Category-level localization

Cordelia Schmid
Category-level localization

• Localization of object outlines

Learning shape-based models

Localizing the objects with the learnt models
Category-level localization

- Localization of object pixels
  - Pixel-level classification, segmentation
Overview

- *Shape-based descriptors*

- Learning deformable shape models
Shape-based features for localization

• Classes with characteristic shape
  – appearance, local patches are not adapted
  – shape-based descriptors are necessary

[Ferrari, Fevrier, Jurie & Schmid, PAMI’08]
Pairs of adjacent segments (PAS)

Contour segment network
[Ferrari et al. ECCV’06]

1. Edgels extracted with Berkeley boundary detector
2. Edgel-chains partitioned into straight contour segments
3. Segments connected at edgel-chains’ endpoints and junctions
Pairs of adjacent segments (PAS)

Contour segment network

PAS = groups of two connected segments

PAS descriptor:

\[
\begin{pmatrix}
l_1 - l_2 \\
\theta_1, \theta_2 \\
\end{pmatrix}
\]

encodes geometric properties of the PAS

scale and translation invariant

compact, 5D
Features: pairs of adjacent segments (PAS)

Example PAS

Why PAS?

+ can cover pure portions of the object boundary

+ intermediate complexity: good repeatability-informativeness trade-off

+ scale-translation invariant

+ connected: natural grouping criterion (need not choose a grouping neighborhood or scale)
PAS codebook

PAS descriptors are clustered into a vocabulary

- Frequently occurring PAS have intuitive, natural shapes
- As we add images, number of PAS types converges to just ~100
- Very similar codebooks come out, regardless of source images

→ general, simple features
Window descriptor

1. Subdivide window into tiles
2. Compute a separate bag of PAS per tile
3. Concatenate these semi-local bags

+ distinctive:
  records *which* PAS appear *where*
  weight PAS by average edge strength

+ flexible:
  soft-assign PAS to types, coarse tiling

+ fast:
  computation with Integral Histograms
Training

1. Learn mean positive window dimensions $M_w \times M_h$
2. Determine number of tiles $T$
3. Collect positive example descriptors

4. Collect negative example descriptors: slide $M_w \times M_h$ window over negative training images
5. Train a linear SVM from positive and negative window descriptors

A few of the highest weighed descriptor vector dimensions (= 'PAS + tile')

+ lie on object boundary (= local shape structures common to many training exemplars)
Testing

1. Slide window of aspect ratio $M_w / M_h$ at multiple scales

2. SVM classify each window + non-maxima suppression
   $\rightarrow$ detections
Experimental results – INRIA horses

Dataset: 170 positive + 170 negative images (training = 50 pos + 50 neg)
wide range of scales; clutter

+ tiling brings a substantial improvement
  optimum at T=30 → used for all other experiments
+ works well: 86% det-rate at 0.3 FPPI (50 pos + 50 neg training images)
Experimental results – INRIA horses

Dataset: 170 positive + 170 negative images (training = 50 pos + 50 neg)
wide range of scales; clutter

+ PAS better than any interest point detector

- all interest point (IP) comparisons with $T=10$, and 120 feature types (= optimum over INRIA horses, and ETHZ Shape Classes)
- IP codebooks are class-specific
Results – ETH shape classes

Dataset: 255 images, 5 classes; large scale changes, clutter
training = half of positive images for a class
    + same number from the other classes (1/4 from each)
testing = all other images
Results – ETH shape classes

Dataset: 255 images, 5 classes; large scale changes, clutter
training = half of positive images for a class
+ same number from the other classes (1/4 from each)
testing = all other images
Generalizing PAS to $k$AS

$k$AS: any path of length $k$ through the contour segment network

scale+translation invariant descriptor with dimensionality $4k-2$

$k = \text{feature complexity}; \text{higher } k \text{ more informative, but less repeatable}$

overall mean det-rates (%)

<table>
<thead>
<tr>
<th></th>
<th>1AS</th>
<th>PAS</th>
<th>3AS</th>
<th>4AS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3 FPPI</td>
<td>69</td>
<td>77</td>
<td>64</td>
<td>57</td>
</tr>
<tr>
<td>0.4 FPPI</td>
<td>76</td>
<td>82</td>
<td>70</td>
<td>64</td>
</tr>
</tbody>
</table>

PAS do best!
Overview

• Localization with shape-based descriptors

• Learning deformable shape models
Learning deformable shape models from images

Training data

Goal: localize boundaries of class instances

Test image

Training: bounding-boxes

Testing: object boundaries

[Ferrari, Jurie, Schmid, IJCV10]
Learn a shape model from training images

Training data

Prototype shape + Deformation model
Match it to the test image
Challenges for learning

