Image Processing for feature extraction
Outline

- Rationale for image pre-processing
- Gray-scale transformations
- Geometric transformations
- Local preprocessing

- Reading: Sonka et al 5.1, 5.2, 5.3
Image (pre)processing for feature extraction (cont’d)

- Pre-processing does not increase the image information content
- It is useful on a variety of situations where it helps to suppress information that is not relevant to the specific image processing or analysis task (i.e. background subtraction)
- The aim of preprocessing is to improve image data so that it suppresses undesired distortions and/or it enhances image features that are relevant for further processing
Image (pre)processing for feature extraction

- Early vision: pixelwise operations; no high-level mechanisms of image analysis are involved
- Types of pre-processing
  - enhancement (contrast enhancement for contour detection)
  - restoration (aim to suppress degradation using knowledge about its nature; i.e. relative motion of camera and object, wrong lens focus etc.)
  - compression (searching for ways to eliminate redundant information from images)
What are image features?

- **Image features can refer to:**
  - **Global properties of an image:**
    - i.e. average gray level, shape of intensity histogram etc.
  - **Local properties of an image:**
    - We can refer to some local features as image primitives: circles, lines, texels (elements composing a textured region)
    - Other local features: shape of contours etc.
Example of global image features

a) apples  

b) oranges

**hue**

(a1)  

(b1)

**saturation**

(a2)  

(b2)

**intensity**

(a3)  

(b3)
Example of local image features

Circumscribed (benign) lesions in digital mammography

Spiculated lesions in (digital mammography)

The feature of interest: shape of contour; regularity of contour

-Can be described by Fourier coefficients

-We can build a feature vector for each contour containing its Fourier coefficients
Image features

- Are local, meaningful, detectable parts of an image:
  - Meaningful:
    - features are associated to interesting scene elements in the image formation process
    - They should be invariant to some variations in the image formation process (i.e. invariance to viewpoint and illumination for images captured with digital cameras)
  - Detectable:
    - They can be located/detected from images via algorithms
    - They are described by a feature vector
Preprocessing

- Pixel brightness transformations (also called gray scale transformations)
  - Do not depend on the position of the pixel in the image
- Geometric transformations
  - Modify both pixel coordinates and intensity levels
- Neighborhood-based operations (filtering)
Outline

- Rationale for image pre-processing
- Gray-scale transformations
- Geometric transformations
- Local preprocessing

- Reading: Sonka et al 5.1, 5.2, 5.3
Gray-scale transformations work on histograms

- A histogram $H(r)$ counts how many times each quantized value occurs.
- It is a 1D array.
- $H(i) =$ number of pixels in image having intensity level $i$.
- Total area of image = total area under the histogram.
- Can convert from the histogram (counting values) to probabilities (percentages) by just dividing by the area:
- This produces a probability density function.

\[ p(r) = \frac{H(r)}{\text{Area}} \]
Contrast adjustment on histograms

- Adjusting contrast causes the histogram to stretch or shrink horizontally:
  - Stretching = more contrast
  - Shrinking = less contrast
Histogram equalization

- Is a gray-scale transformation for the enhancement of the appearance of images
  - Ex: images that are predominantly dark have all the useful information compressed into the dark end of the histogram
- The aim: create an image with equally distributed brightness levels over the whole brightness scale
  - Find a grayscale transformation function that creates an output image with a uniform histogram (or nearly so)
Histogram equalization

**Ideal case**

**Real case**
Histogram equalization: how it works

1. Initialize the array $H = \text{zeros}(1, 256)$
2. Generate the histogram: scan each pixel and place it in the appropriate bin
3. Generate the cumulative histogram $H_c$ (which is the approximation of the discrete distribution function)
4. Map input level $p$ into output level $T[p]$ using

$$T[p] = \text{round}\left(\frac{G-1}{NM} H_c(p)\right)$$
Ideal case: histogram in the input image is gaussian

Figure 6-23. Result of cumulative distribution function (left) on a Gaussian distribution (right)

Figure 6-24. Using the cumulative density function to equalize a Gaussian distribution
Figure 6-22. The image on the left has poor contrast, as is confirmed by the histogram of its intensity values on the right.

Figure 6-25. Histogram equalized results: the spectrum has been spread out.
Adaptive histogram equalization

- Allows localized contrast enhancement:
  - Shadows
  - Background variations
  - Other situations where global enhancement wouldn’t work
- Remap based on local, not global histogram
- Example: 7x7 window around the point
  - Problem: ringing artifacts
Adaptive histogram equalization (cont’d)

Figure 3.23 (a) Original image. (b) Result of global histogram equalization. (c) Result of local histogram equalization using a 7 x 7 neighborhood about each pixel.
Histogram specification

- Histogram equalization: uniform output histogram
- We can instead make it whatever we want it to be
  - Comparing two images
  - “Stitching” multiple images
  - Image-compositing operations
- Use histogram equalization as an intermediate step
Histogram specification

- First, equalize the histogram of the input image:
  \[ T_1(r) = \int_0^r \rho_1(w) \, dw \]

- Then histogram equalize the desired output histogram:
  \[ T_2(r) = \int_0^r \rho_2(w) \, dw \]

- Histogram specification is
  \[ T(r) = T_2^{-1}(T_1(r)) \]
Outline

- Rationale for image pre-processing
- Gray-scale transformations
- Geometric transformations
- Local pre-processing

- Reading: Sonka et al 5.1, 5.2, 5.3
Problem

How are these images related?

