Non-Linear Spatial Filtering

Digital Image Processing
CS 4650/7650, ECE 4655/7655

Instructor: Filiz Bunyak Ersoy
TA: Sara Shojaei (ECE PhD student)

Image Processing

Three classes of operations that are most commonly used:

1. Intensity transformation (Point operators/processes),
2. Local image filtering,
    1. Linear Filtering
    2. Non-Linear Filtering
    3. Morphological Filtering
3. Geometrical transformation

Nonlinear Spatial Filters

	Linear Filters	NonLinear Filters
Neighborhood Operation	√	√
Sliding Window Mechanism	√	√
Operations	Sum(Products) Linear	Rank, min, max … NonLinear
Examples	Averaging Filter, Gaussian (Also Sobel, Previtt… like edge detector)	Median, Bilateral, Morphology

Noise Removal

Option 1: Linear Filters Gaussian or averaging filter

- Remove high frequency parts:
- Reduce noise
- Smooth edges

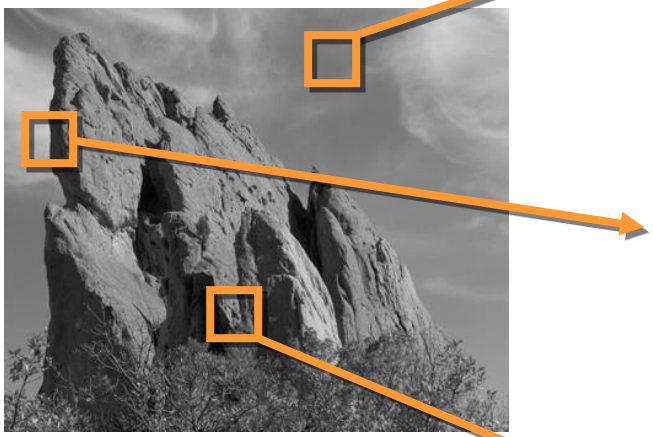
Option 2: Non-linear filters like Median, Bilateral filter…

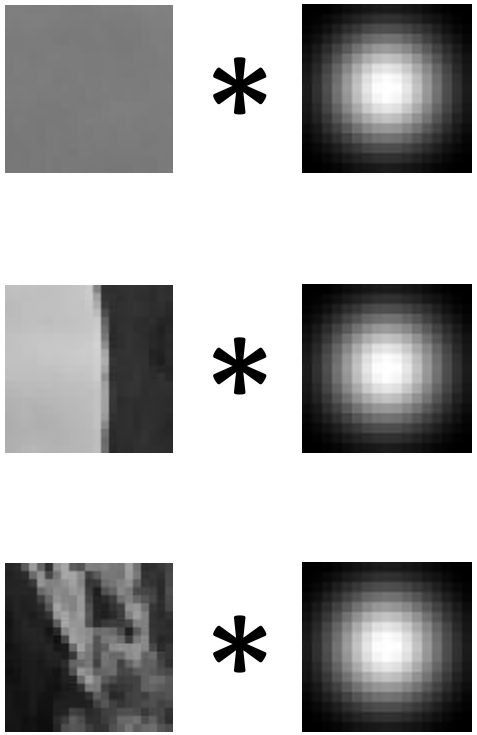
- Remove noise
- Keep edges
- But may still remove corners - no image model

SPATIALLY VARYING FILTERS
BILATERAL FILTER

Constant blur

input





output

Same Gaussian kernel everywhere.

Slides courtesy of Sylvian Paris

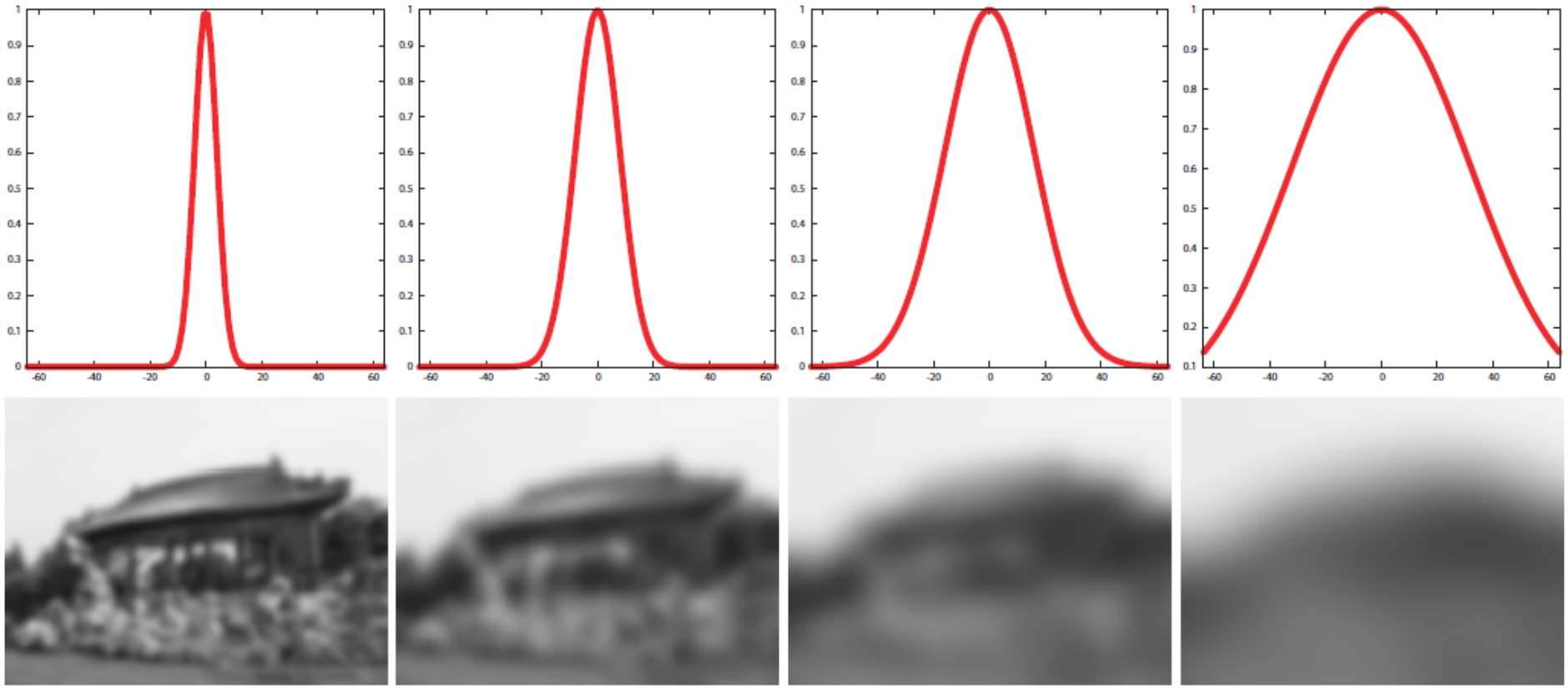
Bilateral Filter

- A **bilateral filter** is a non-linear, edge-preserving, and noise-reducing smoothing filter for images.

References:

- Tomasi, C; Manduchi, R (1998). *Bilateral filtering for gray and color images* (PDF). Sixth International Conference on Computer Vision. Bombay. pp. 839–846. doi:10.1109/ICCV.1998.710815. ---- 11,859 citations
- Paris, Sylvain, Pierre Kornprobst, Jack Tumblin, Frédo Durand . "A gentle introduction to bilateral filtering and its applications." In *ACM SIGGRAPH 2007 courses*, pp. 3-es. 2007.
- A Gentle Introduction to Bilateral Filtering and its Applications by Sylvain Paris, Pierre Kornprobst, Jack Tumblin, and Frédo Durand
  https://people.csail.mit.edu/sparis/bf_course/
  - A class at ACM SIGGRAPH 2008
  - A tutorial at IEEE CVPR 2008
  - A course at ACM SIGGRAPH 2007

Gaussian Filter



$\sigma = 4$ $\sigma = 8$ $\sigma = 16$ $\sigma = 32$

Figure 3: Example of Gaussian linear filtering with different σ. Top row show the profile of a 1D Gaussian kernel and bottom row the result obtained by the corresponding 2D Gaussian blur biltering. Edges are lost with high values of σ since more averaging is performed.

Bilateral Filter – Key ideas

- Spatially varying filter!
- Use the idea of weighted filter kernel.
- Reject (in a soft way) pixels whose values differs too much from the central pixel.

- Weighting coefficient w(i,j,k,l) depends on
  - domain kernel (as before distance to the central pixel)
  - data dependent range kernel that measures intensity (appearance) similarity to the central pixel.

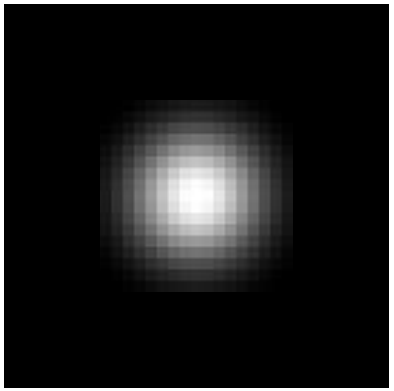
Bilateral Filter Definition: an Additional Edge Term

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

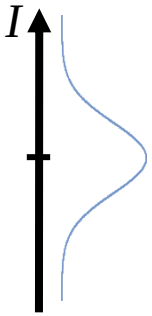
$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|) G_{\sigma_r}(|I_{\mathbf{p}}-I_{\mathbf{q}}|) I_{\mathbf{q}}$$

normalization factor

space weight



range weight

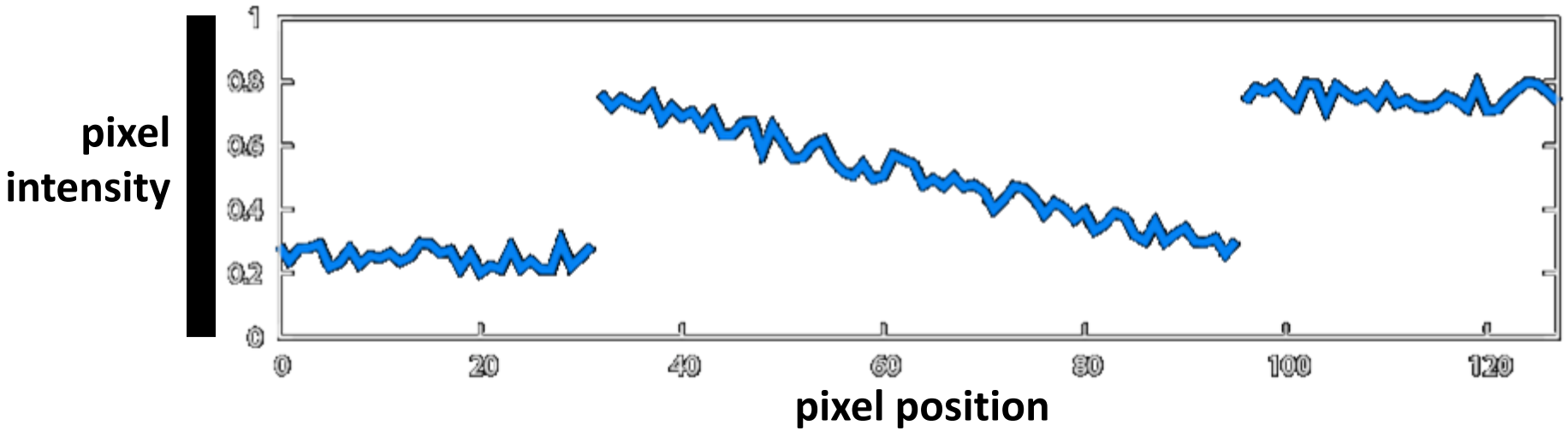


Use the idea of weighted filter kernel.
Reject (in a soft way) pixels whose Values differs too much from the central pixel

