Class logistics

- Tonight midnight, the take-home exam is due.
- Next week: spring break
- Following week, on Thursday, your project proposals are due.
  - Feel free to ask Xiaoxu or me for feedback or ideas regarding the project.
  - Auditors are welcome to do a project, and we'll read them and give feedback.

Generative Models

Bill Freeman, MIT

Some of these slides made with Andrew Blake, Microsoft Research Cambridge, UK

6.869 March 17, 2005

Last class

(a) We looked at ways to fit observations of probabilistic data, and EM.

(b) We're looking at the modularized joint probability distribution described by graphical models.

Making probability distributions modular, and therefore tractable:

Probabilistic graphical models

Vision is a problem involving the interactions of many variables: things can seem hopelessly complex. Everything is made tractable, or at least, simpler, if we modularize the problem. That's what probabilistic graphical models do, and let's examine that.

Readings: Jordan and Weiss intro article—fantastic!
Kevin Murphy web page—comprehensive and with pointers to many advanced topics

A toy example

Suppose we have a system of 5 interacting variables, perhaps some are observed and some are not. There's some probabilistic relationship between the 5 variables, described by their joint probability, P(x1, x2, x3, x4, x5).

If we want to find out what the likely state of variable x1 is (say, the position of the hand of some person we are observing), what can we do?

Two reasonable choices are: (a) find the value of x1 (and of all the other variables) that gives the maximum of P(x1, x2, x3, x4, x5); that's the MAP solution.
Or (b) marginalize over all the other variables and then take the mean or the maximum of the other variables. Marginalizing, then taking the mean, is equivalent to finding the MMSE solution. Marginalizing, then taking the max, is called the max marginal solution and sometimes a useful thing to do.

To find the marginal probability at x1, we have to take this sum:

$$\sum_{x_2, x_3, x_4, x_5} P(x_1, x_2, x_3, x_4, x_5)$$

If the system really is high dimensional, that will quickly become intractable. But if there is some modularity in $P(x_1, x_2, x_3, x_4, x_5)$ then things become tractable again.

Suppose the variables form a Markov chain: x1 causes x2 which causes x3, etc. We might draw out this relationship as follows:

$$x_1 \to x_2 \to x_3 \to x_4 \to x_5$$

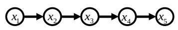
P(a,b) = P(b|a) P(a)

By the chain rule, for any probability distribution, we have:

$$P(x_1, x_2, x_3, x_4, x_5) = P(x_1)P(x_2, x_3, x_4, x_5 \mid x_1)$$
$$= P(x_1)P(x_2 \mid x_1)P(x_3, x_4, x_5 \mid x_1, x_2)$$
$$= P(x_1)P(x_2 \mid x_1)P(x_3 \mid x_1, x_2)P(x_4, x_5 \mid x_1, x_2, x_3)$$
$$= P(x_1)P(x_2 \mid x_1)P(x_3 \mid x_1, x_2)P(x_4 \mid x_1, x_2, x_3)P(x_5 \mid x_1, x_2, x_3, x_4)$$

But if we exploit the assumed modularity of the probability distribution over the 5 variables (in this case, the assumed Markov chain structure), then that expression simplifies:

$$= P(x_1)P(x_2 \mid x_1)P(x_3 \mid x_2)P(x_4 \mid x_3)P(x_5 \mid x_4)$$



Now our marginalization summations distribute through those terms:

$$\sum_{x_2, x_3, x_4, x_5} P(x_1, x_2, x_3, x_4, x_5) = \quad P(x_1)\sum_{x_2} P(x_2 \mid x_1)\sum_{x_3} P(x_3 \mid x_2)\sum_{x_4} P(x_4 \mid x_3)\sum_{x_5} P(x_5 \mid x_4)$$

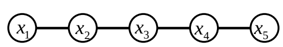
Belief propagation

Performing the marginalization by doing the partial sums is called “belief propagation”.

$$\sum_{x_2, x_3, x_4, x_5} P(x_1, x_2, x_3, x_4, x_5) = \quad P(x_1)\sum_{x_2} P(x_2 \mid x_1)\sum_{x_3} P(x_3 \mid x_2)\sum_{x_4} P(x_4 \mid x_3)\sum_{x_5} P(x_5 \mid x_4)$$

In this example, it has saved us a lot of computation. Suppose each variable has 10 discrete states. Then, not knowing the special structure of P, we would have to perform 10000 additions (10^4) to marginalize over the four variables.
But doing the partial sums on the right hand side, we only need 40 additions (10*4) to perform the same marginalization!

Another modular probabilistic structure, more common in vision problems, is an undirected graph:



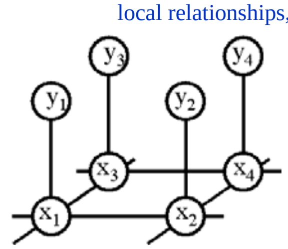
The joint probability for this graph is given by:

$$P(x_1, x_2, x_3, x_4, x_5) = \Phi(x_1, x_2)\Phi(x_2, x_3)\Phi(x_3, x_4)\Phi(x_4, x_5)$$

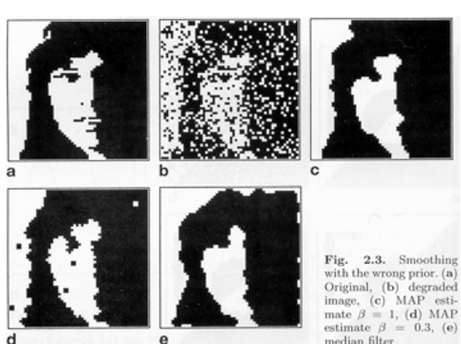
Where $\Phi(x_1, x_2)$ is called a “compatibility function”. We can define compatibility functions we result in the same joint probability as for the directed graph described in the previous slides; for that example, we could use either form.

Markov Random Fields

- Allows rich probabilistic models for images.
- But built in a local, modular way. Learn local relationships, get global effects out.

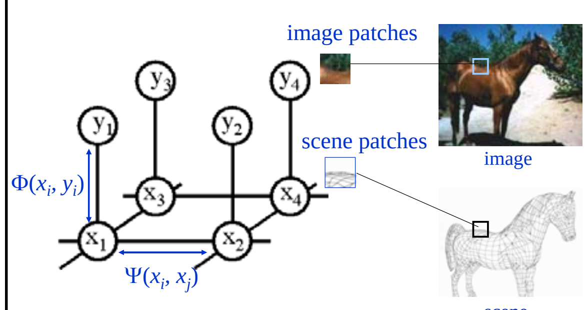


MRF nodes as pixels



Winkler, 1995, p. 32

MRF nodes as patches



Network joint probability

$$P(x, y) = \frac{1}{Z} \prod_{i,j} \Psi(x_i, x_j) \prod_{i} \Phi(x_i, y_i)$$

scene

image

Scene-scene compatibility function

neighboring scene nodes

Image-scene compatibility function

local observations

In order to use MRFs:

- Given observations y, and the parameters of the MRF, how <u>infer</u> the hidden variables, x?
- How <u>learn</u> the parameters of the MRF?