Slide from T. Svoboda, Homography from point pairs
Geometric transformations

- Commonly used in computer graphics
- Used in image analysis as well
  - Eliminate geometric distortion that occurs during image acquisition
  - Useful, for instance, when matching two or several images that correspond to the same object
- Two basic steps:
  - Pixel coordinate transformation
  - Brightness interpolation
Pixel coordinate transformations

- **Special cases**
  - Bilinear transforms: four pairs of corresponding points are sufficient to find the transformation coefficients
    - $x' = a_0 + a_1 x + a_2 y + a_3 xy$
    - $y' = b_0 + b_1 x + b_2 y + b_4 xy$
  - Affine transforms: only three pairs needed
    - $x' = a_0 + a_1 x + a_2 y$
    - $y' = b_0 + b_1 x + b_2 y$
  - Examples of affine transforms:
    - Rotation
    - Change of scale
    - Skewing
Projective transformations

- Projections of a planar scene by a pinhole camera are always related by homographies

- Application: rectifying images of planar scenes to a frontoparallel view
Outline

- Rationale for image pre-processing
- Gray-scale transformations
- Geometric transformations
- Local pre-processing

- Reading: Sonka et al 5.1, 5.2, 5.3
Local preprocessing

- Denotes neighborhood operations
- The output is a function of the pixel’s value and of its neighbors

\[
g(x, y) = Op \left( \begin{array}{ccc}
f(x - 1, y - 1) & f(x, y - 1) & f(x + 1, y - 1) \\
f(x - 1, y) & f(x, y) & f(x + 1, y) \\
f(x - 1, y + 1) & f(x, y + 1) & f(x + 1, y + 1)
\end{array} \right)
\]

- Local preprocessing can be linear or not
- Image smoothing, edge detection etc.

Adapted from Brian Morse, [http://morse.cs.byu.edu/450/home/index.php](http://morse.cs.byu.edu/450/home/index.php)
Kernels

- Most common neighborhood operation: weighted sum
- The weights of the sum constitute the mask or the kernel of the filter

\[
g(x, y) = \sum_{s=-1}^{1} \sum_{t=-1}^{1} w(s, t) f(x + s, y + t)
\]

Adapted from Brian Morse, [http://morse.cs.byu.edu/450/home/index.php](http://morse.cs.byu.edu/450/home/index.php)
Convolution

- Spatial filtering is often referred as convolution of the image by a kernel or mask.

$$g(x, y) = \sum_{s=-1}^{1} \sum_{t=-1}^{1} w(s, t) f(x-s, y-t)$$
Steps in computing the (2,4) output pixel:
a) Rotate the convolution kernel 180 degrees about its center element.
b) Slide the center element of the convolution kernel so that it lies on top of the (2,4) element of A.
c) Multiply each weight in the rotated convolution kernel by the pixel of A underneath. Sum the individual products from step c.

The (2,4) output pixel is:

\[ 1 \cdot 2 + 8 \cdot 9 + 15 \cdot 4 + 7 \cdot 7 + 14 \cdot 5 + 16 \cdot 3 + 13 \cdot 6 + 20 \cdot 1 + 22 \cdot 8 = 575 \]
Filtering with MATLAB Image processing toolbox

- Function `imfilter()` can be used for filtering either by correlation or convolution.

```matlab
I = imread('coins.png');
h = ones(5,5) / 25;
I2 = imfilter(I,h);
imshow(I), title('Original Image');
figure, imshow(I2), title('Filtered Image')
```
Boundary effects

What value should these outside pixels have?

Center of kernel
Boundary effects: zero-padding

Outside pixels are assumed to be 0.

Center of kernel

Original Image

Filtered Image with Black Border
Boundary effects: border replication

These pixel values are replicated from boundary pixels.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>17</th>
<th>24</th>
<th>18</th>
<th>25</th>
<th>2</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6</td>
<td>23</td>
<td>5</td>
<td>11</td>
<td>18</td>
<td>11</td>
<td>15</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>13</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Center of kernel

Filtered Image with Border Replication
Linear filtering for noise removal

- What is noise?
  - In computer vision, noise may refer to any entity, in images, data, or intermediate results, that is not interesting for the purposes of the main computation.
  - For instance:
    - In edge detection algorithms, noise can be the spurious fluctuations of pixel values introduced by the image acquisition system.
    - For algorithms taking as input the results of some numerical computation, noise can be introduced by the computer’s limited precision, round-offs errors etc.
    - We will concentrate on image noise.
Image noise

- We assume that the main image noise is additive and random

\[ \hat{I}(i, j) = I(i, j) + n(i, j) \]