Illustration a 1D Image

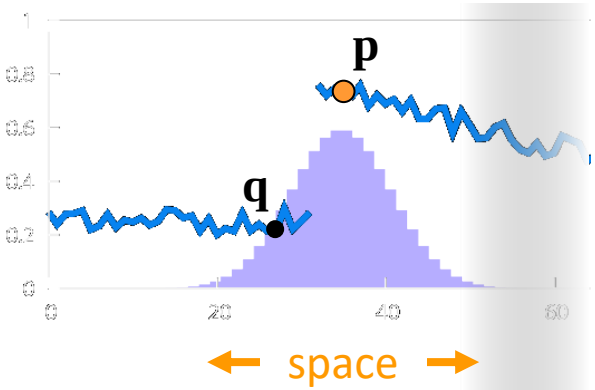
- 1D image = line of pixels

- Better visualized as a plot



Gaussian Blur and Bilateral Filter

Gaussian blur

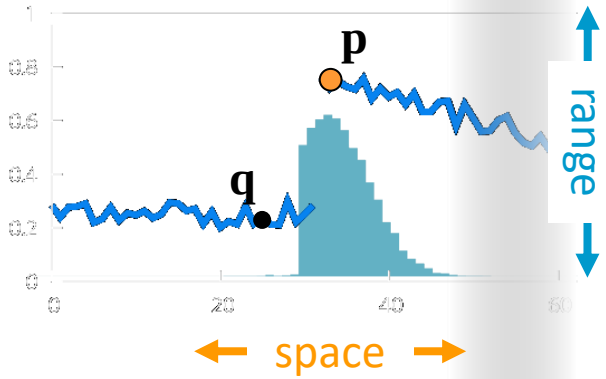


$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q}\in S} G_{\sigma}\left(\|\mathbf{p}-\mathbf{q}\|\right) I_{\mathbf{q}}$$

space

Bilateral filter

[Aurich 95, Smith 97, Tomasi 98]



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

space range

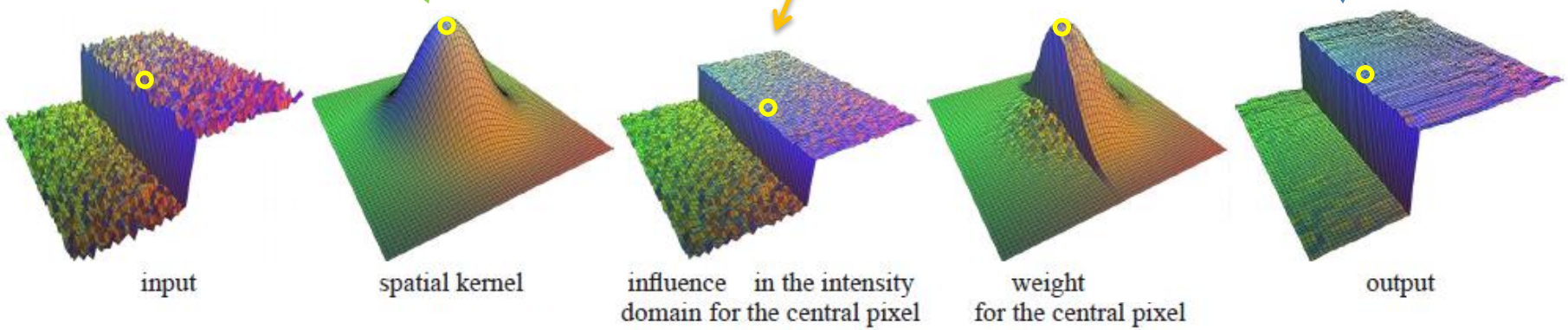
normalization

Ali Farhadi

Spatially varying filters

Some filters vary spatially.

$$\sum_{x'}\sum_{y'} \mathcal{G}_\sigma(x',y')\mathcal{G}_{\sigma'}(f(x,y)-f(x+x',y+y')) = g(x,y)$$



Durand, 02

Bilateral filter

Maintains edges when blurring!

Bilateral Filter

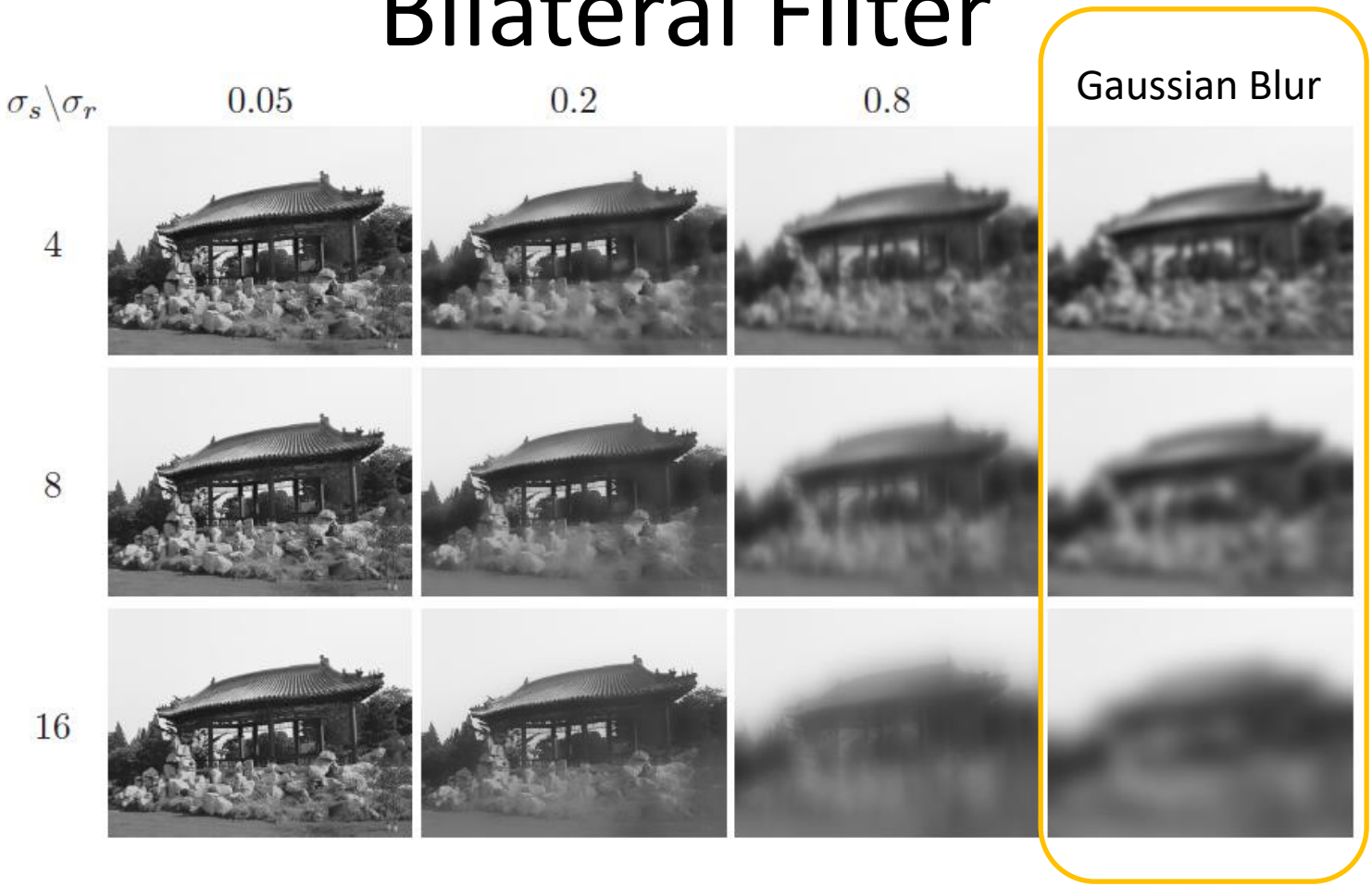
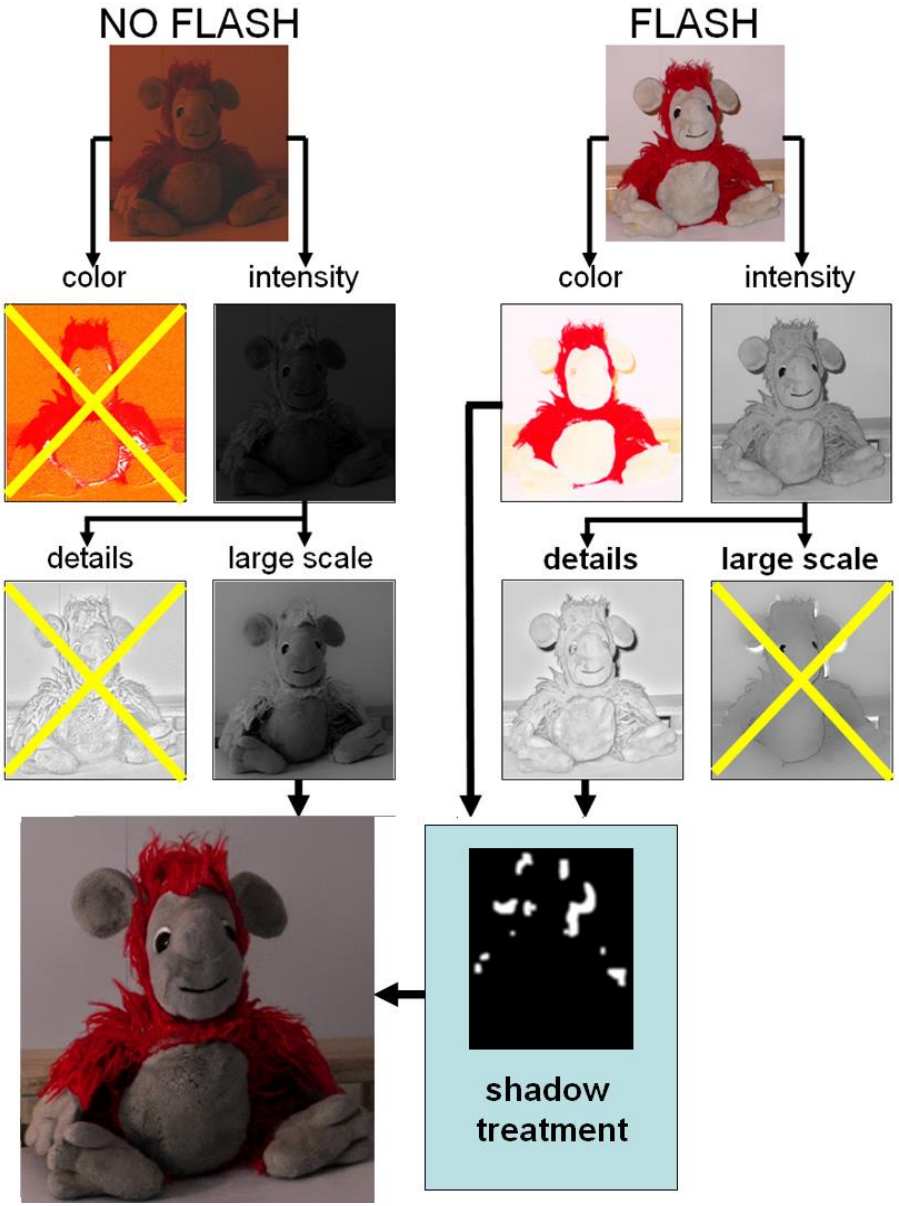


