Generative Models

Bill Freeman, MIT

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

6.869 March 15, 2005

What is the goal of vision?

If you are asking,
"Are there any faces in this image?",
then you would probably want to use discriminative methods.

What is the goal of vision?

If you are asking,
"Are there any faces in this image?",
then you would probably want to use discriminative methods.

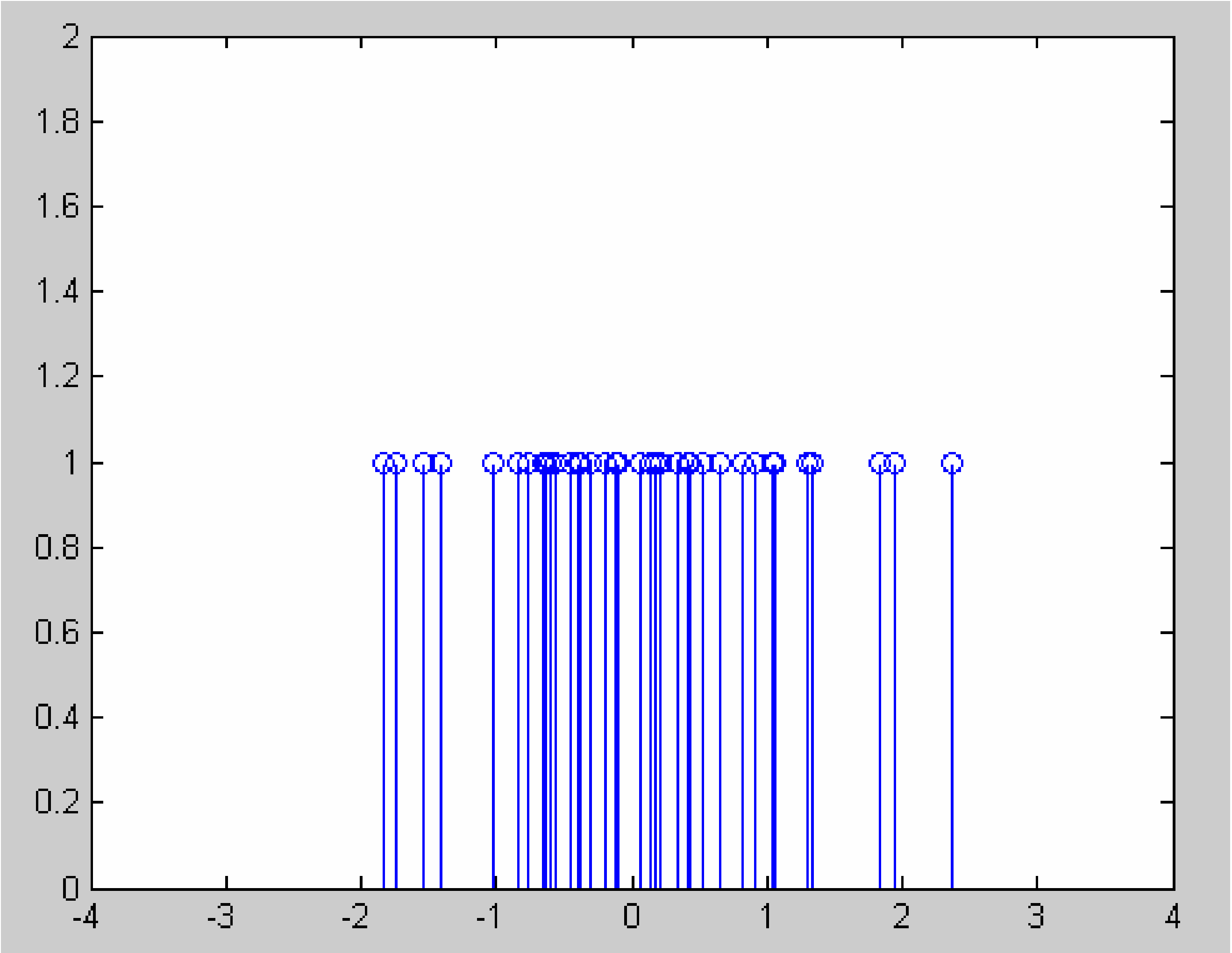
If you are asking,
"Find a 3-d model that describes the runner",
then you would use generative methods.