**Main issue**
which edgels belong
to the class boundaries?

**Complications**
- intra-class variability
- missing edgels
- produce point correspondences
  (learn deformations)
Challenges for detection

- scale changes
- intra-class variability
- clutter
- fragmented and incomplete contours
Local contour features

**PAS**

Pair of Adjacent Segments

+ *robust*
  
  connect also across gaps

+ *clean*
  
  descriptor encodes the two segments *only*

+ *invariant*
  
  to translation and scale

+ *intermediate complexity*
  
  good compromise between repeatability and informativity
Local contour features

**PAS**
Pair of Adjacent Segments

two PAS in correspondence
→ translation+scale transform
→ use in Hough-like schemes

Clustering descriptors
→ codebook of *PAS types*
(here from mug bounding boxes)
Learning: overview

find models parts
assemble an initial shape
refine the shape
Learning: finding model parts

*Intuition*

PAS on class boundaries reoccur at similar locations/scales/shapes.

Background and details specific to individual examples don’t.
Learning: finding model parts

Algorithm
1. align bounding-boxes up to translation/scale/aspect-ratio
2. create a separate voting space per PAS type
3. soft-assign PAS to types
4. PAS cast ‘existence’ votes in corresponding spaces
Learning: finding model parts

Algorithm
1. align bounding-boxes up to translation/scale/aspect-ratio
2. create a separate voting space per PAS type
3. soft-assign PAS to types
4. PAS cast ‘existence’ votes in corresponding spaces
5. local maxima → model parts
Learning: finding model parts

Model parts
- location + size (wrt canonical BB)
- shape (PAS type)
- strength (value of local maximum)
Learning: finding model parts

Why does it work?

Unlikely unrelated PAS have similar location and size and shape

→ form no peaks!

Important properties

+ see all training data at once

→ robust

+ linear complexity

→ efficient large-scale learning
Learning: assembling an initial shape

Not a shape yet
- multiple strokes
- adjacent parts don’t fit together

Why?
- parts are learnt *independently*

Let’s try to assemble parts into a proper whole

We want single-stroked, long continuous lines!
Learning: assembling an initial shape

Observation

- each part has several occurrences
- can assemble shape variations by selecting different occurrences

Idea

- select occurrences so as to form larger connected aggregates
Learning: assembling an initial shape

Hey, this starts to look like a mug!

+ segments fit well within a block
+ most redundant strokes are gone

Can we do better?

- discontinuities between blocks?
- generic-looking?
Learning: shape refinement

Idea

- treat shape as deformable point set
  and *match it back* onto training images

How?

- robust non-rigid point matcher: TPS-RPM
  (thin plat spline – robust point matching)
- strong initialization:
  align model shape BB over training BB
  → likely to succeed

Chui and Rangarajan, *A new point matching algorithm for non-rigid registration*, CVIU 2003
Learning: shape refinement

**Shape refinement algorithm**

1. Match current model shape back to every training image
   
   *backmatched shapes are in full point-to-point correspondence!*

2. Set model to mean shape
3. Remove redundant points
4. If changed → Iterate to 1
Learning: shape refinement

Final model shape

+ clean (almost only class boundaries)
+ smooth, connected lines
+ generic-looking
+ fine-scale structures recovered (handle arcs)
+ accurate point correspondences spanning training images
Learning: shape deformations

From backmatching
intra-class variation examples,
in complete correspondence

Apply Cootes’ technique
1. shapes = vectors in 2p-D space
2. apply PCA

Deformation model
  . top $n$ eigenvectors covering 95% of variance
  . associated eigenvalues $\lambda_i$ (act as bounds)

$\rightarrow$ valid region of shape space

Tim Cootes, *An introduction to Active Shape Models*, 2000
Learning completed!

Automatic learning of shapes, correspondences, and deformations from unsegmented images
Object detection: overview

Goal

given a test image, localize class instances up to their boundaries

How?

1. Hough voting over PAS matches
   \[\text{rough location+scale estimates}\]

2. use to initialize TPS-RPM

   combination enables true pointwise shape matching to cluttered images

3. constrain TPS-RPM with learnt deformation model
   \[\text{better accuracy}\]
Object detection: Hough voting

**Algorithm**

1. soft-match model parts to test PAS

2. each match
   - translation + scale change
   - vote in accumulator space

3. local maxima
   - rough estimates of object candidates

Leibe and Schiele, DAGM 2004; Shotton et al, ICCV 2005; Opelt et al. ECCV 2006
Object detection: Hough voting

Algorithm

1. soft-match model parts to test PAS
2. each match
   -> translation + scale change
   -> vote in accumulator space
3. local maxima
   -> rough estimates of object candidates

initializations for shape matching!

