Edge Detection

Sedat Ozer
Department of Computer Engineering
Bilkent University
sedat@cs.bilkent.edu.tr

ANNOUNCEMENT: Paper Presentations

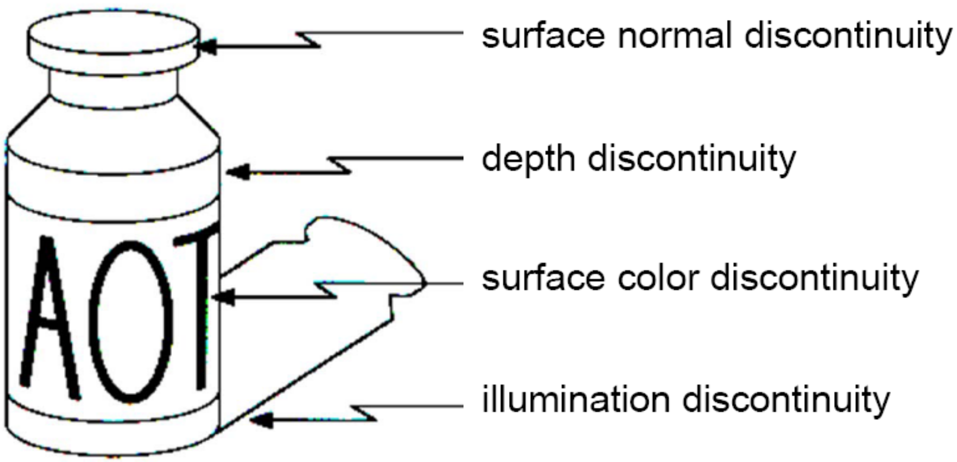
- Groups will present their chosen papers **on April 22 or on April 24**.
  - Each group will be assigned to one of those two days randomly.
- Each group must **submit their presentation slides by April 21 midnight.**
- Each group is assigned a time slot of **15 minutes**.
  - After 15 minutes, you will receive a **penalty for going over the time**.
  - Finish your slides **in 13-14 minutes** and give **one or two minutes** for potential questions. Rehearse a couple times before presenting your paper.
- Each group will decide on who to present the paper.
  - It is not necessary that all members should present a portion of the paper,
  - however, each member is responsible with knowing all the material in their slides.
  - All the group members can can asked random questions about the paper.
- See the course website for organizing your paper presentations (under paper presentations section).
- Check the email that you were sent for more details.

Edge detection

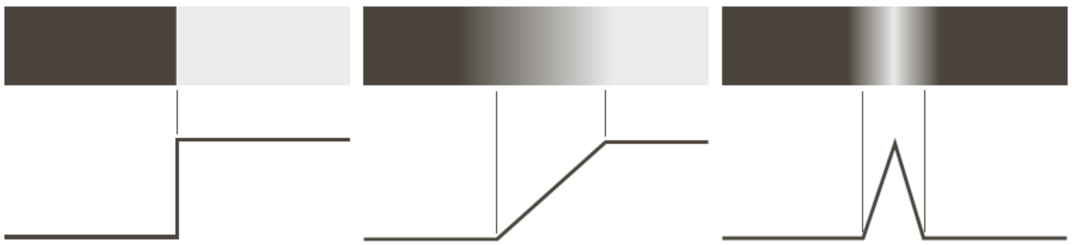
- Edge detection is the process of finding meaningful transitions in an image.
- The points where sharp changes in the brightness occur typically form the border between different objects or scene parts.
- Further processing of edges into lines, curves and circular arcs result in useful features for matching and recognition.
- Initial stages of mammalian vision systems also involve detection of edges and local features.

Edge detection

- Sharp changes in the image brightness occur at:
  - Object boundaries
    - A light object may lie on a dark background or a dark object may lie on a light background.
  - Reflectance changes
    - May have quite different characteristics – zebras have stripes, and leopards have spots.
  - Cast shadows
  - Sharp changes in surface orientation



Edge models



a b c

FIGURE 10.8 From left to right, models (ideal representations) of a step, a ramp, and a roof edge, and their corresponding intensity profiles.

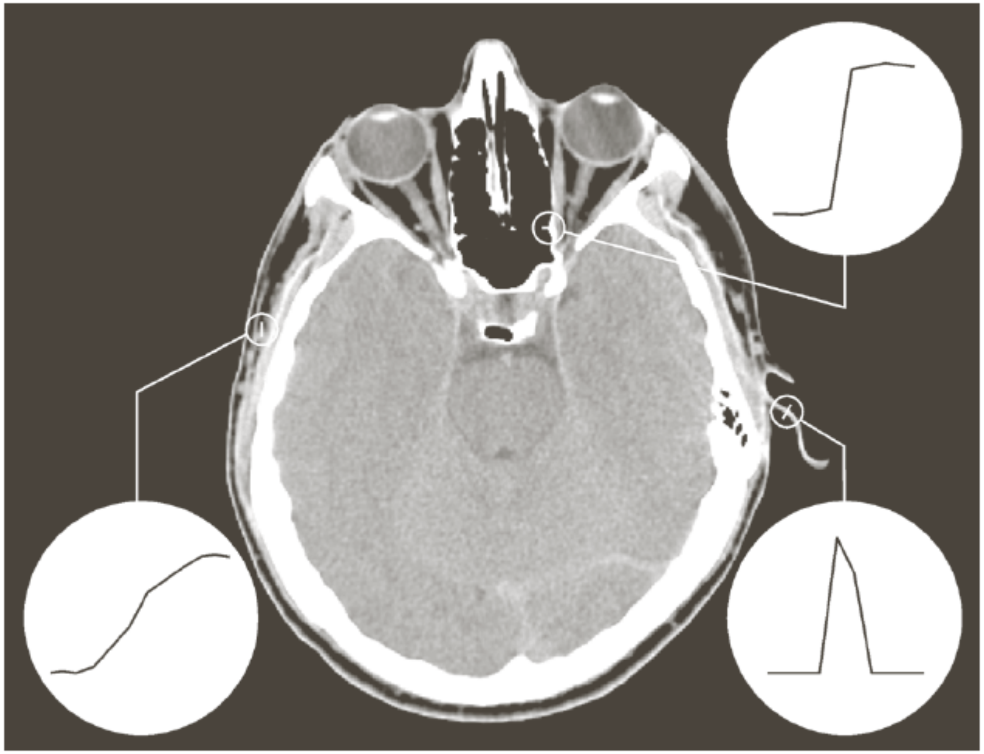
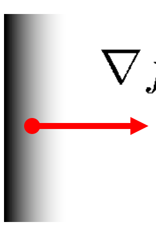


FIGURE 10.9 A 1508 × 1970 image showing (zoomed) actual ramp (bottom, left), step (top, right), and roof edge profiles. The profiles are from dark to light, in the areas indicated by the short line segments shown in the small circles. The ramp and "step" profiles span 9 pixels and 2 pixels, respectively. The base of the roof edge is 3 pixels. (Original image courtesy of Dr. David R. Pickens, Vanderbilt University.)

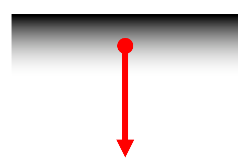
Difference operators for 2D

- Contrast in the 2D picture function $f(x, y)$ can occur in any direction.
- From calculus, we know that the maximum change occurs along the direction of the *gradient*.
- The gradient of an image $f(x, y)$ at location (x, y) is defined as the vector

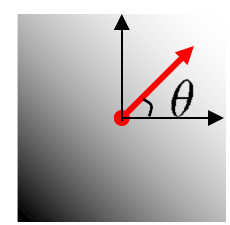
$$\nabla f = \left[\frac{\partial f}{\partial x} \ \ \frac{\partial f}{\partial y} \right]^T.$$



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



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



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

Difference operators for 2D

- The *magnitude of the gradient*

$$|\nabla f| = \left(\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right)^{1/2}$$

  gives the maximum rate of increase of $f(x, y)$ per unit distance in the direction of ∇f.

- The *direction of the gradient*

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

  represents the direction of this change with respect to the x-axis.

Difference operators for 2D

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

−1	0
0	1

0	−1
1	0

Roberts

−1	−1	−1
0	0	0
1	1	1

−1	0	1
−1	0	1
−1	0	1

Prewitt

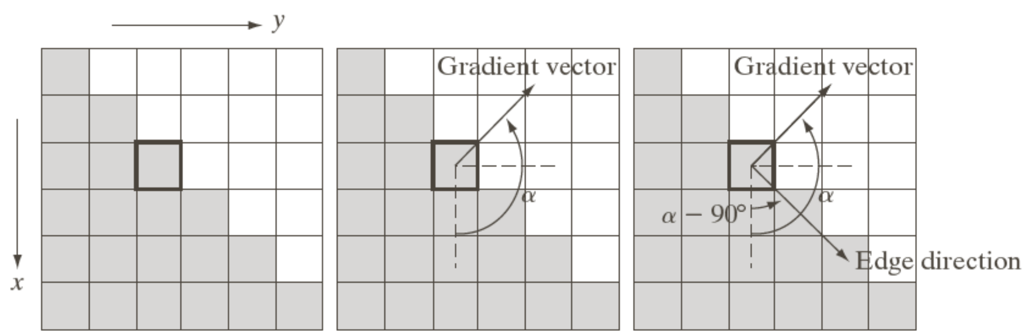
−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

Sobel

a
b c
d e
f g

FIGURE 10.14 A 3 × 3 region of an image (the z's are intensity values) and various masks used to compute the gradient at the point labeled z_5.



a b c

FIGURE 10.12 Using the gradient to determine edge strength and direction at a point. Note that the edge is perpendicular to the direction of the gradient vector at the point where the gradient is computed. Each square in the figure represents one pixel.

Adapted from Gonzales and Woods

Difference operators for 2D

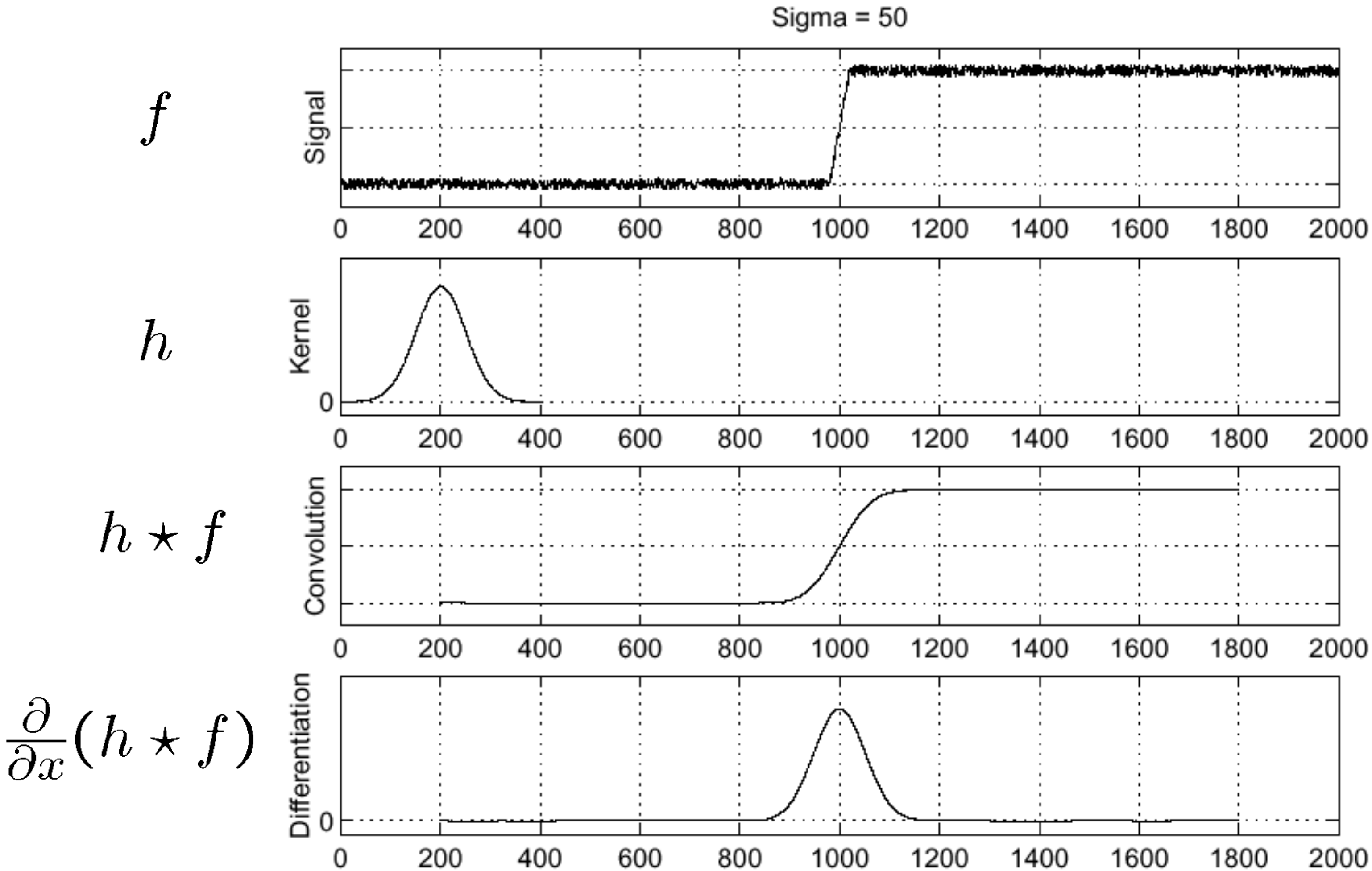
- The *Laplacian* of a 2D function $f(x, y)$ is a second-order derivative defined as

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}.$$

- The Laplacian generally is not used in its original form for edge detection because:
  - It is sensitive to noise.
  - Its magnitude produces double edges.
  - It is unable to detect edge direction.
- However, its zero-crossing property can be used for edge localization.

Difference operators under noise

Solution is to smooth first:



Adapted from Steve Seitz

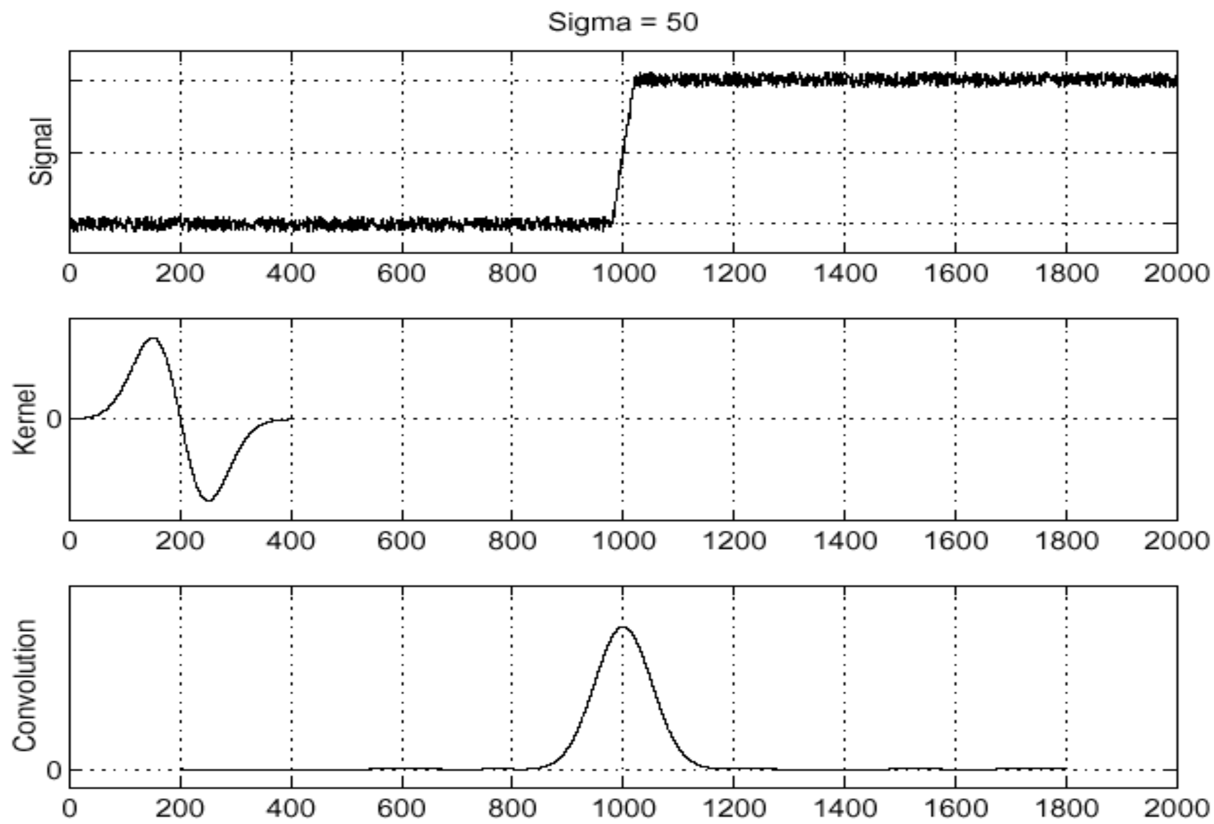
Difference operators under noise

Differentiation property of convolution: $\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$

f

$\frac{\partial}{\partial x}h$

$(\frac{\partial}{\partial x}h) \star f$



Adapted from Steve Seitz

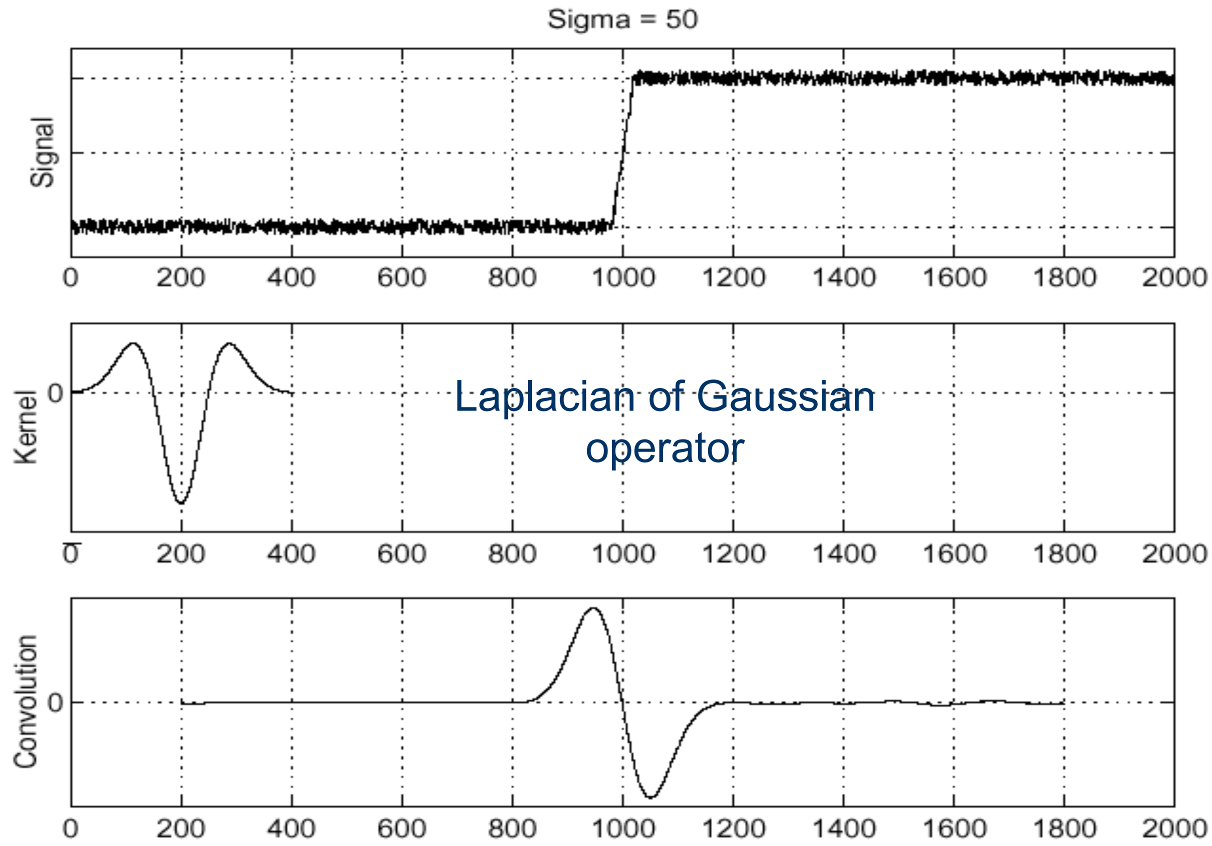
Difference operators under noise

Consider: $\frac{\partial^2}{\partial x^2}(h \star f)$

f

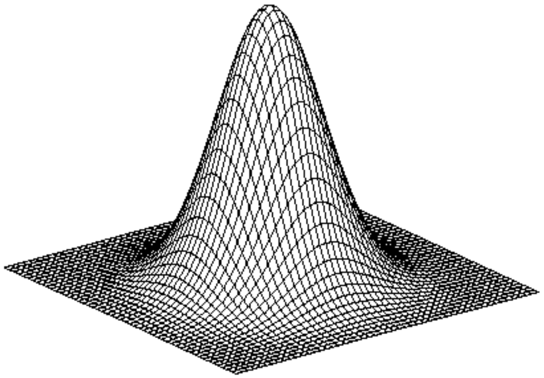
$\frac{\partial^2}{\partial x^2}h$

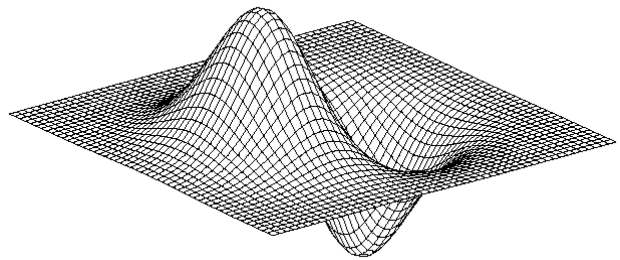
$(\frac{\partial^2}{\partial x^2}h) \star f$



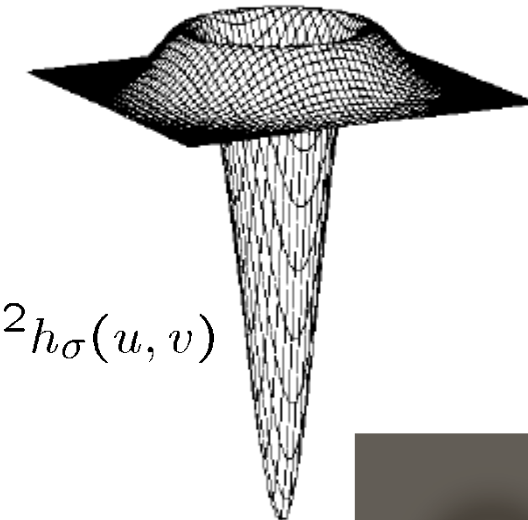
Adapted from Steve Seitz

Edge detection filters for 2D





Laplacian of Gaussian



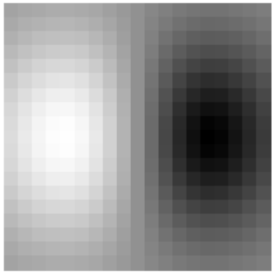
Gaussian

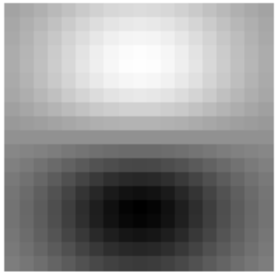
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

derivative of Gaussian

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

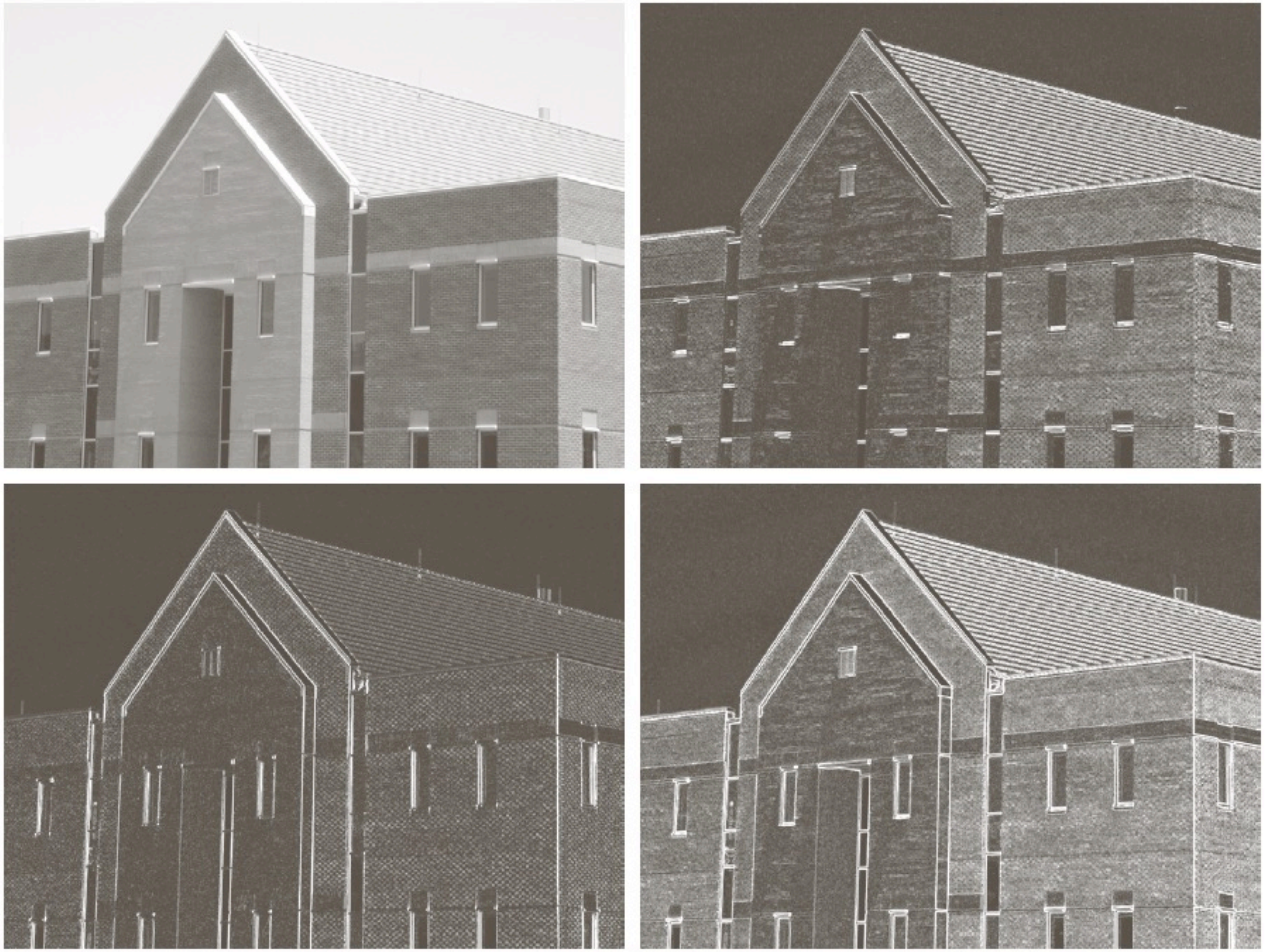
$$\nabla^2 h_\sigma(u,v)$$





Adapted from Steve Seitz, U of Washington

Difference operators for 2D



a	b
c	d

FIGURE 10.16
(a) Original image of size 834×1114 pixels, with intensity values scaled to the range [0, 1]. (b) $|g_x|$, the component of the gradient in the x-direction, obtained using the Sobel mask in Fig. 10.14(f) to filter the image. (c) $|g_y|$, obtained using the mask in Fig. 10.14(g). (d) The gradient image, $|g_x| + |g_y|$.

