Lecture 14
Visual recognition

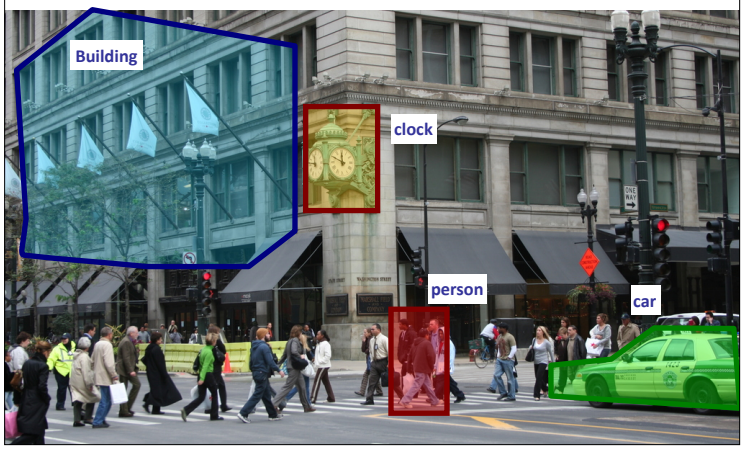
Announcements:
- Mid-term is released today at 12:15pm
- Due on Thursday at 11am

Lecture 14
Visual recognition

- 2D object detection
  - Template based approaches
  - Part-based approaches

Detection

Which object does this image contain? [where?]



In computer vision, detection is the problem of finding objects in images. Whereas image classification seeks to find a single label for an image, in object detection we're trying to find multiple labels and their position in space (often bounding boxes or silhouettes).

Detection

– Recognition task

– Search strategy: Sliding Windows Viola, Jones 2001,

- Simple
- Computational complexity (x,y, S, θ, N of classes)
  - BSW by Lampert et al 08
  - Also, Alexe, et al 10

A common detection strategy is to use a sliding window search. Here, we move a window across the image, checking whether some given object exists at that point in the image. Obviously this has high computation cost, as we have to check every different image position, and every scale and orientation, for every class we are interested in. Despite this complexity, sliding windows are used in many vision applications.

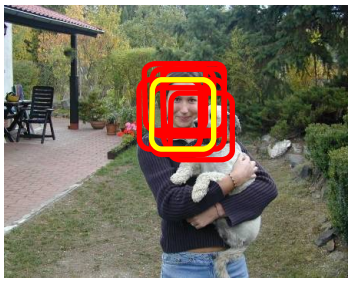
Detection

– Recognition task

– Search strategy: Sliding Windows Viola, Jones 2001,

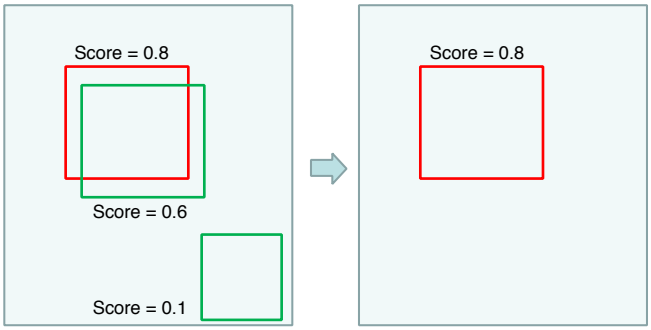
- Simple
- Computational complexity (x,y, S, θ, N of classes)
  - BSW by Lampert et al 08
  - Also, Alexe, et al 10
- Localization
  - Prone to false positive

Non max suppression:
Canny '86
….
Desai et al , 2009



Another problem is that nearby points in the search space (with similar positions or rotations, for example) will often produce strong positive responses when tested for an object class. In the image above, a face detector has determined that 7 bounding boxes contain faces, but obviously there is only one true face. We solve this localization problem with non-maximum suppression.

Non-max suppression



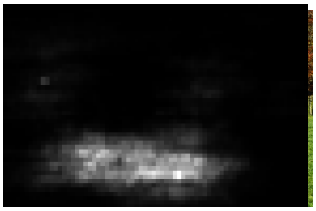
In non-maximum suppression, we suppress bounding boxes that significantly overlap with bounding boxes with a higher detection score. We also suppress bounding boxes with low detection scores. As a result, each actual object in the scene has a unique detection, located in the place with the strong detector response.

Detection

– Recognition task

– Search strategy : Probabilistic “heat maps”

- Fergus et al 03
- Leibe et al 04



An alternative strategy is to produce a heat map representing the probability that an object is present each point in the image. We can then use a mode-seeking algorithm such as Mean Shift to find peaks in this probability distribution.

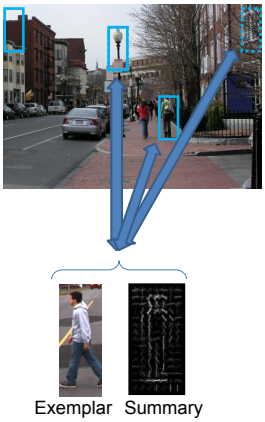
Lecture 14

Visual recognition

- 2D object detection
  - Template based approaches
  - Part-based approaches

Template-based detection

1. Slide a window in image
   – E.g., choose position, scale orientation
2. Compare it with a template
   – Compute similarity to an example object or to a summary representation
3. Compute a score for each comparison and compute non-max suppression to remove weak scores



Template based detection is arguably the most popular sliding window method. For every window position we compare the image within the window to a pre-trained template, resulting in a similarity score. Non-max suppression allows us to remove weaker matches to produce unique detections.

Dalal-Triggs pedestrian detector

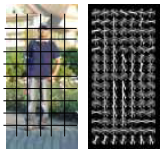
Represent an object as a collection of HoG templates

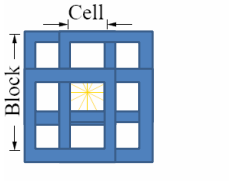
Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

A great example of this approach is the pedestrian detector developed by Dalal and Triggs in 2005. With 6000+ citations this is another of the most influential papers in computer vision! They developed a feature descriptor called Histograms of Oriented Gradients, or HOG, to match images to templates.

