Computer Vision for HCI

Noise Removal

1

Noise in Images

- Images can be noisy
  - Image acquisition process not perfect
  - Different sensors can have different noise and distortion properties
- Filter image to
  - Enhance images
  - Reduce noise in image
  - Emphasize or suppress certain image details
- How?
  - Decisions typically made at a level of local pixel regions (neighborhoods)

2

2

Local Pixel Neighborhoods

- 4-connected pixel region
  - If pixel * connected to four immediate neighbors
    - Left, right, up, down
- 8-connected pixel region
  - If pixel * connected to all eight neighbors

-	1	-
1	*	1
-	1	-

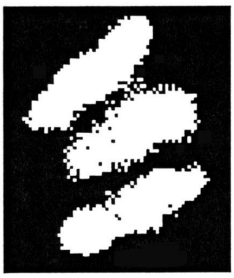
1	1	1
1	*	1
1	1	1

3

3

Simple Removal of Binary Image Noise

- "Salt-and-pepper" noise (binary)
  - Single dark pixels in bright regions
  - Single bright pixels in dark regions
  - Possibly from thresholding errors



Removal of value L isolated from neighborhood of X's

X	X	X
X	**L**	X
X	X	X

8-connected removal of L

→

X	X	X
X	**X**	X
X	X	X

Result

	X	
X	**L**	X
	X	

4-connected removal of L

→

	X	
X	**X**	X
	X	

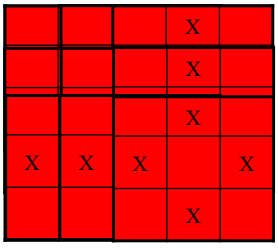
Result

4

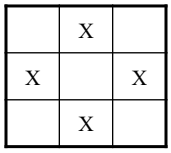
4

Applying to Images

- For *each* valid pixel location, examine neighborhood and save result to ***new*** image



Binary image



4-conn Region

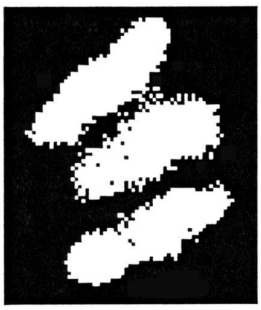
-	-	-	-	-
-	1	1	1	-
-	1	1	1	-
-	1	1	1	-
-	-	-	-	-

Result

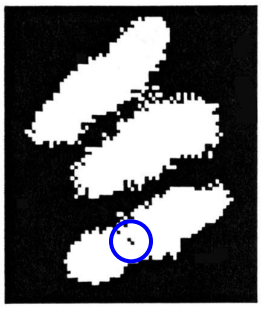
5

5

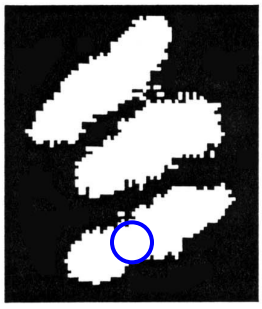
Results



Original binary image



8-connected removal



4-connected removal

Matlab: see bwmorph(), using 'clean' and 'fill'

6

6

Median Filtering

- Assume each pixel in neighborhood will be either
  - Uncorrupted pixel value
  - Noise pixel value
- Also, uncorrupted pixel values should be nearly the same (small neighborhood)
- Furthermore, assume that there are more uncorrupted values than noise values
- Solution: replace a pixel value with the median value of the spatial neighborhood
  - Value in the middle of the sorted distribution of pixel values (half of values greater, half are smaller)
    - Requires sorting operation on pixel values
  - Noise should be at one or both ends of the sorted distribution
- Matlab: see medfilt2() or ordfilt2()

7

7

Median Filtering

5x5 neighborhood of gray values

10	12	9	8	4
12	11	10	10	6
14	12	5	11	11
15	14	12	9	8
13	12	10	8	12

Median of sorted values

[4, 5, 6, 8, 8, 8, 9, 9, 10, 10, 10, 10,

11,

11, 11, 12, 12, 12, 12, 12, 12, 13, 14, 14, 15]

