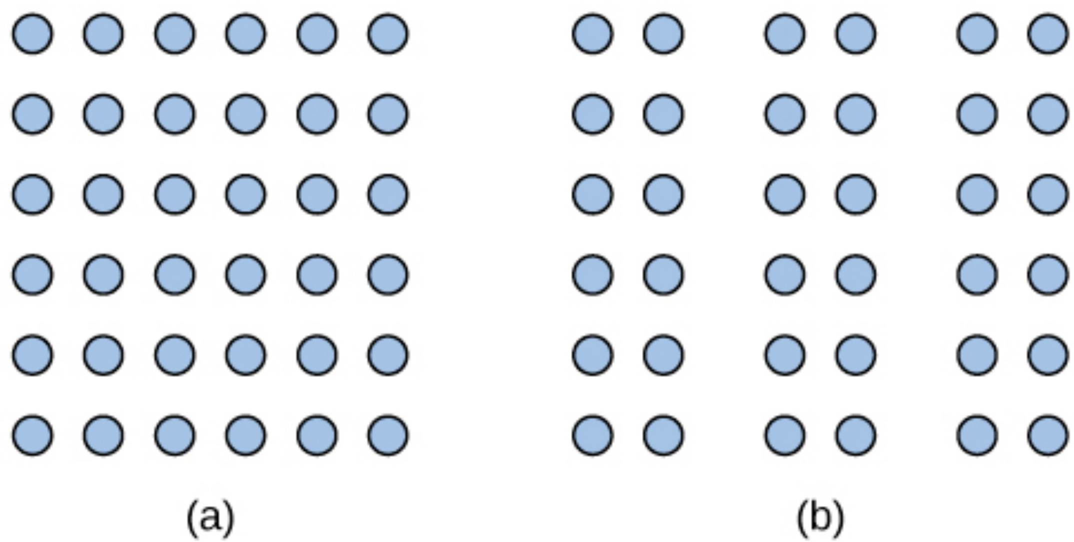
C



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

How do we group things?

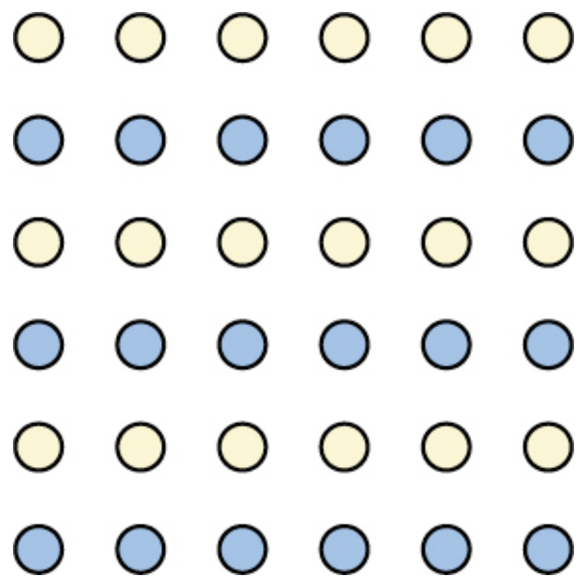
- *Gestalt* principles
- Principle of *proximity*



https://courses.lumenlearning.com/wsu-sandbox/chapter/gestalt-principles-of-perception/

How do we group things?

- Gestalt principles
- Principle of *similarity*



https://courses.lumenlearning.com/wsu-sandbox/chapter/gestalt-principles-of-perception/

How do we group things?

- Gestalt principles
- Principle of *continuity* and *closure*

https://courses.lumenlearning.com/wsu-sandbox/chapter/gestalt-principles-of-perception/

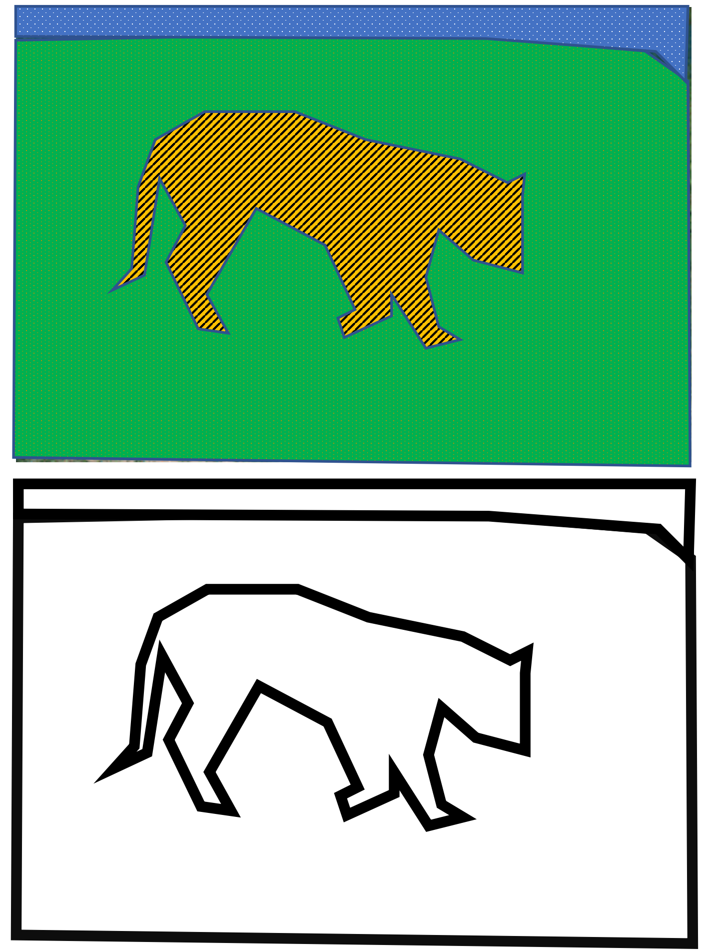
How do we group things?

- Gestalt principles
- Principle of *common fate*

Gestalt principles in the context of images

- Principle of proximity: nearby pixels are part of the same object
- Principle of similarity: similar pixels are part of the same object
    - Look for differences in color, intensity, or texture across the boundary
- Principle of closure and continuity: contours are likely to continue
- High-level knowledge?

Regions ⟷ Boundaries



Designing a good boundary detector

- Differences in color, intensity, or texture across the boundary
- Continuity and closure
- High-level knowledge

Criteria for a good boundary detector

- Criteria for a good boundary detector:
  - **Good detection:** Fire only on real edges, not anywhere else
  - **Good localization**
    - the edges detected must be as close as possible to the true edges
    - the detector must return one point only for each true edge point

Source: L. Fei-Fei

Canny edge detector

- The classic edge detector
- Baseline for all later work on grouping
- Theoretical model: step-edges corrupted by additive Gaussian noise

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

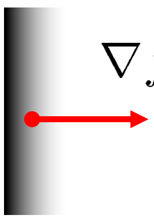
22,000 citations!

Source: L. Fei-Fei

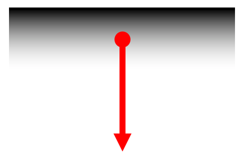
Image gradient

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

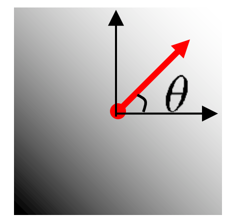
The gradient points in the direction of most rapid increase in intensity



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



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



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

The *edge strength* is given by the gradient magnitude:

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

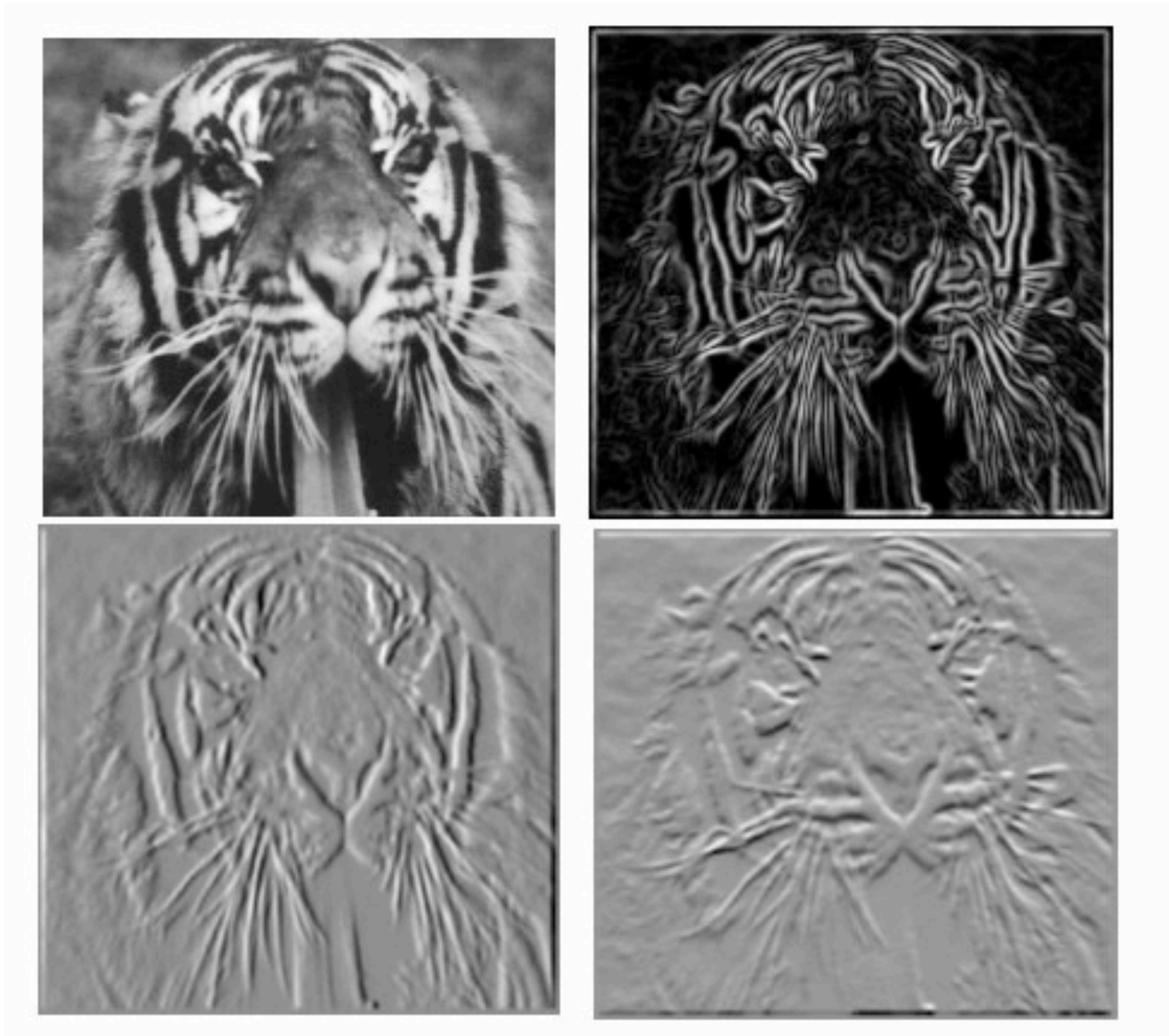
The gradient direction is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

