Chapter 7 Conditioning and Labeling

7.1 Introduction

conditioning: noise removal, background normalization, ...
labeling: thresholding, edge detection, corner finding, ...

7.2 Noise Cleaning

noise cleaning: uses neighborhood spatial coherence
noise cleaning: uses neighborhood pixel value homogeneity

box filter: computes equally weighted average
box filter: separable
box filter: recursive implementation with two +, two -, one / per pixel
=====Fig. 7.2=====

Gaussian filter: linear smoother

$$w(r,c) = ke^{-\frac{1}{2}(\frac{r^2+c^2}{\sigma^2})}$$

for all $(r,c) \in W$, where

$$k = \frac{1}{\Sigma_{(r,c)\in W}\, e^{-\frac{1}{2}(\frac{r^2+c^2}{\sigma^2})}}$$

W: two or three σ from the center
linear noise-cleaning filters: defocusing images, edges blurred

7.2.1 A Statistical Framework for Noise Removal

idealization: if no noise, each neighborhood the same constant

7.2.2 Determining Optimal Weight from Isotropic Covariance Matrices

7.2.3 Outlier or Peak Noise

outlier: peak noise: pixel value replaced by random noise value
neighborhood size: larger than noise, smaller than preserved detail
center-deleted neighborhood: pixel values in neighborhood except center
$x_1, ..., x_N$: center-deleted neighborhood
y: center pixel value
$\hat{\mu}$: mean of center-deleted neighborhood

$$\hat{\mu} = \frac{1}{N} \sum_{n=1}^{N} x_n$$

$z_{\text{outlier removal}}$: output value of neighborhood outlier removal

$$z_{\text{outlier removal}} = \begin{cases} y & \text{if } |y - \hat{\mu}| < \theta \\ \hat{\mu} & \text{otherwise} \end{cases}$$

y: not an outlier value if y reasonably close to $\hat{\mu}$
$\hat{\mu}$: use mean value when outlier
θ: threshold for outlier value
θ too small: edges blurred
θ too large: noise cleaning will not be good

$\hat{\sigma}^2$: center-deleted neighborhood variance

$$\hat{\sigma}^2 = \frac{1}{N-1} \sum_{n=1}^{N} (x_n - \hat{\mu})^2$$

$z_{\text{contrast}-\text{dependent outlier removal}}$: output value with contrast-dependent-threshold

$$z_{\text{contrast}-\text{dependent outlier removal}} = \begin{cases} y & \text{if } \left|\frac{y-\hat{\mu}}{\hat{\sigma}}\right| < \theta \\ \hat{\mu} & \text{otherwise} \end{cases}$$

$\hat{\mu}$: use neighborhood mean if pixel value significantly far from mean

smooth replacement: instead of complete replacement or not at all
$z_{\text{smooth replacement}}$: convex combination of input and mean
K: weighting parameter

$$z_{\text{smooth replacement}} = \frac{\left|\frac{y-\hat{\mu}}{\hat{\sigma}}\right|}{\left|\frac{y-\hat{\mu}}{\hat{\sigma}}\right| + K}\hat{\mu} + \frac{K}{\left|\frac{y-\hat{\mu}}{\hat{\sigma}}\right| + K}y$$

$K = 0$: use neighborhood mean $\hat{\mu}$
$K = \infty$: use input pixel value y

7.2.4 k-Nearest Neighbor

k-nearest neighbor average: equally weighted average of k-nearest neighbors

7.2.5 Gradient Inverse Weighted

gradient inverse weighted: reduces sum-of-squares error within regions

7.2.6 Order Statistic Neighborhood Operators

order statistic: linear combination of neighborhood sorted values
$x_1, ..., x_N$: neighborhood pixel values
$x_{(1)}, ..., x_{(N)}$: sorted neighborhood values from smallest to largest