Figure 6: Effects of the range and spatial parameters, and comparison with Gaussian blur. As soon as one of the weight is close to 0, no smoothing occurs. As a consequence, increasing the spatial sigma has no consequence on a edge as long as the range sigma is less than its amplitude. For instance, the contour of the roof is unaffected for small range values, independently of the spatial setting. The range values are given considering that the intensities span $[0, 1]$.

Other Applications of Bilateral Filter



Denoising of low-light images:
Overview of the flash / no-flash combination of Eisemann and Durand [2004].

The bilateral filter is used to combine

- the illumination component of the no-flash picture and
- the structure component of the flash picture.

Figure reproduced from:
Flash photography enhancement via intrinsic relighting *Eisemann et al.* ACM SIGGRAPH
conference (c) 2004, Association for Computing Machinery, Inc. Reprinted by permission.
http://doi.acm.org/10.1145/1186562.1015778

(a) photograph with flash

(b) photograph without flash

(c) combination

Figure 12: Denoising of low-light images: By combining a flash photograph (a) and a no-flash photograph (b), Eisemann and Durand render a new photograph (c) that has both the warm lighting of the no-flash picture and the crisp details of the flash image. Figure reproduced from: Flash photography enhancement via intrinsic relighting *Eisemann et al.* ACM SIGGRAPH conference (c) 2004, Association for Computing Machinery, Inc. Reprinted by permission. `http://doi.acm.org/10.1145/1186562.1015778`

(a) flash picture

(b) no-flash picture

(c) output of [Petschnigg *et al.*, 2004]

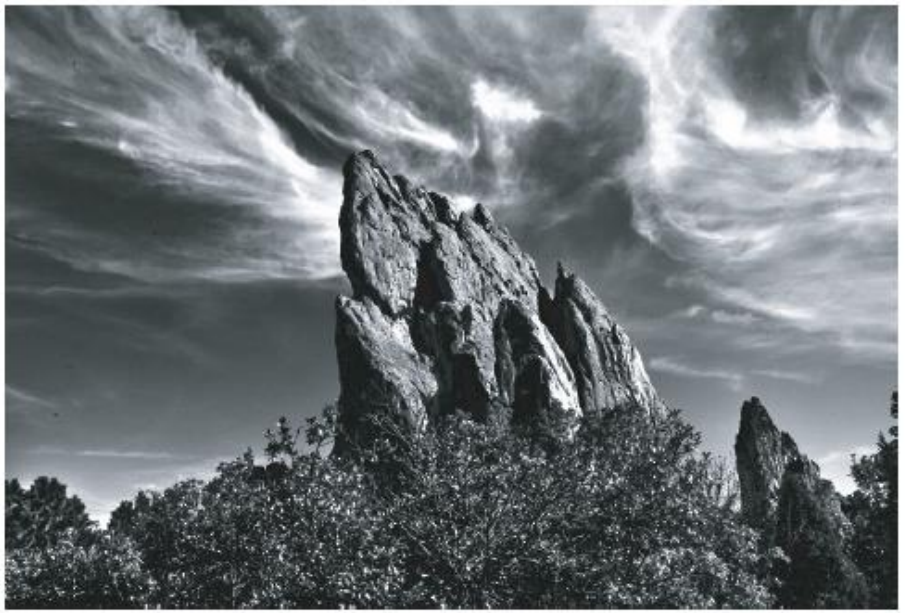
Figure 13: By combining a flash photograph (a) and a no-flash photograph (b), Petschnigg *et al.* render a new photograph (c) that has both the warm lighting of the no-flash picture and the crisp details of the flash image. Figure reproduced from: Digital photography with flash and no-flash image pairs *Petschnigg et al.* ACM SIGGRAPH conference (c) 2004, Association for Computing Machinery, Inc. Reprinted by permission. `http://doi.acm.org/10.1145/1186562.1015777`

(a) input (b) naive compression

(c) compression after Gaussian decomposition (d) output of [Durand and Dorsey, 2002]

Figure 15: Tone Mapping: Direct display of a HDR image (a) is not satisfying because of over- and under-exposed areas. Compressing the intensity values solves this problem but details in clouds and in the city below the horizon are barely visible (b). Isolating the details using Gaussian blur brings back the details but incurs halos near contrasted edges (*e.g.*, near the tree silhouettes) (c). Durand and Dorsey use the bilateral filter to isolate the small variations of the input image without incurring halos (d). Figure reproduced from: Fast bilateral filtering for the display of high-dynamic-range images *Durand and Dorsey* ACM SIGGRAPH conference (c) 2002, Association for Computing Machinery, Inc. Reprinted by permission. `http://doi.acm.org/10.1145/566570.566574`

(a) input



(b) result after after contrast and "textureness" increase

Figure 16: Bae *et al.* [2006] use the bilateral filter to separate and process separately the large-scale and small-scale variation of an image. This example illustrates an increase of the global image contrast and an increase of the amount of texture. Figure reproduced from: Two-scale tone management for photographic look *Bae et al.* ACM SIGGRAPH conference (c) 2006, Association for Computing Machinery, Inc. Reprinted by permission. `http://doi.acm.org/10.1145/1141911.1141935`

Nonlinear Spatial Filters

	Linear Filters	NonLinear Filters
Neighborhood Operation	√	√
Sliding Window Mechanism	√	√
Operations	Sum(Products) Linear	Rank, min, max … NonLinear
Examples	Averaging Filter, Gaussian (Also Sobel, Previtt… like edge detector)	Median, Bilateral, Morphology

Noise Removal

Option 1: Linear Filters Gaussian or averaging filter

- Remove high frequency parts:
- Reduce noise
- Smooth edges

Option 2: Non-linear filters like Median, Bilateral filter…

- Remove noise
- Keep edges
- But may still remove corners - no image model

Common types of noise

- **Salt and pepper noise**: random occurrences of black and white pixels
- **Impulse noise:** random occurrences of white pixels
- **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution

