Neighborhood Operations

CS 355: Introduction to Graphics and Image Processing

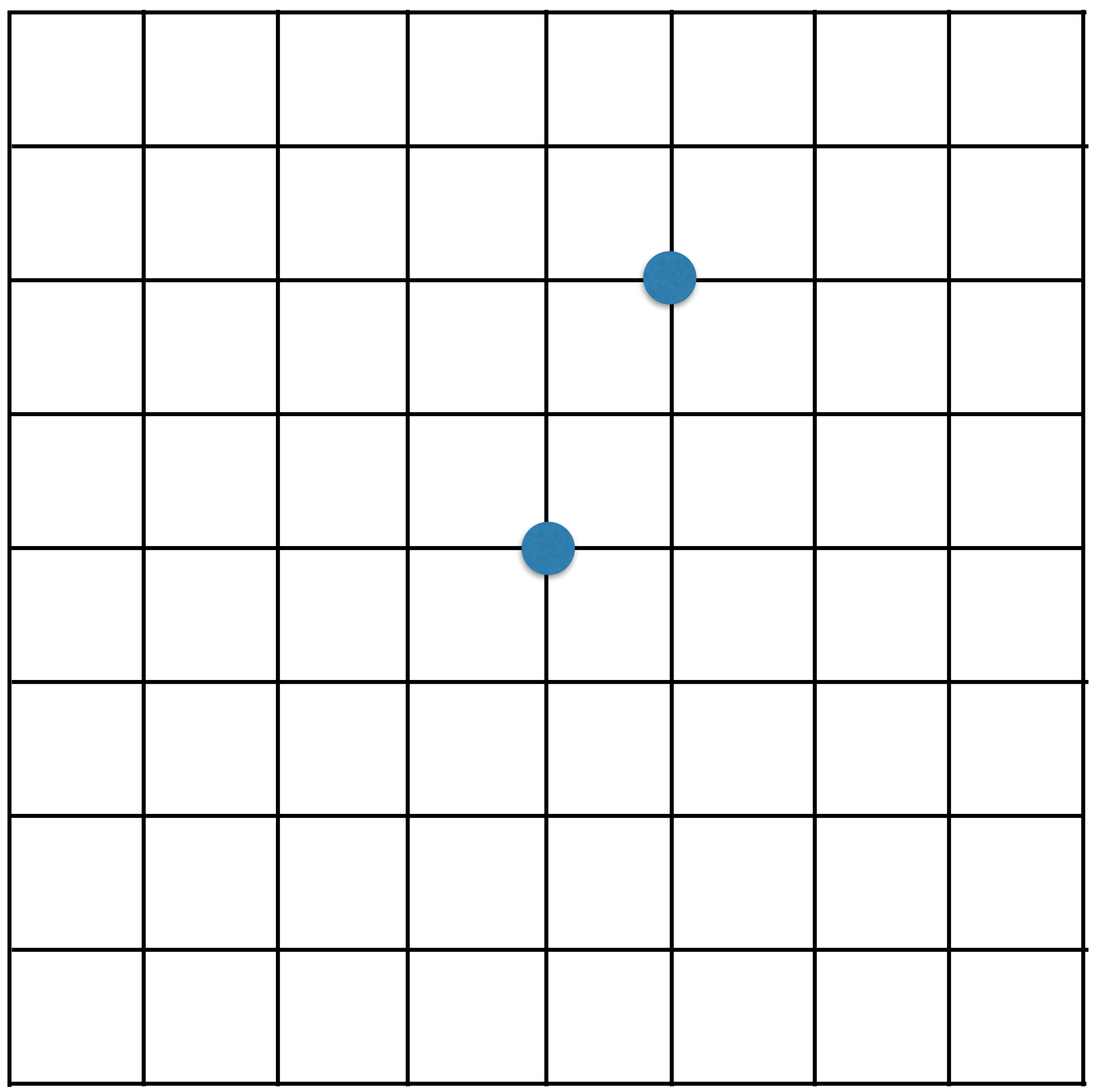
Neighborhood Operations

- Output pixel value is a function of that pixel and its neighbors
- Possible operations: sum, weighted sum, average, weighted average, min, max, median, …
- Most common workhorse in image processing

$$I'(x,y) = f\begin{pmatrix} I(x-1,y-1) & , & I(x,y-1) & , & I(x+1,y-1) & , \\ I(x-1,y) & , & I(x,y) & , & I(x+1,y) & , \\ I(x-1,y+1) & , & I(x,y+1) & , & I(x+1,y+1) & \end{pmatrix}$$