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

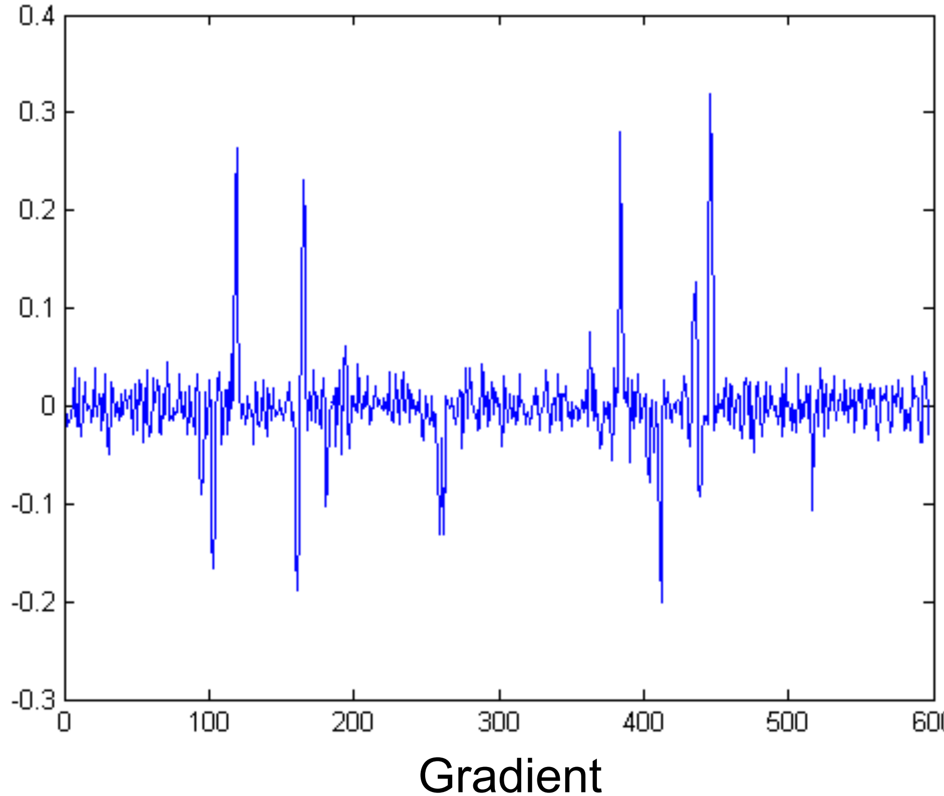
Source: Steve Seitz

Image gradient



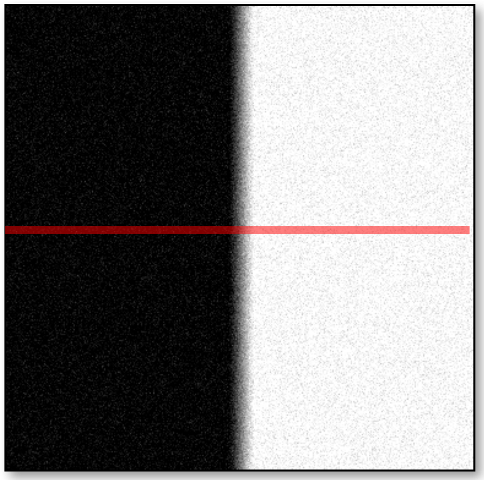
Source: L. Lazebnik

With a little Gaussian noise



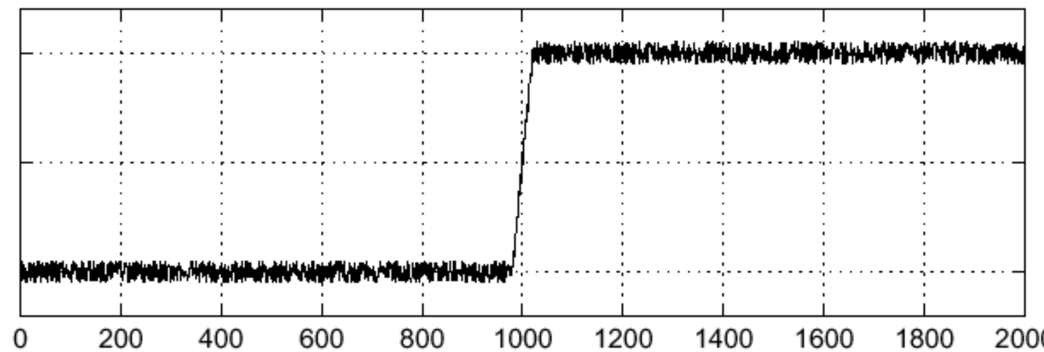
Source: D. Hoiem

Effects of noise

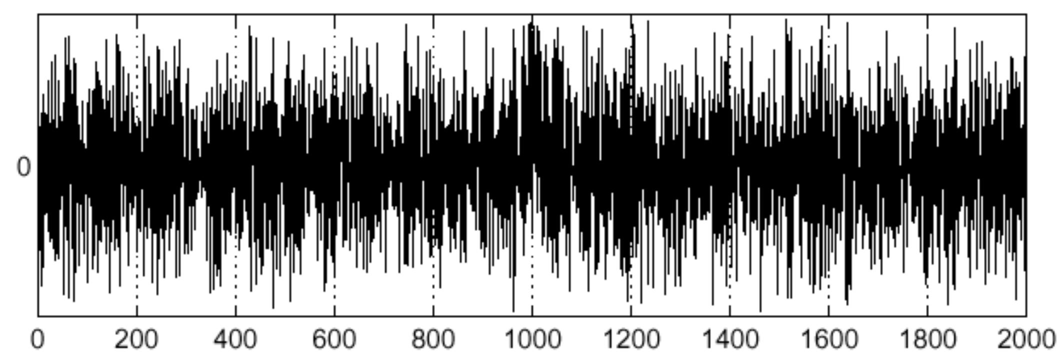


Noisy input image

$$f(x)$$



$$\frac{d}{dx}f(x)$$



Where is the edge?

