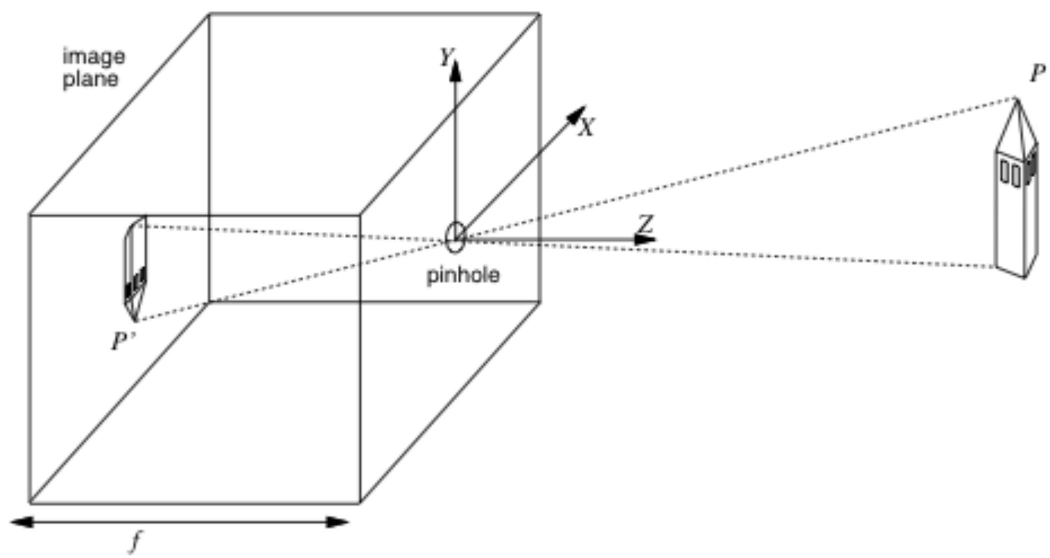
Unit 3: Application Areas

3a: Vision

Perception

- In general we (as humans) perceive a large number of inputs
  - visual, auditory, smell, touch, taste, ESP ;-)
- Somehow we make sense of it all and do the right thing
- Perception is the action of taking raw inputs and producing some beliefs about the world

How images are formed



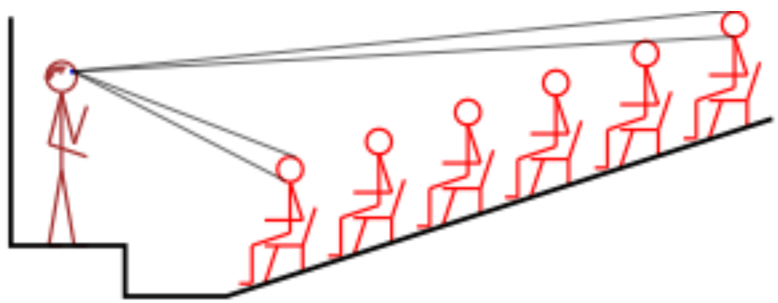
- Pinhole camera: point P in world shows up as image P'
  - P=(X,Y,Z); P'=(x,y,f)
  - x=-fX/Z, y=-fY/Z
  - The scale and distance is ambiguous

Vision as perception

- We can think of our percepts as arising from some view of the world W

  $P = g(W)$
- If g is "graphics", perhaps we can do vision as inverse graphics?

  $W=g^{-1}(P)$?
- Problem: Many worlds are possible

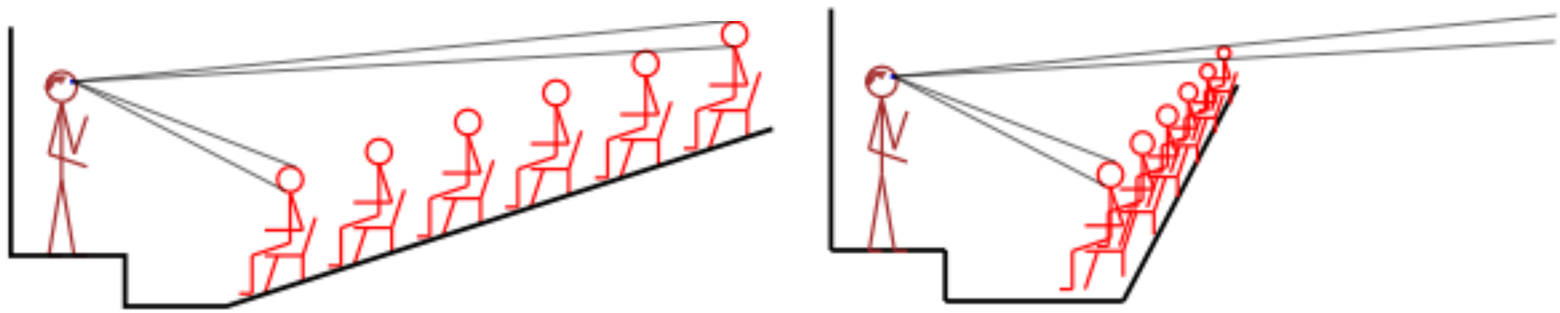


Vision as perception

- We can think of our percepts as arising from some view of the world W

  P = g(W)
- If g is "graphics", perhaps we can do vision as inverse graphics?

  $W=g^{-1}(P)$?
- Problem: Many worlds are possible



Vision as perception

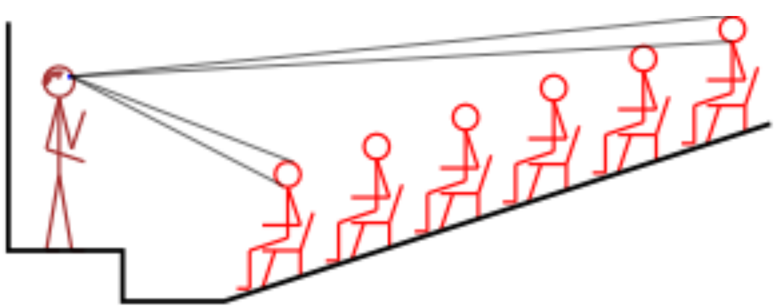
- We can think of our percepts as arising from some view of the world W

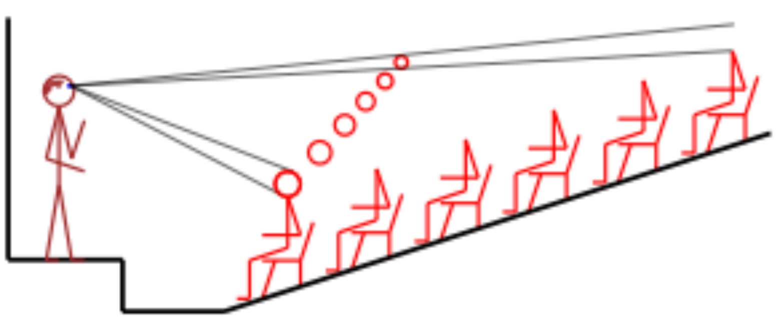
  P = g(W)

- If g is “graphics”, perhaps we can do vision as inverse graphics?

  $W=g^{-1}(P)$?

- Problem: Many worlds are possible





Vision as Bayesian Inference

- Can just treat this as an inference problem
  - $P(W|S) = \propto P(S|W)\,P(W)$
- The only problem then is trying to figure out priors over world states
  - This can be tricky

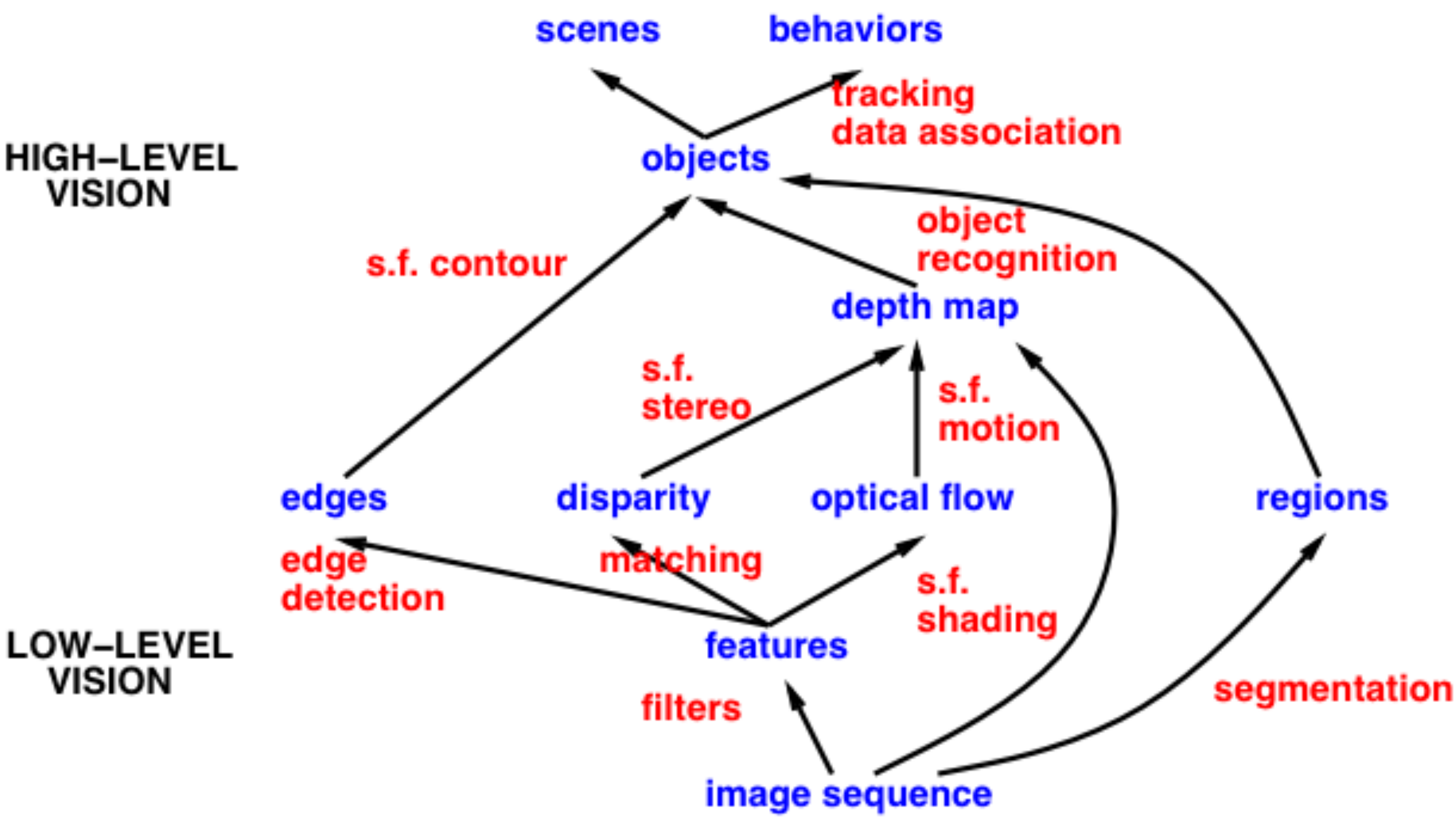
Another approach to vision

- What do you need vision for?
  - Moving around (navigation)
  - Moving things (manipulation)
  - Picking out objects (recognition/identification)
- Don’t bother to recover exact scene
- Just get the information needed to do the job
  - Feature extraction, machine learning

What you can get from vision

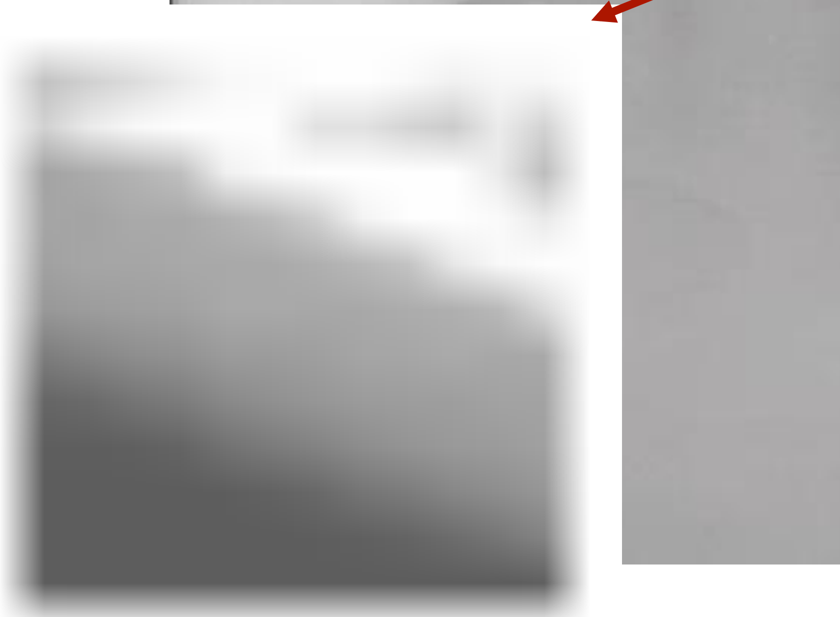
- Depth (how far away is something)
- Shape (is it flat? curved?)
- Boundaries (edges)
- Objects (putting together information)
- Tracking
- Action recognition