Modeling outline

(a) So we want to look at high-dimensional visual data, and fit models to it; forming summaries of it that let us understand what we see.

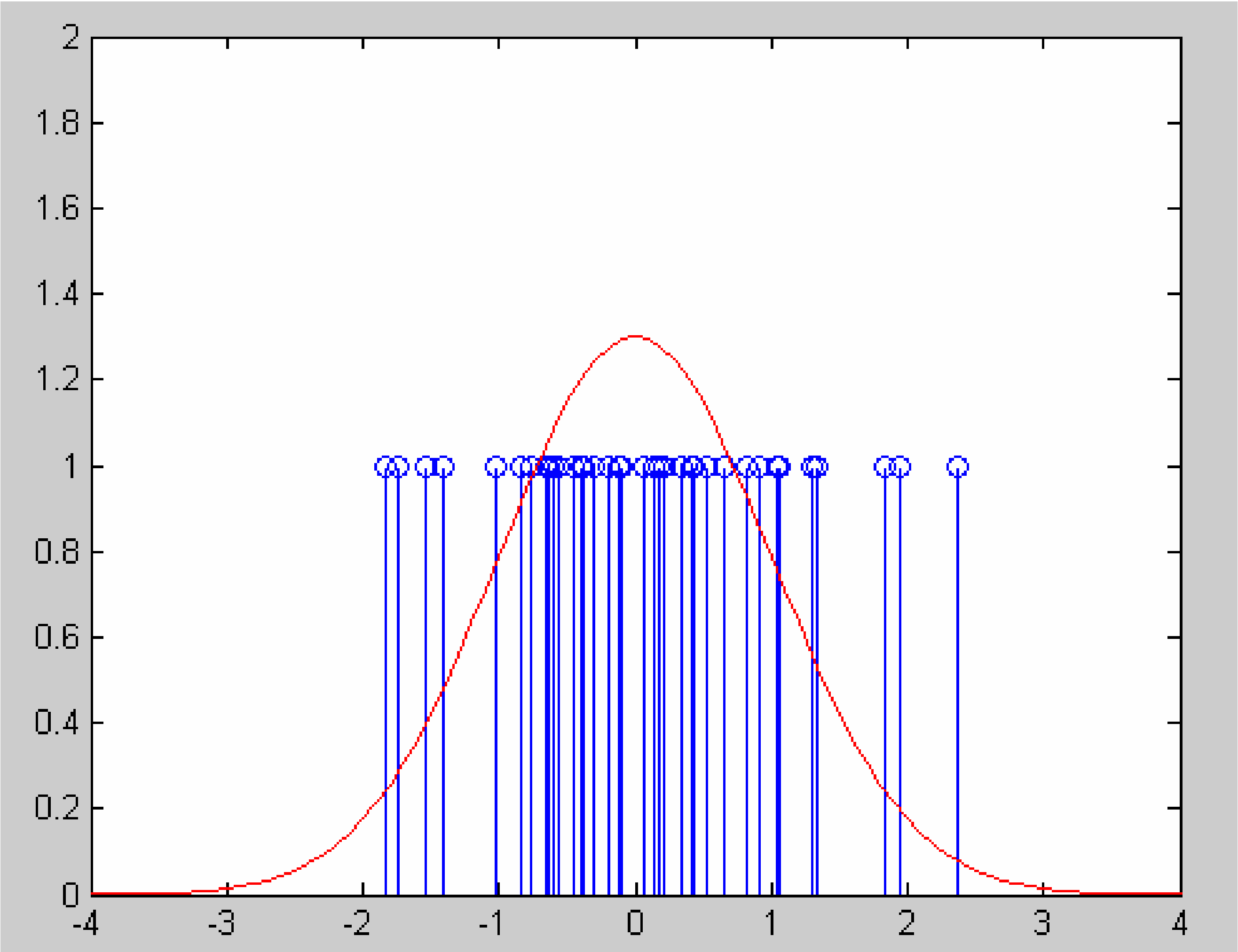
(b) After that, we'll look at ways to modularize the joint probability distribution.