Median Operator

median: most common order statistic operator
median root: fixed-point result of a median filter
median roots: comprise only constant-valued neighborhoods, sloped edges
median: effective for impulsive noise

median: distorts or loses fine detail such as thin lines

$$z_{\text{median}} = x_{(\frac{N+1}{2})}$$

Q: interquartile distance

$$Q = x_{(\frac{3N+2}{4})} - x_{(\frac{N+2}{4})}$$

Trimmed-Mean Operator

trimmed-mean: first k and last k order statistics not used

trimmed-mean: equal weighted average of central $N - 2k$ order statistics

$$z_{\text{trimmed mean}} = \frac{1}{N-2k} \sum_{n=1+k}^{N-k} x_{(n)}$$

Midrange

midrange: noise distribution with light and smooth tails

$$z_{\text{midrange}} = \frac{1}{2}[x_{(1)} + x_{(N)}]$$

7.2.7 A Decision Theoretic Approach to Estimating Mean

=====New 3:12======

7.2.8 Hysteresis Smoothing

hysteresis smoothing: removes minor fluctuations, preserves major transients
hysteresis smoothing: finite state machine with two states: UP, DOWN
hysteresis smoothing: applied row-by-row and then column-by-column
if state DOWN and next one larger, if next local maximum does not exceed threshold then stays current value i.e. small peak cuts flat
otherwise state changes from DOWN to UP and preserves major transients
if state UP and next one smaller, if next local minimum does not exceed threshold then stays current value i.e. small valley filled flat
otherwise state changes from UP to DOWN and preserves major transients
=====Fig. 7.9=====

7.2.9 Sigma Filter

sigma filter: average only with values within two-sigma interval

$$\hat{y} = \frac{1}{\#M} \sum_{n \in M} x_n$$

where $M = \{n | y - 2\sigma \leq x_n \leq y + 2\sigma\}$

7.2.10 Selected-Neighborhood Averaging

selected-neighborhood averaging: assumes pixel a part of homogeneous region noise-filtered value: mean value from lowest variance neighborhood

7.2.11 Minimum Mean Square Noise Smoothing

minimum mean square noise smoothing: additive or multiplicative noise
each pixel in true image: regarded as a random variable

7.2.12 Noise-Removal Techniques-Experiments

=====Fig. 7.10=====
types of noise

- uniform
- Gaussian
- salt and pepper
  $i_{\min}$: minimum gray value for noise pixels
  $i_{\max}$: maximum gray value for noise pixels
  p: fraction of image to be corrupted with noise
  u: uniform random variable in $[0, 1]$
  $I(r, c)$: gray value at given pixel (r, c) in input image

$O(r, c)$: gray value at given pixel (r, c) in output image

$$O(r,c) = \begin{cases} I(r,c) & \text{if } u > p \\ i_{\min} + (i_{\max} - i_{\min}) * u & \text{otherwise} \end{cases}$$

- varying noise (the noise energy varies across the image)

S/N ratio: signal to noise ratio: $S = 10 * \log_{10}(VS/VN)$
VS: image gray level variance
VN: noise variance
=====Fig. 7.11=====
contrast-dependent noise removal ($z_{\text{contrast−dependent outlier removal}}$, 9×9, $\theta = 0.1$)
=====Fig. 7.12=====
smooth replacement ($z_{\text{smooth replacement}}$, 9×9, $\theta = 0.4$, $K = 0.2$)
=====Fig. 7.13=====
default peak noise removal ($z_{\text{outlier removal}}$, 9×9, $\theta = 0.2$)
=====Fig. 7.14=====
hysteresis smoothing
=====Fig. 7.15=====
interquartile mean filter ($z_{\text{trimmed mean}}$, 7×7, $k = N/4$)
=====Fig. 7.16=====
neighborhood midrange filter (z_{midrange}, 7×7)
=====Fig. 7.17=====

neighborhood running-mean filter (box filter, 7×7)
=====Fig. 7.18=====
sigma filter (Eq. 7.18, 9×9, $\sigma = 0.2$)
=====Fig. 7.19=====
neighborhood weighted median filter (7×7)

$$\begin{array}{ccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 & 2 & 2 & 1 \\ 1 & 2 & 3 & 3 & 3 & 2 & 1 \\ 1 & 2 & 3 & 4 & 3 & 2 & 1 \\ 1 & 2 & 3 & 3 & 3 & 2 & 1 \\ 1 & 2 & 2 & 2 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}$$