The visual hierarchy



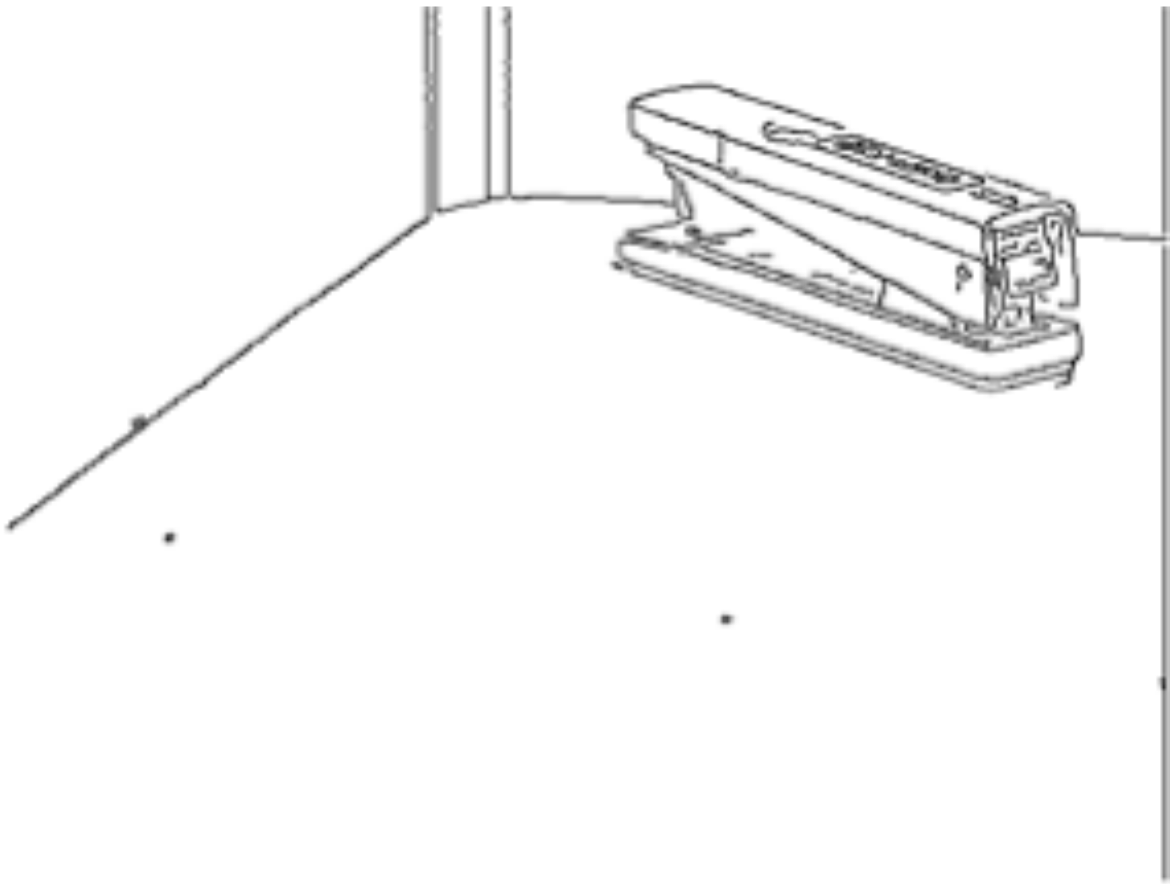
Images

Images

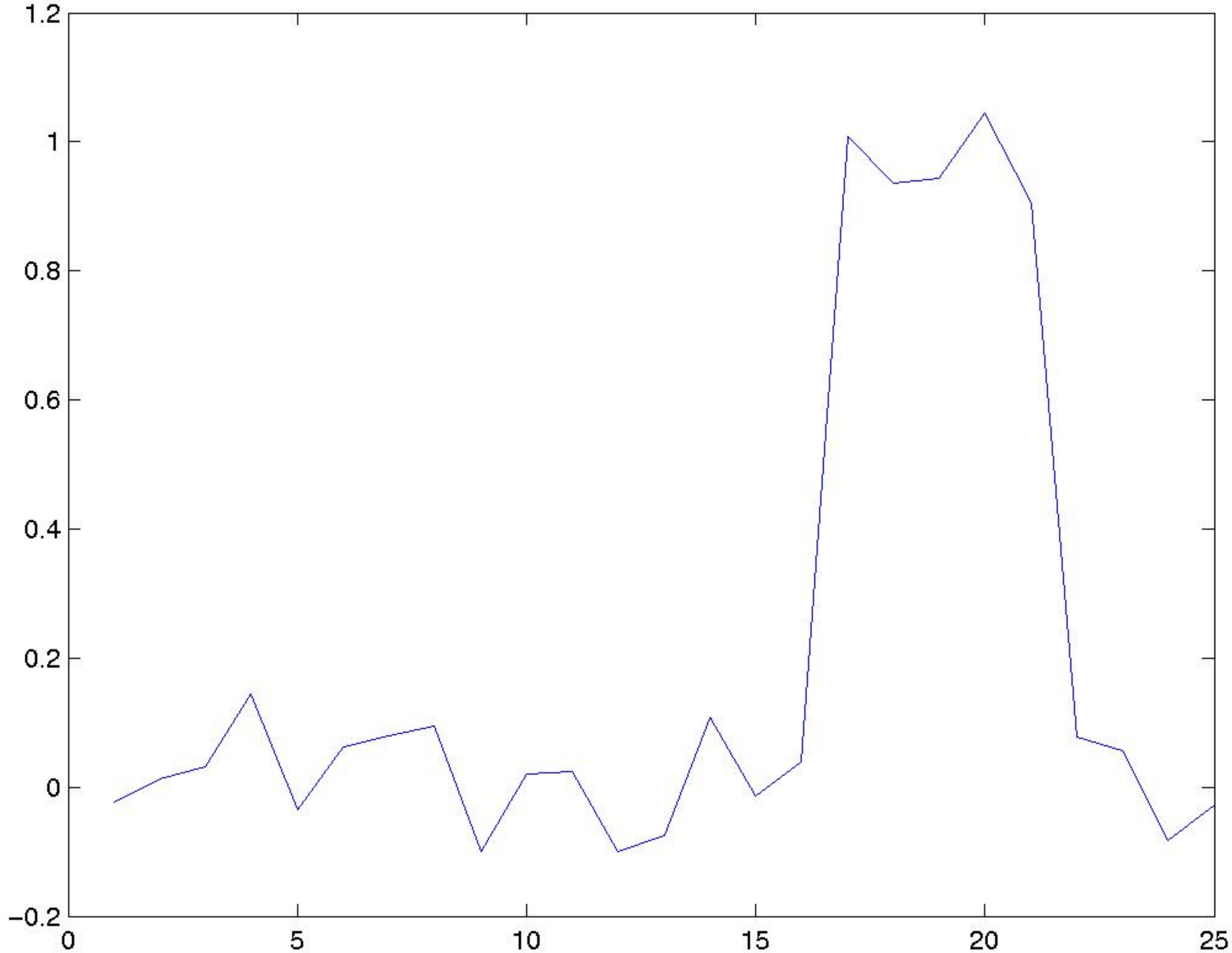


195	209	221	235	249	251	254	255	250	241	247	248
210	236	249	254	255	254	225	226	212	204	236	211
164	172	180	192	241	251	255	255	255	255	235	190
167	164	171	170	179	189	208	244	254	255	251	234
162	167	166	169	169	170	176	185	196	232	249	254
153	157	160	162	169	170	168	169	171	176	185	218
126	135	143	147	156	157	160	166	167	171	168	170
103	107	118	125	133	145	151	156	158	159	163	164
095	095	097	101	115	124	132	142	117	122	124	161
093	093	093	093	095	099	105	118	125	135	143	119
093	093	093	093	093	093	095	097	101	109	119	132
095	093	093	093	093	093	093	093	093	093	093	119

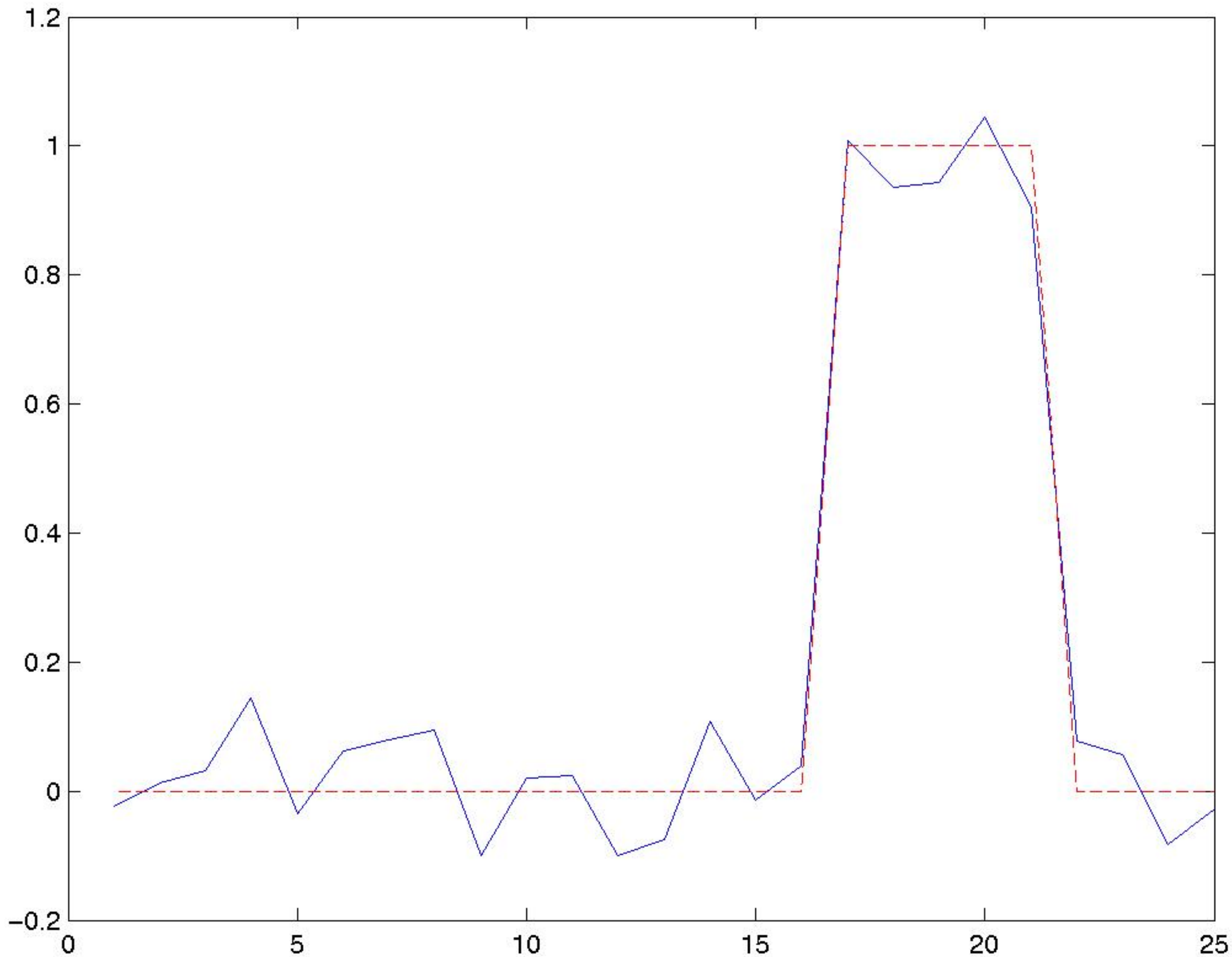
Edge detection: first step in object recognition