The simplest data to model: a set of 1–d samples



Fit this distribution with a Gaussian

$$P(z_n|\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \frac{-(z_n-\mu)^2}{2\sigma^2}$$



How find the parameters of the best-fitting Gaussian?

Posterior probability

Likelihood function

Prior probability

$$P(\mu, \sigma | z) = \frac{P(z | \mu, \sigma) P(\mu, \sigma)}{P(z)}$$

mean

std. dev.

data points

Evidence

By Bayes rule

How find the parameters of the best-fitting Gaussian?

Posterior probability | Likelihood function | Prior probability

$$P(\mu, \sigma | z) = \frac{P(z|\mu,\sigma)P(\mu,\sigma)}{P(z)}$$

mean | std. dev. | data points | Evidence

Maximum likelihood parameter estimation:

$$\hat{\mu}, \hat{\sigma} = \text{argmax}_{\mu,\sigma} \; P(z|\mu, \sigma)$$

Derivation of MLE for Gaussians

Observation density

$$p(z_n|\mu,\sigma^2) \propto \frac{1}{\sigma} \exp - \frac{1}{2\sigma^2}(z_n - \mu)^2$$

$$L_n = \text{const} - \log \sigma - \frac{1}{2\sigma^2}(z_n - \mu)^2$$

Log likelihood

$$L = \sum_n L_n.$$

Maximisation

$$0 = \frac{\partial L}{\partial \mu} = \frac{1}{2\sigma^2} \sum_n (z_n - \mu)$$

$$0 = \frac{\partial L}{\partial \sigma} = \sum_n \left(-\frac{1}{\sigma} + \frac{1}{\sigma^3}(z_n - \mu)^2 \right)$$

Basic Maximum Likelihood Estimate (MLE) of a Gaussian distribution

$$0 = \frac{\partial L}{\partial \mu} = \frac{1}{2\sigma^2} \sum_n (z_n - \mu)$$

$$0 = \frac{\partial L}{\partial \sigma} = \sum_n \left(-\frac{1}{\sigma} + \frac{1}{\sigma^3}(z_n - \mu)^2 \right)$$