Outline of MRF section

- Inference in MRF's.
  - Gibbs sampling, simulated annealing
  - Iterated condtional modes (ICM)
  - Variational methods
  - Belief propagation
  - Graph cuts
- Vision applications of inference in MRF's.
- Learning MRF parameters.
  - Iterative proportional fitting (IPF)

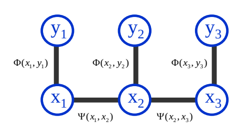
Variational methods

- Reference: Tommi Jaakkola's tutorial on variational methods,
  http://www.ai.mit.edu/people/tommi/
- Example: mean field
  - For each node
    - Calculate the expected value of the node, conditioned on the mean values of the neighbors.

Outline of MRF section

- Inference in MRF's.
  - Gibbs sampling, simulated annealing
  - Iterated condtional modes (ICM)
  - Variational methods
  - Belief propagation
  - Graph cuts
- Vision applications of inference in MRF's.
- Learning MRF parameters.
  - Iterative proportional fitting (IPF)

Derivation of belief propagation

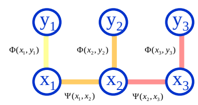


$$x_{1MMSE} = \underset{x_1}{\text{mean}} \; \underset{x_2}{\text{sum}} \; \underset{x_3}{\text{sum}} \; P(x_1, x_2, x_3, y_1, y_2, y_3)$$

The posterior factorizes

$$x_{1MMSE} = \underset{x_1}{\text{mean}}\ \underset{x_2}{\text{sum}}\ \underset{x_3}{\text{sum}}\ P(x_1, x_2, x_3, y_1, y_2, y_3)$$

$$= \underset{x_1}{\text{mean}}\ \underset{x_2}{\text{sum}}\ \underset{x_3}{\text{sum}}\ \Phi(x_1, y_1)\ \Phi(x_2, y_2)\ \Psi(x_1, x_2)\ \Phi(x_3, y_3)\ \Psi(x_2, x_3)$$

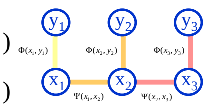


Propagation rules

$$x_{1MMSE} = \underset{x_1}{\text{mean}}\ \underset{x_2}{\text{sum}}\ \underset{x_3}{\text{sum}}\ P(x_1, x_2, x_3, y_1, y_2, y_3)$$

$$x_{1MMSE} = \underset{x_1}{\text{mean}}\ \underset{x_2}{\text{sum}}\ \underset{x_3}{\text{sum}}\ \Phi(x_1, y_1)\ \Phi(x_2, y_2)\ \Psi(x_1, x_2)\ \Phi(x_3, y_3)\ \Psi(x_2, x_3)$$

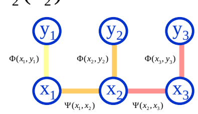
$$x_{1MMSE} = \underset{x_1}{\text{mean}}\ \Phi(x_1, y_1)\ \underset{x_2}{\text{sum}}\ \Phi(x_2, y_2)\ \Psi(x_1, x_2)\ \underset{x_3}{\text{sum}}\ \Phi(x_3, y_3)\ \Psi(x_2, x_3)$$



Propagation rules

$$x_{1MMSE} = \underset{x_1}{\text{mean}}\ \Phi(x_1, y_1)\ \underset{x_2}{\text{sum}}\ \Phi(x_2, y_2)\ \Psi(x_1, x_2)\ \underset{x_3}{\text{sum}}\ \Phi(x_3, y_3)\ \Psi(x_2, x_3)$$

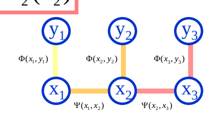
$$M_1^2(x_1) = \underset{x_2}{\text{sum}}\ \Psi(x_1, x_2)\ \Phi(x_2, y_2)\ M_2^3(x_2)$$



Propagation rules

$$x_{1MMSE} = \underset{x_1}{\text{mean}}\ \Phi(x_1, y_1)\ \underset{x_2}{\text{sum}}\ \boxed{\Phi(x_2, y_2)\ \Psi(x_1, x_2)}\ \underset{x_3}{\text{sum}}\ \boxed{\Phi(x_3, y_3)\ \Psi(x_2, x_3)}$$

$$\boxed{M_1^2(x_1)} = \underset{x_2}{\text{sum}}\ \Psi(x_1, x_2)\ \Phi(x_2, y_2)\ \boxed{M_2^3(x_2)}$$



Belief propagation: the nosey neighbor rule

"Given everything that I know, here's what I think you should think"

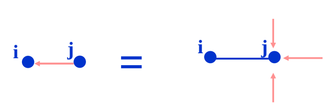
(Given the probabilities of my being in different states, and how my states relate to your states, here's what I think the probabilities of your states should be)

Belief propagation messages

A message: can be thought of as a set of weights on each of your possible states

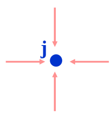
To send a message: Multiply together all the incoming messages, except from the node you're sending to, then multiply by the compatibility matrix and marginalize over the sender's states.

$$M_i^j(x_i) = \sum_{x_j} \psi_{ij}(x_i, x_j) \prod_{k \in N(j)\backslash i} M_j^k(x_j)$$



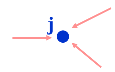
Beliefs

To find a node's beliefs: Multiply together all the messages coming in to that node.



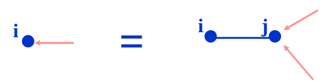
$$b_j(x_j) = \prod_{k \in N(j)} M_j^k(x_j)$$

Belief, and message updates



$$b_j(x_j) = \prod_{k \in N(j)} M_j^k(x_j)$$

$$M_i^j(x_i) = \sum_{x_j} \psi_{ij}(x_i, x_j) \prod_{k \in N(j) \backslash i} M_j^k(x_j)$$

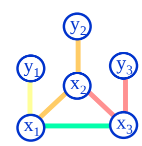


Optimal solution in a chain or tree: Belief Propagation

- "Do the right thing" Bayesian algorithm.
- For Gaussian random variables over time: Kalman filter.
- For hidden Markov models: forward/backward algorithm (and MAP variant is Viterbi).