Difference operators for 2D

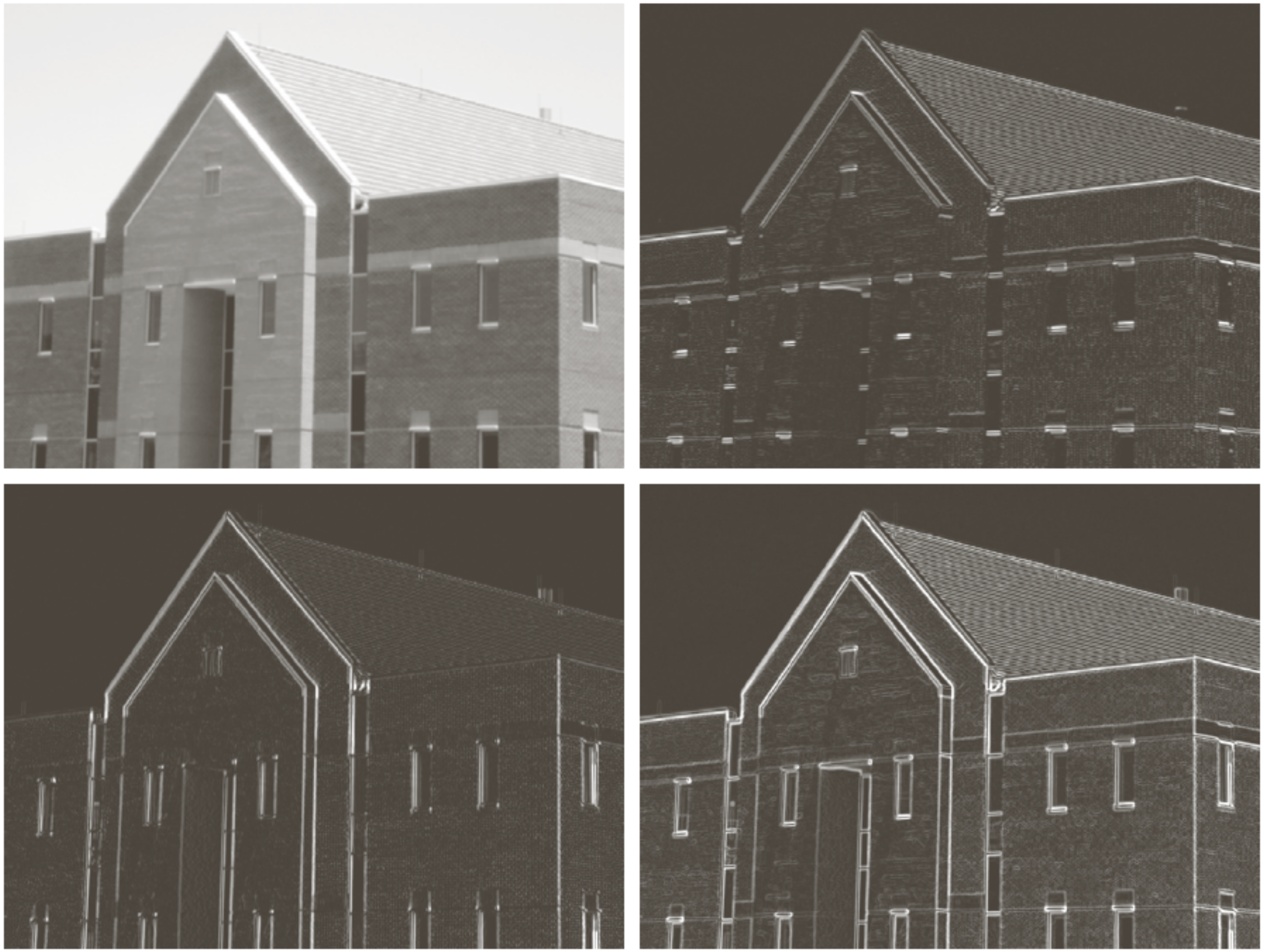


FIGURE 10.18 Same sequence as in Fig. 10.16, but with the original image smoothed using a 5 × 5 averaging filter prior to edge detection.

Edge detection

- Three fundamental steps in edge detection:
  1. Image smoothing: to reduce the effects of noise.
  2. Detection of edge points: to find all image points that are potential candidates to become edge points.
  3. Edge localization: to select from the candidate edge points only the points that are true members of an edge.

Canny edge detector

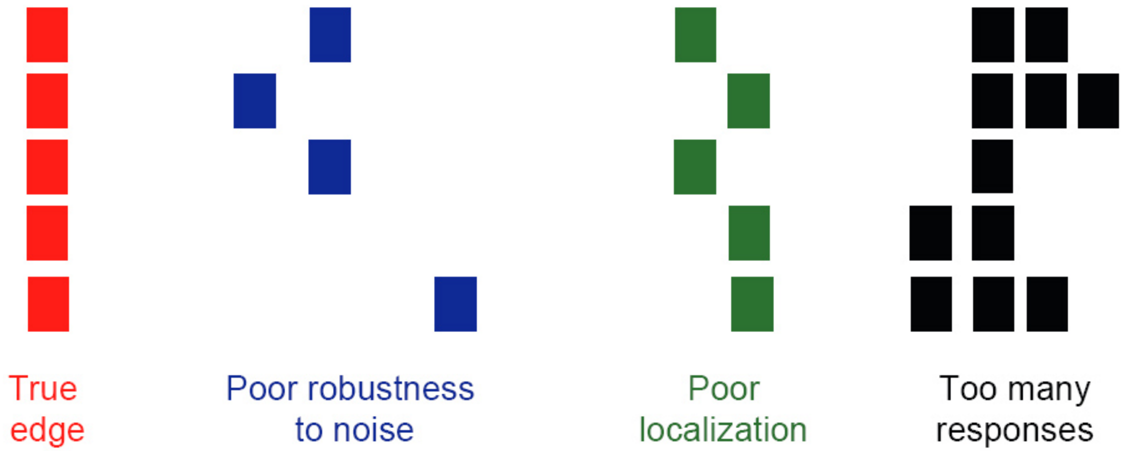
- The Canny operator gives single-pixel-wide images with good continuation between adjacent pixels.
- It is the most widely used edge operator today; no one has done better since it came out in the late 80s. Many implementations are available.
- It is very sensitive to its parameters, which need to be adjusted for different application domains.

Canny edge detector

- The Canny operator gives single-pixel-wide images with good continuation between adjacent pixels.
- It is the **most widely used edge operator** today; Many implementations are available.
- It is very sensitive to its parameters, which need to be adjusted for different application domains.

Designing an edge "detector"

- Criteria for an "optimal" edge detector:
  - **Good detection:** the optimal detector must **minimize the probability of false positives** (detecting spurious edges caused by noise), as well as that **of false negatives** (missing real edges)
  - **Good localization:** the edges detected must be as close as possible to the true edges
  - **Single response:** the detector must return one point only for each true edge point; that is, minimize the number of local maxima around the true edge



Source: L. Fei-Fei

Canny edge detector

1. Smooth the image with a Gaussian filter with spread σ.
2. Compute gradient magnitude and direction at each pixel of the smoothed image.
3. Zero out any pixel response less than or equal to the two neighboring pixels on either side of it, along the direction of the gradient (non-maxima suppression).
4. Track high-magnitude contours using thresholding (hysteresis thresholding).
5. Keep only pixels along these contours, so weak little segments go away.

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

Canny edge detector

Original image (Lena)

Adapted from Steve Seitz, U of Washington

Canny edge detector

Magnitude of the gradient

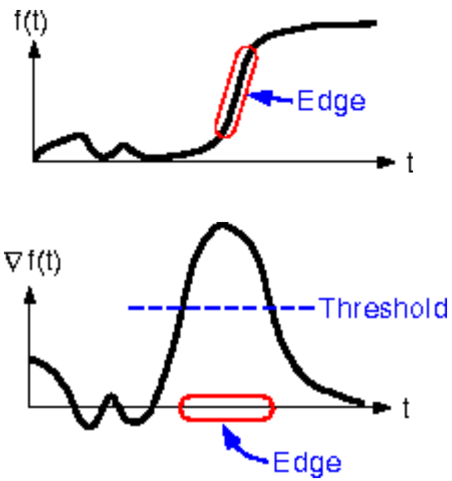
Adapted from Steve Seitz, U of Washington

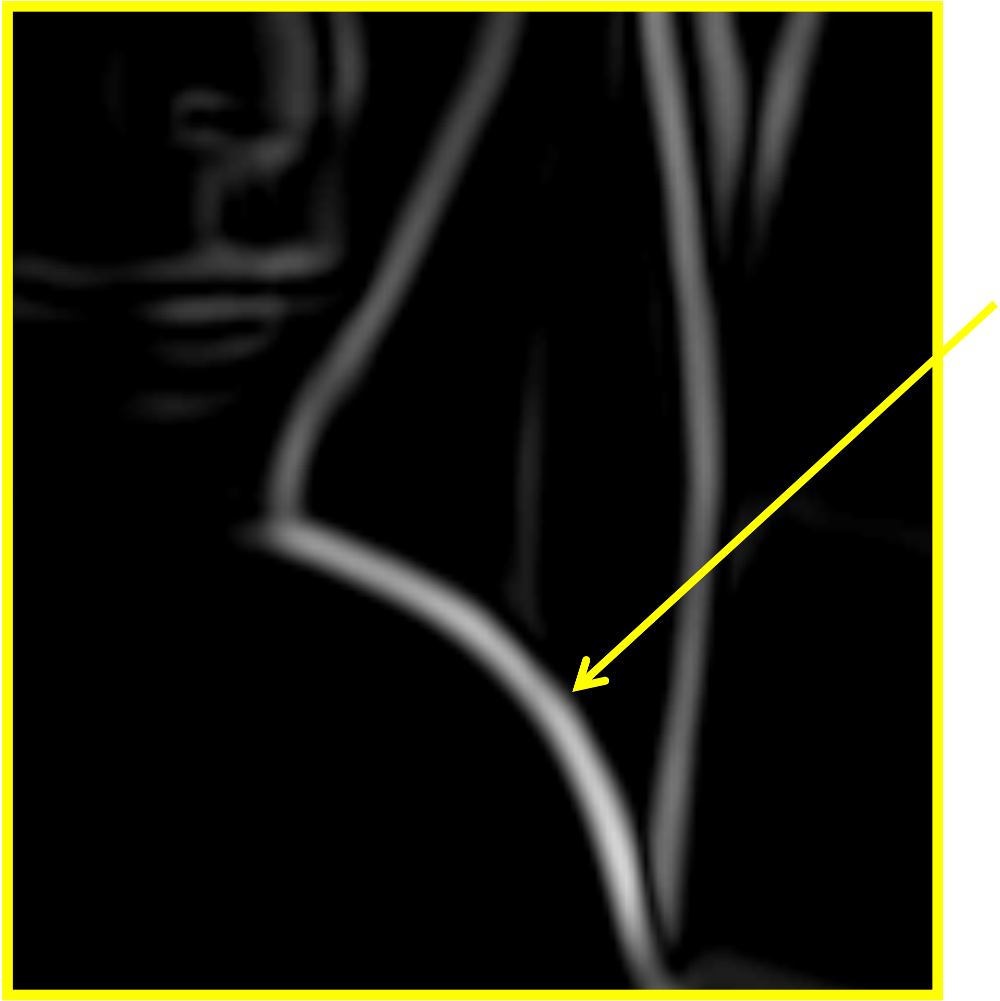
Canny edge detector

Thresholding

Adapted from Steve Seitz, U of Washington

Canny edge detector



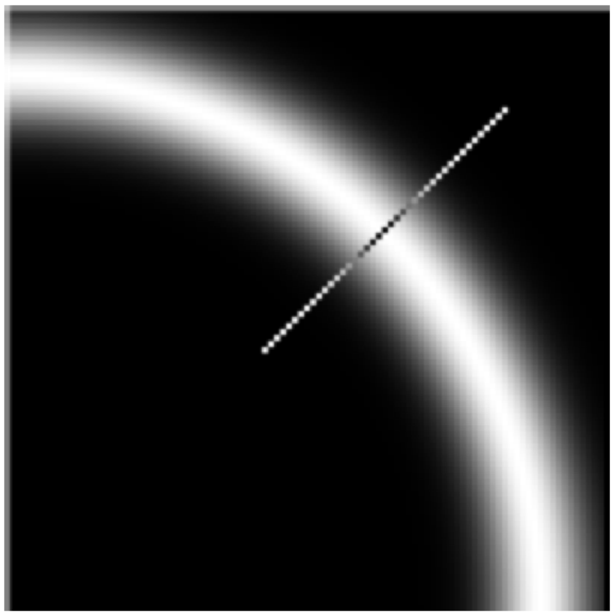


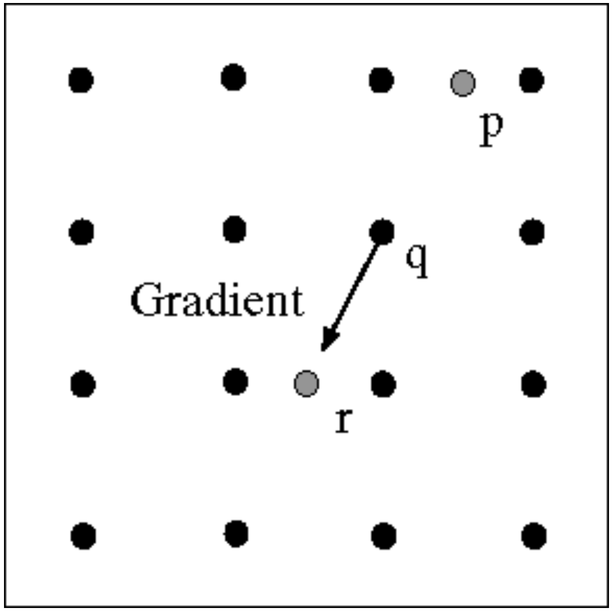
How to turn these thick regions of the gradient into curves?

Adapted from Steve Seitz, U of Washington

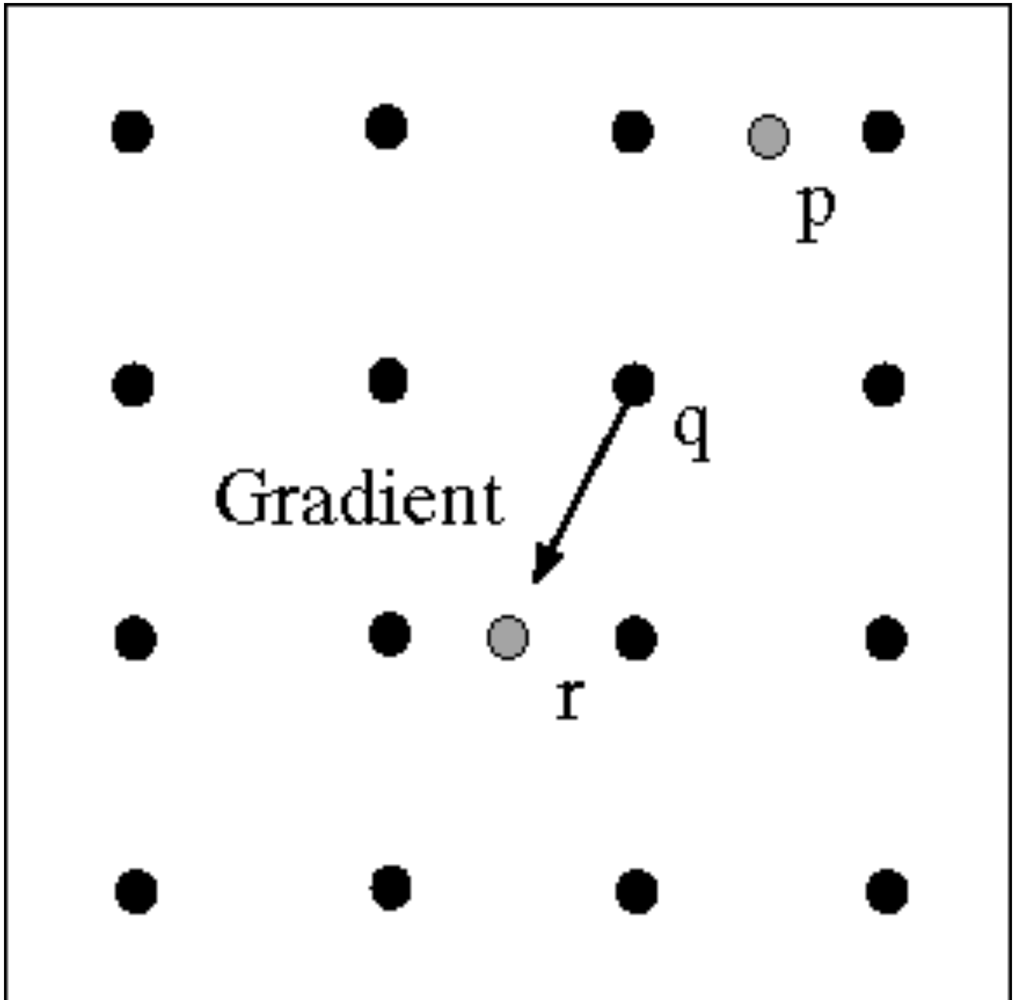
Canny edge detector

- Non-maxima 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.
  - This operation can be used with any edge operator when thin boundaries are wanted.

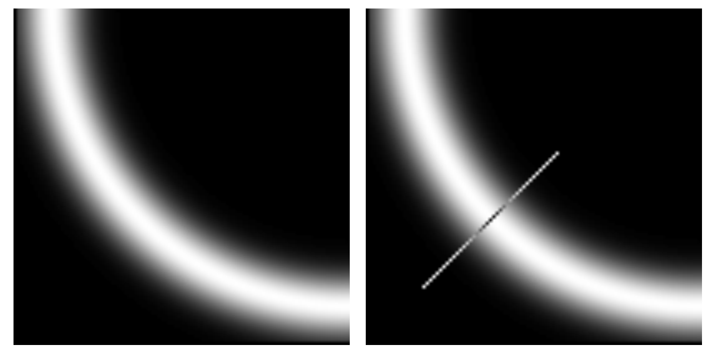




Non-maximum suppression

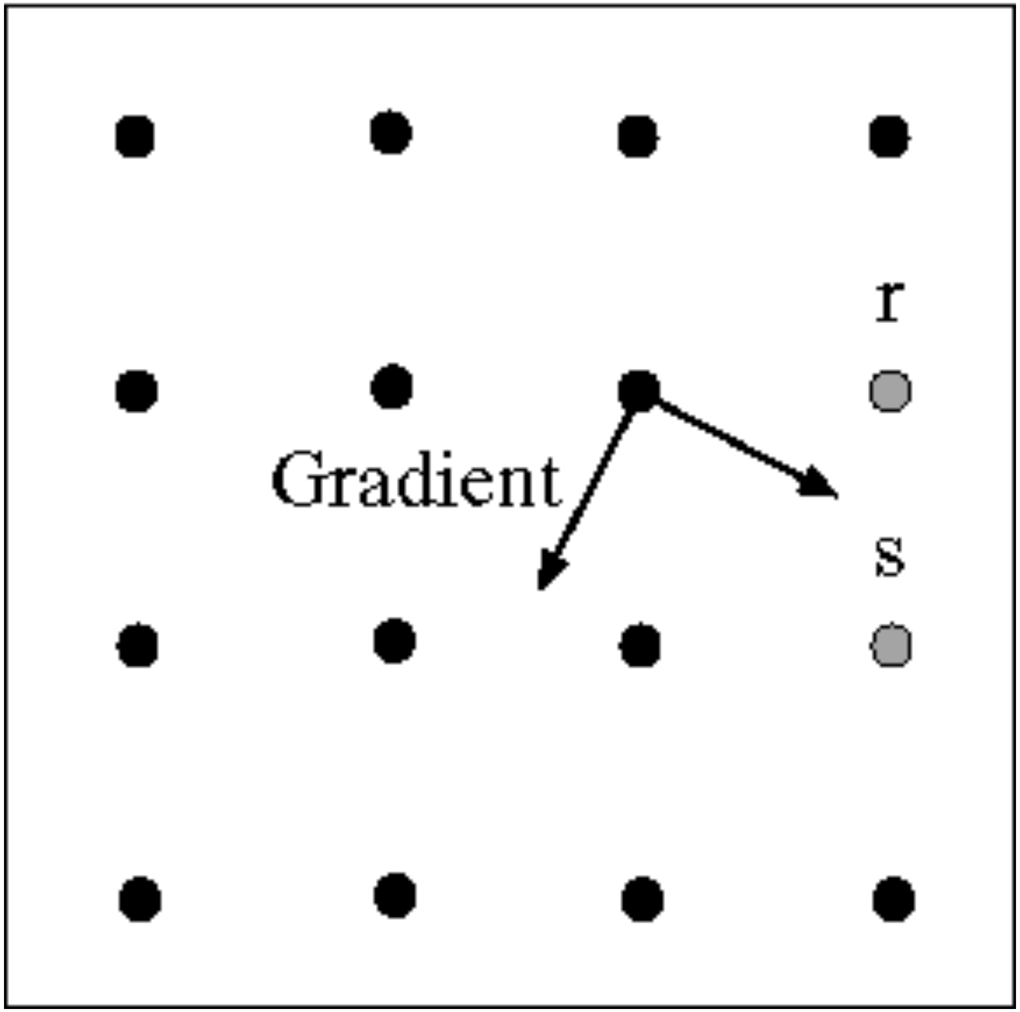


At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.

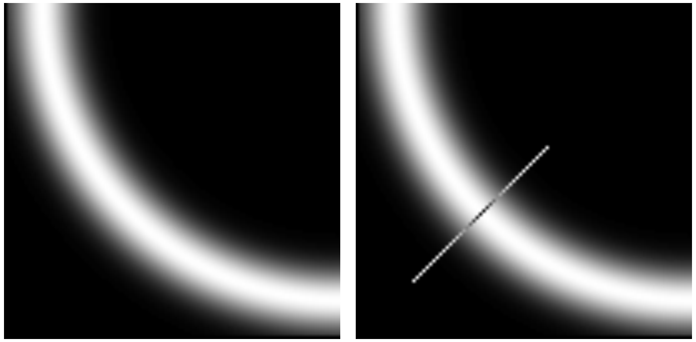


Source: D. Forsyth

Predicting the next edge point



Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s).



Source: D. Forsyth

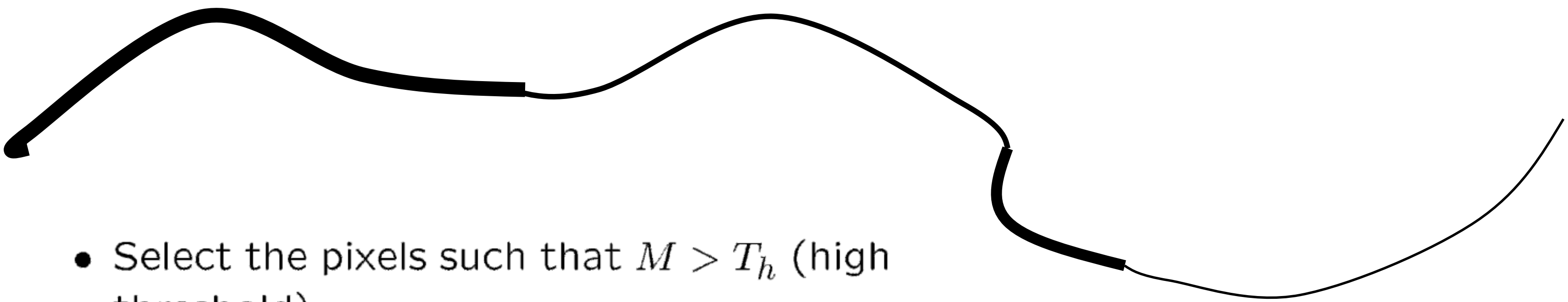
Canny edge detector

Problem: pixels along this edge did not survive the thresholding

Adapted from Steve Seitz, U of Washington

Canny edge detector

- Hysteresis thresholding:
  - Use a high threshold to start edge curves, and a low threshold to continue them.



- Select the pixels such that $M > T_h$ (high threshold)
- Collect the pixels such that $M > T_l$ (low threshold) that are neighbors of already collected edge points

Hysteresis thresholding

original image

high threshold
(strong edges)

low threshold
(weak edges)

hysteresis threshold

Source: L. Fei-Fei

Canny edge detector

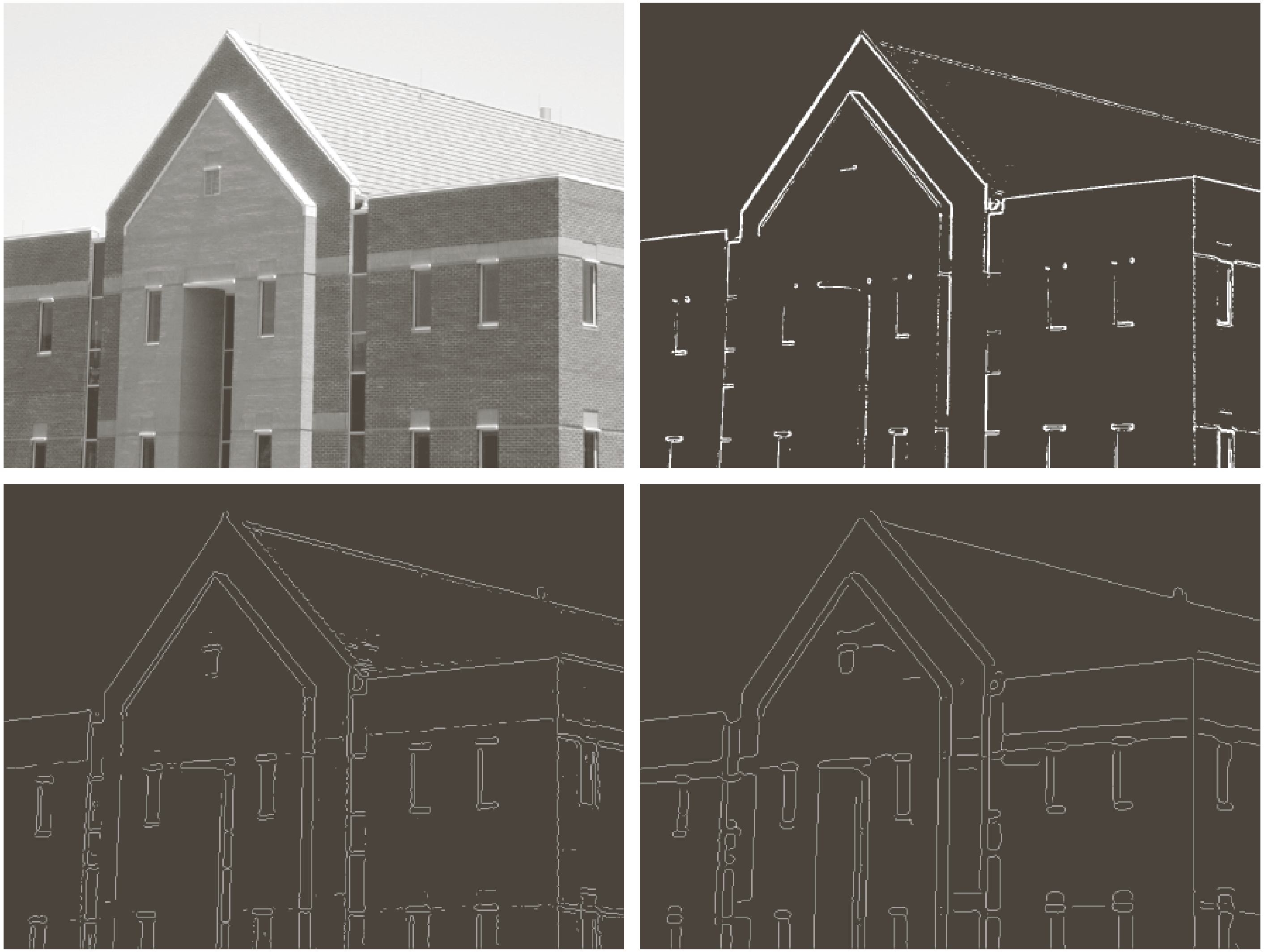


FIGURE 10.25
(a) Original image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1].
(b) Thresholded gradient of smoothed image.
(c) Image obtained using the Marr-Hildreth algorithm.
(d) Image obtained using the Canny algorithm. Note the significant improvement of the Canny image compared to the other two.