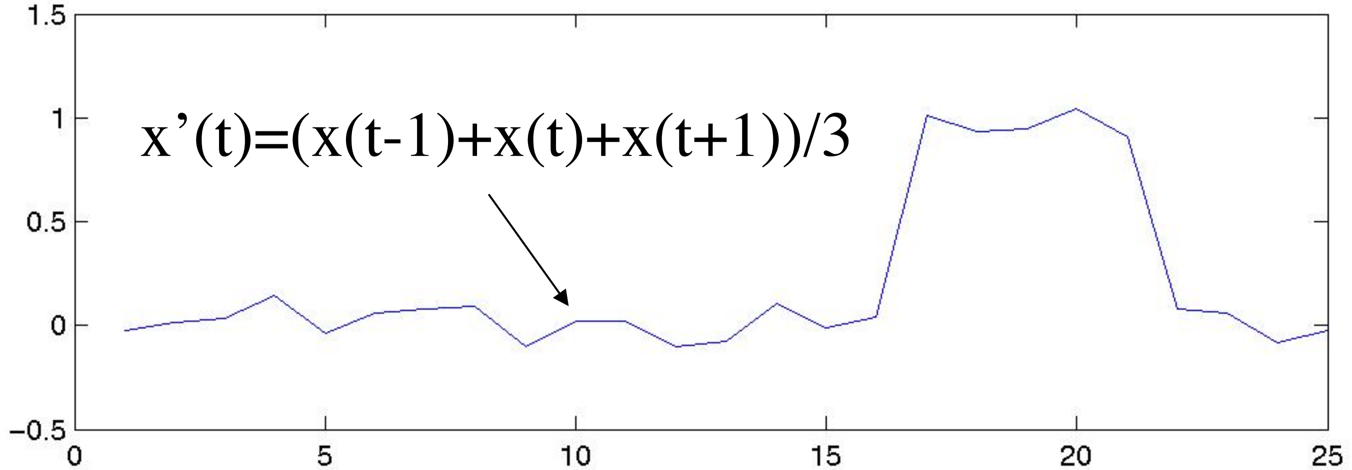
1-dimensional signal

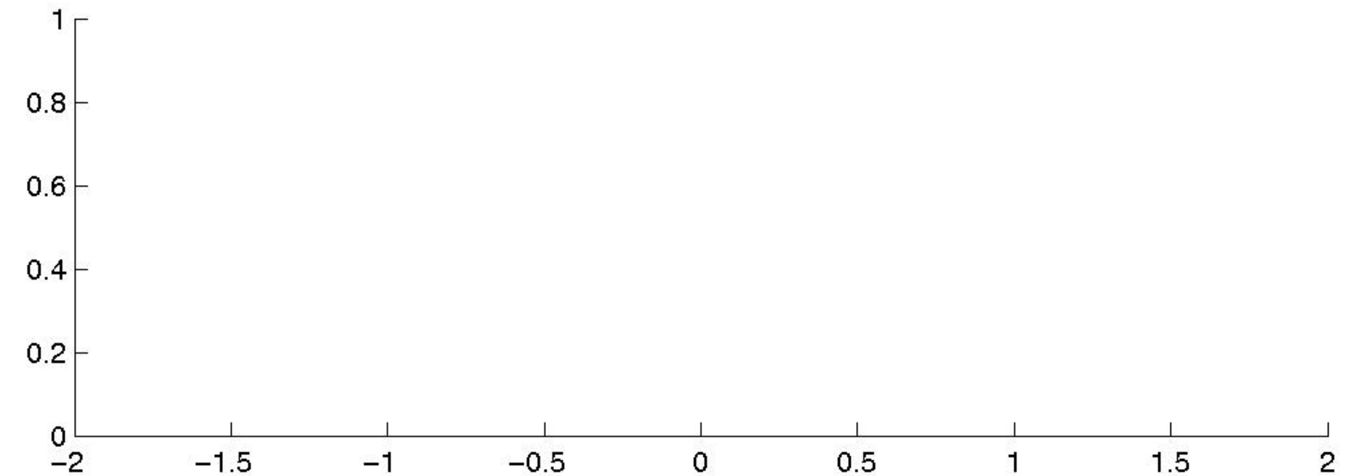


1-dimensional signal

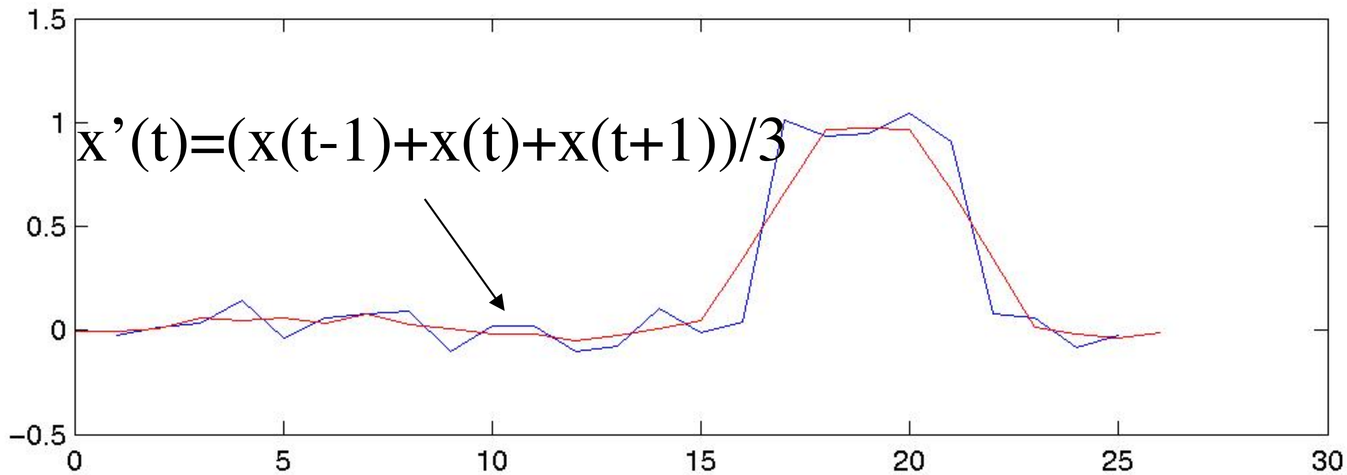


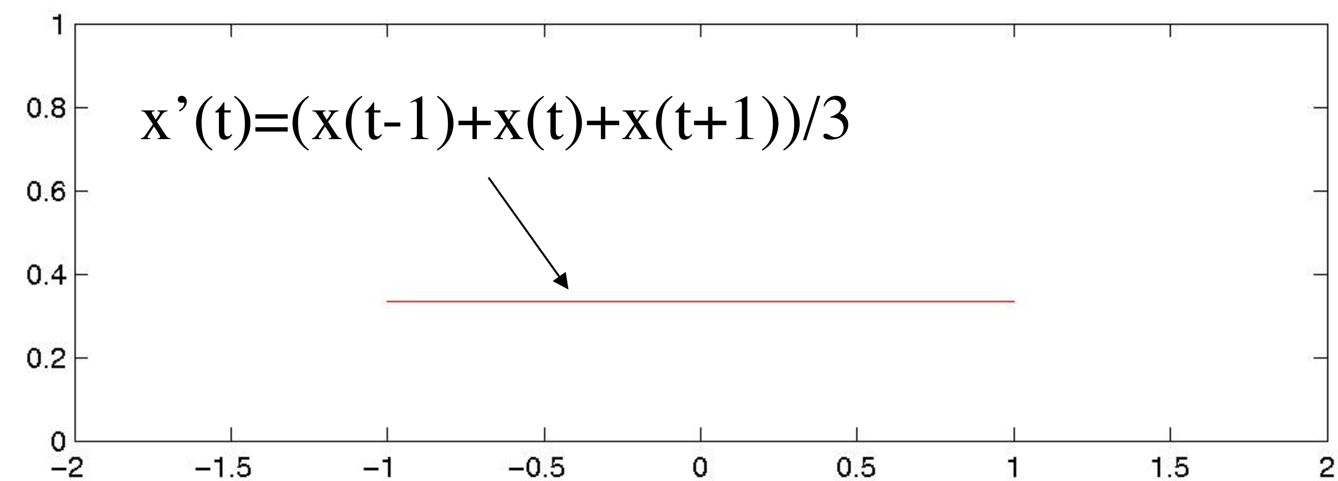
Smoothing it out: averaging



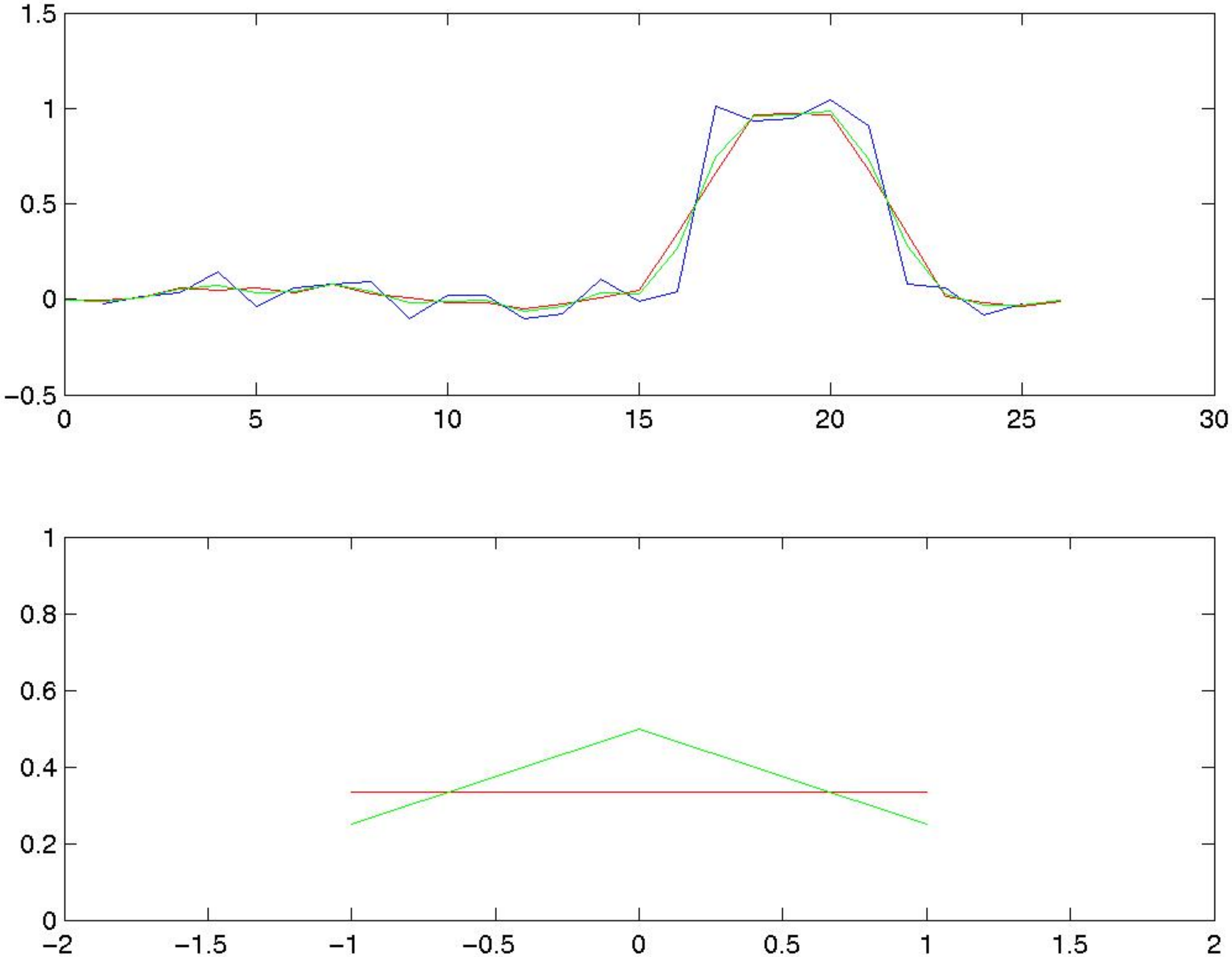


Smoothing it out: averaging

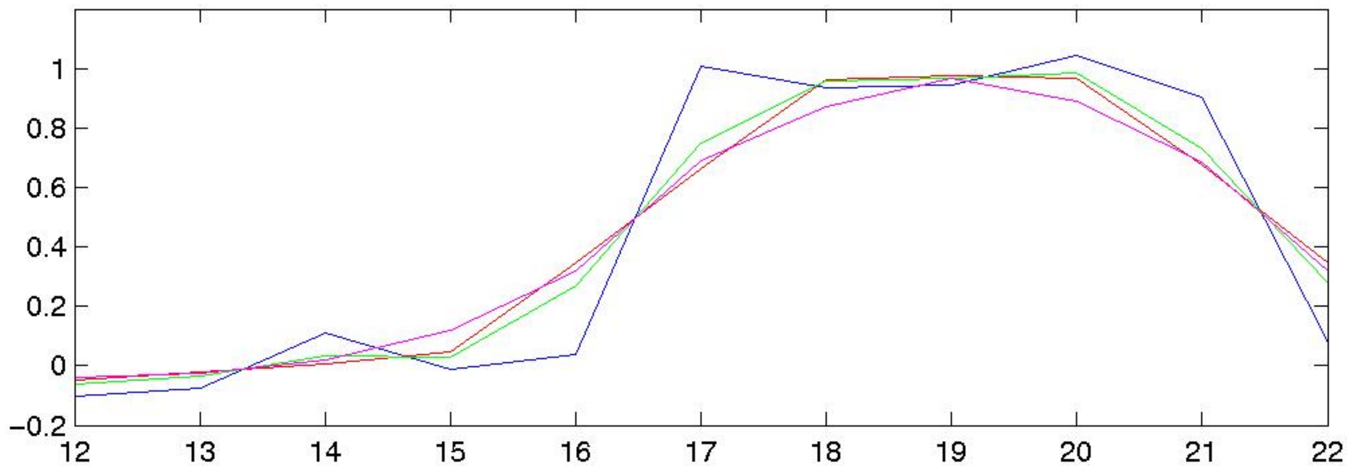


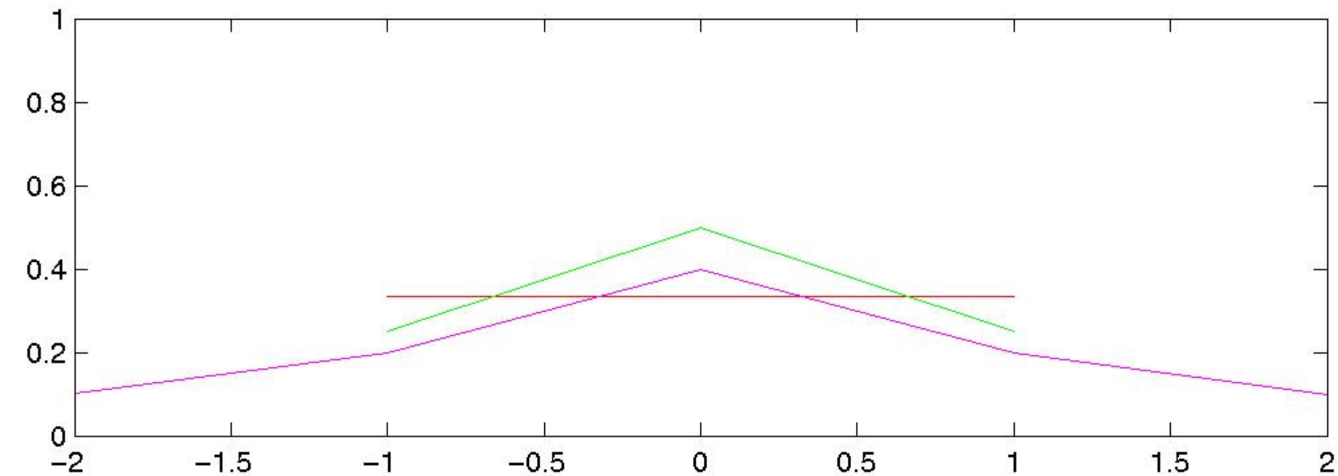


Smoothing it out: averaging



Smoothing it out: averaging





Convolution

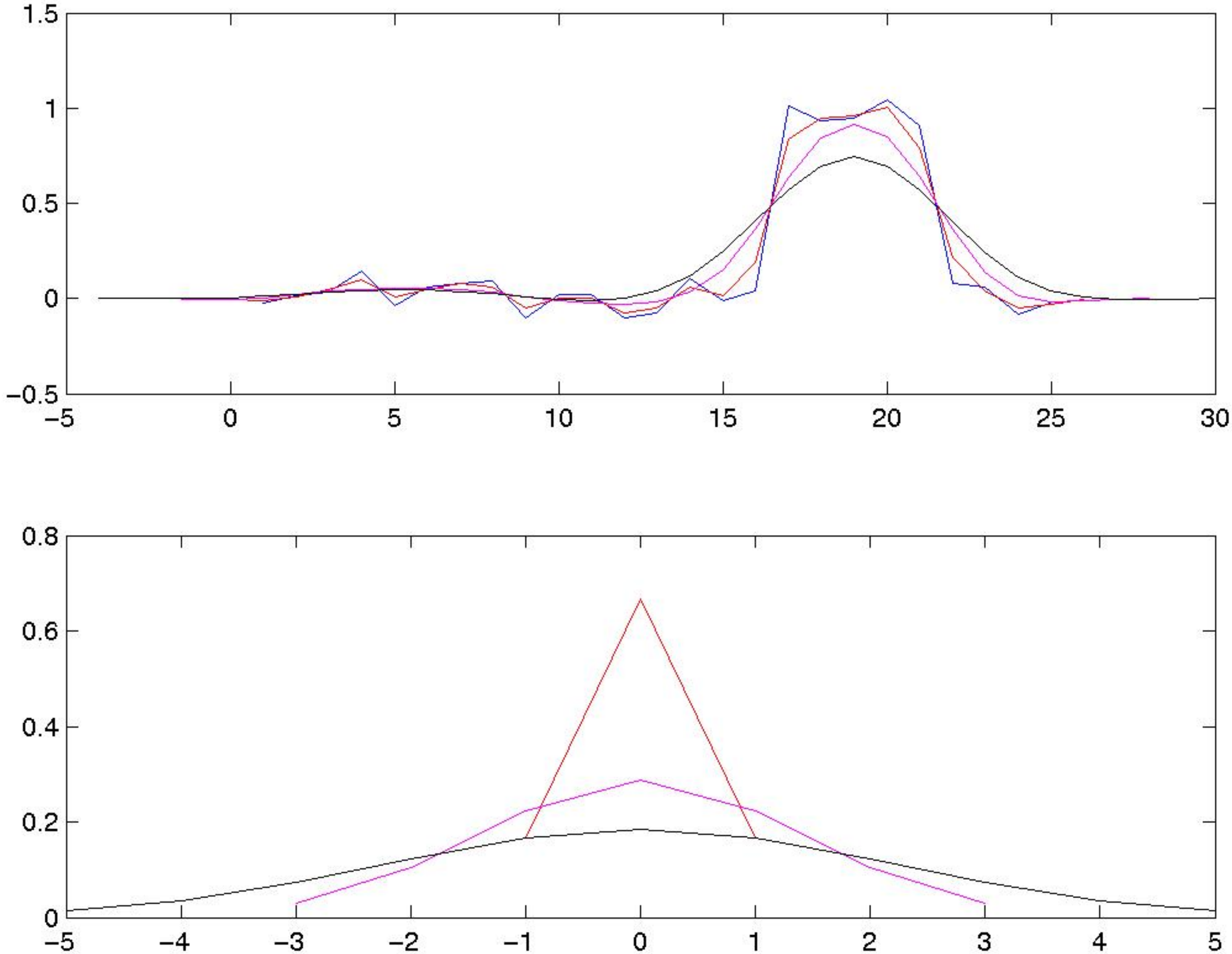
- Method of applying a filter to a signal
- Discrete-case equation

