Noise Removal

Computer Vision (CS0029)

Noise

- Noise image
  - Image acquisition process is not perfect
  - Different camera can have different noise
- Filter image to
  - Enhance images
  - Reduce noise in image
  - Emphasize certain image details
- Idea of noise removal
  - Decision typically made at a level of local pixel region

Pixel Neighborhoods

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

-	1	-
1	*	1
-	1	-

- 8-connected pixel region
  - If pixel * connected to all eight neighbors

1	1	1
1	*	1
1	1	1

Removal of Binary Image Noise

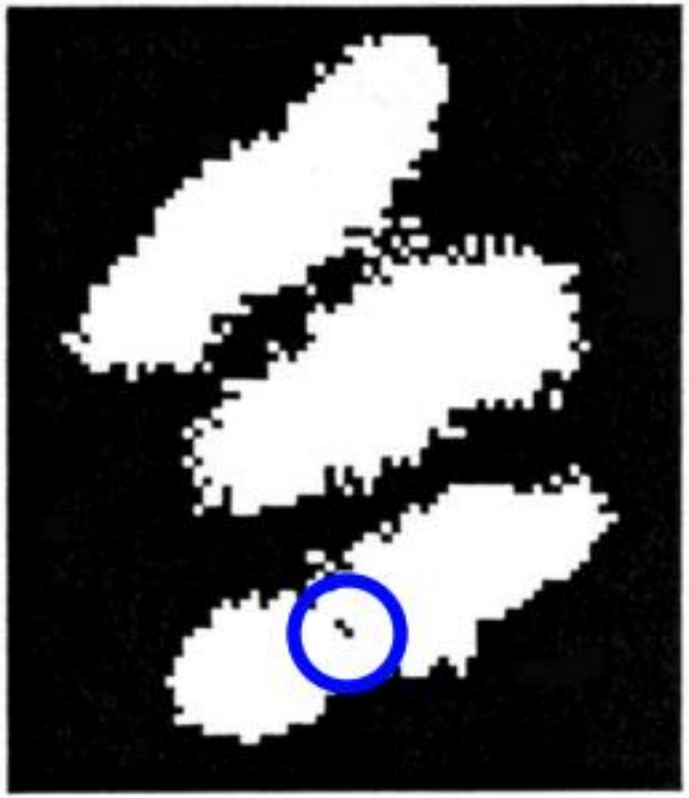
- Single dark pixel in bright regions
- Single bright pixels in dark regions
- Possibly from thresholding errors

Removal of value L isolated from neighborhood of X’s

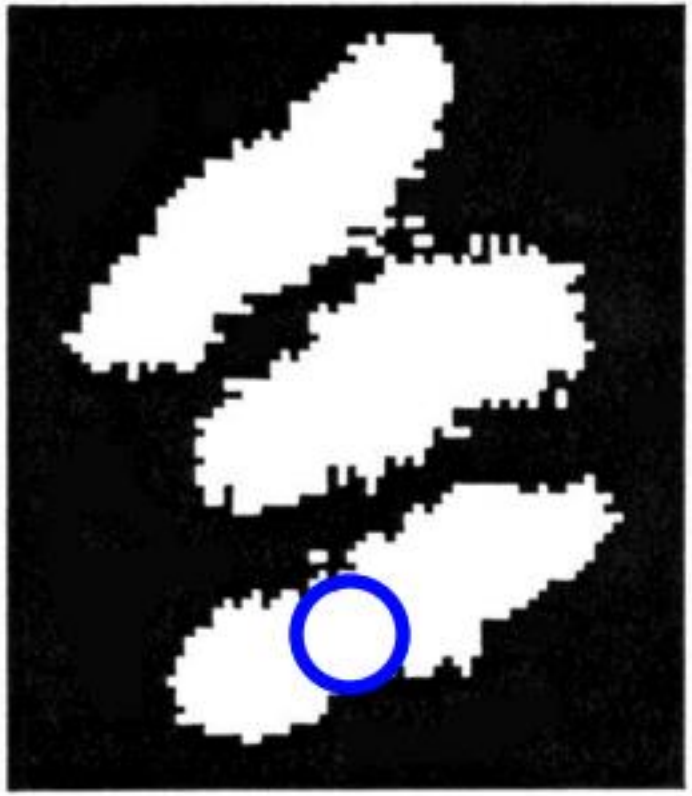
8-connected removal of L — Result — 4-connected removal of L — Result

Results

Original binary image



8-connected removal



4-connected removal

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 value 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 end of the sorted distribution

