Matching

Hao Jiang
Computer Science Department
Model Fitting

- An object model constrains the kind of object we wish to find in images.
Type of 2D Models

- Rigid models
- Deformable models
- Articulated models
Template Matching

Template

Target image
Template Matching

Template

Target image
Template Matching

Template

Target image
Template Matching

Template

Target image
Template Matching

Template

Target image
Matching Criteria

At each location \((x,y)\) the matching cost is:

\[
\sum_{(\delta_x, \delta_y) \in W} d(T(\delta_x, \delta_y), I(x + \delta_x, y + \delta_y))
\]
Dealing with Rotations

Template

Target image
Dealing with Scale Changes

Template

Target image in multiple scales

Sweep window and find the best match
Multi-scale Refinement

Start from here

size \( j \)

size \( k \)

Template

size \( m \)

size \( n \)

Target image in multiple scales

Complexity reduced from \( O(n^2 k^2) \) to \( O(\log(n/m) + m^2 j^2) \)
Multi-scale Refinement

1. Generate the image pyramids

2. Exhaustive search the smallest image and get location \((x,y)\)

3. Go to the next level of the pyramid and check the locations \((2x, 2y), (2x+1,2y), (2x,2y+1), (2x+1,2y+1)\), pick up the best and update the estimate.

4. If at the bottom of the pyramid, stop; otherwise go to 3.

5. Output the estimate \((x,y)\)
Binary Template Matching

G.M. Gavrila, “Pedestrian detection from a moving vehicle”, ECCV 2000
Matching Binary Template

- Template
- Clutter
- Target object

Binary target image
For each point $p$ on the red curve, find the closed point $q$ on the green curve. Denote $d(p, q) = d_{\text{red}}(p)$.

The distance from the red curve to the green curve is $\text{mean}_{p \in \text{red}} d_{\text{red}}(p)$.

Similarly, the distance from the green curve to the red curve is $\text{mean}_{q \in \text{green}} d_{\text{green}}(q)$. 
Distance Transform

- Transform binary image into an image whose pixel values equal the closest distance to the binary foreground.
Chamfer Matching

\[ d_{\text{red}}(p) = \text{distI}(p) \]
Chamfer Matching in Matlab

% distance transform and matching
dist = bwdist(imTarget, ‘euclidean’);
map = imfilter(dist, imTemplate, ‘corr’, ‘same’);

% look for the local minima in map to locate objects
smap = ordfilt2(map, 25, true(5));
[r, c] = find((map == smap) & (map < threshold));
Hough Transform

- Template matching may become very complex if transformations such as rotation and scale is allowed.

- A voting based method, Hough Transform, can be used to solve the problem.

- Instead of trying to match the target at each location, we collect votes in a parameter space using features in the original image.
Line Detection
Line Detection
(1) Where is the line?
(2) How many lines?
(3) Which point belongs to which line?
For each point, there are many lines that pass it.
All the lines passing \((x_1,y_1)\) can be represented as another line in the *parameter* space.

\[ b = -x_1a + y_1 \]
The co-linear points vote for the same point in the a-b space.

\[ b = -x_1a + y_1 \]

\[ b = -x_2a + y_2 \]
The co-linear points vote for the same point in the a-b space. The point that receives the most votes corresponds to the line.
Line Parameterization

\[ x \cos(\theta) + y \sin(\theta) = d \]
Parameter Space

\[ x \cos(\theta) + y \sin(\theta) = d \]
Hough Transform Algorithm

Using the polar parameterization:

\[ x \cos \theta + y \sin \theta = d \]

Basic Hough transform algorithm

1. Initialize \( H[d, \theta] = 0 \)
2. for each edge point \( I[x, y] \) in the image
   
   for \( \theta = 0 \) to 180  // some quantization

   \[
   d = x \cos \theta + y \sin \theta \\
   H[d, \theta] += 1
   \]
3. Find the value(s) of \((d, \theta)\) where \( H[d, \theta] \) is maximum
4. The detected line in the image is given by \( d = x \cos \theta + y \sin \theta \)

Hough line demo

Time complexity (in terms of number of votes)?

\[ d = x \cos \theta + y \sin \theta \]

H: accumulator array (votes)

Source: Steve Seitz
Example: Hough transform for straight lines

Image space edge coordinates

Votes

Bright value = high vote count
Black = no votes

Source: K. Grauman
Impact of noise on Hough Image space edge coordinates

Votes

What difficulty does this present for an implementation?

Source: K. Grauman
Impact of noise on Hough

Here, everything appears to be “noise”, or random edge points, but we still see peaks in the vote space.

Source: K. Grauman
Extensions

Extension 1: Use the image gradient

1. same
2. for each edge point $l[x,y]$ in the image
   $\theta = \text{gradient at } (x,y)$
   
   $d = x \cos \theta + y \sin \theta$
   
   $H[d, \theta] += 1$
3. same
4. same
   
   (Reduces degrees of freedom)

\begin{align*}
\nabla f &= \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right] \\
\theta &= \tan^{-1}\left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)
\end{align*}

Source: K. Grauman
Extensions

Extension 1: Use the image gradient

1. same
2. for each edge point \( I[x,y] \) in the image
   - compute unique \((d, \theta)\) based on image gradient at \((x,y)\)
   \[
   H[d, \theta] += 1
   \]
3. same
4. same

(Reduces degrees of freedom)

Extension 2

- give more votes for stronger edges (use magnitude of gradient)

Extension 3

- change the sampling of \((d, \theta)\) to give more/less resolution

Extension 4

- The same procedure can be used with circles, squares, or any other shape…

Source: K. Grauman