Canny edge detector

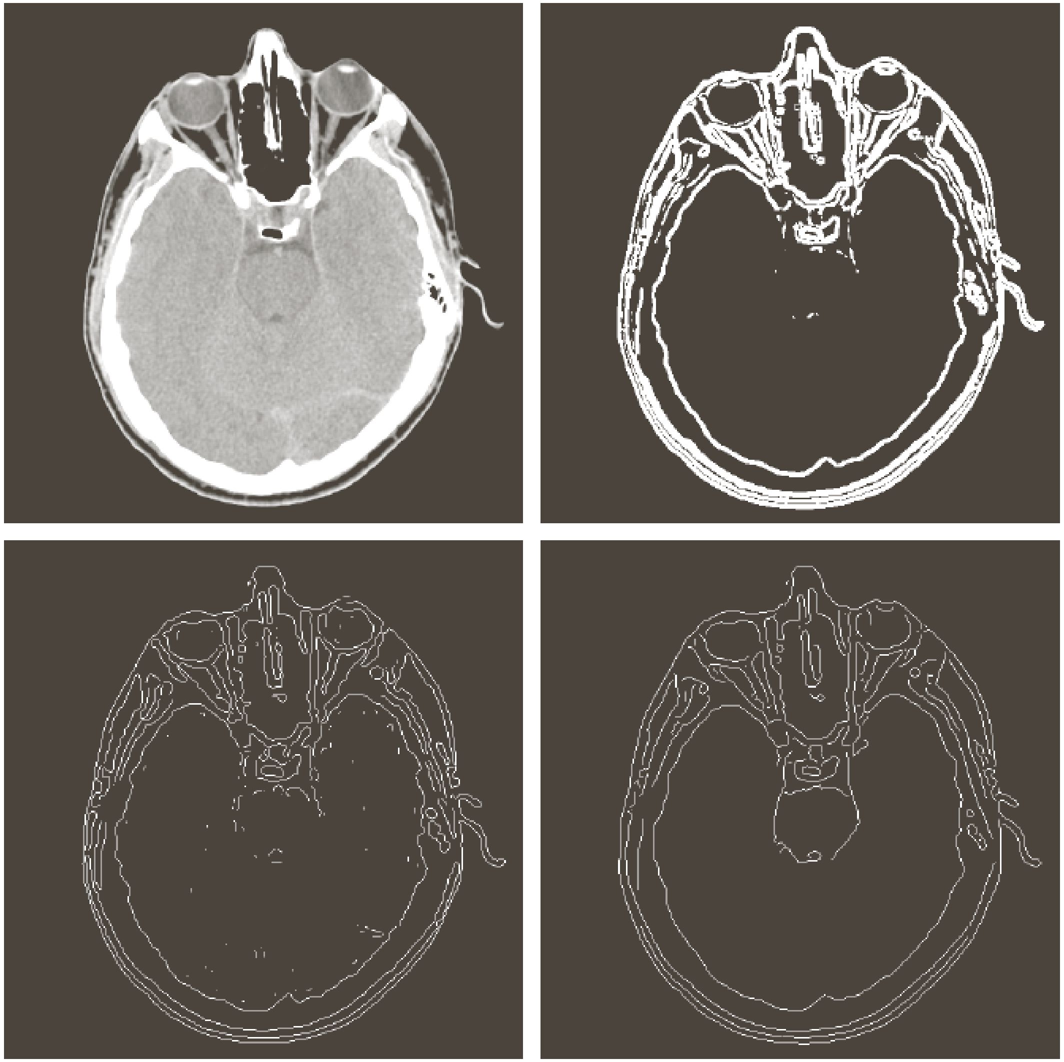


FIGURE 10.26
(a) Original head CT image of size 512 $\times$ 512 pixels, with intensity values scaled to the range [0, 1]. (b) Thresholded gradient of smoothed image. (c) Image obtained using the Marr-Hildreth algorithm. (d) Image obtained using the Canny algorithm. (Original image courtesy of Dr. David R. Pickens, Vanderbilt University.)

Edge linking

- Hough transform
  - Finding line segments
  - Finding circles
- Model fitting
  - Fitting line segments
  - Fitting ellipses
- Edge tracking

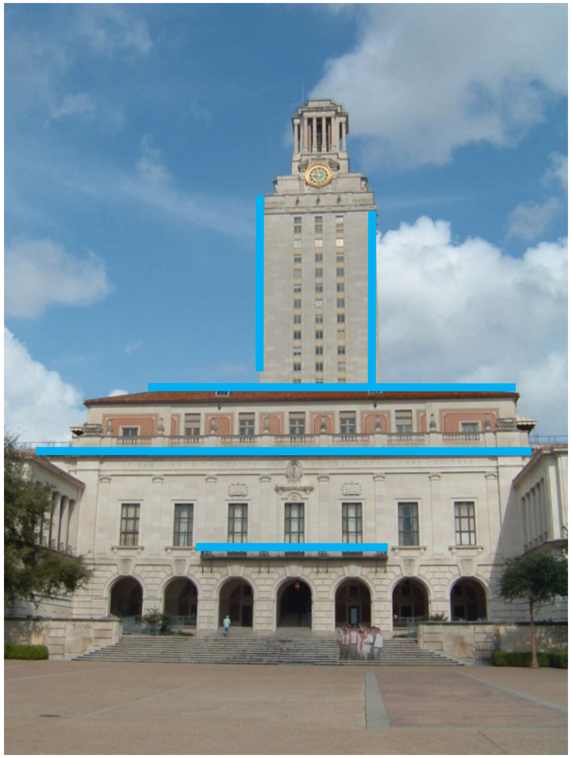
Fitting: main idea

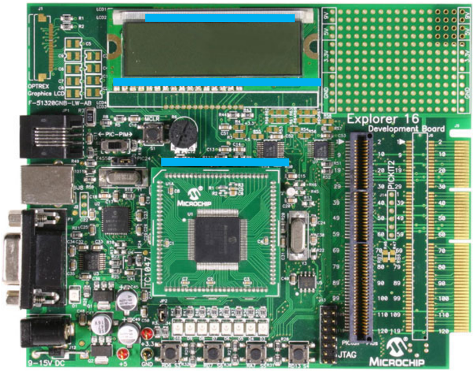
- Choose a parametric model to represent a set of features
- Membership criterion is not local
  - Cannot tell whether a point belongs to a given model just by looking at that point
- Three main questions:
  - What model represents this set of features best?
  - Which of several model instances gets which feature?
  - How many model instances are there?
- Computational complexity is important
  - It is infeasible to examine every possible set of parameters and every possible combination of features

Adapted from Kristen Grauman

Example: line fitting

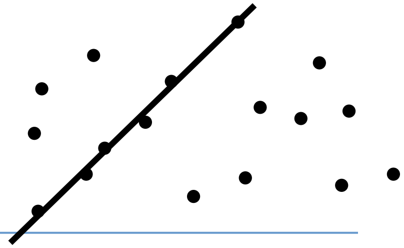
- Why fit lines?
  - Many objects characterized by presence of straight lines

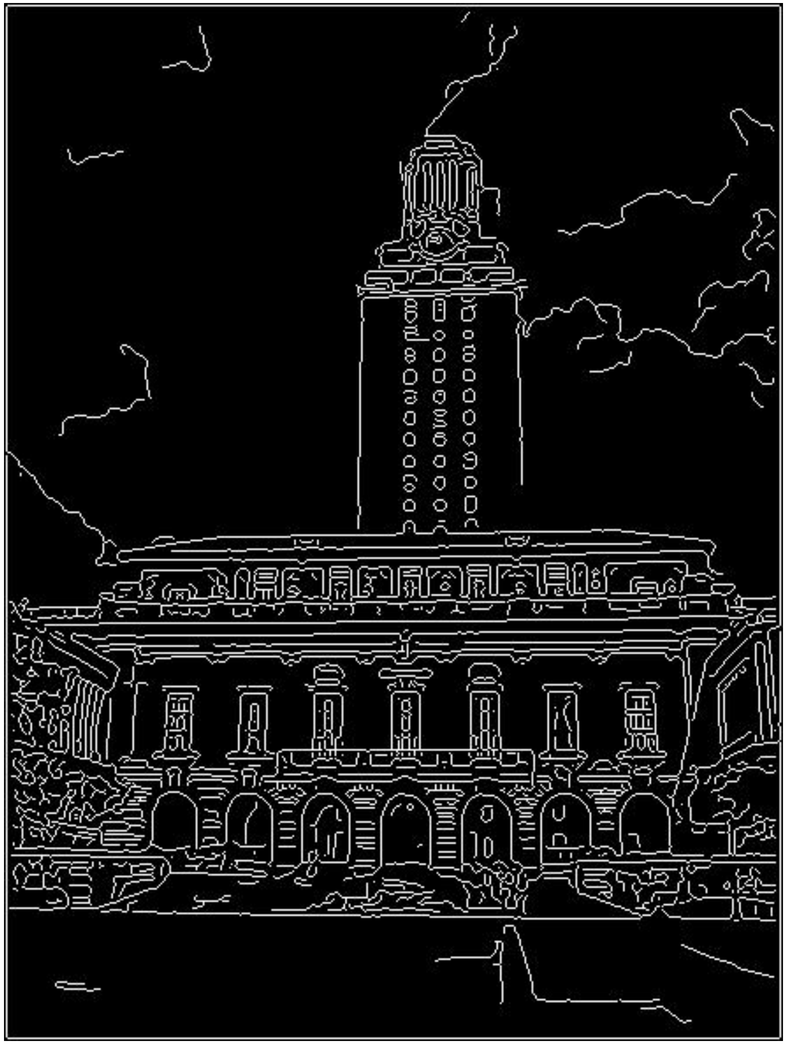




Adapted from Kristen Grauman

Difficulty of line fitting





- **Extra** edge points (clutter), multiple models:
  - which points go with which line, if any?
- Only some parts of each line detected, and some parts are **missing**:
  - how to find a line that bridges missing evidence?
- **Noise** in measured edge points, orientations:
  - how to detect true underlying parameters?

Adapted from Kristen Grauman

Voting

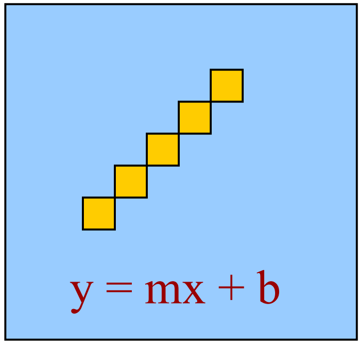
- It is not feasible to check all combinations of features by fitting a model to each possible subset.
- Voting is a general technique where we let each feature vote for all models that are compatible with it.
  - Cycle through features, cast votes for model parameters.
  - Look for model parameters that receive a lot of votes.
- Noise and clutter features will cast votes too, but typically their votes should be inconsistent with the majority of "good" features.

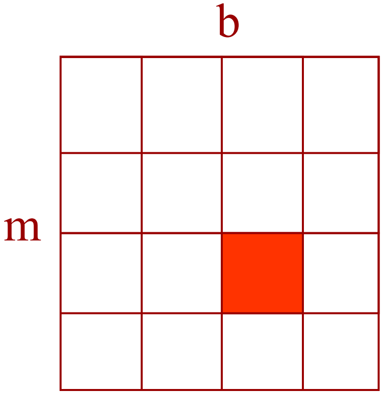
Adapted from Kristen Grauman

Hough transform

- The Hough transform is a method for detecting lines or curves specified by a parametric function.
- If the parameters are p_1, p_2, … p_n, then the Hough procedure uses an n-dimensional accumulator array in which it accumulates votes for the correct parameters of the lines or curves found on the image.

image





accumulator

Adapted from Linda Shapiro, U of Washington

Hough transform: line segments

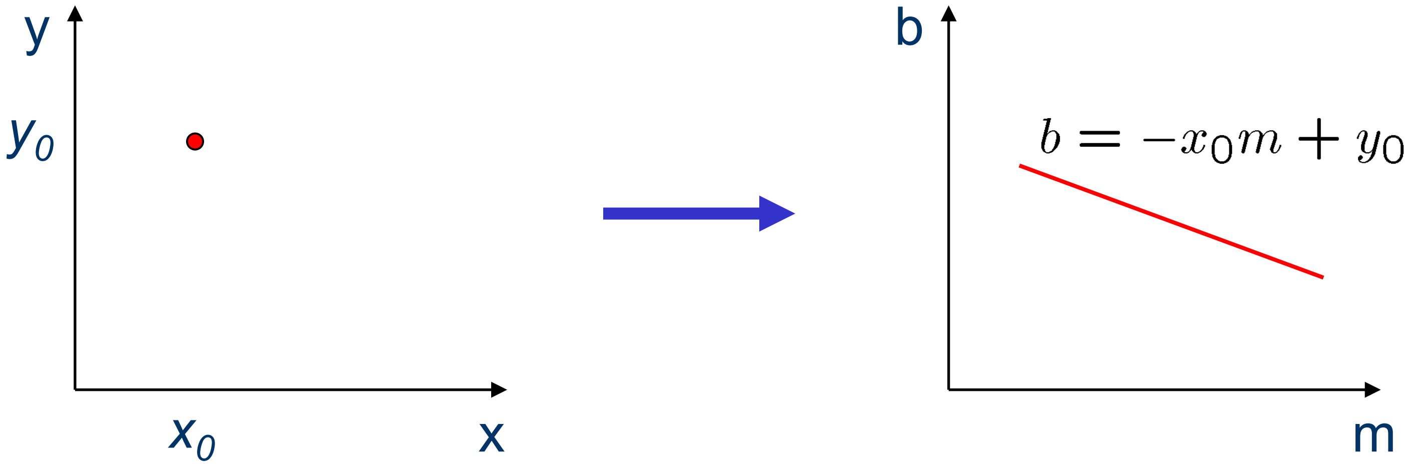


Image space

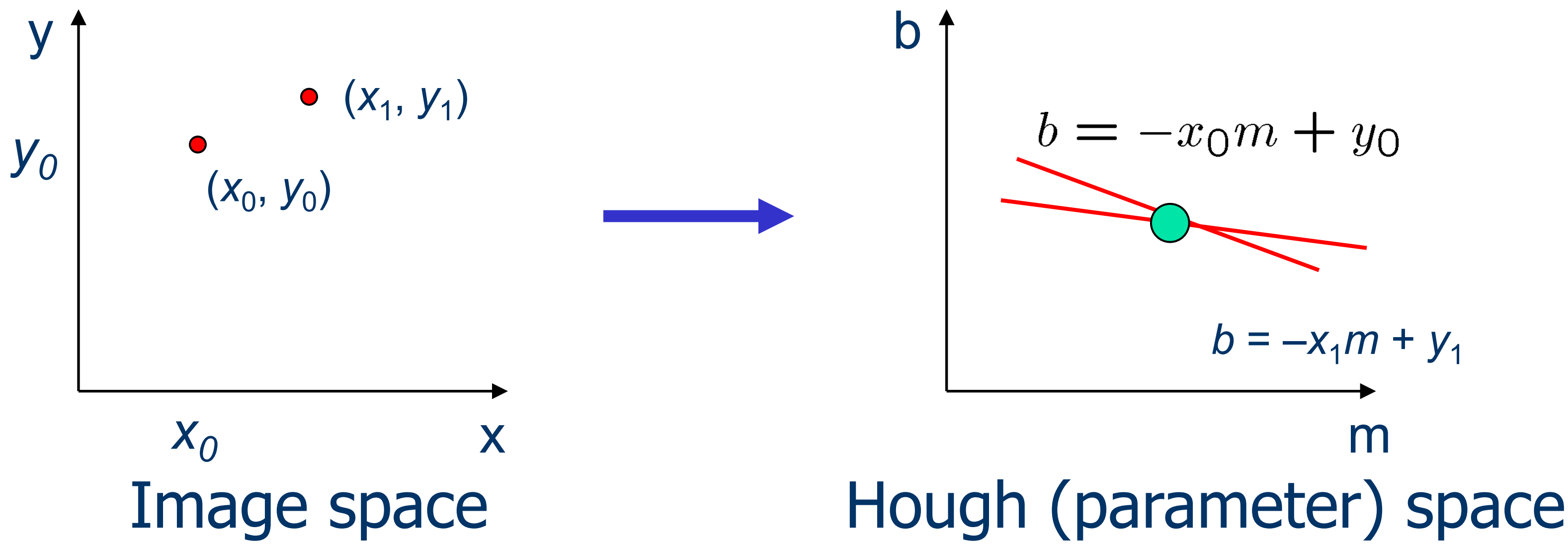
Hough (parameter) space

Connection between image (x,y) and Hough (m,b) spaces

- A line in the image corresponds to a point in Hough space
- To go from image space to Hough space:
  - given a set of points (x,y), find all (m,b) such that y = mx + b
- What does a point (x_0, y_0) in the image space map to?
  - Answer: the solutions of $b = -x_0 m + y_0$
  - This is a line in Hough space

Adapted from Steve Seitz, U of Washington

Hough transform: line segments



What are the line parameters for the line that contains both (x_0, y_0) and (x_1, y_1)?

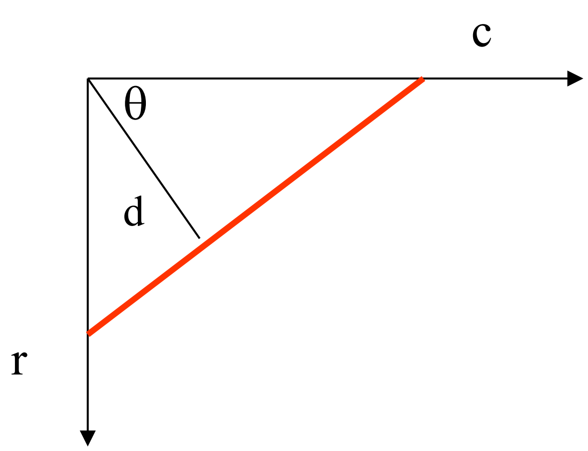
- It is the intersection of the lines $b = -x_0 m + y_0$ and $b = -x_1 m + y_1$

Adapted from Steve Seitz, U of Washington

Hough transform: line segments

- y = mx + b is not suitable (why?)
- The equation generally used is:

$$d = r \sin(\theta) + c \cos(\theta).$$



d is the distance from the line to origin.

θ is the angle the perpendicular makes with the column axis.

Adapted from Linda Shapiro, U of Washington

Hough transform: line segments

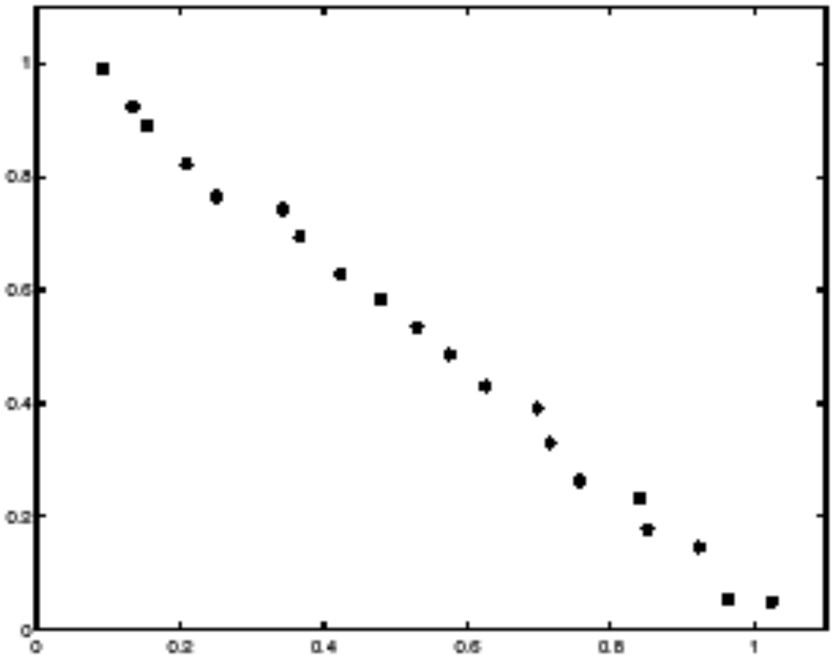
Accumulate the straight line segments in gray-tone image S to accumulator A.
S[**R**, **C**] is the input gray-tone image.
NLINES is the number of rows in the image.
NPIXELS is the number of pixels per row.
A[**DQ**, **THETAQ**] is the accumulator array.
DQ is the quantized distance from a line to the origin.
THETAQ is the quantized angle of the normal to the line.

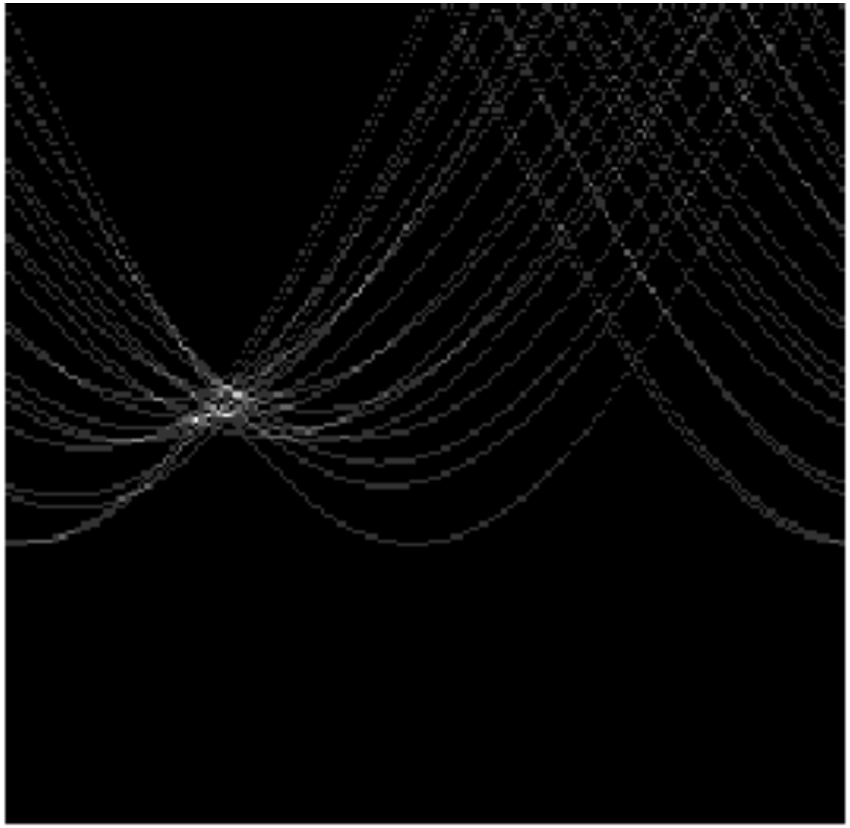
```
procedure accumulate_lines(S,A);
{
A := 0;
PTLIST := NIL;
for R := 1 to NLINES
   for C := 1 to NPIXELS
      {
      DR := row_gradient(S,R,C);
      DC := col_gradient(S,R,C);
      GMAG := gradient(DR,DC);
      if GMAG > gradient_threshold
         {
         THETA := atan2(DR,DC);
         THETAQ := quantize_angle(THETA);
         D := abs(C*cos(THETAQ) - R*sin(THETAQ));
         DQ := quantize_distance(D);
         A[DQ,THETAQ] := A[DQ,THETAQ]+GMAG;
         PTLIST(DQ,THETAQ) := append(PTLIST(DQ,THETAQ),[R,C])
         }
      }
}
```

Algorithm 7: Hough Transform for Finding Straight Lines

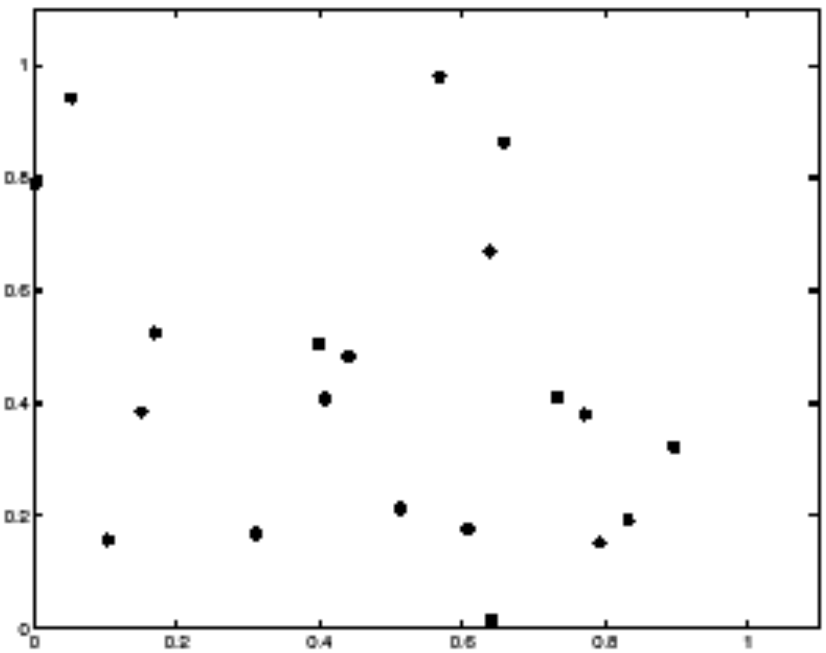
Adapted from Shapiro and Stockman

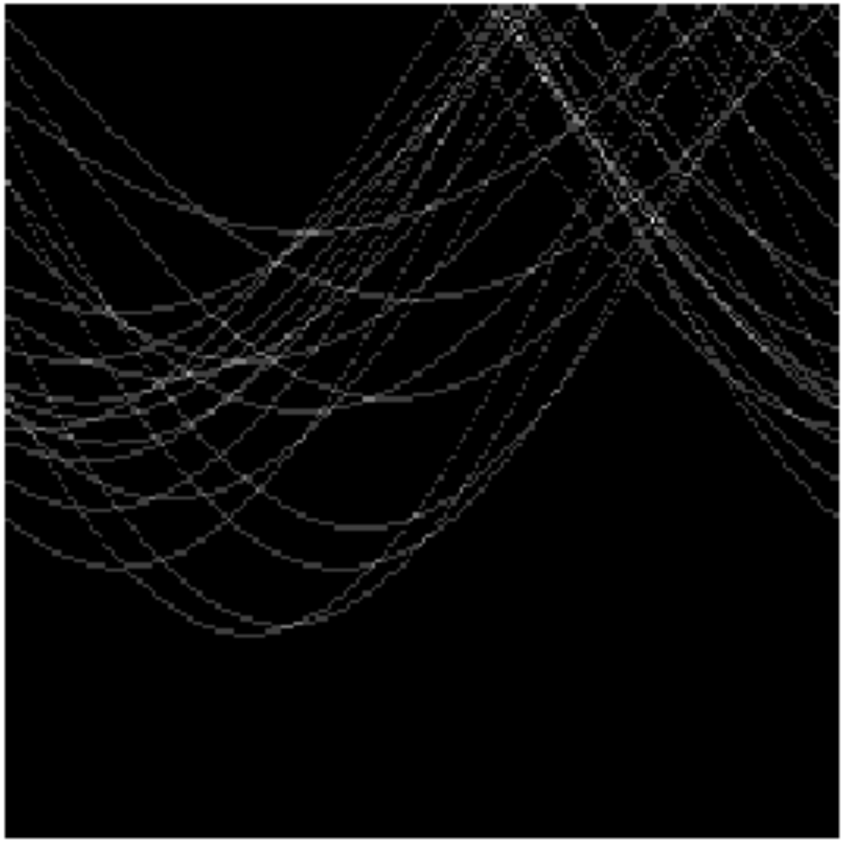
Hough transform: line segments





Hough transform: line segments





Hough transform: line segments

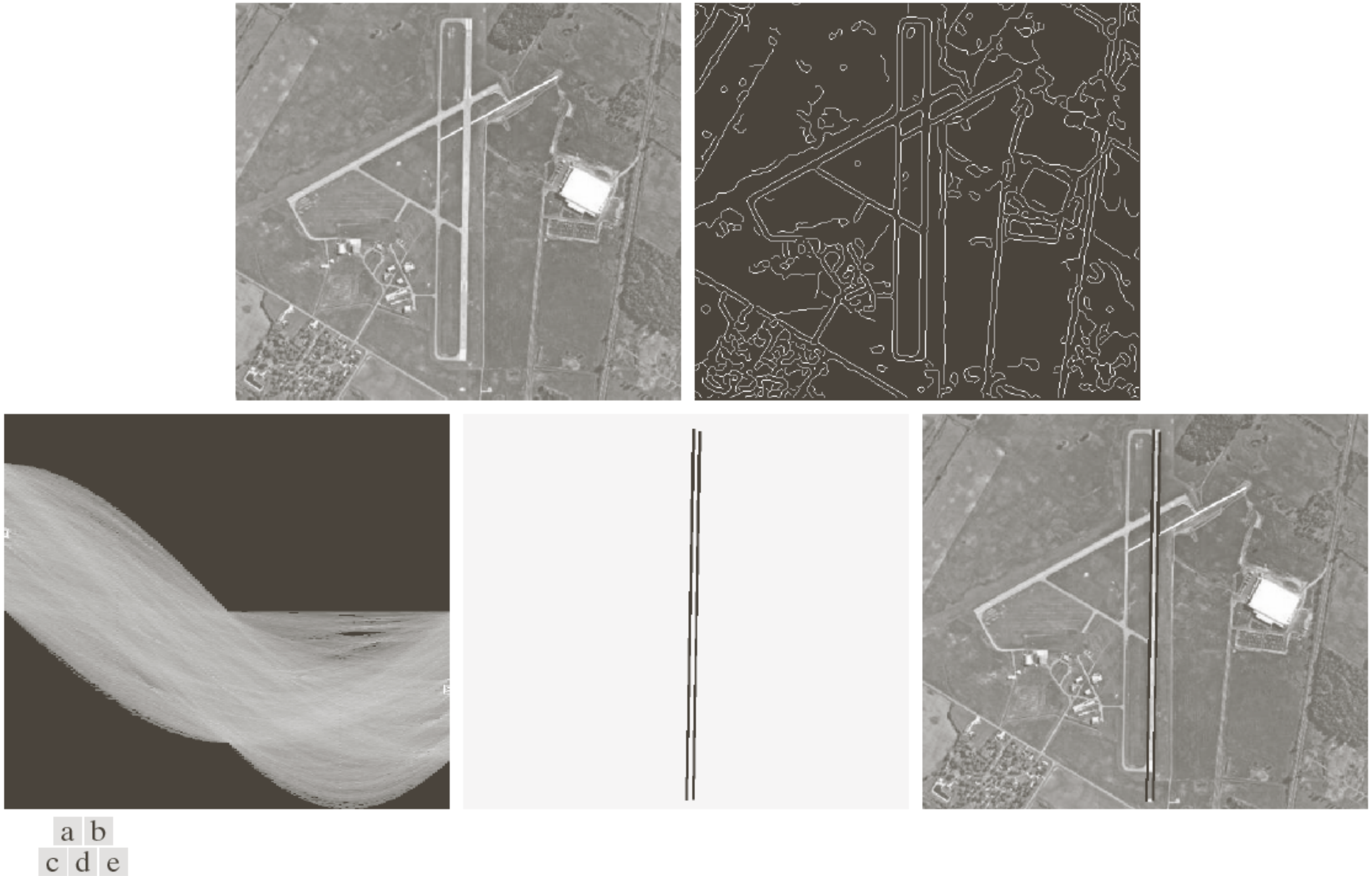


FIGURE 10.34 (a) A 502 × 564 aerial image of an airport. (b) Edge image obtained using Canny's algorithm. (c) Hough parameter space (the boxes highlight the points associated with long vertical lines). (d) Lines in the image plane corresponding to the points highlighted by the boxes). (e) Lines superimposed on the original image.

