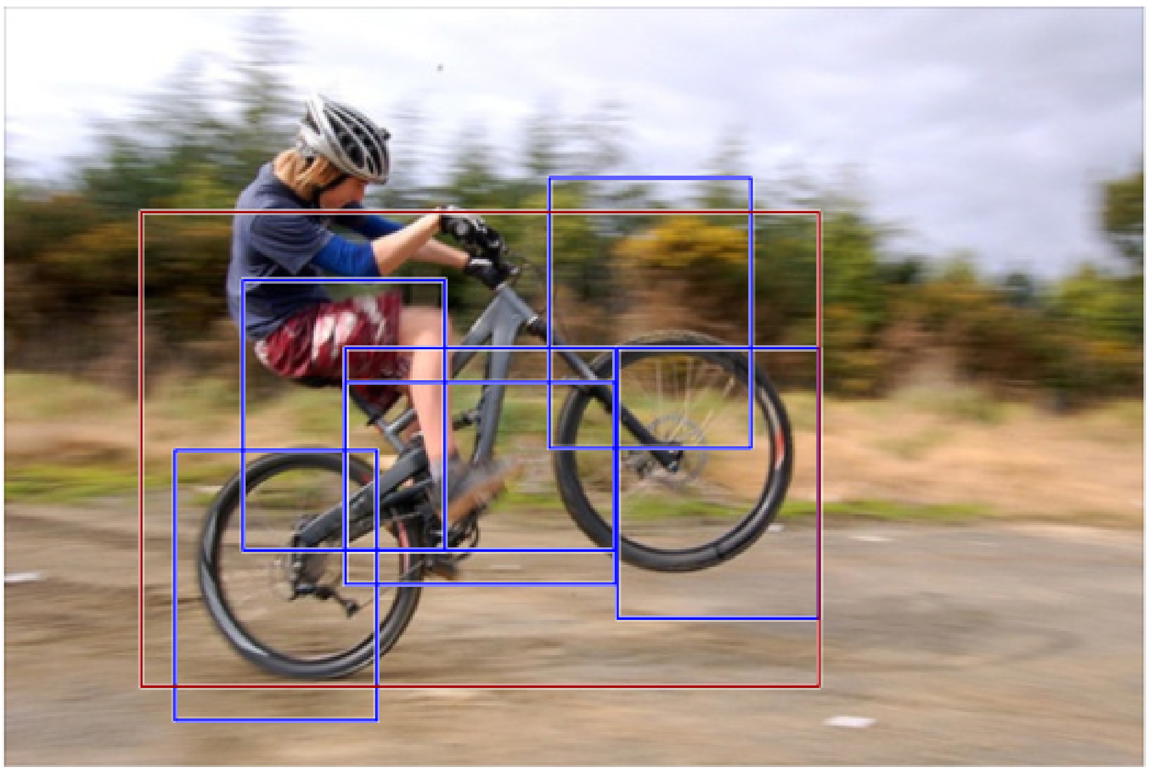
Object detection with heuristic coarse-to-fine search

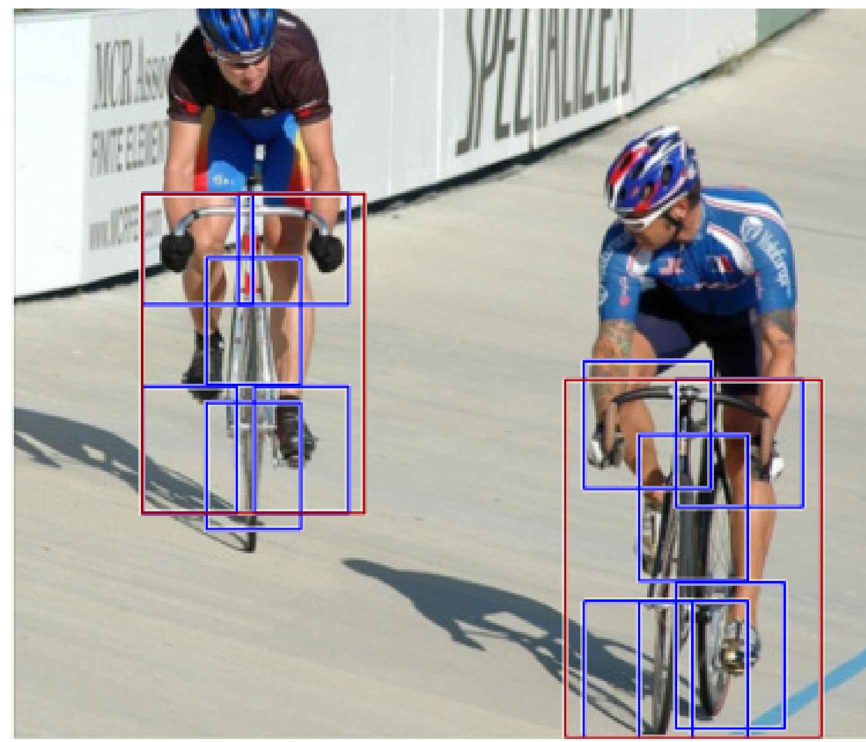
Ross B. Girshick
Department of Computer Science
University of Chicago

Joint work with Pedro Felzenszwalb (UofC) and
David McAllester (TTI-C)

What's the problem?

Localize instances of a generic object category in a real-world image.