The Pixel Grid

- Many of the things we do involve using "neighboring" pixels
- Common approaches:
  - 4-connected (N, S, E, W)
  - 8-connected (add NE, SE, SW, NW)
- Distance?
  - Euclidean (as the crow flies)
  - 4-connected ("city block", "Manhattan")
  - 8-connected ("chessboard")



Spatial Filtering

- Most common is to multiply each of the pixels in the neighborhood by a respective weight and add them together
- The local weights are called a *mask* or *kernel*

I(x-1,y-1)	I(x,y-1)	I(x+1,y-1)
I(x-1,y)	I(x,y)	I(x+1,y)
I(x-1,y+1)	I(x,y+1)	I(x+1,y+1)

w(-1,-1)	w(0,-1)	w(1,-1)
w(-1,0)	w(0,0)	w(1,0)
w(-1,1)	w(0,1)	w(1,1)

Spatial Filtering

I(x-1,y-1)	I(x,y-1)	I(x+1,y-1)
I(x-1,y)	I(x,y)	I(x+1,y)
I(x-1,y+1)	I(x,y+1)	I(x+1,y+1)

w(-1,-1)	w(0,-1)	w(1,-1)
w(-1,0)	w(0,0)	w(1,0)
w(-1,1)	w(0,1)	w(1,1)

$$I'(x,y) = \sum_{s=-1}^{1} \sum_{t=-1}^{1} w(s,t)\; I(x+s, y+t)$$

Convolution

I(x-1,y-1)	I(x,y-1)	I(x+1,y-1)
I(x-1,y)	I(x,y)	I(x+1,y)
I(x-1,y+1)	I(x,y+1)	I(x+1,y+1)

w(1,1)	w(0,1)	w(-1,1)
w(1,0)	w(0,0)	w(-1,0)
w(1,-1)	w(0,-1)	w(-1,-1)

$$I'(x,y) = \sum_{s=-1}^{1} \sum_{t=-1}^{1} w(s,t)\ I(x-s, y-t)$$

Convolution is the same thing with the mask flipped

Correlation vs. Convolution

- Technically, spatial filtering is *correlation*, different from *convolution*
- They are the same up to flipping the mask/kernel
- Many casually use them interchangeably (be careful with the details)

w(-1,-1)	w(0,-1)	w(1,-1)
w(-1,0)	w(0,0)	w(1,0)
w(-1,1)	w(0,1)	w(1,1)

w(1,1)	w(0,1)	w(-1,1)
w(1,0)	w(0,0)	w(-1,0)
w(1,-1)	w(0,-1)	w(-1,-1)

Spatial Filtering

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

*

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

=

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

notation for convolution operator

$$I' = I * w$$

Spatial Filtering

- What do you do outside the image boundaries?
  - Assume zero (tends to darken borders if blurring)
  - Assume other constant value (perhaps average of entire image)
  - Wrap around
  - Assume same as closest pixel still in image
  - Or just don’t go there

Spatial Filtering

- Applications:
  - Blurring
  - Sharpening
  - Edge detection
  - and many more…

Smoothing

- If we can average multiple images together to remove noise, why not average multiple pixels?
- What does this assume?
- Effects:
    - Reduces noise
    - Causes blurring

Smoothing

- Any kernel with all positive weights does smoothing / blurring
- To average rather than add, divide by the sum of the weights
- Can be any size (larger means more blurring)

