Previously

• Filters allow local image neighborhood to influence our description and features
  – Smoothing to reduce noise
  – Derivatives to locate contrast, gradient

• Seam carving application:
  – use image gradients to measure “interestingness” or “energy”
  – remove 8-connected seams so as to preserve image’s energy.

Today

• Edge detection and matching
  – process the image gradient to find curves/contours
  – comparing contours

• Binary image analysis
  – blobs and regions

Edge detection

• **Goal**: map image from 2d array of pixels to a set of curves or line segments or contours.
• **Why?**

  • **Main idea**: look for strong gradients, post-process

Thresholding

• Choose a threshold value $t$
• Set any pixels less than $t$ to zero (off)
• Set any pixels greater than or equal to $t$ to one (on)

Gradients -> edges

Primary edge detection steps:
1. Smoothing: suppress noise
2. Edge enhancement: filter for contrast
3. Edge localization
   Determine which local maxima from filter output are actually edges vs. noise
   • Threshold, Thin

Thresholding

• Choose a threshold value $t$
• Set any pixels less than $t$ to zero (off)
• Set any pixels greater than or equal to $t$ to one (on)
Canny edge detector

- Filter image with derivative of Gaussian
- Find magnitude and orientation of gradient
- **Non-maximum suppression:**
  - Thin wide “ridges” down to single pixel width
- **Linking and thresholding (hysteresis):**
  - Define two thresholds: low and high
  - Use the high threshold to start edge curves and the low threshold to continue them

- MATLAB: `edge(image, 'canny');`
- `>>help edge`

Source: D. Lowe, L. Fei-Fei

The Canny edge detector

Slide credit: Steve Seitz
The Canny edge detector

How to turn these thick regions of the gradient into curves?

Non-maximum suppression

Check if pixel is local maximum along gradient direction, select single max across width of the edge

- requires checking interpolated pixels p and q

Hysteresis thresholding

- Use a high threshold to start edge curves, and a low threshold to continue them.
Recap: Canny edge detector

- Filter image with derivative of Gaussian
- Find magnitude and orientation of gradient
- **Non-maximum suppression:**
  - Thin wide “ridges” down to single pixel width
- **Linking and thresholding (hysteresis):**
  - Define two thresholds: low and high
  - Use the high threshold to start edge curves and the low threshold to continue them

- **MATLAB:** `edge(image, 'canny');`
- `>>help edge`

Low-level edges vs. perceived contours

Berkeley segmentation database:

http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/
What features are responsible for perceived edges?

Feature profiles (oriented energy, brightness, color, and texture gradients) along the patch’s horizontal diameter

Kristen Grauman, UT-Austin

State-of-the-Art in Contour Detection

Canny, Canny+opt, thresholds

Human agreement

Learned with combined features

UC Berkeley: Jitendra Malik
http://www.cs.berkeley.edu/~malik/malik-talks-ptrs.html

Today

• Edge detection and matching
  – process the image gradient to find curves/contours
    – comparing contours

• Binary image analysis
  – blobs and regions

Figure from Belongie et al.
Chamfer distance

• Average distance to nearest feature

\[ D_{chamfer}(T, I) = \frac{1}{|T|} \sum_{t \in T} d_I(t) \]

\[ I \] = Set of points in image

\[ T \] = Set of points on (shifted) template

\[ d_I(t) = \text{Minimum distance between point } t \]
\[ \text{and some point in } I \]

Distance transform

Distance Transform is a function \( D(p) \) that for each image pixel \( p \) assigns a non-negative number \( D(p) \) corresponding to distance from \( p \) to the nearest feature in the image \( I \).

Features could be edge points, foreground points, ...

Source: Yuri Boykov

Distance transform (1D)

Two pass \( O(n) \) algorithm for 1D \( L_1 \) norm

1. Initialize: For all \( j \)
   \[ D[j] \leftarrow 1_{p[j]} \]
   // 0 if \( j \in P \), infinity otherwise

Adapted from D. Huttenlocher
Distance Transform (2D)

- 2D case analogous to 1D
  - Initialization
  - Forward and backward pass
    - Fwd pass finds closest above and to left
    - Bwd pass finds closest below and to right

Chamfer distance

- Average distance to nearest feature

\[ D_{\text{cham}}(f, I) = \frac{1}{|T|} \sum_{t \in T} d_f(t) \]

Chamfer distance: properties

- Sensitive to scale and rotation
- Tolerant of small shape changes, clutter
- Need large number of template shapes
- Inexpensive way to match shapes