Example: find all bicycles in these images.
(from the PASCAL 2007 dataset)

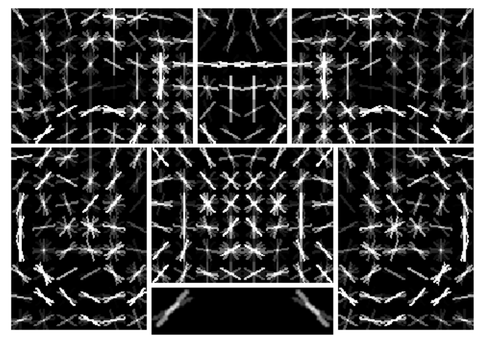
Our family of models

1. Multiscale deformable part models
2. Mixtures of multiscale star models
3. Visual grammars

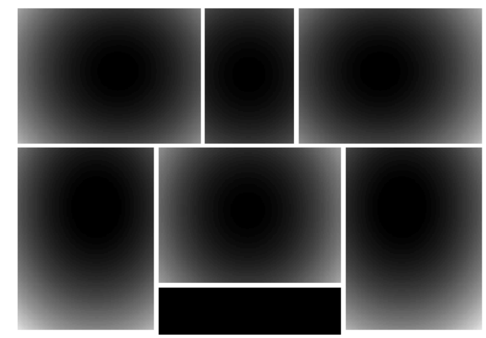
$$1 \subset 2 \subset 3$$

1. Multiscale deformable part models ("star models")

root filter

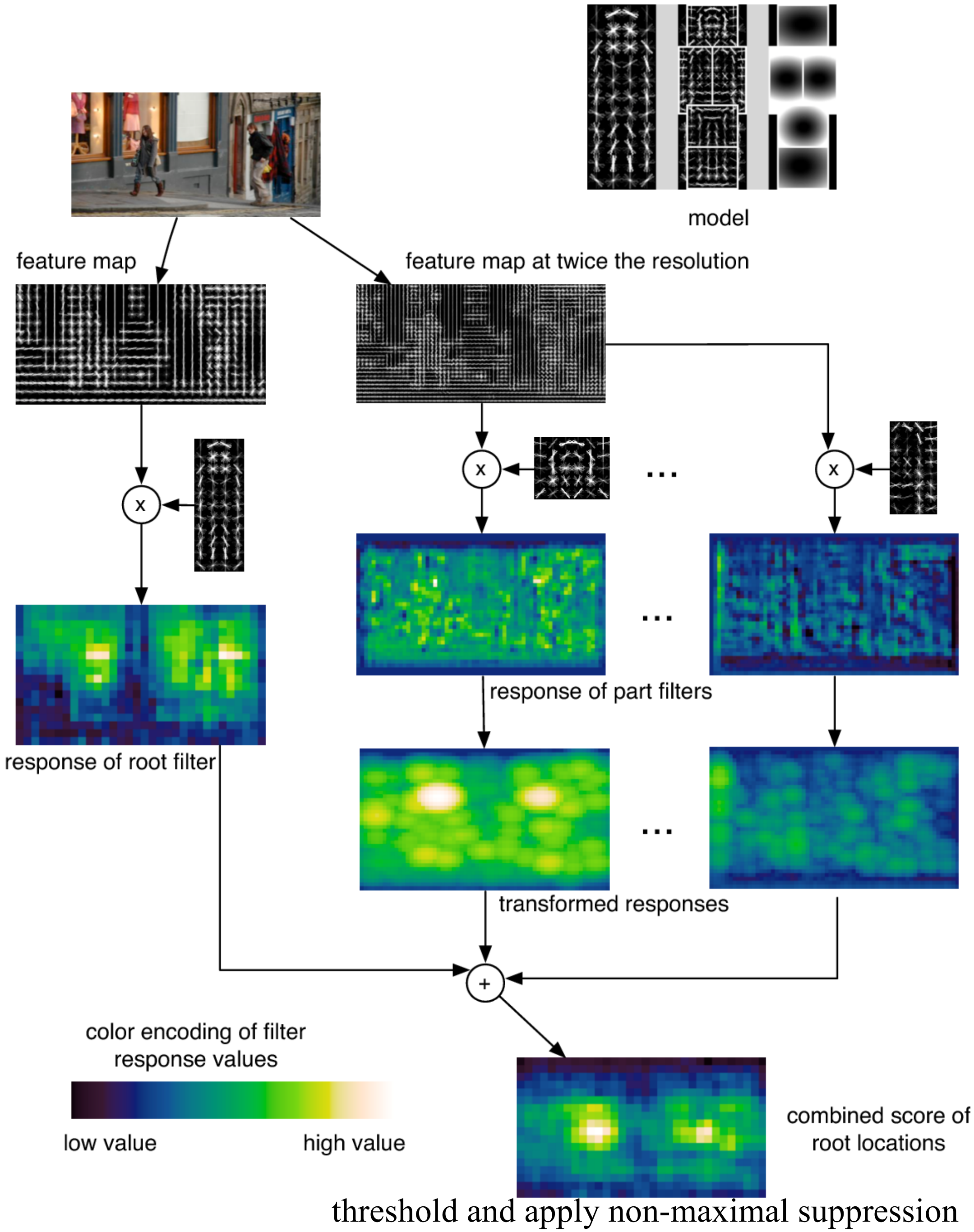


part filters



deformation costs

Detection with multiscale star models



Find local maxima of:

$$S_M(L) = \sum_{i=1}^{n} m_i(l_i) - \sum_{i=2}^{n} d_i(l_1, l_i),$$

above a threshold T.

$L = (l_1, \ldots, l_n) =$ object hypothesis.