No factorization with loops!

$$x_{1MMSE} = \underset{x_1}{\text{mean}}\, \Phi(x_1, y_1) \underset{x_2}{\text{sum}}\, \Phi(x_2, y_2)\, \Psi(x_1, x_2) \underset{x_3}{\text{sum}}\, \Phi(x_3, y_3)\, \Psi(x_2, x_3)\ \Psi(x_1, x_3)$$



Justification for running belief propagation in networks with loops

- Experimental results:
  - Error-correcting codes — Kschischang and Frey, 1998; McEliece et al., 1998
  - Vision applications — Freeman and Pasztor, 1999; Frey, 2000
- Theoretical results:
  - For Gaussian processes, means are correct. — Weiss and Freeman, 1999
  - Large neighborhood local maximum for MAP. — Weiss and Freeman, 2000
  - Equivalent to Bethe approx. in statistical physics. — Yedidia, Freeman, and Weiss, 2000
  - Tree-weighted reparameterization — Wainwright, Willsky, Jaakkola, 2001

Statistical mechanics interpretation

U - TS = Free energy

U = avg. energy = $\sum_{states} p(x_1, x_2, \ldots) E(x_1, x_2, \ldots)$

T = temperature

S = entropy = $-\sum_{states} p(x_1, x_2, \ldots) \ln p(x_1, x_2, \ldots)$

Free energy formulation

Defining

$$\Psi_{ij}(x_i,x_j)=e^{-E(x_i,x_j)/T} \qquad \Phi_i(x_i)=e^{-E(x_i)/T}$$

then the probability distribution $P(x_1,x_2,\ldots)$ that minimizes the F.E. is precisely the true probability of the Markov network,

$$P(x_1,x_2,\ldots)=\prod_{ij}\Psi_{ij}(x_i,x_j)\prod_i\Phi_i(x_i)$$

Approximating the Free Energy

Exact: $F[p(x_1,x_2,\ldots,x_N)]$

Mean Field Theory: $F[b_i(x_i)]$

Bethe Approximation : $F[b_i(x_i),b_{ij}(x_i,x_j)]$

Kikuchi Approximations:

$$F[b_i(x_i),b_{ij}(x_i,x_j),b_{ijk}(x_i,x_j,x_k),\ldots]$$

Mean field approximation to free energy

U - TS = Free energy

$$F_{MeanField}(b_i)=\sum_{(ij)}\sum_{x_i,x_j}b_i(x_i)b_j(x_j)E_{ij}(x_i,x_j)+\sum_i\sum_{x_i}b_i(x_i)T\ln b_i(x_i)$$

The variational free energy is, up to an additive constant, equal to the Kllback-Leibler divergence between b(x) and the true probability, P(x). KL divergence:

$$D_{KL}(b\,\|\,P)=\sum_{x_1,x_2,\ldots}\prod_i b_i(x)\ln\frac{\prod_i b_i(x)}{P(x)}$$

Setting deriv w.r.t b_i=0

U - TS = Free energy

Corresponds to eq. 18 in Jordan and Weiss ms.

$$b_i(x_i)=\alpha\exp\Big(-\sum_{(ij)}\sum_{x_j}b_j(x_j)E_{ij}(x_i,x_j)/T\Big)$$

In words: "Set the probability of each state x_i at node i to be proportional to e to the minus expected energy corresponding to each state x_i, given the expected values of all the neighboring states."

Bethe Approximation

On tree-like lattices, exact formula:

$$p(x_1,x_2,\ldots,x_N)=\prod_{(ij)}p_{ij}(x_i,x_j)\prod_i[p_i(x_i)]^{1-q_i}$$

$$F_{Bethe}(b_i,b_{ij})=\sum_{(ij)}\sum_{x_i,x_j}b_{ij}(x_i,x_j)(E_{ij}(x_i,x_j)+T\ln b_{ij}(x_i,x_j))$$
$$+\sum_i(1-q_i)\sum_{x_i}b_i(x_i)(E_i(x_i)+T\ln b_i(x_i))$$

Gibbs Free Energy

$$F_{Bethe}(b_i,b_{ij})+\sum_{(ij)}\gamma_{ij}\{\sum_{x_i,x_j}b_{ij}(x_i,x_j)-1\}$$
$$+\sum_{x_j}\sum_{(ij)}\lambda_{ij}(x_j)\{\sum_{x_i}b_{ij}(x_i,x_j)-b_j(x_j)\}$$

Gibbs Free Energy

$$F_{Bethe}(b_i, b_{ij}) + \sum_{(ij)} \gamma_{ij}\{\sum_{x_i,x_j} b_{ij}(x_i,x_j) - 1\}$$

$$+\sum_{x_j}\sum_{(ij)} \lambda_{ij}(x_j)\{\sum_{x_i} b_{ij}(x_i,x_j) - b_j(x_j)\}$$

Set derivative of Gibbs Free Energy w.r.t. b_{ij}, b_i terms to zero:

$$b_{ij}(x_i,x_j) = k\,\Psi_{ij}(x_i,x_j)\exp\left(\frac{-\lambda_{ij}(x_i)}{T}\right)$$

$$b_i(x_i) = k\,\Phi(x_i)\exp\left(\frac{\sum_{j\in N(i)} \lambda_{ij}(x_i)}{T}\right)$$

Belief Propagation = Bethe

Lagrange multipliers $\lambda_{ij}(x_j)$

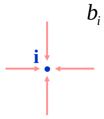
enforce the constraints $b_j(x_j) = \sum_{x_i} b_{ij}(x_i,x_j)$

Bethe stationary conditions = message update rules

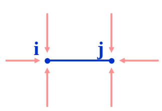
with $\lambda_{ij}(x_j) = T\ln \prod_{k\in N(j)\backslash i} M_j^k(x_j)$

Region marginal probabilities

$$b_i(x_i) = k\,\Phi(x_i)\prod_{k\in N(i)} M_i^k(x_i)$$

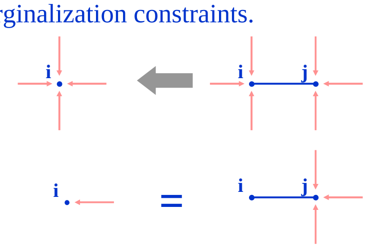


$$b_{ij}(x_i,x_j) = k\,\Psi(x_i,x_j)\prod_{k\in N(i)\backslash j} M_i^k(x_i)\prod_{k\in N(j)\backslash i} M_j^k(x_j)$$



Belief propagation equations

Belief propagation equations come from the marginalization constraints.