=====Fig. 7.20=====
gain in S/N ratio
=====Table 7.5=====

7.3 Sharpening

unsharp masking: subtract fraction of neighborhood mean and scale result

$$z_{\text{sharpened}} = s[y - k\hat{\mu}] = \hat{\mu} + s[y - \hat{\mu}]$$

where $\hat{\mu} = \frac{1}{N} \Sigma_{n=1}^{N} x_n$ is neighborhood mean

y: center pixel value
s: scaling constant
k: fraction, reasonable between $\frac{1}{5}, \frac{2}{3}$
possible to replace neighborhood mean with neighborhood median

7.3.1 Extremum Sharpening

extremum-sharpening: output closer of neighborhood maximum or minimum

$$z_{\min} = \min\{f(r+r', c+c') | (r', c') \in W\} = f \ominus W$$

$$z_{\max} = \max\{f(r+r', c+c') | (r', c') \in W\} = f \oplus W$$

$$z_{\text{extremum sharpened}} = \begin{cases} z_{\min} & \text{if } f(r,c) - z_{\min} < z_{\max} - f(r,c) \\ z_{\max} & \text{otherwise} \end{cases}$$

W: neighborhood, (value 0 in gray scale morphology)
=====Oldie 34:14=====

7.4 Edge Detection

digital edge: boundary where brightness values significantly different
edge: brightness value appears to jump
The Archer, working model, by Henry Moore, 1964
=====Nalwa, *A Guided Tour of Computer Vision*, Fig. 3.1=====
profiles of various types of image-intensity edges

=====Nalwa, *A Guided Tour of Computer Vision*, Fig. 3.2=====
physical events in a scene that may lead to intensity edges
=====Nalwa, *A Guided Tour of Computer Vision*, Fig. 3.3=====

7.4.1 Gradient Edge Detectors
Roberts operators: two 2×2 masks to calculate gradient
=====Fig. 7.21=====
gradient magnitude: $\sqrt{r_1^2 + r_2^2}$
where r_1, r_2 are values from first, second masks respectively

Prewitt edge detector: two 3×3 masks in row, column direction
=====Fig. 7.22=====
gradient magnitude: $g = \sqrt{p_1^2 + p_2^2}$
gradient direction: $\theta = \arctan(p_1/p_2)$ clockwise w.r.t. column axis
where p_1, p_2 are values from first, second masks respectively

Sobel edge detector: two 3×3 masks in row, column direction
=====Fig. 7.23=====
gradient magnitude: $g = \sqrt{s_1^2 + s_2^2}$
gradient direction: $\theta = \arctan(s_1/s_2)$ clockwise w.r.t. column axis

where s_1, s_2 are values from first, second masks respectively
Sobel edge detector: 2×2 smoothing followed by 2×2 gradient

$$\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} -1 & +1 \\ -1 & +1 \end{bmatrix}$$

$$\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} +1 & +1 \\ -1 & -1 \end{bmatrix}$$

Frei and Chen edge detector: two in a set of nine orthogonal masks (3×3)
=====Fig. 7.24=====
gradient magnitude: $g = \sqrt{f_1^2 + f_2^2}$
gradient direction: $\theta = \arctan(f_1/f_2)$ clockwise w.r.t. column axis
where f_1, f_2 are values from first, second masks respectively

Kirsch: set of eight compass template edge masks
=====Fig. 7.25=====
gradient magnitude

$$g = \max_{n, n=0,\ldots,7} k_n$$

gradient direction: $\theta = 45° \arg\max k_n$

Robinson: compass template mask set with only $0, \pm 1, \pm 2$
=====Fig. 7.26=====
done by only four masks: since negation of each mask is also a mask
gradient magnitude and direction same as Kirsch operator

Nevatia and Babu: set of six 5×5 compass template masks
=====Fig. 7.27=====
Robinson and Kirsch compass operator: detect lineal edges
=====Fig. 7.28=====
edge contour direction: along edge, right side bright, left side dark
edge contour direction: 90° more than gradient direction

four important properties an edge operator might have

1. accuracy in estimating gradient magnitude
2. accuracy in estimating gradient direction
3. accuracy in estimating step edge contrast
4. accuracy in estimating step edge direction

gradient direction: used for edge organization, selection, linking
=====Fig. 7.31=====

