Binary Image Proc: week 2

- Imaging devices
- Getting a binary image
- Morphological filtering
- Connected components extraction
- Extracting numerical features from regions
Imaging Devices

Summary of several common imaging devices covered

(Chapter 2 of Shapiro and Stockman)
Major issues

What radiation is sensed?
Is motion used to scan the scene or are all sensing elements static?
How fast can sensing occur?
What are the sensing elements?
How is resolution determined?
  -- in intensity
  -- in space
CCD type camera

Array of small fixed elements
Can read faster than TV rates

*Can add refracting elts to get color in 2x2 neighborhoods*

8-bit intensity common
Color CCD – Bayer Filter
Blooming Problem with Arrays

Difficult to insulate adjacent sensing elements.

Charge often leaks from hot cells to neighbors, making bright regions larger.

http://www.imaging-resource.com/news/2016/08/18/mit-researchers-have-developed-a-camera-that-will-never-over-expose-an-image
8-bit intensity can be clipped

Dark grid intersections at left were actually brightest of scene. In A/D conversion the bright values were clipped to lower values.
“Barrel distortion” of rectangular grid is common for cheap lenses ($50). “Pin cushion” opposite.

Precision lenses can cost $1000 or more.

Zoom lenses often show severe distortion.
Other important effects/problems

Discrete pixel effect: signal is integrated

Mixed pixel effect: materials mix

Thin features can be missed

Measurements in pixels have error
Orbiting satellite scanner

View earth 1 pixel at a time (through a straw)
Prism produces multispectral pixel
Image row by scanning boresight
All rows by motion of satellite in orbit
Scanned area of earth is a parallelogram, not a rectangle
Human eye as a spherical camera

100M sensing elts in retina
Rods sense intensity
Cones sense color
Fovea has tightly packed elts, more cones
Periphery has more rods
Focal length is about 20mm
Pupil/iris controls light entry
Surface data (2.5D) sensed by structured light sensor

Projector projects plane of light on object
Camera sees bright points along an imaging ray
Compute 3D surface point via line-plane intersection

REF: Minolta Vivid 910 camera

TRIANGULATING 3D SENSOR

Plane equation

\[ ax_w + by_w + cz_w = d \]

Laser stripe projector

Camera

CCD

X_c

Y_c

Z_c

3D object

Imaging ray yields two linear equations in the 3 unknowns

\( (x_w, y_w, z_w) \)
Structured light

2.5D face image from Minolta Vivid 910 scanner in the CSE PRIP Lab
Freehand Laser Scanning Using Mobile Phone

https://www.youtube.com/watch?v=_Nr1Jh4NsFs
LIDAR also senses surfaces

Single sensing element scans scene
Laser light reflected off surface and returned
Phase shift codes distance
Brightness change codes albedo

https://www.youtube.com/watch?v=eBUCGxZq_xg
SICK sensor range image

Note human hands at the left and right and roll of tape at the center. Used by NASA navigator and Axion Racing Team in DARPA autonomous navigation grand challenge.

New IBEO device yields 4 line video
What about human vision?

Do we compute the surfaces in our environment?
Do we represent them in our memory as we move around or search?
Do we save representations of familiar objects?

(See David Marr, Vision, 1982; Aloimonus and Rosenfeld 1991.)
3D scanning technology

3D image of voxels obtained
Usually computationally expensive
reconstruction of 3D from many 2D scans (CAT computer-aided-
tomography)

More info later in the course.
Magnetic Resonance Imaging

Sense density of certain chemistry
S slices x R rows x C columns
Volume element (voxel) about 2mm per side
At left is shaded 2D image created by “volume rendering” a 3D volume: darkness codes depth

Single slice through human head

MRIs are computed structures, computed from many views.

At left is MRA (angiograph), which shows blood flow.

CAT scans are computed in much the same manner from X-ray transmission data.
Other variations

Microscopes, telescopes, endoscopes, ...

X-rays: radiation passes through objects to sensor elements on the other side

Fibers can carry image around curves; in bodies, in machine tools

Pressure arrays create images (fingerprints, butts)

Sonar, stereo, focus, etc can be used for range sensing (see Chapters 12 and 13)
Summary of some digital imaging problems.

