Shape Alignment
Shape context & Geometric Blur

Jianbo Shi

Many Slides Taken From Alex Berg
Object Category Recognition
Deformable Template Matching with Exemplars for Recognition

- Use exemplars as deformable templates
- Find a correspondence between the query image and each template
Deformable Template Matching with Exemplars for Recognition

- Use exemplars as deformable templates
- Find a correspondence between the query image and each template

Best matching template is a helicopter
D’Arcy Thompson: On Growth and Form, 1917
studied transformations between shapes of organisms
Correspondence for Deformable Template Matching

- Evaluate correspondence based on:
  - **Similarity of appearance** near feature points
  - **Similarity in configuration** of the feature points
Correspondence for Deformable Template Matching

- Evaluate correspondence based on:
  - Similarity of appearance near feature points
  - Similarity in configuration of the feature points
Correspondence for Deformable Template Matching

- Evaluate correspondence based on:
  - Similarity of appearance near feature points
  - Similarity in configuration of the feature points
Correspondence for Deformable Template Matching

- Evaluate correspondence based on:
  - Similarity of appearance near feature points
  - Similarity in configuration of the feature points
Correspondence Result
Interpolated Correspondence
Using Thin Plate Splines
Step 1: feature correspondence

Goal: high precision (low miss rate), low false positive (false alarm)

Method: use many features, and add geometric context

Features: Shape Context, Geometric Blur
Shape Context

Detect edges, and subsample edge nodes
Shape Context

Count the number of points inside each bin, e.g.:

- Count = 4
- ...
- Count = 10

\[ \Phi \] Compact representation of distribution of points relative to each point
Shape contexts are histograms

Angle bin

radial distance bin
Properties of Shape Context

1) Invariant under translation and scale
2) Can be made invariant to rotation by using local tangent orientation frame
3) Tolerant to small affine distortion
   Log-polar bins make spatial blur proportional to r

Cf. Spin Images (Johnson & Hebert) - range image registration
Matching point features using Shape Context

Compute matching costs using Chi Squared distance:

\[ C_{ij} = \frac{1}{2} \sum_{k=1}^{K} \frac{[h_i(k) - h_j(k)]^2}{h_i(k) + h_j(k)} \]

Recover correspondences by solving linear assignment problem with costs \( C_{ij} \)
Global match between all feature point can be done using Hungarian Bipartite graph matching method.
Example: similar and different shapes
Find best match for the shape context at only a few random points and add up cost

\[ \text{dist}(S_{\text{query}}, S_i) = \sum_{j=1}^{r} \chi^2 (SC^j_{\text{query}}, SC^*_i) \]

\[ SC^*_i = \arg \min_u \chi^2 (SC^j_{\text{query}}, SC^u_i) \]
But if we want to use color and texture features?
Blurry images...

Paul Debevec 1992

CS283 Course Project,
“A Neural Network for Facial Feature Location”
Geometric Blur
(Local Appearance Descriptor)

Compute sparse channels from image
Extract a patch in each channel

Berg & Malik '01
Geometric Blur
(Local Appearance Descriptor)

- Compute sparse channels from image
- Extract a patch in each channel
- Apply spatially varying blur
- Descriptor is robust to small affine distortions

Berg & Malik '01
Geometric blur vs. Gaussian Blur

Geometric Blur is a spatially variant convolution:

\[ G_I(x) = \int_y I(x - y)K_x(y)dy \]

\[ K_x(y) = f(\alpha|x| + \beta)G_{\alpha|x|+\beta}(y), \]
Geometric blur vs. Gaussian Blur

How to compute it fast:

1) compute gaussian blur at different scale across the entire image
2) for each feature point, pick up blurred image at different scale depends on the radial distance to the feature point
Geometric Blur
(Local Appearance Descriptor)

Compute sparse channels from image
Extract a patch in each channel
Apply spatially varying blur and sub-sample
Descriptor is robust to small affine distortions

Geometric Blur Descriptor
(Idealized signal)

Berg & Malik '01
Geometric Blur
(Local Appearance Similarity)
Are Features Enough?

Color indicates similarity using Geometric Blur Descriptor

Not Quite...
Linear Assignment
(e.g. Hungarian)
STEP 2: Enforce Geometric Configuration

- Evaluate correspondence based on:
- Similarity of appearance near feature points
- Similarity in configuration of the feature points
Idea: Thin-Plate Spline Function to "regularize" the geometrical transformation

From a shape to another: TPS transformations

The goal is to find \( f_x, f_y : \mathbb{R}^2 \leftrightarrow \mathbb{R}^2 \) such as:

\[
\begin{align*}
\forall i \; f_x(x_i, y_i) &= x_i' \\
\min_{f_x} I_g &= \int \int_{\mathbb{R}^2} \left( \frac{\partial^2 g}{\partial x^2} \right)^2 + \left( \frac{\partial^2 g}{\partial x \partial y} \right)^2 + \left( \frac{\partial^2 g}{\partial y^2} \right)^2 \\
f_x(x, y) &= v + v_xx + v_yy + \sum_{i=1}^{n} w_i U(\|(x_i, y_i) - (x, y)\|)
\end{align*}
\]

where \( U(r) = r^2 \log r^2 \), and with the same conditions on \( f_y \).
From a shape to another: TPS transformations