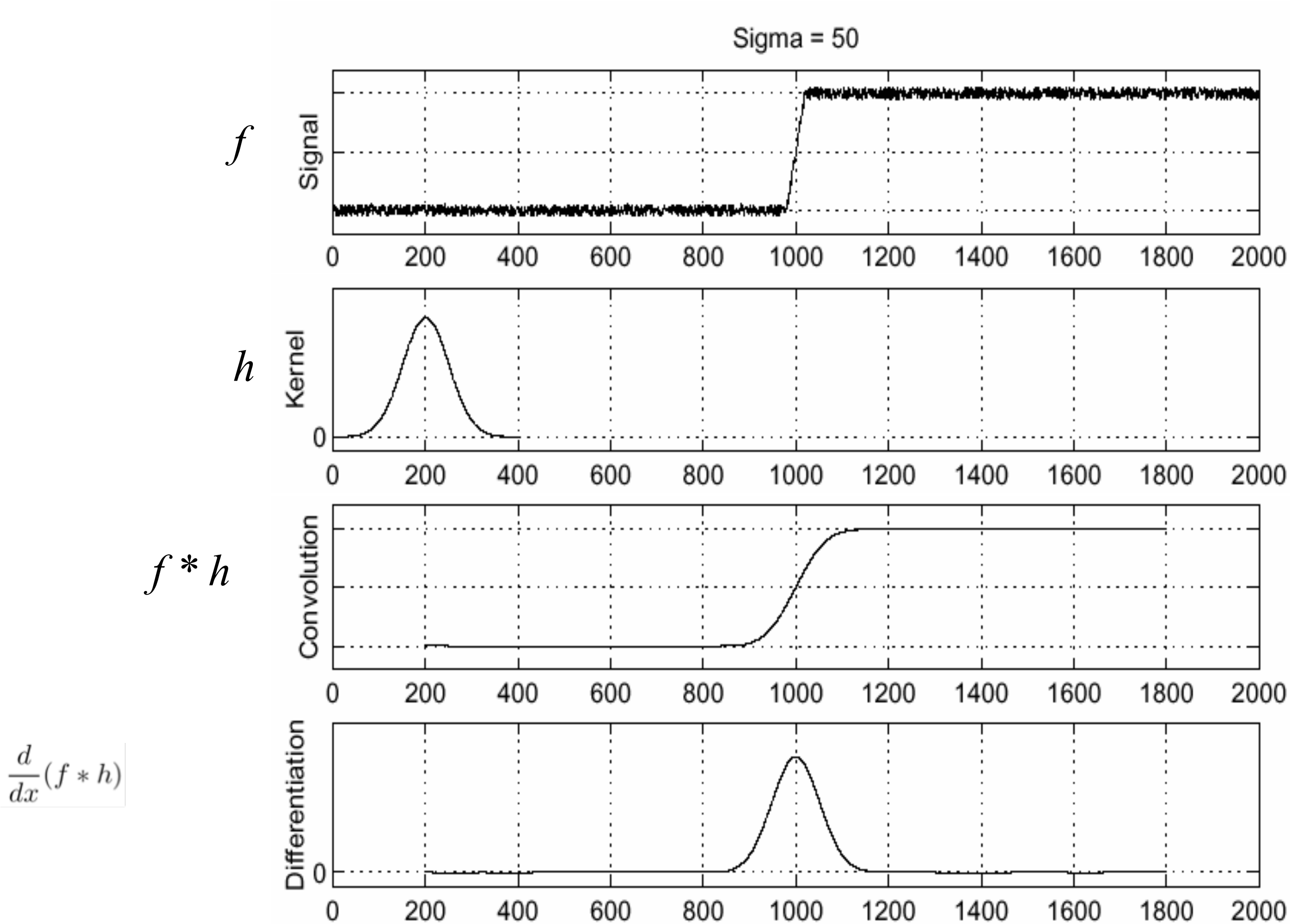
Source: S. Seitz

Effects of noise

- Noise is high frequency
- Differentiation accentuates noise

$$\frac{d \sin \omega x}{dx} = \omega \cos \omega x$$

Solution: smooth first



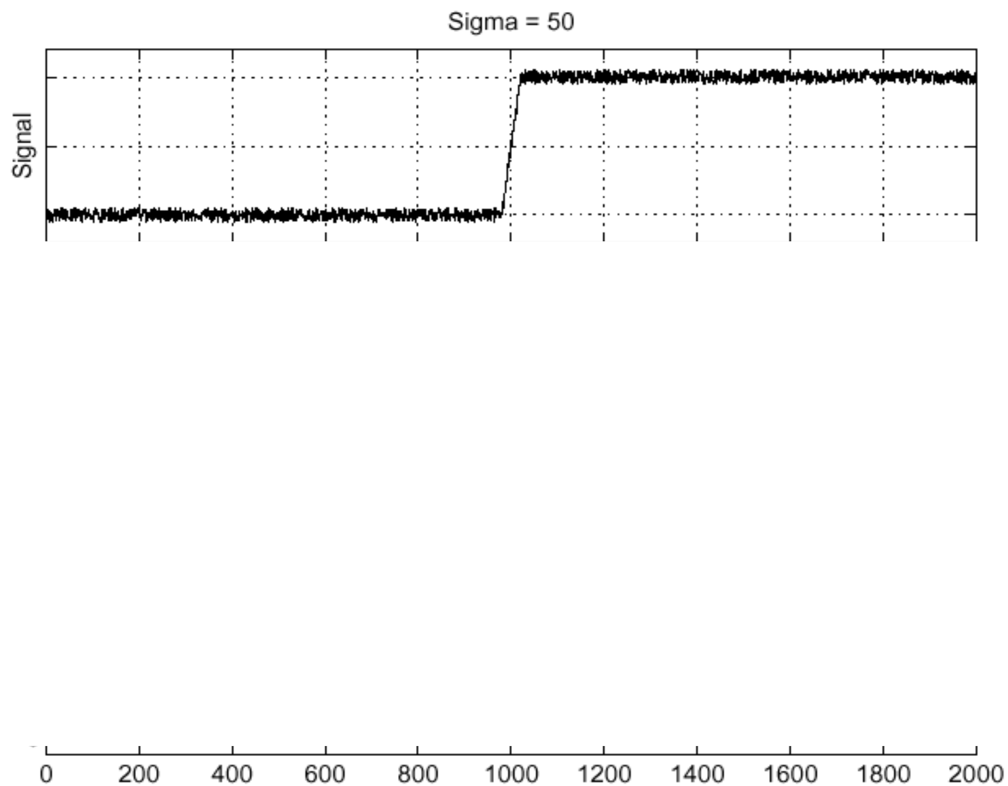
To find edges, look for peaks in $\frac{d}{dx}(f * h)$

Source: S. Seitz

Associative property of convolution

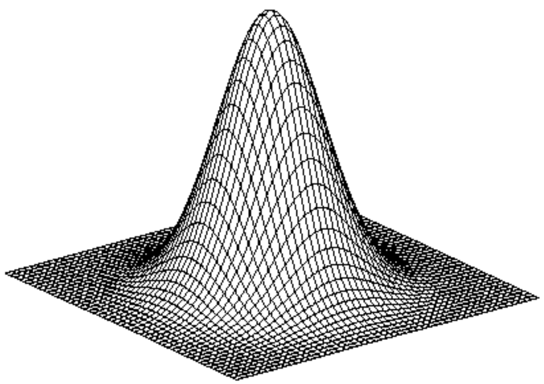
- Differentiation is a convolution
- Convolution is associative: $\frac{d}{dx}(f * h) = f * \frac{d}{dx}h$
- This saves us one operation:

f



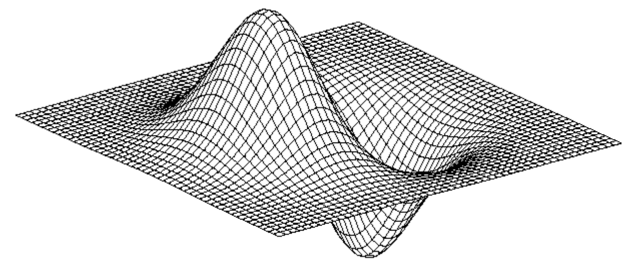
Source: S. Seitz

2D edge detection filters



Gaussian

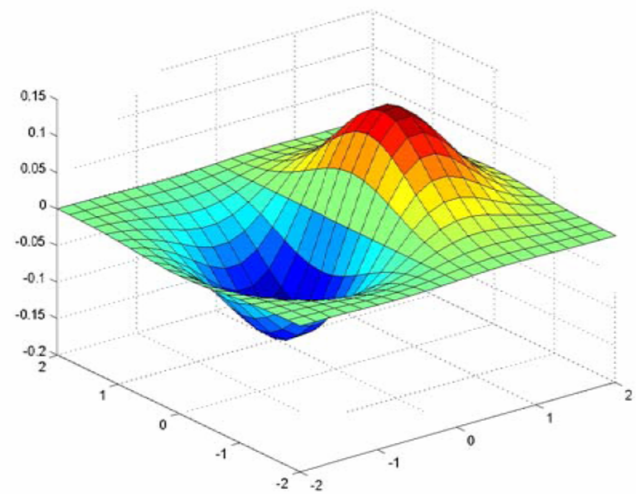
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



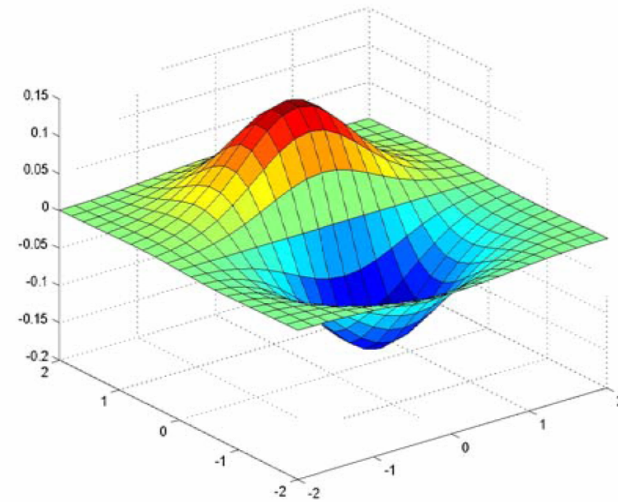
derivative of Gaussian (x)