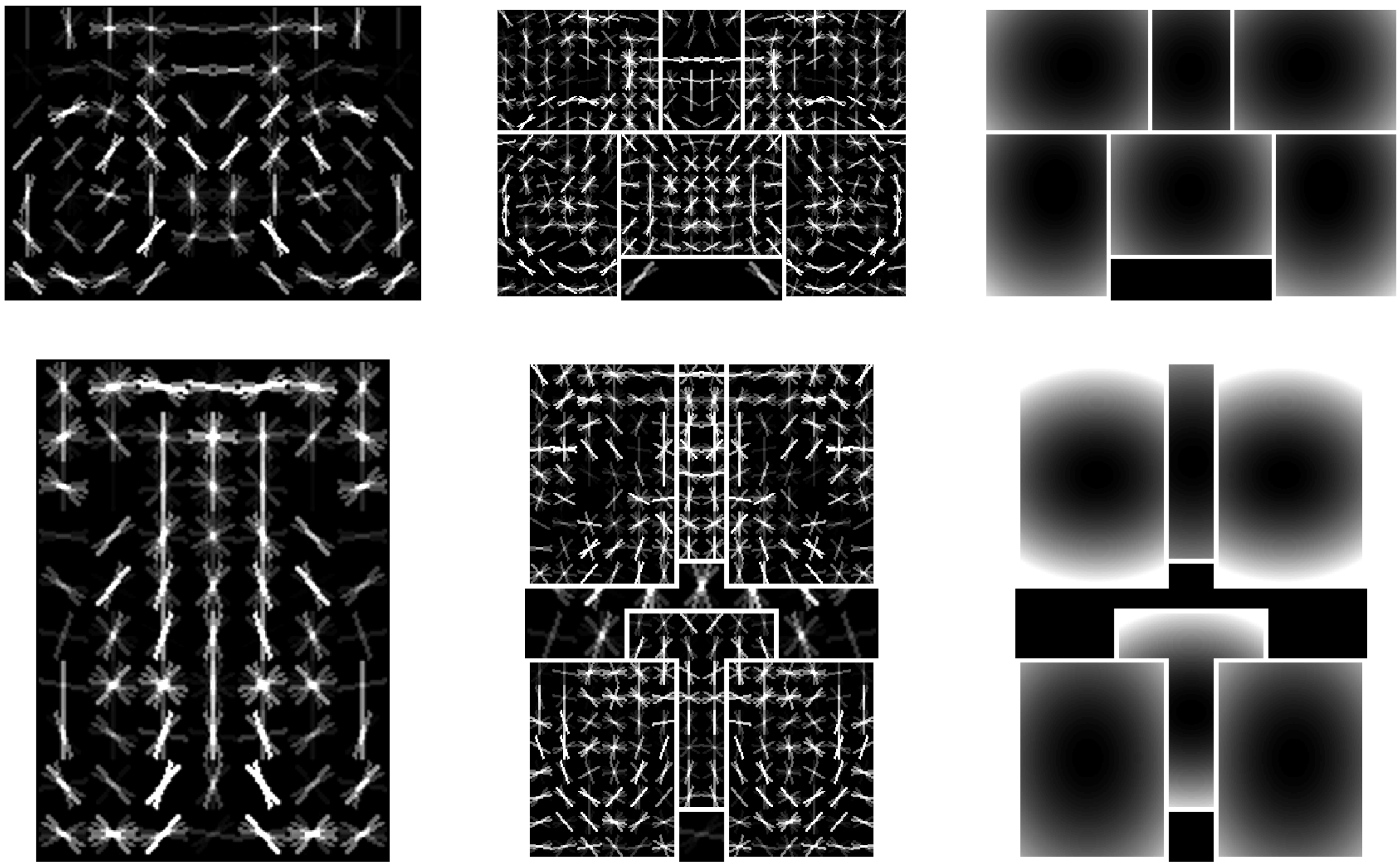
$l_i =$ filter locations in feature pyramid.

Use dynamic programming and distance transforms.

- linear in # of filters
- the constant factor is large, e.g., 640x480 image → ~250M fp mults

[P. Felzenszwalb, D. Huttenlocher 2000]
[P. Felzenszwalb, D. McAllester, and D. Ramanan 2008]

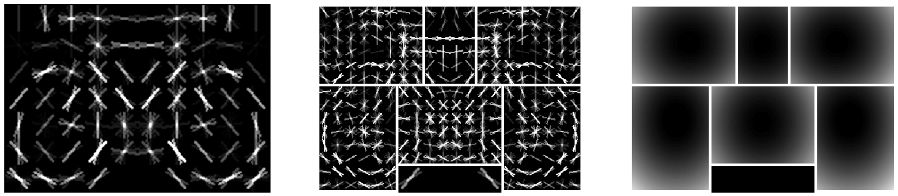
2. Mixtures of multiscale star models



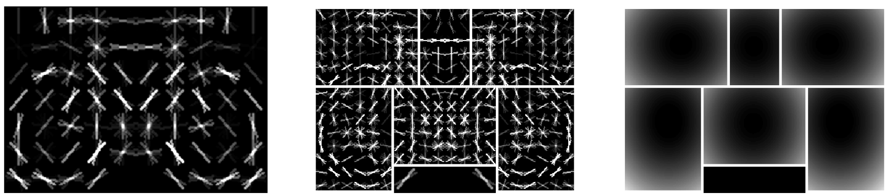
Detection: apply the same procedure to each component *independently*, and then take the max over component scores.

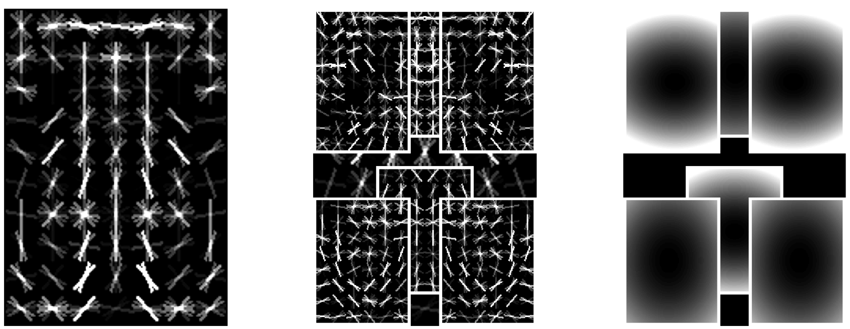
[P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. 2009]

3. Visual grammars



$$B \rightarrow R$$
$$R \rightarrow P_1 P_2 P_3 P_6 P_5 P_6$$





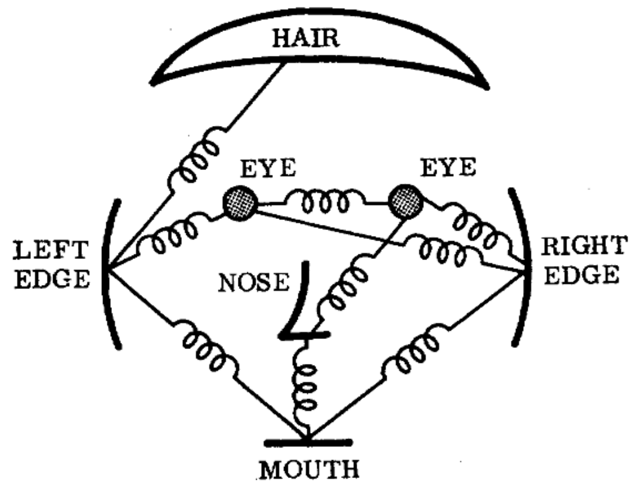
$$B \rightarrow R_1 | R_2$$
$$R_1 \rightarrow P_1 P_2 P_3 P_6 P_5 P_6$$
$$R_2 \rightarrow P_7 P_8 P_9 P_{10} P_{11} P_{12}$$