Model

Target
model  target
Overall Shape Matching Steps:

1) Compute interesting points on image A and B. This could be edges, or corners. Any stable features would do.
2) Extract Shape Context or Geometric Blur features on both images.
3) Run Hungarian matching to compute point-wise correspondence between A and B.
4) Compute Thin-plate spline (TPS) mapping between A and B.
5) Warp image A using TPS computed in 4), and repeat step 1-4.
• MNIST 60 000:
  – linear: 12.0%
  – 40 PCA+ quad: 3.3%
  – 1000 RBF +linear: 3.6%
  – K-NN: 5%
  – K-NN (deskewed): 2.4%
  – K-NN (tangent dist.): 1.1%
  – SVM: 1.1%
  – LeNet 5: 0.95%

• MNIST 60 000 (distortions):
  – LeNet 5: 0.8%
  – SVM: 0.8%
  – Boosted LeNet 4: 0.7%

• MNIST 20 000:
  – K-NN, Shape Context matching: 0.63%
Model Building for Segmentation

Rough correspondence to each example image

Average quality of alignment

Threshold
Correspondence Examples
(Shape Matching)
Correspondence Examples
(Shape Matching)
Correspondence Examples (Shape Matching)
Correspondence Examples
(Shape Matching)
Correspondence Examples (Shape Matching)
Correspondence Examples
(Shape Matching)
Correspondence Examples (Shape Matching)
Correspondence Examples
(Shape Matching)
Recognizing Objects in Adversarial Clutter:

Greg Mori and Jitendra Malik
What is a CAPTCHA?

- CAPTCHA: Completely Automated Public Turing test to Tell Computers and Humans Apart (Blum et al., CMU)
  - Generates and grades tests that
    - Most humans can pass
    - Current computer programs can’t pass

- Different varieties
  - Word based (Gimpy, EZ-Gimpy)
  - Image based (Pix)
  - Sound based (Bongo)
EZ-Gimpy

• Word-based CAPTCHA
  – Task is to read a single word obscured in clutter

• Currently in use at Yahoo! and Ticketmaster
  – Filters out ‘bots’ from obtaining free email accounts, buying blocks of tickets
CAPTCHAs as Object Recognition Datasets

• Pros:
  – Large number of objects (600 words)
  – Practically infinite set of test images
  – Quantitative results
  – “Adversarial”, intended to be difficult for computers

• Cons:
  – No variation due to pose, lighting
  – Synthetic objects
Object Recognition Framework

• Match objects using shape cues
  – Represented as a point set extracted using Canny edge detection

• A two stage approach
  – Fast pruning
    • Quick tests to construct a shortlist of candidate objects
    • Database of known objects could be large
  – Detailed matching
    • Perform computationally expensive comparisons on only the few shapes in the shortlist
Shape contexts (Belongie et al. 2001)

Count the number of points inside each bin, e.g.:

Count = 8

...

Count = 7

F Compact representation of distribution of points relative to each point
Features: Generalized Shape Contexts

- Can put more than just point counts in bins
  - Oriented Energy
  - Colour info
  - Optical flow
Fast Pruning: Representative Shape Contexts

- Pick k points in the image at random
  - Compare to all shape contexts for all known letters
  - Vote for closely matching letters
- Keep all letters with scores under threshold
Two Instances

• Algorithm A
  – Bottom up, parts-based approach
  – Find letters first, then form words

• Algorithm B
  – Top down, holistic approach
  – Find entire words immediately
Algorithm A

- Look for letters
  - Representative Shape Contexts
- Find pairs of letters that are “consistent”
  - Letters nearby in space
- Search for valid words
- Give scores to the words
EZ-Gimpy Results with Algorithm A

- 158 of 191 images correctly identified: 83%
  - Running time: ~10 sec. per image (MATLAB, 1 Ghz P3)
Gimpy

- Multiple words, task is to find 3 words in the image
- Clutter is other objects, not texture
Algorithm B: Letters are not enough

- Hard to distinguish single letters with so much clutter
- Find words instead of letters
  - Use long range info over entire word
  - Stretch shape contexts into ellipses
- Search problem becomes huge
  - # of words 600 vs. # of letters 26
  - Prune set of words using opening/closing bigrams
### Results with Algorithm B

<table>
<thead>
<tr>
<th># Correct words</th>
<th>% tests (of 24)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 or more</td>
<td>92%</td>
</tr>
<tr>
<td>2 or more</td>
<td>75%</td>
</tr>
<tr>
<td>3</td>
<td>33%</td>
</tr>
<tr>
<td>EZ-Gimpy</td>
<td>92%</td>
</tr>
</tbody>
</table>

**Images:**
- **Left:** medical, cheese, there, clear, cheese
- **Middle:** dry clear medical
- **Right:** card, arch, plate, door, farm, important
Conclusion

• CAPTCHAs useful as datasets for studying object recognition

• Two stage approach
  – Fast pruning
  – Detailed matching

• 92% success rate on EZ-Gimpy
  – OCR (+hacks) 10-50%

• 33% success rate on Gimpy