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

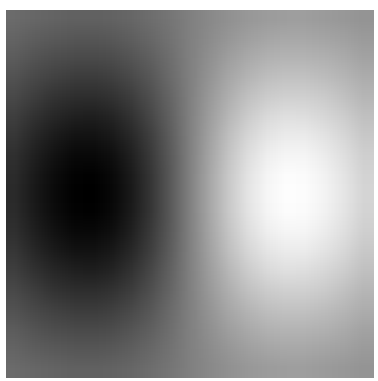
Derivative of Gaussian filter

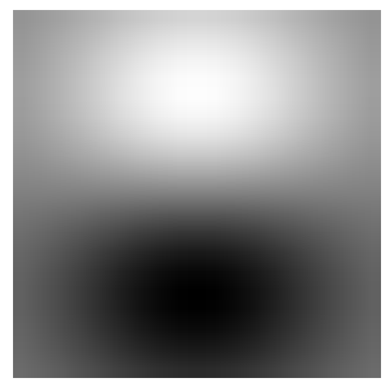


x-direction

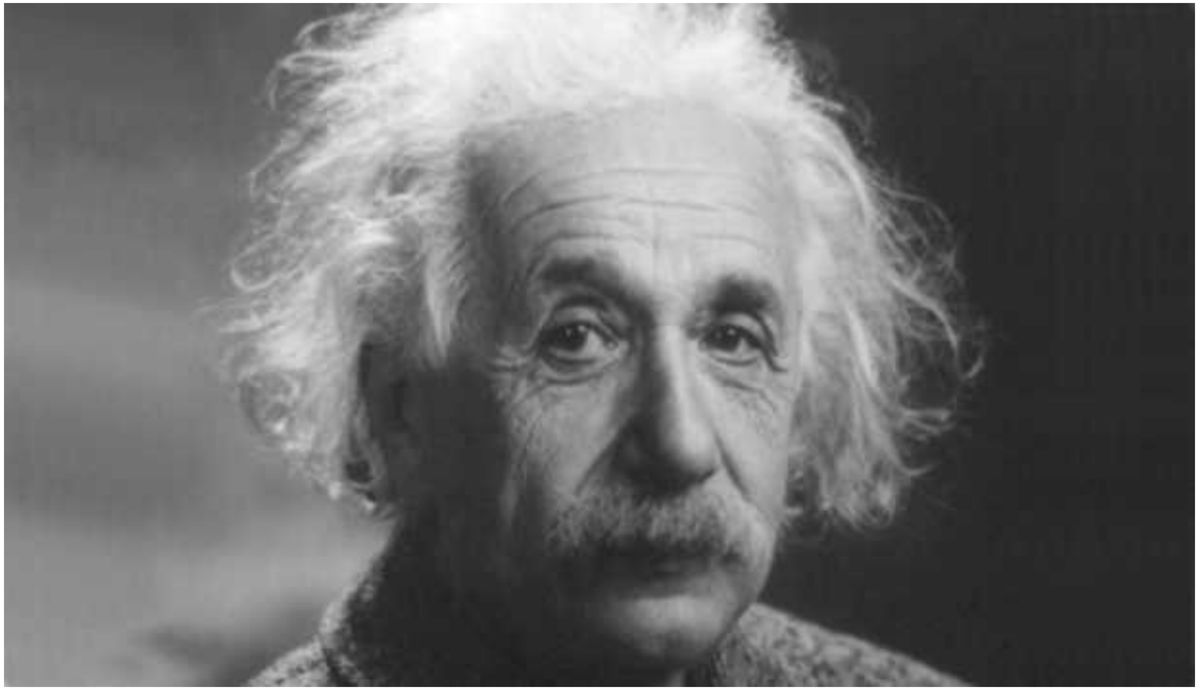


y-direction



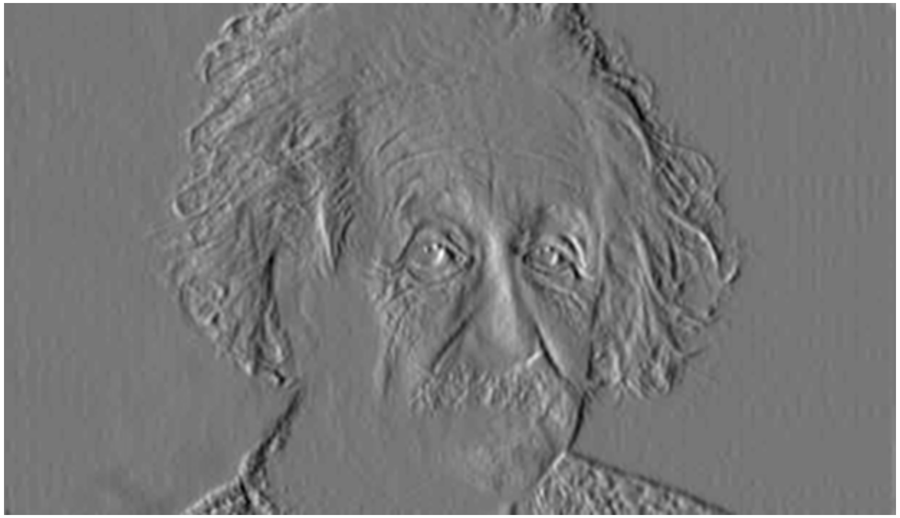


Example

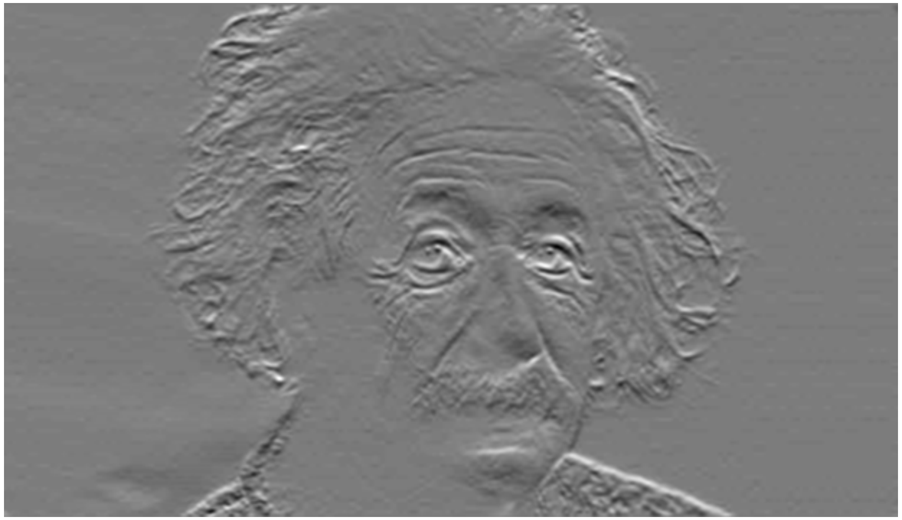


original image

Compute Gradients (DoG)



X-Derivative of Gaussian

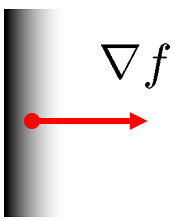


Y-Derivative of Gaussian

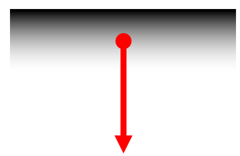
Image gradient

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

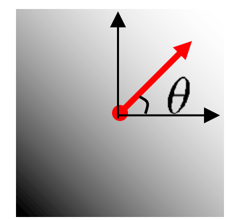
  The gradient points in the direction of most rapid increase in intensity



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



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



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

The *edge strength* is given by the gradient magnitude:

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

The gradient direction is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$$

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

Source: Steve Seitz

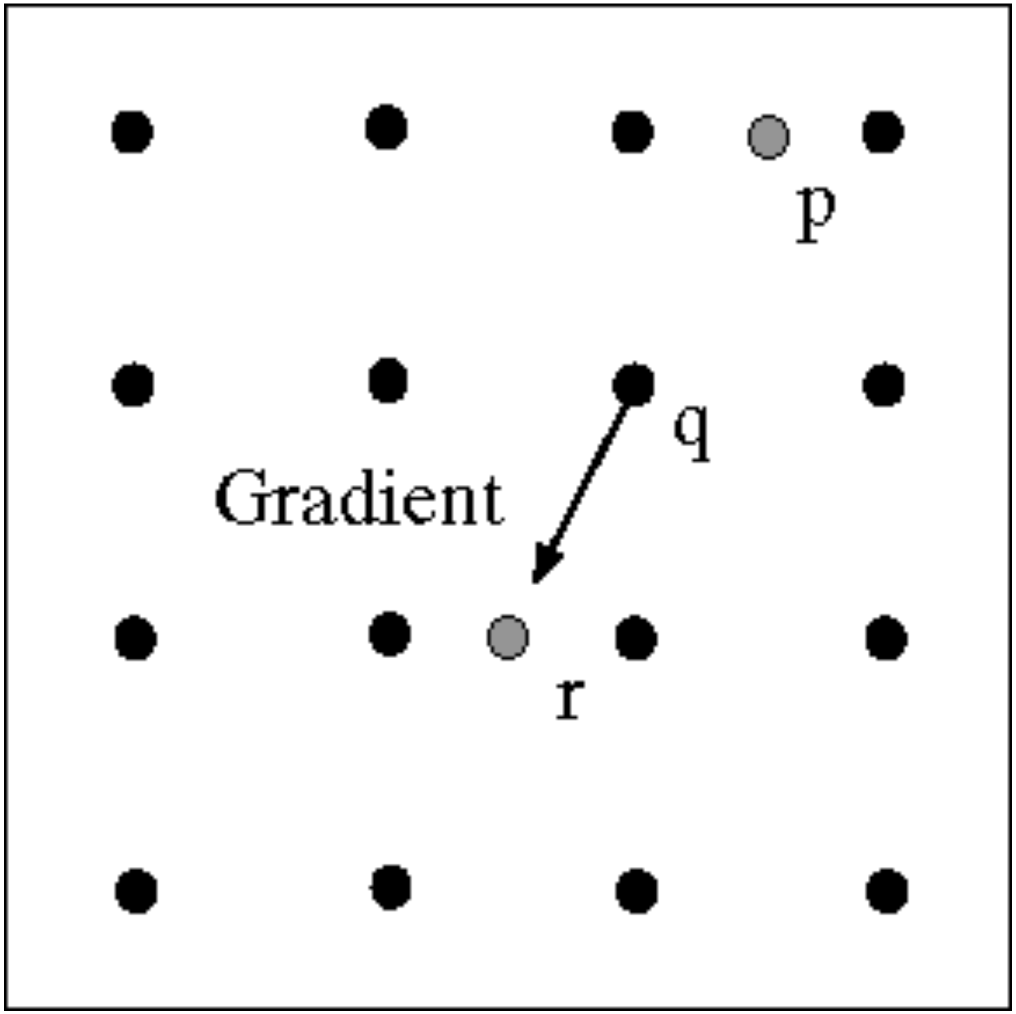
Gradient magnitude and orientation

- Orientation is undefined at pixels with 0 gradient

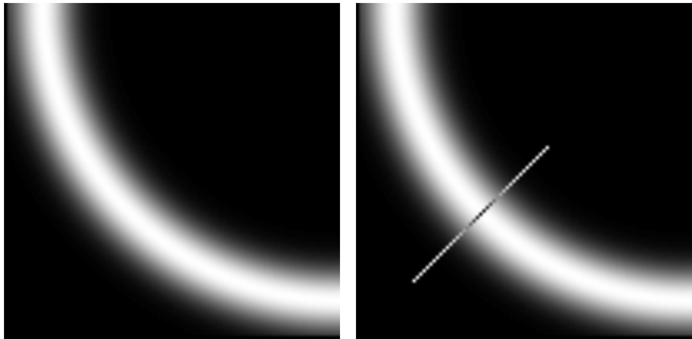
Magnitude

Orientation
theta = numpy.arctan2(gy, gx)

Non-maximum suppression for each orientation



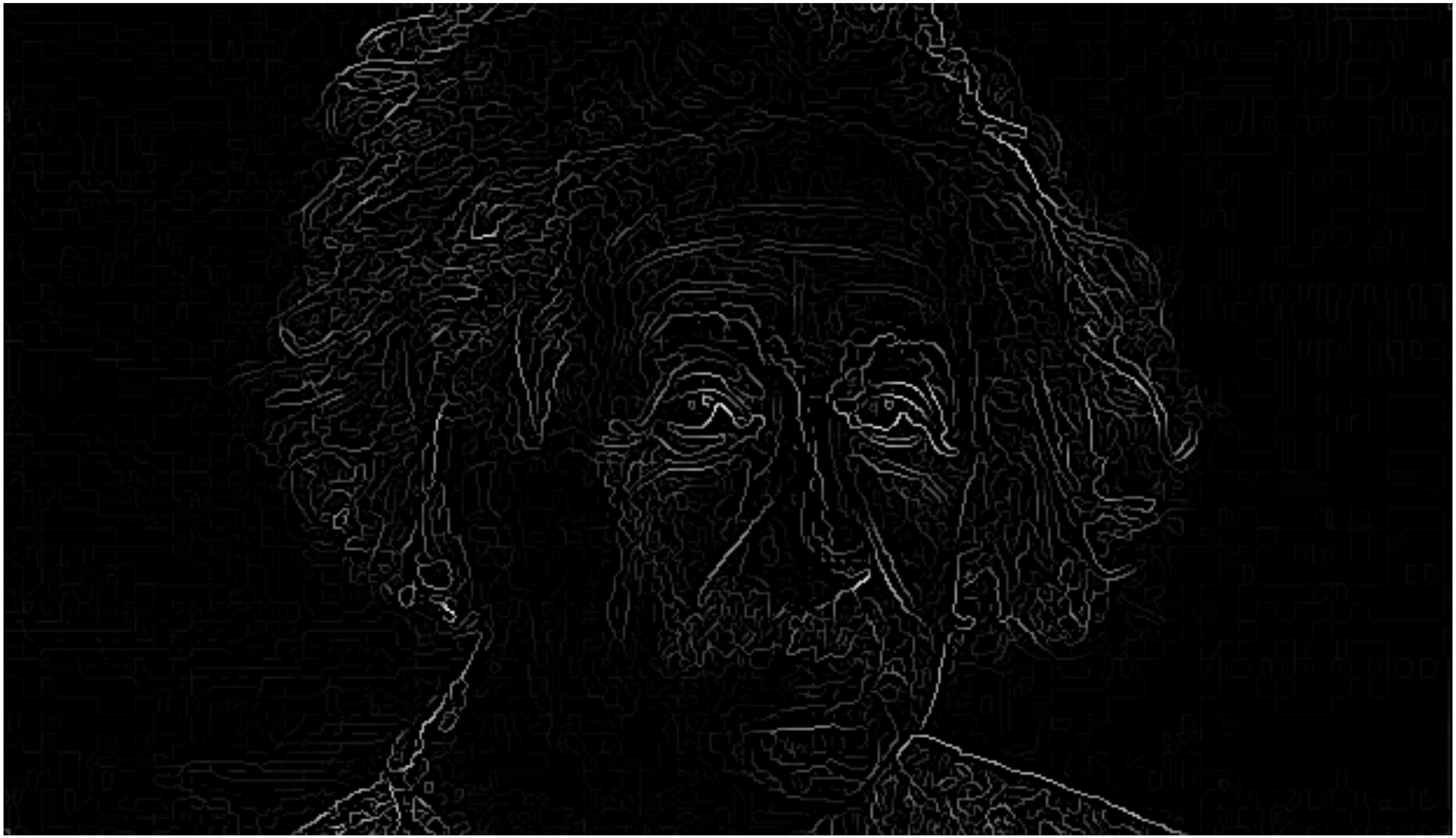
At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.