Mixed pixel problem: mixed material in the world/scene map to a single pixel
Loss of thin/small features due to resolution
Variations in area or perimeter due to variation in object location and thresholding
Path problems: refraction, dispersion, etc. takes radiation off straight path (blue light bends more or less than red light going through lens?)
Quick look at thresholding

- Separate objects from background.
- 2 class or many class problem?
- How to do it?
- Discuss methods later.
Cherry image shows 3 regions

- Background is black
- Healthy cherry is bright
- Bruise is medium dark
- Histogram shows two cherry regions (black background has been removed)

Use this gray value to separate
Choosing a threshold

- Common to find the deepest valley between two modes of bimodal histogram
- Or, can level-slice using the intensities values a and b that bracket the mode of the desired objects
- Can fit two or more Gaussian curves to the histogram
- Can do optimization on above (Ohta et al)
Example red blood cell image

- Many blood cells are separate objects
- Many touch – bad!
- Salt and pepper noise from thresholding
- How useable is this data?
Cleaning up thresholding results

- Can delete object pixels on boundary to better separate parts.
- Can fill small holes
- Can delete tiny objects
- (last 2 are “salt-and-pepper” noise)
Removing salt-and-pepper

- *Change pixels all (some?) of whose neighbors are different* (coercion!): see hole filled at right

- Delete objects that are tiny relative to targets: see some islands removed at right
Simple morphological cleanup

- Can be done just after thresholding
  -- remove salt and pepper
- Can be done after connected components are extracted
  -- discard regions that are too small or too large to be the target
Dilation & Erosion

Basic operations

Are dual to each other:

Erosion shrinks foreground, enlarges background

Dilation enlarges foreground, shrinks background

Slides contributed by Volker Krüger & Rune Andersen, University of Aalborg, Esbjerg
A first Example: Erosion

Erosion is an important morphological operation.

Applied Structuring Element:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Set of coordinate points =

\{ (-1, -1), (0, -1), (1, -1), \\
(-1, 0), (0, 0), (1, 0), \\
(-1, 1), (0, 1), (1, 1) \}
Example for Erosion

Input image:

```
1 0 0 0 1 1 1 1 0 1 1 1
```

Structuring Element:

```
1 1 1
```

Output Image:

```
0
```
Example for Erosion

Input image:

1 0 0 0 1 1 1 1 0 1 1

Structuring Element:

1 1 1

Output Image:

0 0
Example for Erosion

Input image

Structuring Element

Output Image
Example for Erosion

Input image:

```
1 0 0 0 1 1 1 1 0 1 1
```

Structuring Element:

```
1 1 1 1
```

Output Image:

```
0 0 0 0 0
```
Example for Erosion

Input image: 1 0 0 0 1 1 1 1 0 1 1

Structuring Element: 1 1 1

Output Image: 0 0 0 0 0 1
Example for Erosion

Input Image:

| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |

Structuring Element:

| 1 | 1 | 1 |

Output Image:

| 0 | 0 | 0 | 0 | 1 | 0 |   |   |   |   |   |
Example for Erosion

Input image:

1 0 0 0 1 1 1 0 1 1

Structuring Element:

1 1 1

Output Image:

0 0 0 0 1 0 0 0
Example for Erosion

Input image

1 0 0 0 1 1 1 1 0 1 1 1

Structuring Element

1 1 1 1

Output Image

0 0 0 0 1 0 0 0 0
Another example of erosion

White = 0, black = 1, dual property, image as a result of erosion gets darker
Counting Coins

Counting coins is difficult because they touch each other!

Solution: Binarization and Erosion separates them!
Example: Dilation

Dilation is an important morphological operation.

Applied Structuring Element:

```
1 1 1
1 1 1
1 1 1
```

Set of coordinate points =

\{ (-1, -1), (0, -1), (1, -1),
    (-1, 0), (0, 0), (1, 0),
    (-1, 1), (0, 1), (1, 1) \}
Example for Dilation

Input image:

1 0 0 0 1 1 1 1 0 1 1

Structuring Element:

1 1 1

Output Image:

1
Example for Dilation

Input image: 

\[
\begin{array}{cccccccccc}
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\
\end{array}
\]

Structuring Element: 