HoG = Histogram of Oriented Gradients

- Like SIFT, but…
  - Sampled on a dense, regular grid around the object
  - Gradients are contrast normalized in overlapping blocks

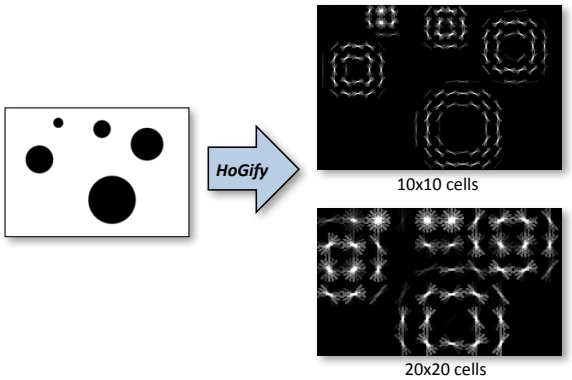




Courtesy of J Hayes

HoG is another descriptor similar to SIFT. The image is split into cells, and a histogram of edge gradients in each cell is computed. To make HOG more invariant to illumination changes, neighboring groups of cells (known as blocks) are normalized together.

Histogram of Oriented Gradients (HoG)



[Dalal and Triggs, CVPR 2005]

Courtesy of N Snavely

One of the nice features of the HOG descriptor is that it is easily visualized. Here you can see the transformation between image space and HOG space.

Dalal-Triggs pedestrian detector

1. Extract fixed-sized window at each position and scale
2. Compute HOG (histogram of gradient) features within each window
3. Score the window with a linear SVM classifier
4. Perform non-maxima suppression to remove overlapping detections with lower scores

Courtesy of J Hayes

The Dalal-Triggs is the prototypical template-detection algorithm. A HOG feature vector is computed for every window, which is passed to a linear SVM classifier to score the window. As usual, non-max suppression is used to eliminate overlapping detections.

Dalal-Triggs pedestrian detector

Results

Tricks of the trade

- Details in feature computation really matter
  - E.g., normalization in Dalal-Triggs significantly improves detection rate at fixed false positive rate
- Template size
  - Typical choice is size of smallest detectable object
- "Jittering" to create synthetic positive examples
  - Create slightly rotated, translated, scaled, mirrored versions as extra positive examples
- Bootstrapping to get hard negative examples
  1. Randomly sample negative examples
  2. Train detector
  3. Keep negative examples that score > T
  4. Repeat until all high-scoring negative examples fit in memory

Courtesy of J Hayes

Of course, there are some "engineering tricks" that get used to improve the performance of these methods. Normalizing the HOG template provides significantly more detections for a given false-positive rate. We also need to choose a template size appropriate to the object being detected.

One of the biggest challenges is training a good SVM detector. Given a limited amount of labeled training data, we can create synthetic examples to build a much larger training set. To do this we slightly transform the positive example images (changing scale, rotation, contrast, truncation, etc) and use these as extra training examples.

Because the space of possible images is so large, it is easy to find negative examples that are very easy to classify correctly (for example, patches of sky that could never be confused for a pedestrian). We want a very discriminative classifier, so we want to train on the hardest negative examples. We find the hardest examples by training an initial detector, and only keeping the negatives with a high score. We repeat this process until we have found enough "hard negatives" to train our final classifier on.

Limitation of template based approaches

They work

- *very well* for faces
- *fairly well* for cars and pedestrians
- *badly* for cats and dogs

- Why are some classes easier than others?

Courtesy of J Hayes

Template based methods have varying amount of success with different object classes. The difficulty of detecting a given class depends on how much objects within the class vary, in pose, shape and appearance.

Limitation of template based approaches

Strengths

- Works very well for non-deformable objects with canonical orientations: faces, cars, pedestrians
- Fast detection

Weaknesses

- Not so well for highly deformable objects or "stuff"
- Not robust to occlusion
- Requires lots of training data if view points need to be encoded

Courtesy of J Hayes

Template models work well when objects are non-deformable, have canonical orientations, and are reasonably similar to one another (within a class). They fail when trying to detect "stuff" (a technical term for amorphous objects such as the sky, sidewalks and roads), and when objects are occluded. If we want to detect objects from different viewpoints we need a huge amount of training data to capture every view.

Classic template-based Detectors

- Sung-Poggio (1994, 1998) : ~2000 citations
  - Basic idea of statistical template detection, bootstrapping to get "face-like" negative examples, multiple whole-face prototypes (in 1994)
- Rowley-Baluja-Kanade (1996-1998) : ~3600
  - "Parts" at fixed position, non-maxima suppression, simple cascade, rotation, pretty good accuracy, fast
- Schneiderman-Kanade (1998-2000,2004) : ~1700
  - Careful feature engineering, excellent results, cascade
- Viola-Jones (2001, 2004) : ~11,000
  - Haar-like features, Adaboost as feature selection, hyper-cascade, very fast, easy to implement
- Dalal-Triggs (2005) : ~6500
  - Careful feature engineering, excellent results, HOG feature, online code

Courtesy of J Hayes

The following are some very famous template-based detectors.

Lecture 14
Visual recognition

- 2D object detection
  - Template based approaches
  - Part-based approaches

Part Based Representation

- Object as set of parts
- Model:
  - Relative locations between parts
  - Appearance of part

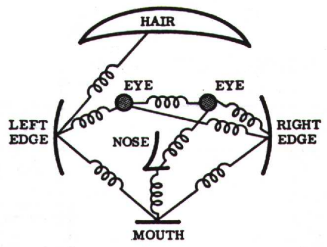
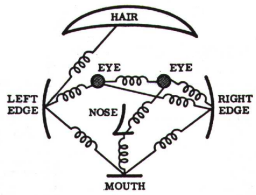


Figure from [Fischler & Elschlager 73]

An alternative approach to a fixed template is to represent an object as a collection of parts, encoding the appearance of parts and their locations with respect to one another.

