Announcements

• See links on web page for reading on binary image processing (e-reserves)
• Reading on filtering is in text

Binary Image Processing: Basic Steps

1. Labeling pixels as foreground/background (0,1).
2. Morphological operators (sometimes)
3. Find pixels corresponding to a region
4. Compute properties of each region

Color wrapup

• CIE color space
• RGB-> HSV
• Metamers

Histogram-based Segmentation

Ex: bright object on dark background:

<table>
<thead>
<tr>
<th>Gray value</th>
<th>Number of pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td></td>
</tr>
</tbody>
</table>

- Select threshold
- Create binary image:
  - I(x,y) < T -> O(x,y) = 0
  - I(x,y) > T -> O(x,y) = 1

P-Tile Method

- If the size of the object is approx. known, pick T s.t. the area under the histogram corresponds to the size of the object:
"Peakiness" Detection Algorithm

- Find the two HIGHEST LOCAL MAXIMA at a MINIMUM DISTANCE APART: $g_i$ and $g_j$
- Find lowest point between them: $g_k$
- Measure “peakiness”:
  - $\min(H(g_i),H(g_j))/H(g_k)$
- Find $(g_i, g_j, g_k)$ with highest peakiness

Recursive Labeling
Connected Component Exploration

Procedure Label (Pixel)
BEGIN
Mark(Pixel) <- Marker;
FOR neighbor in Neighbors(Pixel) DO
  IF Image (neighbor) = 1 AND Mark(neighbor)=nil THEN
    Label(neighbor)
END
BEGIN Main
Marker <- 0;
FOR Pixel in Image DO
  IF Image(Pixel) = 1 AND Mark(Pixel)=nil THEN
    BEGIN
      Marker <- Marker + 1;
      Label(Pixel);
    END;
  END
END

Globals:
Marker: integer
Mark: Matrix same size as Image, initialized to NIL

Properties extracted from binary image
- A tree showing containment of regions
- Properties of a region
  1. Genus – number of holes
  2. Centroid
  3. Area
  4. Perimeter
  5. Moments (e.g., measure of elongation)
  6. Number of “extrema” (indentations, bulges)
  7. Skeleton

Moments
(related to moments of intertia)
$\sum S = \{(x, y)|f(x, y) = 1\}$

Given a pair of non-negative integers $(j,k)$ the discrete $(j,k)^{th}$ moment of $S$ is:

$M_{jk}(S) = \sum_{(x,y) \in S} x^j y^k$

- Fast way to implement computation over n by m image or window
- One object
**Area: Moment** $M_{00}$

$S = \{(x, y) | f(x, y) = 1\}$

$M_{jk}(S) = \sum_{(x,y) \in S} x^j y^k$

Example:

$M_{00}(S) = \sum_{(x,y) \in S} x^0 y^0 = \sum_{(x,y) \in S} 1 = \#(S)$

**Computing the centroid with Moments**

$S = \{(x, y) | f(x, y) = 1\}$

$M_{jk}(S) = \sum_{(x,y) \in S} x^j y^k$

Example:

$M_{00}(S) = \sum_{(x,y) \in S} x^0 y^0 = \sum_{(x,y) \in S} 1 = \#(S)$

$M_{10}(S) \frac{\sum_{(x,y) \in S} x}{\#(S)} = \overline{x}$

$M_{01}(S) \frac{\sum_{(x,y) \in S} y}{\#(S)} = \overline{y}$

Center of gravity (Centroid) of $S$!!

**Shape recognition by Moments**

Recognition could be done by comparing moments

However, moments $M_{jk}$ are not invariant under:

- Translation
- Scaling
- Rotation
- Skewing

**Central Moments**

Given a pair of non-negative integers $(j,k)$ the central $(j,k)^{th}$ moment of $S$ is given by:

$\mu_{jk}(S) = \sum_{(x,y) \in S} (x - \overline{x})^j (y - \overline{y})^k$

**Normalized Moments**

Given a pair of non-negative integers $(j,k)$ the normalized $(j,k)^{th}$ moment of $S$ is given by:

$m_{jk}(S) = \sum_{(x,y) \in S} \left( \frac{x - \overline{x}}{\sigma_x} \right)^j \left( \frac{y - \overline{y}}{\sigma_y} \right)^k$
Normalized Moments
\[ S = \{(x, y)|f(x, y) = 1\} \]

Scaling by (a, c) and translating by T = (b, d):
\[ S_{ST} = \{(x^*, y^*)|x^* = ax+b, y^* = cy+d, (x, y) \in S\} \]

Scaling and translation INVARIANT!