$\ldots$

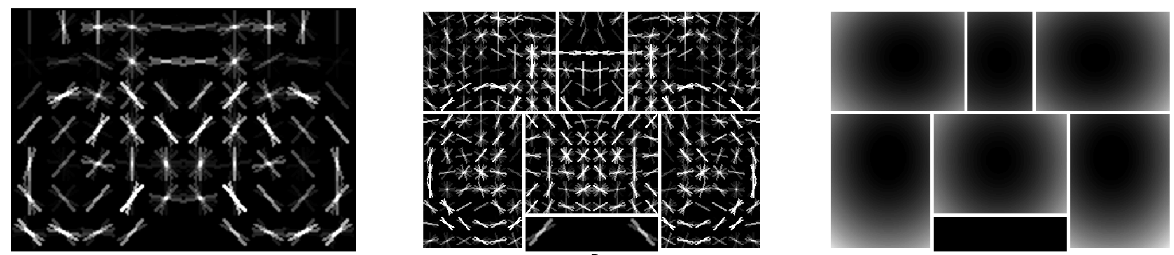
$\ldots$

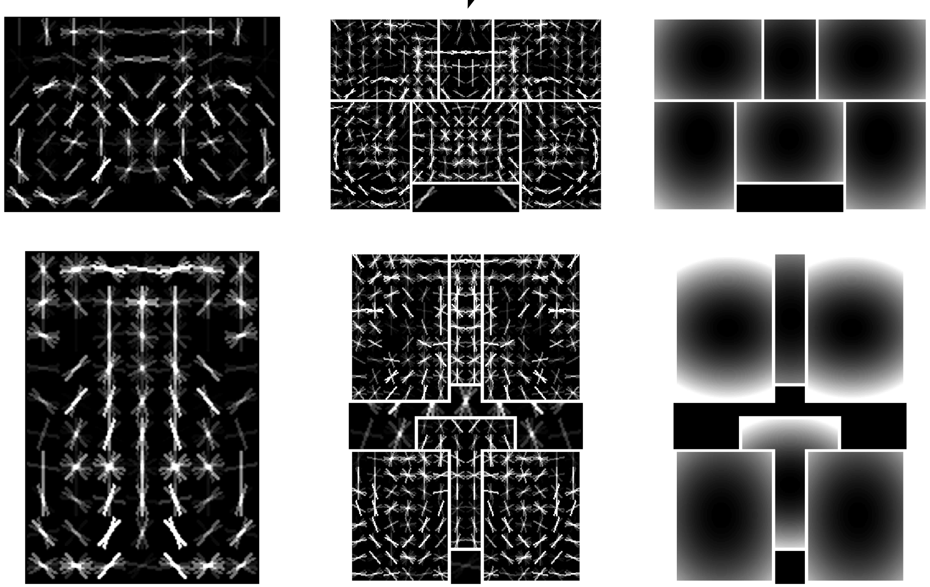
$$B \rightarrow R_1 | R_2 | \ldots | R_k$$
$$R_1 \rightarrow P_1 P_2 \ldots P_{n_1}$$
$$\vdots$$
$$R_k \rightarrow P_1 P_5 P_7$$
$$P_1 \rightarrow MP_1 | MP_2$$
$$MP_1 \rightarrow SP_1 SP_2$$
$$MP_2 \rightarrow SP_1 SP_3$$
$$\vdots$$

The big picture (evolution from the pictorial structure model)



[M.A. Fischler and R.A. Elschlager.
"The representation and matching of pictorial structures."]





$$B \rightarrow R_1 | R_2 | \ldots | R_k$$
$$R_1 \rightarrow P_1 P_2 \ldots P_{n_1}$$
$$\vdots$$
$$R_k \rightarrow P_1 P_5 P_7$$
$$P_1 \rightarrow MP_1 | MP_2$$
$$MP_1 \rightarrow SP_1 SP_2$$
$$MP_2 \rightarrow SP_1 SP_3$$
$$\vdots$$

Challenges on the path to rich grammar models

1. Model initialization and training
   - How do we initialize parts, subparts, mixtures of parts, shared part dictionaries?
   - How do we train rich models from bounding boxes? (latent SVM)
2. Computational efficiency
   - Rich model → expensive detection/inference.
   - Inference must be fast enough to make rich grammar models usable in practice.
   - What if we have models for 10,000 object classes?

Challenges on the path to rich grammar models

1. Model initialization and training
   - How do we initialize parts, subparts, mixtures of parts, shared part dictionaries?
   - How do we train rich models from bounding boxes? (latent SVM)

Our focus today:

2. Computational efficiency
   - Rich model → expensive detection/inference.
   - Inference must be fast enough to make rich grammar models usable in practice.
   - What if we have models for 10,000 object classes?

The problem: we need a real parsing algorithm, but still limited to a small number of filters.