Source: D. Forsyth

Before Non-max Suppression

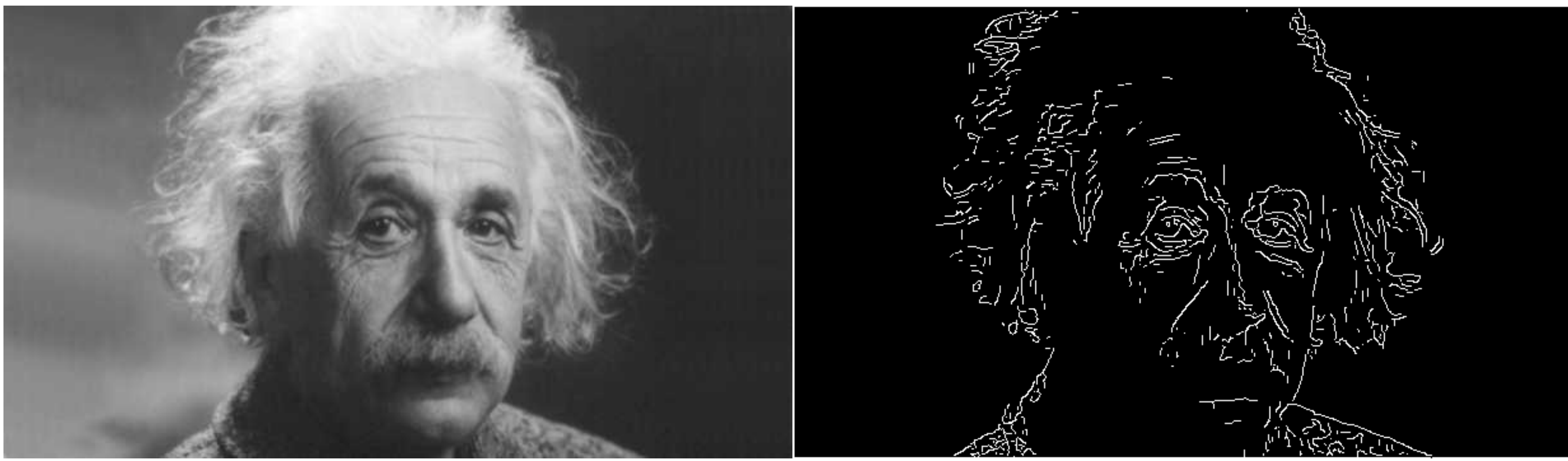
After Non-max Suppression



Hysteresis thresholding

- Threshold at low/high levels to get weak/strong edge pixels
- Do connected components, starting from strong edge pixels

Final Canny Edges

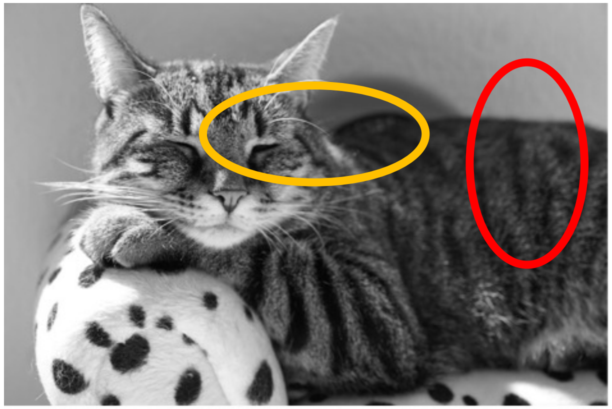


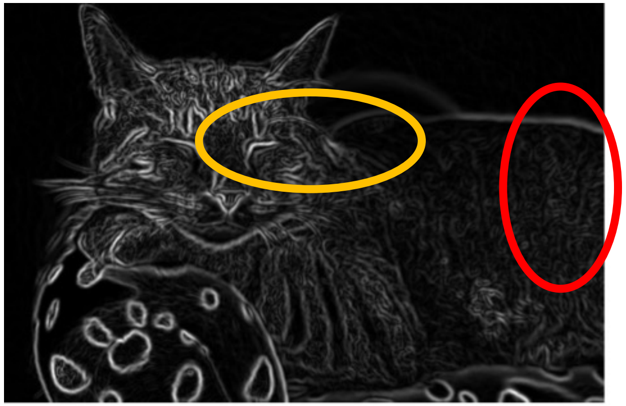
Canny edge detector

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

Source: D. Lowe, L. Fei-Fei

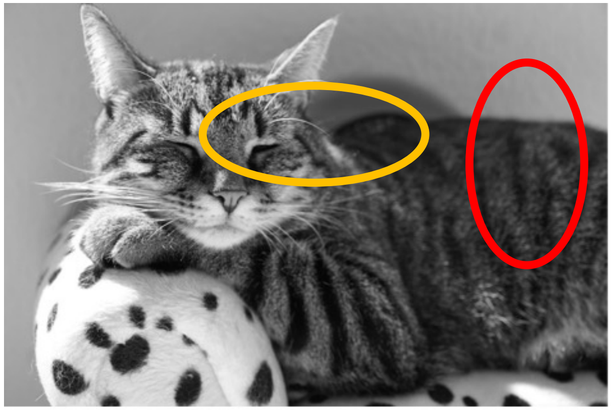
Does Canny always work?