Hough transform: circles

- Circle detection

- Circle equation:

$$(x-a)^2 + (y-b)^2 = r^2$$

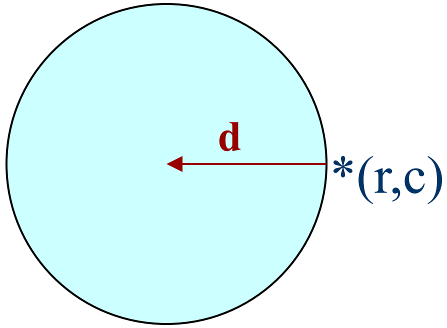
- the point (a,b) is the center of the circle.

- 3 parameters: a, b, r
- If you are looking for circles with known r, than you have only two parameters!

Adapted from Linda Shapiro, U of Washington

Hough transform: circles

- Main idea: The gradient vector at an edge pixel passes from the center of the circle.
- Circle equations:
  - $r = r_0 + d \sin(\theta)$ r_0, c_0, d are parameters
  - $c = c_0 + d \cos(\theta)$



Adapted from Linda Shapiro, U of Washington

Hough transform: circles

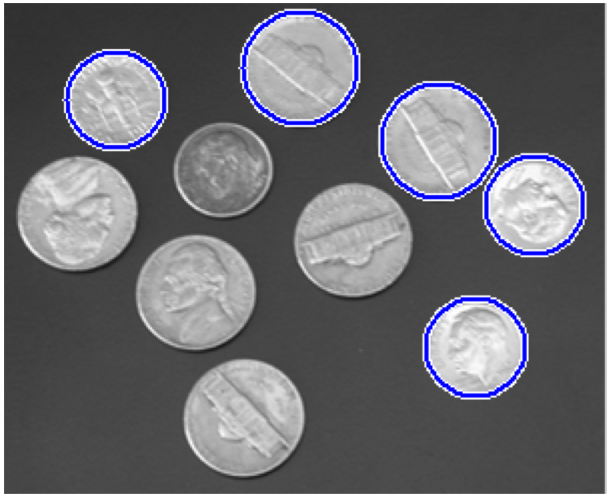
Accumulate the circles in gray-tone image S to accumulator A.
S[**R**, **C**] is the input gray-tone image.
NLINES is the number of rows in the image.
NPIXELS is the number of pixels per row.
A[**R**, **C**, **RAD**] is the accumulator array.
R is the row index of the circle center.
C is the column index of the circle center.
RAD is the radius of the circle.

```
procedure accumulate_circles(S,A);
{
A := 0;
PTLIST := 0;
for R := 1 to NLINES
  for C := 1 to NPIXELS
    for each possible value RAD of radius
      {
      THETA := compute_theta(S,R,C,RAD);
      R0 := R - RAD*cos(THETA);
      C0 := C - RAD*sin(THETA);
      A[R0,C0,RAD] := A[R0,C0,RAD]+1;
      PTLIST(R0,C0,RAD) := append(PTLIST(R0,C0,RAD),[R,C])
      }
}
```

Adapted from Shapiro and Stockman

Algorithm 9: Hough Transform for Finding Circles

Hough transform: circles

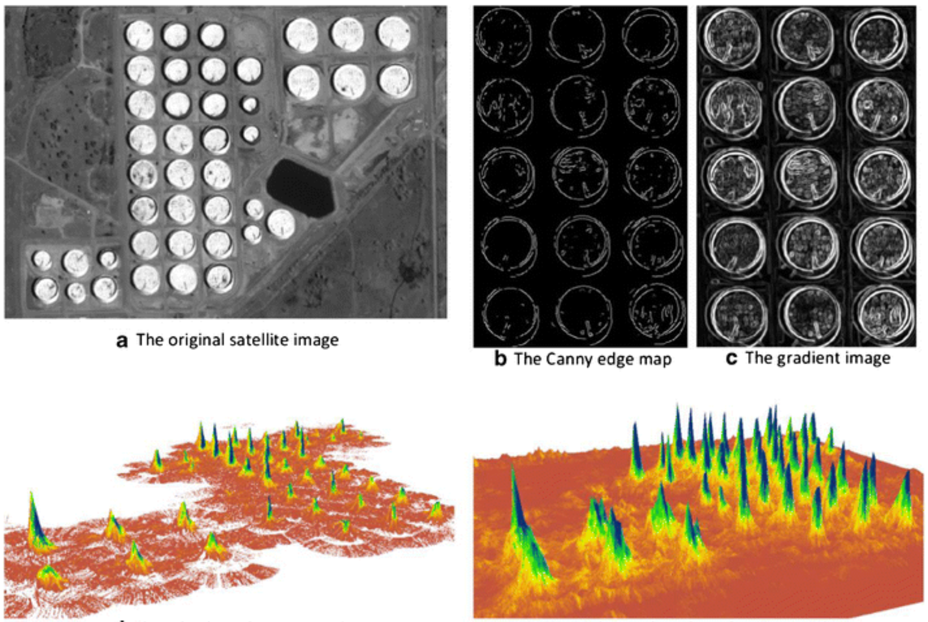


https://www.mathworks.com/help/images/ref/imfindcircles.html

http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_circle/hough_circle.html

http://shreshai.blogspot.com.tr/2015/01/matlab-tutorial-finding-center-pivot.html

Hough transform: circles



a The original satellite image

b The Canny edge map

c The gradient image

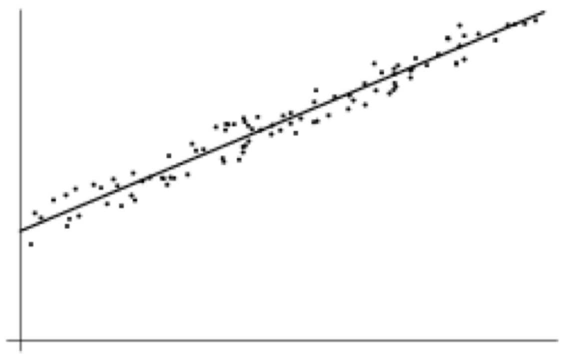
d The edge based voting result

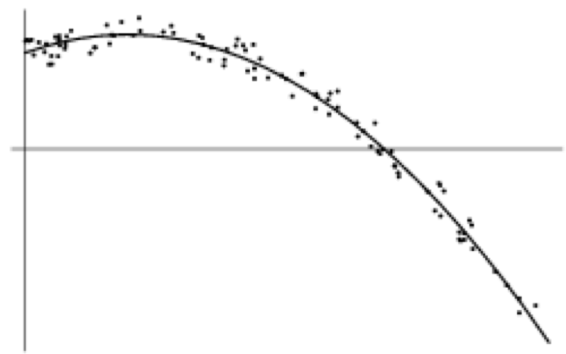
e The gradient based voting result

Zhao et al., "Oil Tanks Extraction from High Resolution Imagery Using a Directional and Weighted Hough Voting Method", Journal of the Indian Society of Remote Sensing, September 2015

Model fitting

- Mathematical models that fit data not only reveal important structure in the data, but also can provide efficient representations for further analysis.
- Mathematical models exist for lines, circles, cylinders, and many other shapes.
- We can use the method of least squares for determining the parameters of the best mathematical model fitting the observed data.





Model fitting: line segments

Adapted from Martial Hebert, CMU

Model fitting: line segments

- Given a set of observed points $\{(x_i, y_i), i = 1, \ldots, n\}$.
- A straight line can be modeled as a function with two parameters:

$$y = ax + b.$$

- To measure how well a model fits a set of n observations can be computed using the *least-squares error criteria*:

$$LSE = \sum_{i=1}^{n} (ax_i + b - y_i)^2$$

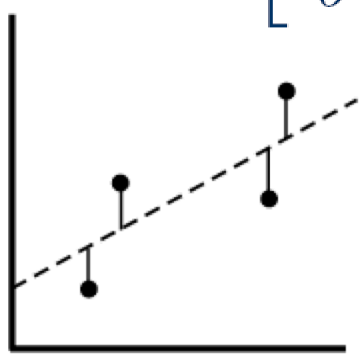
where $ax_i + b - y_i$ is the algebraic distance.

- The best model is the model with the paramaters minimizing this criteria.

Model fitting: line segments

- For the model $y = ax + b$, the parameters that minimize LSE can be found by taking partial derivatives and solving for the unknowns.
- The parameters of the best line are:

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} 1 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} y_i \end{bmatrix}.$$



vertical offsets

Model fitting: line segments

- Problems in fitting:
  - Outliers
  - Error definition (algebraic vs. geometric distance)
  - Statistical interpretation of the error (hypothesis testing)
  - Nonlinear optimization
  - High dimensionality (of the data and/or the number of model parameters)
  - Additional fit constraints

Model fitting: ellipses

- Fitting a general conic represented by a second-order polynomial

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

  can be approached by minimizing the sum of squared algebraic distances.

- See Fitzgibbon *et al.* (PAMI 1999) for an algorithm that constrains the parameters so that the conic representation is forced to be an ellipse.

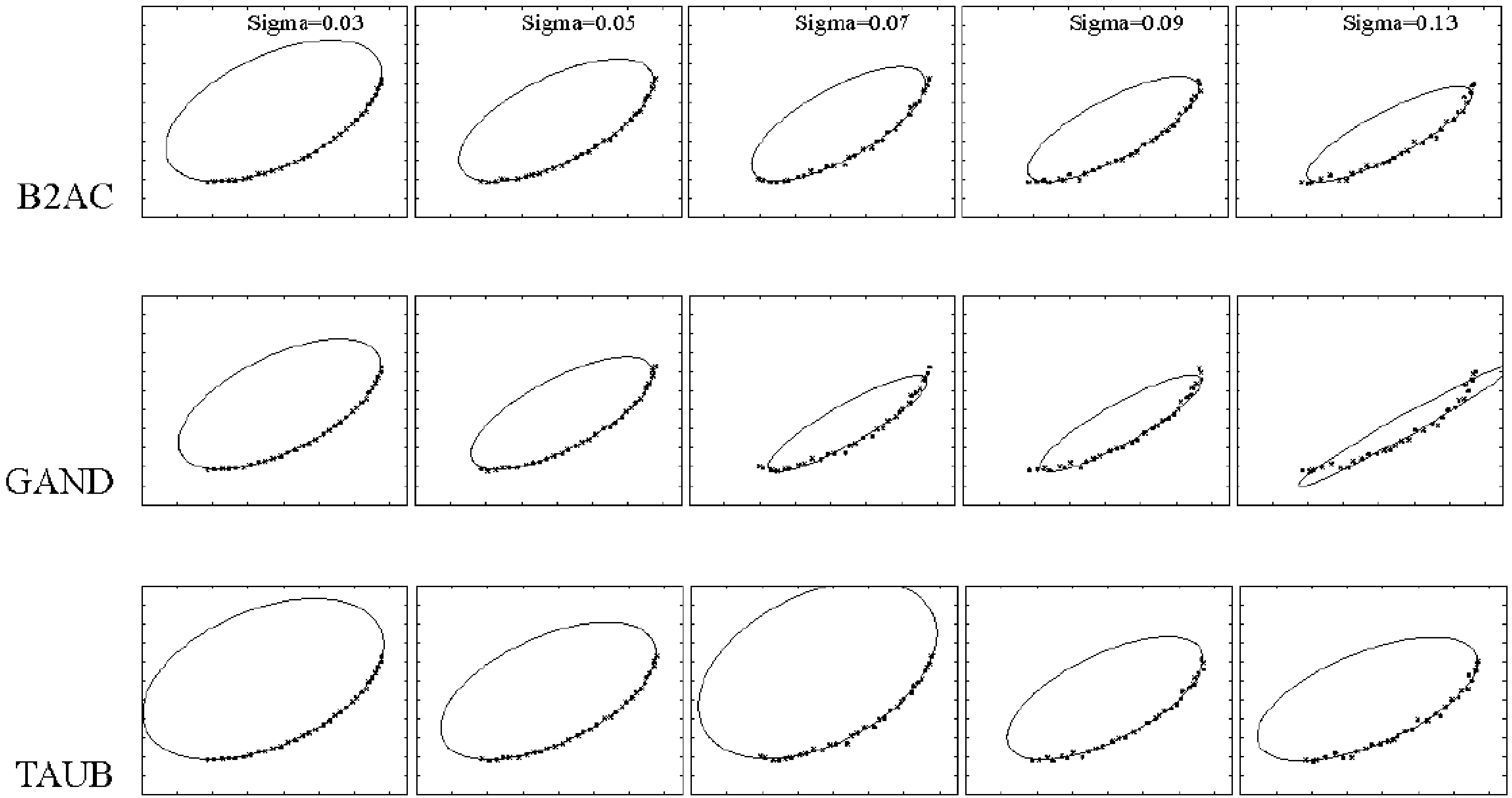
Model fitting: ellipses

```
% x,y are vectors of coordinates
function  a=fit_ellipse(x,y)
% Build design matrix
  D = [ x.*x x.*y y.*y x y ones(size(x)) ];
% Build scatter matrix
  S = D'*D;
% Build 6x6 constraint matrix
  C(6,6)=0; C(1,3)=-2; C(2,2)=1; C(3,1)=-2;
% Solve generalised eigensystem
  [gevec, geval] = eig(S,C);
% Find the only negative eigenvalue
  [NegR, NegC] = find(geval<0 & ~isinf(geval));
% Get fitted  parameters
  a = gevec(:,NegC);
```

Simple six-line Matlab implementation of the ellipse fitting method.

Adapted from Andrew Fitzgibbon, PAMI 1999

Model fitting: ellipses

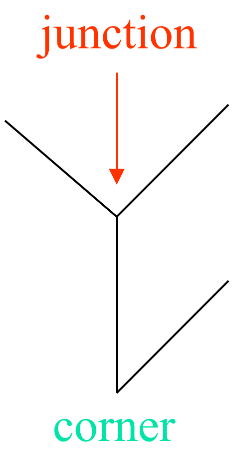


Fits to arc of ellipse with increasing noise level.

Adapted from Andrew Fitzgibbon, PAMI 1999

Edge tracking

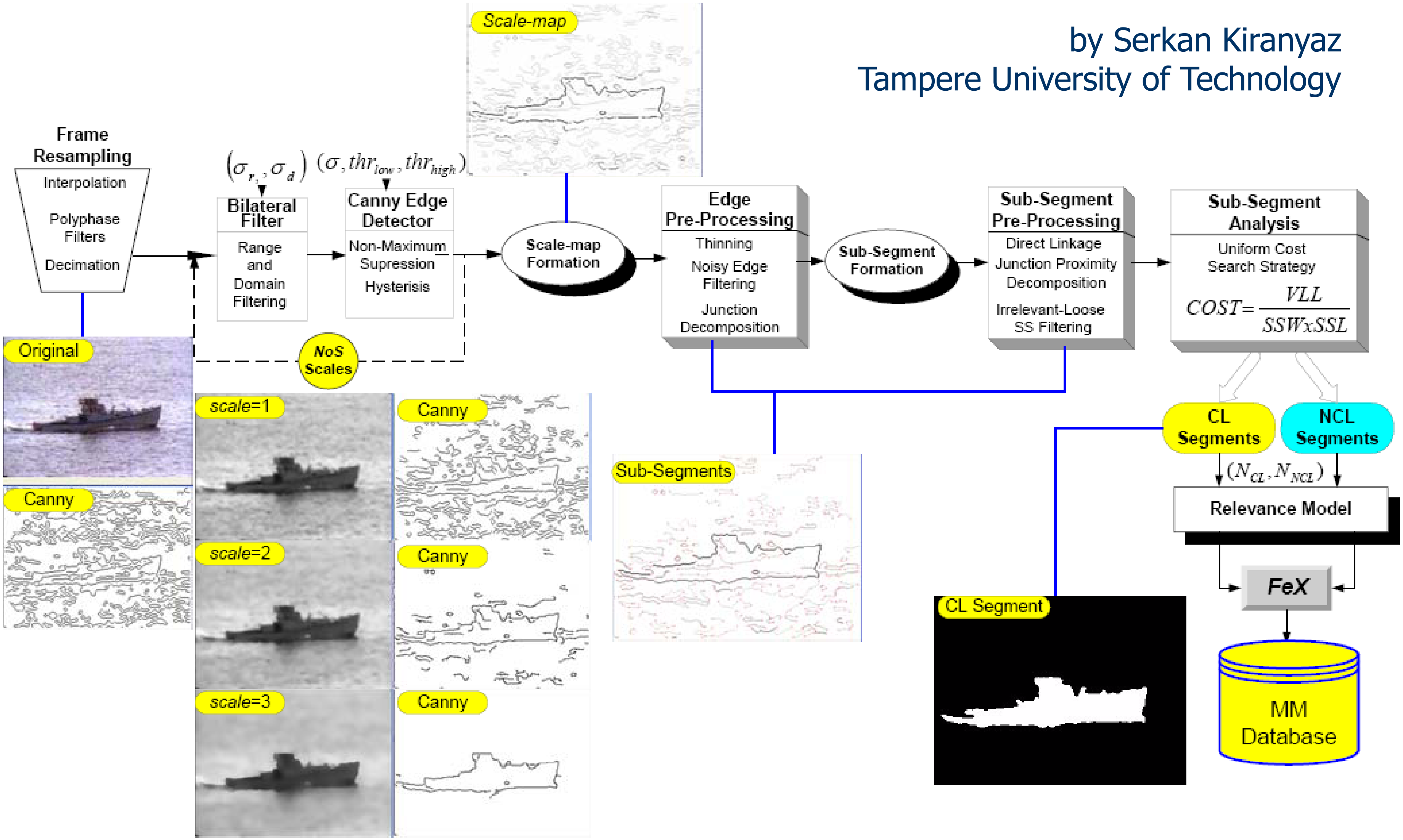
- Mask-based approach uses masks to identify the following events:
  - start of a new segment,
  - interior point continuing a segment,
  - end of a segment,
  - junction between multiple segments,
  - corner that breaks a segment into two.



Adapted from Linda Shapiro, U of Washington

Example: object extraction

by Serkan Kiranyaz
Tampere University of Technology



Example: object extraction

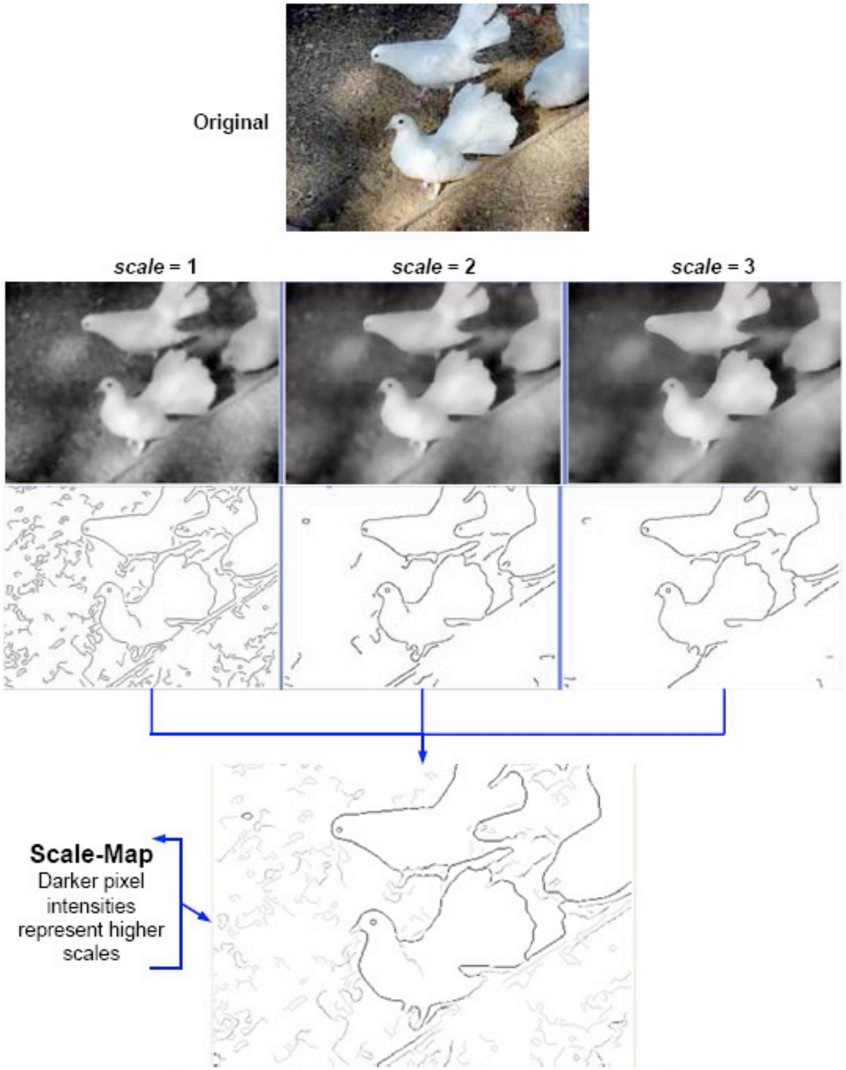


Figure 5: A sample scale-map formation.

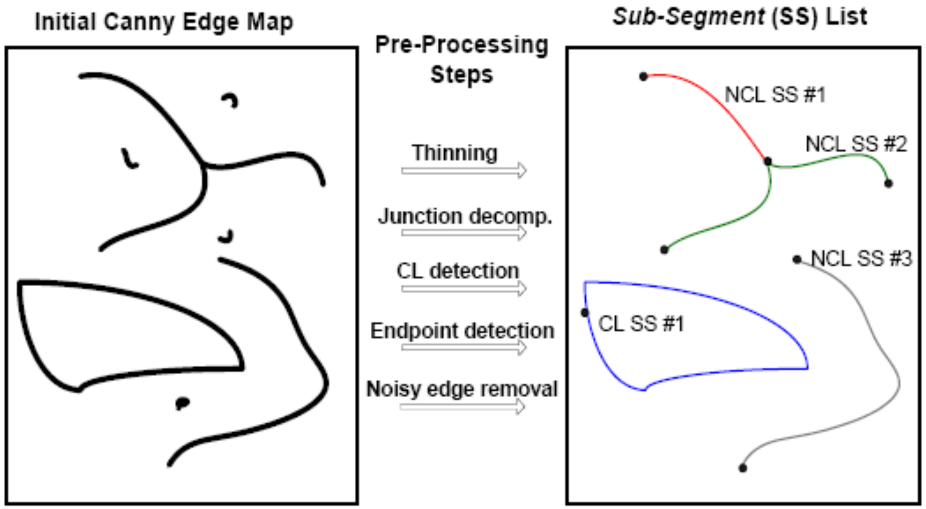


Figure 6: Sub-segment formation from an initial Canny edge field.

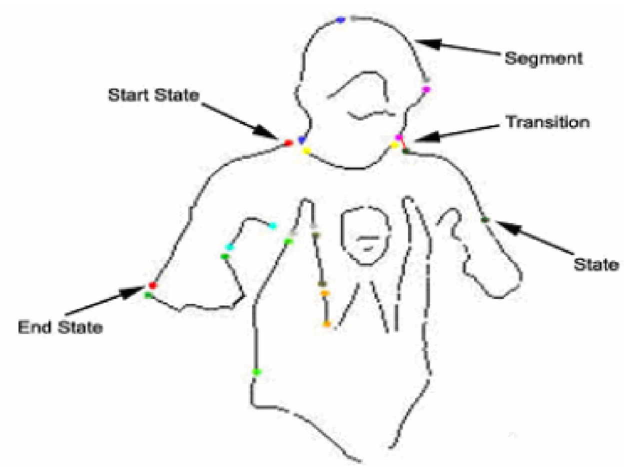


Figure 9: State space for a given sub-segment layout.