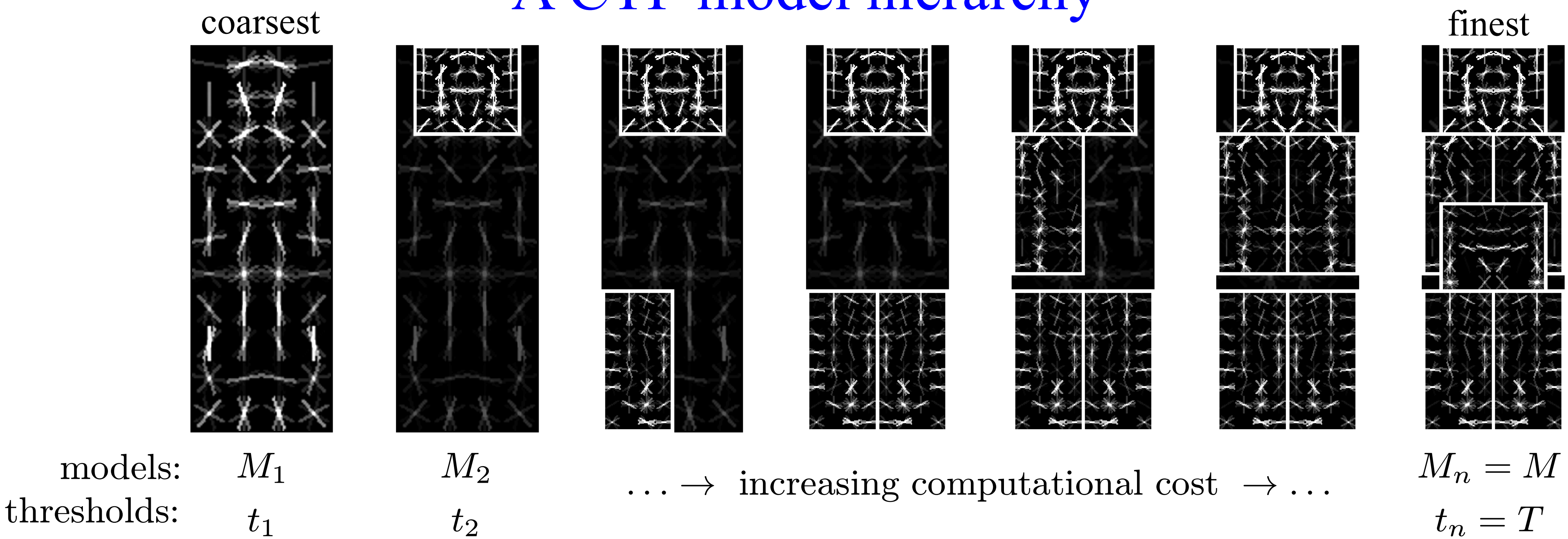
Our approach: coarse-to-fine detection + heuristic best-first search

Motivation

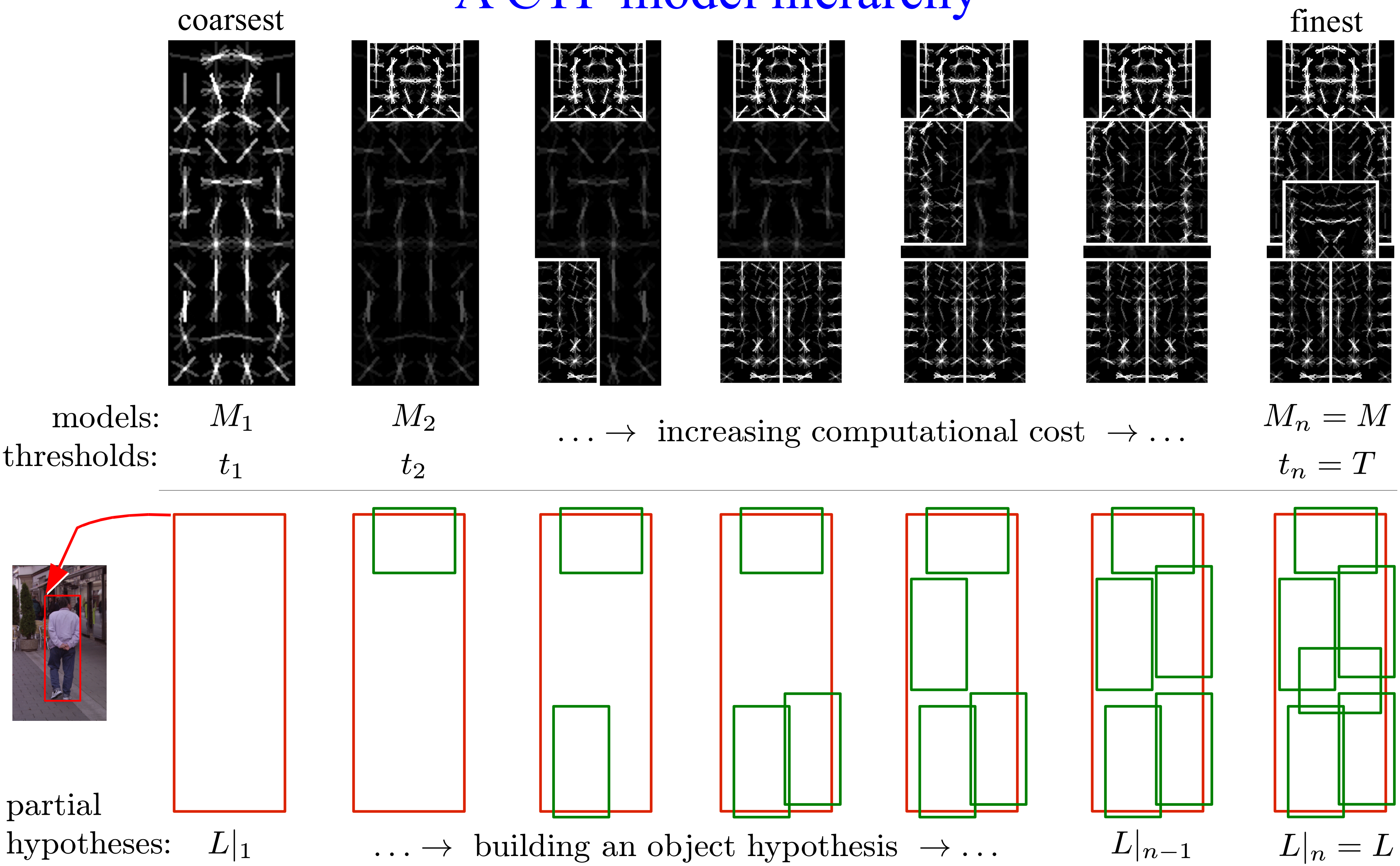
1. Coarse-to-fine detection (somewhat obvious)
   i. Exploit sparseness
   ii. Early termination of parses
2. Heuristic best-first search (less obvious)
   i. Non-maximal suppression of competing parses

This talk: start with star models.
Current work: general visual grammar parsing & learning.