History of Parts and Structure approaches

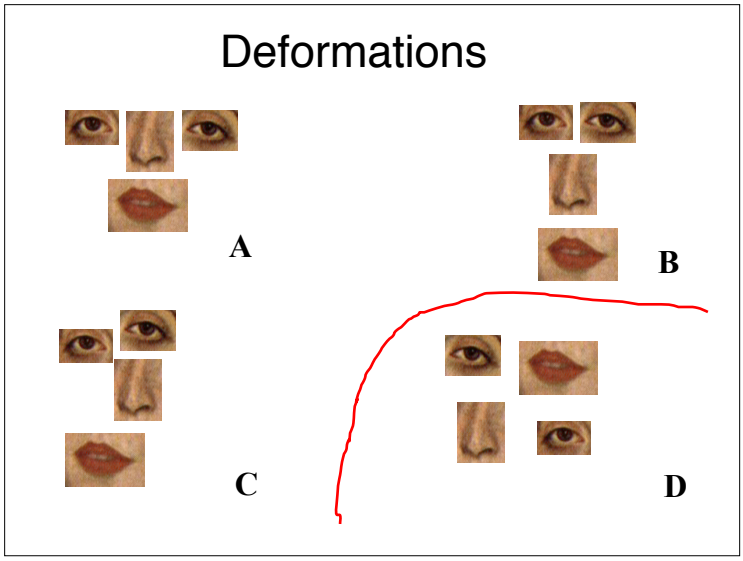
- Fischler & Elschlager 1973



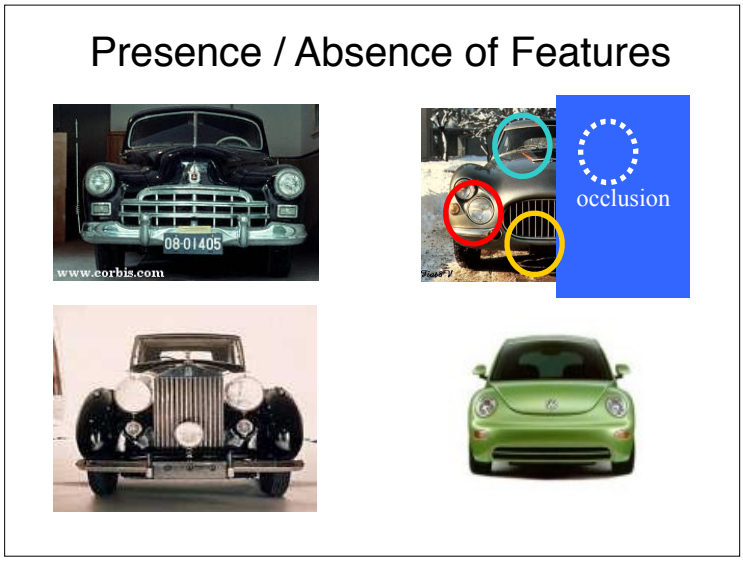
- Yuille '91
- Brunelli & Poggio '93
- Lades, v.d. Malsburg et al. '93
- Cootes, Lanitis, Taylor et al. '95
- Amit & Geman '95, '99
- Perona et al. '95, '96, '98, '00, '03, '04, '05
- Ullman et al. 02
- Felzenszwalb & Huttenlocher '00, '04
- Crandall & Huttenlocher '05, '06
- Leibe & Schiele '03, '04

- Many papers since 2000

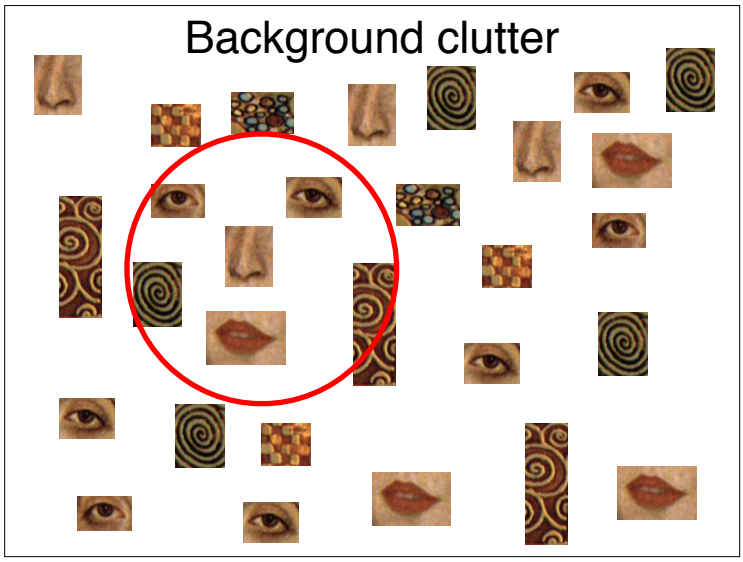
Some of the earliest attempts at building object detectors used a part-based approach, but these had only mixed success. More recently these approaches have become very successful, most notably the Deformable Parts Model of Felzenszwalb et al (PAMI 2010).



Part-based models can capture more variation in appearance because they allow parts to deform relative to one another. This may not be so useful for faces (when we know pretty well where every part should be), but it is very beneficial for more varied classes such as cars.



If our method is robust to missing parts we can naturally handle occlusion.



We can also mitigate the effect of background clutter because we are only looking for specific parts, not whole objects.

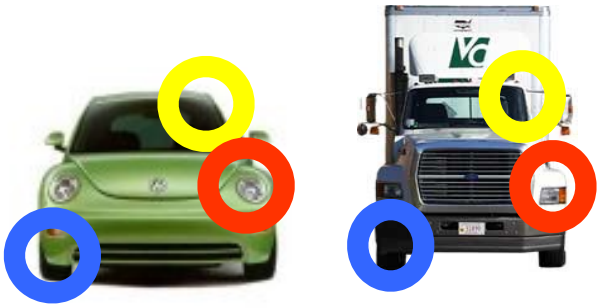
Sparse representation

Computationally tractable (10^5 pixels → 10^1 -- 10^2 parts)
But throw away potentially useful image information

By reducing an object image into a small collection of parts we can create a sparse representation, although we can obviously lose information in this transformation process if we are not careful.

Discriminative

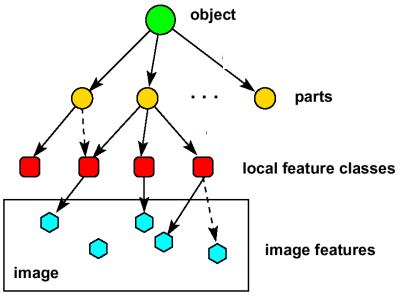
Parts need to be distinctive to separate from other classes



We require the parts we choose to be discriminative, that is, to successfully separate object classes. For example, a part for the green hood of the car would not be discriminative – a visually similar part could occur on a wall or a patch of grass.

Hierarchical representations

- Pixels → Pixel groupings → Parts → Object

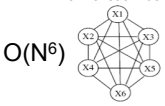


Images from [Amit98,Bouchard05]

We also need to choose how we are going to group the parts. A common and intuitive method is to use a hierarchical representation, where a root node represents the object as a whole, and further down the tree represents the finer details of the object.