1	1	1
1	1	1
1	1	1

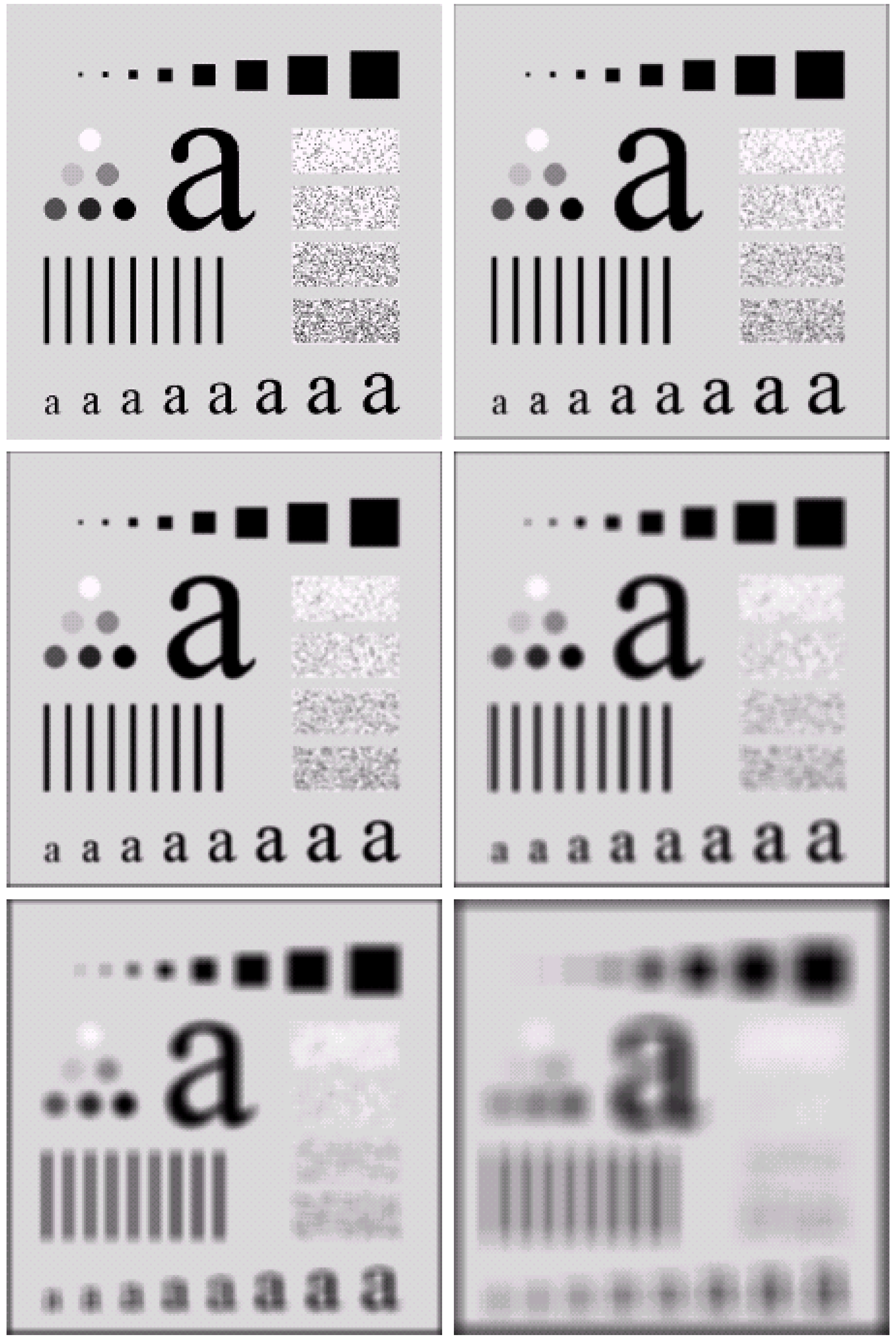
1	1	1
1	2	1
1	1	1

1	2	1
2	4	2
1	2	1

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

$$I'(x,y) = \frac{\sum_s \sum_t w(s,t)\ I(x+s, y+t)}{\sum_s \sum_t w(s,t)}$$

Smoothing



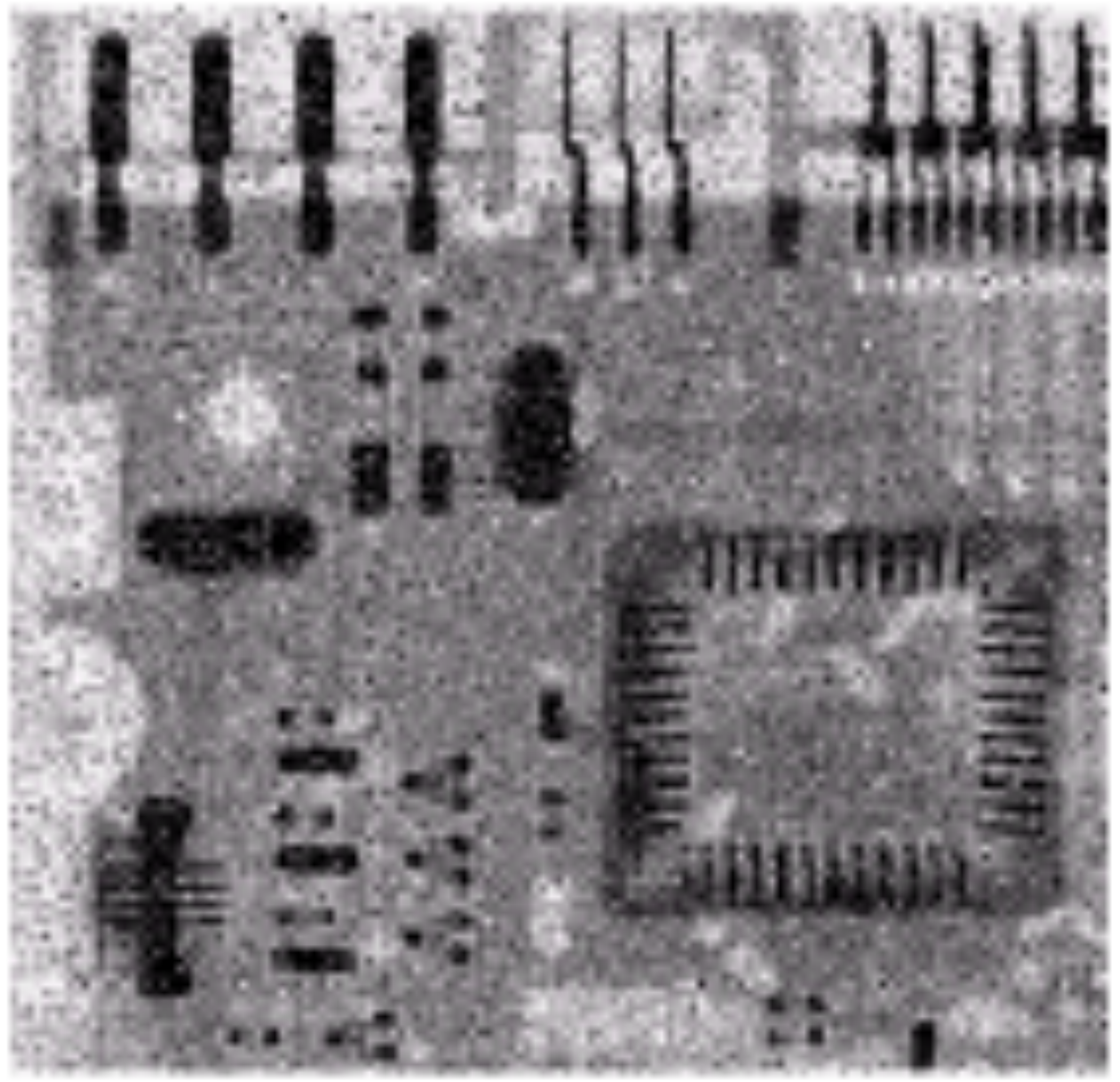
Nonlinear Smoothing

- Spatial filtering is linear, but many neighborhood operators are not
- Some do noise reduction:
  - Trimmed mean
  - Median filter
  - Bilateral filtering (or other adaptive weights)
- These try to be less sensitive to outliers and/or respect edges