A CTF model hierarchy

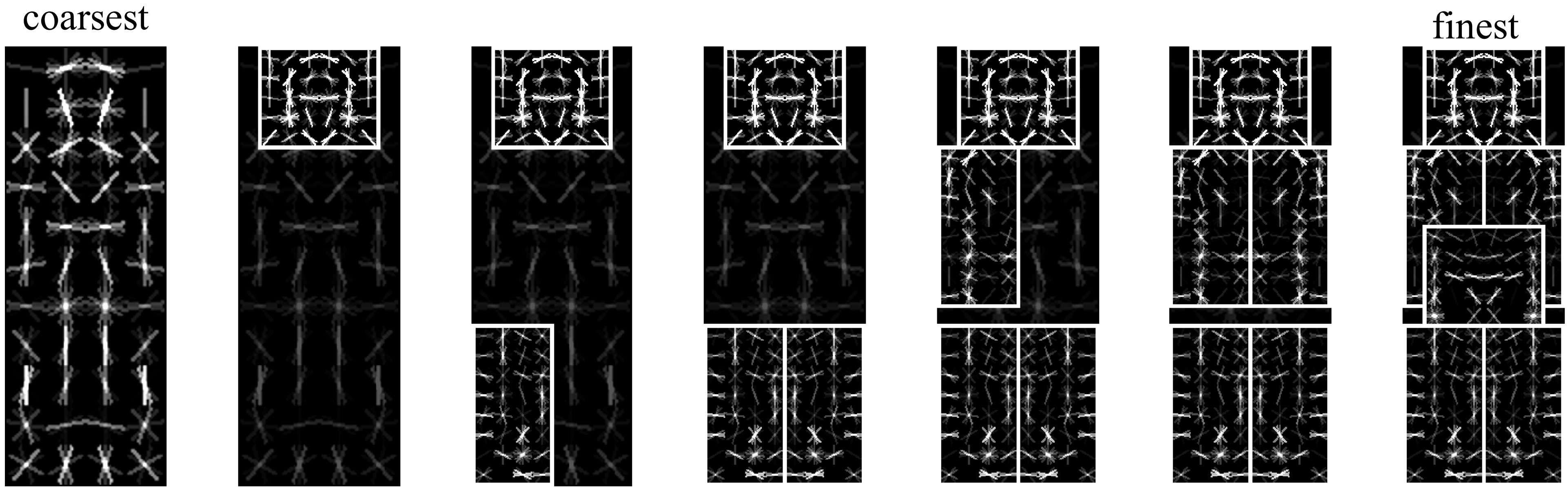


A CTF model hierarchy

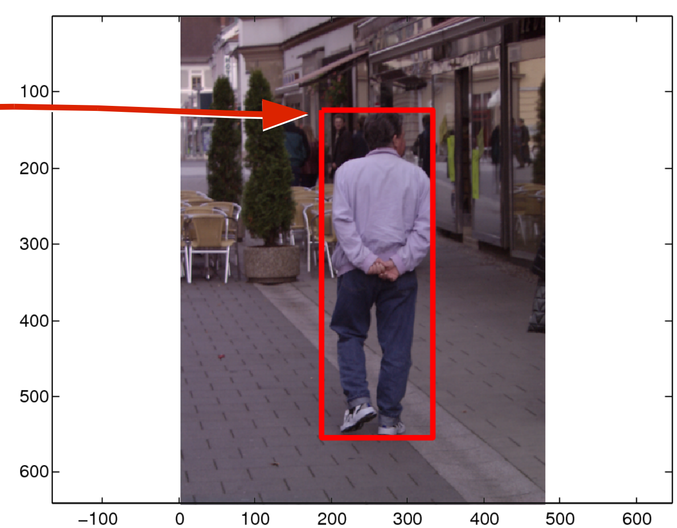


Require: $S_{M_i}(L|_i) \geq t_i$

CTF detection algorithm



$$\ldots \rightarrow \text{continue while } S_{M_i}(L|_i) \geq t_i \rightarrow \ldots$$

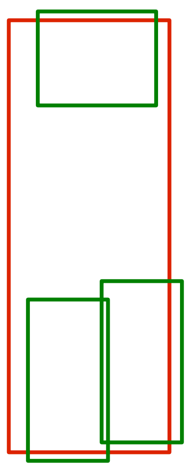


For each root location:
apply models in coarse to fine order while above termination threshold

Now, add heuristic best-first search →
“Heuristic coarse-to-fine detection”

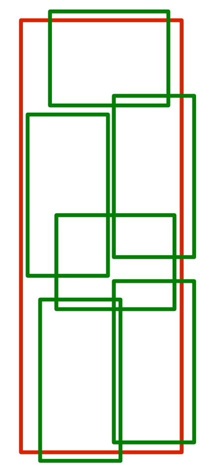
Best-first heuristic: *an upper bound on how much the score of a partial hypothesis can improve.*

e.g.:



$L|_4$

$$h(L|_4) \geq S_M(L|_{4*}) - S_{M_4}(L|_4)$$



$L|_{4*}$

best-first heuristic if it holds for any pair $L|_4$ and $L|_{4*}$.

Heuristic coarse-to-fine detection algorithm

- Don't iterate over locations in feature pyramid
- **Instead: priority queue of partial object hypotheses**
  - **Order queue by partial score + heuristic function**
  - **Apply non-maximal suppression on the fly → extra pruning**
  - Typically very fast if only looking for the single best detection

Heuristic coarse-to-fine detection algorithm

- Don't iterate over locations in feature pyramid
- **Instead: priority queue of partial object hypotheses**
  - **Order queue by partial score + heuristic function**
  - **Apply non-maximal suppression on the fly → extra pruning**
  - Typically very fast if only looking for the single best detection

Problem:
We don't know how to select admissible heuristics
that yield good best-first search order!

Solution: select *inadmissible* heuristics

Let (I, L) be an (image, object hypothesis) pair where $S_M(L) \geq T$.

Assume there's an unknown distribution D over an arbitrary set of (I, L) pairs.

D induces a distribution D_i over $h_i^*(L) = S_M(L) - S_{M_i}(L|_i)$, where $(I, L) \sim D$.

Let $\mathcal{H}_i$ be a sample from D_i.

Claim:

$$\hat{h}_i = \max\left(\mathcal{H}_i\right)$$

for $i = 1, \ldots, n$ is a "good" rule.

Theoretical justification: *Probably Approximately Admissible*

The rule is *good* in the sense that we can provide a PAC-like bound on the error rate.

Let $err(\hat{h}_1, \ldots, \hat{h}_n) = P_{(I,L)\sim D}(\hat{h}_i < h_i^*(L))$ for any $i = 1, \ldots, n$.