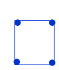
$$M_i^j(x_i) = \sum_{x_j} \psi_{ij}(x_i,x_j)\prod_{k\in N(j)\backslash i} M_j^k(x_j)$$

Results from Bethe free energy analysis

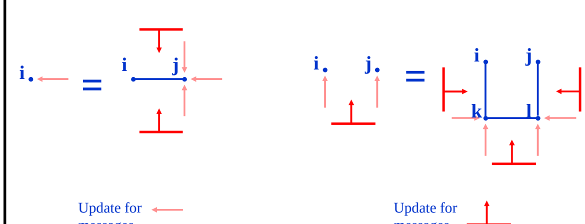
- Fixed point of belief propagation equations iff. Bethe approximation stationary point.
- Belief propagation always has a fixed point.
- Connection with variational methods for inference: both minimize approximations to Free Energy,
  - **variational**: usually use primal variables.
  - **belief propagation**: fixed pt. equs. for dual variables.
- Kikuchi approximations lead to more accurate belief propagation algorithms.
- Other Bethe free energy minimization algorithms—Yuille, Welling, etc.

Kikuchi message-update rules

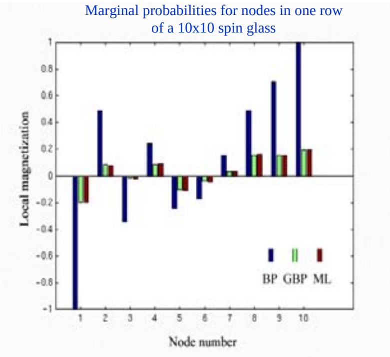
Groups of nodes send messages to other groups of nodes.



Typical choice for Kikuchi cluster.



Generalized belief propagation



References on BP and GBP

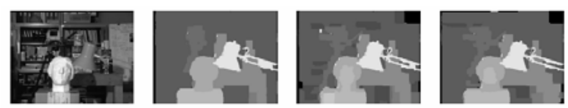
- J. Pearl, 1985
  - classic
- Y. Weiss, NIPS 1998
  - Inspires application of BP to vision
- W. Freeman et al learning low-level vision, IJCV 1999
  - Applications in super-resolution, motion, shading/paint discrimination
- H. Shum et al, ECCV 2002
  - Application to stereo
- M. Wainwright, T. Jaakkola, A. Willsky
  - Reparameterization version
- J. Yedidia, AAAI 2000
  - The clearest place to read about BP and GBP.

Graph cuts

- Algorithm: uses node label swaps or expansions as moves in the algorithm to reduce the energy. Swaps many labels at once, not just one at a time, as with ICM.
- Find which pixel labels to swap using min cut/max flow algorithms from network theory.
- Can offer bounds on optimality.
- See Boykov, Veksler, Zabih, IEEE PAMI 23 (11) Nov. 2001 (available on web).

Comparison of graph cuts and belief propagation

Comparison of Graph Cuts with Belief Propagation for Stereo, using Identical MRF Parameters, ICCV 2003.
Marshall F. Tappen William T. Freeman



(a) Tsukuba Image (b) Graph Cuts (c) Synchronous BP (d) Accelerated BP

Figure 3. Results produced by the three algorithms on the Tsukuba image. The parameters used to generate this field were $s = 50, T = 4, P = 2$. Again, Graph Cuts produces a much smoother solution. Belief Propagation does maintain some structures that are lost in the Graph Cuts solution, such as the camera and the face in the foreground.

Ground truth, graph cuts, and belief propagation disparity solution energies

Image	Energy of MRF Labelling Returned ($\times 10^3$) Ground-Truth	Graph Cuts	Synchronous Belief Prop	% Energy from Occluded Matching Costs
Map	757	383	442	61%
Sawtooth	6591	1652	1713	79%
Tsukuba	1852	663	775	61%
Venus	5739	1442	1501	76%

Figure 2. Field Energies for the MRF labelled using ground-truth data compared to the energies for the fields labelled using Graph Cuts and Belief Propagation. Notice that the solutions returned by the algorithms consistently have a much lower energy than the labellings produced from the ground-truth, showing a mismatch between the MRF formulation and the ground-truth. The final column contains the percentage of each ground-truth solution's energy that comes from matching costs of occluded pixels.

Graph cuts versus belief propagation

- Graph cuts consistently gave slightly lower energy solutions for that stereo-problem MRF, although BP ran faster, although there is now a faster graph cuts implementation than what we used…
- However, here's why I still use Belief Propagation:
  - Works for any compatibility functions, not a restricted set like graph cuts.
  - I find it very intuitive.
  - Extensions: sum-product algorithm computes MMSE, and Generalized Belief Propagation gives you very accurate solutions, at a cost of time.

MAP versus MMSE

(a) MAP Estimate (b) MMSE Estimate

Figure 7. Comparison of MAP and MMSE estimates on a different MRF formulation. The MAP estimate chooses the most likely discrete disparity level for each point, resulting in a depth-map with stair-stepping effects. Using the MMSE estimate assigns sub-pixel disparities, resulting in a smooth depth map.

Show program comparing some methods on a simple MRF

testMRF.m

Outline of MRF section

- Inference in MRF’s.
  – Gibbs sampling, simulated annealing
  – Iterated condtional modes (ICM)
  – Variational methods
  – Belief propagation
  – Graph cuts
- Vision applications of inference in MRF’s.
- Learning MRF parameters.
  – Iterative proportional fitting (IPF)

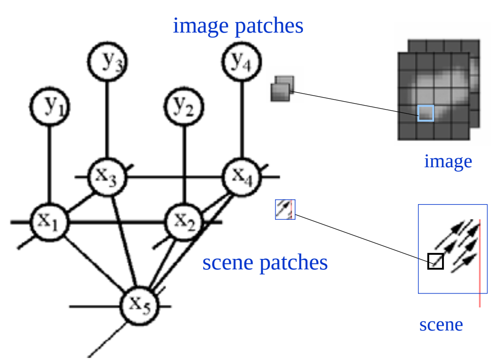
Vision applications of MRF’s

- Stereo
- Motion estimation
- Super-resolution
- Many others…

Vision applications of MRF’s

- Stereo
- Motion estimation
- Super-resolution
- Many others…

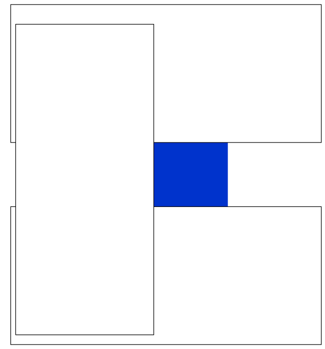
Motion application



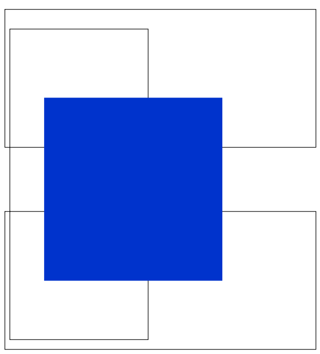
What behavior should we see in a motion algorithm?