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

Output Image: 

\[
\begin{array}{c}
1 & 0 \\
\end{array}
\]
Example for Dilation

Input image

Structuring Element

Output Image
Example for Dilation

Input image

| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |

Structuring Element

| 1 | 1 | 1 | 1 |

Output Image

| 1 | 0 | 1 | 1 |
Example for Dilation

Input image:

```
1 0 0 0 1 1 1 1 0 1 1 1
```

Structuring Element:

```
1 1 1
```

Output Image:

```
1 0 1 1 1
```
Example for Dilation

Input image:

```
1 0 0 0 1 1 1 1 0 1 1
```

Structuring Element:

```
1 1 1
```

Output Image:

```
1 0 1 1 1 1 1
```
Example for Dilation

Input image: 1 0 0 0 1 1 1 1 0 1 1

Structuring Element: 1 1 1

Output Image: 1 0 1 1 1 1 1 1 1
Example for Dilation

Input image

Structuring Element

Output Image
Another Dilation Example

Image gets lighter, more uniform intensity
Edge Detection

1. Dilate input image
2. Subtract input image from dilated image
3. Edges remain!
Opening & Closing

Important operations

Derived from the fundamental operations

Erosion next dilation -> opening
Dilation next erosion -> closing

Usually applied to binary images, but gray value images are also possible

Opening and closing are dual operations
Connected components

- Assume thresholding obtained binary image
- Aggregate pixels of each object
- 2 different program controls
- Different uses of data structures
- Related to paint/search algs
- Compute features from each object region
Notes on Binary Image Proc

- Connected Components Algorithms
- Separate objects from background
- Aggregate pixels for each object
- Compute features for each object
- Different ways of program control
- Different uses of data structures
- Related to paint/search algs
CC analysis of red blood cells

- 63 separate objects detected
- Single cells have area about 50
- Noise spots
- Gobs of cells

<table>
<thead>
<tr>
<th>Object</th>
<th>Area</th>
<th>Centroid</th>
<th>Bounding Box</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>383</td>
<td>(8.8, 20)</td>
<td>[1 22 1 39]</td>
</tr>
<tr>
<td>2</td>
<td>83</td>
<td>(5.8, 50)</td>
<td>[1 11 42 55]</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>(1.5, 57)</td>
<td>[1 2 55 60]</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>(1, 62)</td>
<td>[1 1 62 62]</td>
</tr>
</tbody>
</table>
| 5      | 1048 | (19, 75) | [1 40 35 100]| gobs
| 32     | 45   | (43, 32) | [40 46 28 35]|
| 33     | 11   | (44, 1e+02)|[41 47 98 100]| cell
| 34     | 52   | (45, 87) | [42 48 83 91]| cell
| 35     | 54   | (48, 53) | [44 52 49 57]| cell
| 60     | 44   | (88, 78) | [85 90 74 82]| cell
| 61     | 1    | (85, 94) | [85 85 94 94]| cell
| 62     | 8    | (90, 2.5)| [89 90 1 4]  | cell
| 63     | 1    | (90, 6)  | [90 90 6 6]  |
More control of imaging

- More uniform objects
- More uniform background
- Thresholding works
- Objects actually separated
Results of “pacmen” analysis

- 15 objects detected
- Location known
- Area known
- 3 distinct clusters of 5 values of area; 85, 145, 293
Results of “coloring” objects

- Each object is a connected set of pixels
- Object label is “color”
- How is this done?
Extracting “white regions”

- Program learns white from training set of sample pixels.
- Aggregate similar white neighbors to form regions.
- Regions can then be classified as characters.
- (Work contributed by David Moore, CSE 803.)

(Left) input RGB image of a NASA sign

(Right) output is a labeled image.
Extracting components: Alg A

- Collect connected foreground pixels into separate objects – label pixels of same object with same color
- A) collect by “random access” of pixels using “paint” or “fill” algorithm
paint/fill algorithm

- Obj region must be bounded by background
- Start at any pixel \([r, c]\) inside obj
- Recursively color neighbors
Events of paint/fill algorithm

- PP denotes “processing point”
- If PP outside image, return to prior PP
- If PP already labeled, return to prior PP
- If PP is background pixel, return to prior PP
- If PP is unlabeled object pixel, then
  1) label PP with current color code
  2) recursively label neighbors N1, ..., N8 (or N1, ..., N4)
Recursive Paint/Fill Alg: 1 region

- Color closed boundary with L
- Choose pixel \([r,c]\) inside boundary
- Call FILL