Theorem
Using the rule $\hat{h}_i = \max(\mathcal{H}_i)$, for fixed ϵ and δ,
if $\lvert\mathcal{H}_i\rvert > \frac{n}{\epsilon} \ln \frac{n}{\delta}$, then $P(err(\hat{h}_1, \ldots, \hat{h}_n) > \epsilon) < \delta$.

That is, the heuristic is "*approximately*" *admissible* with *high probability*.

Choosing inadmissible coarse-to-fine thresholds

1. Similar procedure as for heuristics.
2. Thresholds t_i are lower bounds on $S_{M_i}(L|_i) \implies$ min rule.
3. *Probably approximately admissible* theorem applies again.

Justification of the standard trick:
pick thresholds that yield a low false negative rate on training or validation data.

Equivalent to Zhang and Viola's "multiple-instance pruning."

[C. Zhang and P. Viola. "Multiple-instance pruning for learning efficient cascade detectors." NIPS, pages 1681–1688. 2007]

Experimental results I: PASCAL 2007 (comp3)

We used heuristic coarse-to-fine detection in *training and testing*.
Results are for two-component mixture models.

PASCAL 2007 Testing Time			
class	DP	HCTF	
aeroplane	5.70h	3.86h	1.48
bicycle	5.79h	2.37h	2.44
bottle	4.54h	2.28h	1.99
bus	5.75h	2.85h	2.02
car	4.37h	3.82h	1.15
cow	6.09h	3.40h	1.79
horse	6.00h	4.27h	1.41
motorbike	6.01h	2.21h	2.72
person	4.95h	4.45h	1.11
sheep	4.81h	2.85h	1.69
train	6.59h	2.54h	2.59
tvmonitor	9.63h	3.07h	3.13

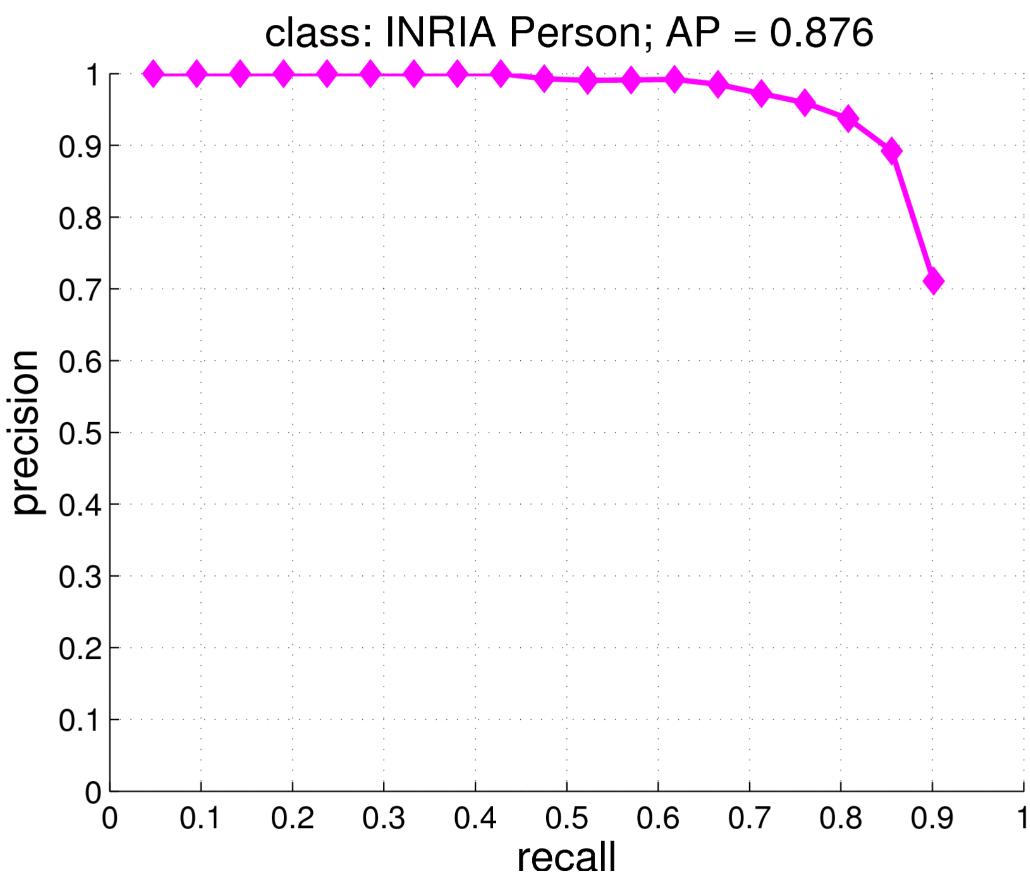
(~5000 images on a single CPU)

PASCAL 2007 Average Precision*			
class	DP	HCTF	
aeroplane	0.281	0.285	1.41%
bicycle	0.558	0.548	-1.80%
bottle	0.269	0.261	-3.07%
bus	0.437	0.443	1.42%
car	0.465	0.464	-0.06%
cow	0.207	0.195	-5.93%
horse	0.438	0.432	-1.23%
motorbike	0.384	0.397	3.48%
person	0.332	0.336	1.31%
sheep	0.196	0.200	2.43%
train	0.340	0.371	9.02%
tvmonitor	0.384	0.370	-3.84%

(* prior to any post-processing steps)