Region orientation from Second Moment Matrix
1. Compute second centralized moment matrix
\[
\begin{bmatrix}
\mu_{20} & \mu_{11} \\
\mu_{11} & \mu_{02}
\end{bmatrix}
\]
2. Compute Eigenvectors of Moment Matrix to obtain orientation
3. Eigenvalues are independent of orientation, translation!

Scale, Rotation, Translation Invariance
1. Compute second normalized moment matrix
2. Eigenvectors give orientation of object.
3. Eigenvalues are translation, rotation, and scale invariance.

Binary System Summary
1. Acquire images and binarize (thresholding, color labels, etc.).
2. Possibly clean up image using morphological operators.
3. Determine regions (blobs) using connected component exploration.
4. Compute position, area, and orientation of each blob using Moments.
5. Compute features that are rotation, scale, and orientation invariant using Moments (e.g., Eigenvalues of Normalized Moments).

Binarization using Color
- Object’s in robocup are distinguished by color.
- How do you binarize the image so that pixels where ball is located are labeled with 1, and other locations are 0?
- Let \( C_b = (r \ g \ b)^T \) be the color of the ball.

Binarization using Color
- Let \( c(u,v) \) be the color of pixel \((u,v)\)
- Simple method
\[
b(u,v) = \begin{cases} 
1 & \text{if} \|c(u,v) - c_b\|^2 \leq \varepsilon \\
0 & \text{otherwise}
\end{cases}
\]
- Better alternative (why?)
  - Convert \( c(u,v) \) to HSV space \( H(u,v), S(u,v) \ V(u,v) \)
  - Convert \( c_b \) to HSV
  - Check that HS distance is less than threshold \( \varepsilon \) and brightness is greater than a threshold \( V > \tau \)
Blob Tracking

Main tracking notions

- State: usually a finite number of parameters (a vector) that characterizes the “state” (e.g., location, size, moments, pose) of thing being tracked. (e.g., $\Phi$)
- Dynamics: How does the state change over time? How is that changed constrained? (e.g., $d\Phi/dt$)
- Trajectory: $\Phi(t)$
- Prediction: Given the state at time $t-1$, what is an estimate of the state at time $t$?
- Data Association: Given predicted state, and measurement of multiple blobs in image at time $t$, which blob is being tracked?

Filtering

Image Filtering

Why perform filtering?

1. Reduce the effect of “noise”
2. Extract descriptions of a neighborhood of an image about a point.
3. Extract features of an image (edges, texture, corners).
4. Enhance aspects of an image, visual improvement, etc.

What is image filtering?

- Modify the pixels in an image based on some function of a local neighborhood of the pixels.

(From Bill Freeman)
Linear Filters

- General process:
  - Form new image whose pixels are a weighted sum of original pixel values, using the same set of weights at each point.
- Properties
  - Output is a linear function of the input
  - Output is a shift-invariant function of the input (i.e., shift the input image two pixels to the left, the output is shifted two pixels to the left).
- Example: smoothing by averaging
- Example: smoothing with a Gaussian
- Example: finding a derivative

Example: smoothing by averaging
- Form the average of pixels in a neighborhood

Example: smoothing with a Gaussian
- Form a weighted average of pixels in a neighborhood

Example: finding a derivative
- Form a weighted average of pixels in a neighborhood

Convolution

Image (I) * Kernel (K)

Note: Typically Kernel is relatively small in vision applications.

Kernel size is $m+1$ by $m+1$

Convolution: $R = K \ast I$

$R(i, j) = \sum_{h=-m}^{m} \sum_{k=-m}^{m} K(h, k)I(i-h, j-k)$

Kernel size is $m+1$ by $m+1$
Convolution: $R = K \ast I$

Kernel size is $m+1$ by $m+1$

$$R(i, j) = \sum_{h=-m}^{m} \sum_{k=-m}^{m} K(h, k) I(i-h, j-k)$$
Convolution: \( R = K \ast I \)

- Kernel size is \( m+1 \) by \( m+1 \)

\[
R(i, j) = \sum_{h=-m}^{m} \sum_{k=-m}^{m} K(h, k) I(i-h, j-k)
\]