FILL (I, r, c, L)

If \([r,c]\) is out, return
If \(I[r,c] == L\), return
\(I[r,c] \leftarrow L\)  // color it
For all neighbors \([rn,cn]\)
  FILL(I, rn, cn, L)
Recall different neighborhoods

- 4-connected: right, up, left, down only
- 8-connected: have 45 degree diagonals also
- If the pixels \([r,c]\) of an image region \(R\) are 4-connected, then there exists a path from any pixel \([r_1,c_1]\) in \(R\) to any other pixel \([r_2,c_2]\) in \(R\) using only 4 neighbors
- *The coloring algorithm is guaranteed to color all connected pixels of a region. Why?*
Correctness of recursive coloring

- If any pixel P is colored L, then all of its object neighbors must be colored L (because FILL will be recursively called on each neighbor).
- If pixels P and Q are in connected region R, then there must be a path from pixel P to Q inside of R.
- If FILL does not color Q, let X be the first uncolored pixel on a path from P to Q.
- If a neighbor N of X is colored, then so must be X! This contradiction proves the property – all of the path must be colored.
Connected components using recursive Paint/Fill

- Raster scan until object pixel found
- Assign new color for new object
- Search through all connected neighbors until the entire object is labeled
- Return to the raster scan to search for another object pixel
Extracting 5 objects

a) binary image

b) 5 components
Outside material to cover

- Look at C++ functions for raster scanning and pixel “propagation”
- Study related fill algorithm
- Discuss how the recursion works
- Prove that all pixels connected to the start pixel must be colored the same
- Runtime stack can overflow if region has many pixels.
Alg B: raster scan control

Visit each image pixel once, going by row and then column.
Propagate color to neighbors below and to the right.
Keep track of merging colors.
Have caution in programming

- Note how your image tool coordinates the pixels.
- Different tools or programming languages handle rows and columns differently
- Text uses EE-TV coordinates or standard math coordinates
Events controlled by neighbors

- If all \( N_i \) background, then PP gets new color code
- If all \( N_i \) same color \( L \), then PP gets \( L \)
- If \( N_i \neq N_j \), then take smallest code and “make” all same
- See Ch 3.4 of S&S
Merging connecting regions

Detect and record merges while raster scanning.

Use equivalence table to recode

(in practice, probably have to start with color = 2, not 1)
Alg A   versus   alg B

- Visits pixels more than once
- Needs full image
- Recursion or stacking slower than B
- No need to recolor
- Can compute features on the fly
- Can quit if search object found (tracking?)

“visits” each pixel once
Needs only 2 rows of image at a time
Need to merge colors and region features when regions merge
Typically faster
Not suited for heuristic (smart) start pixel
Outside material

- More examples of raster scanning
- Union-find algorithm and parent table
- Computing features from labeled object region
- More on recursion and C++
Computing features of regions

Can postprocess results of CC alg.
Or, can compute as pixels are aggregated
Area and centroid

- We denote the set of pixels in a region by $R$.
- assuming square pixels:
  - area:
    
    $A = \sum_{(r,c)\in R} 1$
  
  - centroid:
    
    $\bar{r} = \frac{1}{A} \sum_{(r,c)\in R} r$
    $\bar{c} = \frac{1}{A} \sum_{(r,c)\in R} c$

- $(\bar{r}, \bar{c})$ is generally not a pair of integers.

- A precision of tenths of a pixel is often justifiable for the centroid.
Bounding box: BB

- Smallest rectangle containing all pixels
- r-low is lowest row value; c-low is lowest column value
- r-high, c-high are the highest values
- Bounding box has diagonal corners [r-low, c-low] and [r-high, c-high]
- BBs summarize where an object is; they can be used to rule out near collisions
Second moments

second-order row moment:

\[ \mu_{rr} = \frac{1}{A} \sum_{(r,c) \in R} (r - \bar{r})^2 \]

second-order mixed moment:

\[ \mu_{rc} = \frac{1}{A} \sum_{(r,c) \in R} (r - \bar{r})(c - \bar{c}) \]

second-order column moment:

\[ \mu_{cc} = \frac{1}{A} \sum_{(r,c) \in R} (c - \bar{c})^2 \]

These are invariant to object location in the image.
Contrast second moments

- For the letter ‘I’
- Versus the letter ‘O’
- Versus the underline ‘_’
Perimeter pixels and length

- Let perimeter $P$ be the actual set of boundary pixels.
- $P$ must be ordered in a sequence $P = (r_0, c_0, \ldots, r_{K-1}, c_{K-1})$.
- Each pair of successive pixels in $P$ are neighbors, including the first and last pixels.