Different connectivity structures

Fergus et al. '03
Fei-Fei et al. '03



O(N^6)

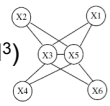
a) Constellation [13]

Crandall et al. '05
Leibe 05; Felzenszwalb 09

O(N^2)

b) Star shape [9, 14]

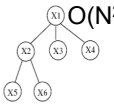
Crandall et al. '05



O(N^3)

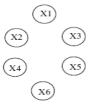
c) k-fan ($k = 2$) [9]

Felzenszwalb &
Huttenlocher '00

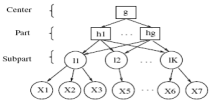


O(N^2)

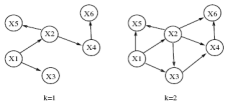
d) Tree [12]



e) Bag of features [10, 21]
Csurka '04
Vasconcelos '00



f) Hierarchy [4]
Bouchard & Triggs '05



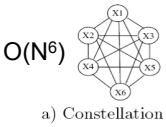
g) Sparse flexible model
Carneiro & Lowe '06

from Sparse Flexible Models of Local Features
Gustavo Carneiro and David Lowe, ECCV 2006

The connections between the parts also determine the complexity of the relationship we need specify. As this is often learned from data, a more complex model requires more training data.

Different connectivity structures

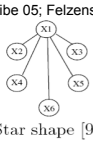
Fergus et al. '03
Fei-Fei et al. '03



O(N^6)

a) Constellation [13]

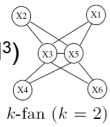
Crandall et al. '05
Leibe 05; Felzenszwalb 09



O(N^2)

b) Star shape [9, 14]

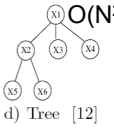
Crandall et al. '05



O(N^3)

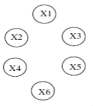
c) k-fan ($k = 2$) [9]

Felzenszwalb &
Huttenlocher '00

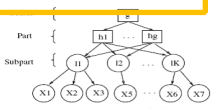


O(N^2)

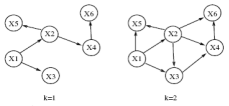
d) Tree [12]



e) Bag of features [10, 21]
Csurka '04
Vasconcelos '00



f) Hierarchy [4]
Bouchard & Triggs '05

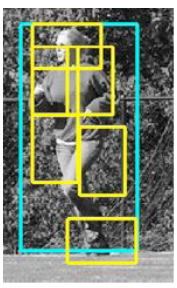


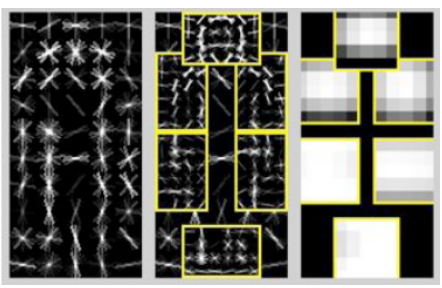
g) Sparse flexible model
Carneiro & Lowe '06

from Sparse Flexible Models of Local Features
Gustavo Carneiro and David Lowe, ECCV 2006

The star model (a root node with many leaf node children) is a very popular model, use in DPM (Felzenszwalb '09)

Star models by Latent SVM

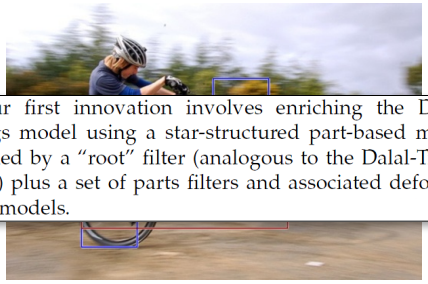




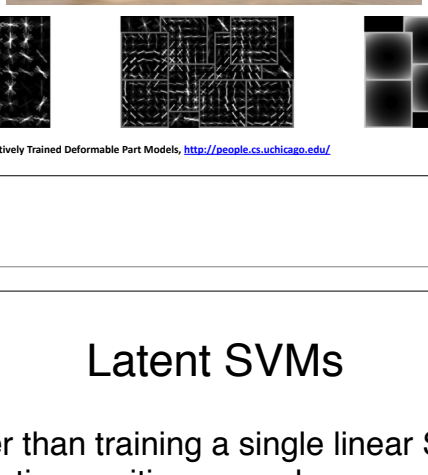
Felzenszwalb, McAllester, Ramanan, 08
• Source code:

DPM learns a large root filter (center-left), which captures the general appearance of the object. It also learns a collection of part filters (center-right), which are higher resolution are capture fine details of the object. The spatial relationship between these parts and the root is also learned (far right), which determines how much deformation is permissible, and what cost should be paid (in terms of detection score) for this deformation.

Deformable Part Models (DPM)



Our first innovation involves enriching the Dalal-Triggs model using a star-structured part-based model defined by a "root" filter (analogous to the Dalal-Triggs filter) plus a set of parts filters and associated deformation models.



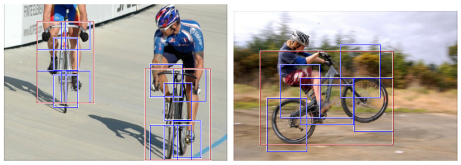
Felzenszwalb, et al., **Discriminatively Trained Deformable Part Models,** http://people.cs.uchicago.edu/~pff/latent/

Latent SVMs

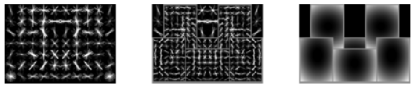
- Rather than training a single linear SVM separating positive examples…
- … cluster positive examples into "components" and train a classifier for each (using all negative examples)

DPM is trained using a variant of SVMs called Latent SVMs. Refer to the paper for details of this approach. One of its advantages is that it allows objects to be represented as mixtures of templates, known as components. These are found by clustering the positive examples, and often correspond to different viewpoints.