- The amount of noise in an image can be estimated by the means of \( \sigma_n \), the standard deviation of \( n(i,j) \)
  - Signal to noise ratio:
    - \[ SNR = \frac{\sigma_s}{\sigma_n} \]
    - \[ SNR_{dB} = 10 \log_{10} \frac{\sigma_s}{\sigma_n} \]
Additive stationary Gaussian noise

- The simplest noise model
- The intensity of each pixel has added to it a value chosen from the same Gaussian probability distribution.
- Model parameters: 
  \[ \hat{I}(i, j) = I(i, j) + n(i, j) \]
- first intended to describe thermal noise in cameras:
  - Electrons can be freed from the CCD material itself through thermal vibration and then, trapped in the CCD well, be indistinguishable from "true" photoelectrons.
- The Gaussian noise model is often a convenient approximation when we do not know and we cannot estimate the noise characteristics.
Limitations of stationary Gaussian noise

- this model allows noise values that could be greater than maximum camera output or less than 0.
- Functions well only for small standard deviations
- may not be stationary (e.g. thermal gradients in the ccd)
sigma=1
sigma = 16
Salt-and-pepper noise

- Salt-and-pepper noise: presence of single dark pixels in bright regions (‘salt’) or single bright pixels in white regions (‘pepper’); also called impulsional, spot, or peak noise
a) Synthetic image of a grey-level checkerboard and grey-level profile along a row

b) After adding Gaussian noise ($\sigma=5$)

c) After adding salt-and-pepper noise

From Trucco and Verri
Linear filtering for noise removal: smoothing

- **Goal**: eliminate/reduce noise without altering the signal too much.
- **Response of a linear filter to additive gaussian noise**:

\[
\hat{E}(i, j) = E(i, j) + n(i, j)
\]

\[
E_f(i, j) = A \cdot \hat{E} = \sum_{h=-\frac{m}{2}}^{\frac{m}{2}} \sum_{k=-\frac{m}{2}}^{\frac{m}{2}} A(h,k) \hat{E}(i-h, j-k)
\]

\[
\mu_f = \sum_{h=-\frac{m}{2}}^{\frac{m}{2}} \sum_{k=-\frac{m}{2}}^{\frac{m}{2}} A(h,k) \mu(\hat{E}(i-h, j-k)) = 0
\]

\[
\sigma_f = \sigma^2 \sum_{h,k} A^2(h,k)
\]
Example: Smoothing by Averaging
Limits of the uniform filter

- It creates ringing
- The ringing phenomenon can be explained by aliasing.
Smoothing with a Gaussian
Gaussian Kernel

Idea: Weight contributions of neighboring pixels by nearness

\[ G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \]
Smoothing with a Gaussian kernel

- The FT of a Gaussian is a Gaussian and thus has no secondary lobes
- Gaussian smoothing is isotropic

A smoothing kernel proportional to $\exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$
Design of a Gaussian filter

- we need to produce a discrete approximation to the Gaussian function before we can perform the convolution.
- In theory, the Gaussian distribution is non-zero everywhere, which would require an infinitely large convolution mask.
- In practice it is effectively zero more than about three standard deviations from the mean, and so we can truncate the mask.
- The size of the mask is chosen according to $\sigma$. $w = 3\times \sigma$
Gaussian filtering

- Theorem of central limit: repeated convolution of a uniform 3X3 mask with itself yields a Gaussian filter.
- This is also called Gaussian smoothing by repeated averaging (RA)
  - Convolving a 3x3 mask n times with an image I approximates the Gaussian convolution of I with a Gaussian mask of
    \[ \sigma = \sqrt{n / 3} \]
  - and size 3(n+1)-n=2n+3
Smoothing with non-linear filters

- Main problems of the averaging filter:
  1) Ringing introduces additional noise
  2) Impulsive noise is only attenuated and diffused, not removed
  3) Sharp boundaries of objects are blurred. Blurring will affect the accuracy of boundary detection

Note: first problem is solved by Gaussian filters
Second and third problems are addressed by non-linear filters (i.e. filters that can not be modeled as a convolution)
Averaging using a rotating mask

- A non-linear smoothing method that avoids edge blurring by searching for the homogeneous part of the current pixel neighborhood.
- The homogeneity of a ‘sub-neighborhood’ is measured using a brightness dispersion $\sigma^2$.

$$\sigma^2 = \frac{1}{n} \sum_{(i,j) \in R} \left( g(i,j) - \frac{1}{n} \sum_{(i,j) \in R} g(i,j) \right)^2$$

- The resulting image is in fact sharpened.
Averaging using a rotating mask (cont’d)

- Try eight different oriented regions
- Calculate the brightness dispersion in each
- Use the average of the oriented neighborhood with the lowest dispersion

Figure 4.11 8 possible rotated 3 x 3 masks.
Median filtering

- In a set of ordered values, the median is the central value.
- Median filtering reduces blurring of edges.
- The idea: replace the current point in the image by the median of the brightness in its neighborhood.
Median filtering: discussion

- is not affected by individual noise spikes
- eliminates impulsive noise quite well
- does not blur edges much and can be applied iteratively.

- Main disadvantage of median filtering in a rectangular neighborhood: damaging of thin lines and sharp corners in the image