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 values

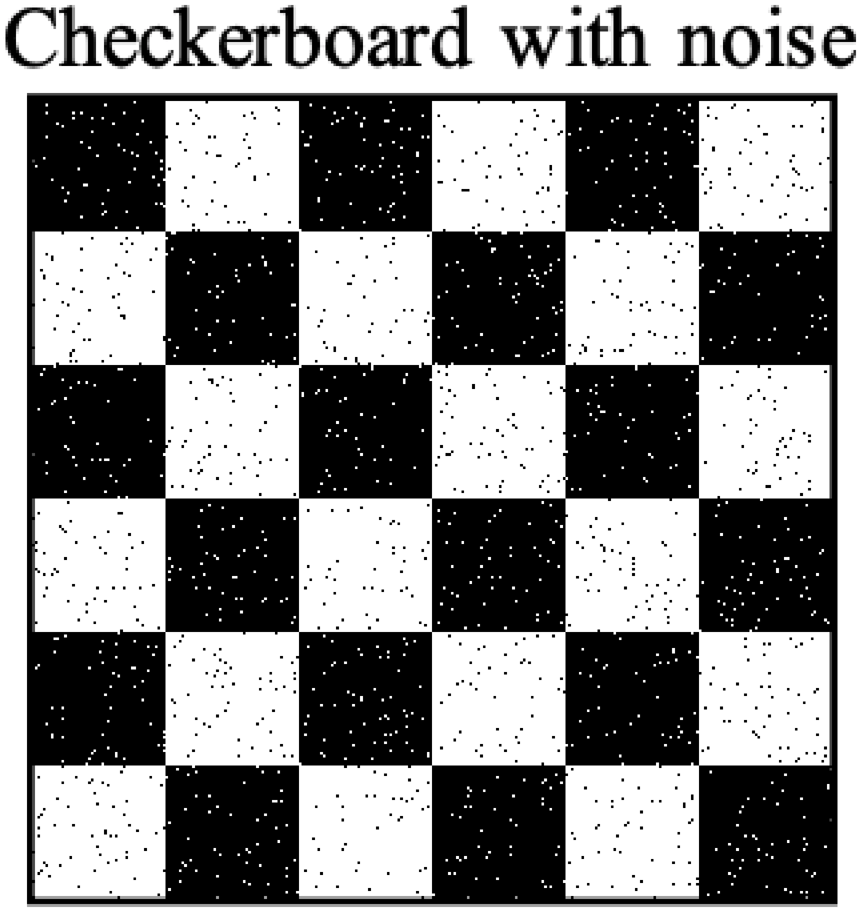
[4, 5, 6, 8, 8, 8, 9, 9, 10, 10, 10, 10,

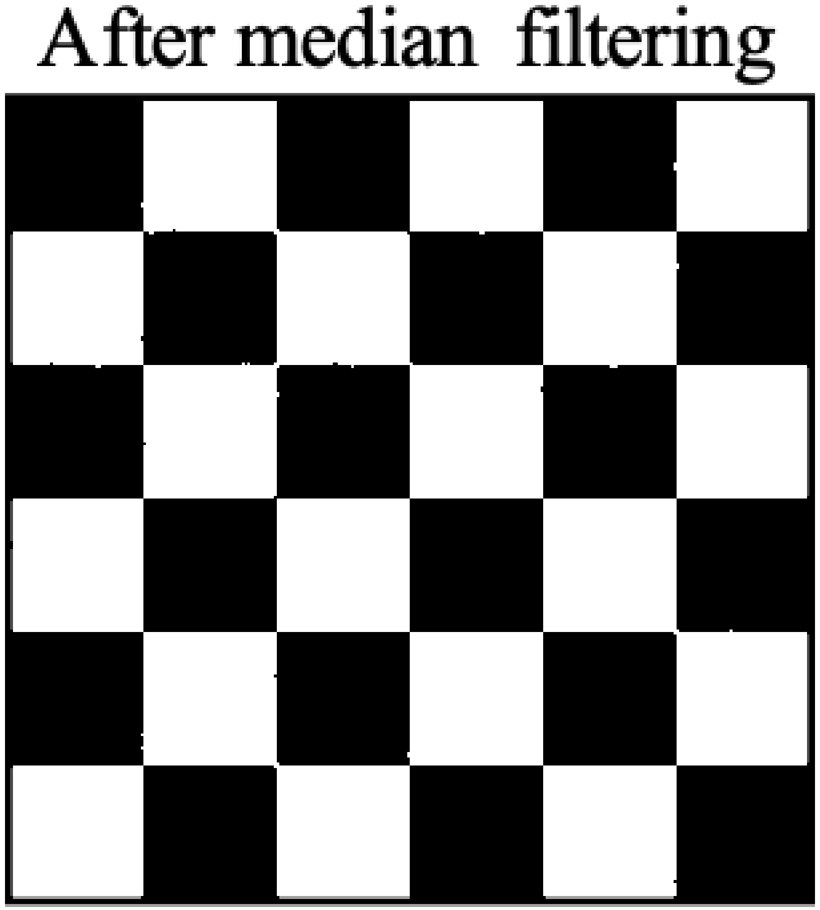
11,

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

Replace value 5 with the
median 11

Example #1





Example #2

Grayscale image with noise

After median filtering

Applying Filter Masks/Filters to Images

- Convolution 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
  - Center mask position is 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