- Aperture problem
- Resolution through propagation of information
- Figure/ground discrimination

The aperture problem



The aperture problem

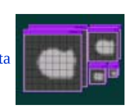


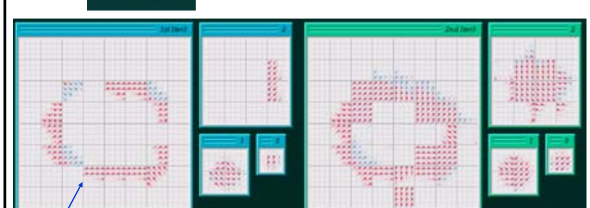
Program demo

Inference: **Motion estimation results**

(maxima of scene probability distributions displayed)

Image data





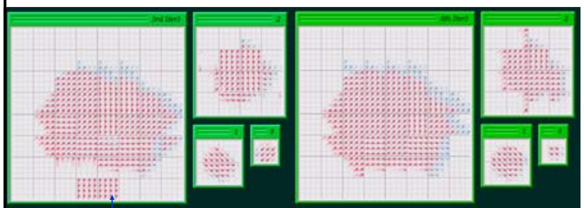
Iterations 0 and 1

Initial guesses only show motion at edges.

Motion analysis: related work

- **Markov network**
  - Luettgen, Karl, Willsky and collaborators.
- **Neural network or learning-based**
  - Nowlan & T. J. Senjowski; Sereno.
- **Optical flow analysis**
  - Weiss & Adelson; Darrell & Pentland; Ju, Black & Jepson; Simoncelli; Grzywacz & Yuille; Hildreth; Horn & Schunk; etc.

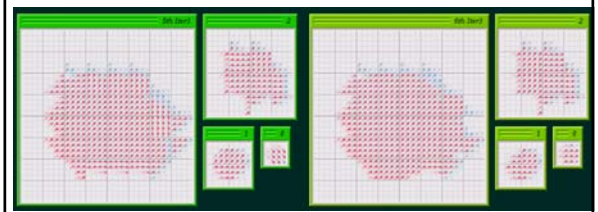
Motion estimation results
(maxima of scene probability distributions displayed)



Iterations 2 and 3

Figure/ground still unresolved here.

Motion estimation results
(maxima of scene probability distributions displayed)



Iterations 4 and 5

Final result compares well with vector quantized true (uniform) velocities.

Vision applications of MRF's

- Stereo
- Motion estimation
- Super-resolution
- Many others…

Super-resolution

- Image: low resolution image
- Scene: high resolution image

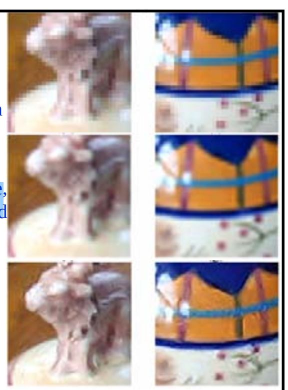
ultimate goal...

Pixel-based images are not resolution independent

Pixel replication

Cubic spline, sharpened

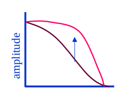
Training-based super-resolution



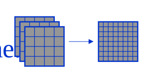
Polygon-based graphics images are resolution independent

3 approaches to perceptual sharpening

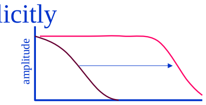
(1) Sharpening; boost existing high frequencies.



(2) Use multiple frames to obtain higher sampling rate in a still frame



(3) Estimate high frequencies not present in image, although implicitly defined.



In this talk, we focus on (3), which we'll call "super-resolution".

Super-resolution: other approaches

- Schultz and Stevenson, 1994
- Pentland and Horowitz, 1993
- fractal image compression (Polvere, 1998; Iterated Systems)
- astronomical image processing (eg. Gull and Daniell, 1978; "pixons" http://casswww.ucsd.edu/puetter.html)

Training images, **~100,000 image/scene patch pairs**

Images from two Corel database categories: "giraffes" and "urban skyline".

Do a first interpolation

Zoomed low-resolution

Low-resolution

Zoomed low-resolution

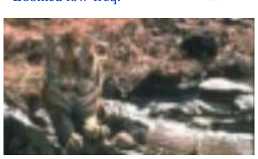
Full frequency original

Low-resolution

Representation

Zoomed low-freq.

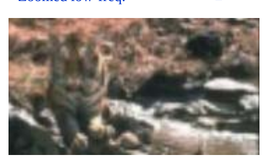
Full freq. original

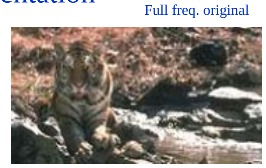


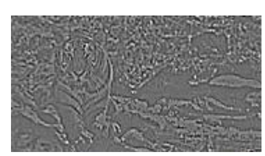
Representation

Zoomed low-freq.

Full freq. original





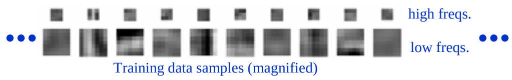


True high freqs

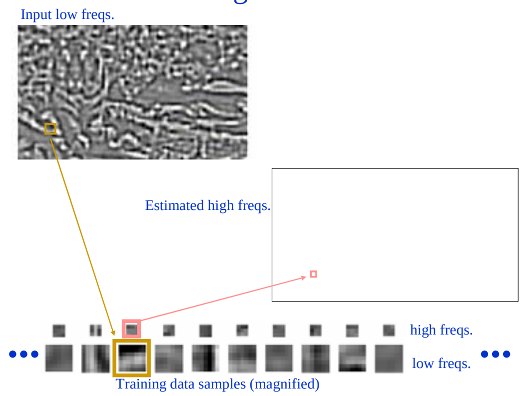
Low-band input (contrast normalized, PCA fitted)

(to minimize the complexity of the relationships we have to learn, we remove the lowest frequencies from the input image, and normalize the local contrast level).