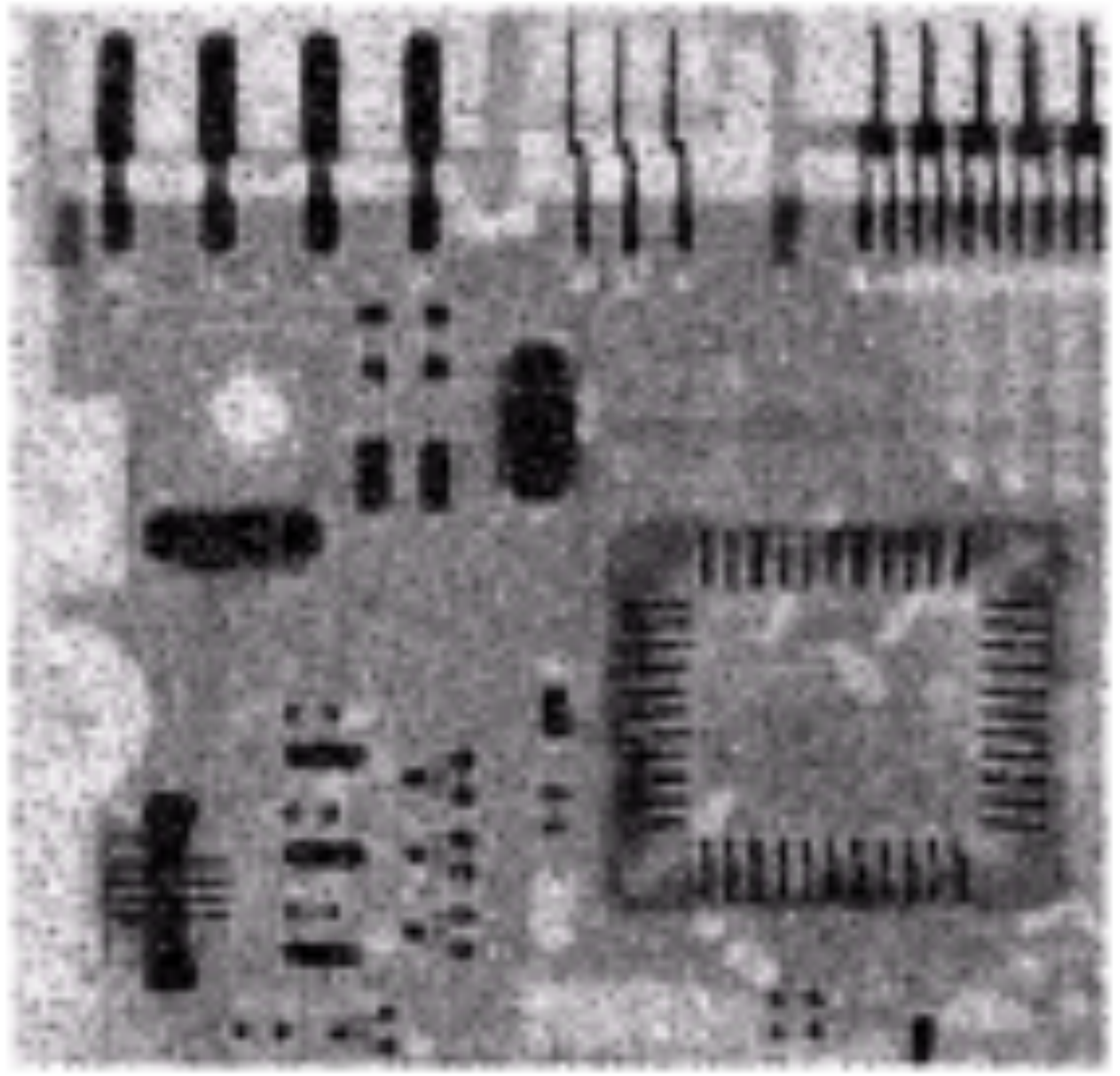
Median Filtering

- Output is the median (not the mean) of the neighborhood pixels
  - More robust to outliers
    (great for “salt and pepper” noise)
  - Tries to respect edges
    (goes with local majority)
  - But often rounds corners or
    loses very small/thin things