$$y(t)=\sum^{\infty}_{n=-\infty} x(t)h(t-n)$$

also written $y=x*h$

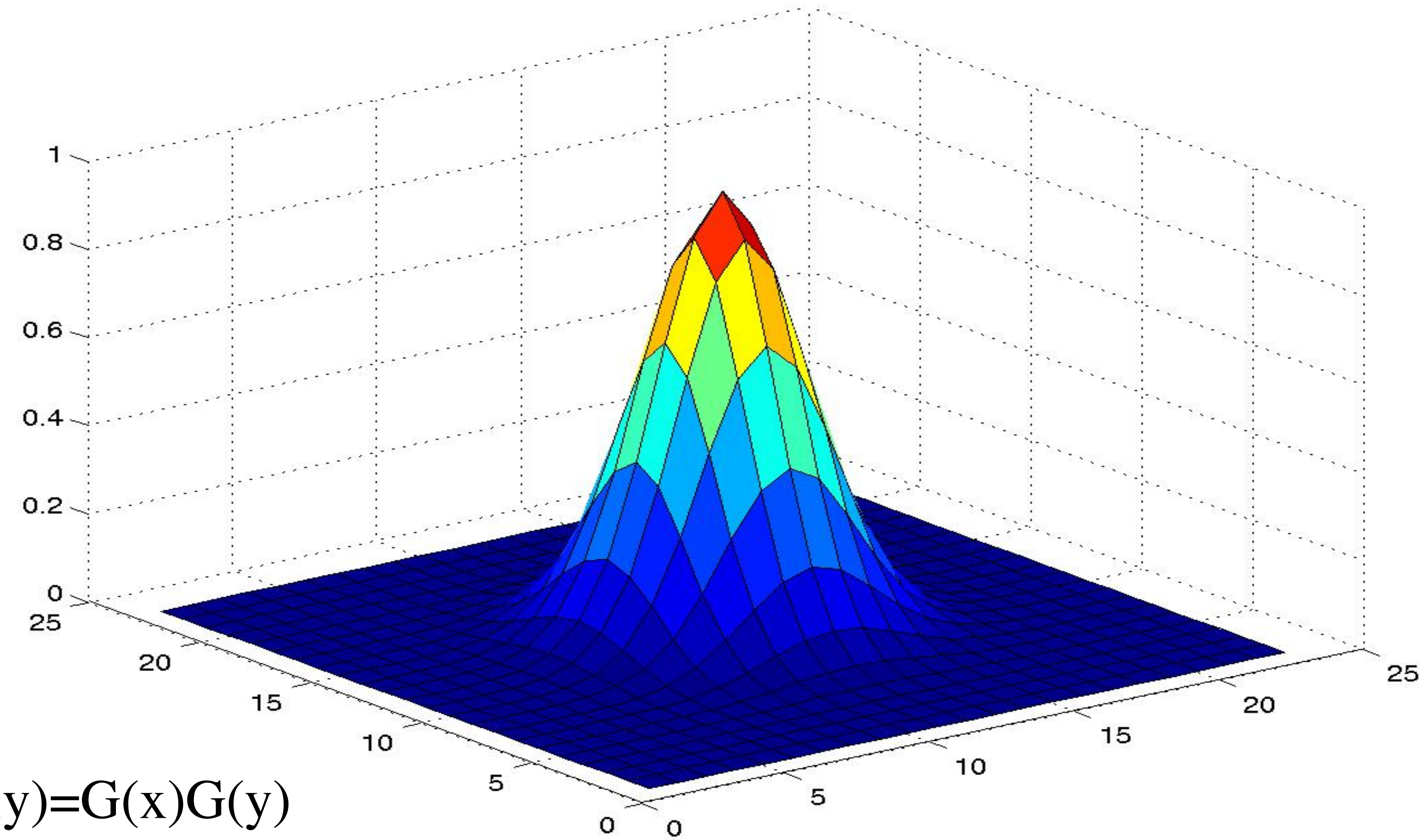
Note: convolution is commutative: $y=x*h=h*x$

Number of points if h,x finite: $n_y=n_x+n_h-1$


1.5
1
0.5
0
−0.5
−5
0
5
10
15
20
25
30
0.8
0.6
0.4
0.2
0
−5
−4
−3
−2
−1
0
1
2
3
4
5

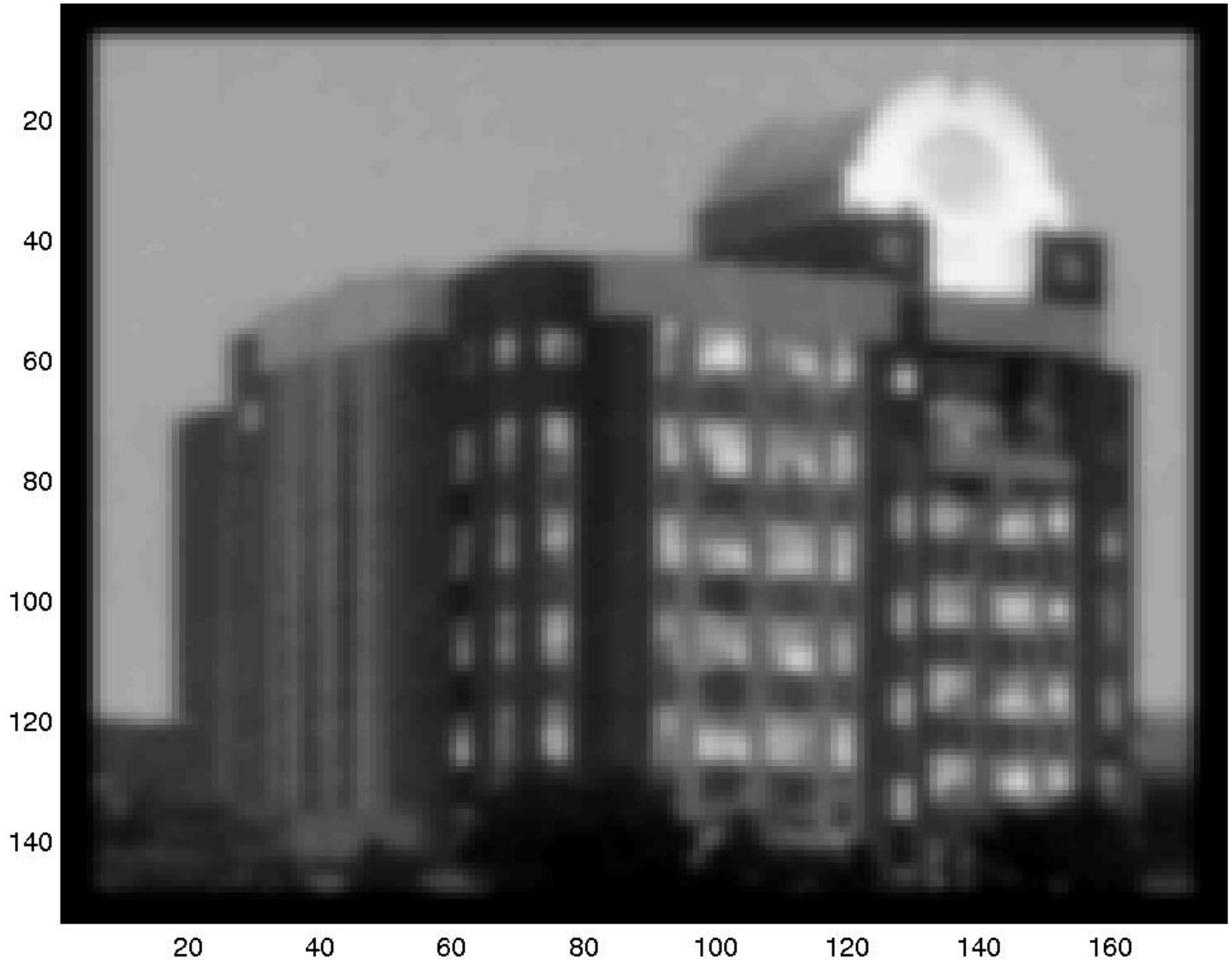
Raw image

2-D Gaussian function

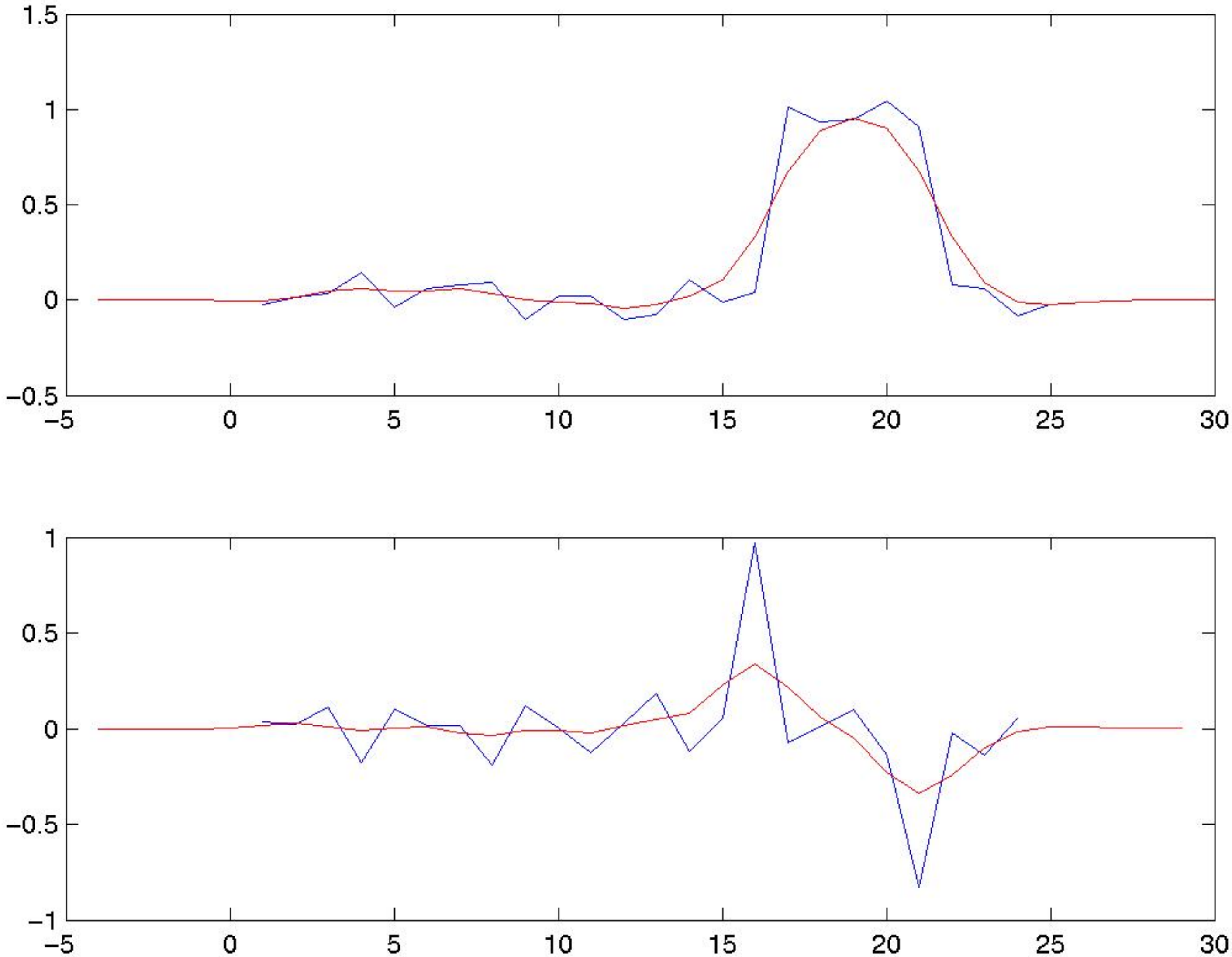


G(x,y)=G(x)G(y)

Gaussian-smoothed image



Detecting edges: 1-D



Detecting edges: 1-D

- Take first derivative of smoothed signal

  y=(x*g)’

  - can be estimated by derivative filter d=[-1 1]

    y=(x*g)*d

- Theorem: derivative of convolved signal is equivalent to convolving signal with derivative of filter

  y=x*(g’)≈x*(g*d)

  - Combine two steps into 1!