Today

- Edge detection and matching
  - process the image gradient to find curves/contours
  - comparing contours
- Binary image analysis
  - blobs and regions

Binary images
Binary image analysis: basic steps

- Convert the image into binary form
  - Thresholding
- Clean up the thresholded image
  - Morphological operators
- Extract separate blobs
  - Connected components
- Describe the blobs with region properties

Thresholding

- Grayscale -> binary mask
- Useful if object of interest’s intensity distribution is distinct from background
  
  \[
  F(i, j) = \begin{cases} 
  1 & \text{if } F(i, j) \geq T \\
  0 & \text{otherwise},
  \end{cases}
  \]

  \[
  F(i, j) = \begin{cases} 
  1 & \text{if } T_1 \leq F(i, j) \leq T_2 \\
  0 & \text{otherwise},
  \end{cases}
  \]

  \[
  F(i, j) = \begin{cases} 
  1 & \text{if } F(i, j) \leq Z \\
  0 & \text{otherwise}.
  \end{cases}
  \]

  Example: edge detection

  - Gradient magnitude

  Looking for pixels where gradient is strong.

  ```
  fg_pix = find(gradient_mag > t);
  ```

Thresholding

- Given a grayscale image or an intermediate matrix \( \rightarrow \) threshold to create a binary output.

Example: background subtraction

- Looking for pixels that differ significantly from the “empty” background.

  ```
  fg_pix = find(diff > t);
  ```

Example: intensity-based detection

- Looking for dark pixels

  ```
  fg_pix = find(im < 65);
  ```
Thresholding

- Given a grayscale image or an intermediate matrix → threshold to create a binary output.

Example: color-based detection

Looking for pixels within a certain hue range.

A nice case: bimodal intensity histograms

Not so nice cases

Issues

- What to do with "noisy" binary outputs?
  - Holes
  - Extra small fragments

- How to demarcate multiple regions of interest?
  - Count objects
  - Compute further features per object

Morphological operators

- Change the shape of the foreground regions via intersection/union operations between a scanning structuring element and binary image.

- Useful to clean up result from thresholding

- Basic operators are:
  - Dilation
  - Erosion

Dilation

- Expands connected components
- Grow features
- Fill holes
Erosion
• Erode connected components
• Shrink features
• Remove bridges, branches, noise

Before erosion After erosion

Structuring elements
• Masks of varying shapes and sizes used to perform morphology, for example:

• Scan mask across foreground pixels to transform the binary image

>> help strel

Dilation vs. Erosion
At each position:
• Dilation: if current pixel is foreground, OR the structuring element with the input image.

Example for Dilation (1D)

Input image
\[ \begin{array}{cccccccc}
1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
\end{array} \]

Structuring Element
\[ \begin{array}{ccc}
1 & 1 & 1 \\
\end{array} \]

Output Image
\[ \begin{array}{ccc}
1 & 1 & 1 \\
\end{array} \]

Adapted from T. Moeslund

Example for Dilation

Input image
\[ \begin{array}{cccccccc}
1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
\end{array} \]

Structuring Element
\[ \begin{array}{ccc}
1 & 1 & 1 \\
\end{array} \]

Output Image
\[ \begin{array}{ccc}
1 & 1 & 1 \\
\end{array} \]
Example for Dilation

Input image: 1 0 0 0 1 1 1 0 1 1
Structuring Element: 1 1 1
Output Image: 1 1 0 0 0 0 0 0 0 0

Example for Dilation

Input image: 1 0 0 0 1 1 1 0 1 1
Structuring Element: 1 1 1
Output Image: 1 1 0 0 0 0 0 0 0 0

Example for Dilation

Input image: 1 0 0 0 1 1 1 0 1 1
Structuring Element: 1 1 1
Output Image: 1 1 0 1 1 1 1 1 1 1

Example for Dilation

Input image: 1 0 0 0 1 1 1 0 1 1
Structuring Element: 1 1 1
Output Image: 1 1 0 1 1 1 1 1 1 1

Note that the object gets bigger and holes are filled.

>> help imdilate
2D example for dilation

Dilation vs. Erosion

At each position:
- **Dilation**: if current pixel is foreground, OR the structuring element with the input image.
- **Erosion**: if every pixel under the structuring element’s nonzero entries is foreground, OR the current pixel with S.