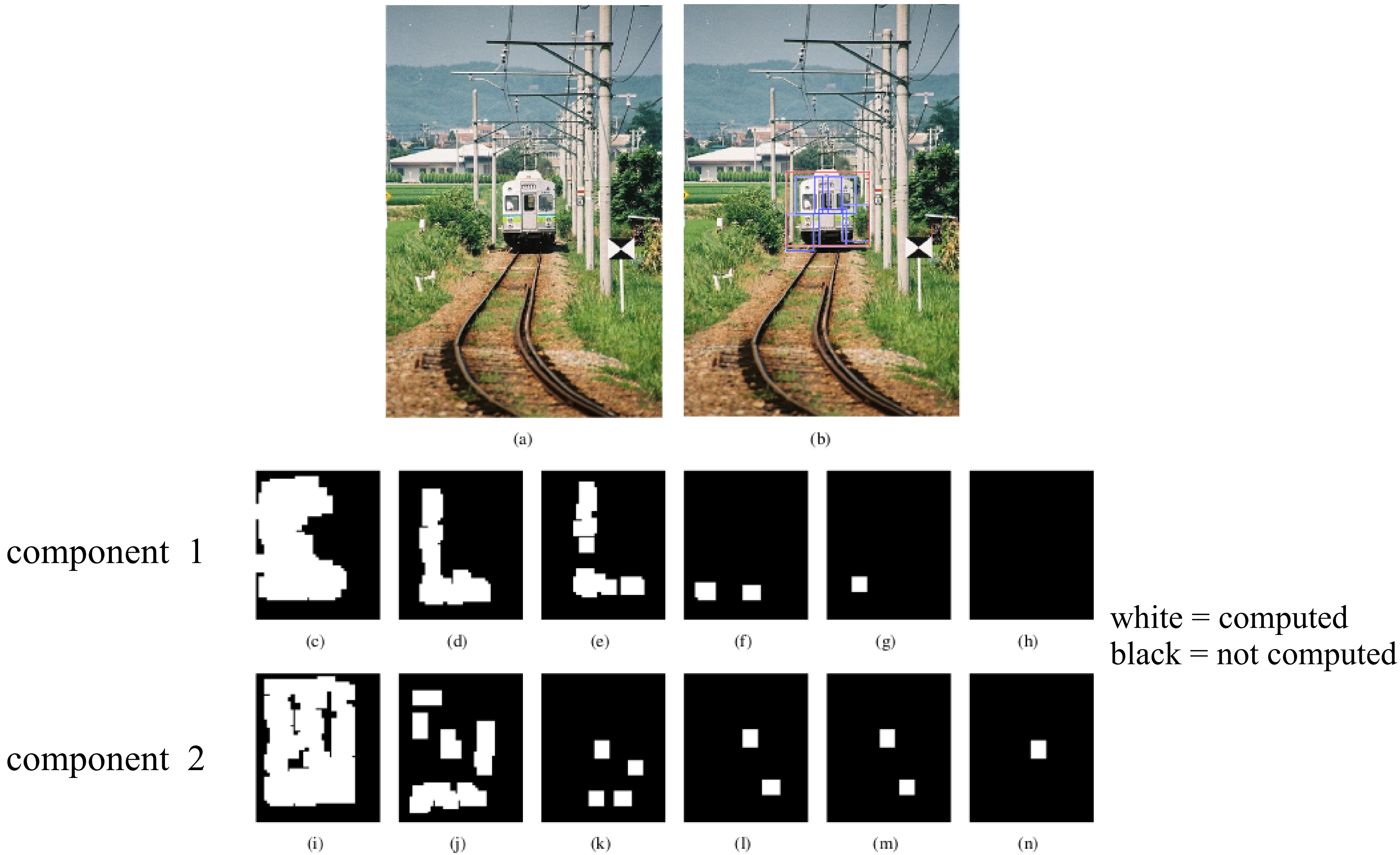
The theoretical bounds are somewhat loose. Around 200-300 examples are sufficient in practice.

Experimental results II: INRIA Person Dataset



Method DP scored AP = 0.878. Testing time = 40.2 minutes.
Method HCTF scored AP = 0.876. Testing time = 15.0 minutes (2.68x faster).

Pruning efficiency: where are filter scores computed?



Computation of $m_i(l)$ for $i = 2, ..., 7$. The square in (g) is 11x11 HOG cells.

Pruning efficiency: by the numbers

PASCAL 2007:

HCTF Pruning Efficiency

M_i	aero	bike	bottle	bus	car	cow	horse	mbike	person	sheep	train	tv
1	55.8%	88.4%	58.5%	67.5%	35.3%	72.9%	51.3%	86.1%	32.8%	38.4%	75.5%	60.3%
2	11.0%	8.9%	29.2%	28.2%	39.7%	14.7%	22.9%	11.6%	19.7%	41.8%	20.2%	17.7%
3	9.7%	1.4%	9.7%	2.3%	10.9%	9.7%	16.8%	1.4%	16.3%	17.3%	2.9%	15.0%
4	9.8%	1.0%	1.6%	0.7%	6.2%	2.6%	6.7%	0.6%	14.6%	2.1%	0.9%	5.9%
5	5.3%	0.2%	0.9%	0.9%	6.4%	0.1%	1.7%	0.3%	11.6%	0.2%	0.3%	0.5%
6	8.4%	0.1%	0.0%	0.4%	0.8%	0.0%	0.5%	0.0%	3.7%	0.1%	0.1%	0.2%
7	0.1%	0.0%	0.0%	0.1%	0.6%	0.0%	0.1%	0.0%	1.3%	0.1%	0.0%	0.2%

INRIA Person:

HCTF Pruning Efficiency

M_i	INRIA
1	92.9%
2	4.77%
3	1.12%
4	1.02%
5	0.10%
6	0.03%
7	0.02%

Some criticisms

- More complicated than parsing with dynamic programming
- Maybe a GPU implementation will be fast enough (even for 10,000 classes)?
- Loss of some robustness to occlusion (due to CTF hierarchy)
- Best-first search destroys cache coherence :-(

Conclusions and next steps

Conclusions:

- Heuristic coarse-to-fine detection *with inadmissible heuristics and thresholds* works well for our mixture models.
- Should see more significant payoffs on richer models (richer models have more discriminating filters → better pruning).
- Still, 2-3x speedup for some classes with 2-component mixture.

Next:

- More work is needed to successfully apply this technique to grammar models (beyond mixture models – AO* search)
- Continue progression to rich models: shared part dictionary, more than 2 levels, part-level mixtures, etc.

Thank you.

Questions?

Download source code*
for training and detection at
http://people.cs.uchicago.edu/~pff/latent/

*Code for our PASCAL 2008 system – not HCTF search or visual grammars.

Thank you.

Questions?

Download source code*
for training and detection at
http://people.cs.uchicago.edu/~pff/latent/

*Code for our PASCAL 2008 system – not HCTF search or visual grammars.

Thank you.

Questions?

Download source code*
for training and detection at
http://people.cs.uchicago.edu/~pff/latent/

*Code for our PASCAL 2008 system – not HCTF search or visual grammars.