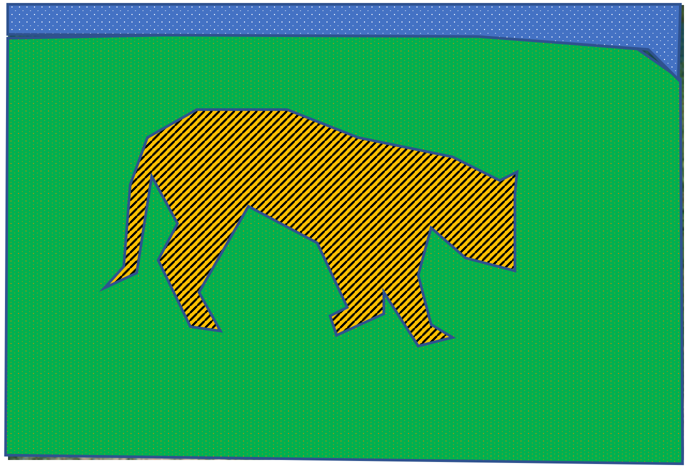


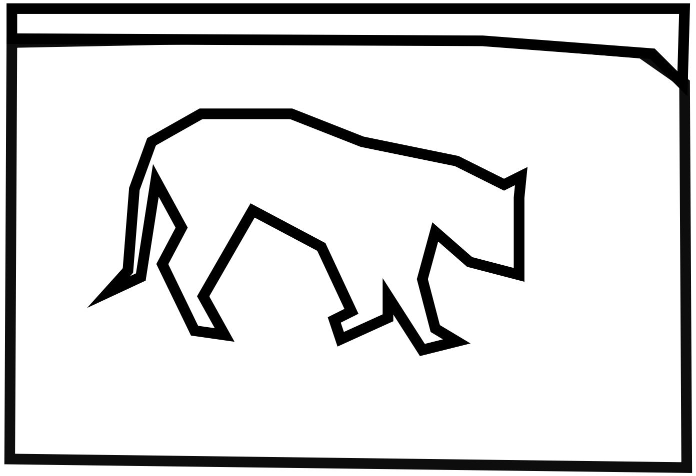
The challenges of edge detection

- Texture
- Low-contrast boundaries

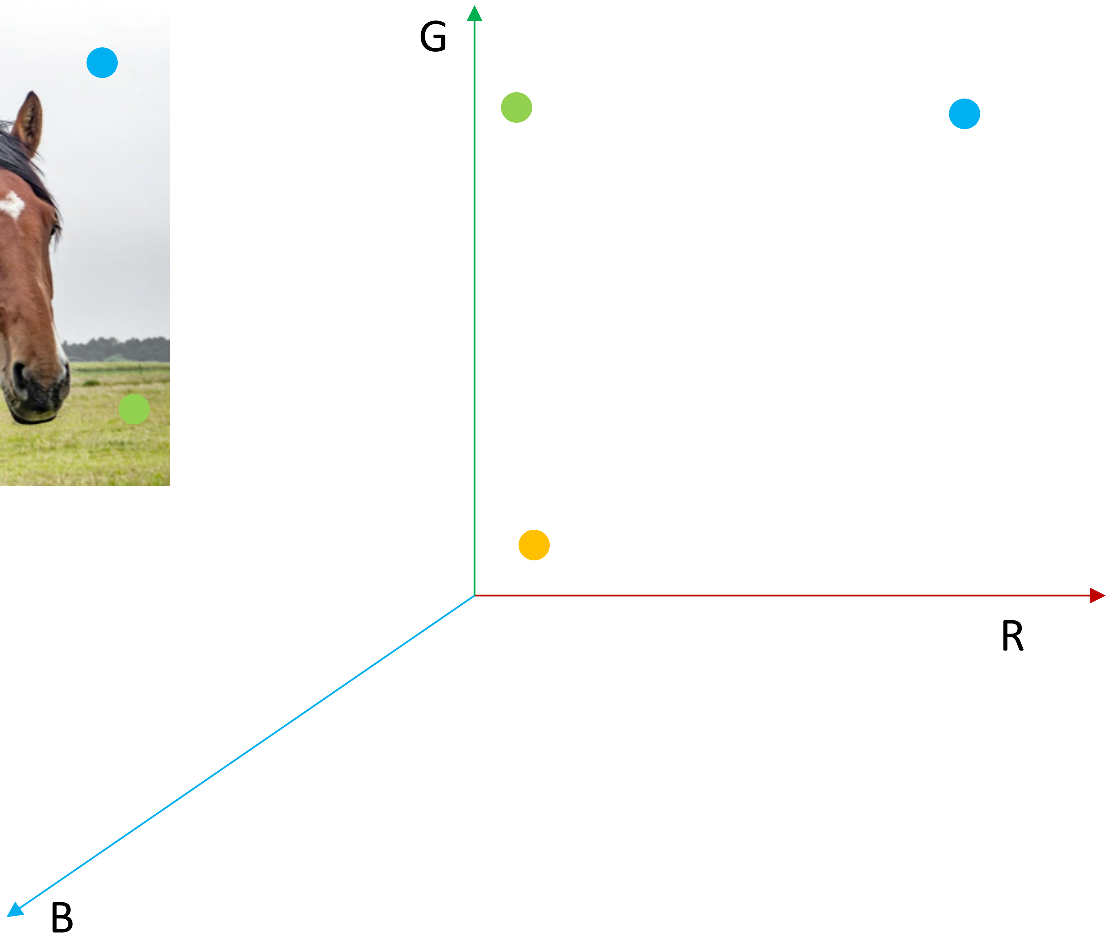


Regions ⟷ Boundaries



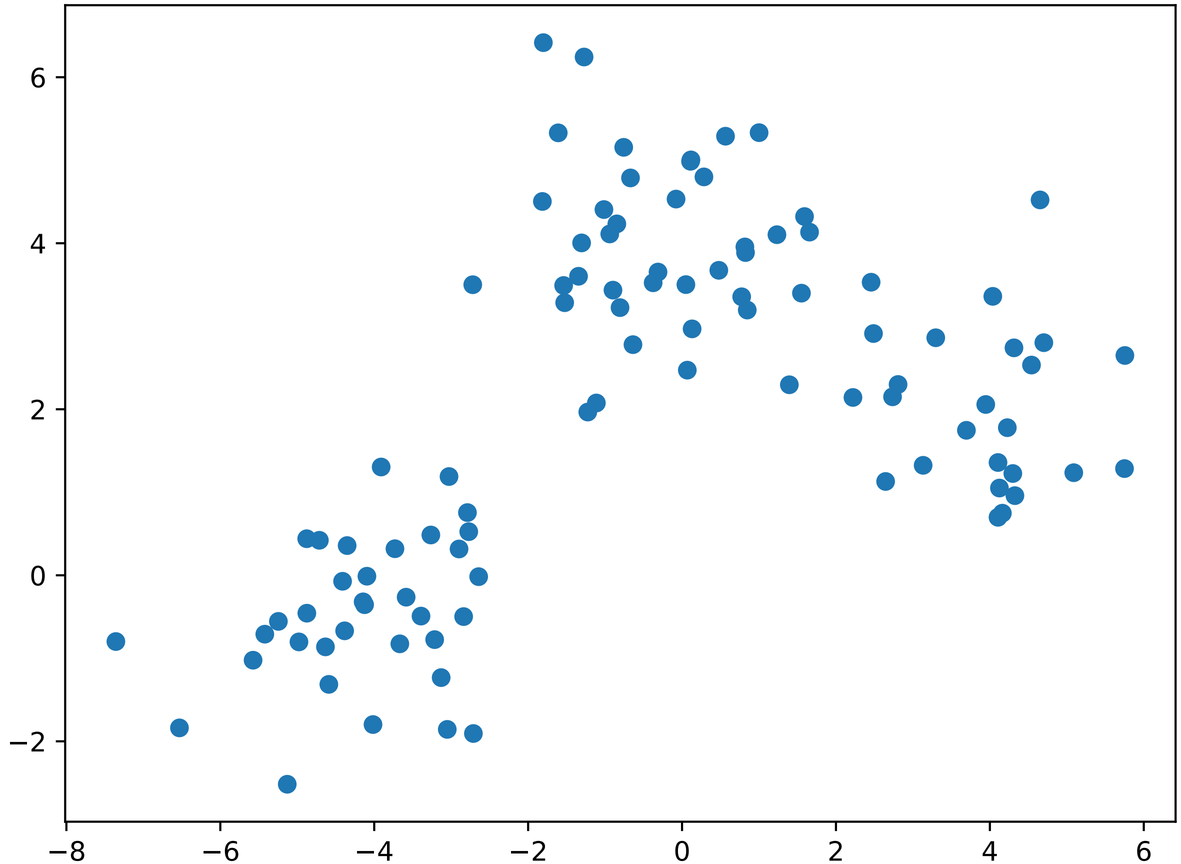


Grouping by clustering



Grouping by clustering

- Idea: embed pixels into high-dimensional space (e.g. 3-dimensions)
- Each pixel is a point
- Group together points



K-means

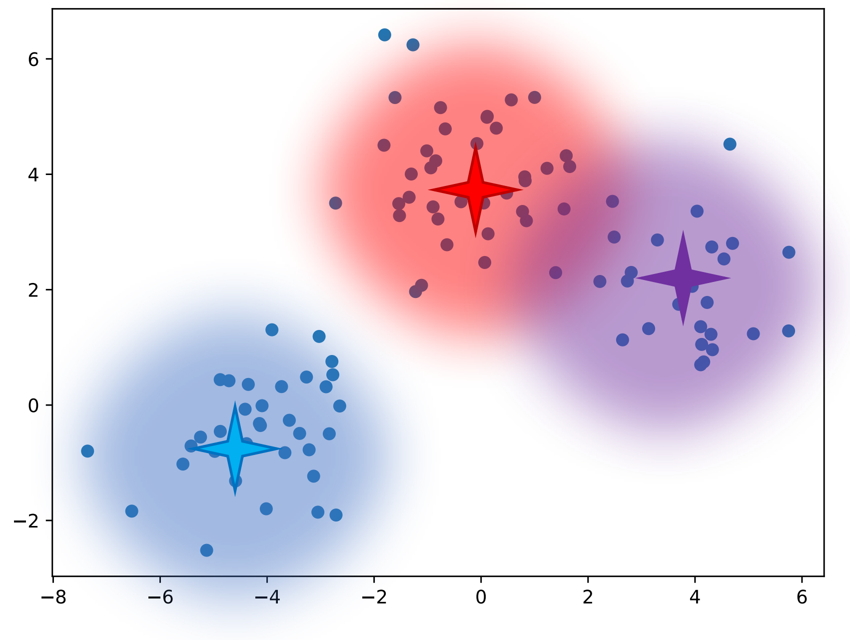
- Assumption: each group is a Gaussian with different means and same standard deviation

$$P(x_i|\mu_j) \propto e^{-\frac{1}{2\sigma^2}\|x_i-\mu_j\|^2}$$

- Suppose we know all μ_j. Which group should a point x_i belong to?
  - The j with highest $P(x_i|\mu_j)$
  - = The j with smallest $||x_i - \mu_j||^2$

K-means

- Assumption: each group = a Gaussian with different means and same standard deviation
- If means are known, then trivial assignment to groups. How?
- Assign data point to *nearest mean*!



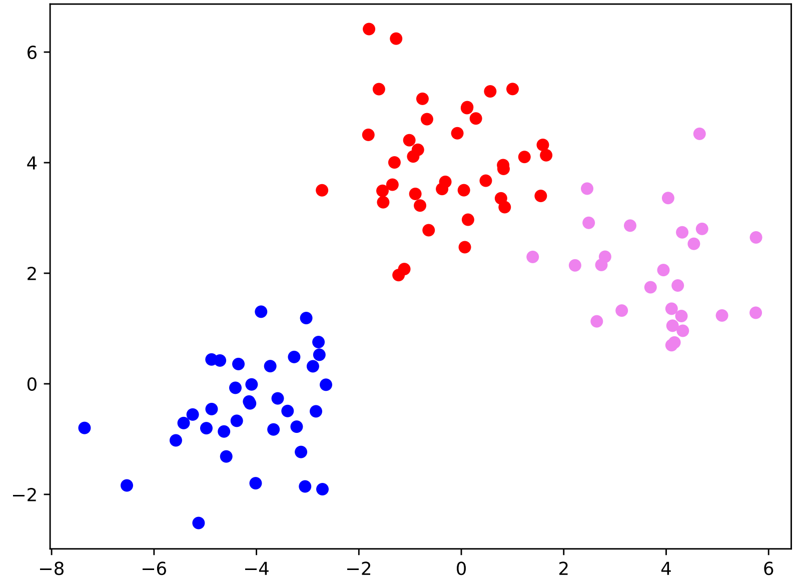
K-means

- Problem: means are not known
- What if we know a set of points from each cluster?
- $x_{k_1}, x_{k_2}, \ldots, x_{k_n}$ belong to cluster k
- What should be μ_k?

$$\mu_k = \frac{(x_{k_1} + x_{k_2} + \ldots + x_{k_n})}{n}$$

K-means

- Problem: means are not known
- If assignment of points to clusters is known, then finding means is easy
- How? Compute the mean of every cluster!