Example for Erosion (1D)

Example for Erosion (1D)

Example for Erosion

Example for Erosion
Example for Erosion

Input image: 1 0 0 1 1 1 0 1 1
Structuring Element: 1 1 1
Output Image: 0 0 0 0 0 0 0 0 0

Example for Erosion

Input image: 1 0 0 1 1 1 0 1 1
Structuring Element: 1 1 1
Output Image: 0 0 0 0 0 0 1 0 0

Example for Erosion

Input image: 1 0 0 1 1 1 0 1 1
Structuring Element: 1 1 1
Output Image: 0 0 0 0 0 1 0 0 0

Example for Erosion

Input image: 1 0 0 1 1 1 0 1 1
Structuring Element: 1 1 1
Output Image: 0 0 0 0 0 1 0 0 0

Note that the object gets smaller
>> help imerode
2D example for erosion

Opening

- Erode, then dilate
- Remove small objects, keep original shape

Closing

- Dilate, then erode
- Fill holes, but keep original shape

Morphology operators on grayscale images

- Dilation and erosion typically performed on binary images.
- If image is grayscale: for dilation take the neighborhood max, for erosion take the min.

Issues

- What to do with “noisy” binary outputs?
  - Holes
  - Extra small fragments
- How to demarcate multiple regions of interest?
  - Count objects
  - Compute further features per object

Connected components

- Various algorithms to compute
  - Recursive (in memory)
  - Two rows at a time (image not necessarily in memory)
  - Parallel propagation strategy
Recursive connected components

- Find an unlabeled pixel, assign it a new label
- Search to find its neighbors, and recursively repeat to find their neighbors till there are no more
- Repeat

Demo [http://www.cosc.canterbury.ac.nz/mukundan/covn/Label.html](http://www.cosc.canterbury.ac.nz/mukundan/covn/Label.html)

Connected components

- Identify distinct regions of “connected pixels”

![Connected components](image)

Connectedness

- Defining which pixels are considered neighbors

4-connected

![4-connected](image)

8-connected

![8-connected](image)

Source: Chailanya Chandra

Sequential connected components

- Labeling a pixel only requires to consider its prior and superior neighbors.
- It depends on the type of connectivity used for foreground (4-connectivity here).

![Sequential connected components](image)

Adapted from J. Serra

Sequential connected components

- Labeling a pixel only requires to consider its prior and superior neighbors.
- It depends on the type of connectivity used for foreground (4-connectivity here).

![Sequential connected components](image)

Adapted from J. Serra

Connected components

- We’ll consider a sequential algorithm that requires only 2 passes over the image.

- **Input**: binary image
- **Output**: “label” image, where pixels are numbered per their component

- Note: foreground here is denoted with black pixels.
**Region properties**

- Given connected components, can compute simple features per blob, such as:
  - Area (num pixels in the region)
  - Centroid (average x and y position of pixels in the region)
  - Bounding box (min and max coordinates)
  - Circularity (ratio of mean dist. to centroid over std)

**Binary image analysis: basic steps (recap)**

- Convert the image into binary form
  - Thresholding
- Clean up the thresholded image
  - Morphological operators
- Extract separate blobs
  - Connected components
- Describe the blobs with region properties
Matlab

- \( N = \text{hist}(Y,M) \)
- \( L = \text{bwlabel}(BW,N); \)
- \( \text{STATS} = \text{regionprops}(L,\text{PROPERTIES}); \)
  - 'Area'
  - 'Centroid'
  - 'BoundingBox'
  - 'Orientation', ...
- \( \text{IM2} = \text{imerode}([\text{IM}, \text{SE}]); \)
- \( \text{IM2} = \text{imdilate}([\text{IM}, \text{SE}]); \)
- \( \text{IM2} = \text{imclose}([\text{IM}, \text{SE}]); \)
- \( \text{IM2} = \text{imopen}([\text{IM}, \text{SE}]); \)

Example using binary image analysis: OCR

Example using binary image analysis: segmentation of a liver

Example using binary image analysis: Bg subtraction + blob detection

Binary images

- Pros
  - Can be fast to compute, easy to store
  - Simple processing techniques available
  - Lead to some useful compact shape descriptors

- Cons
  - Hard to get "clean" silhouettes
  - Noise common in realistic scenarios
  - Can be too coarse of a representation
  - Not 3d
### Summary

- **Operations, tools**
  - Derivative filters
  - Smoothing, morphology
  - Thresholding
  - Connected components
  - Matched filters
  - Histograms

- **Features, representations**
  - Edges, gradients
  - Blobs/regions
  - Local patterns
  - Textures (next)
  - Color distributions

### Next

- Texture: read 10.5
- Pset 1 out tonight, due in 2 weeks