Mean

$$\hat{\mu} = m \equiv \frac{1}{N} \sum_{n=1}^{N} z_n$$

Variance

$$\hat{\sigma^2} = S \equiv \frac{1}{N} \sum_{n=1}^{N} (z_n - \mu)^2$$

Basic Maximum Likelihood Estimate (MLE) of a Gaussian distribution

Mean

$$\hat{\mu} = m \equiv \frac{1}{N} \sum_{n=1}^{N} z_n$$

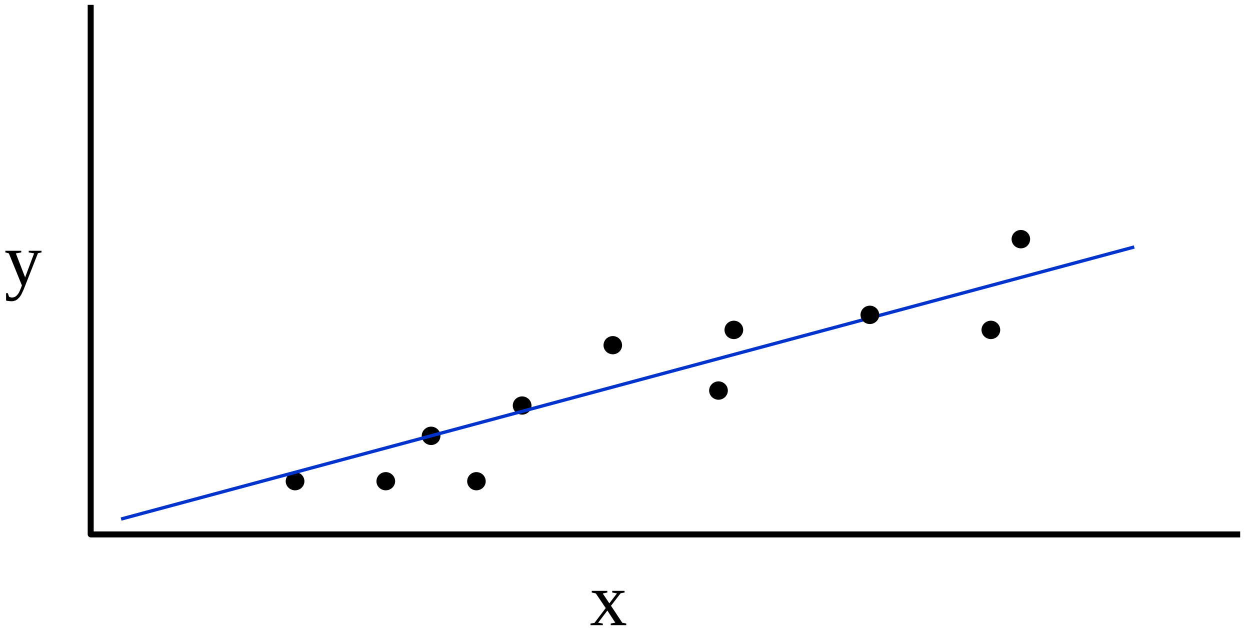
Variance

$$\hat{\sigma}^2 = S \equiv \frac{1}{N} \sum_{n=1}^{N} (z_n - \mu)^2$$

For vector-valued data, we have the Covariance Matrix

$$\hat{P} = S \equiv \frac{1}{N} \sum_{n=1}^{N} (z_n - \mu)(z_n - \mu)^\top$$

Model fitting example 2:
Fit a line to observed data



Maximum likelihood estimation for the slope of a single line

data: $(X_n, Y_n), n = 1 \ldots, N$

model: $Y = aX + w$

where $w \sim N(\mu = 0, \sigma = 1)$.

Data likelihood for point n:

$$P(X_n, Y_n | a) = c \exp[-(Y_n - aX_n)^2/2]$$

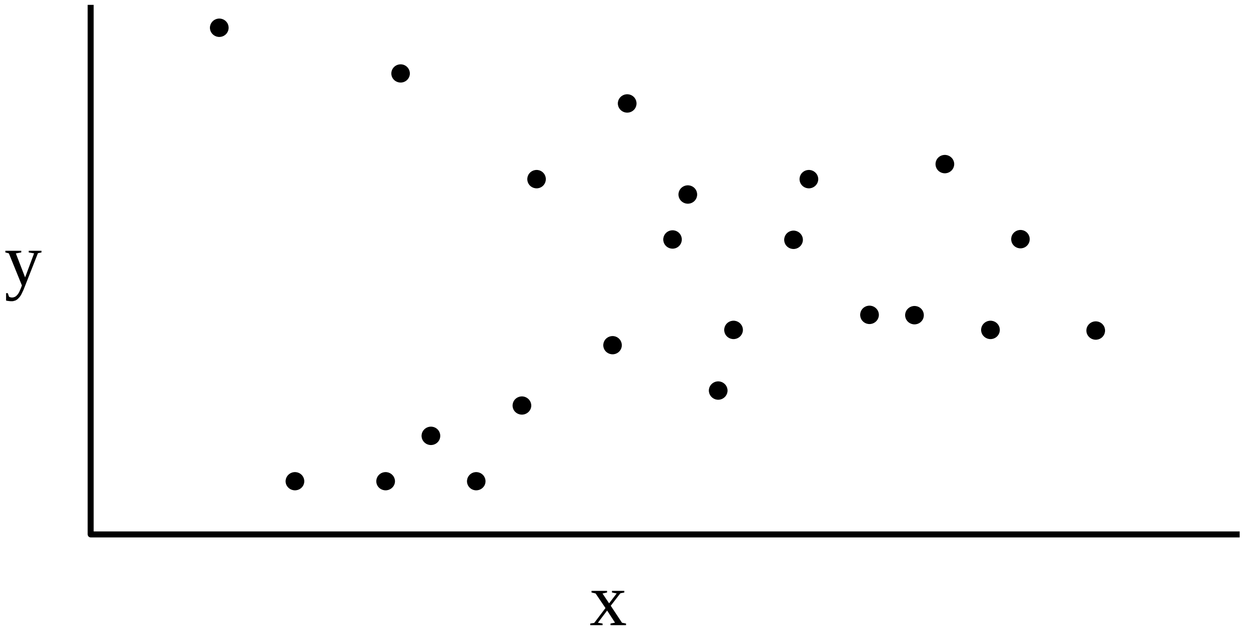
Maximum likelihood estimate:

$$\hat{a} = \arg\max_a p(Y_1, \ldots, Y_n | a) = \arg\max_a \sum_n -d(Y_n; a)^2/2$$

where $d(Y_n; a) = |Y_n - aX_n|$

gives regression formula $\hat{a} = \frac{\sum_n Y_n X_n}{\sum_n X_n^2}$.