Detecting edges: 2-D

- Remember: gaussian components independent

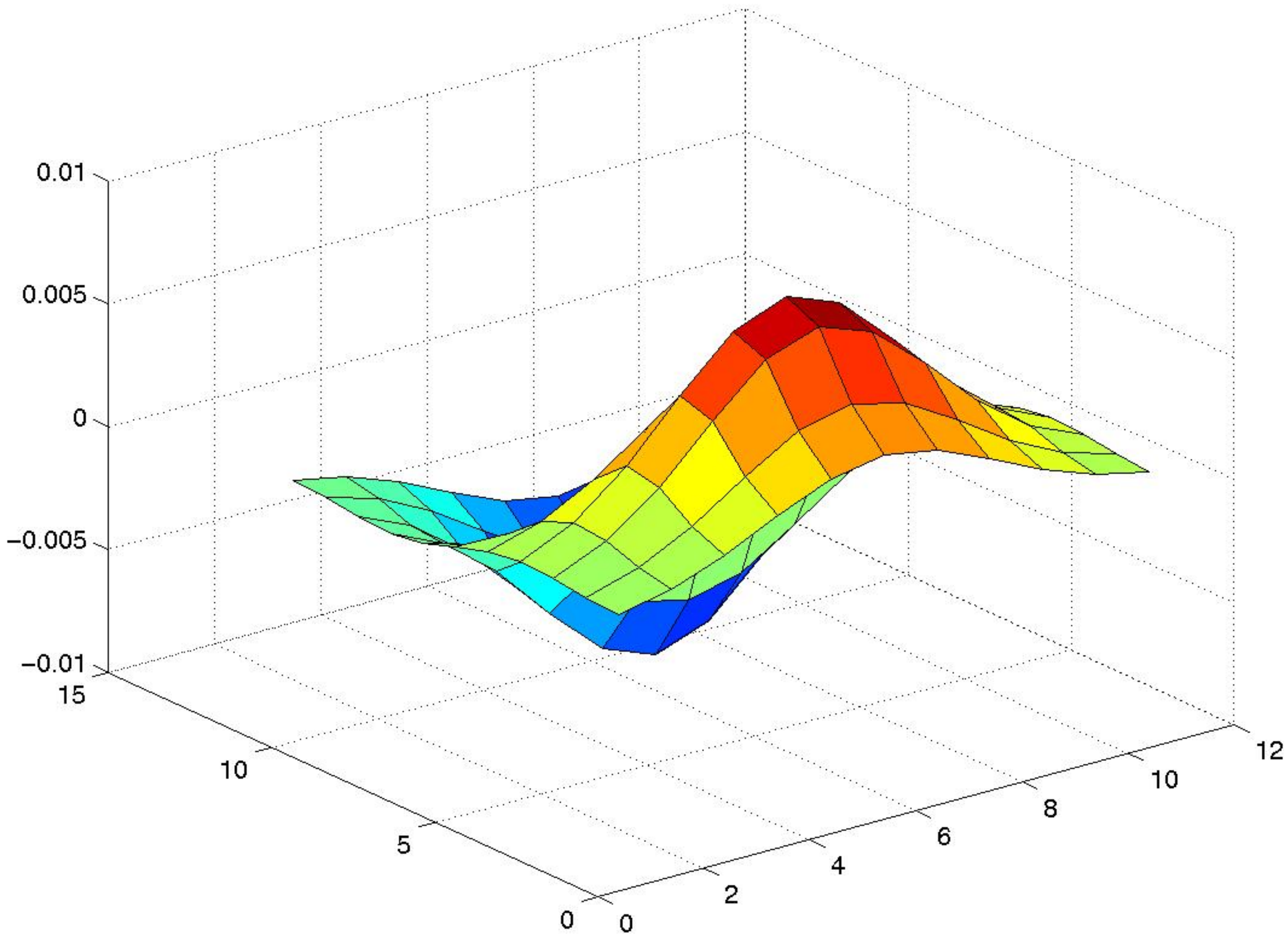
  $G(x,y)=G(x)G(y)$

- How do you find vertical edges?
  - Derivative in x-direction will change

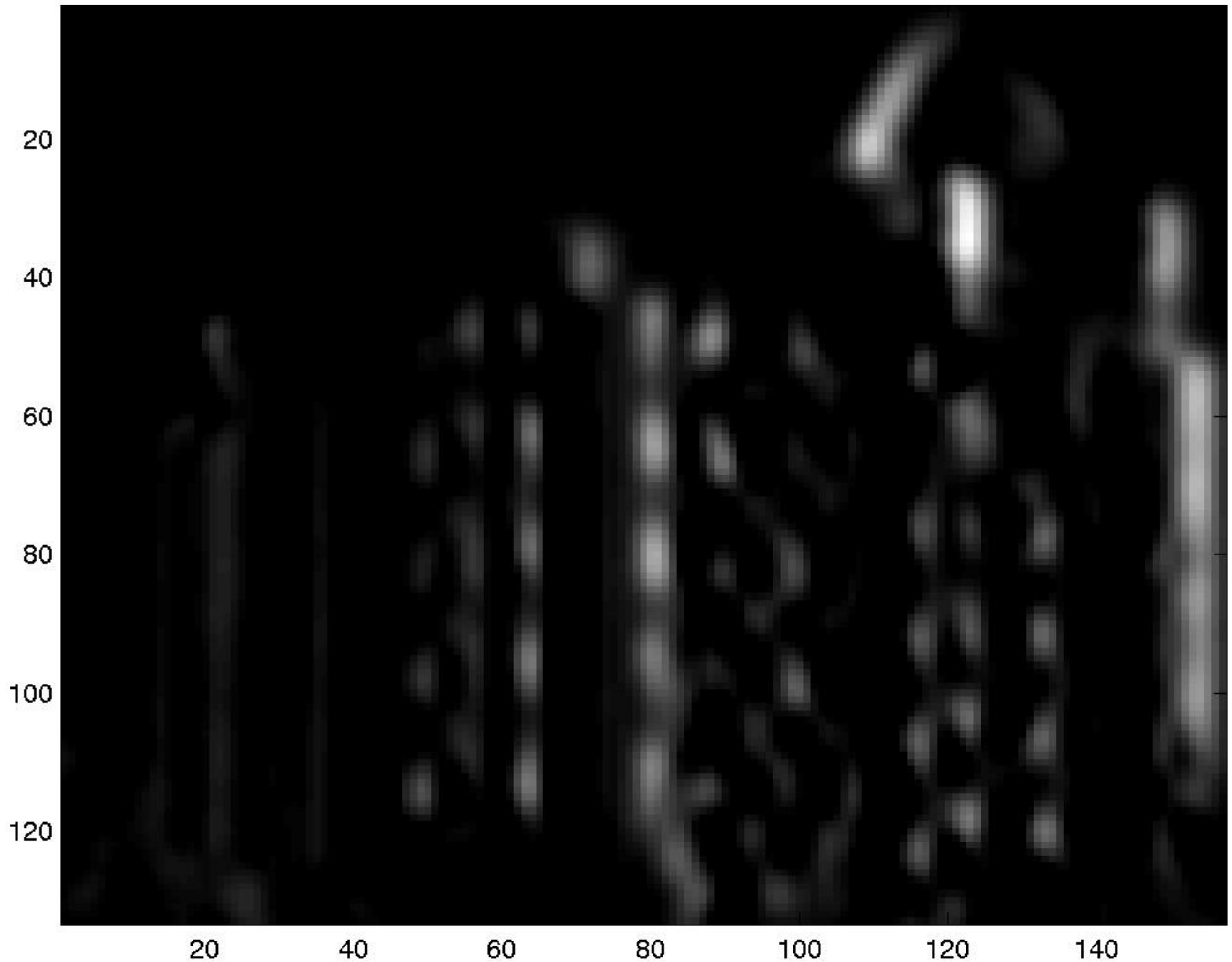
  $G_x(x,y)=G'(x)G(y)$

- Similar derivative in y-direction

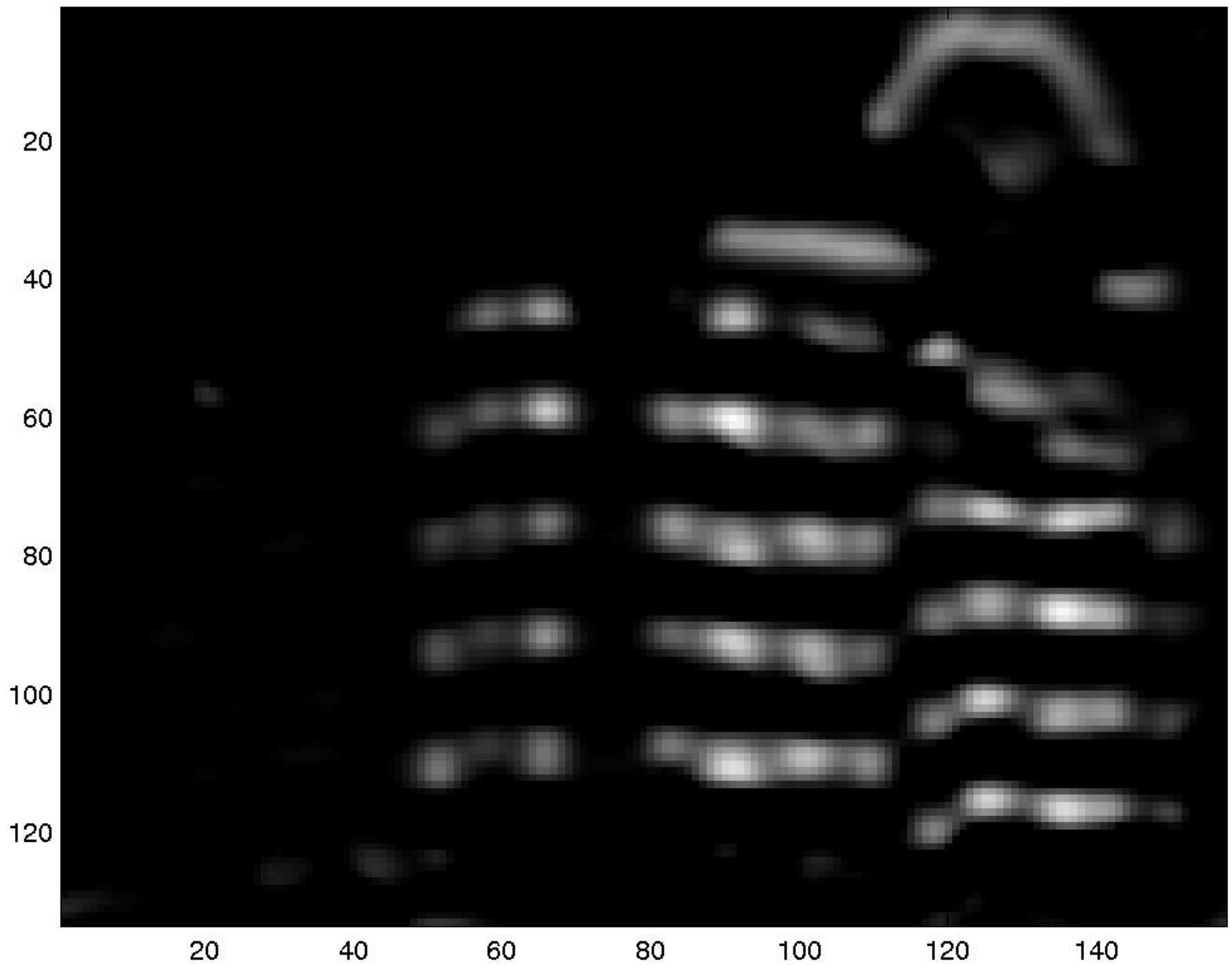
Gaussian Derivatives



Vertical lines: x-derivative

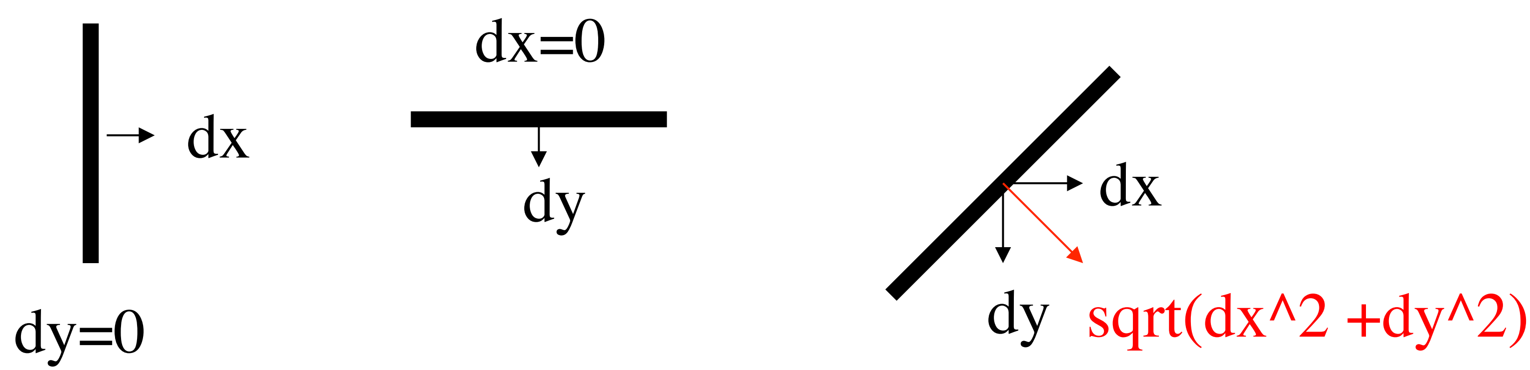


Horizontal lines: y-derivative



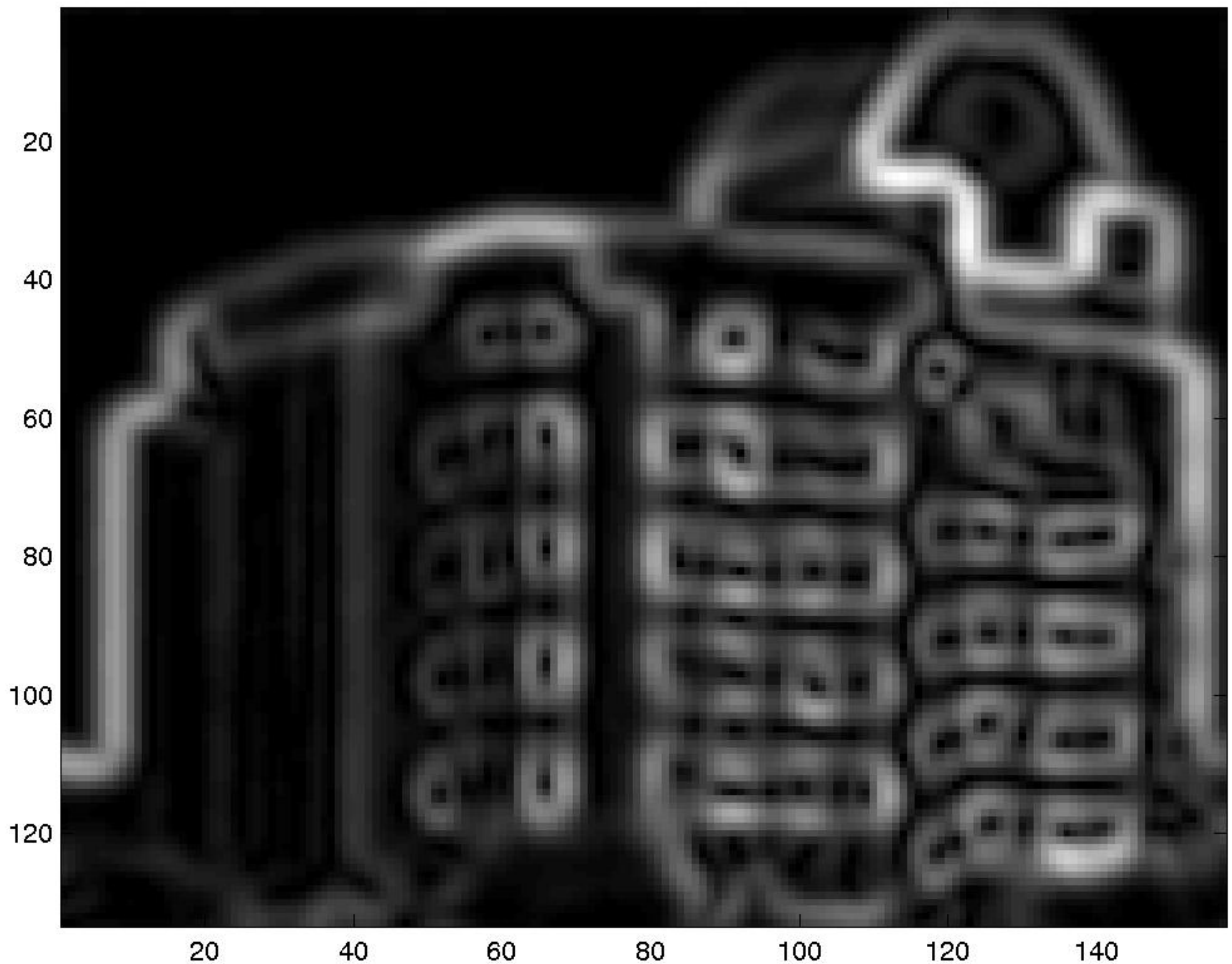
Non-horizontal/vertical lines

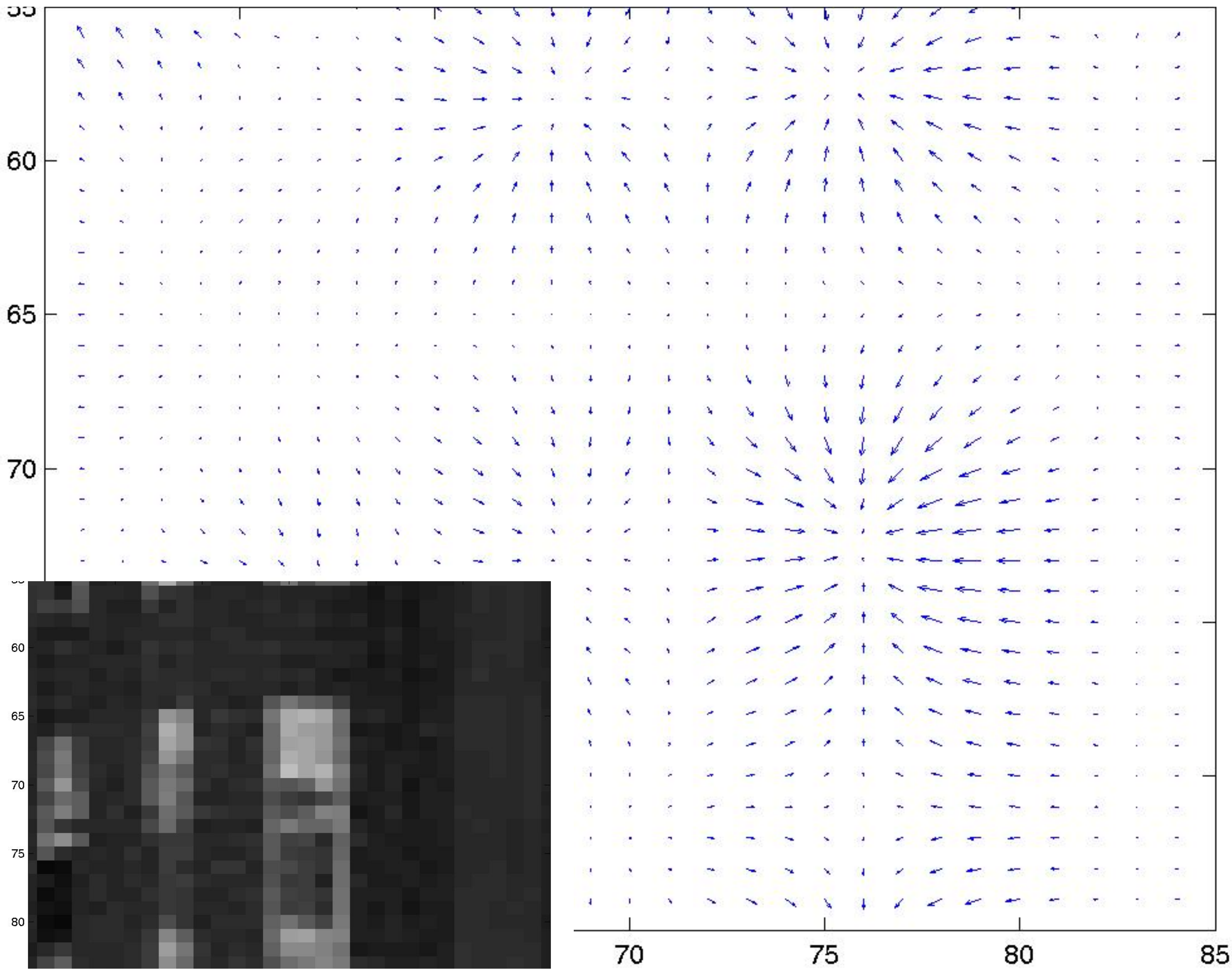
- Can combine x and y components to get gradient — normal to edge



- Angle of gradient is 90 degrees off from edge

Combined derivatives

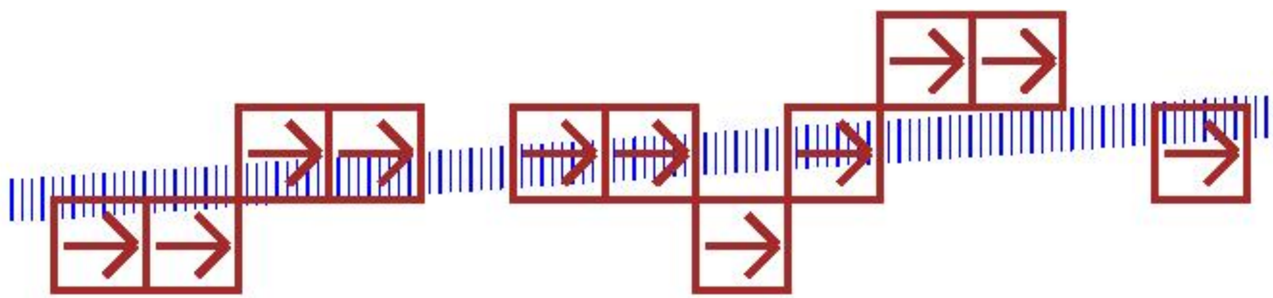




60
65
70
70
75
80
85
60
65
70
75
80

Canny edge detection