Example: object extraction

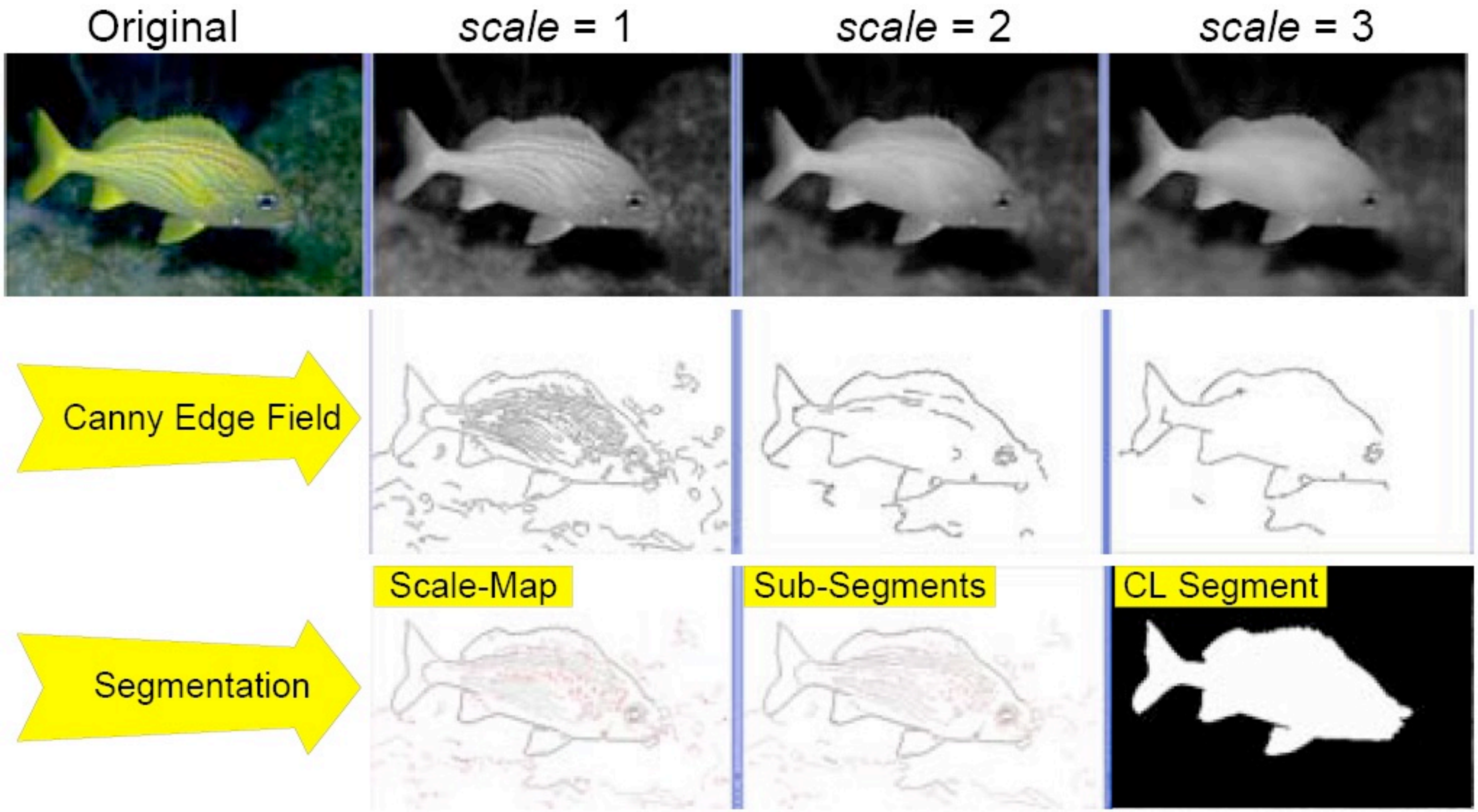
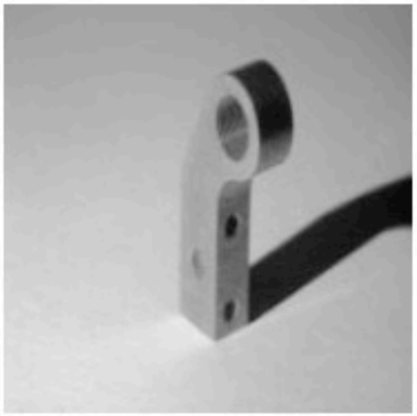


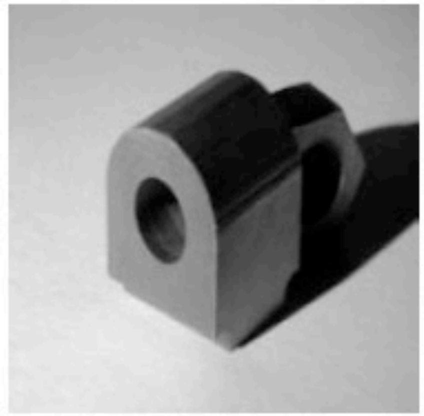
Figure 12: 3-scale simplification process over a natural image and the final CL segment extracted.

Example: object recognition

- Mauro Costa’s dissertation at the University of Washington for recognizing 3D objects having planar, cylindrical, and threaded surfaces:
  - Detects edges from two intensity images.
  - From the edge image, finds a set of high-level features and their relationships.
  - Hypothesizes a 3D model using relational indexing.
  - Estimates the pose of the object using point pairs, line segment pairs, and ellipse/circle pairs.
  - Verifies the model after projecting to 2D.

Example: object recognition

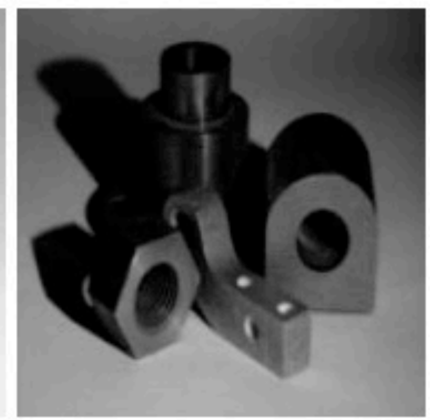

(d) Image 4 (left)


(e) Image 5 (left)

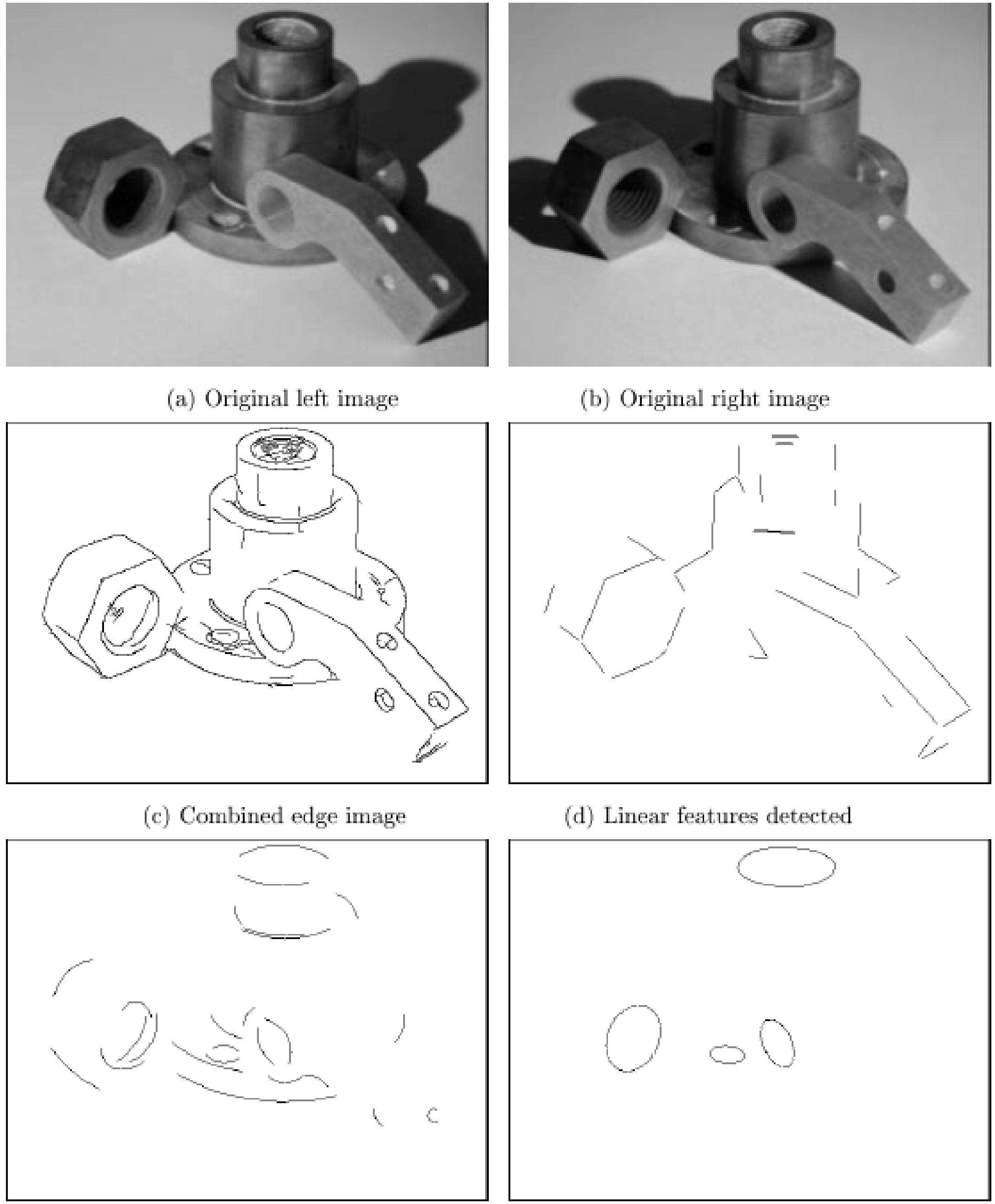

(f) Image 6 (right)


(g) Image 7 (left)


(h) Image 8 (right)


(i) Image 9 (right)

Example scenes used. The labels “left” and “right” indicate the direction of the light source.



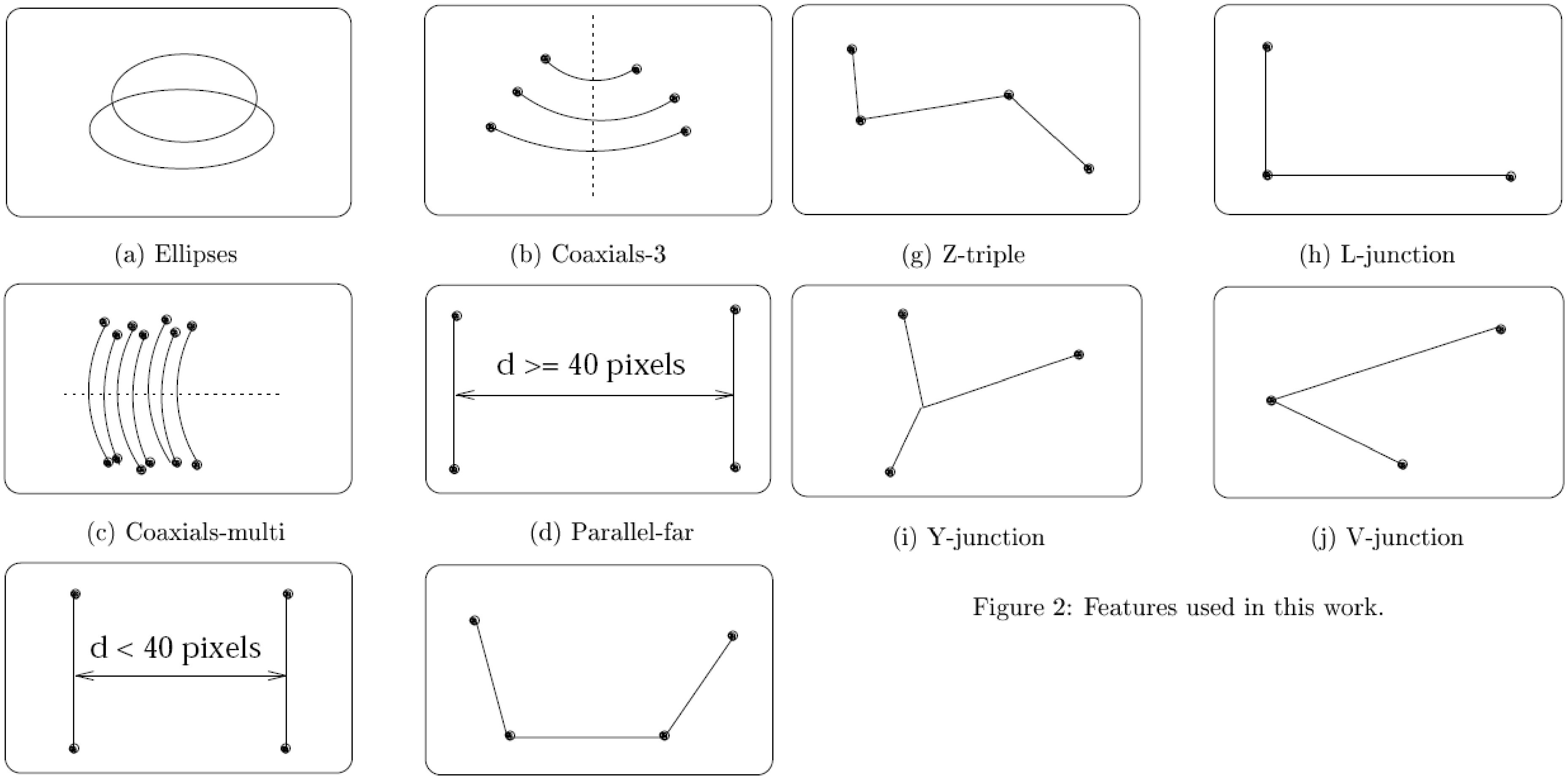
(a) Original left image (b) Original right image

(c) Combined edge image (d) Linear features detected

(e) Circular arc features detected (f) Ellipses detected

Figure 22: Sample run of the system.

Example: object recognition



(a) Ellipses (b) Coaxials-3 (g) Z-triple (h) L-junction

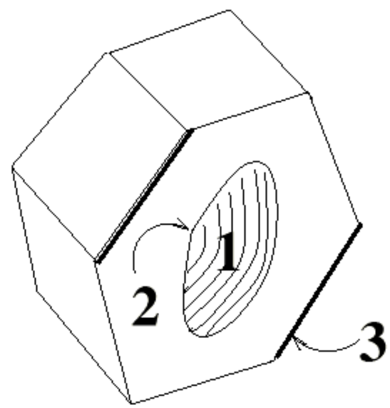
(c) Coaxials-multi (d) Parallel-far (i) Y-junction (j) V-junction

(e) Parallel-close (f) U-triple

Figure 2: Features used in this work.

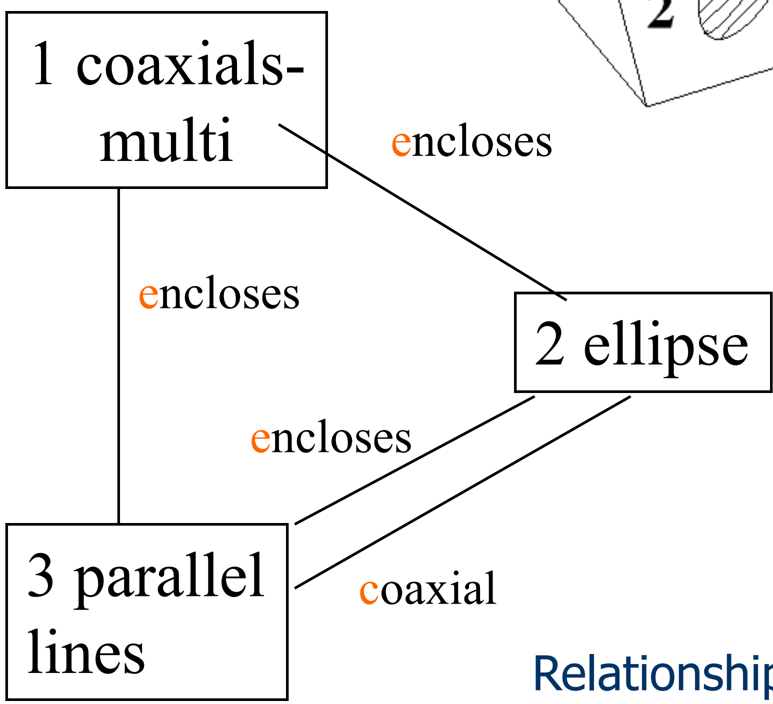
Example: object recognition

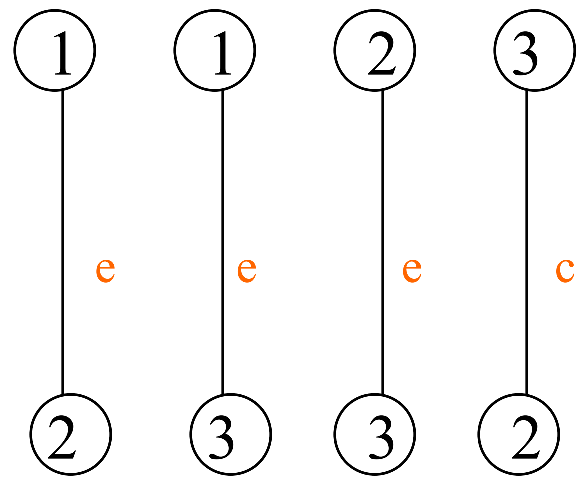
MODEL-VIEW



RELATIONS:
a: encloses
b: coaxial

FEATURES:
1: coaxials-multi
2: ellipse
3: parallel lines

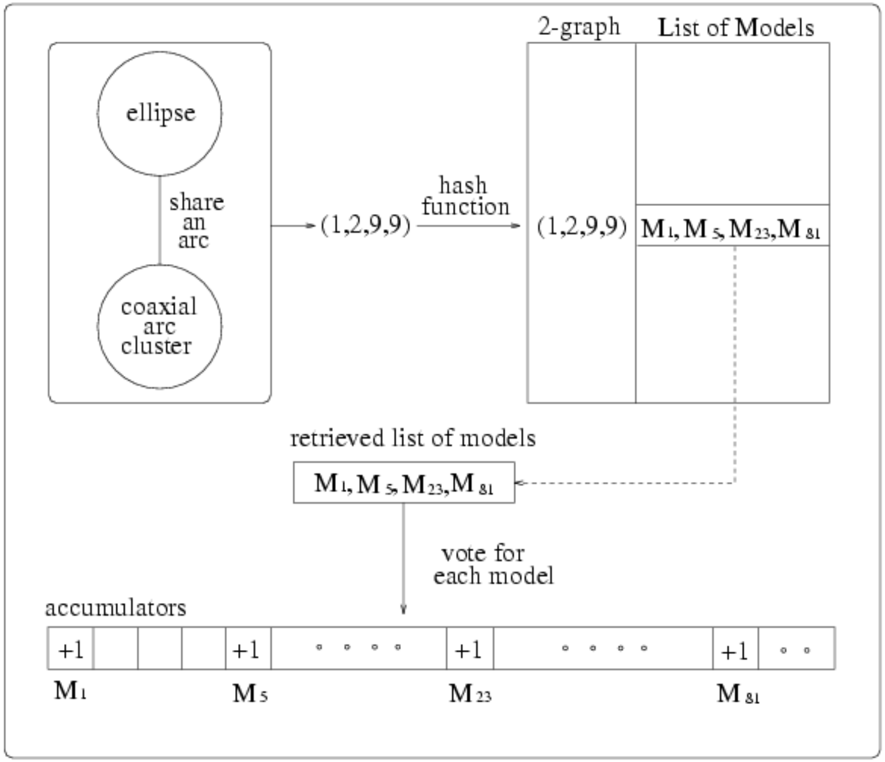




Relationship graph and the corresponding 2-graphs.

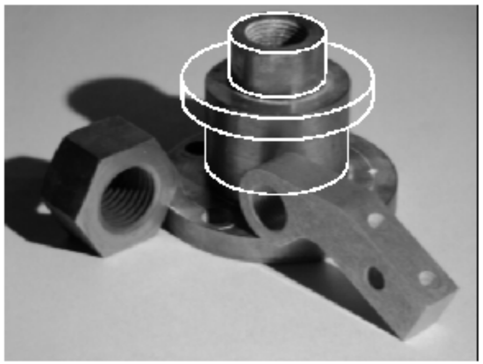
Example: object recognition

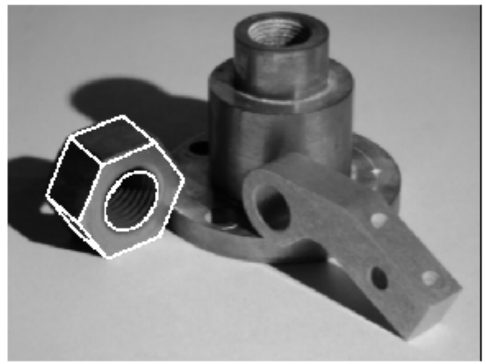
- Learning phase: relational indexing by encoding each 2-graph and storing in a hash table.
- Matching phase: voting by each 2-graph observed in the image.

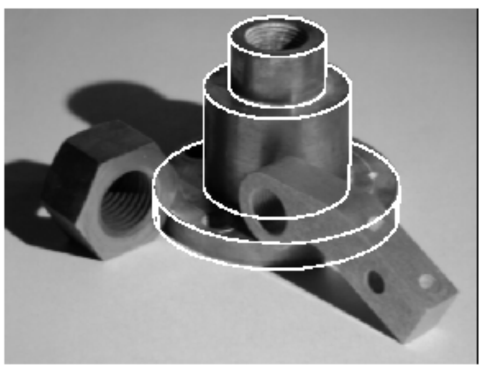


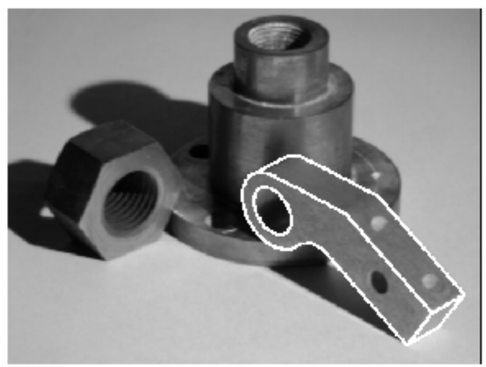
Example: object recognition

Incorrect hypothesis









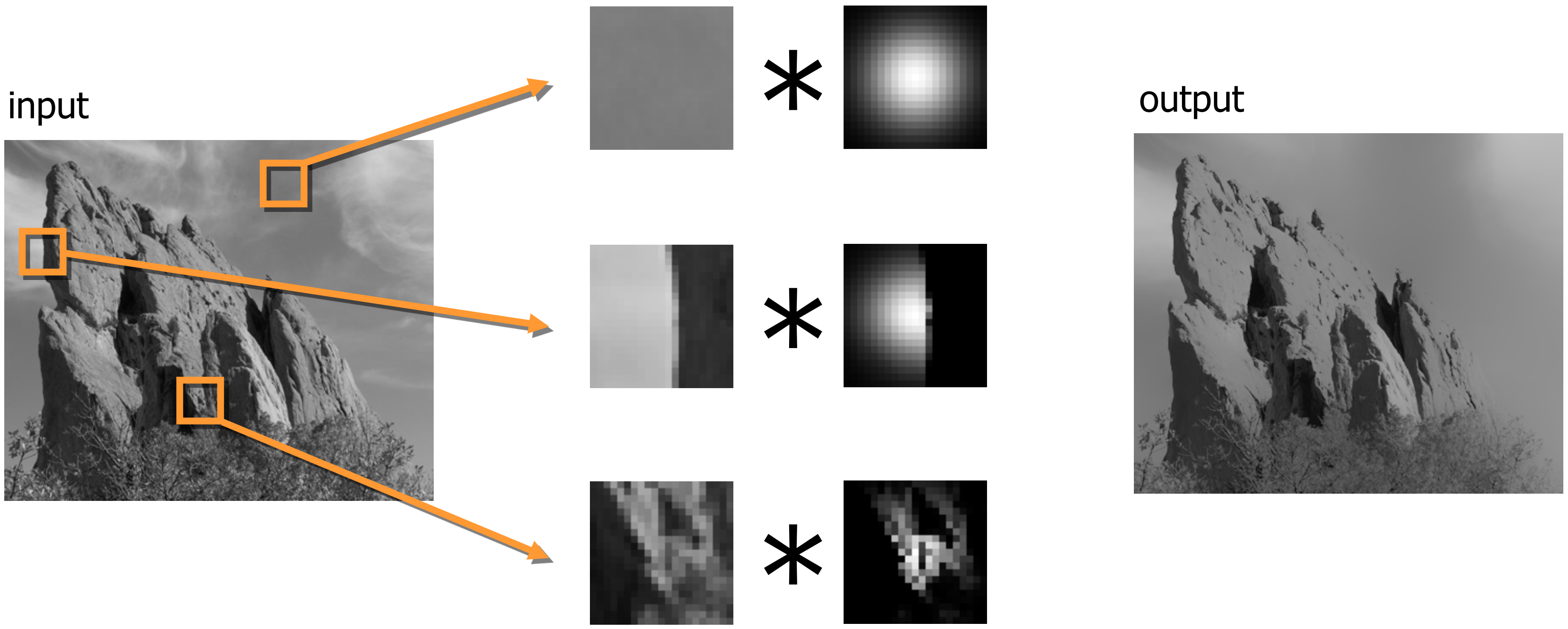
1. The matched features of the hypothesized object are used to determine its **pose**.
2. The **3D mesh** of the object is used to project all its features onto the image.
3. A **verification procedure** checks how well the object features line up with edges on the image.

Canny Edge Detection

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

Bilateral Filtering

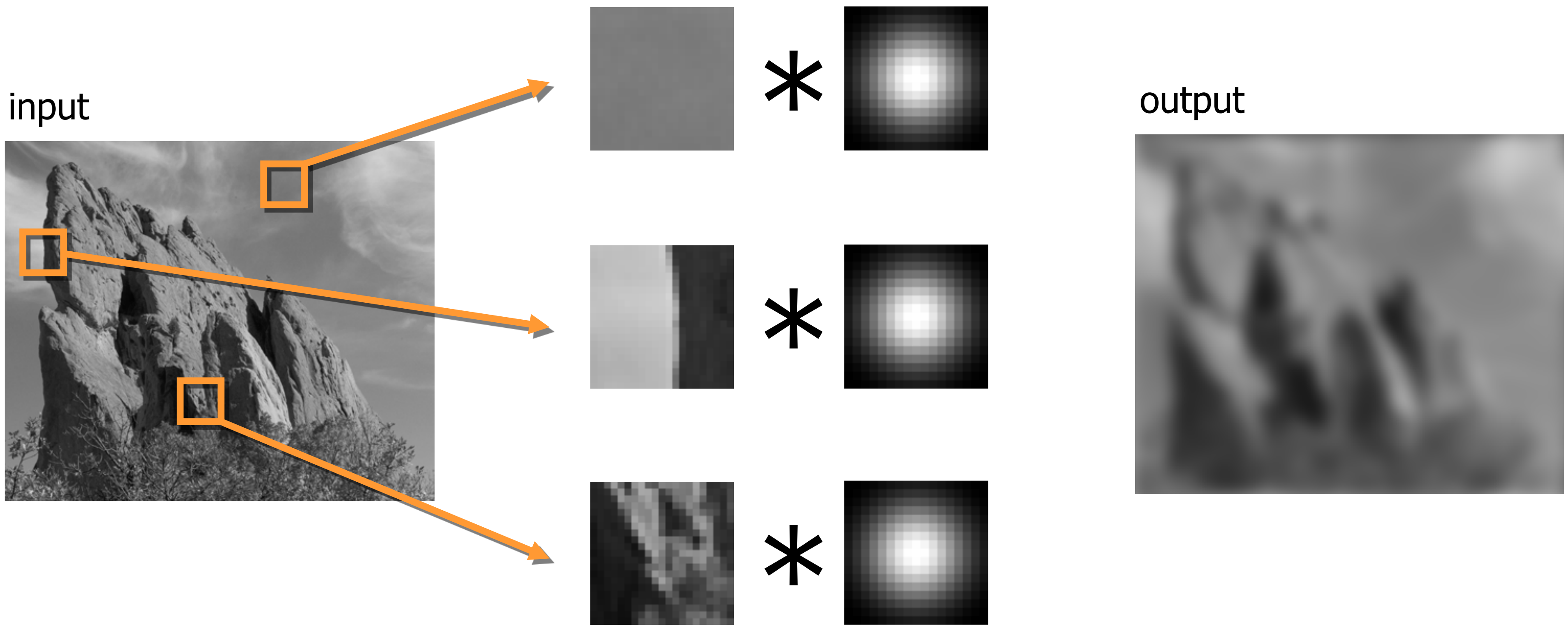
Spatially varying filters



Bilateral filter: kernel depends on the local image content.
See the Szeliski book for details.