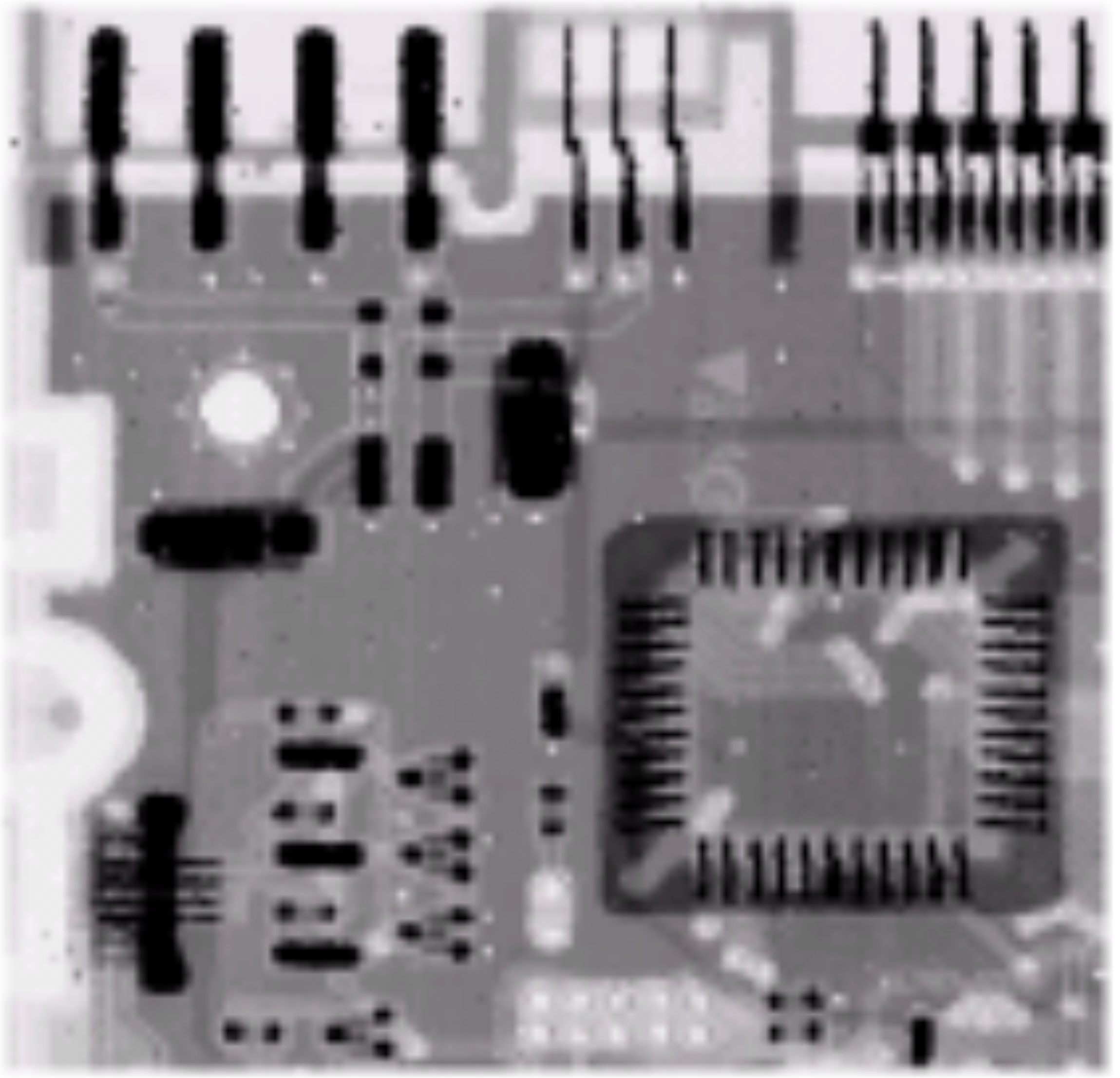
Median Filtering



Original



Mean



Median

Bilateral Filtering

- Spatially adapt the weights of the mask
  - Closer neighbors get more weight
  - Similar neighbors get more weight
- Many similar approaches use this idea, but this is most popular now
- Computationally expensive, but there are efficient approximations