Applying Masks to Images

- For each valid pixel location x, y, put mask with origin lying on the pixel
- Image value under mask are multiplied with mask weights and then summed

99	96	51	36	9
15	123	15	99	63
72	66	3	51	87
54	60	96	87	102
75	60	72	93	84

Grayscale image

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Averaging Mask

-	-	-	-	-
-	59	60	46	-
-	56	53	67	-
-	62	58	75	-
-	-	-	-	-

Result

Average Filtering

- Mean Filtering or Box Filter
- Average filter to smooth over local region

$$
\begin{aligned}
I'(x,y) = {} & I(x-1,y-1) * \left(\frac{1}{9}\right) + I(x,y-1) * \left(\frac{1}{9}\right) + I(x+1,y-1) * \left(\frac{1}{9}\right) \\
& + I(x-1,y) * \left(\frac{1}{9}\right) + I(x,y) * \left(\frac{1}{9}\right) + I(x+1,y) * \left(\frac{1}{9}\right) \\
& + I(x-1,y+1) * \left(\frac{1}{9}\right) + I(x,y+1) * \left(\frac{1}{9}\right) + I(x+1,y+1) * \left(\frac{1}{9}\right)
\end{aligned}
$$

$$
\begin{array}{|c|c|c|}
\hline
1 & 1 & 1 \\
\hline
1 & 1 & 1 \\
\hline
1 & 1 & 1 \\
\hline
\end{array}
\; / \, 9
$$

Average Filtering

Grayscale image with noise

After average filtering

Median and Average Filtering - OpenCV

Raw image

Average? Median? Filtering

Average? Median? Filtering

```python
import cv2
import numpy as np

img = cv2.imread('opencv_logo.png')

blur = cv2.blur(img,(5,5))

cv2.imwrite('avgBlue.png', blur)

median = cv2.medianBlur(img,5)

cv2.imwrite('median.png', median)
```

General Properties of Smoothing Masks

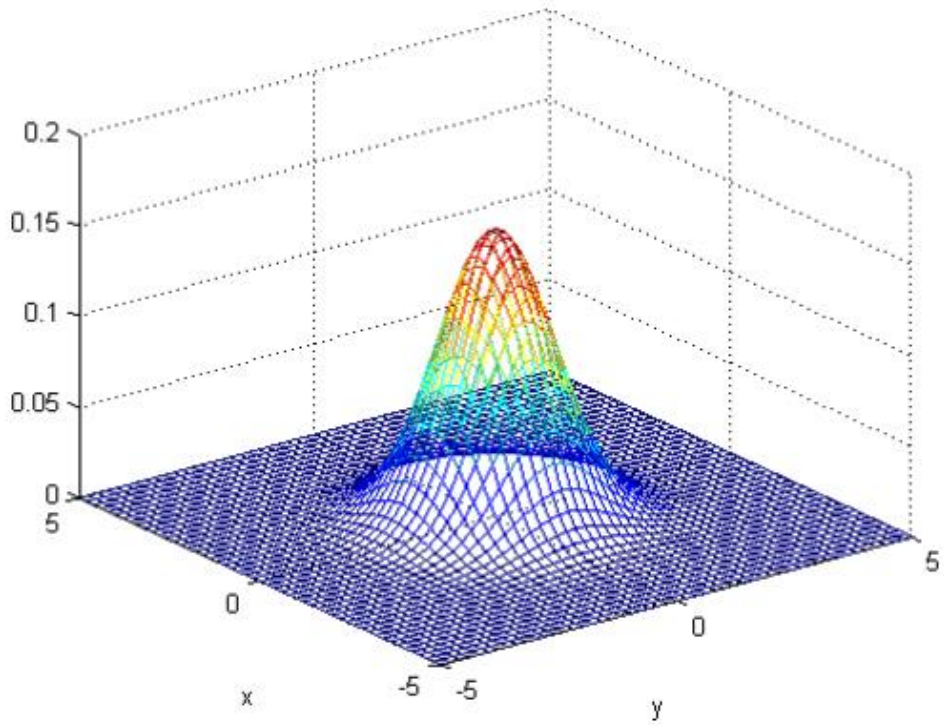
- Values of mask are all positive and sum to one
  - So that output on constant regions are same as input
  - New pixel value still within the data range
- Amount of smoothing is proportional to mask size
  - Bigger masks smooth more

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 2D
  - Isotropic function
- Consider Gaussian distribution as a weighting function