Two-component bicycle model



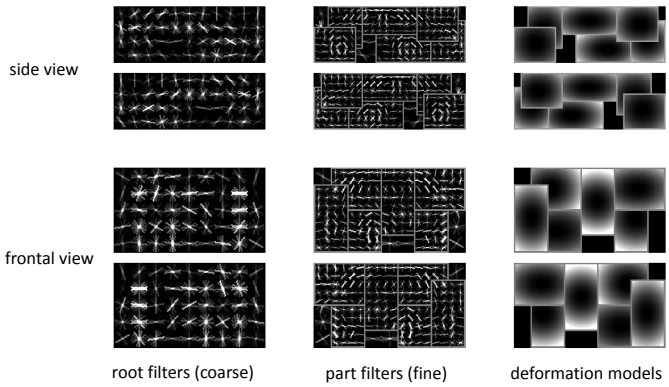
"side" component



"frontal" component

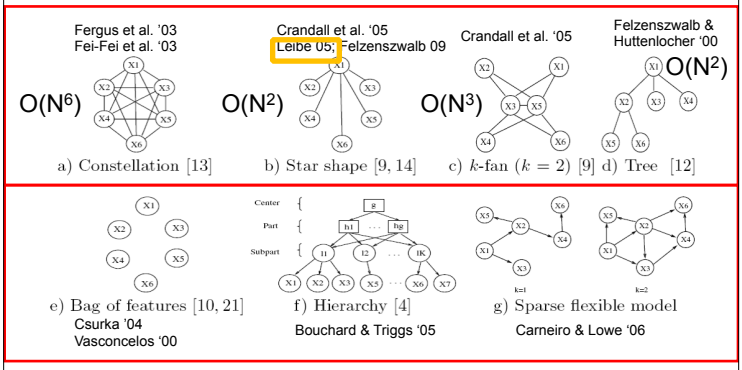
For example, here we see DPM templates for a bicycle, including front and side components.

Six-component car model



Cars are represented with 6 different components.

Different connectivity structures



from Sparse Flexible Models of Local Features
Gustavo Carneiro and David Lowe, ECCV 2006

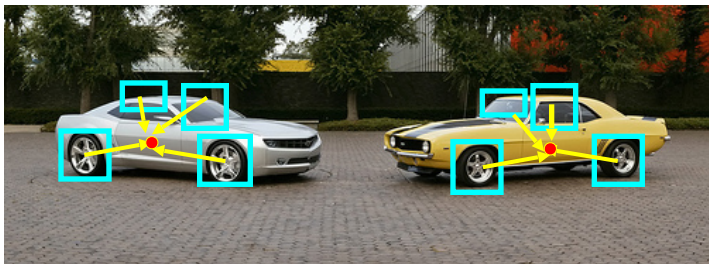
We'll now discuss the method from Leibe et al, who also use star shaped models.

Implicit shape models by generalized Hough voting

B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, ECCV Workshop on Statistical Learning in Computer Vision 2004

Their method uses an approach called Hough voting, an early technique for detecting shapes in images.

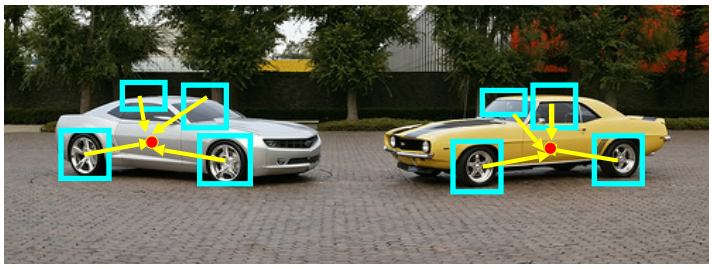
Object representation:
Constellation of parts w.r.t object centroid



B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, ECCV Workshop on Statistical Learning in Computer Vision 2004

Similar to DPM, objects are represented as collections on parts around the object centroid.

Object representation:
How to capture constellation of parts?
Using Hough Voting



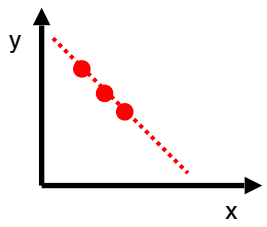
B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, ECCV Workshop on Statistical Learning in Computer Vision 2004

Each part "votes" on a position of the object as a whole, using a method we will now describe.

Hough transform

P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,* Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

Given a set of points, find the curve or line that explains the data points best



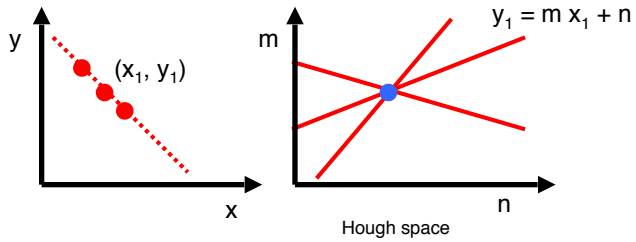
The Hough (pronounced "Huff") transform was developed in 1959 to analyze images from a bubble chamber.
It its simplest incarnation, the Hough transform is used to find lines in an image.

- A line is represented as y=mx+n. Every line in the image corresponds to a point in the parameter space (n,m)
- Every point in the image domain corresponds to a line in the parameter space (why ? Fix (x,y), (m,n) can change on the line . In other words, for all the lines passing through (x,y) in the image space, their parameters form a line in the parameter space)
- Points along a line in the space correspond to lines passing through the same point in the parameter space, because they share m and n values.

Hough transform

P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,* Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

Given a set of points, find the curve or line that explains the data points best



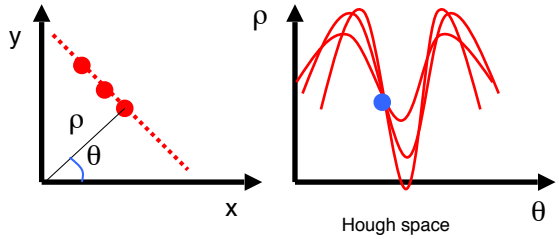
$y = m\,x + n$

Here we can see the mapping of points in image (x,y) space, to lines in Hough (n,m) space. The intersection of these lines describe the parameter values (n,m) that fit a line through all 3 points in image space.

Hough transform

P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,* Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

- Use a polar representation for the parameter space

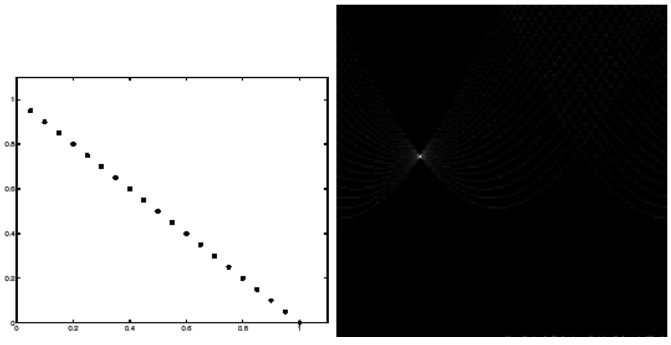


$$x\cos\theta + y\sin\theta = \rho \qquad \text{[Eq. 1]}$$

Parameterizing a line as y = mx+n is bad because we cannot represent a vertical line x = c. Instead, we use the parameterization in Eq. 1, with parameters (theta, p). P is the perpendicular distance of the line from the origin, and theta is the angle shown in the figure above.