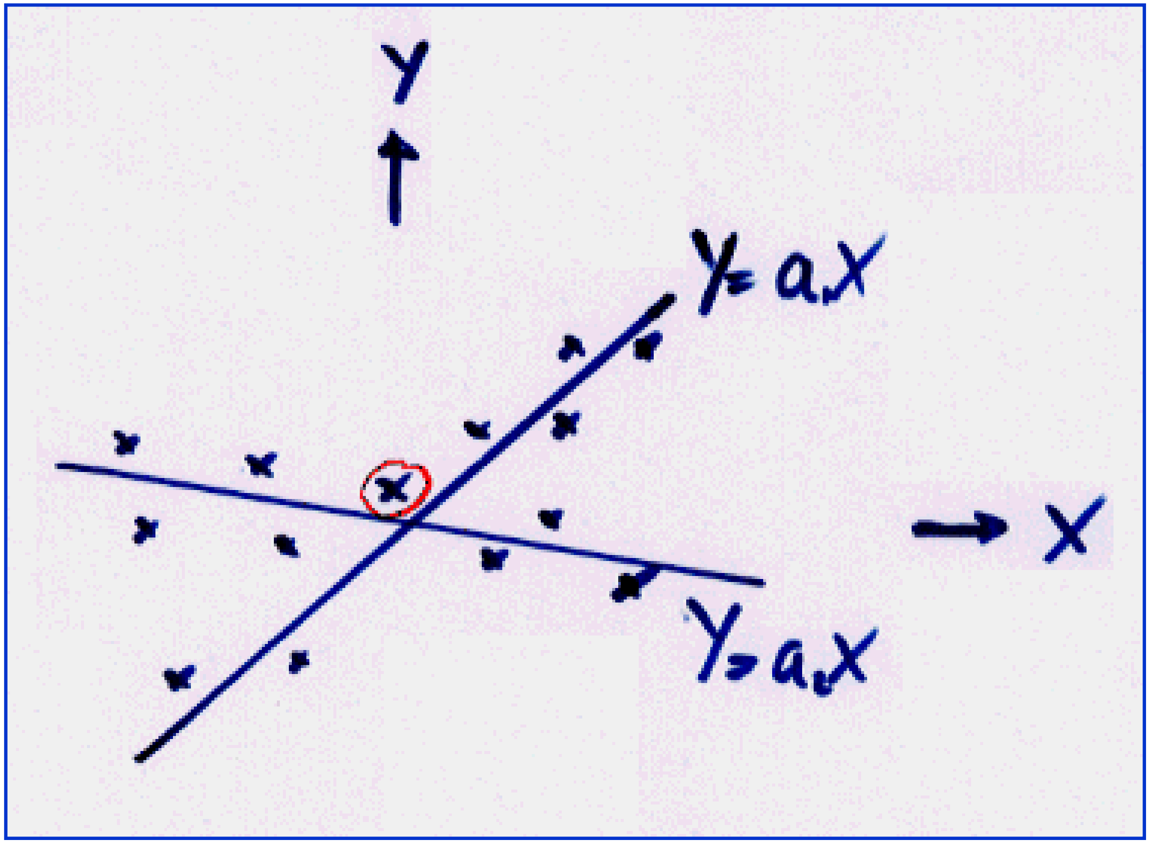
Model fitting example 3:
Fitting two lines to observed data