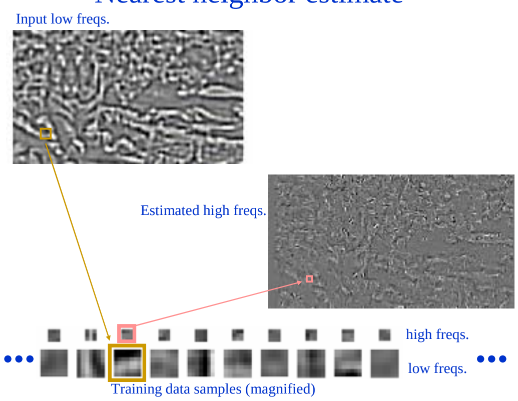
Gather ~100,000 patches



Nearest neighbor estimate



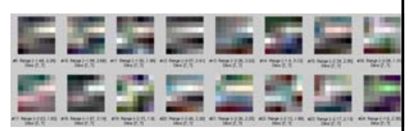
Nearest neighbor estimate



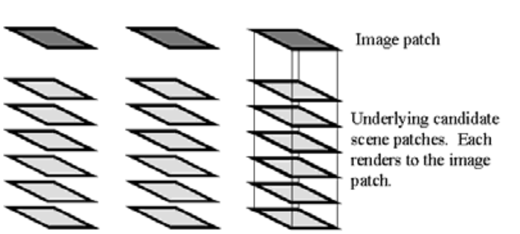
Example: input image patch, and closest matches from database

Input patch

Closest image patches from database



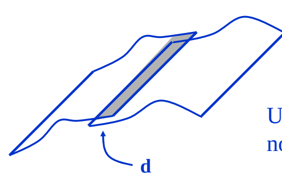
Corresponding high-resolution patches from database



Scene-scene compatibility function, $\Psi(x_i, x_j)$

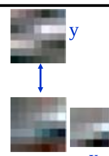
Assume overlapped regions, d, of hi-res. patches differ by Gaussian observation noise:

$$\Psi(x_i, x_j) = \exp^{-|d_i - d_j|^2/2\sigma^2}$$



Uniqueness constraint, not smoothness.

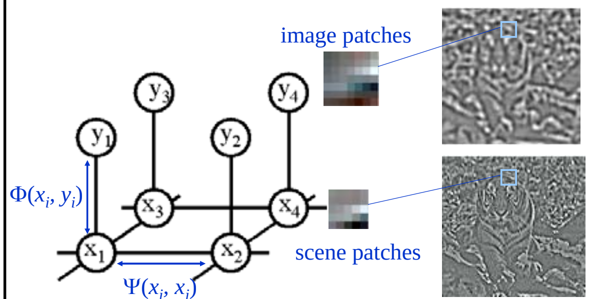
Image-scene compatibility function, $\Phi(x_i, y_i)$



Assume Gaussian noise takes you from observed image patch to synthetic sample:

$$\Phi(x_i, y_i) = \exp^{-|y_i - y(x_i)|^2 / 2\sigma^2}$$

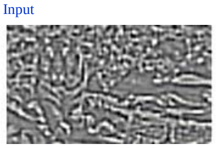
Markov network

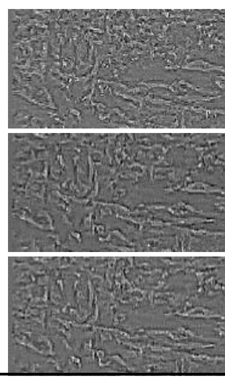


Belief Propagation

After a few iterations of belief propagation, the algorithm selects spatially consistent high resolution interpretations for each low-resolution patch of the input image.

Input





Iter. 0

Iter. 1

Iter. 3

Zooming 2 octaves

We apply the super-resolution algorithm recursively, zooming up 2 powers of 2, or a factor of 4 in each dimension.

85 x 51 input

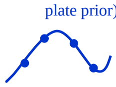
Cubic spline zoom to 340x204

Max. likelihood zoom to 340x204

Now we examine the effect of the prior assumptions made about images on the high resolution reconstruction. First, cubic spline interpolation.

Original 50x58

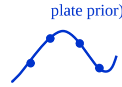
(cubic spline implies thin plate prior)



True 200x232

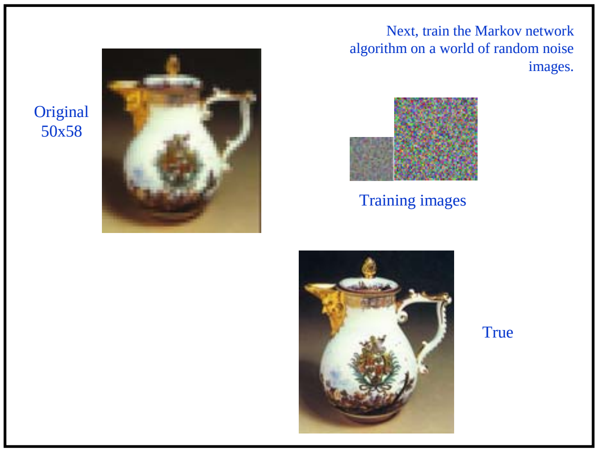
Original 50x58

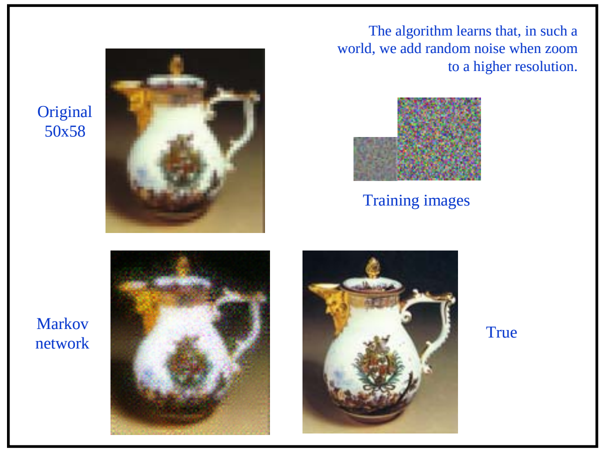
(cubic spline implies thin plate prior)