Adapted from Sylvian Paris

Spatially varying filters



Compare to the result of using the same Gaussian kernel everywhere

Adapted from Sylvian Paris

Acknowledgement

Rest of the slides in this file are taken from:

https://people.csail.mit.edu/sparis/bf_course/

From: **A Gentle Introduction to Bilateral Filtering and its Applications**

Goal: Image Smoothing

Split an image into:

- large-scale features, structure
- small-scale features, texture

Naïve Approach: Gaussian Blur

input

BLUR

smoothed
(*structure, large scale*)

HALOS

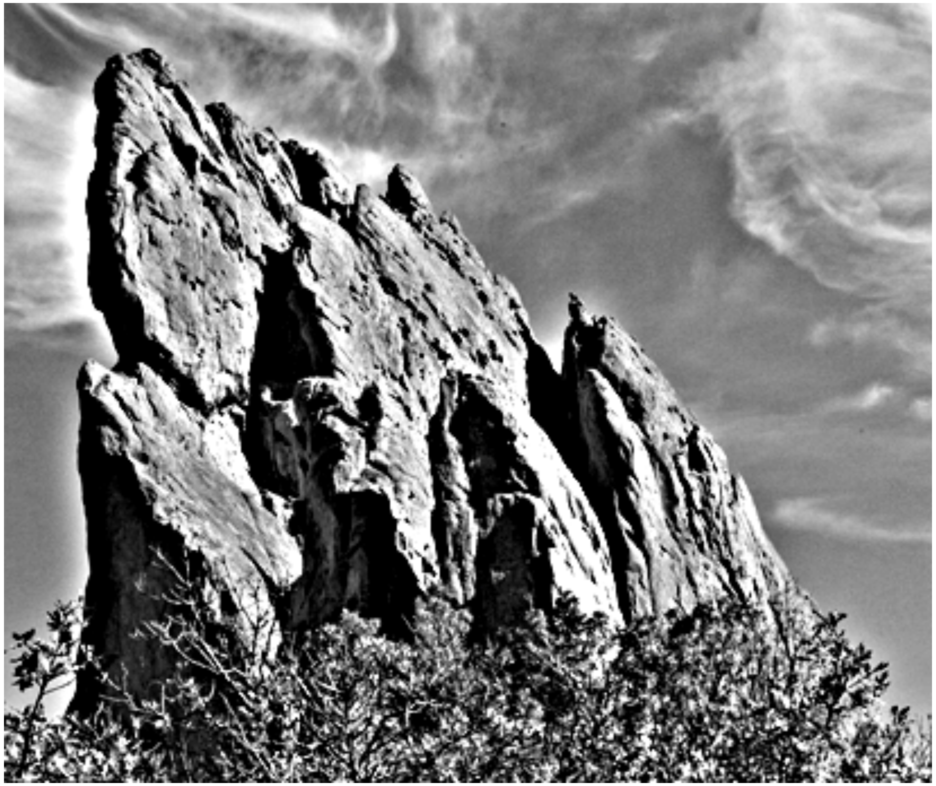
residual
(*texture, small scale*)

Gaussian Convolution

Impact of Blur and Halos

- If the decomposition introduces blur and halos, the final result is corrupted.

Sample manipulation:
increasing texture
(residual $\times$ 3)



Bilateral Filter: no Blur, no Halos

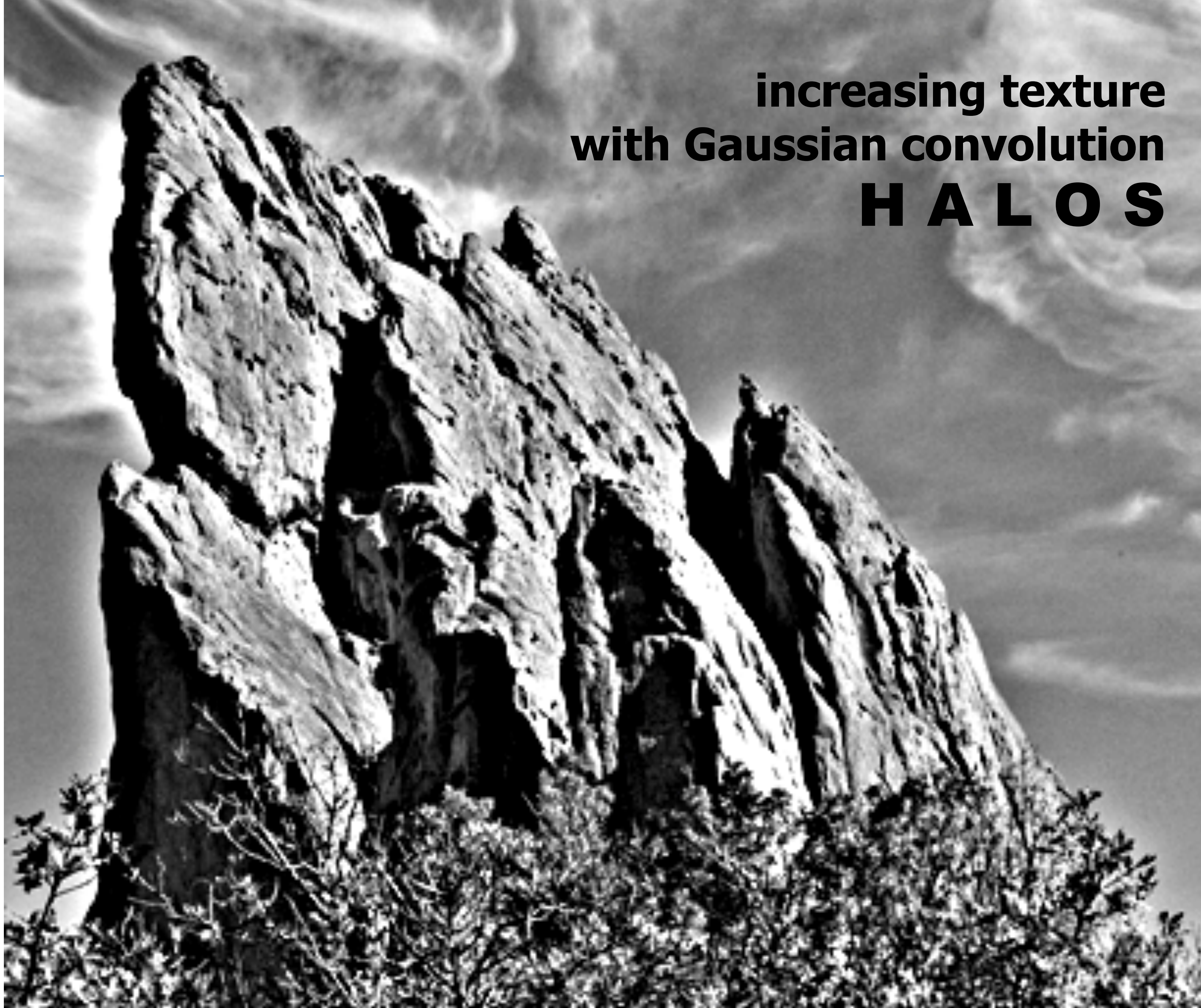
input

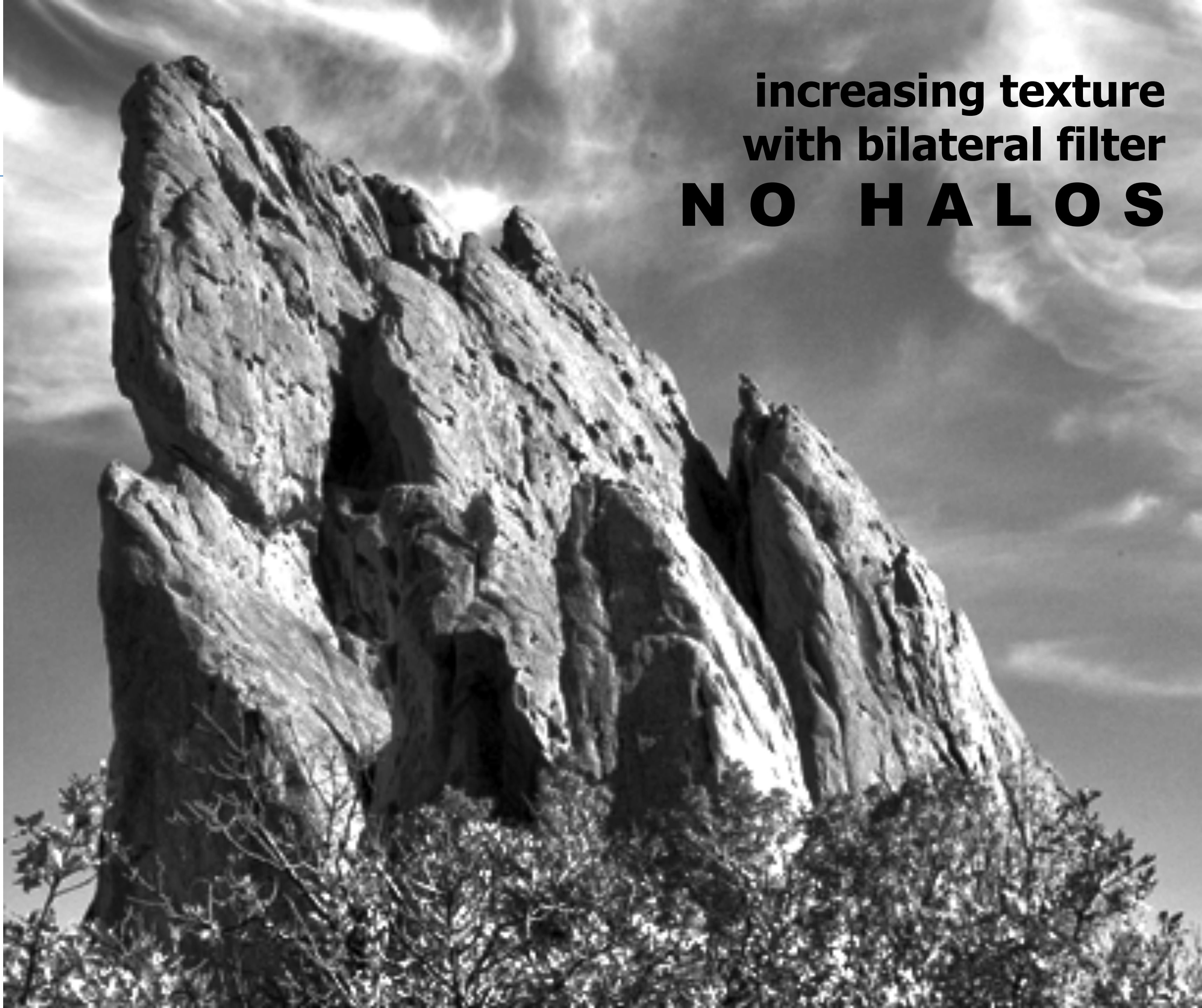
smoothed
(*structure, large scale*)

residual
(*texture, small scale*)

edge-preserving: Bilateral Filter


input


increasing texture
with Gaussian convolution
H A L O S

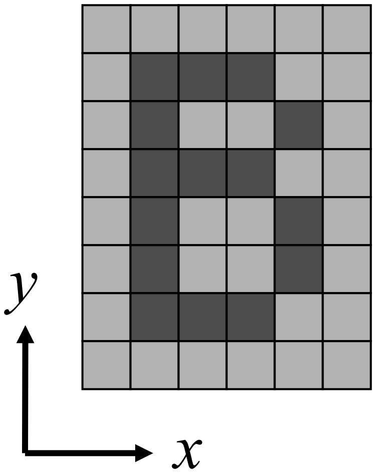

increasing texture
with bilateral filter
N O H A L O S

Many Other Options

- Bilateral filtering is not
  the only image smoothing filter
  - Diffusion, wavelets, Bayesian…

- We focus on bilateral filtering
  - Suitable for strong smoothing used in computational photography
  - Conceptually simple

Notation and Definitions

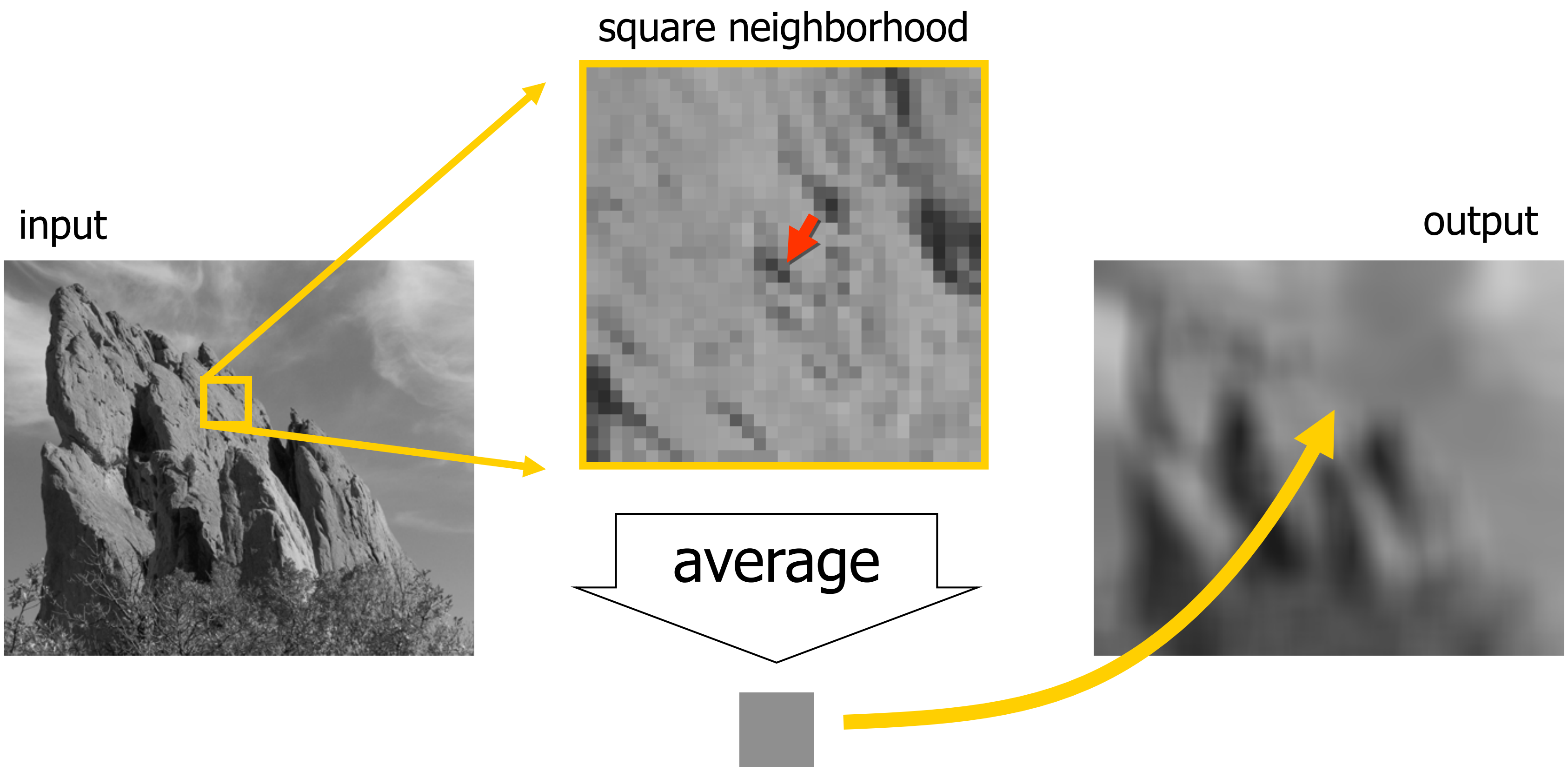


- Image = 2D array of pixels
- Pixel = intensity (scalar) or color (3D vector)
- $I_{\mathbf{p}}$ = value of image I at position: $\mathbf{p} = (p_x, p_y)$
- $F[I]$ = output of filter F applied to image I

Strategy for Smoothing Images

- Images are not smooth because adjacent pixels are different.
- Smoothing = making adjacent pixels
  look more similar.
- Smoothing strategy
  pixel $\rightarrow$ average of its neighbors

Box Average



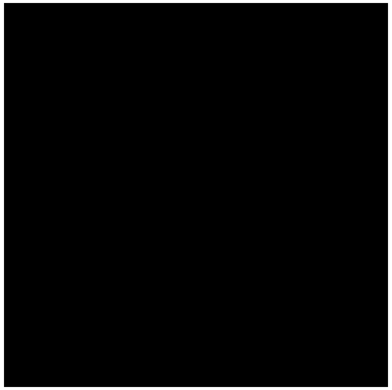
Equation of Box Average

$$BA[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in S} B_{\sigma}(\mathbf{p} - \mathbf{q})\, I_{\mathbf{q}}$$

result at pixel $\mathbf{p}$

sum over all pixels $\mathbf{q}$

normalized box function



intensity at pixel $\mathbf{q}$

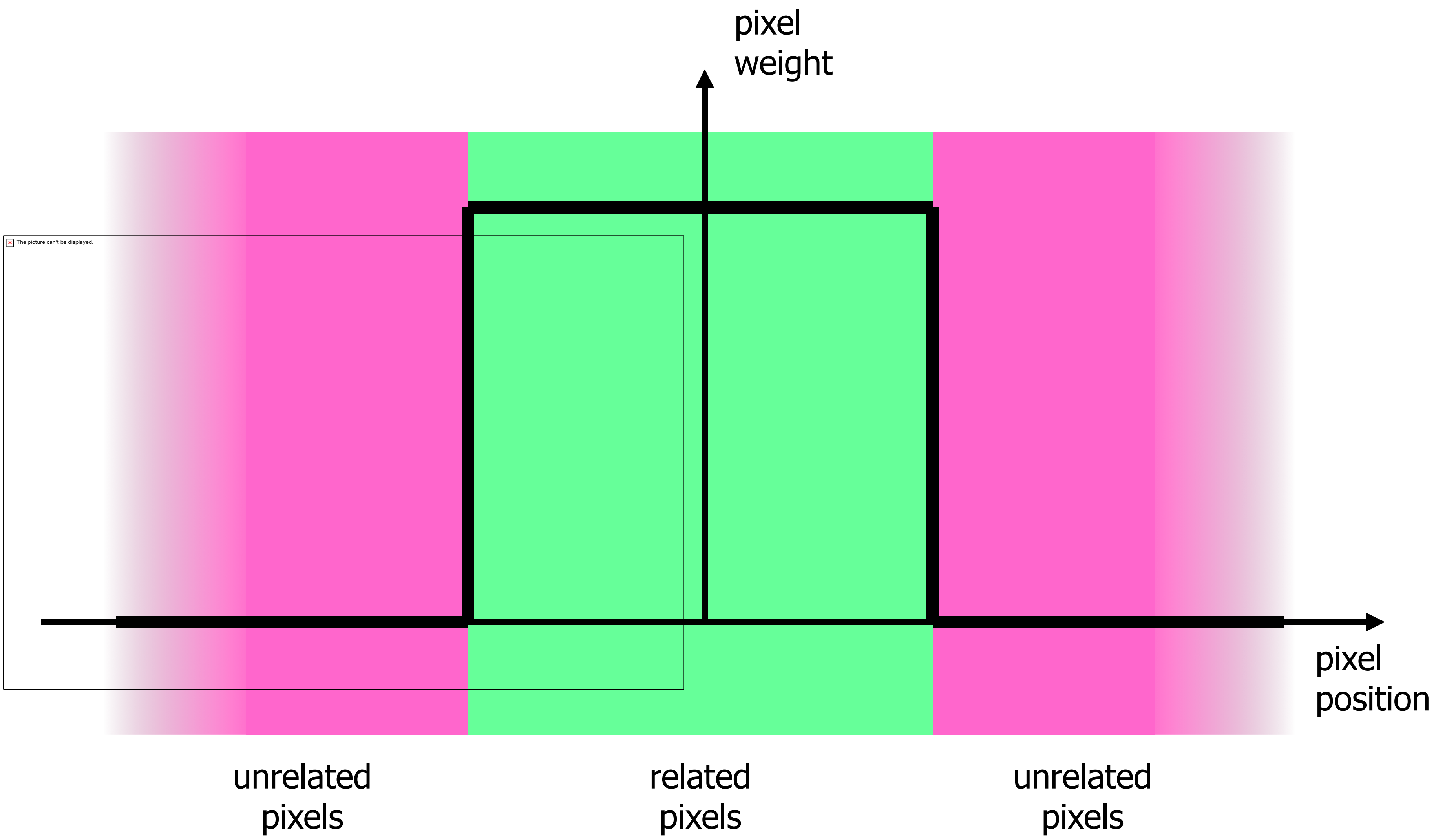
Square Box Generates Defects

- Axis-aligned streaks
- Blocky results

input

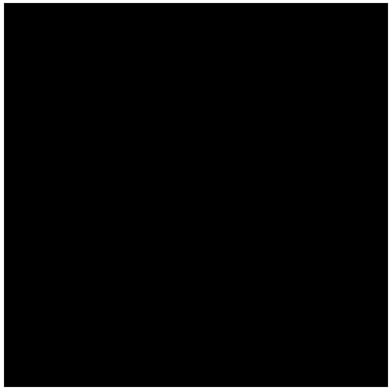
output

Box Profile

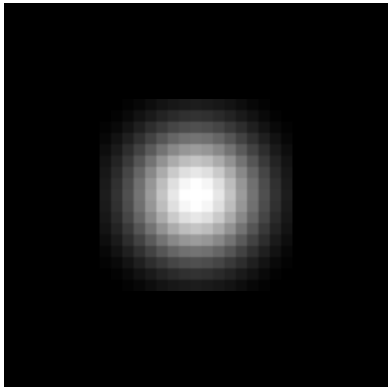

pixel
weight
pixel
position
unrelated
pixels
related
pixels
unrelated
pixels

Strategy to Solve these Problems

- Use an isotropic (*i.e.* circular) window.
- Use a window with a smooth falloff.

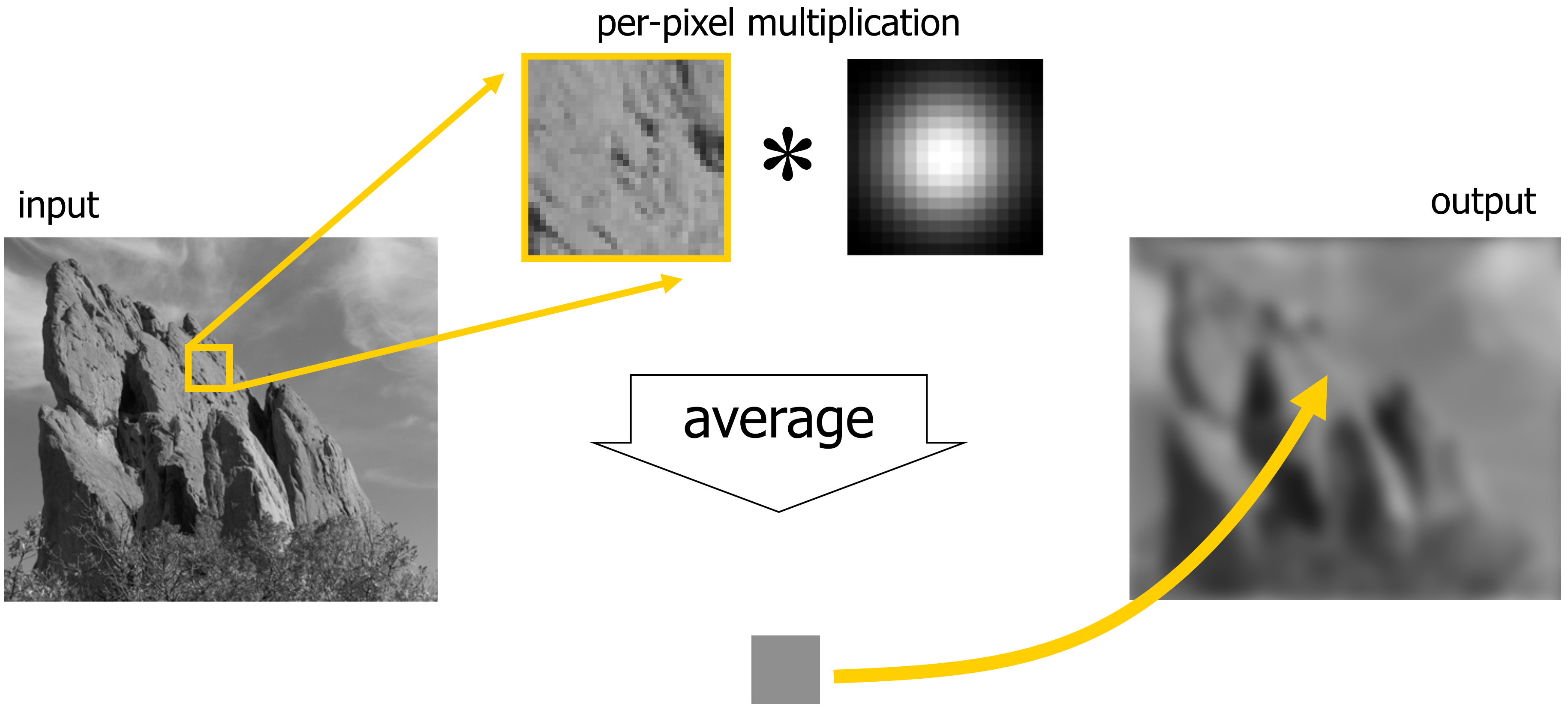


box window



Gaussian window

Gaussian Blur




input

box average

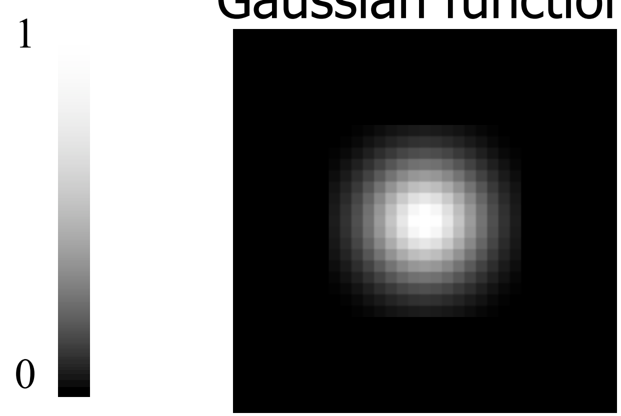
Gaussian blur

Equation of Gaussian Blur

Same idea: **weighted average of pixels**.