Original

Salt and pepper noise

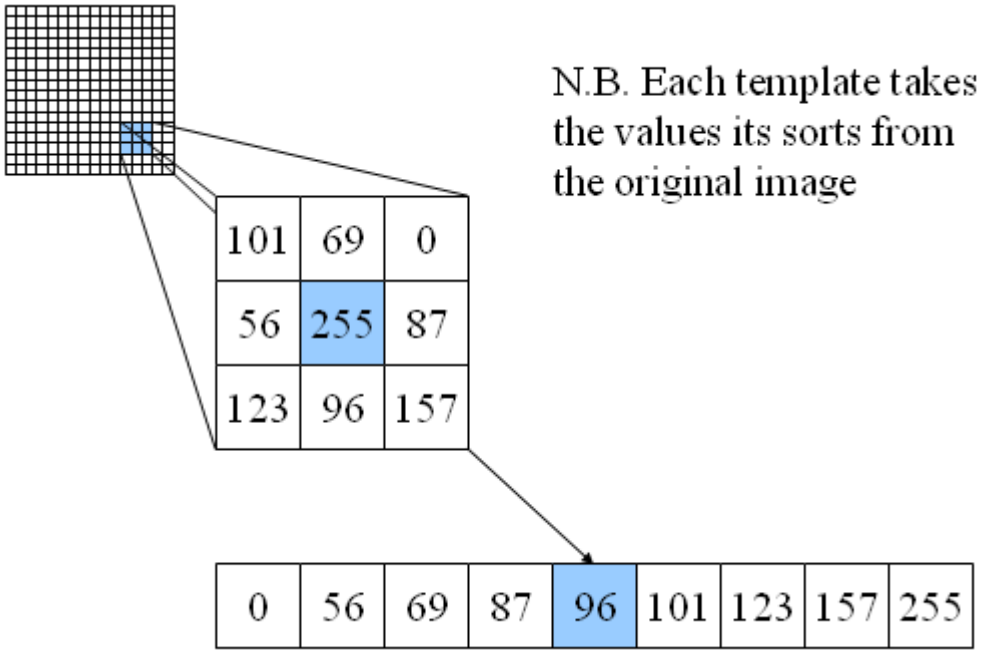
Impulse noise

Gaussian noise

Source: S. Seitz

Median Filter

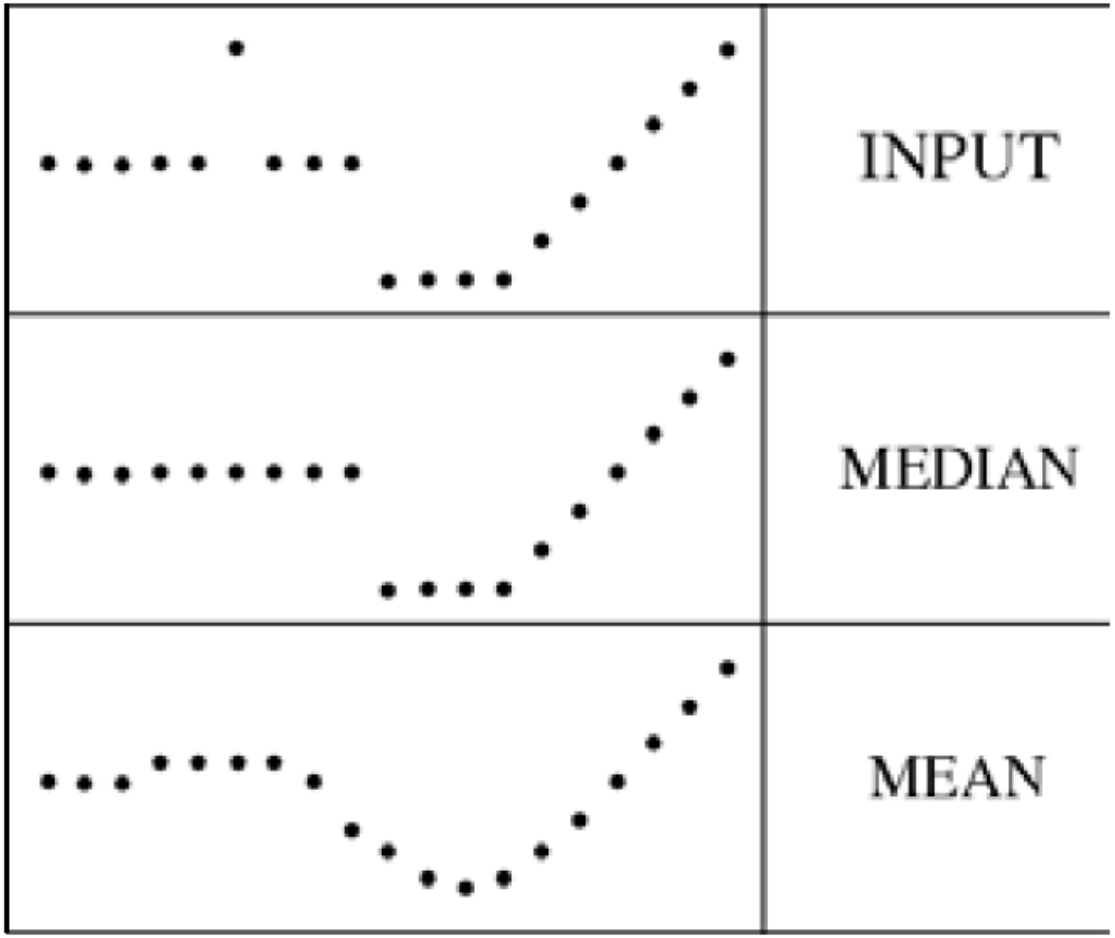
- Median filtering is used to remove "salt and pepper" noise.
- The template size defines how much filtering takes place.
- Median filtering will not remove gaussian noise.



Is median filter a kind of convolution?

Median filter

- Median filter is edge preserving



S. Grauman

Median and Bilateral Filtering

Original image
with Gaussian Noise

Gaussian Filter

Median Filter

Bilateral Filter

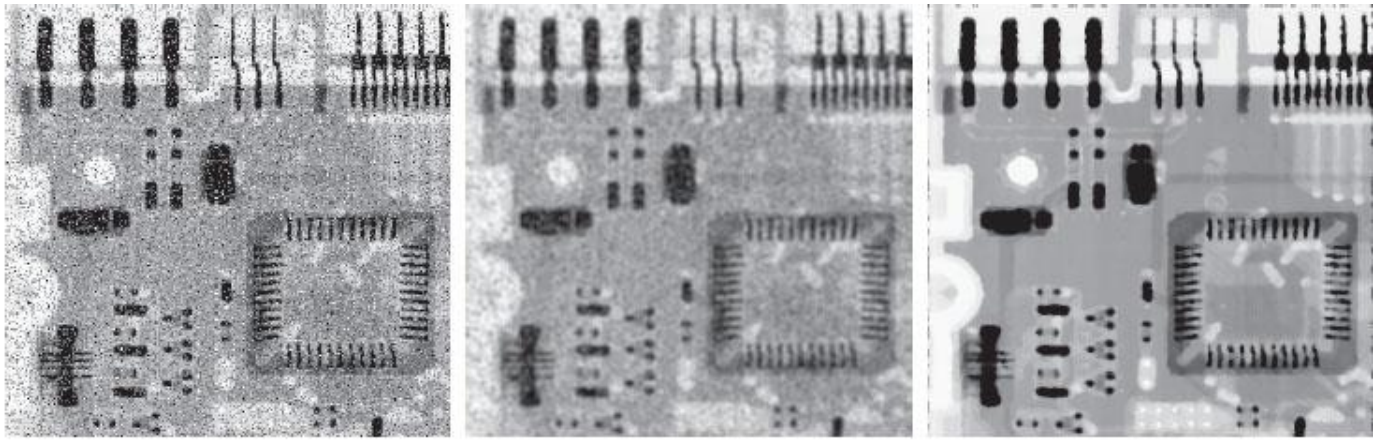
Original image
with shot noise

Noise Removal

(a) X-ray image of a circuit board, corrupted by salt-and-pepper noise.
(b) Noise reduction using a 19x19 Gaussian filter (sigma=3)
(c) Noise reduction using 7x7 median filter.

Gonzalez & Woods Figure 3.49

(Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)



a b c

Some neighborhood operations

(a) original image

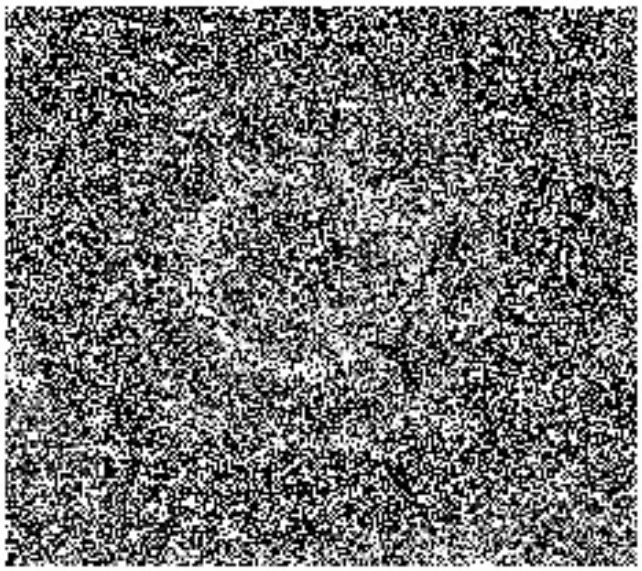
(b) blurred

(c) sharpened

(d) smoothed with edge-preserving filter

Median Filter under High Noise

IMnoise80 = imnoise(IM, 'salt & pepper', 0.80);



Median Filter under High Noise

Matlab:

```
IMnoise80 = imnoise(IM, 'salt & pepper', 0.80);
IMnoise80_med3 = colfilt(IMnoise80, [3,3],'sliding', @median);
```

SciPy:

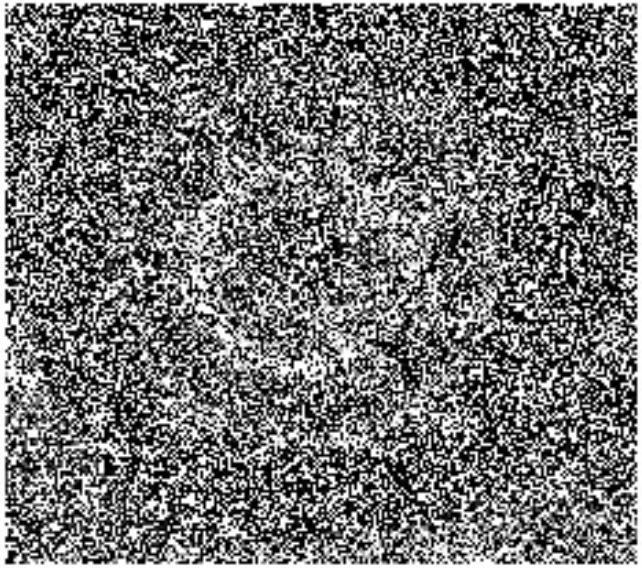
```
scipy.signal.medfilt2d(input, kernel_size=3)
```

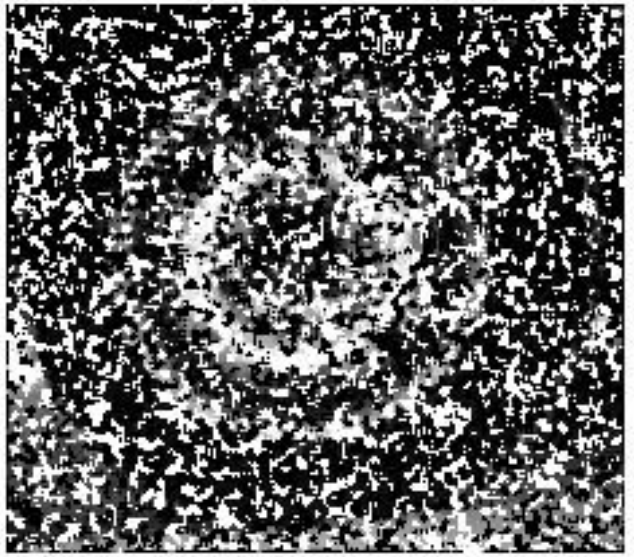
OpenCV:

```
median = cv2.medianBlur(img,3)
```

Other functions you can use:

- medfilt2(I,[3,3])
- blockproc(I,[3,3],fun)





Median Filter under High Noise

Matlab:

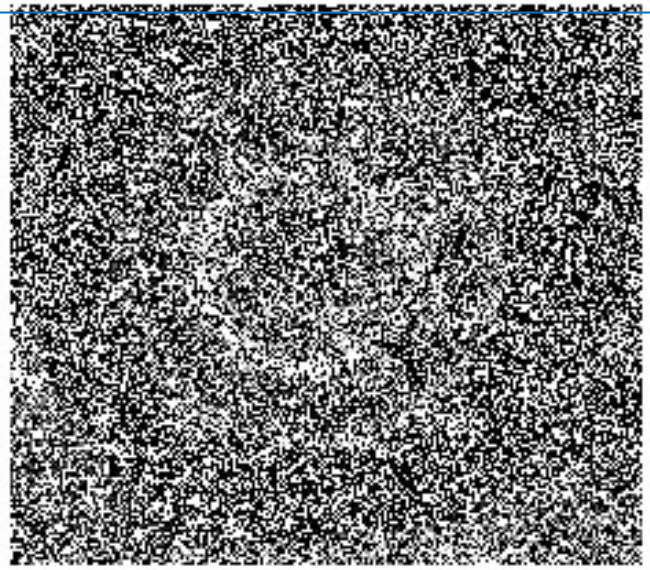
```
IMnoise80 = imnoise(IM, 'salt & pepper', 0.80);
IMnoise80_med3 = colfilt(IMnoise80, [5,5],'sliding', @median);
```