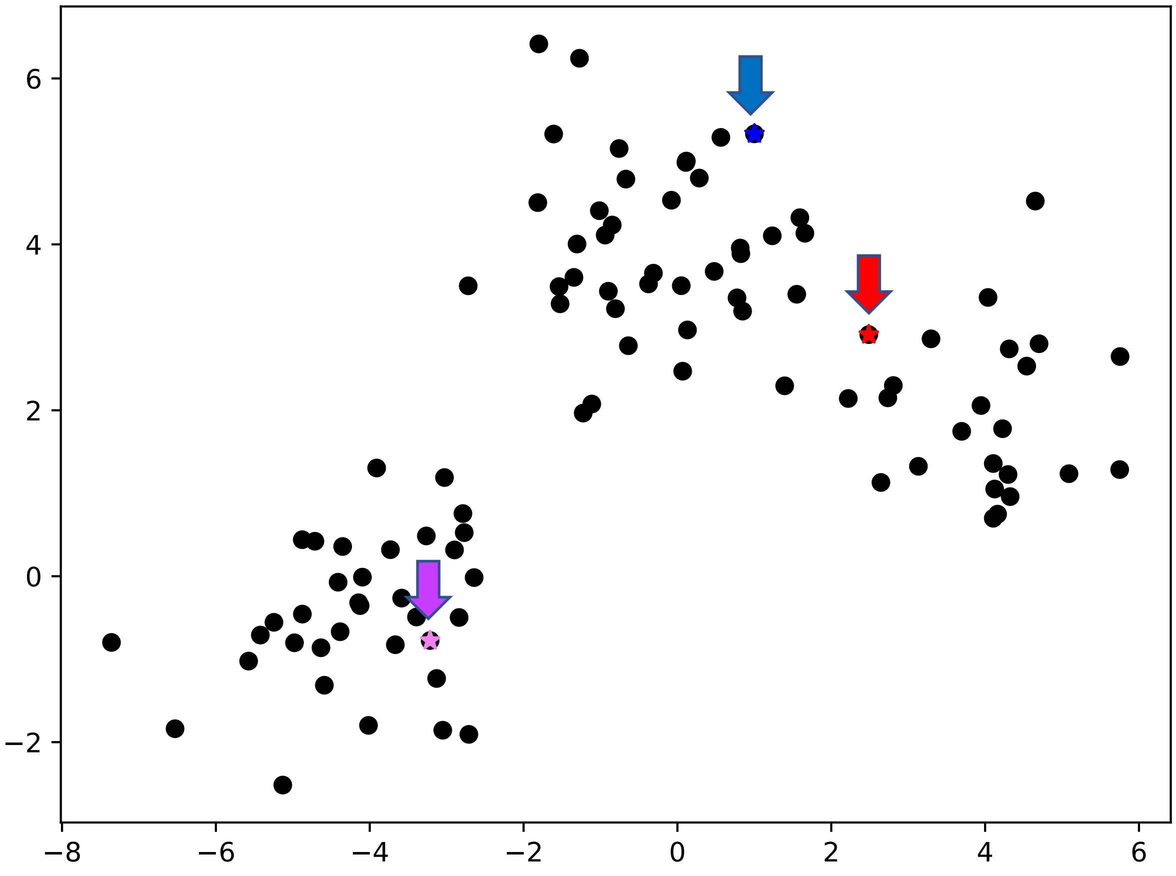


K-means

- Given means, can assign points to clusters
- Given assignments, can compute means
- Idea: iterate!