In this parameterization, every point in the image maps to a sine wave in Hough space.

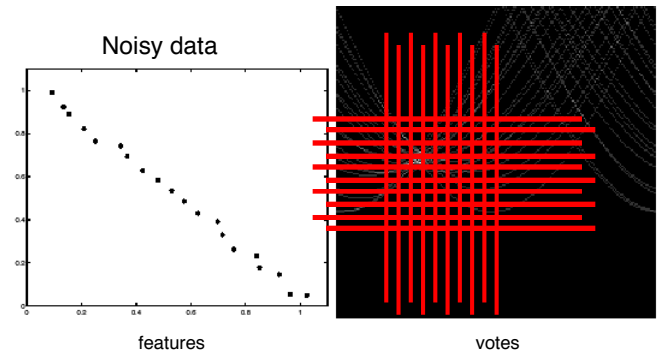
Hough transform - experiments



features votes

Here we can see feature points in an image being mapped into Hough space (brighter areas indicate more votes). We can clearly see the intersection of the sine waves at the correct parameter values.

Hough transform - experiments

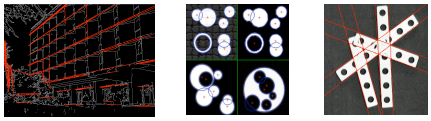


IDEA: introduce a grid a count intersection points in each cell
Issue: Grid size needs to be adjusted…

The noisier the data, the larger the grid cells need to be.

Generalized Hough Transform

- Parts in query image vote for a learnt model
- Significant aggregations of votes correspond to models
- Complexity : # parts * # votes
  - Significantly lower than brute force search (e.g., sliding window detectors)
- Popular for detecting parameterized shapes
  - Hough'59, Duda&Hart'72, Ballard'81,…



Slide Modified from *S. Maji*

In principle, the Hough transform can be used to find any parameterized shape. Its complexity is lower than sliding window search.

Generalized Hough Transform

- GOAL: detect arbitrary shapes defined by boundary points and a reference point

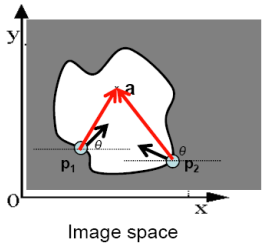


Image space

Learning a model:

At each boundary point, compute displacement vector: $\mathbf{r} = \mathbf{a} - \mathbf{p}_i$. [Eq. 2]

For a given model shape: store these vectors in a table indexed by gradient orientation θ.

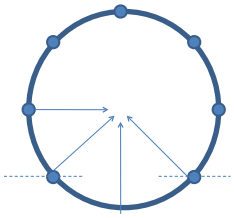
[Dana H. Ballard, Generalizing the Hough Transform to Detect Arbitrary Shapes, 1980]

Credit slide: *C. Grauman*

The generalized Hough transform allows us to find arbitrary shapes in images from their boundary. We describe a shape by a collection of boundary points, computing r (Eq. 2), the vector from the boundary to the reference point. These vectors are stored in a table indexed by the image gradient orientation at the boundary to form the shape model.

Example

Circle model



θ	rx	ry
0	*1*	*0*
45	*0.7*	*0.7*
90	*0*	*1*
135	*-0.7*	*0.7*
…		
270	*0.7*	*-0.7*

Here we see a model of a circle. Gradient orientation is measured from the x axis, so the first point in the table describes the furthest left point in the circle.

Generalized Hough Transform

Detecting the *model shape in a new image:*

- For each edge point
  - Index into table with its gradient orientation ϑ
  - Use retrieved r vectors to vote for position of reference point
- Peak in this Hough space is reference point with most supporting edges

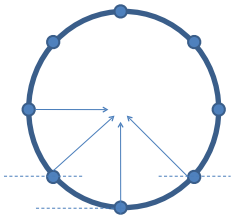
Assuming translation is the only transformation here, i.e., orientation and scale are fixed.

Credit slide: C. Grauman

During detection, we again consider only edge points. We index into the shape table by the edge orientation, and recover r vectors that share this gradient orientation. Each vector becomes a vote for the shape reference point in the Hough space, and the peak corresponds to the actual position of the object in image space. Note that this method cannot typically handle changes in orientation or scale.

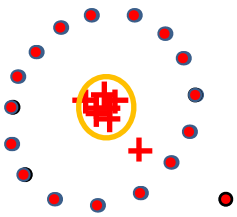
Example

Circle model



θ	rx	ry
0	*1*	*0*
45	*0.7*	*0.7*
90	*0*	*1*
135	*-0.7*	*0.7*
…		
270	*0.7*	*-0.7*

Query



P1 → θ = 0 → R = [rx,ry] = [1,0] → C1 = P1 + R

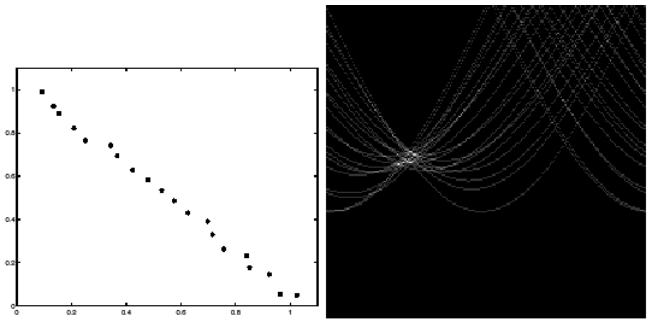
P2 → θ = 45 → R = [rx,ry] = [.7,.7] → C2 = P2 + R

Pk → θ = -180 → R = [rx,ry] = [-1, 0] → Ck = Pk + R

⋮

With a large enough bin size in Hough space, we can detect circles the are slightly deformed.

Conceptually similar to

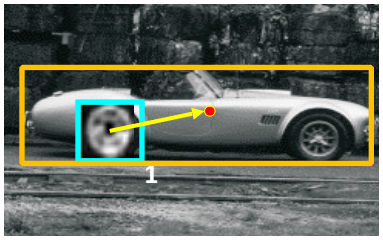


The concept is similar to the line detection algorithm we've already described, but now we don't need a mathematical description of the shape we are searching for.