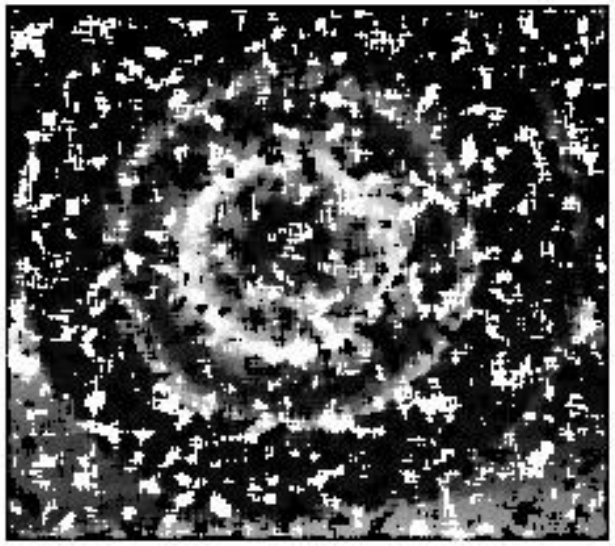
SciPy:

```
scipy.signal.medfilt2d(input, kernel_size=5)
```

OpenCV:

```
median = cv2.medianBlur(img,5)
```





Median Filter under High Noise

Matlab:

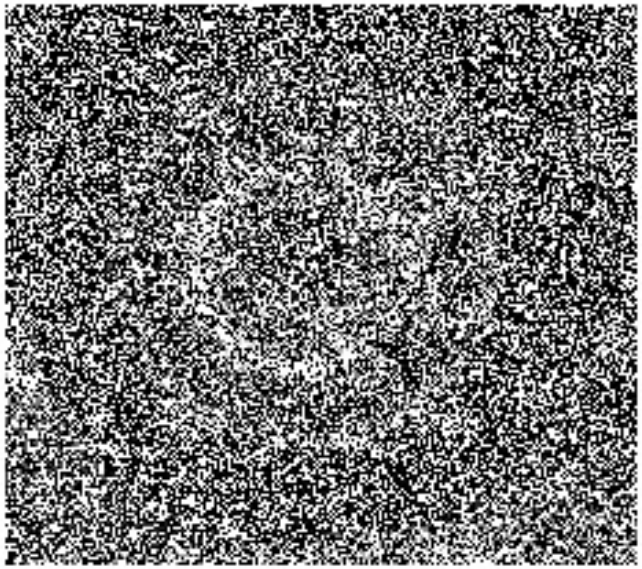
```
IMnoise80 = imnoise(IM, 'salt & pepper', 0.80);
IMnoise80_med3 = colfilt(IMnoise80, [7,7],'sliding', @median);
```

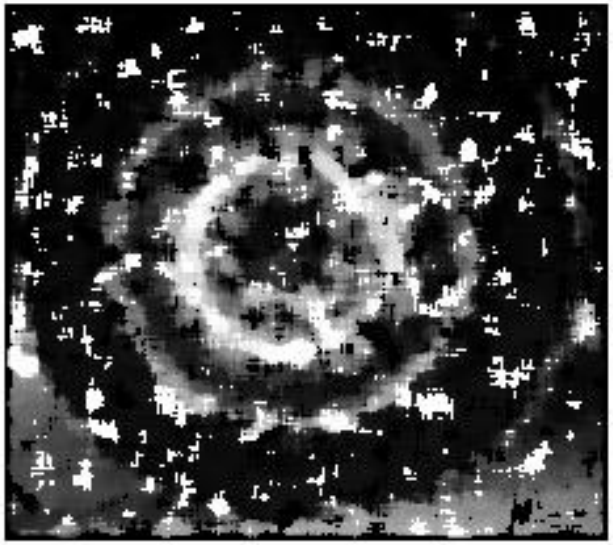
SciPy:

```
scipy.signal.medfilt2d(input, kernel_size=7)
```

OpenCV:

```
median = cv2.medianBlur(img,7)
```





Median Filter under High Noise

Matlab:

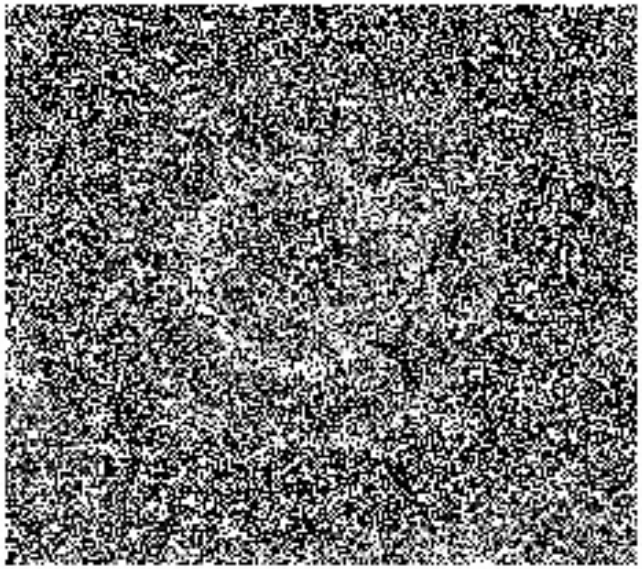
```
IMnoise80 = imnoise(IM, 'salt & pepper', 0.80);
IMnoise80_med3 = colfilt(IMnoise80, [9,9],'sliding', @median);
```

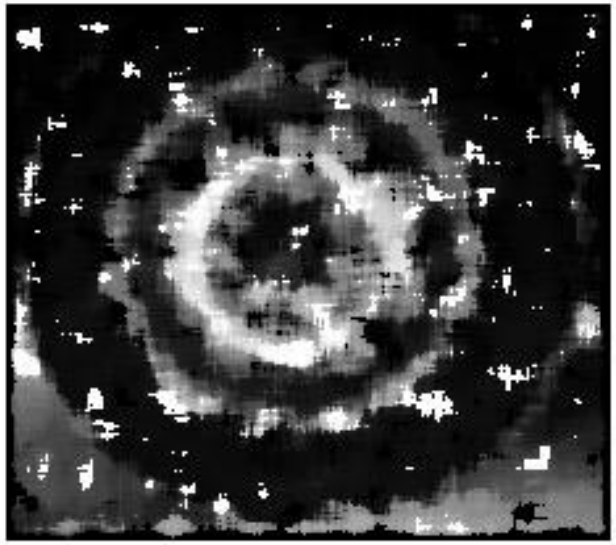
SciPy:

```
scipy.signal.medfilt2d(input, kernel_size=9)
```

OpenCV:

```
median = cv2.medianBlur(img,9)
```





Median Filter under High Noise

Matlab:

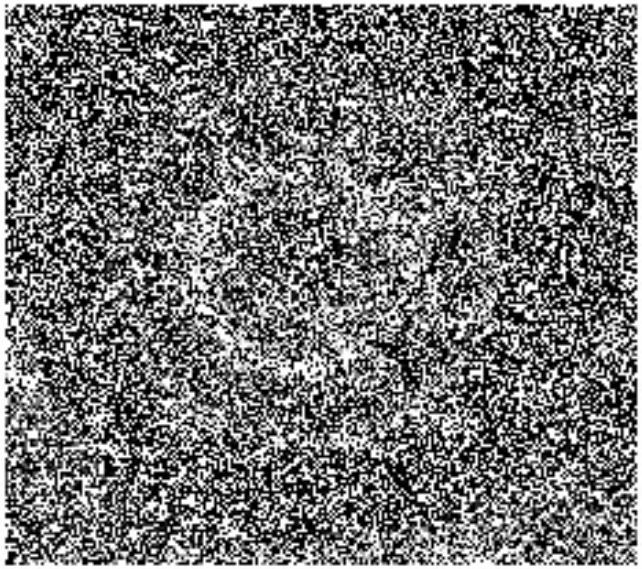
```
IMnoise80 = imnoise(IM, 'salt & pepper', 0.80);
IMnoise80_med3 = colfilt(IMnoise80, [11,11],'sliding', @median);
```

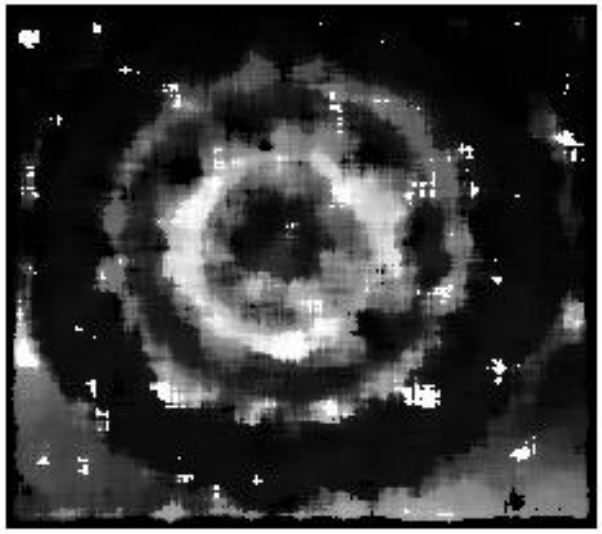
SciPy:

```
scipy.signal.medfilt2d(input, kernel_size=11)
```

OpenCV:

```
median = cv2.medianBlur(img,11)
```





Median Filter under High Noise

Matlab:

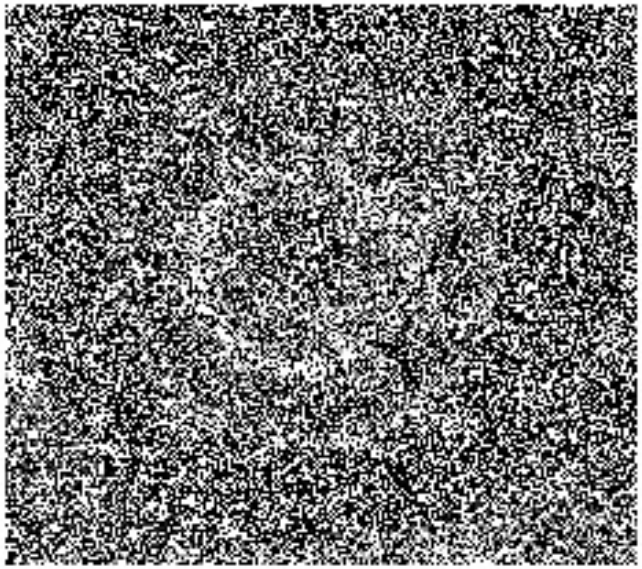
```
IMnoise80 = imnoise(IM, 'salt & pepper', 0.80);
IMnoise80_med3 = colfilt(IMnoise80, [13,13],'sliding', @median);
```