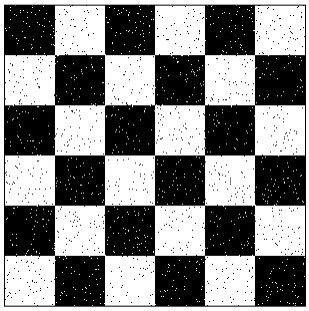
Replace value 5 with the median 11

8

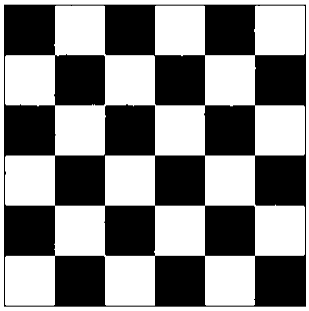
8

Median Example #1

Checkerboard with noise



After median filtering

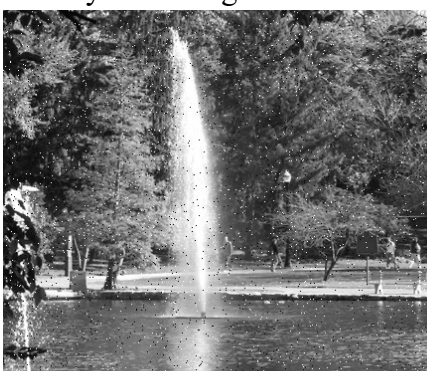


9

9

Median Example #2

Grayscale image with noise



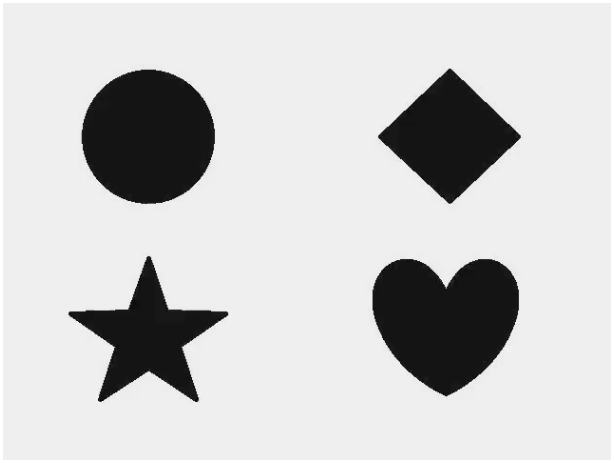
After median filtering

Tends to **preserve edge detail**, rather than blurring/smearing boundary between regions

10

10

Iterative Sequential Median???



11

11

Applying Filter Masks to Images

- Convolution/Correlation in image processing
- Mask is a set of pixel positions and corresponding values (weights)
  - Generally odd-numbered rows or columns
- Each mask has an origin (home position)
  - Center (or top-left) mask position most common

1	1	1
1	**1**	1
1	1	1

1	2	1
2	**4**	2
1	2	1

1
1
1
1
1

12

12

Applying Masks to Images

- For each valid pixel location x,y, place mask with origin lying on that pixel
- Image values under mask are multiplied with mask weights and then summed

1/9	1/9	1/9	1/9	1/9
1/9	1/9	1/9	1/9	1/9
1/9	1/9	1/9	1/9	1/9
1/9	1/9	1/9	**1/9**	1/9
1/9	1/9	1/9	1/9	1/9

Grayscale image

1/9	1/9	1/9
1/9	**1/9**	1/9
1/9	1/9	1/9

Averaging Mask

-	-	-	-	-
-	53	67	80	-
-	53	67	80	-
-	53	67	80	-
-	-	-	-	-

Result

13

13

Average (Mean) Filtering

- Sometimes called a "box filter"
- Average filter to smooth over local region

V = I(x-1, y-1)**1/9** + I(x, y-1)**1/9** + I(x+1, y-1)**1/9**
+ I(x-1, y)**1/9** + I(x, y)**1/9** + I(x+1, y)**1/9**
+ I(x-1, y+1)**1/9** + I(x,y+1)**1/9** + I(x+1,y+1)**1/9**

1	1	1
1	1	1
1	1	1

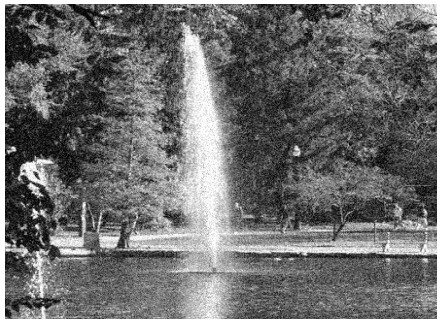
/ 9

14

14

Average Filtering

Grayscale image with noise



After average filtering

Matlab: See fspecial() with 'average', and imfilter()

15

15

Comparison (Zoomed)

After median filtering

After average filtering

16

16

General Properties of Smoothing Masks

- Values of mask are all positive and sum to one
  - So that output on constant regions are same as input
- Amount of smoothing is proportional to mask size
  - Bigger masks smooth more