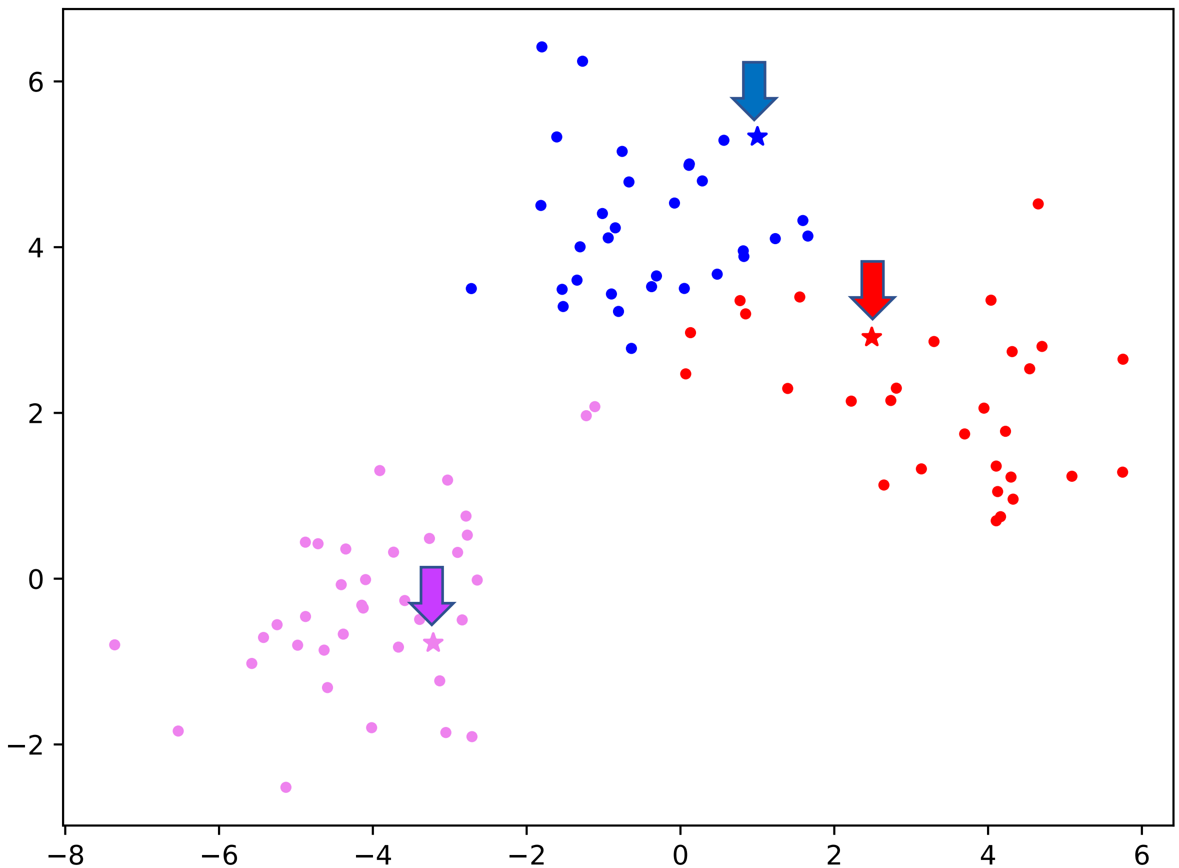
K-means

- Step-1 : randomly pick k centers



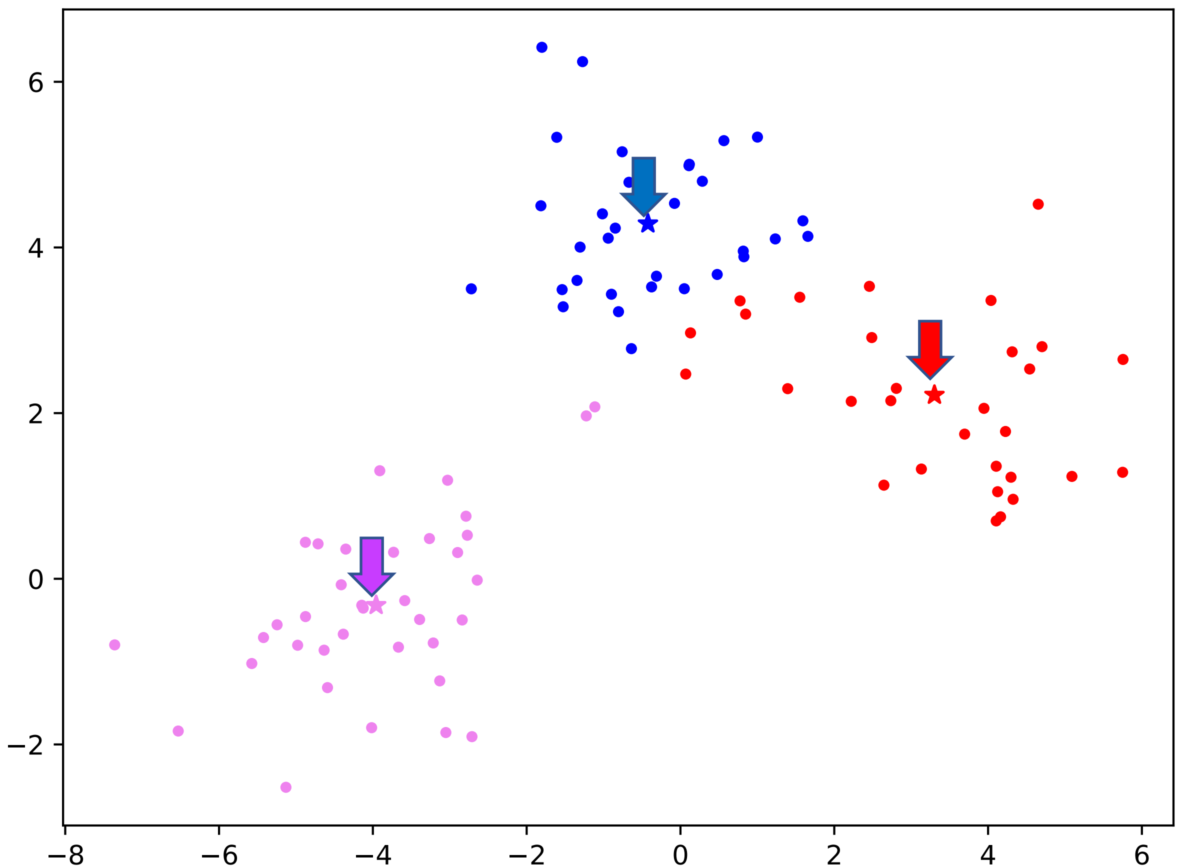
K-means

- Step 2: Assign each point to nearest center



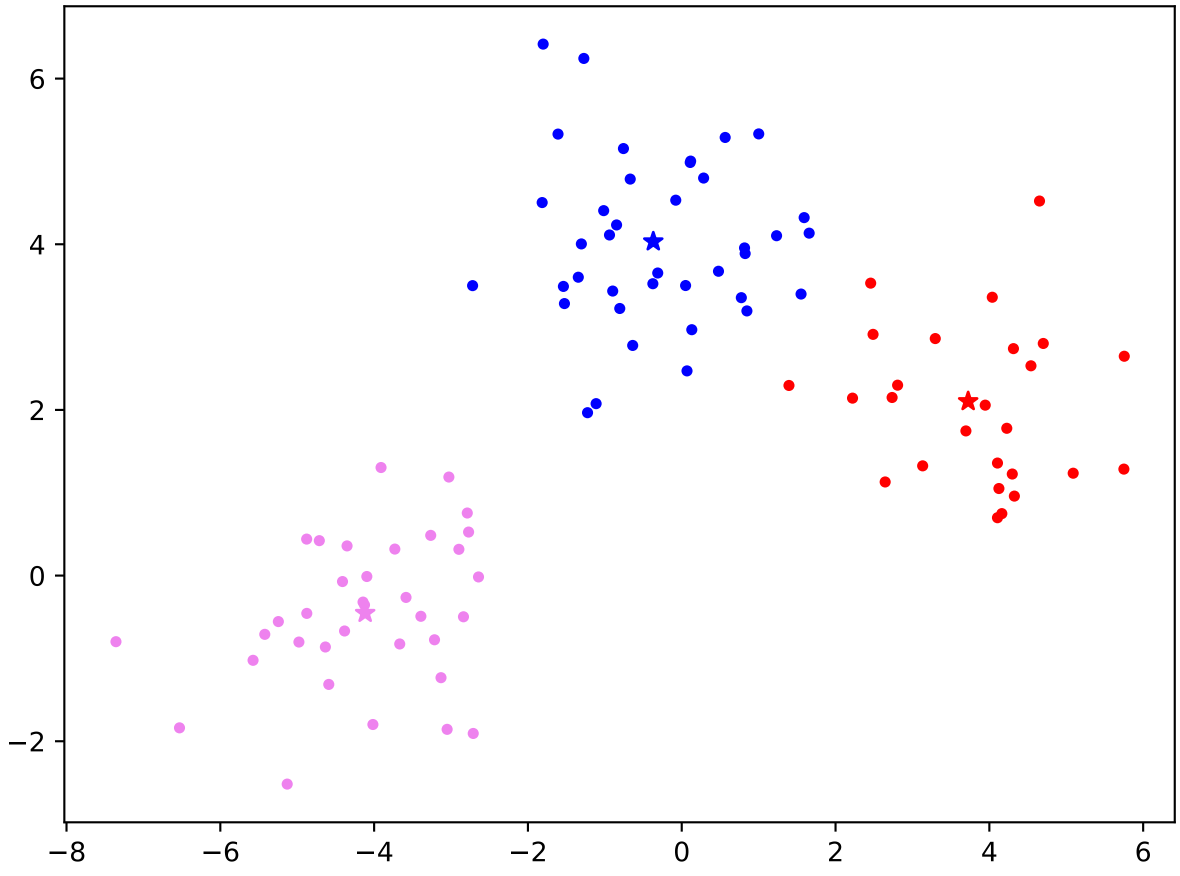
K-means

- Step 3: re-estimate centers

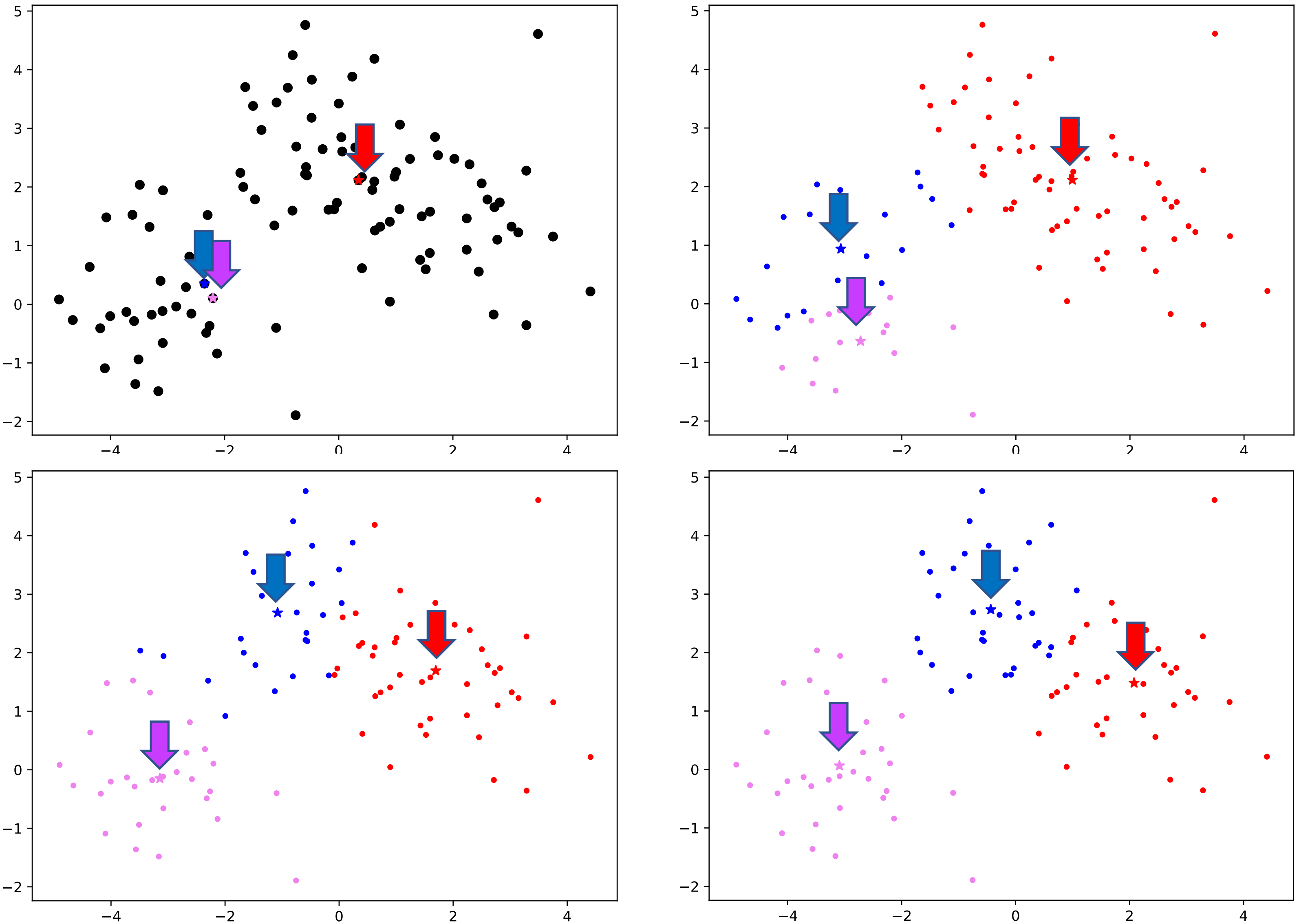


K-means

- Step 4: Repeat



K-means - another example



K-means

Input: set of data points, k

1. Randomly pick k points as means
2. For i in [0, maxiters]:
    1. Assign each point to nearest center
    2. Re-estimate each center as mean of points assigned to it

K-means - the math

Input: set of data points X, k

1. Randomly pick k points as means $\mu_i, i = 1, \ldots, k$
2. For iteration in [0, maxiters]:
   1. Assign each point to nearest center

   $$y_i = \arg\min_j \|x_i - \mu_j\|^2$$

   2. Re-estimate each center as mean of points assigned to it

   $$\mu_j = \frac{\sum_{i:y_i=j} x_i}{\sum_{i:y_i=j} 1}$$

K-means - the math

- An objective function that must be minimized:

$$\min_{\mu, y} \sum_i \|x_i - \mu_{y_i}\|^2$$

- Every iteration of k-means takes a downward step:
  - Fixes μ and sets y to minimize objective
  - Fixes y and sets μ to minimize objective

K-means on image pixels

K-means on image pixels

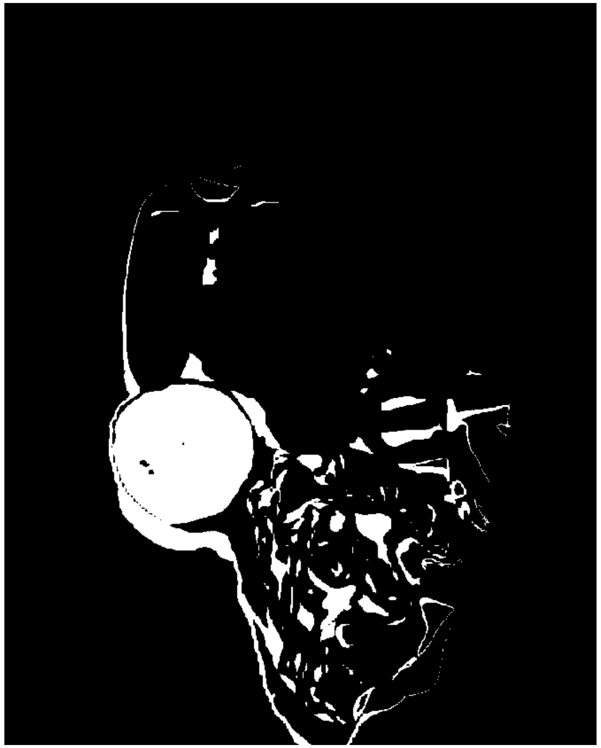
Picture courtesy David Forsyth

One of the clusters from k-means

K-means on image pixels

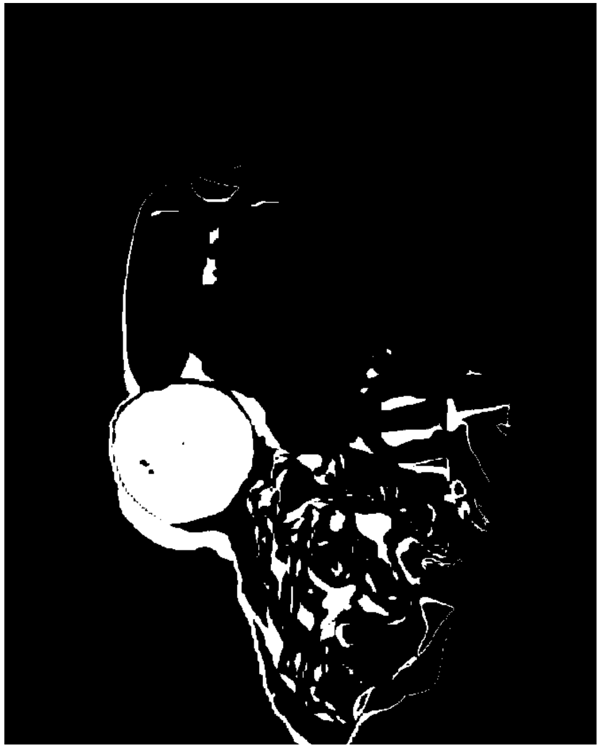
- What is wrong?
- Pixel position
  - Nearby pixels are likely to belong to the same object
  - Far-away pixels are likely to belong to different objects
- How do we incorporate pixel position?
  - Instead of representing each pixel as (r,g,b)
  - Represent each pixel as (r,g,b,x,y)

K-means on image pixels



The issues with k-means

- Captures pixel similarity but
  - Doesn’t capture continuity
  - Captures proximity only weakly
  - Can merge far away objects together
- Requires knowledge of k!



Oversegmentation and superpixels

- We don’t know k. What is a safe choice?
- Idea: Use large k
  - Can potentially break big objects, but will hopefully not merge unrelated objects
  - Later processing can decide which groups to merge
  - Called *superpixels*