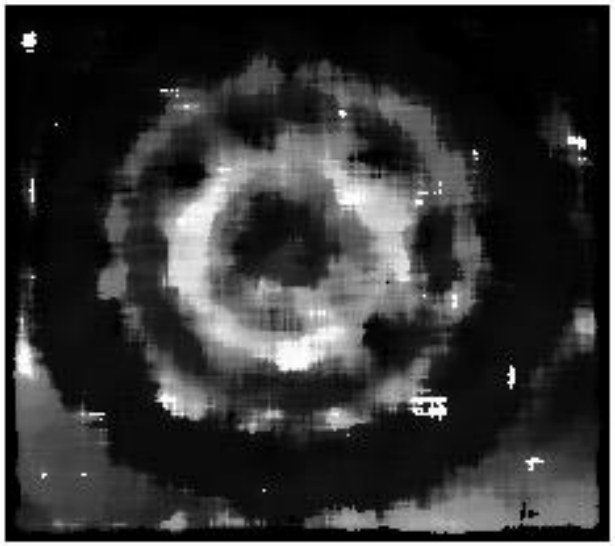
SciPy:

```
scipy.signal.medfilt2d(input, kernel_size=13)
```

OpenCV:

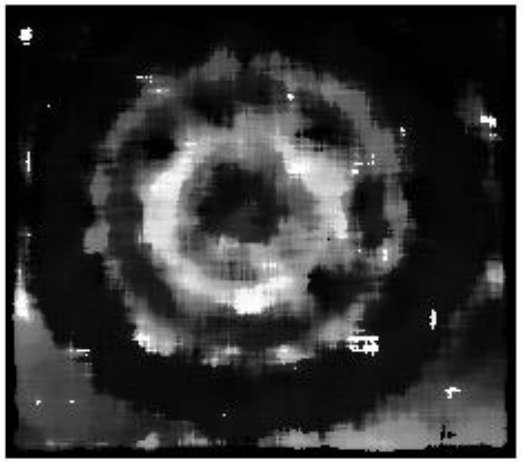
```
median = cv2.medianBlur(img,13)
```

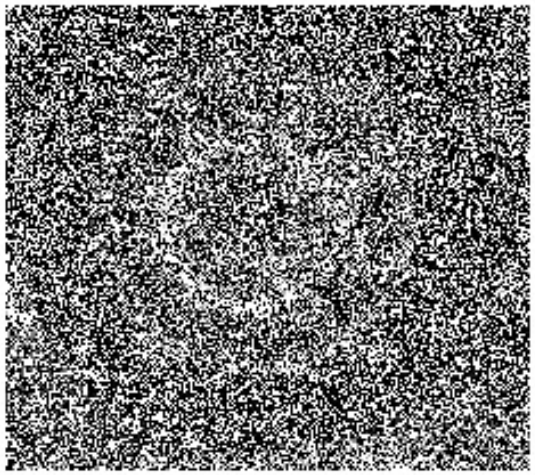




Median Filter under High Noise

Median Filter (Fixed) 13x13



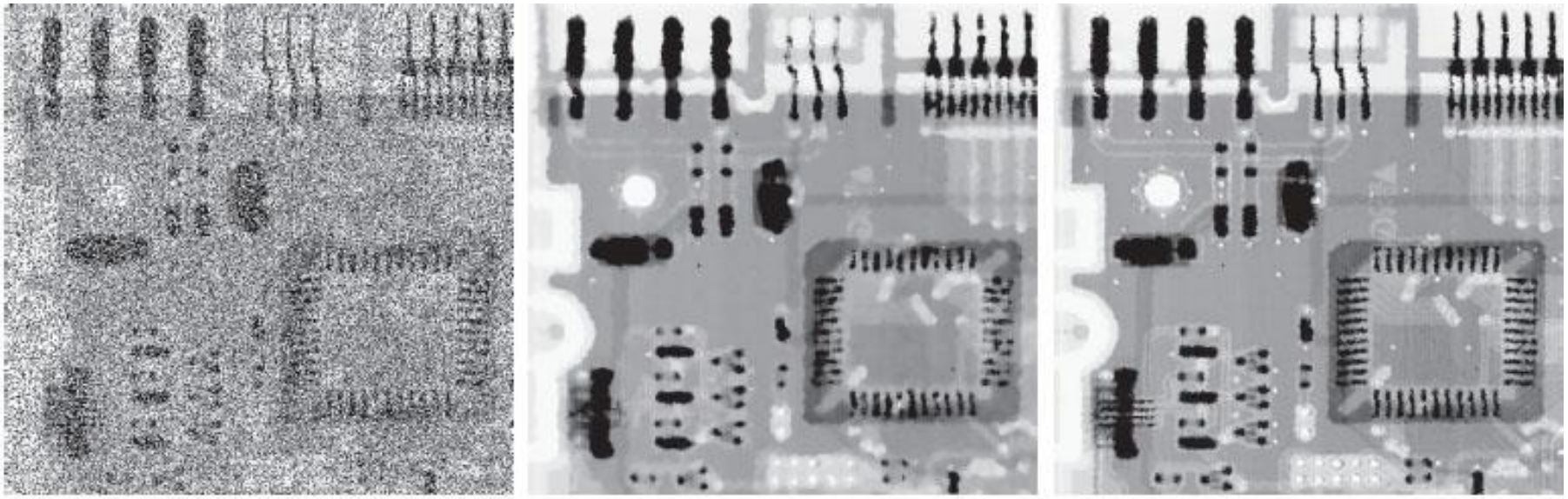


Adaptive Median Filter

Adaptive Median Filter

Gonzalez & Woods Figure 5.14

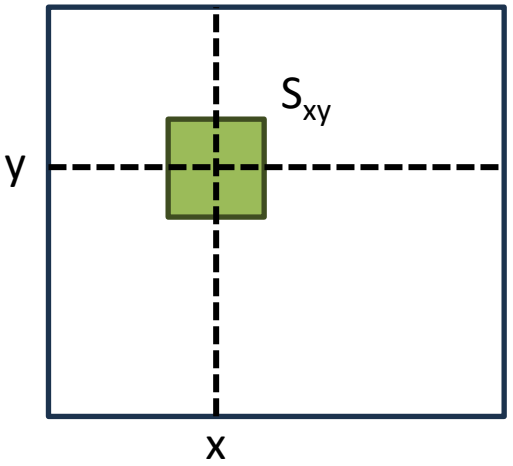
(a) Image corrupted by salt-and-pepper noise with probabilities $P_s = P_p = 0.25$. (b) Result of filtering with a 7×7 median filter. (c) Result of adaptive median filtering with $S_{\max} = 7$.



a b c

Adaptive Median Filter (Gonzalez & Woods 5.3)

Fundamental idea : increase the size of the neighborhood S_{xy} until we are sufficiently sure that z_{xy} is impulsive or not.
If z_{xy} is impulsive then output the median otherwise output z_{xy}



S_{xy}	window centered at (x,y)
S_{max}	maximum allowed size of S_{xy}
z_{min}	minimum gray value in S_{xy}
z_{max}	maximum gray value in S_{xy}
z_{med}	median gray value in S_{xy}
z_{xy}	gray level value at (x,y)

Stage A: IF $z_{med} > z_{min}$ and $z_{max} > z_{med}$
THEN goto **Stage B**
ELSE increase the window size S_{xy}
IF WindowSize $\leq S_{max}$
THEN goto **Stage A**
ELSE output z_{med}

Stage B: IF $z_{xy} > z_{min}$ and $z_{max} > z_{xy}$
THEN output z_{xy}
ELSE output z_{med}

Adaptive Median Filter

Median Filter

- Performs relatively well on impulse noise as long as the spatial density of the impulse noise is not large (~ Ps<0.2 Pp<0.2)
- Replaces every point in the image with neighborhood median

  => unnecessary loss of details!

Adaptive Median Filter

- Can handle much more spatially dense impulse noise
- Also performs some smoothing for non-impulse noise
- Similar to other filters we discussed so far, works with a rectangular neighborhood Sxy centered at pixel (x,y).
- Output of the filter is a single value that will replace value of the pixel at (x,y)
- **Key insight:** filter size changes depending on the characteristics of the image

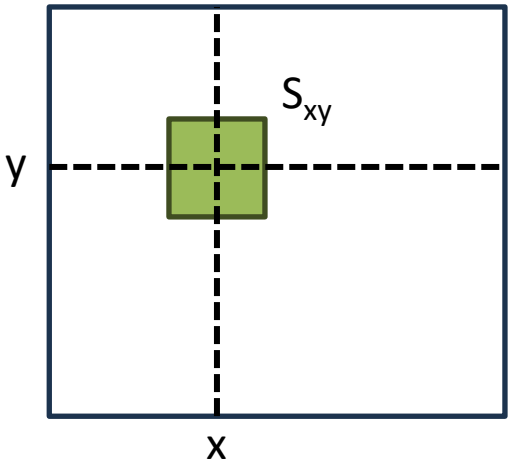
Objectives

1. Remove salt-and-pepper noise (impulse noise)
2. Smooth other noise that may be impulsive
3. Reduce distortion --- excessive thinning or thickening of object boundaries

Adaptive Median Filter (Gonzalez & Woods 5.3)

Fundamental idea : increase the size of the neighborhood S_{xy} until we are sufficiently sure that z_{xy} is impulsive or not.
If z_{xy} is impulsive then output the median otherwise output z_{xy}

S_{xy}	window centered at (x,y)
S_{max}	maximum allowed size of S_{xy}
z_{min}	minimum gray value in S_{xy}
z_{max}	maximum gray value in S_{xy}
z_{med}	median gray value in S_{xy}
z_{xy}	gray level value at (x,y)



Stage A:
IF $z_{med}>z_{min}$ and $z_{max}>z_{med}$
THEN goto **Stage B**
ELSE increase the window size S_{xy}
IF WindowSize $\leq S_{max}$
THEN goto **Stage A**
ELSE output z_{med}

Stage B:
IF $z_{xy}>z_{min}$ and $z_{max}>z_{xy}$
THEN output z_{xy}
ELSE output z_{med}

If $z_{max}>z_{med}>z_{min}$ then z_{med} is NOT an impulse and we go to Stage B. Otherwise, Stage A continues to increase the neighborhood S_{xy} until z_{med} is not an impulse.