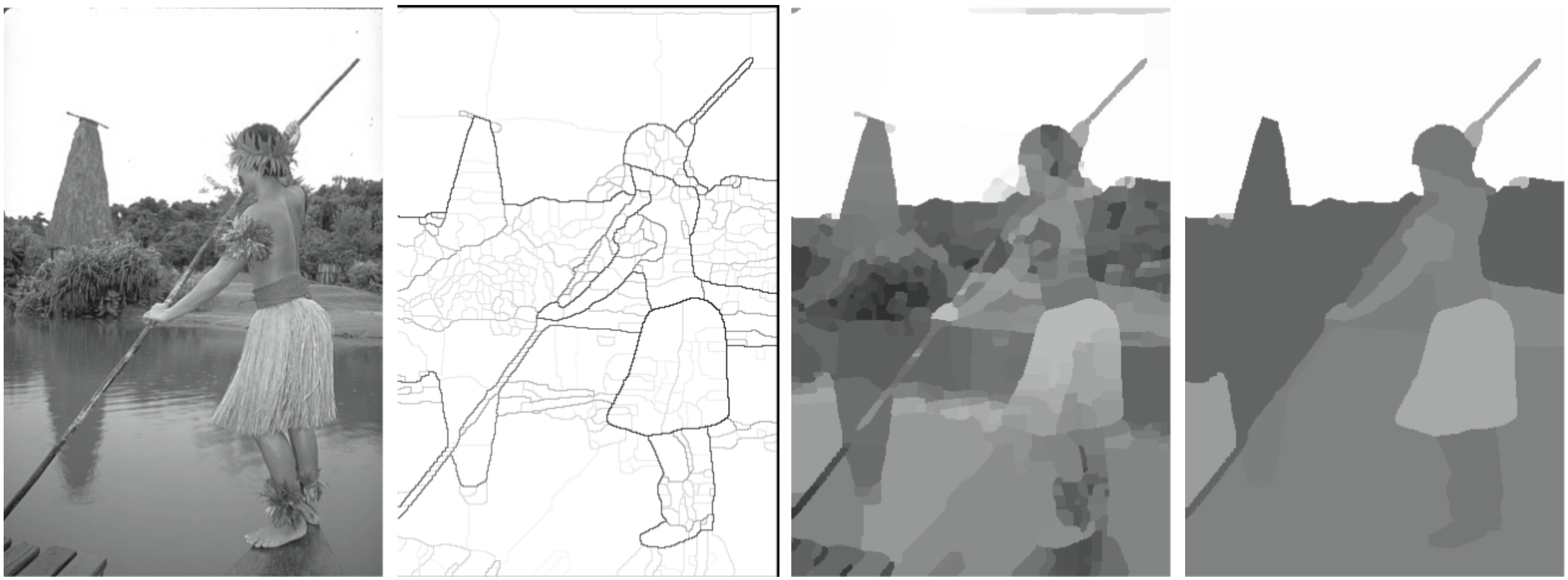
- Smooth with gaussian
- Find gradient of image (I.e., derivative in x and y directions)
- Find edge pixels next to each other that point in similar direction



Thresholding

- Sometimes need binary decision on edges
  - Doesn’t always work well, might need something more clever

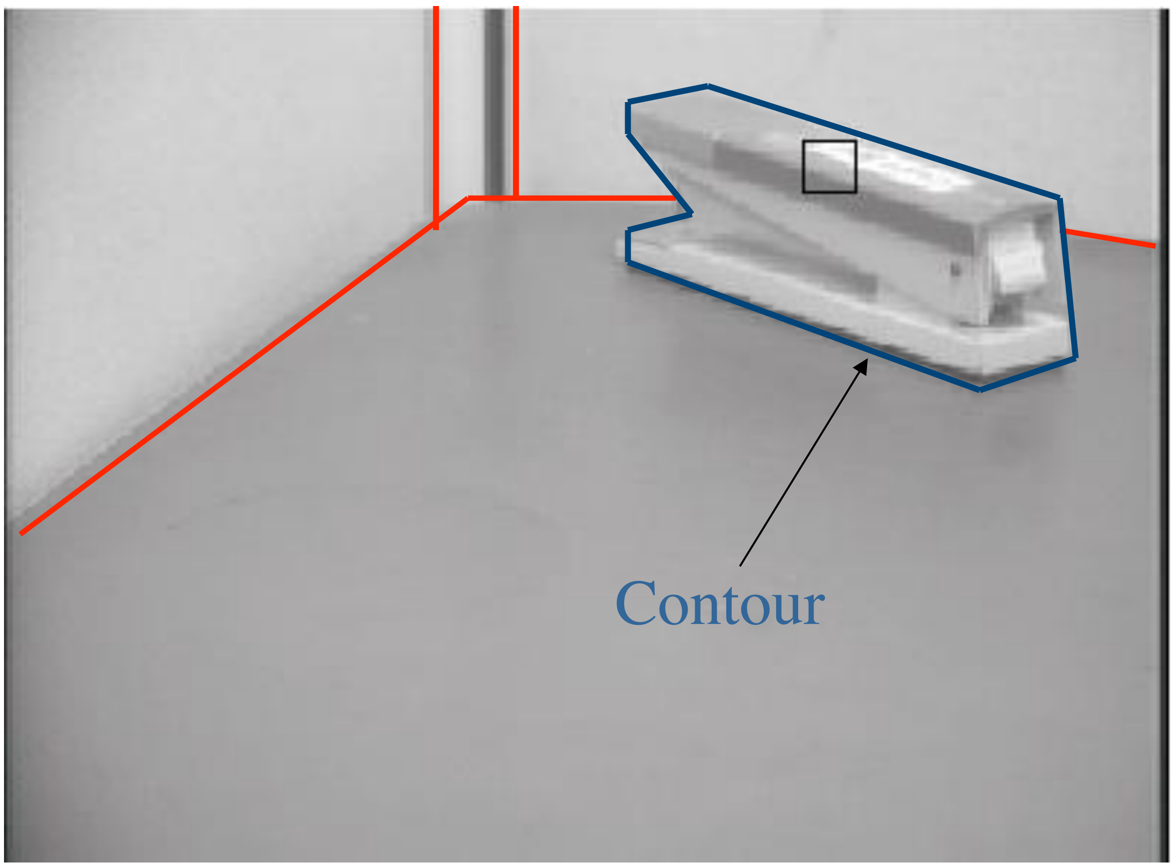
Image segmentation



Cues from prior knowledge

shading

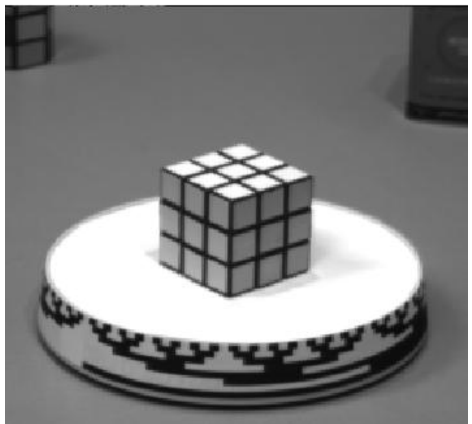
Cues from prior knowledge

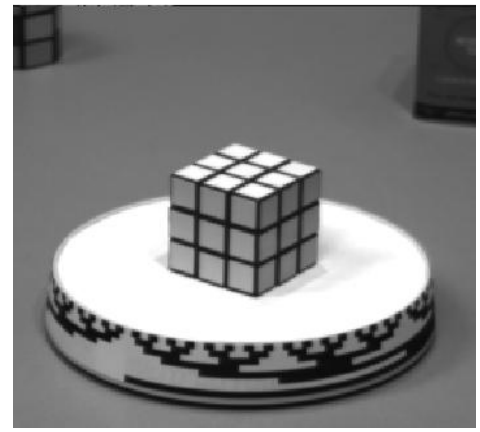


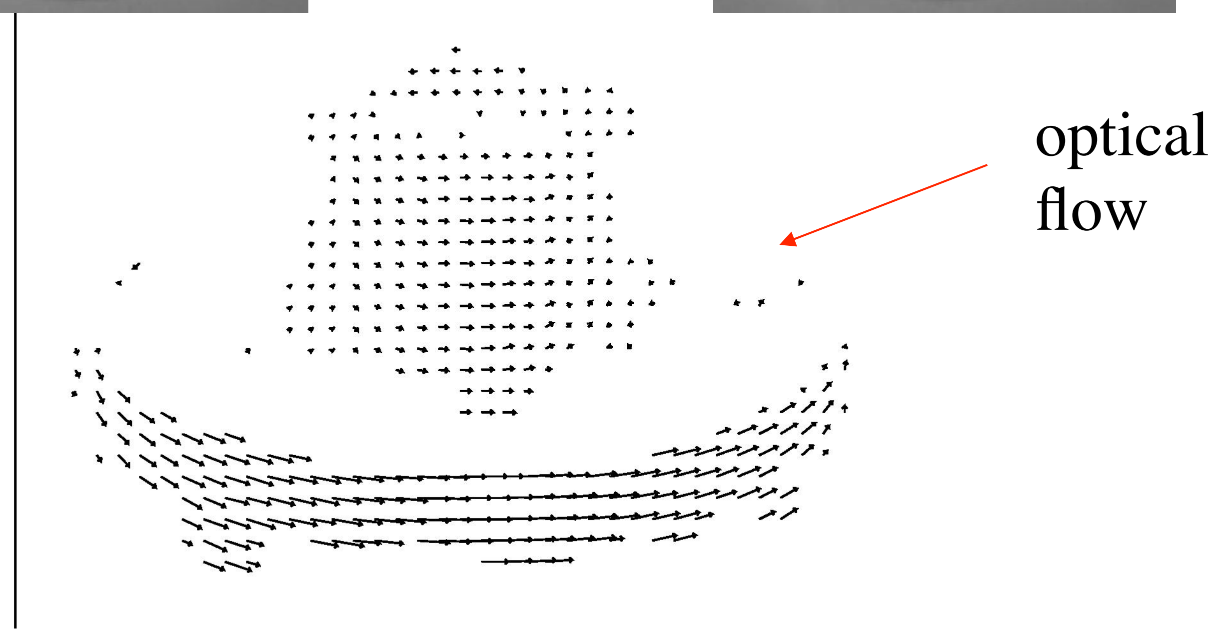
Cues from prior knowledge

- Shading: assumes uniform reflectance
- Contour: assumes minimal curvature
- Texture: assumes uniform texture
- Motion: assumes rigid bodies, continuous motion
- Stereo: assumes solid, contiguous, non-repeating bodies