Implicit shape models

- Instead of indexing displacements by gradient orientation, index by "visual codeword"

→ Visual codebook is used to index votes for object position [center] and scale



CW	rx	ry
1	*0.9*	*0.1*

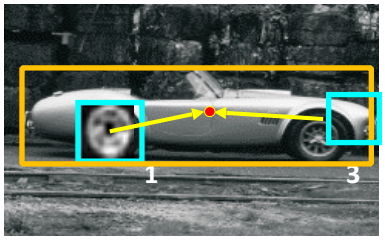
B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, ECCV Workshop on Statistical Learning in Computer Vision 2004

Implicit shape models (Leibe et al) use this same principle of Hough voting. Instead of indexing each part by gradient orientation, they use "visual codewords", which capture the appearance of a part of the object. Each codeword votes for a position of the object reference point.

Implicit shape models

- Instead of indexing displacements by gradient orientation, index by "visual codeword"

→ Visual codebook is used to index votes for object position [center] and scale



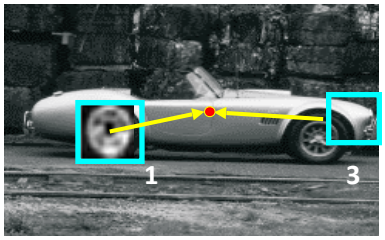
CW	rx	ry
1	*0.9*	*.1*
3	*?*	*?*

B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, ECCV Workshop on Statistical Learning in Computer Vision 2004

Implicit shape models

- Instead of indexing displacements by gradient orientation, index by “visual codeword”

→ Visual codebook is used to index votes for object position [center] and scale

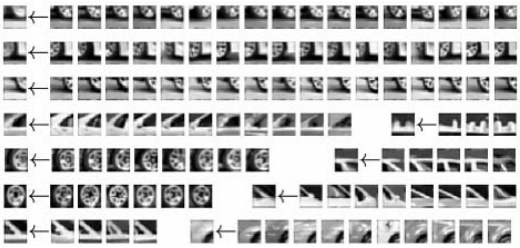


CW	rx	ry
1	*0.9*	*.1*
3	*-1*	*0*
…	*…*	*…*
N	*0.7*	*-0.7*

B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, ECCV Workshop on Statistical Learning in Computer Vision 2004

Implicit shape models: Training

1. Build codebook of patches around extracted interest points using clustering

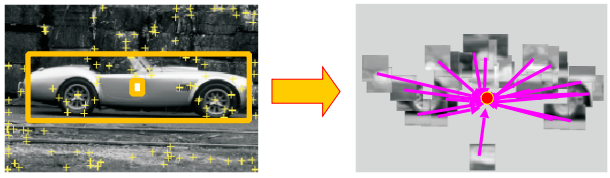


Credit slide: S. Lazebnik

To train the codebook, we cluster patches extracted from around detected interest points.

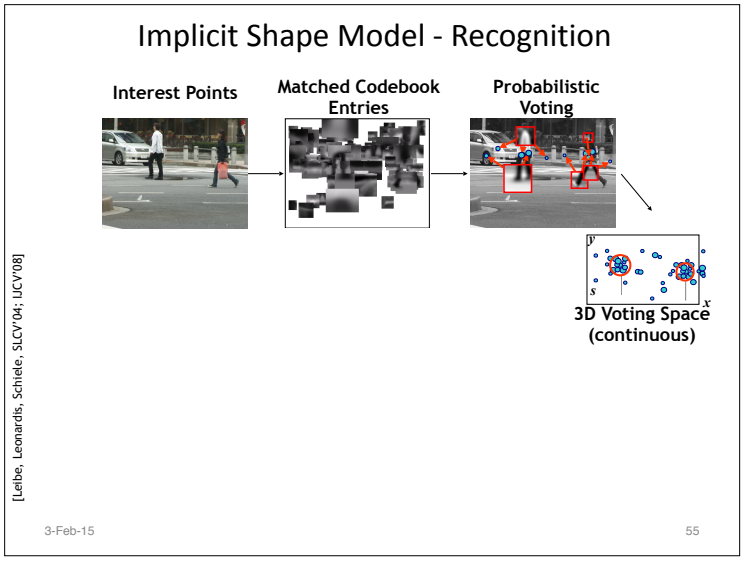
Implicit shape models: Training

1. Build codebook of patches around extracted interest points using clustering
2. Map the patch around each interest point to closest codebook entry
3. For each codebook entry, store all positions relative to object center [center is given] and scale [bounding box is given]