If $z_{max}>z_{xy}>z_{min}$ then z_{xy} is NOT an impulse and we output z_{xy}
otherwise
we output the median z_{med}.

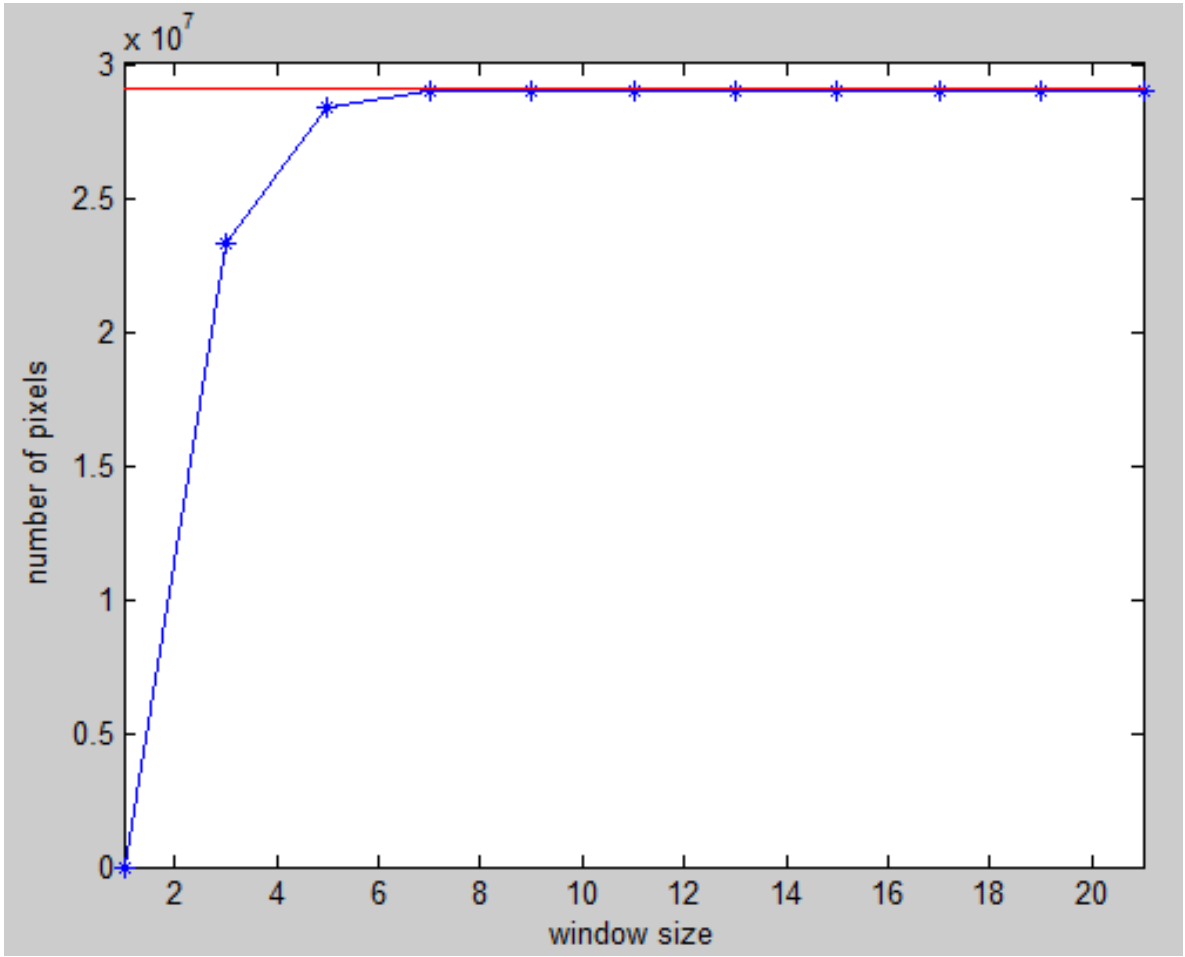
Implementation Issues

1. Finding z_{min}, z_{max}, z_{med} expensive for large neighborhoods S_{xy}
2. Loops are expensive particularly for Matlab

Hints

1. There are large overlaps in S_{xy} for neighboring pixels. Take advantage of this overlap.
2. Rather than trying different neighborhood sizes for a pixel (x,y) than moving to the next point, consider locating z_{min}, z_{max}, z_{med} for all pixels for a given window size. (When is this suitable ? When it is not?)
3. Keep track of finalized pixels at each iteration.
4. Consider using sliding block functions i.e. blockproc, nfilter,colfilt in Matlab
5. Morphological filters erode and dilate may serve as min, max filters

Window size versus Number of Pixels Reconstructed



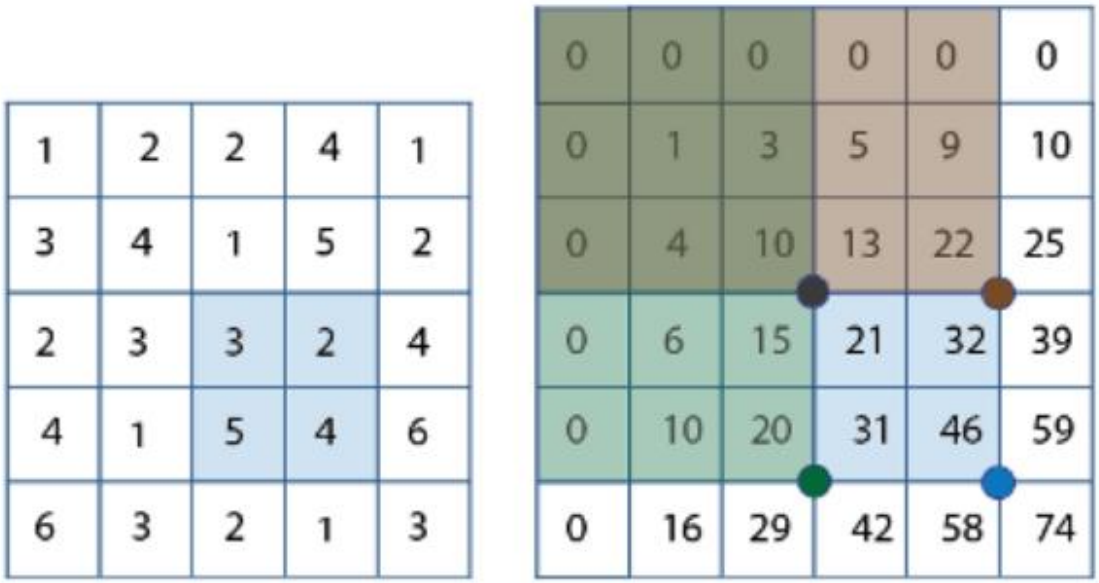
ABQ_Noise60

	Wmax 21	Wmax 9
Time	667sec	82sec
MSE	52.42	53.42
PSNR	30.93	30.85

Early stopping when 0.1% of pixels are left.

Integral Image

In an *integral image*, every pixel is the summation of the pixels above and to the left of it.



1	2	2	4	1
3	4	1	5	2
2	3	3	2	4
4	1	5	4	6
6	3	2	1	3

input image

0	0	0	0	0	0
0	1	3	5	9	10
0	4	10	13	22	25
0	6	15	21	32	39
0	10	20	31	46	59
0	16	29	42	58	74

integral image

Image courtesy: https://www.mathworks.com/help/images/integral-image.html

Integral Image (Summed Area Table)

Integral Image Computation

$$I(x,y) = i(x,y) + I(x,y-1) + I(x-1,y) - I(x-1,y-1)$$

Integral image

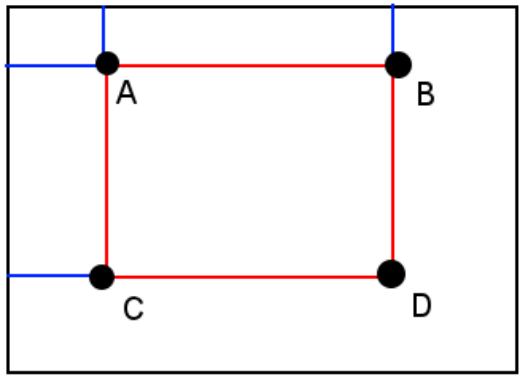
Input image

Given integral image local sum computation

A=(x_0, y_0), B=(x_1, y_0), C=(x_0, y_1) and D=(x_1, y_1),

sum of i(x,y) over the rectangle spanned by A, B, C and D

$$\sum_{\substack{x_0 < x \le x_1 \\ y_0 < y \le y_1}} i(x,y) = I(D) + I(A) - I(B) - I(C)$$