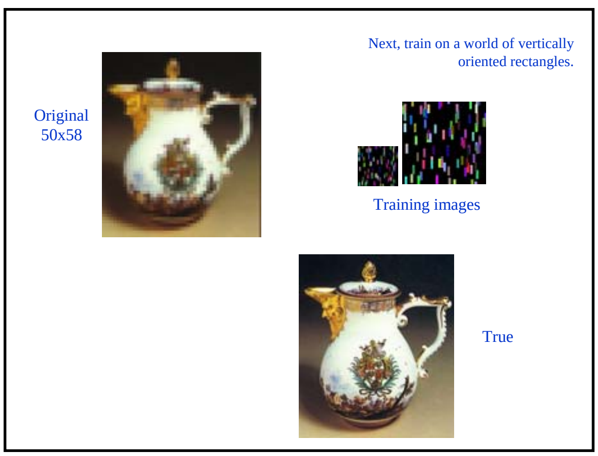


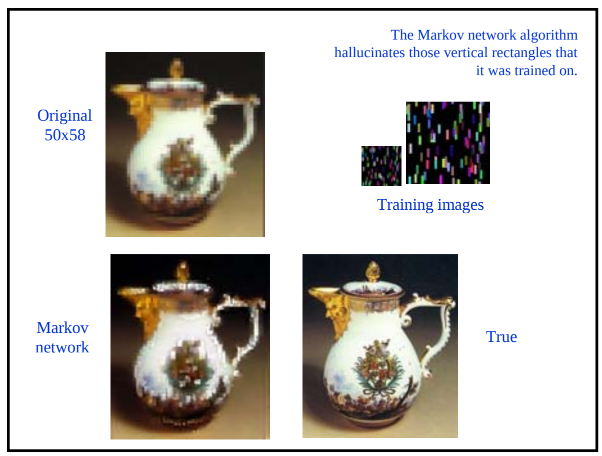
Cubic spline

True 200x232

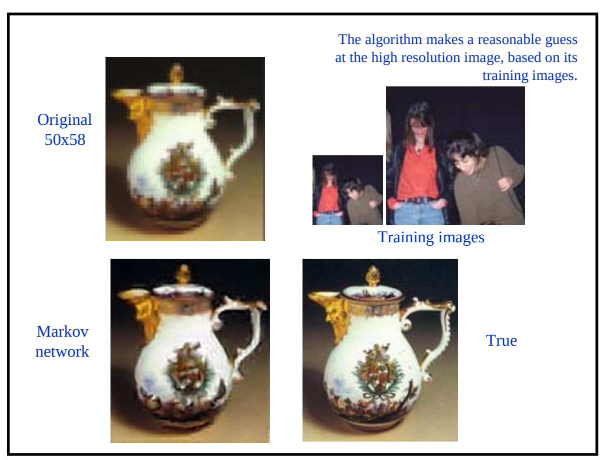

Next, train the Markov network algorithm on a world of random noise images.
Original 50x58
Training images
True


The algorithm learns that, in such a world, we add random noise when zoom to a higher resolution.
Original 50x58
Training images
Markov network
True


Next, train on a world of vertically oriented rectangles.
Original 50x58
Training images
True


The Markov network algorithm hallucinates those vertical rectangles that it was trained on.
Original 50x58
Training images
Markov network
True


Now train on a generic collection of images.
Original 50x58
Training images
True


The algorithm makes a reasonable guess at the high resolution image, based on its training images.
Original 50x58
Training images
Markov network
True

Generic training images

Next, train on a generic set of training images. Using the same camera as for the test image, but a random collection of photographs.

Original 70x70

Cubic Spline

Markov net, training: generic

True 280x280

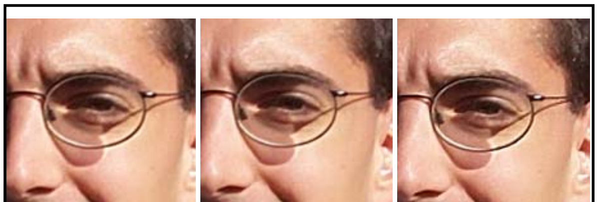
Kodak Imaging Science Technology Lab test.

3 test images, 640x480, to be zoomed up by 4 in each dimension.

8 judges, making 2-alternative, forced-choice comparisons.

Algorithms compared

- Bicubic Interpolation
- Mitra's Directional Filter
- Fuzzy Logic Filter
- Vector Quantization
- VISTA



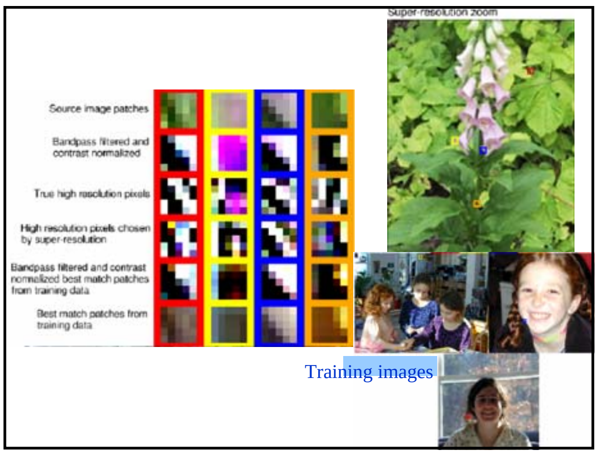
Bicubic spline | Altamira | VISTA

Bicubic spline | Altamira | VISTA

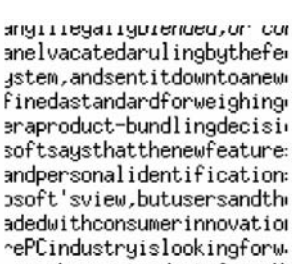
User preference test results

"The observer data indicates that six of the observers ranked Freeman's algorithm as the most preferred of the five tested algorithms. However the other two observers rank Freeman's algorithm as the least preferred of all the algorithms….

Freeman's algorithm produces prints which are by far the sharpest out of the five algorithms. However, this sharpness comes at a price of artifacts (spurious detail that is not present in the original scene). Apparently the two observers who did not prefer Freeman's algorithm had strong objections to the artifacts. The other observers apparently placed high priority on the high level of sharpness in the images created by Freeman's algorithm."



Training image



Processed image

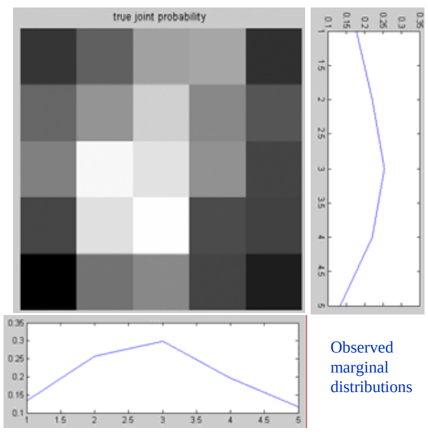
Outline of MRF section

- Inference in MRF's.
  - Gibbs sampling, simulated annealing
  - Iterated conditional modes (ICM)
  - Variational methods
  - Belief propagation
  - Graph cuts
- Vision applications of inference in MRF's.
- Learning MRF parameters.
  - Iterative proportional fitting (IPF)

Learning MRF parameters, labeled data

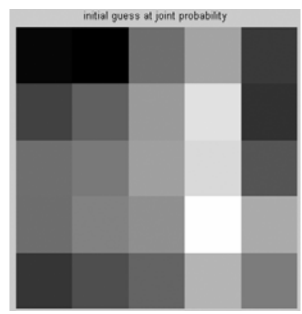
Iterative proportional fitting lets you make a maximum likelihood estimate of a joint distribution from observations of various marginal distributions.