17

17

Gaussian Smoothing

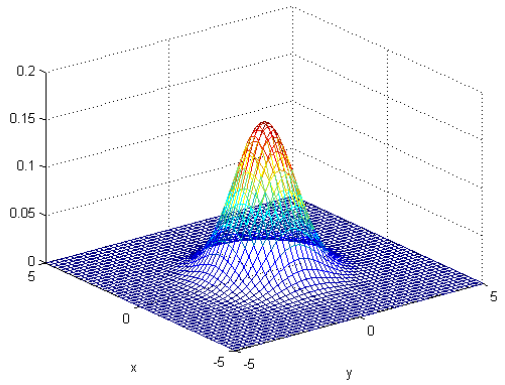
- Weight the influence of pixels by their distance to the center pixel
  - Weight decreases smoothly to 0 as move more distant from origin
- Symmetric in 2-D (x,y)
  - Isotropic function
- Consider Gaussian (Normal) distribution as weighting function

18

18

Gaussian Smoothing

$$g(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-x_\circ)^2+(y-y_\circ)^2}{2\sigma^2}} = ce^{-\frac{x^2+y^2}{2\sigma^2}}$$



19

19

Gaussian Smoothing

- The standard deviation σ controls the spread of the function
  - There is 95% of total weight within 2σ
  - There is 99.7% of total weight within 3σ
- To determine a mask size for a particular spread
  - Set mask size = ceil(2σ)*2+1 (95% of weight)
  - Or set mask size = ceil(3σ)*2+1 (99.7% of weight)
- Fill mask values with g(x,y; σ)
  - x-range: [-ceil(3σ) : ceil(3σ)] (could use 2σ)
  - y-range: [-ceil(3σ) : ceil(3σ)]
- Divide by sum of mask values so sums to 1

Matlab: can easily create with fspecial() using 'gaussian', and then filter with imfilter()

20

20

Gaussian Smoothing Masks

1	2	1
2	4	2
1	2	1

/ 16

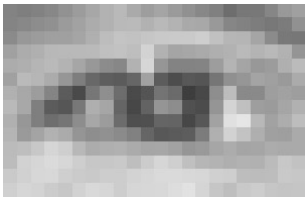
3x3

1	3	7	9	7	3	1
3	12	26	33	26	12	3
7	26	55	70	55	26	7
9	33	70	90	70	33	9
7	26	55	70	55	26	7
3	12	26	33	26	12	3
1	3	7	9	7	3	1

/ 1098

7x7

Original image



After Gaussian filtering

21

21

Gaussian Smoothing

Original image

$\sigma = 3$

$\sigma = 6$

22

22

Anisotropic Diffusion, Bilateral Edge-Preserving Filtering

- Methods to smooth an image, while preserving boundaries and structures of interest
- Basic idea is to adjust the smoothing level in a region based on the edge structure in the neighborhood
  - Homogenous regions are highly smoothed
  - Strong edge regions are barely smoothed (to preserve the structure)