Gaussian Smoothing

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



Gaussian Smoothing

- The standard deviation σ control the spread of the function
  - 68–95–99.7 rule
  - 95% of the total weight within 2σ
  - 99.7% of the total weight within 3σ
- To determine a mask size for a particular spread
  - Set mask size = ceil(2σ)*2+1 (95% of weight)
  - Set mask size = ceil(3σ)*2+1 (99.7% of weight)
- Fill mask values with $g(x, y;\ \sigma)$
  - X-range: [-ceil(3σ) : ceil(3σ)] (or use 2σ)
  - Y-range: [-ceil(3σ) : ceil(3σ)]
- Divide by sum of mask values so sums to 1

Gaussian Smoothing Mask

1	2	1
2	4	2
1	2	1

/ 16

3x3

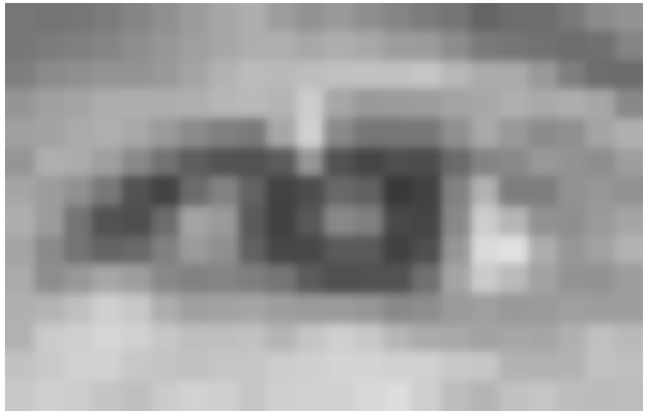
1	3	7	9	7	3	1
3	12	26	33	26	12	3
7	26	55	70	55	26	7
9	33	70	90	70	33	9
7	26	55	70	55	26	7
3	12	26	33	26	12	3
1	3	7	9	7	3	1

/ 1098

7x7

Original image

After Gaussian filtering



Gaussian Smoothing

Original image $\sigma = 3$ $\sigma = 6$

Gaussian Smoothing - OpenCV

Raw image

Gaussian Smoothing

```
import cv2
import numpy as np

img = cv2.imread('opencv_logo.png')

blur = cv2.GaussianBlur(img,(5,5),0)

cv2.imwrite('gaussianBlur.png', blur)
```

Separability of Filter

- For (N*N) image and (n*n) mask, computational complexity is $O(N^2 * n^2)$
  - For each pixel, we sum up a function of the n^2 values
- Can reduce complexity by using two 1-D filters
  - Step1: move along rows, put into temporary image
  - Step2: move along columns of temporary image
  - This only 2n operations
- Complexity is reduced to $O(N^2 * n)$

Average (mean) filter

a	b	c	d	e
f	g	h	i	j
k	l	m	n	o
p	q	r	s	t
u	v	w	x	y

3 x 1 along rows

-	(a+b+c) /3	(b+c+d) /3	(c+d+e) /3	-
-	(f+g+h) /3	(g+h+i) /3	(h+i+j) /3	-
-	(k+l+m) /3	(l+m+n) /3	(m+n+o) /3	-
-	(p+q+r) /3	(q+r+s) /3	(r+s+t) /3	-
-	(u+v+w) /3	(v+w+x) /3	(w+x+y) /3	-

3 x 3 along rows/columns

1 x 3 along columns

-	-	-	-	-
-	(a+b+c+ f+g+h+ k+l+m) /9	(b+c+d+ g+h+i+ l+m+n) /9	(c+d+e+ h+i+j+ m+n+o) /9	-
-	(f+g+h+ k+l+m+ p+q+r) /9	(g+h+i+ l+m+n+ q+r+s) /9	(h+i+j+ m+n+o+ r+s+t) /9	-
-	(k+l+m+ p+q+r+ u+v+w) /9	(l+m+n+ q+r+s+ v+w+x) /9	(m+n+o +r+s+t+ w+x+y) /9	-
-	-	-	-	-

Gaussian Separability

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

$$\mathrm{I(x,y)} \longrightarrow \boxed{\mathrm{G}(\mathrm{x};\ \sigma)} \longrightarrow \boxed{\mathrm{G}(\mathrm{y};\ \sigma)} \longrightarrow \mathrm{I(x,y)*G(x,y;}\ \sigma)$$

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

Median filter **NOT** separable

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

3 x 1 along rows

-	10	9	8	-
-	11	10	10	-
-	12	11	11	-
-	14	12	9	-
-	12	10	10	-

1 x 3 along columns

-	-	-	-	-
-	11	**10** (11)	**10** (9)	-
-	12	11	10	-
-	12	11	10	-
-	-	-	-	-

True median