Leibe and Schiele, DAGM 2004; Shotton et al, ICCV 2005; Opelt et al. ECCV 2006
Object detection: Hough voting

*Remember … soft!*

- vote $\alpha\alpha$ shape similarity
- vote $\alpha\alpha$ edge strength of test PAS
- vote $\alpha\alpha$ strength of model part
- spread vote to neighboring location and scale bins
Object detection: shape matching by TPS-RPM

Initialize
get point sets V (model) and X (edge points)

Goal
find correspondences M & non-rigid TPS mapping

M = (|X|+1)x(|V|+1) soft-assign matrix

Algorithm
1. Update M based on 
   \[ \text{dist}(\text{TPS},X) + \text{orient}(\text{TPS},X) + \text{strength}(X) \]

2. Update TPS:
   - Y = MX
   - fit regularized TPS to V \[\leftrightarrow\] Y

Deterministic annealing:
iterate with T decreasing

\[\rightarrow\] M less fuzzy (looks closer)
\[\rightarrow\] TPS more deformable

Chui and Rangarajan, *A new point matching algorithm for non-rigid registration*, CVIU 2003
TPS-RPM in action!
Object detection: constrained TPS-RPM

Output of TPS-RPM
nice, but sometimes inaccurate
or even not mug-like

Why?
generic TPS deformation model
(prefers smoother transforms)

Constrained shape matching
constrain TPS-RPM by learnt
class-specific deformation model

+ only shapes similar to class members
+ improve detection accuracy
Object detection: constrained TPS-RPM

General idea

- constrain optimization to explore only region of shape space spanned by training examples

How to modify TPS-RPM?

1. Update M
2. Update TPS:
   - $Y = MX$
   - $Y \leftarrow Y^c$
   - fit regularized TPS to $V \leftarrow Y$

Hard constraint, sometimes too restrictive
Object detection: constrained TPS-RPM

**General idea**

constrain optimization to explore only region of shape space spanned by training examples

**Soft constraint variant**

1. Update M
2. Update TPS:
   - \( Y = MX \)
   - \( Y \leftarrow Y^c + \frac{T}{T_{init}} (Y^c - Y) \)
   - fit regularized TPS to \( V \leftarrow Y \)

*soft constraint, Y is *attracted* by the valid region*
Soft constrained TPS-RPM in action!

Original V and X

Transformed V + X

TPS Warping

Transformed V + X

Estimated Shape Y=MX
Object detection: constrained TPS-RPM

Soft constrained TPS-RPM

+ shapes fit data more accurately
+ shapes resemble class members
+ in spirit of deterministic annealing!
+ truly alters the search
  (not fix a posteriori)

Does it really make a difference?
when it does, it’s really noticeable
(about 1 in 4 cases)
Datasets: ETHZ Shape Classes

• 255 images from Google-images, and Flickr
  - uncontrolled conditions
  - variety: indoor, outdoor, natural, man-made, …
  - wide range of scales (factor 4 for swans, factor 6 for apple-logos)

• all parameters are kept fixed for all experiments

• training images: 5x random half of positive; test images: all non-train
Datasets: INRIA Horses

• 170 horse images + 170 non-horse ones
  - clutter, scale changes, various poses

• all parameters are kept fixed for all experiments
• training images: 5x random 50; test images: all non-train images
Results: all learned models
Results: all learned models
Results: all learned models
Results: apple logos
Results: mugs
Results: giraffes
Results: bottles
Results: swans
Results: horses
Results: detection-rate vs false-positives per image

- **Swans**
  - Accuracy: 3.0
  - Full system (green)
  - Hough alone (black)

- **Bottles**
  - Accuracy: 2.4
  - Full system (PASCAL: $\cap \cup >50\%$)

- **Apple logos**
  - Accuracy: 1.5
  - Hough alone (PASCAL)

- **Mugs**
  - Accuracy: 3.1
  - Full system (green)
  - Hough alone (black)

- **Giraffes**
  - Accuracy: 3.5
  - Full system (green)
  - Hough alone (black)

- **INRIA Horses**
  - Accuracy: 5.4
  - Full system (green)
  - Hough alone (black)
Results: Hand-drawings

Same protocol as Ferrari et al, ECCV 2006: match each hand-drawing to all 255 test images
Results: detection-rate vs false-positives per image

- our approach
- Ferrari, ECCV06
- chamfer (with orientation planes)
- chamfer (no orientation planes)
Conclusions

1. learning shape models from images

2. matching them to new cluttered images

+ detect object boundaries while needing only BBs for training
+ effective also with hand-drawings as models
+ deals with extensive clutter, shape variability, and large scale changes

- can’t learn highly deformable classes (e.g. jellyfish)
- model quality drops with very high training clutter/fragmentation (giraffes)