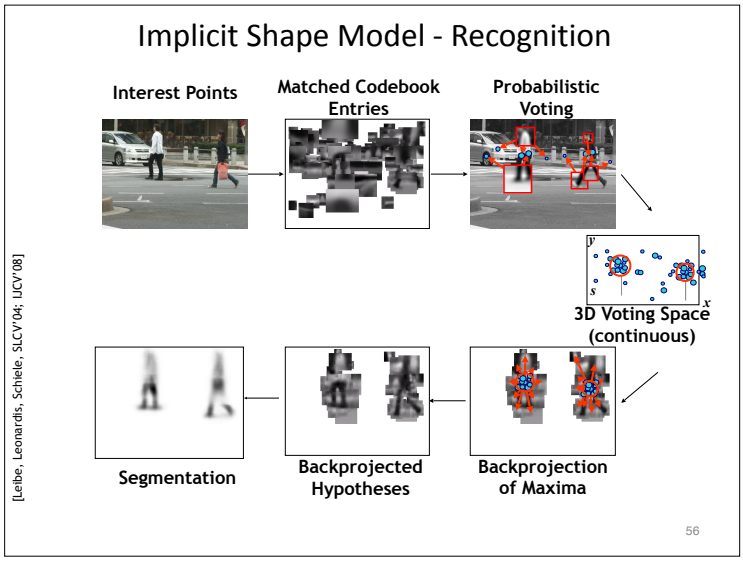


Credit slide: S. Lazebnik

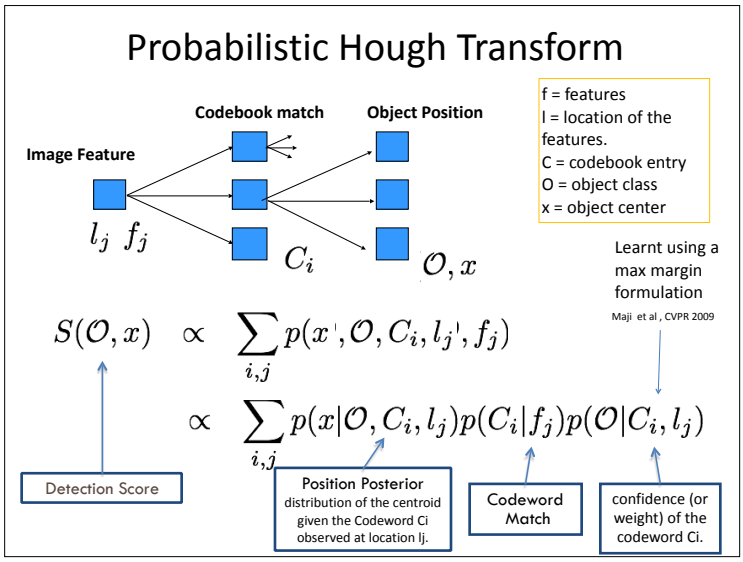
Once we have found the codebook, we match every patch to its nearest neighbor in the codebook. The relative position of the object center from the patch, as well as the scale of the object, are added to the table indexed by the nearest codebook entry. We assume object center and bounding box is given during training.



During testing, interest points are found and matched to codebook entries, each of which votes for a collection of parameter values in the 3D voting space.



Given the maxima in Hough space, we back-project the model into the image, using the parameters described by the maxima. This gives us hypothesis parts which we can further process into bounding boxes or sementations.



Maji et al (CVPR '09) describe the Hough transform in a probabilistic manner. The detection score can be approximated as the sum of independent votes p(O; x; fj ; sj ; lj) from each feature f_j observed at a location l_j.

Let C be the learned codebook,

let f denote the features

l the location of the features.

P(xlO,Ci,l_j) is the distribution of the centroid given the Codeword Ci observed at location lj.

Each feature is matched to a codebook as given by p(Cilfj).

The last term p(OlCi,lj) is the confidence (or weight) of the codeword Ci.

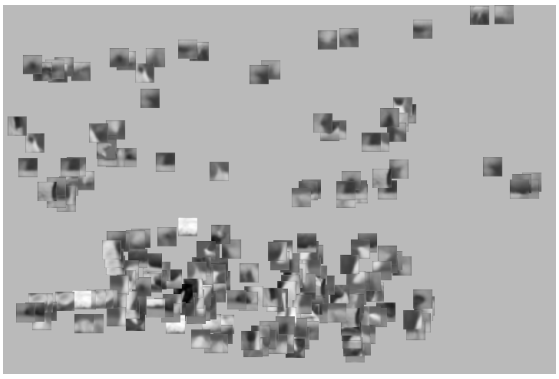
Example: Results on Cows

Original image

Example: Results on Cows

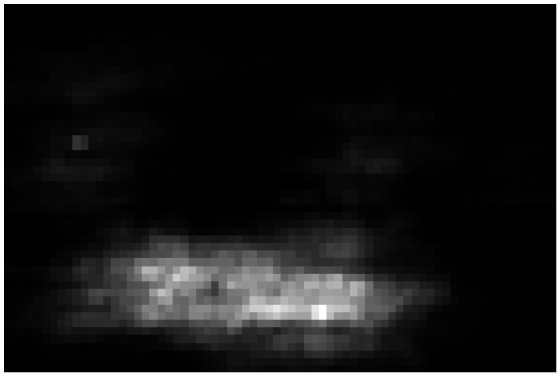
Interest points

Example: Results on Cows



Matched patches

Example: Results on Cows



Prob. Votes

Example: Results on Cows

1st hypothesis

Example: Results on Cows

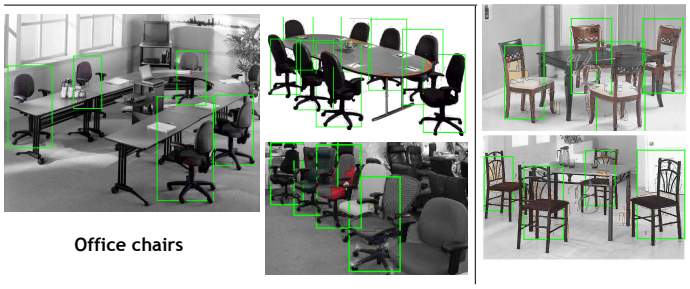
2nd hypothesis

Example: Results on Cows

3rd hypothesis

Example Results: Chairs

Dining room chairs



Office chairs

You Can Try It At Home…

- Linux binaries available
  - Including datasets & several pre-trained detectors
  - http://www.vision.ee.ethz.ch/bleibe/code

Conclusions

- Pros:
  - Works well for many different object categories
    - Both rigid and articulated objects
  - Flexible geometric model
    - Can recombine parts seen on different training examples
  - Learning from relatively few (50-100) training examples
  - Optimized for detection, good localization properties
- Cons:
  - Needs supervised training data
    - Object bounding boxes for detection
    - Segmentations for top-down segmentation
  - No discriminative learning

Influential Works in Detection

- Sung-Poggio (1994, 1998) : ~2000 citations
  - Basic idea of statistical template detection, bootstrapping to get "face-like" negative examples, multiple whole-face prototypes (in 1994)
- Rowley-Baluja-Kanade (1996-1998) : ~3600
  - "Parts" at fixed position, non-maxima suppression, simple cascade, rotation, pretty good accuracy, fast
- Schneiderman-Kanade (1998-2000,2004) : ~1700
  - Careful feature engineering, excellent results, cascade
- Viola-Jones (2001, 2004) : ~11,000
  - Haar-like features, Adaboost as feature selection, hyper-cascade, very fast, easy to implement
- Dalal-Triggs (2005) : ~6500
  - Careful feature engineering, excellent results, HOG feature, online code
- Felzenszwalb-Huttenlocher (2000): ~2100
  - Efficient way to solve part-based detectors
- Weber et al. (2000)
  - Part-based model learnt in a unsupervised fashion; generative
- Felzenszwalb-McAllester-Ramanan (2008): ~1300
  - Excellent template/parts-based blend
- Leibe et al. (2005)
  - Generative approach to detection using hough voting

Courtesy of J Hayes

These papers on object detection are essential reading for those interested in the problem.

Next lecture

- 3D Object Detection