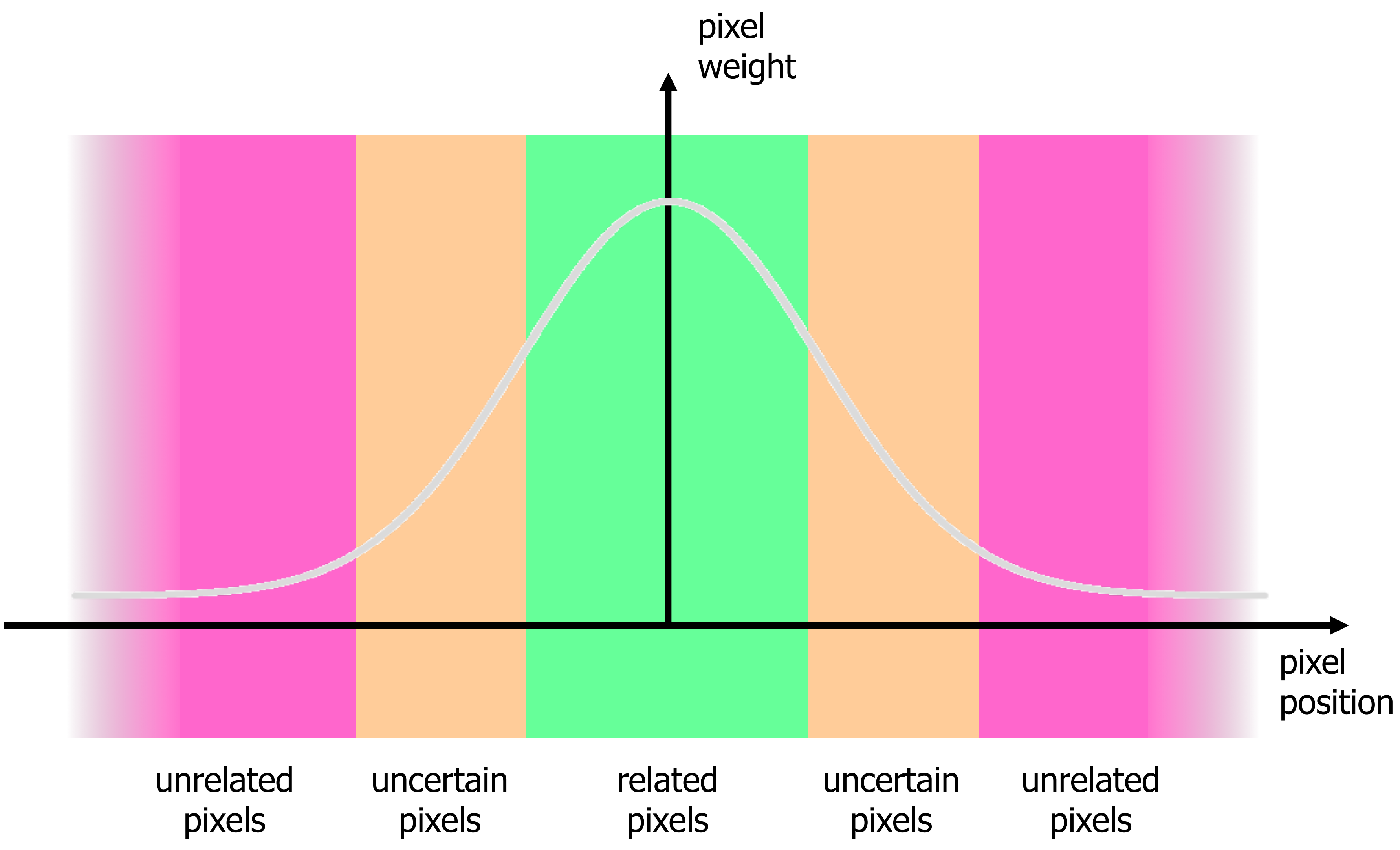
$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in S} \boxed{G_{\sigma}\left(\| \mathbf{p} - \mathbf{q} \|\right)} I_{\mathbf{q}}$$

normalized
Gaussian function

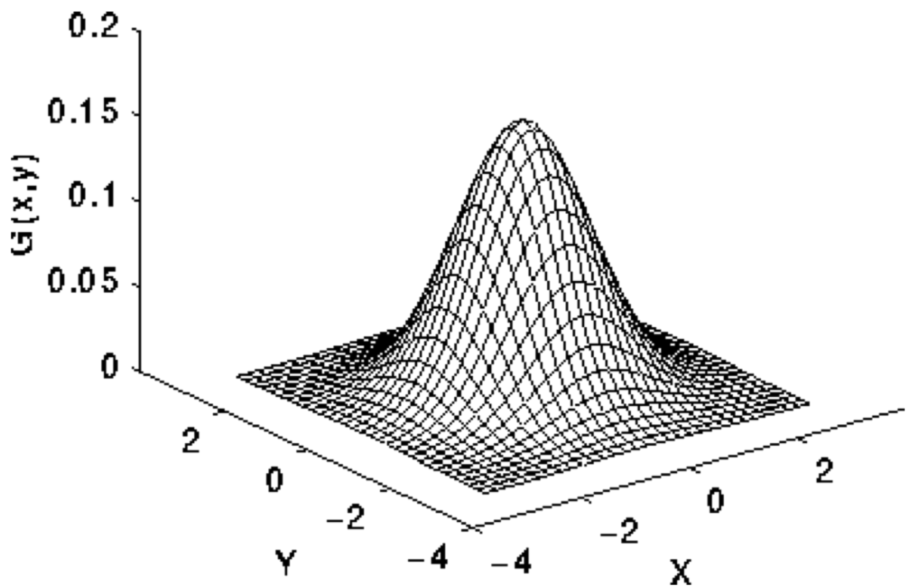


Gaussian Profile

$$G_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$



Gaussian Filter



2-D Gaussian distribution with mean (0,0) and σ =1

5x5 Gaussian filter with sigma=1.0

$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Discrete approximation to Gaussian function with σ =1.0

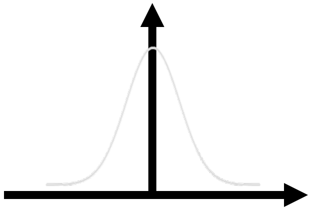
Image source: https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm

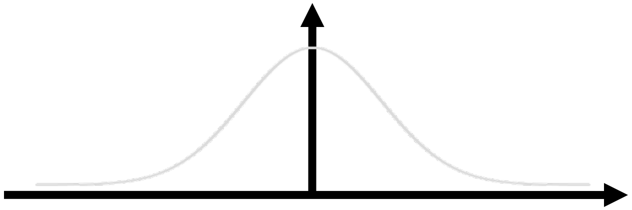
Spatial Parameter


input

$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in S} G_{\boxed{\sigma}}(\|\mathbf{p}-\mathbf{q}\|)\, I_{\mathbf{q}}$$

size of the window


small σ


large σ


limited smoothing


strong smoothing

How to set σ

- Depends on the application.

- Common strategy: proportional to image size
  - e.g. 2% of the image diagonal
  - property: independent of image resolution

Properties of Gaussian Blur

- Weights independent of spatial location
  - linear convolution
  - well-known operation
  - efficient computation (recursive algorithm, FFT…)

Properties of Gaussian Blur

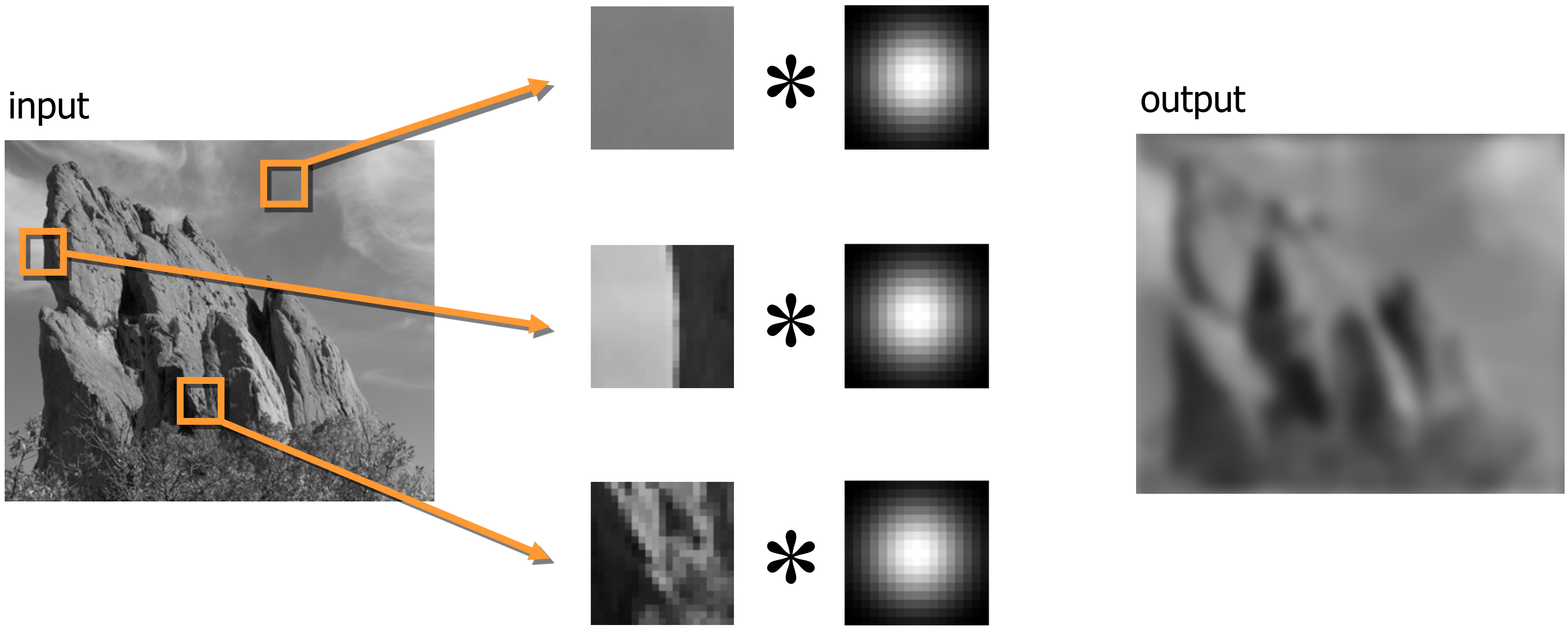
- Does smooth images
- But smoothes too much: **edges are blurred**.
  - Only spatial distance matters
  - No edge term

input

output

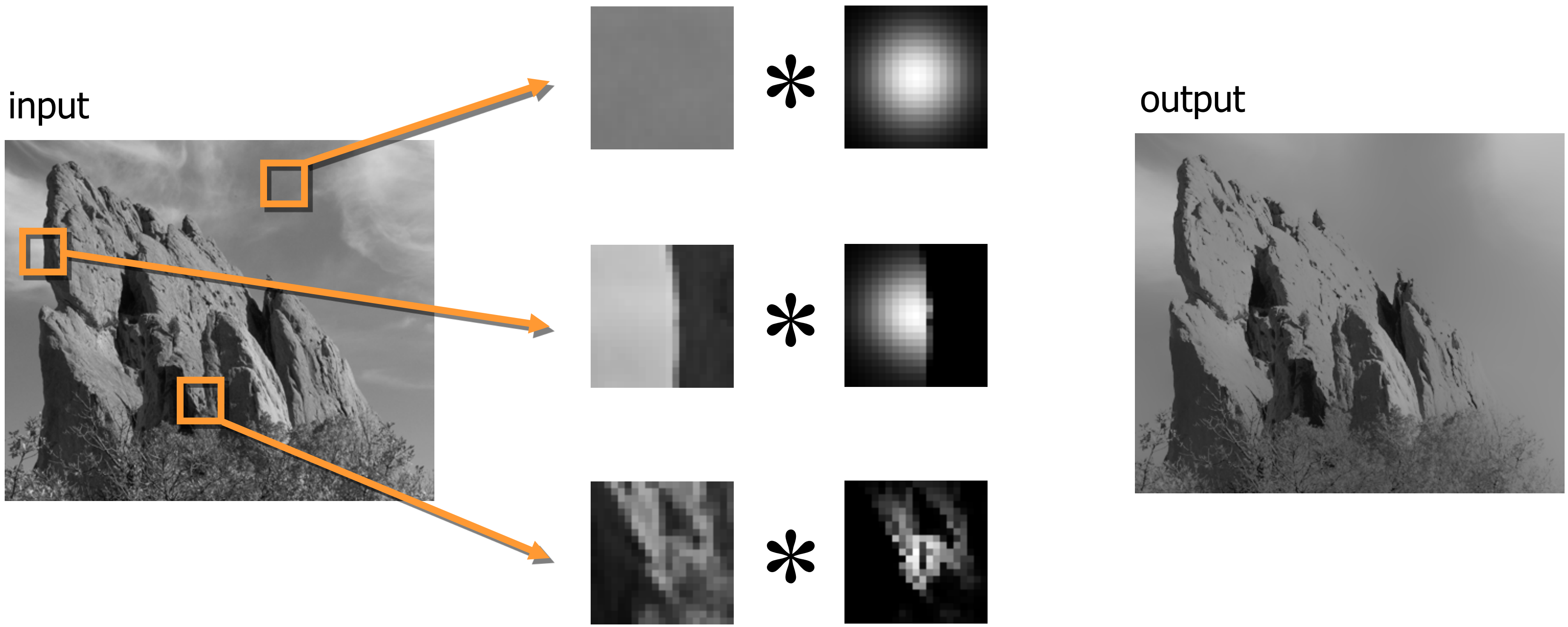
$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in S} \underbrace{G_{\sigma}(\| \mathbf{p} - \mathbf{q} \|)}_{\text{space}} I_{\mathbf{q}}$$

Blur Comes from Averaging across Edges



Same Gaussian kernel everywhere.

Bilateral Filter: No Averaging across Edges



The kernel shape depends on the image content.

[Aurich 95, Smith 97, Tomasi 98]

Bilateral Filter Definition: an Additional Edge Term

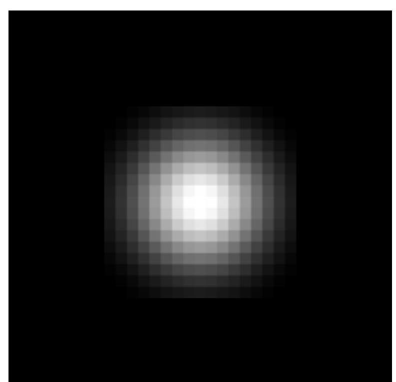
Same idea: **weighted average of pixels**.

new — not new — new

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}\left(\| \mathbf{p} - \mathbf{q} \|\right) G_{\sigma_r}\left(| I_{\mathbf{p}} - I_{\mathbf{q}} |\right) I_{\mathbf{q}}$$

normalization factor

space weight



range weight

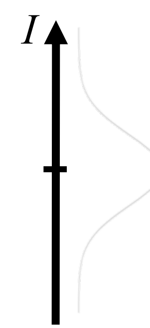
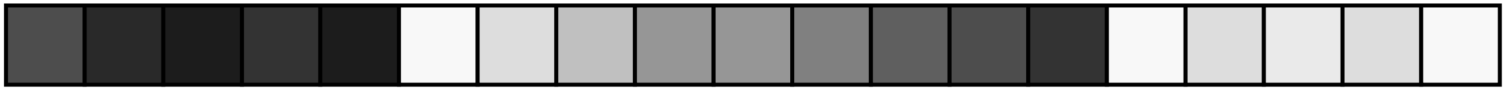
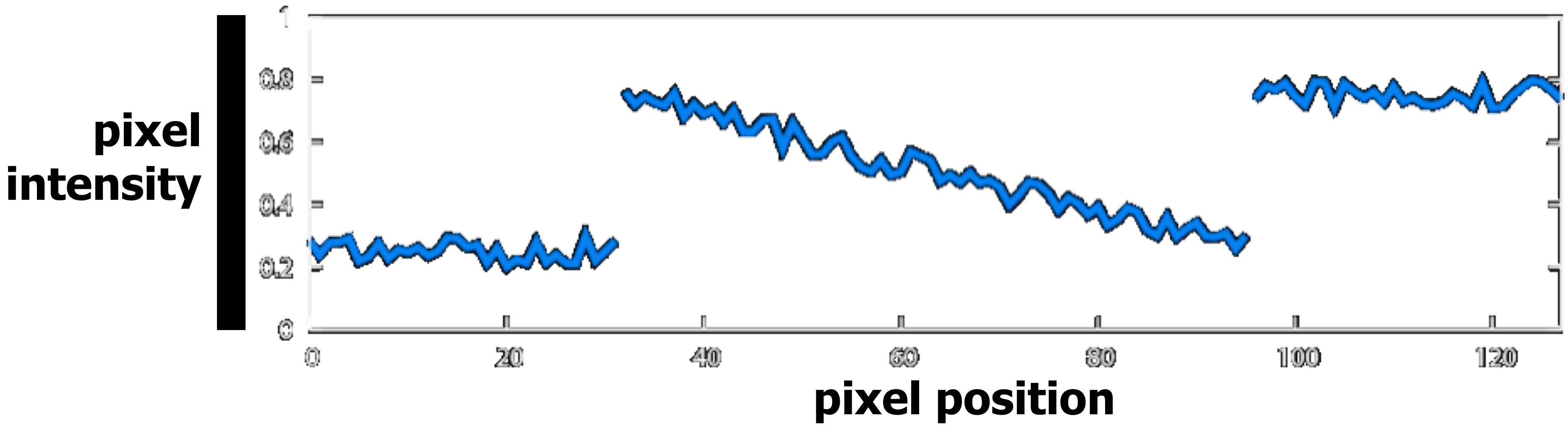


Illustration a 1D Image

- 1D image = line of pixels

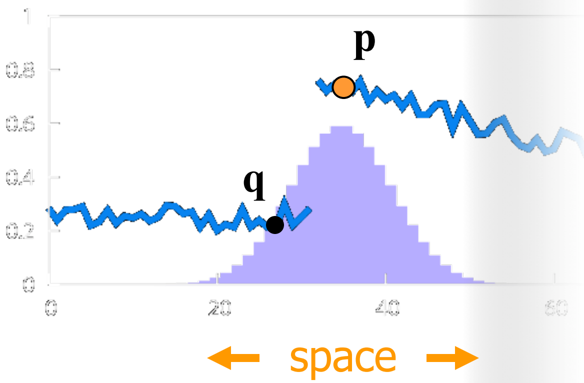


- Better visualized as a plot



Gaussian Blur and Bilateral Filter

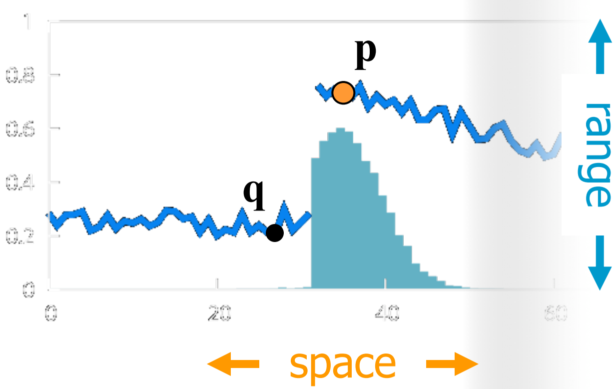
Gaussian blur



$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in S} \underbrace{G_{\sigma}(\| \mathbf{p} - \mathbf{q} \|)}_{\text{space}} I_{\mathbf{q}}$$

Bilateral filter

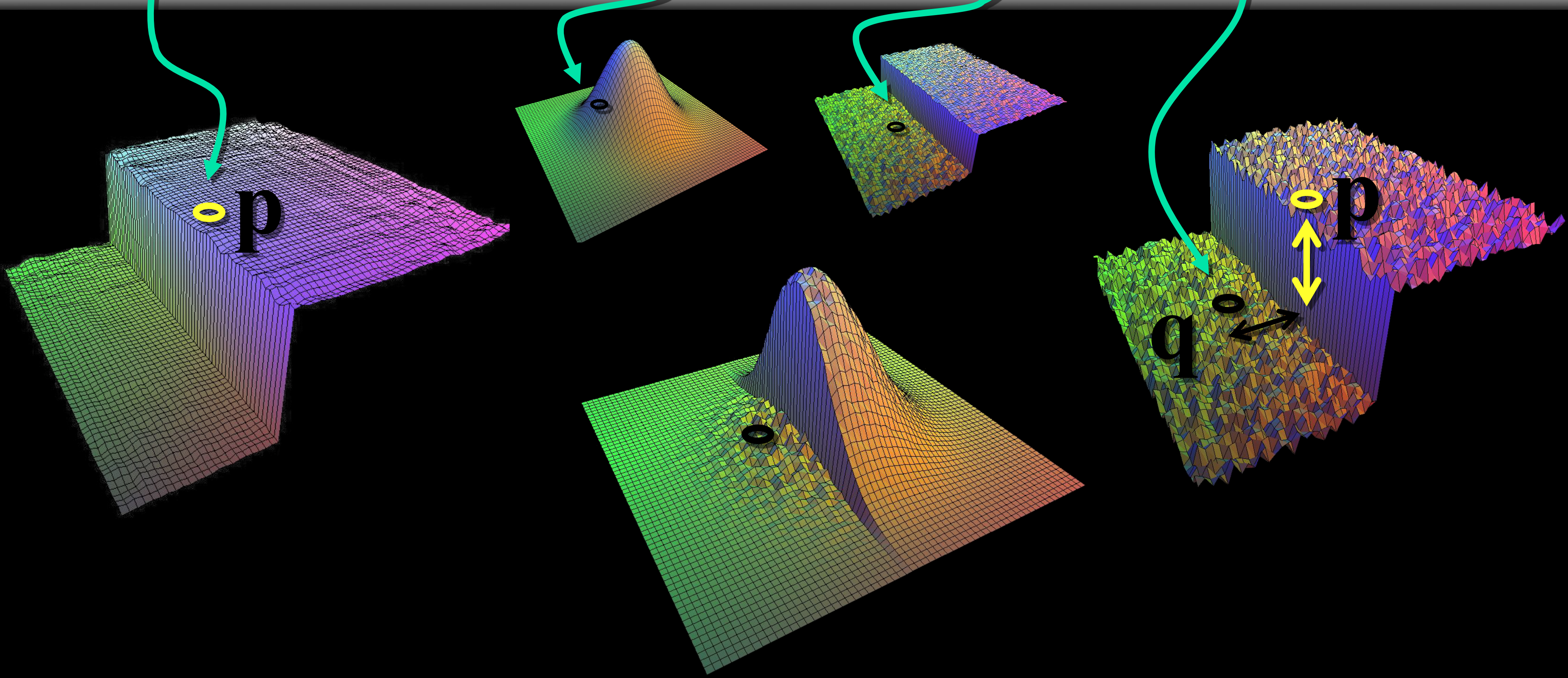
[Aurich 95, Smith 97, Tomasi 98]



$$BF[I]_{\mathbf{p}} = \underbrace{\frac{1}{W_{\mathbf{p}}}}_{\text{normalization}} \sum_{\mathbf{q} \in S} \underbrace{G_{\sigma_s}(\| \mathbf{p} - \mathbf{q} \|)}_{\text{space}} \underbrace{G_{\sigma_r}(| I_{\mathbf{p}} - I_{\mathbf{q}} |)}_{\text{range}} I_{\mathbf{q}}$$

Bilateral Filter on a Height Field

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|) \; G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|) \; I_{\mathbf{q}}$$



reproduced from [Durand 02]

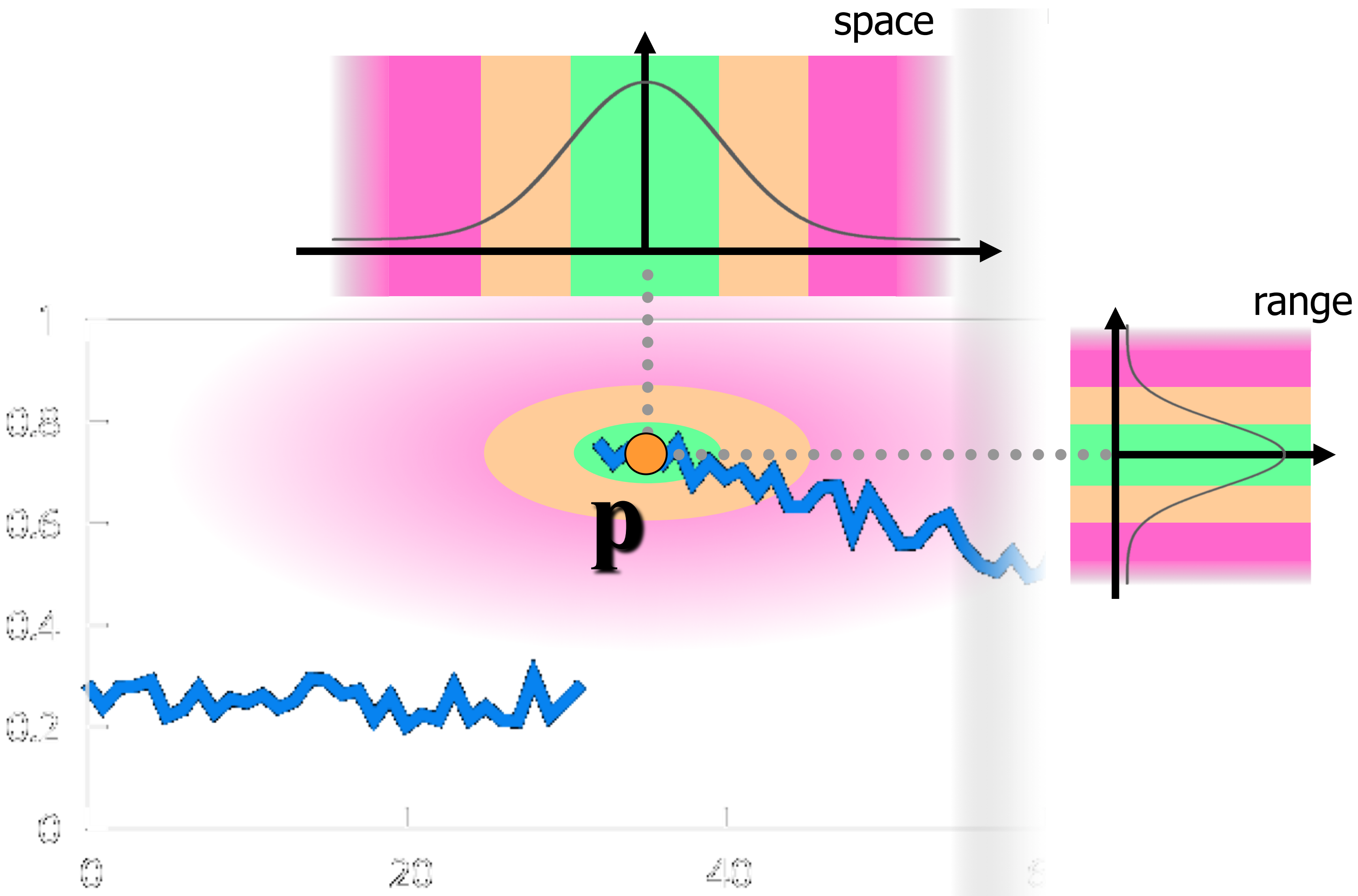
Space and Range Parameters

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_{\mathrm{s}}}(\| \mathbf{p} - \mathbf{q} \|) \, G_{\sigma_{\mathrm{r}}}(| I_{\mathbf{p}} - I_{\mathbf{q}} |) \, I_{\mathbf{q}}$$

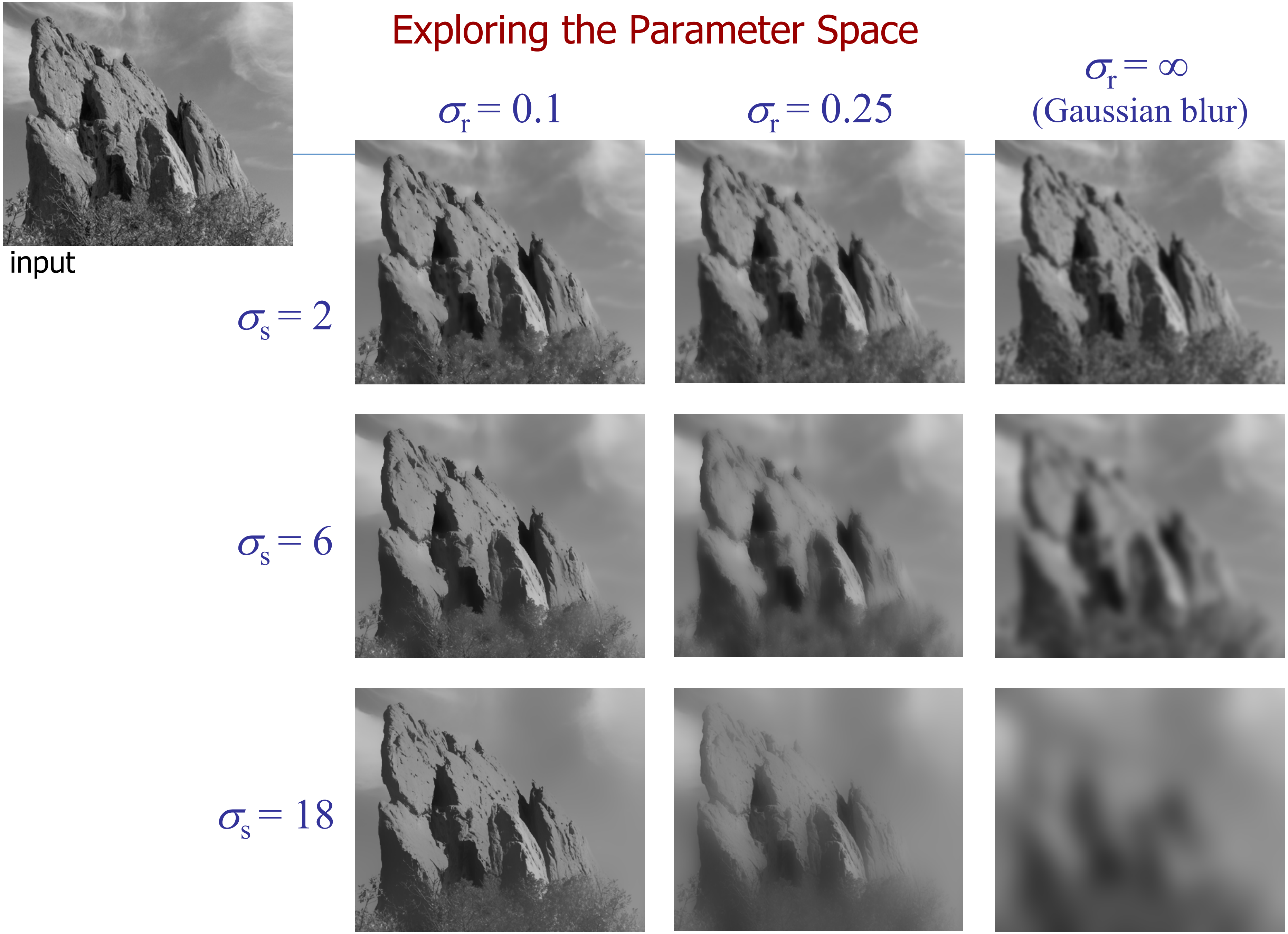
- space σ_{s} : spatial extent of the kernel, size of the considered neighborhood.
- range σ_{r} : "minimum" amplitude of an edge