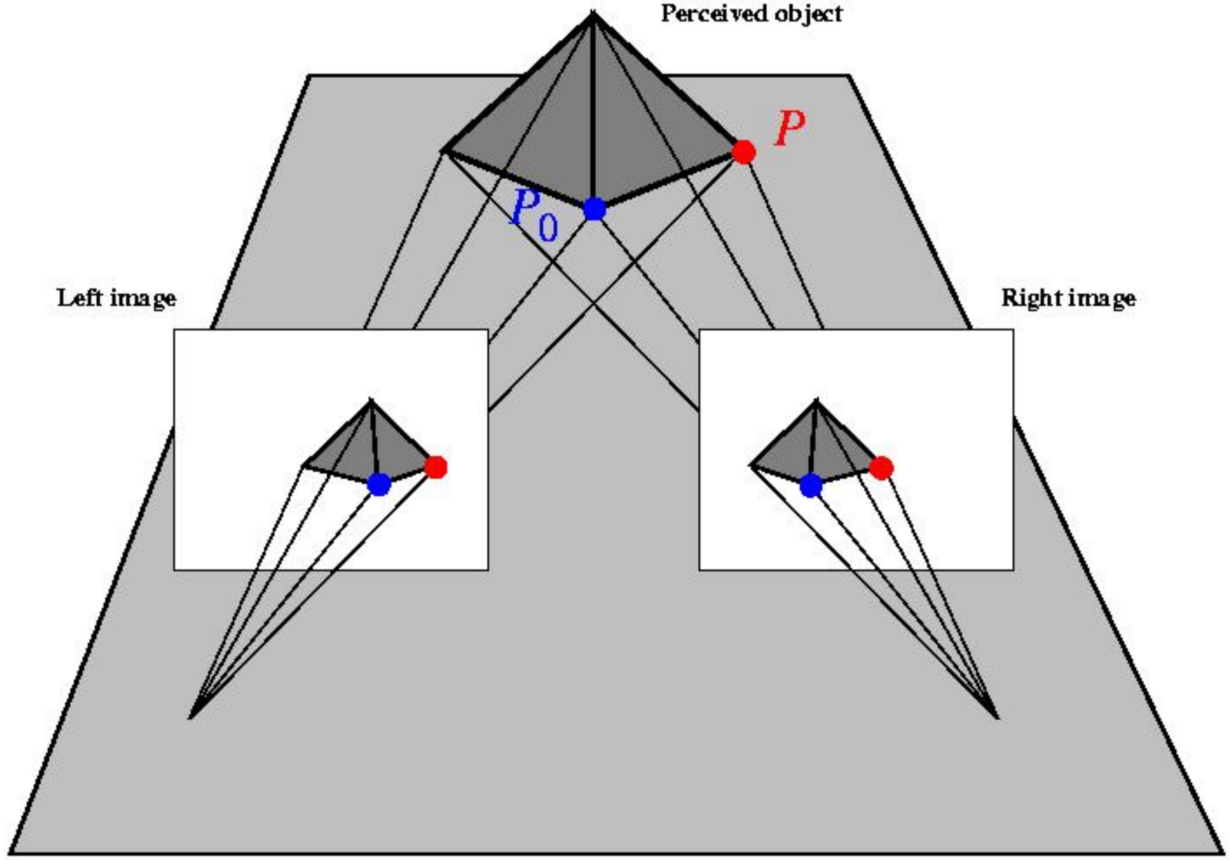
Motion: dx/dt, dy/dt



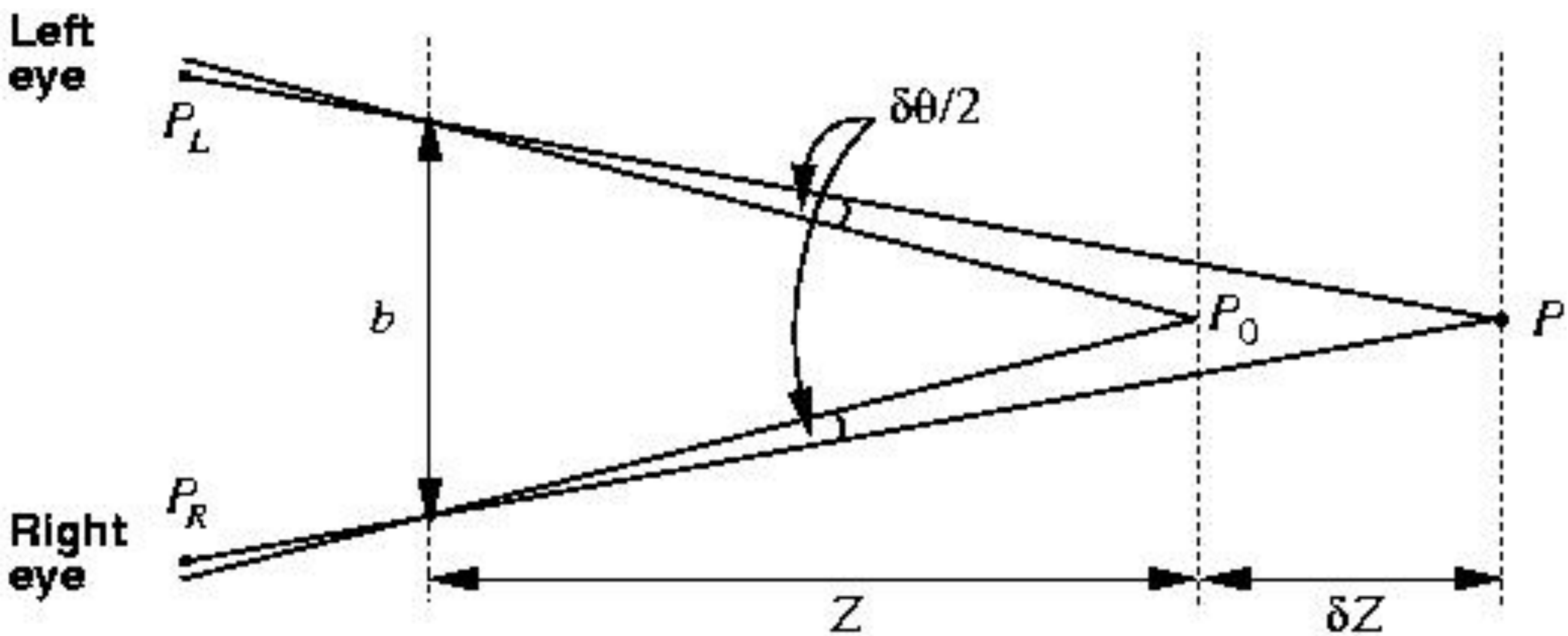




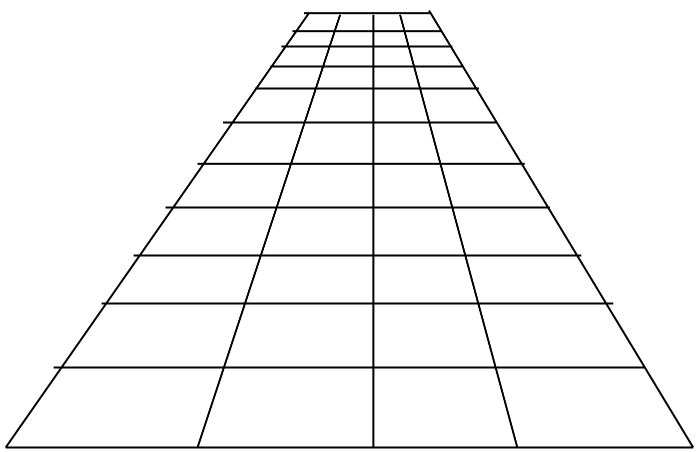
Stereo vision



Stereo vision: depth perception

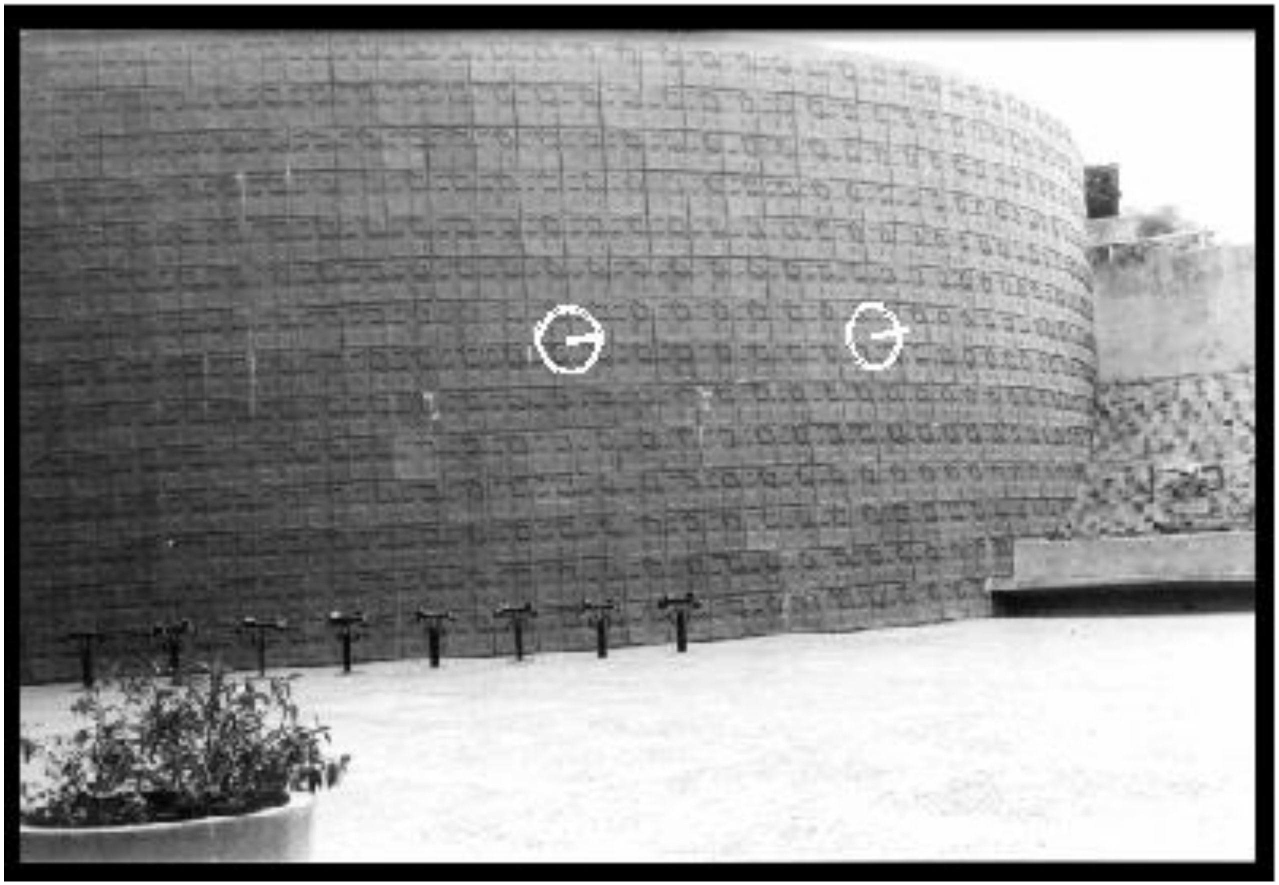


Texture



- Patterns repeat in environment
- Use correspondences between parts of pattern to determine orientation of surface

Texture

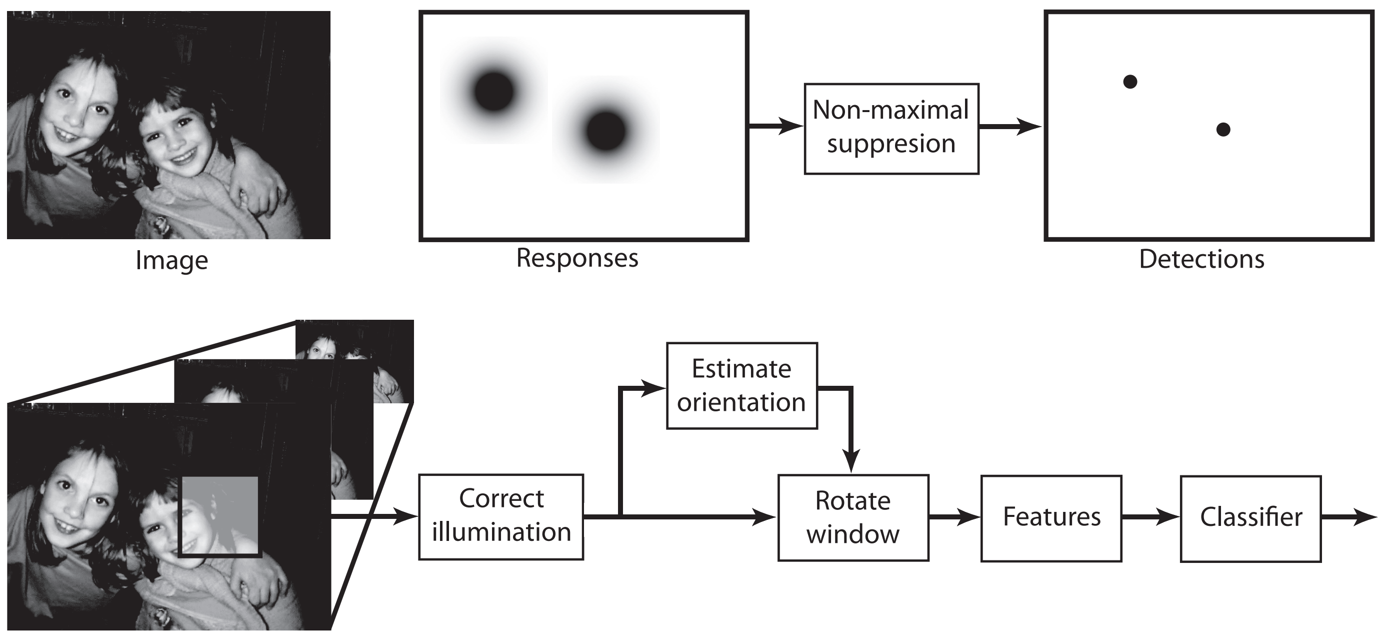


Object recognition

- First: segment the world
- Then: put the pieces back together and figure out what's there
  - Biometric identification: fingerprints, iris scans
  - Content-based image retrieval:
    - Find a photo with "X" in it
  - Handwriting recognition
    - E.g. zip codes

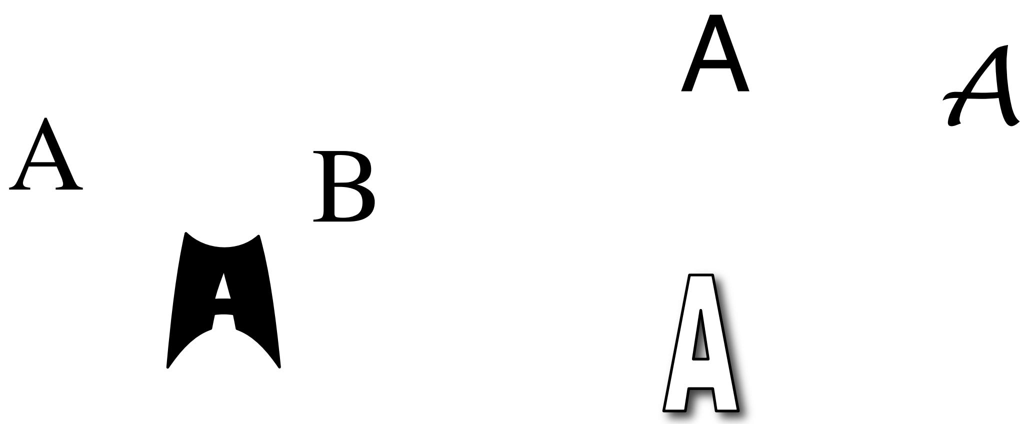
Image content filtering

- How would you find faces in a picture?
- How would you find pictures of trees?

