Lecture 7  Chamfer Matching and Distance Transform

Chamfer Matching: 2D Model projection to edge matching

For every edge point in the transformed object, compute the distance to the nearest image edge point. Sum distances.

\[
\sum_{i=1}^{n} \min(\| p_i, q_1 \|, \| p_i, q_2 \|, \ldots \| p_i, q_m \|)\]
Given:
- binary image, B, of edge and local feature locations
- binary “edge template”, T, of shape we want to match

Let D be an array in registration with B such that D(i,j) is the distance to the nearest “1” in B.
- this array is called the distance transform of B

Goal: Find placement of T in D that minimizes the sum, M, of the distance transform multiplied by the pixel values in T
- if T is an exact match to B at location (i,j) then M(i,j) = 0
- but if the edges in B are slightly displaced from their ideal locations in T, we still get a good match using the distance transform technique

Computing the distance transform
- Brute force, exact algorithm, is to scan B and find, for each “0”, its closest “1” using the Euclidean distance.
  – expensive in time, and difficult to implement
<table>
<thead>
<tr>
<th>Distance Metric</th>
<th>Description</th>
<th>Illustration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean</td>
<td>The Euclidean distance is the straight-line distance between two pixels.</td>
<td><img src="image1" alt="Illustration" /></td>
</tr>
<tr>
<td>City Block</td>
<td>The city block distance metric measures the path between the pixels based on a 4-connected neighborhood. Pixels whose edges touch are 1 unit apart; pixels diagonally touching are 2 units apart.</td>
<td><img src="image2" alt="Illustration" /></td>
</tr>
<tr>
<td>Chessboard</td>
<td>The chessboard distance metric measures the path between the pixels based on an 8-connected neighborhood. Pixels whose edges or corners touch are 1 unit apart.</td>
<td><img src="image3" alt="Illustration" /></td>
</tr>
<tr>
<td>Quasi-Euclidean</td>
<td>The quasi-Euclidean metric measures the total Euclidean distance along a set of horizontal, vertical, and diagonal line segments.</td>
<td><img src="image4" alt="Illustration" /></td>
</tr>
</tbody>
</table>

```matlab
a =
0   0   0   0   0   0
0   0   0   0   0   0
0   0   0   0   0   0
0   0   0   0   0   0
0   0   0   0   0   0
0   0   0   0   0   0

>> c = bwdist(a, 'Quasi-Euclidean')
c =
2.8284  2.4142  2.0000  2.4142  2.8284
2.4142  1.4142  1.0000  1.4142  2.4142
2.0000  1.0000  0.0000  1.0000  2.0000
2.4142  1.4142  1.0000  1.4142  2.4142
2.8284  2.4142  2.0000  2.4142  2.8284
```
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>'chessboard'</td>
<td>In 2-D, the chessboard distance between ((x_1, y_1)) and ((x_2, y_2)) is (\max(</td>
</tr>
<tr>
<td>'cityblock'</td>
<td>In 2-D, the cityblock distance between ((x_1, y_1)) and ((x_2, y_2)) is (</td>
</tr>
<tr>
<td>'euclidean'</td>
<td>In 2-D, the Euclidean distance between ((x_1, y_1)) and ((x_2, y_2)) is (\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}) This is the default method.</td>
</tr>
<tr>
<td>'quasi-euclidean'</td>
<td>In 2-D, the quasi-Euclidean distance between ((x_1, y_1)) and ((x_2, y_2)) is (</td>
</tr>
</tbody>
</table>

matlab examples.
Variations

• Sum a different distance
  – $f(d) = d^2$
  – or *Manhattan distance*.
  – $f(d) = 1$ if $d >$ threshold, 0 otherwise.
    – This is called *bounded error*.

• Use maximum distance instead of sum.
  – This is called: *directed Hausdorff distance*.

• Use other features
  – Corners.
  – Lines. Then position and angles of lines must be similar.
    • Model line may be subset of image line.

Other “constrained” comparisons

• Enforce the constraint that each image feature can match only one model feature.
• Enforce continuity, ordering along curves.
• These are more complex to optimize.