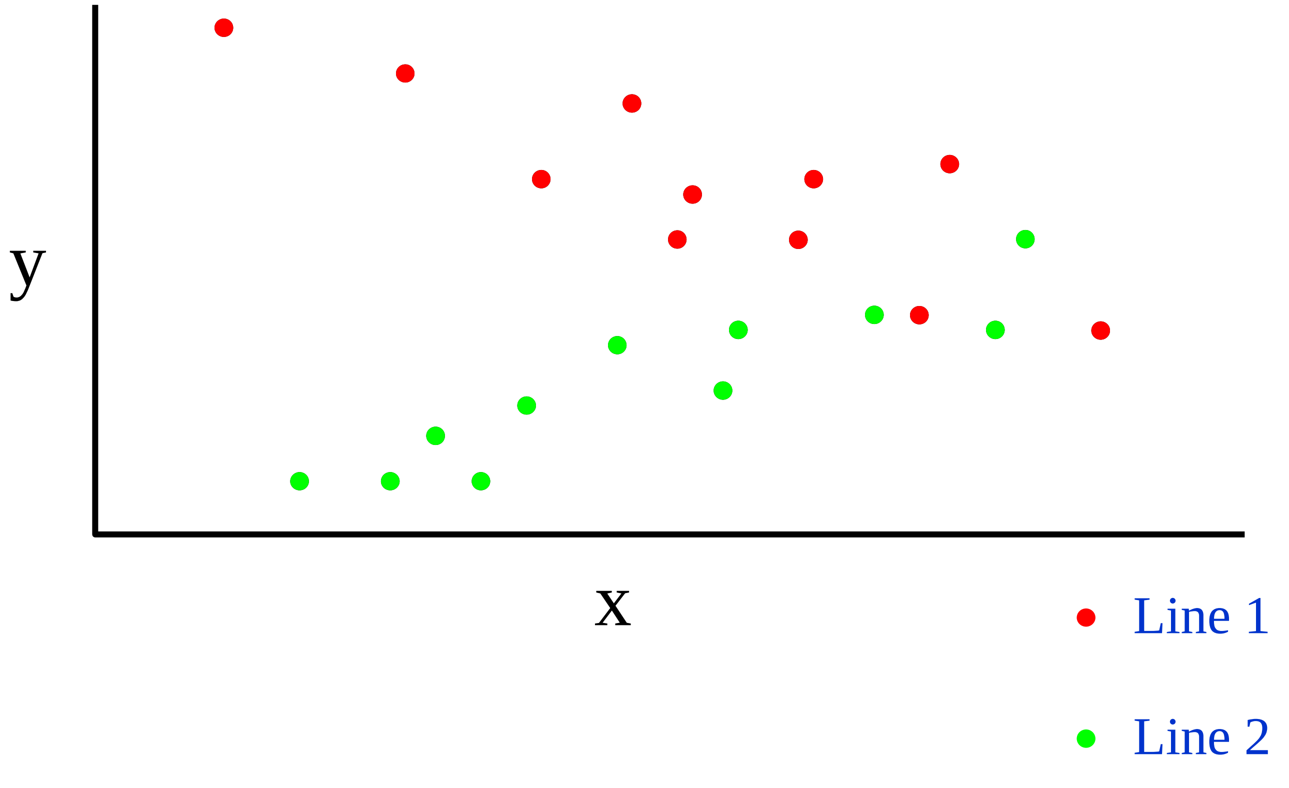
MLE for fitting a line pair

Lines $Y = a_1 X + w$ or $Y = a_2 X + w$, with $w \sim \mathcal{N}(0,1)$.

(a form of mixture dist. for $\boldsymbol{Y}$)