Influence of Pixels

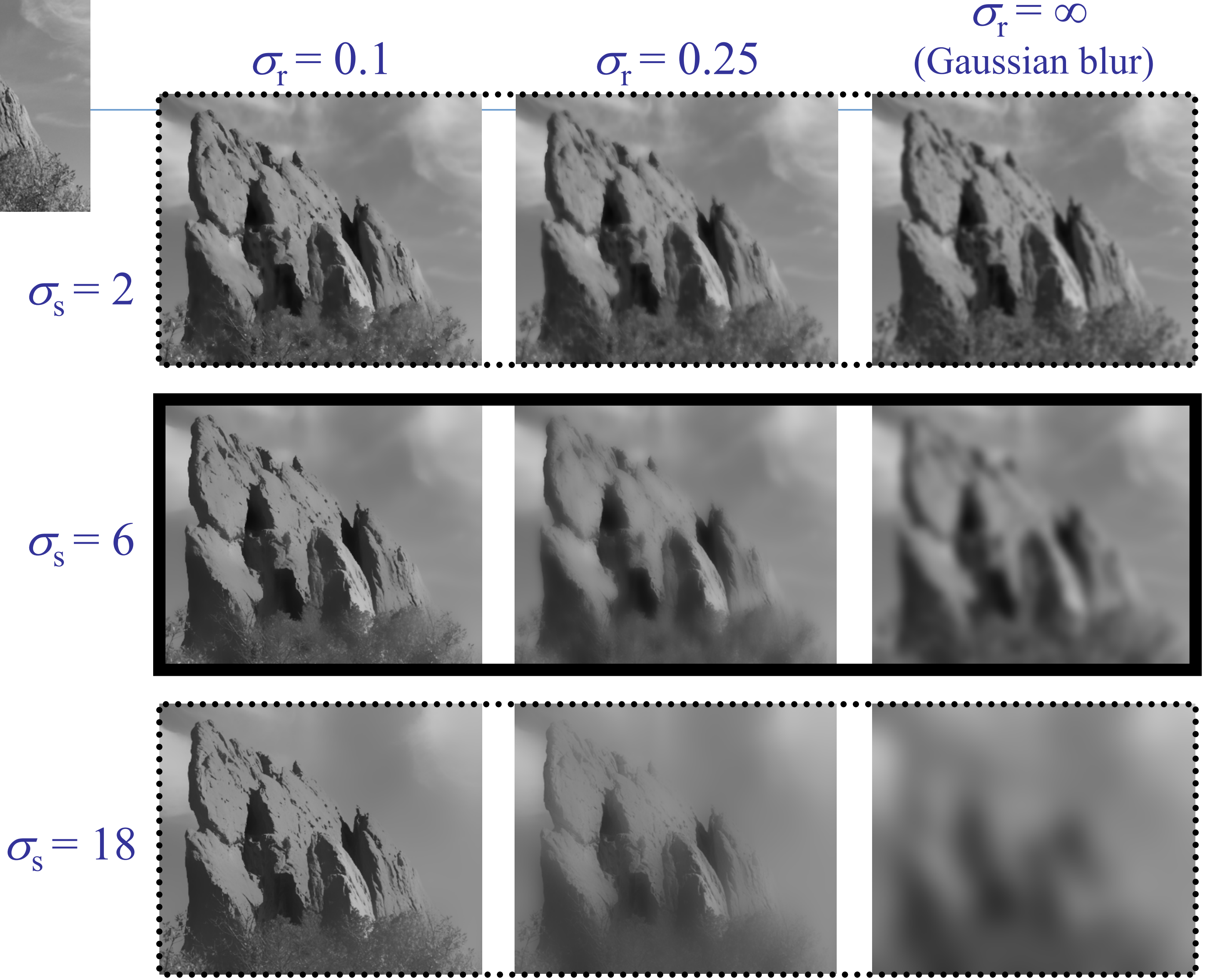
Only pixels close in space and in range are considered.



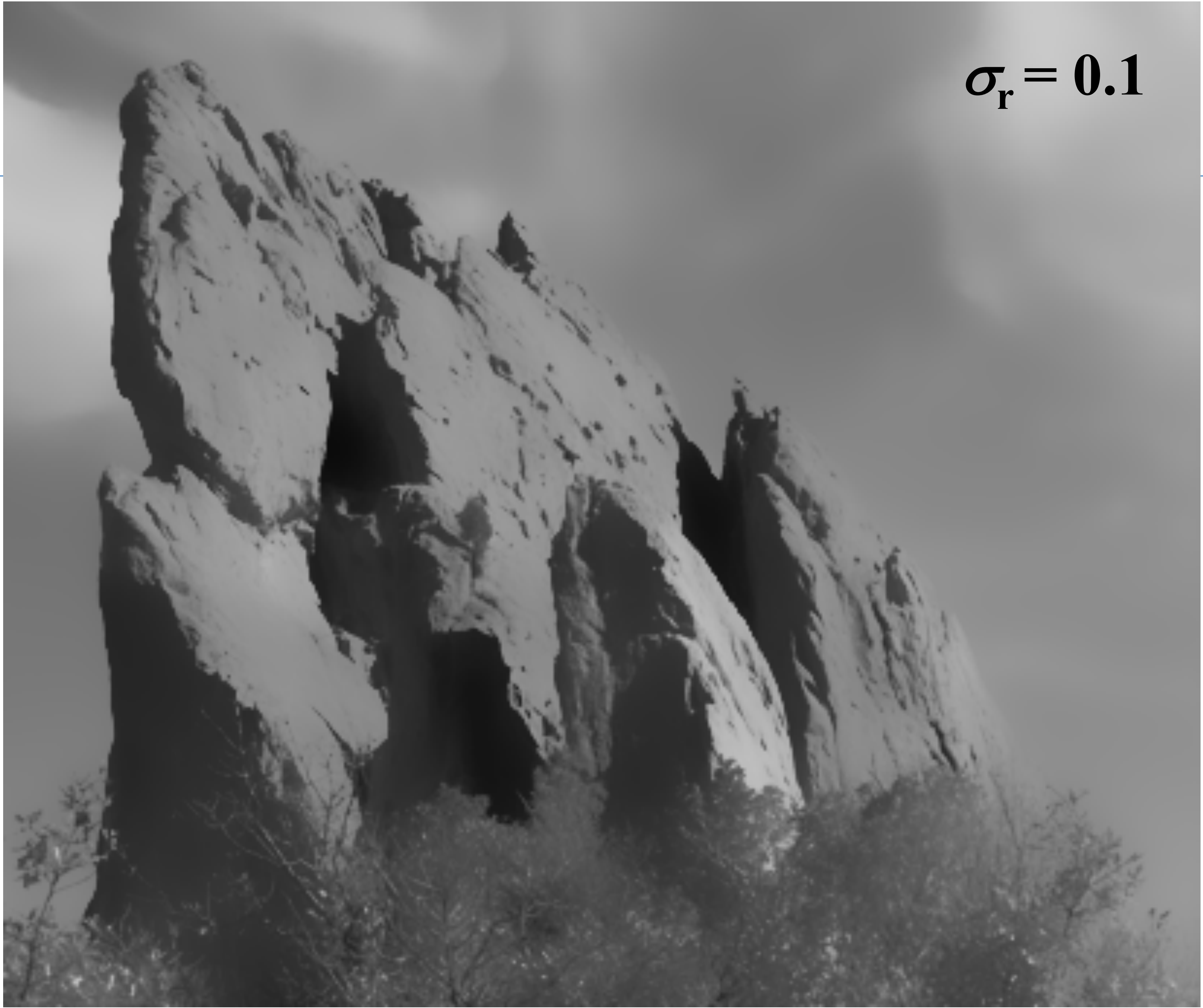
Exploring the Parameter Space

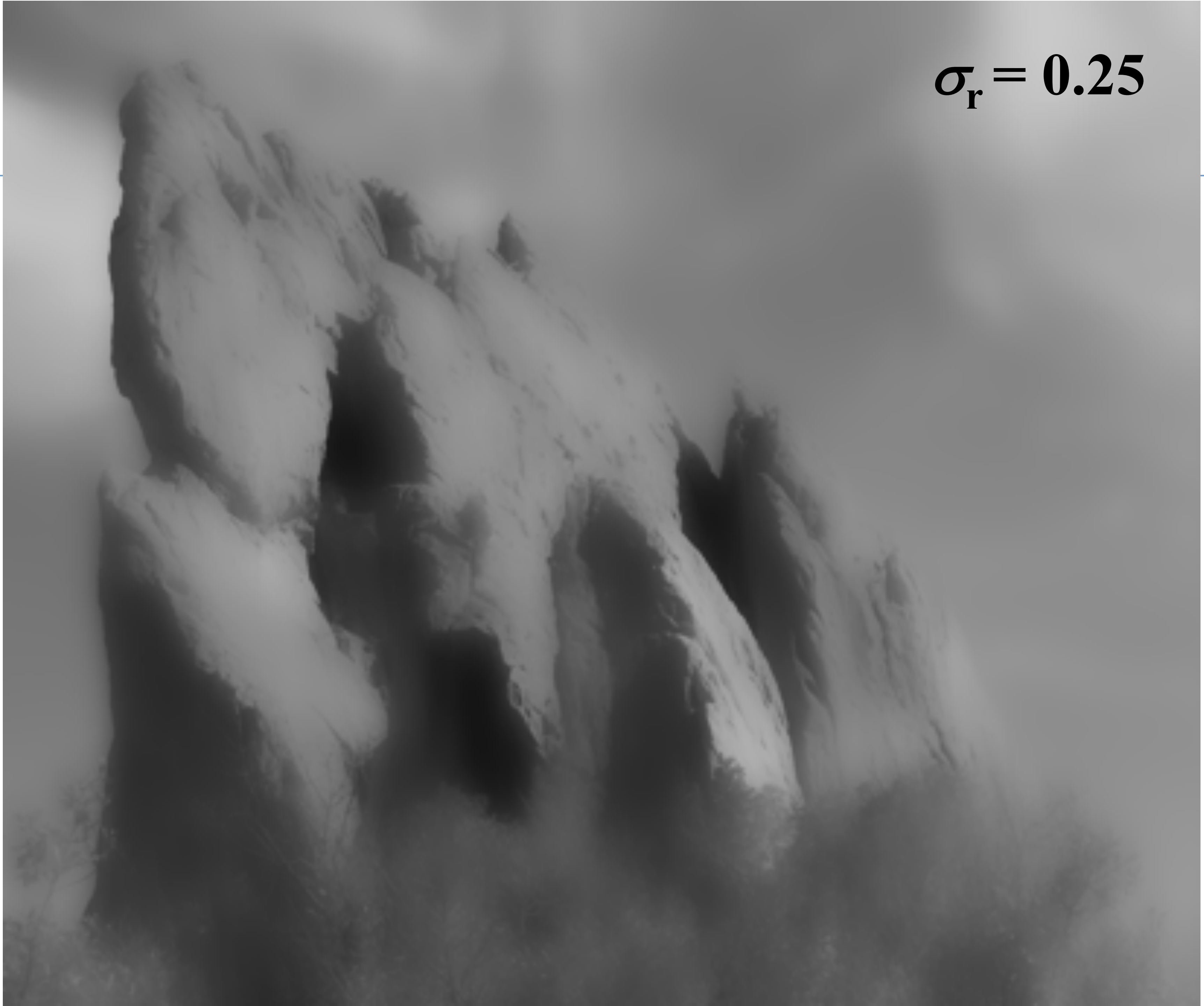


Varying the Range Parameter




input


$\sigma_r = 0.1$


$\sigma_r = 0.25$


$\sigma_r = \infty$
(Gaussian blur)

Varying the Space Parameter

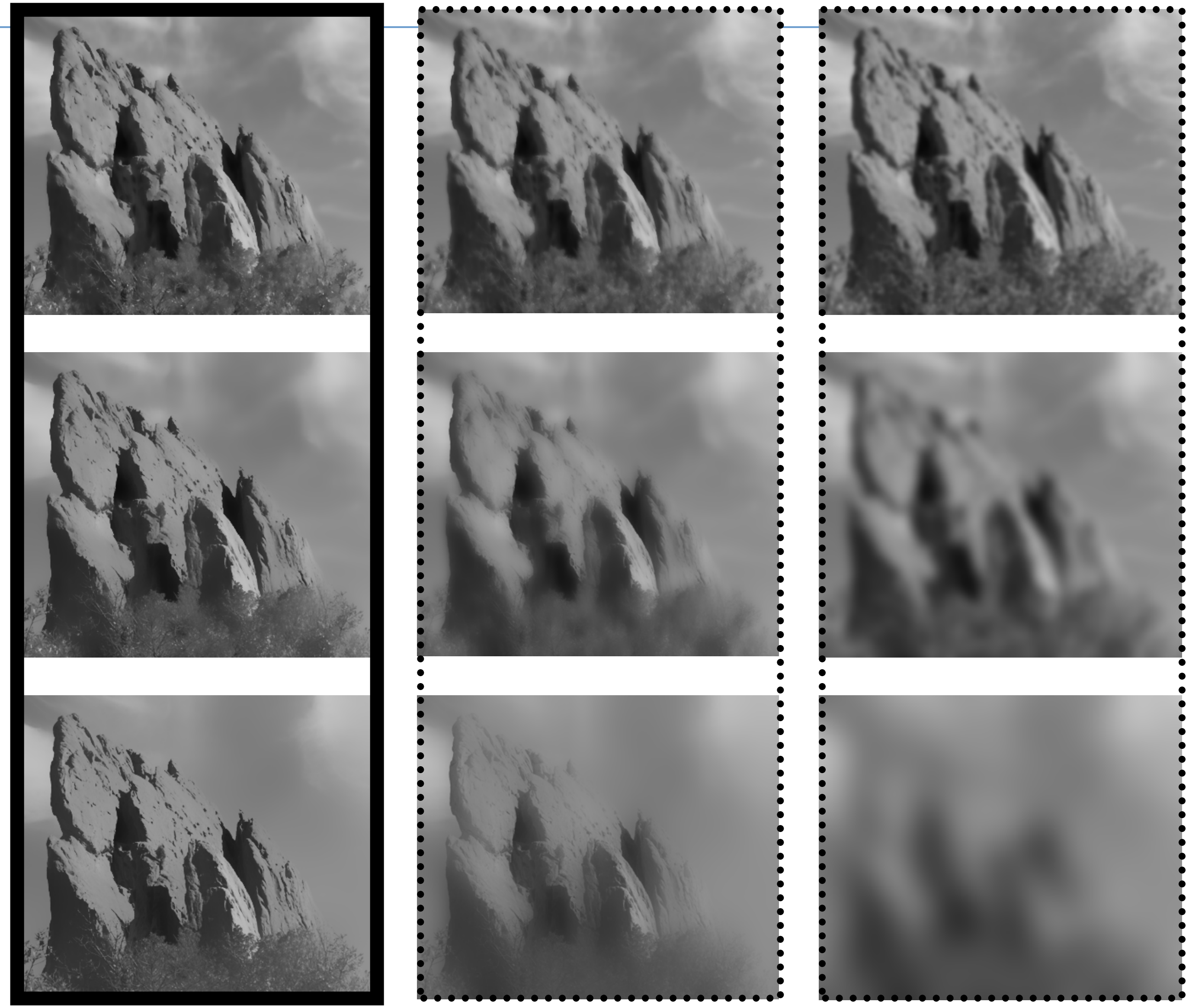
input

$\sigma_r = 0.1$ | $\sigma_r = 0.25$ | $\sigma_r = \infty$ (Gaussian blur)

$\sigma_s = 2$

$\sigma_s = 6$

$\sigma_s = 18$




input


$\sigma_s = 2$


$\sigma_s = 6$


$\sigma_s = 18$

How to Set the Parameters

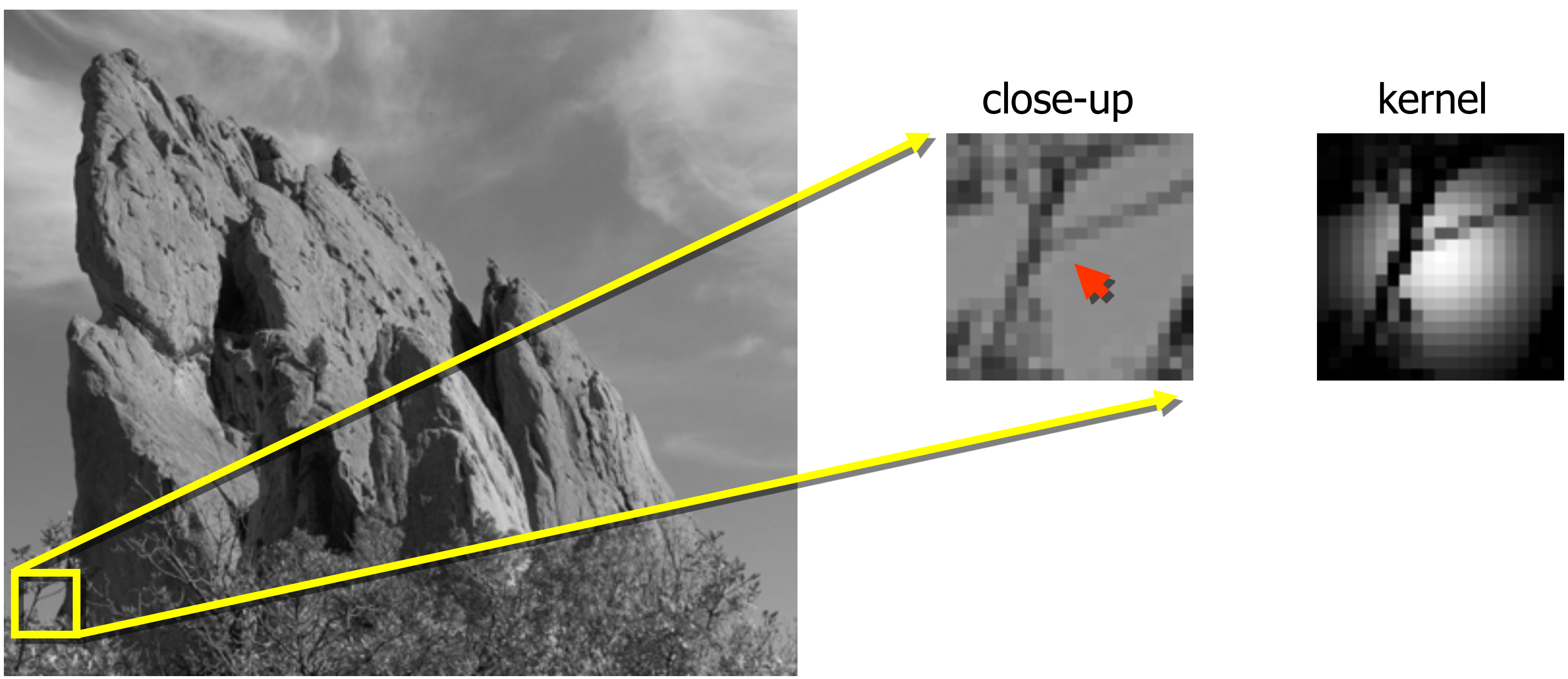
Depends on the application. For instance:

- space parameter: proportional to image size
  - e.g., 2% of image diagonal
- range parameter: proportional to edge amplitude
  - e.g., mean or median of image gradients
- independent of resolution and exposure

A Few
More Advanced
Remarks

Bilateral Filter Crosses Thin Lines

- Bilateral filter averages across features thinner than $\sim 2\sigma_s$
- Desirable for smoothing: more pixels = more robust
- Different from diffusion that stops at thin lines



Iterating the Bilateral Filter

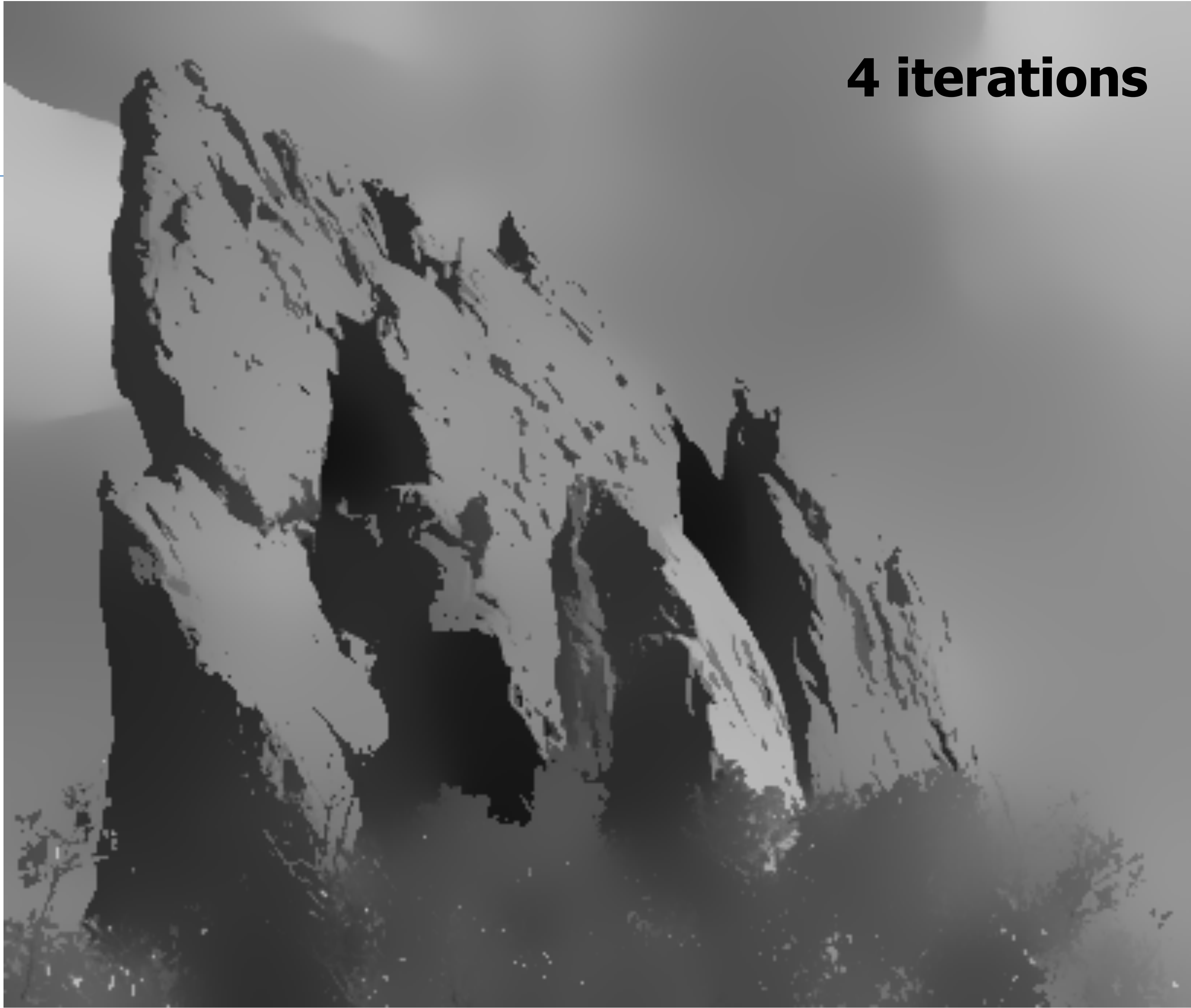
$$I_{(n+1)} = BF\left[I_{(n)} \right]$$

- Generate more piecewise-flat images
- Often not needed in computational photo.


input


1 iteration


2 iterations


4 iterations

Bilateral Filtering Color Images

For gray-level images

intensity difference

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}\in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_r}\left(\boxed{|I_{\mathbf{p}}-I_{\mathbf{q}}|}\right) \boxed{I_{\mathbf{q}}}$$

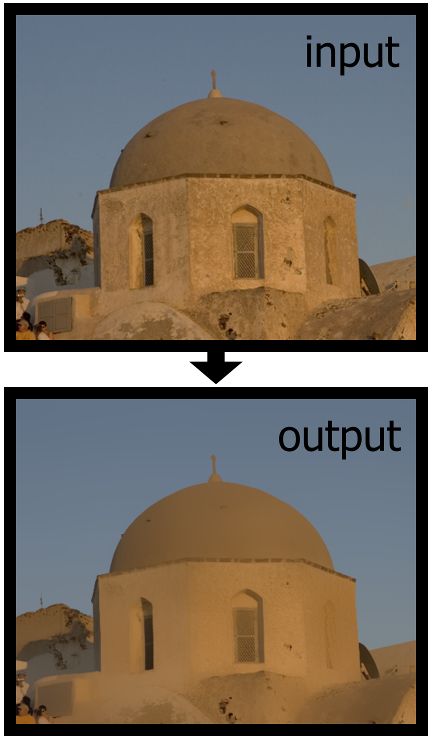
scalar

For color images

color difference

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}\in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_r}\left(\boxed{\|\mathbf{C}_{\mathbf{p}}-\mathbf{C}_{\mathbf{q}}\|}\right) \boxed{\mathbf{C}_{\mathbf{q}}}$$

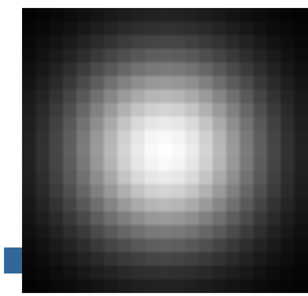
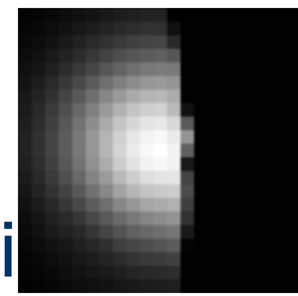
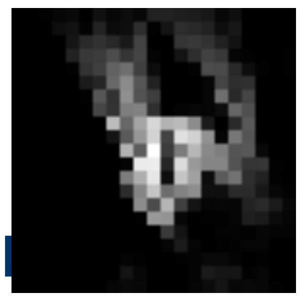
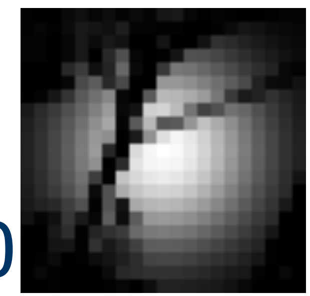
3D vector
(RGB, Lab)



The bilateral filter is extremely easy to adapt to your need.

Hard to Compute

- Nonlinear $$BF[I]_{\mathbf{p}} = \boxed{\frac{1}{W_{\mathbf{p}}}} \sum_{\mathbf{q}\in S} G_{\sigma_s}(\| \mathbf{p} - \mathbf{q} \|) \boxed{G_{\sigma_r}(| I_{\mathbf{p}} - I_{\mathbf{q}} |)} I_{\mathbf{q}}$$
- Complex, spatially varying kernels
  - Cannot be precomputed, no FFT…

- orce i ntatio > 10