**Perimeter length:**

$$|P| = \# \{k | (r_{k+1}, c_{k+1}) \in N_4(r_k, c_k) \}$$

$$+ \sqrt{2} \# \{ k | (r_{k+1}, c_{k+1}) \in N_8(r_k, c_k) - N_4(r_k, c_k) \}$$

where $k + 1$ is computed modulo $K$.

How do we find these boundary pixels?

- Perimeter can vary significantly with object orientation.
Circularity or elongation

- common measure of circularity of a region is length of the perimeter squared divided by area. 
  \[ \text{circularity}(1): \]
  \[ C_1 = \frac{|P|^2}{A} \]

Computing $P$ is more trouble than computing $A$. 
Circularity as variance of "radius"

- A second measure uses variation off of a circle circularity (2):

\[ C_2 = \frac{\mu_R}{\sigma_R} \]

where \( \mu_R \) and \( \sigma_R^2 \) are the mean and variance of the distance from the centroid of the shape to the boundary pixels \((r_k, c_k)\).

- Mean radial distance:

\[ \mu_R = \frac{1}{K} \sum_{k=0}^{K-1} \| (r_k, c_k) - (\bar{r}, \bar{c}) \| \]

- Variance of radial distance:

\[ \sigma_R^2 = \frac{1}{K} \sum_{k=0}^{K-1} [\| (r_k, c_k) - (\bar{r}, \bar{c}) \| - \mu_R]^2 \]
Axis of least (most) inertia

- gives object oriented, or natural, coordinate system
- passes through centroid
- axis of most inertia is perpendicular to it
- concept extends to 3D and nD
Approach (in Cartesian/math coordinates)

- First assume axis goes through centroid
- Assume axis makes arbitrary angle $\Theta$ with the horizontal axis
- Write formula for the angular momentum (inertia) to rotate a single unit of mass (pixel) about the axis
- Write formula for the sum over all pixels
- Take derivative of formula to find $\Theta$ for extreme values
Rotational inertia of pixel

Axis of inertia

Centroid of pixels

$\Theta + 90$
Overall steps

- \( d = [x, y] \) projected along normal to axis
  \[ = -x \sin \Theta + y \cos \Theta \]
- Inertia of single pixel is \( d \) squared
- Inertia of entire region is sum over all pixels
  \[ I = \sum (x \sin \Theta - y \cos \Theta)^2 \]
  \[ = \sin^2 \Theta \sum x^2 - 2\sin \Theta \cos \Theta \sum xy + \cos^2 \Theta \sum y^2 \]
  \[ = \sin^2 \Theta a - 2\sin \Theta \cos \Theta b + \cos^2 \Theta c \]

where \( a, b, c \) are the second moments!
Final form

- By differentiating İ with respect to Θ and setting to 0
  - 0 = a tan 2 Θ - 2b - c tan 2 Θ
  - tan 2 Θ = 2b/(a-c)  pretty simple
- What if all pixels [x,y] lie on a circle?
- What if they are all on a line?
As in text with centroid

\[
\tan 2\hat{\alpha} = \frac{2 \sum (r - \bar{r})(c - \bar{c})}{\sum (r - \bar{r})(r - \bar{r}) - \sum (c - \bar{c})(c - \bar{c})}
\]

\[
= \frac{1 \over A} \frac{2 \sum (r - \bar{r})(c - \bar{c})}{\sum (r - \bar{r})(r - \bar{r}) - \frac{1}{A} \sum (c - \bar{c})(c - \bar{c})}
\]

\[
= \frac{2 \mu_{rc}}{\mu_{rr} - \mu_{cc}} \quad (3.23)
\]
formula for best axis

- use least squares to derive the tangent angle $\theta$ of the axis of least inertia
- express $\tan 2\theta$ in terms of the 3 second moments
- interpret the formula for a circle of pixels and a straight line of pixels
Applications

An iterative approach for fitting multiple connected ellipse structure to silhouette, PRL, 2010