7.4.2 Zero-Crossing Edge Detectors
first derivative maximum: exactly where second derivative zero crossing
=====D. Marr, *Vision*, Fig. 2.8=====
Laplacian of a function $I(r,c)$

$$\nabla^2 I = (\frac{\partial^2}{\partial r^2} + \frac{\partial^2}{\partial c^2})I = \frac{\partial^2 I}{\partial r^2} + \frac{\partial^2 I}{\partial c^2}$$

two common 3×3 masks to calculate digital Laplacian
=====Fig. 7.33=====
the 3×3 neighborhood values of an image function
=====Fig. 7.34=====
3×3 mask computing a digital Laplacian $e = -(4a + 4b), 2a + b = 1$
=====Fig. 7.35=====
3×3 mask for computing minimum-variance digital Laplacian
=====Fig. 7.36=====
Laplacian of the Gaussian kernel
=====Fig. 7.37=====

A pixel is declared to have a zero crossing if it is less than $-t$ and one of its eight neighbors is greater than t, or if it is greater than t and one of its eight neighbors is less than $-t$, for some fixed threshold t
=====Fig. 7.38=====
=====D. Marr, *Vision*, Fig. 2.12=====
=====D. Marr, *Vision*, Fig. 2.13=====
=====D. Marr, *Vision*, Fig. 2.14=====

7.4.3 Edge Operator Performance
edge detector performance characteristics: misdetection/false-alarm rate
the Canny edge operator applied to a parts image
=====Nalwa, *A Guided Tour of Computer Vision*, Fig. 3.8=====
edges at two different scales detected by the Canny operator
=====Nalwa, *A Guided Tour of Computer Vision*, Fig. 3.9=====
the Nalwa-Binford and Marr-Hildreth edge operators
=====Nalwa, *A Guided Tour of Computer Vision*, Fig. 3.16=====
improvement of the resolution of an edge detector by image interpolation
=====Nalwa, *A Guided Tour of Computer Vision*, Fig. 3.17=====
edgel aggregation and edge description
=====Nalwa, *A Guided Tour of Computer Vision*, Fig. 3.18=====

7.5 Line Detection

A line segment on an image can be characterized as an elongated rectangular region having a homogeneous gray level bounded on both its longer sides by homogeneous regions of a different gray level

one-pixel-wide lines can be detected by compass line detectors

=====Fig. 7.39=====

semilinear line detector by step edge on either side of the line

=====Fig. 7.40=====

detecting lines of one to three pixels in width

=====Fig. 7.41=====

=====joke=====

Project due Dec. 7

Write the following programs:

1. Generate additive white Gaussian noise:

   $I(nim, i, j) = I(im, i, j) + amplitude * N(0, 1)$

   $N(0, 1)$: Gaussian random variable with zero mean and st. dev. 1

   amplitude determines signal-to-noise ratio, try 10, 30

   =====zooma.im=====

   =====zooma.gaus.8.06.im=====

   =====zooma.gaus.25.5.im=====

2. Generate salt-and-pepper noise:
   $I(nim, i, j) = 0$ if $uniform(0, 1) < 0.05$
   $I(nim, i, j) = 255$ if $uniform(0, 1) > 1 - 0.05$
   $I(nim, i, j) = I(im, i, j)$ otherwise
   $uniform(0, 1)$: random variable uniformly distributed over [0,1]
   try both 0.05 and 0.1
   =====zooma.pep.0.05.im=====
   =====zooma.pep.0.1.im=====
3. Run box filter (3×3, 5×5) on all noisy images
4. Run median filter (3×3, 5×5) on all noisy images
5. Run opening followed by closing or closing followed by opening

Project due Dec. 14

Write programs to generate the following gradient magnitude images and choose proper thresholds to get the binary edge images:

1. Roberts operator
2. Prewitt edge detector
3. Sobel edge detector
4. Frei and Chen gradient operator
5. Kirsch compass operator
6. Robinson compass operator

7. Nevatia-Babu 5X5 operator