23

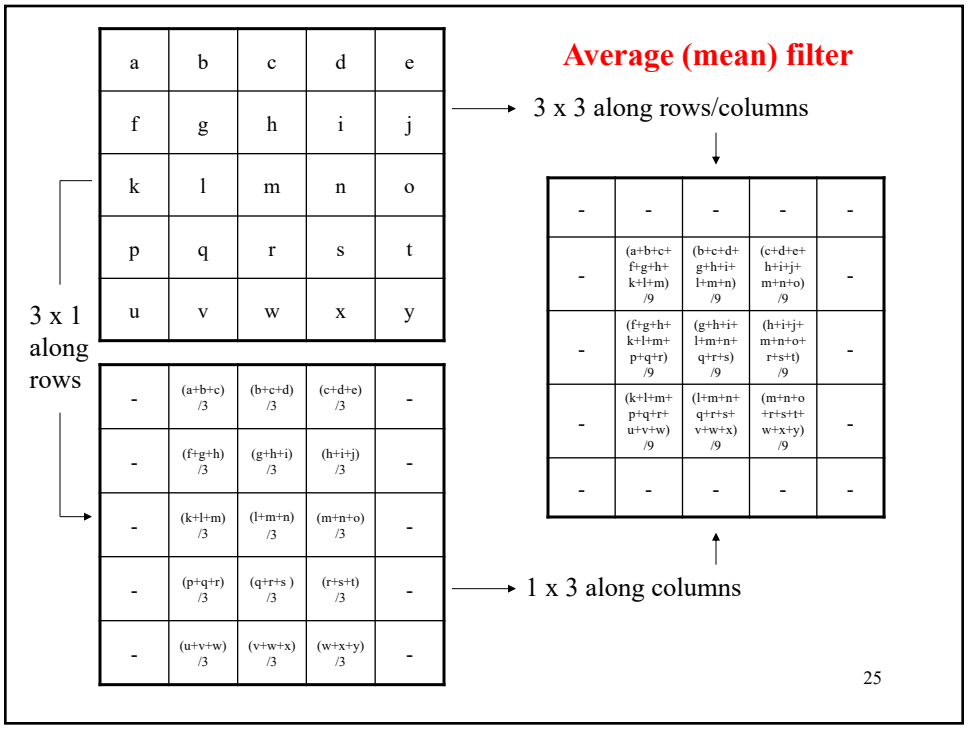
23

Separability of Filters

- For (N x N) image and (n x n) mask, computational complexity is O(N^2 x n^2)
  - For each pixel, we sum up a function of the n^2 values
- Can reduce complexity by using two 1-D filters
  - First, move along rows, put into temporary image
  - Second, move along columns of temporary image
  - Now require only $2n$ operations, instead of n^2
- Complexity is reduced to O(N^2 x n)

24

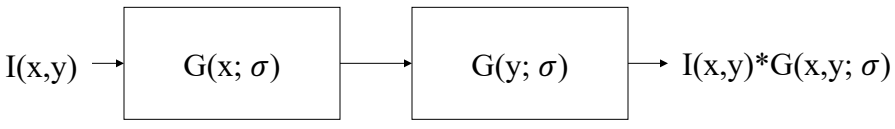
24



25

Gaussian Separability

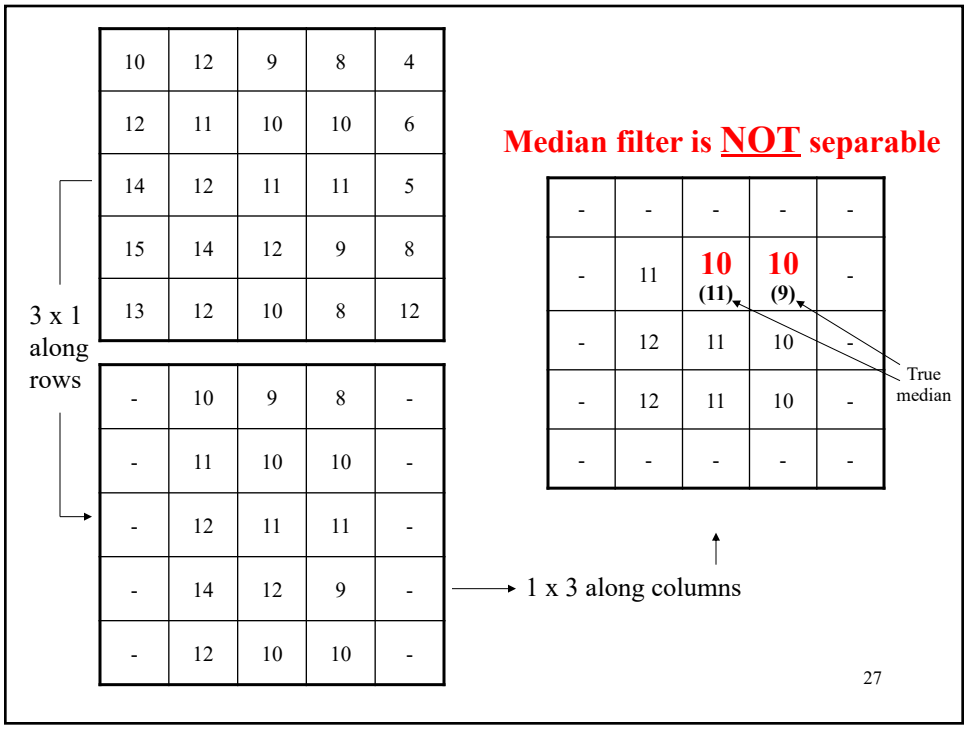
- Two-dimensional Gaussian can be separated into two 1-D Gaussians
  - One along *x* dimension, then along *y* dimension



$$g(x,y;\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-x_\circ)^2+(y-y_\circ)^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-x_\circ)^2}{2\sigma^2}} \times \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y-y_\circ)^2}{2\sigma^2}}$$

26

26



27

Summary

- Noise removal for images
  - At local neighborhood
- Median filter
  - Good for "salt-and-pepper" noise
  - Preserves edges
  - Not separable
- Average filter
  - Smooth image
  - Separable filter
- Gaussian filter
  - Weight influence of pixels by their distance to the center pixel
  - Normal distribution
  - Spread parameter
  - Separable filter

28

28