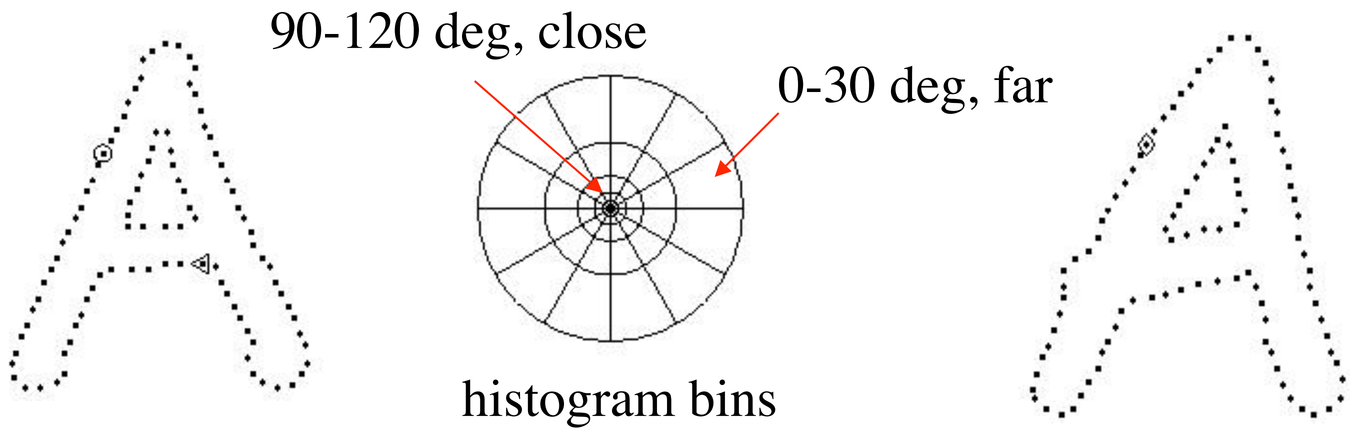


Feature-based recognition

- Extract features (e.g., edges, surfaces, etc)
- Compare against a shape library
- Which of these is not like the other?

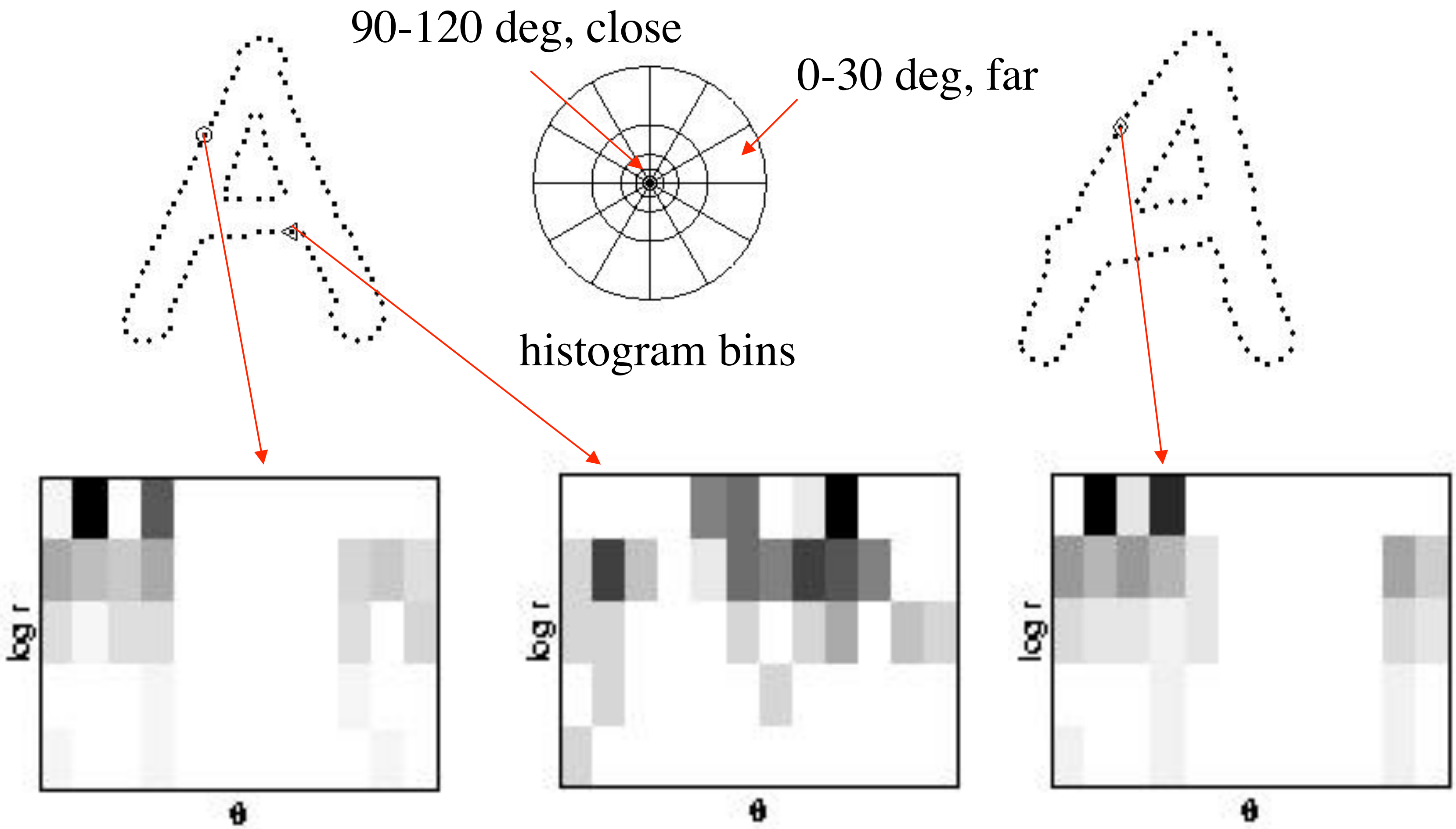


Shape matching

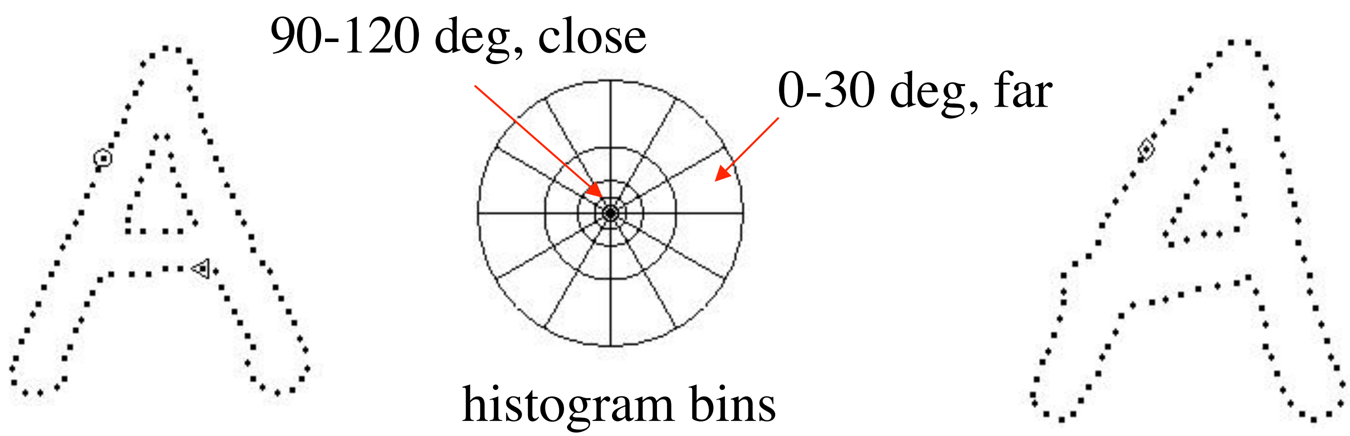


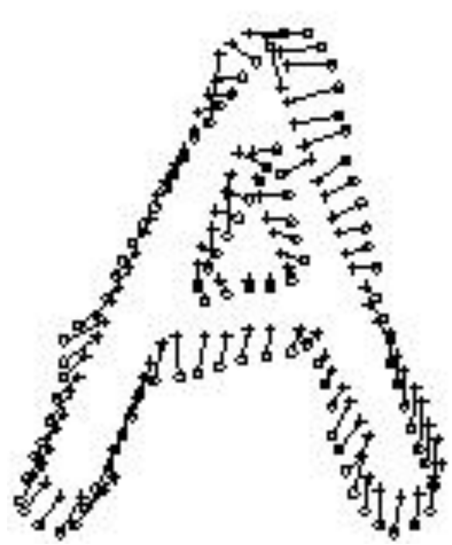
- Don't try to recognize lines
- Distribute a number of points equally around shape
- At every point, take a histogram of the surrounding points
  - Group by angle, log distance

Shape matching



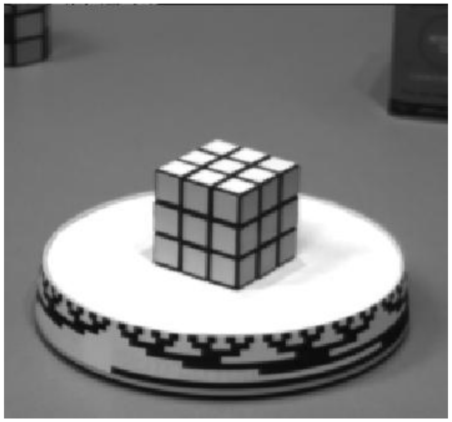
Shape matching

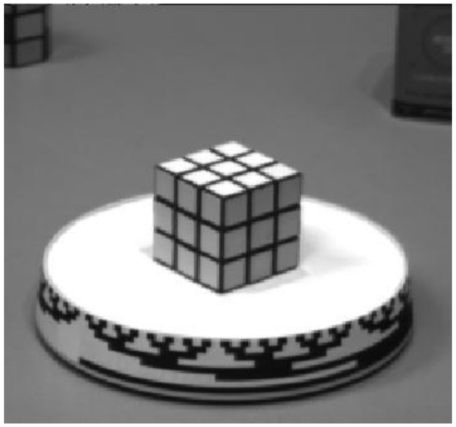




Best shape: minimizes sum of distances under the best matching

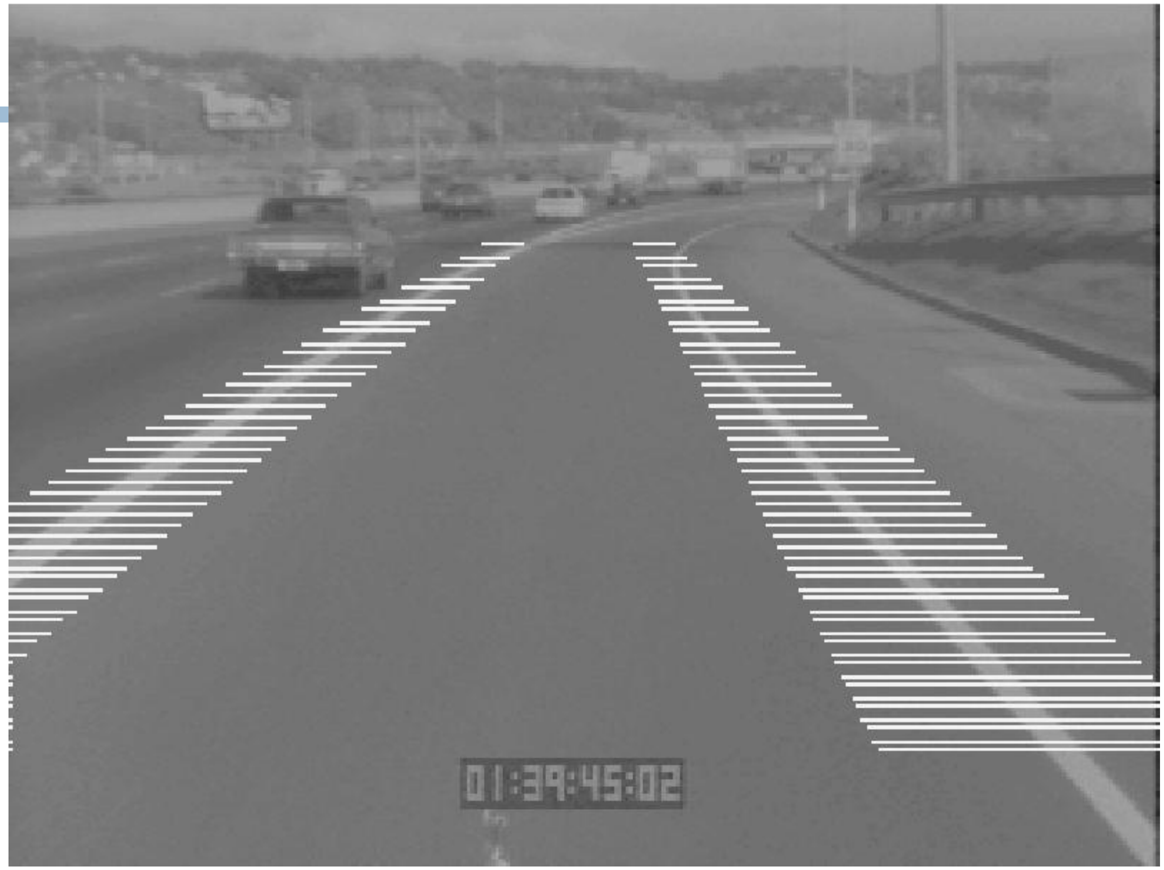
Pose estimation





- Once you match up an object, can you tell which way it's facing
- Determine rotation R and translation t

Computer vision in the real world



What would you need to do this?