True joint probability



Observed marginal distributions

Initial guess at joint probability



IPF update equation

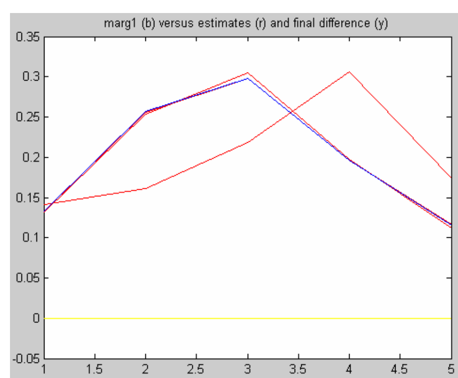
$$P(x_1, x_2, \ldots, x_d)^{(t+1)} = P(x_1, x_2, \ldots, x_d)^{(t)} \frac{P(x_i)^{\text{observed}}}{P(x_i)^{(t)}}$$

Scale the previous iteration's estimate for the joint probability by the ratio of the true to the predicted marginals.

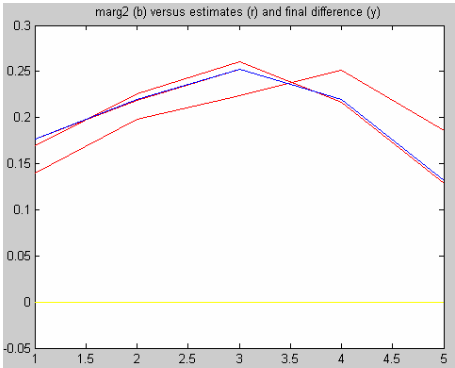
Gives gradient ascent in the likelihood of the joint probability, given the observations of the marginals.

See: Michael Jordan's book on graphical models

Convergence of to correct marginals by IPF algorithm

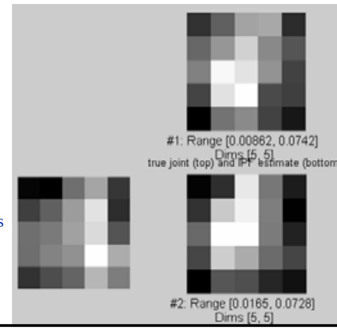


Convergence of to correct marginals by IPF algorithm



IPF results for this example: comparison of joint probabilities

True joint probability



Initial guess

Final maximum entropy estimate

Application to MRF parameter estimation

- Can show that for the ML estimate of the clique potentials, $\phi_c(x_c)$, the empirical marginals equal the model marginals,

$$\tilde{p}(x_c) = p(x_c)$$

- This leads to the IPF update rule for $\phi_c(x_c)$

$$\phi_C^{(t+1)}(x_c) = \phi_c^{(t)}(x_c)\frac{\tilde{p}(x_c)}{p^{(t)}(x_c)}$$

- Performs coordinate ascent in the likelihood of the MRF parameters, given the observed data.

Reference: unpublished notes by Michael Jordan

More general graphical models than MRF grids

- In this course, we've studied Markov chains, and Markov random fields, but, of course, many other structures of probabilistic models are possible and useful in computer vision.
- For a nice on-line tutorial about Bayes nets, see Kevin Murphy's tutorial in his web page.

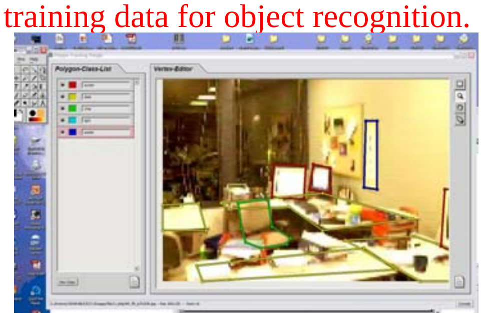
"Top-down" information: a representation for image context

Images

80-dimensional representation

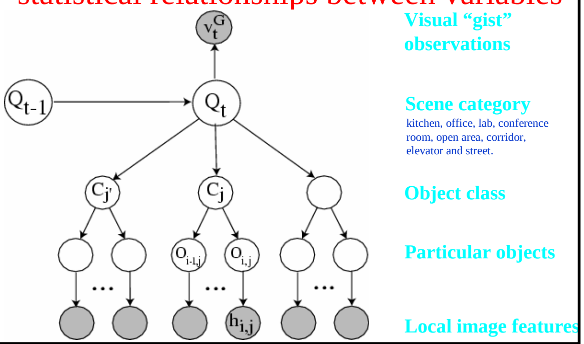
Credit: Antonio Torralba

"Bottom-up" information: labeled training data for object recognition.

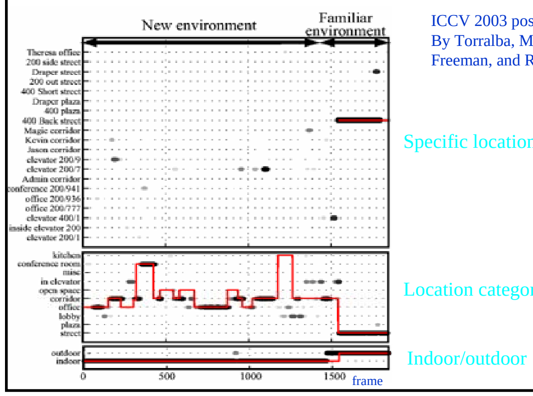


•Hand-annotated 1200 frames of video from a wearable webcam
•Trained detectors for 9 types of objects: bookshelf, desk, screen (frontal) , steps, building facade, etc.
•100-200 positive patches, > 10,000 negative patches

Combining top-down with bottom-up: graphical model showing assumed statistical relationships between variables

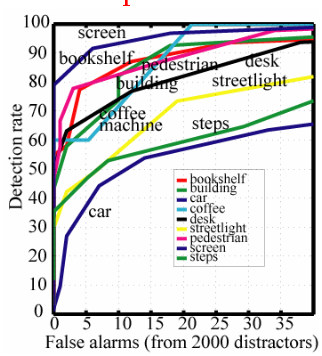


Categorization of new places



ICCV 2003 poster
By Torralba, Murphy, Freeman, and Rubin

Bottom-up detection: ROC curves



ICCV 2003 poster
By Torralba, Murphy, Freeman, and Rubin