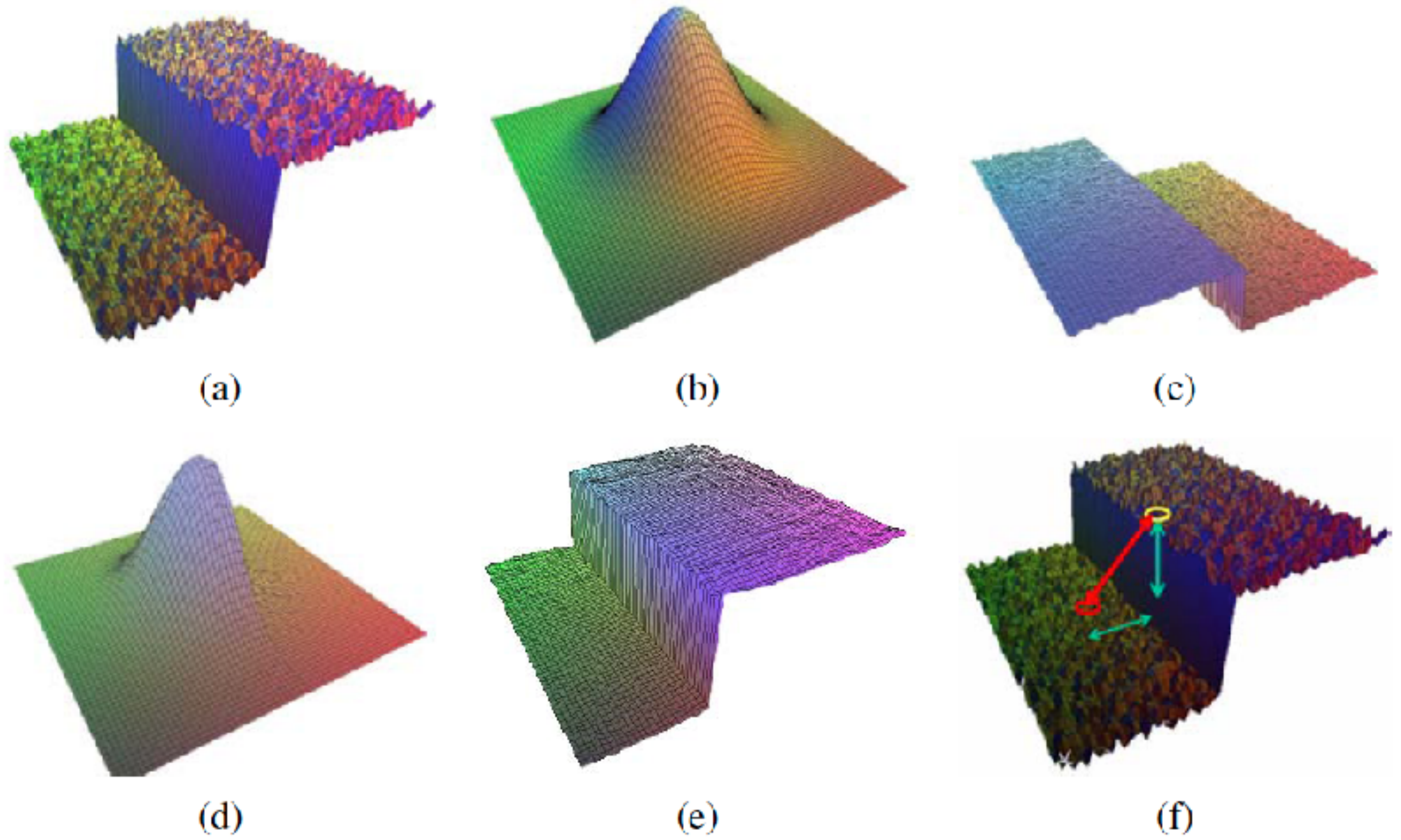
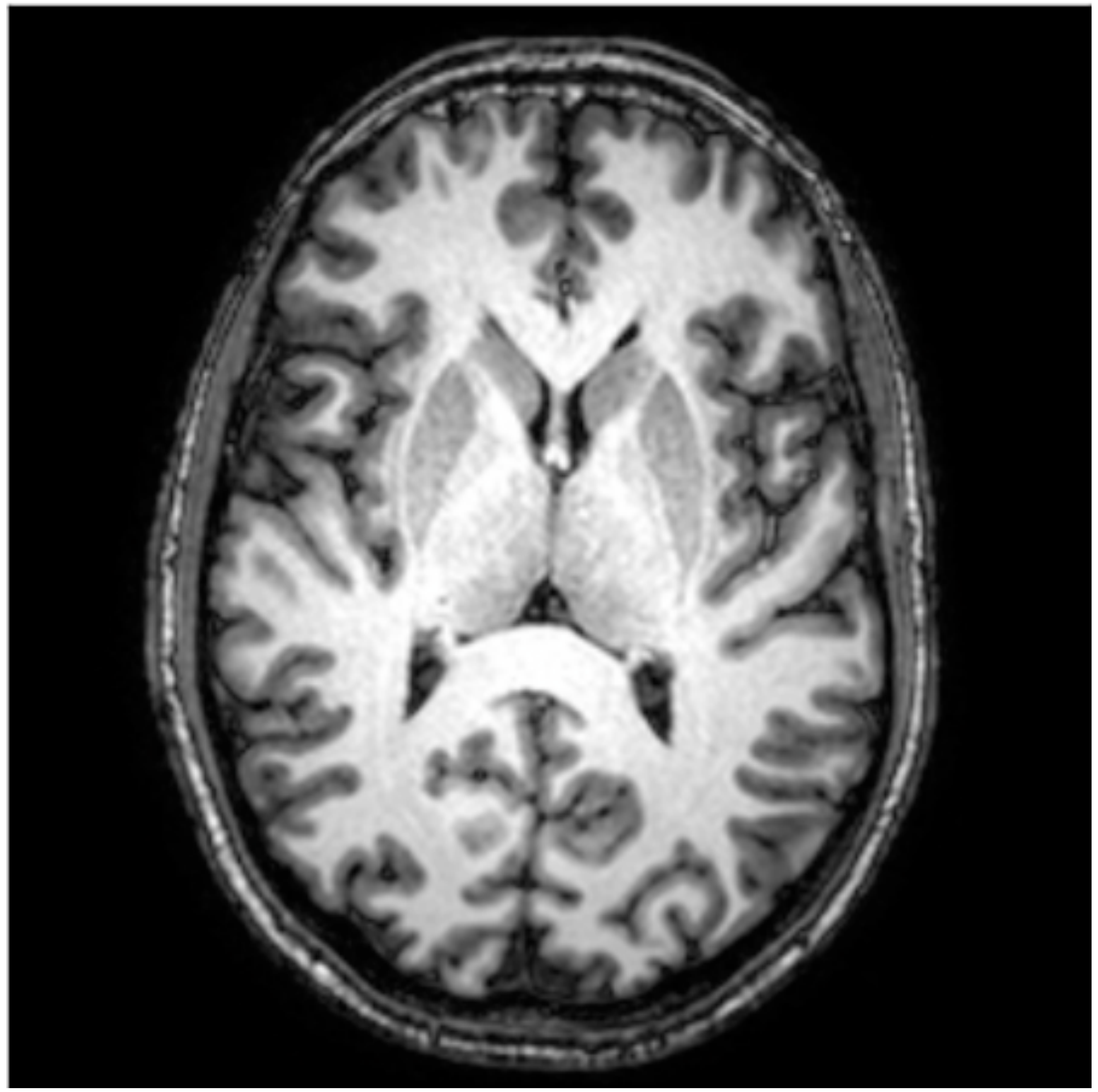
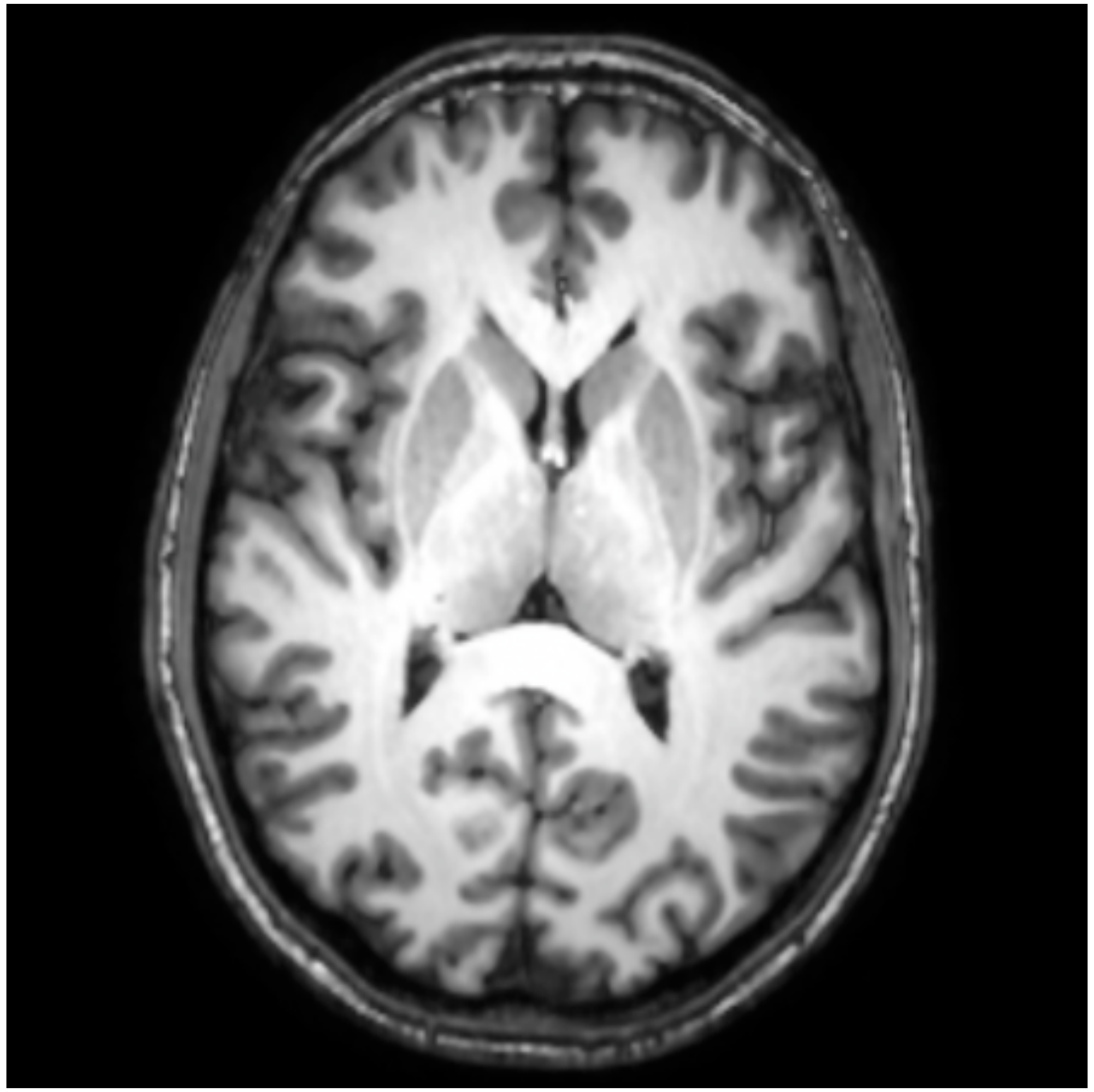


Figure 3.20 Bilateral filtering (Durand and Dorsey 2002) © 2002 ACM: (a) noisy step edge input; (b) domain filter (Gaussian); (c) range filter (similarity to center pixel value); (d) bilateral filter; (e) filtered step edge output; (f) 3D distance between pixels.

Anisotropic Diffusion





Iteratively diffuse (blur) based on neighbor similarity

Coming up...

- More neighborhood operations:
    - sharpening
    - edge detection