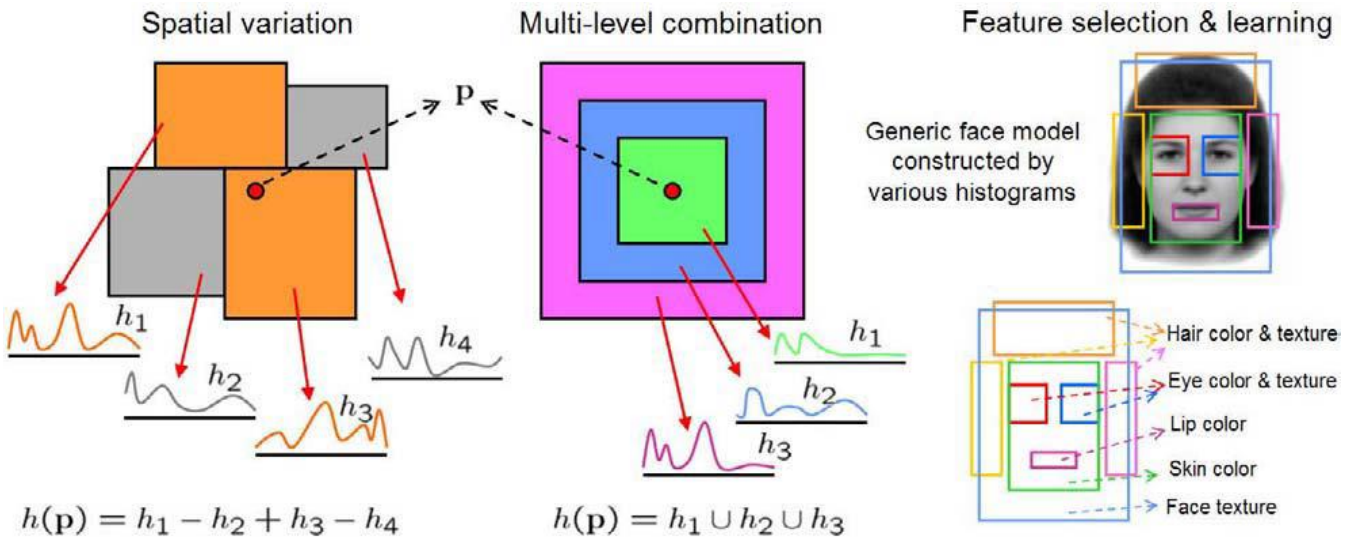


Sum = D - B - C + A

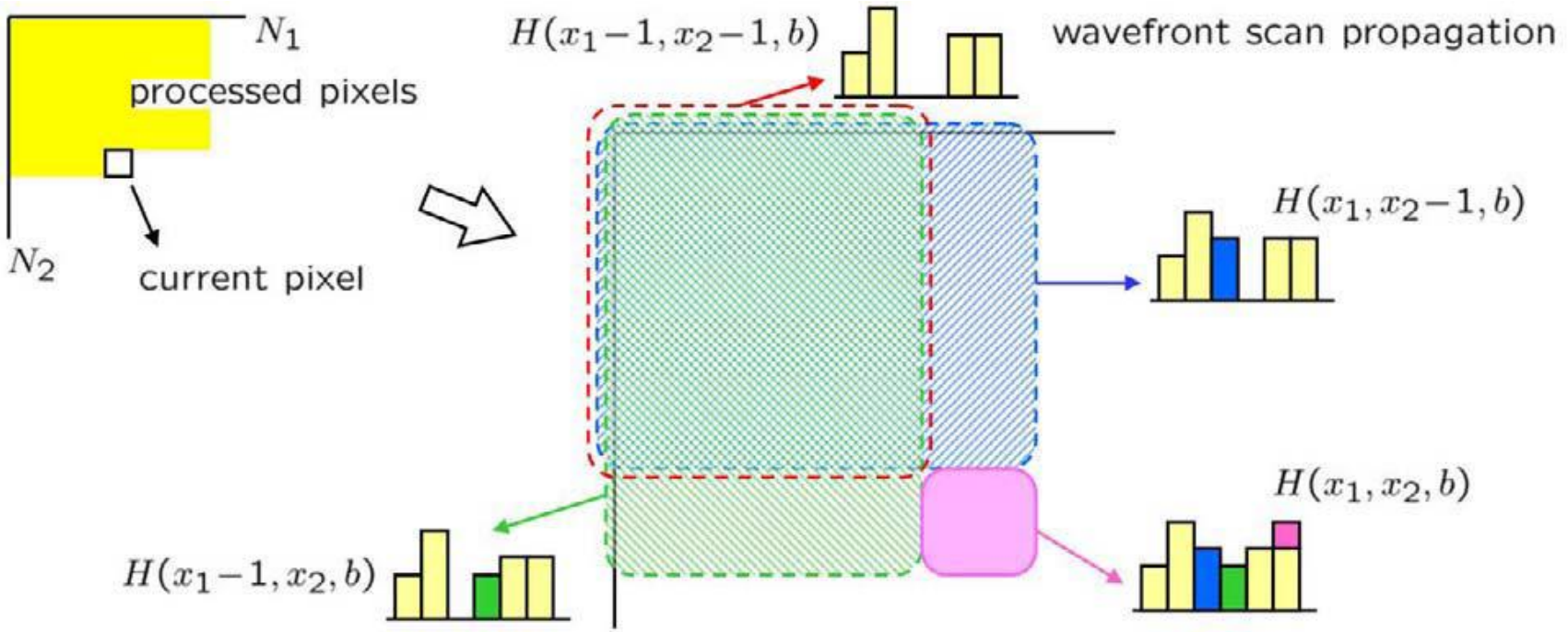
Local sum computation using Integral Image: independent of window size always 4 operations!

Integral Histogram

- Reference: Porikli, Fatih. "Integral histogram: A fast way to extract histograms in cartesian spaces." In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 829-836, 2005.



Computing Integral Histogram



$$H(x_1, x_2, b) = H(x_1-1, x_2, b) + H(x_1, x_2-1, b) - H(x_1-1, x_2-1, b) + Q(f(x_1, x_2))$$

Vectorized Matlab Code

Apply median, min, max operations on a sliding window over the whole Image at once (without looping over each pixel) .

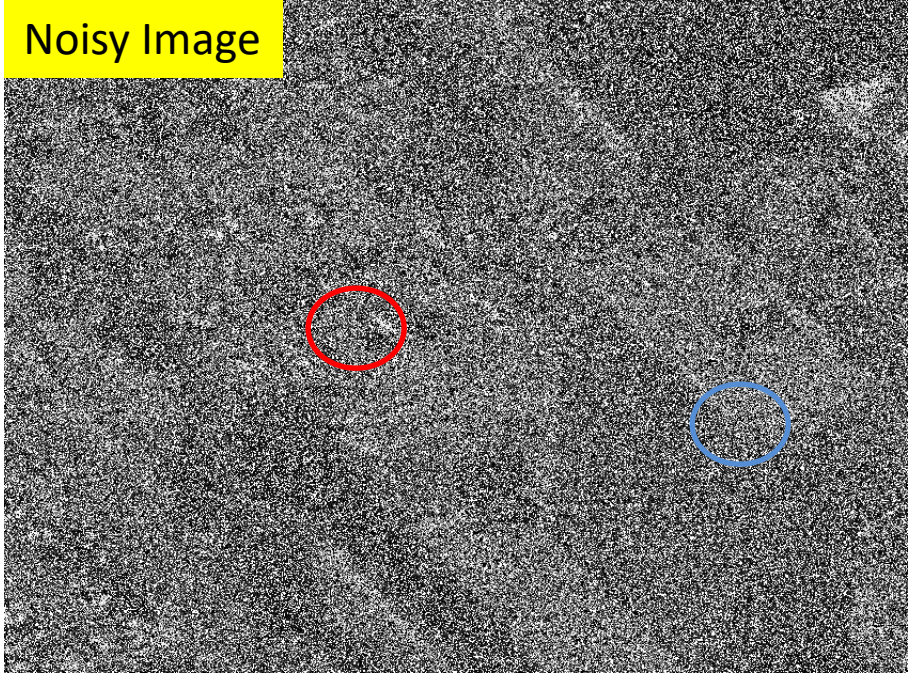
Gray level dilation→ max
Gray level erosion → min

Alternative 1:
Use **colfilt** or **nfilter** (general sliding window operations).
Example:

```
fun = @(x) median(x(:));
B = nlfilter(A,[3 3],fun);
```

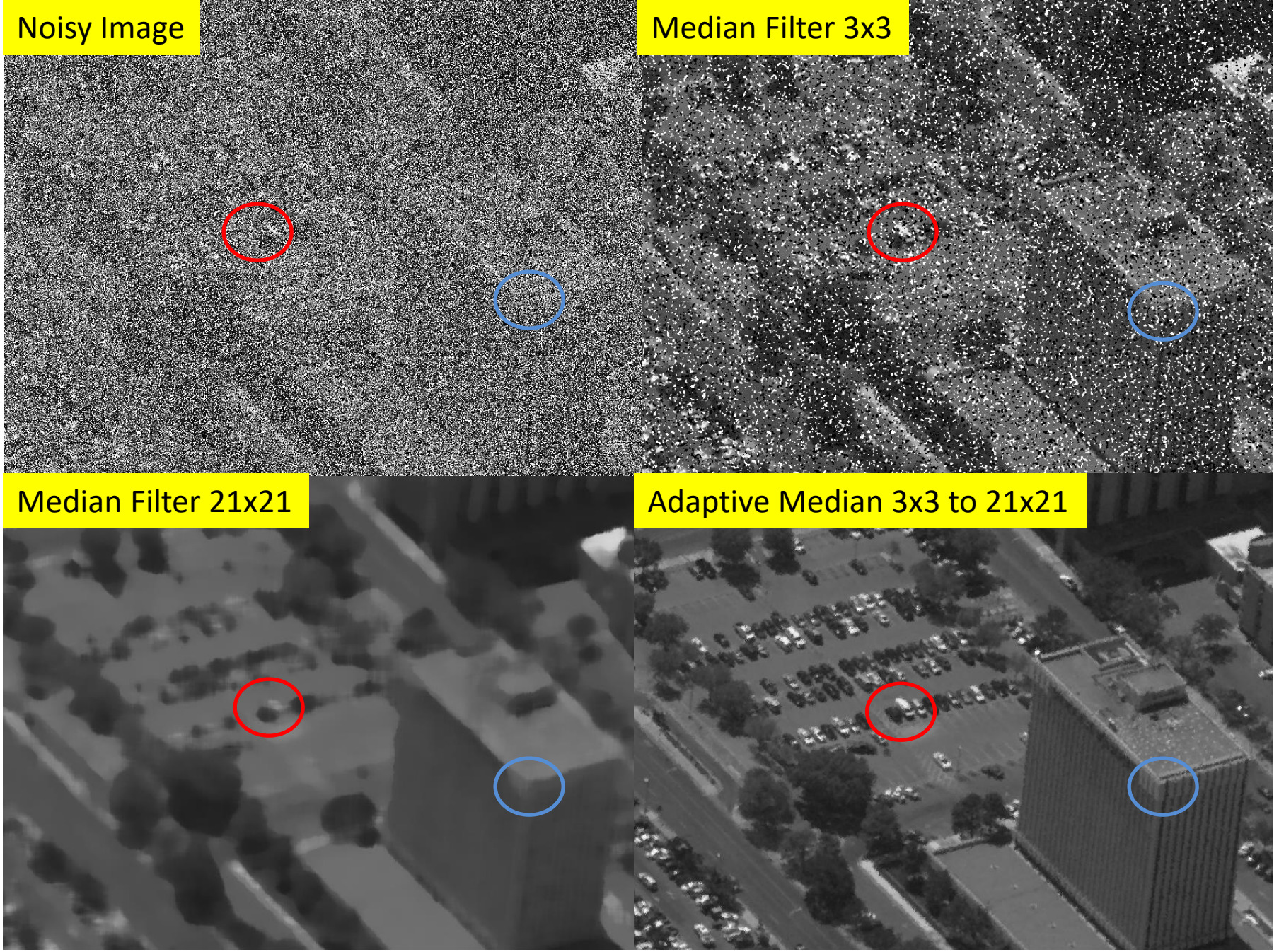
Advantage: Because loops over each pixel is avoided, very efficient for early Iterations.
Disadvantage: Inefficient for later iterations when only small number of pixels should be processed.

Alternative 2:
Write the function in C (with loops),



What do you see within red circle?

What do you see within blue circle?



- Small filters can not handle large noise
- Large filters over smooth the image and looses the details

- Red: car appearance –important for tracking
- Blue: building features – important for 3D reconstruction