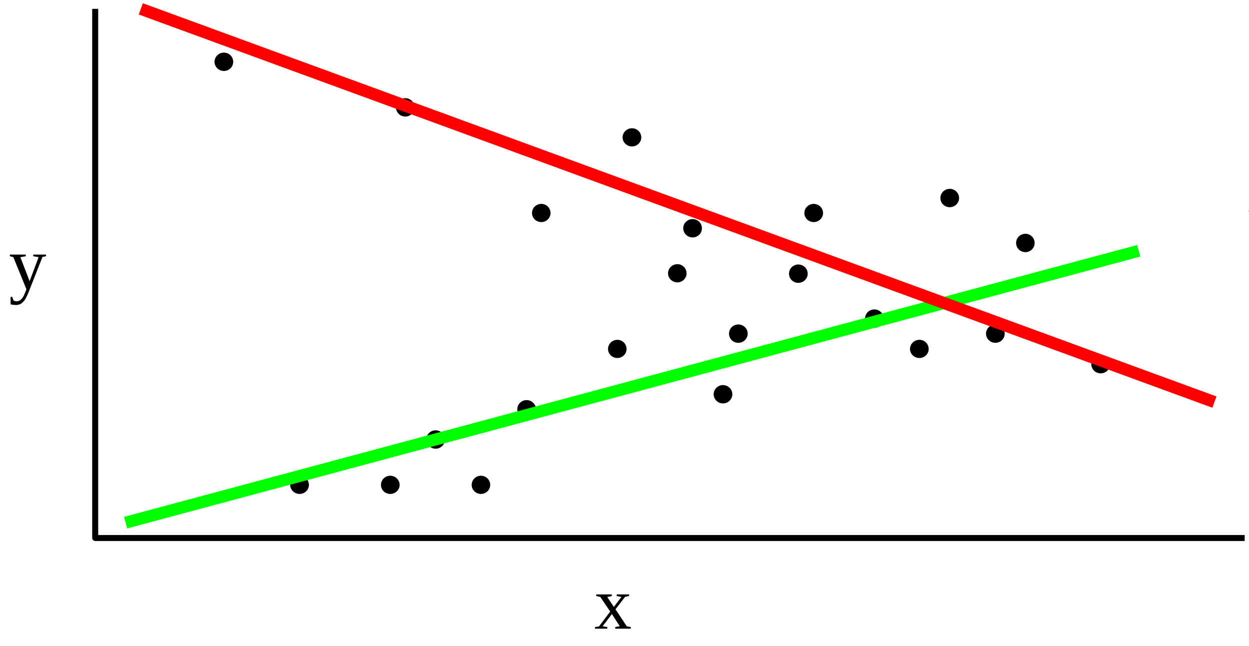


Fitting two lines: on the one hand…



If we knew which points went with which lines, we'd be back at the single line-fitting problem, twice.

Fitting two lines, on the other hand…



We could figure out the probability that any point came from either line if we just knew the two equations for the two lines.

Expectation Maximization (EM): a solution to chicken-and-egg problems

MLE with hidden/latent variables: Expectation Maximisation

General problem:

$$y = (Y_1, \ldots, Y_N); \ \theta = (a_1, a_2); \ z = (z_1, \ldots, z_N)$$

data parameters hidden variables

For MLE, want to maximise the log likelihood

$$\widehat{\theta} = \text{arg max}_\theta \log p(y|\theta)$$

$$= \text{arg max}_\theta \log \textstyle\sum_z p(y, z|\theta)$$

The sum over z inside the log gives a complicated expression for the ML solution.

The EM algorithm

We don't know the values of the labels, z_i , but let's use its expected value under its posterior with the current parameter values, θ_{old}. That gives us the "expectation step":

"E-step" $$Q(\theta; \theta_{old}) = \sum_z p(z|y, \theta_{old}) \log p(y|z\ \theta)$$

Now let's maximize this Q function, an expected log-likelihood, over the parameter values, giving the "maximization step":

"M-step" $$\theta_{new} = \arg\ \max_\theta Q(\theta; \theta_{old})$$

Each iteration increases the total log-likelihood log $p(y|\theta)$

Expectation Maximisation applied to fitting the two lines

Hidden variables $z_n = i$ associate data point n with line i

and probabilities of association are $w_i(n), \ i = 1, 2,$:

Need:

$$w_i(n) = p(z_n = i|y, \theta) \propto p(y|z_n = i, \theta) \propto \exp[-d(Y_n; a_i)^2/2]$$

and then:

$$Q(y, \theta, \theta_{old}) = \sum_n -\frac{1}{2}\left(w_1(n)d(Y_n; a_1)^2 + w_2(n)d(Y_n; a_2)^2\right)$$

and maximising that gives

$$\hat{a}_i = \frac{\sum_n w_i(n) Y_n X_n}{\sum_n w_i(n) X_n^2}.$$

EM fitting to two lines

with $w_i(n) \propto \exp -d(Y_n; a_i)^2/2$

and $w_1(n) + w_2(n) = 1$

“E-step”

↑

repeat

↓

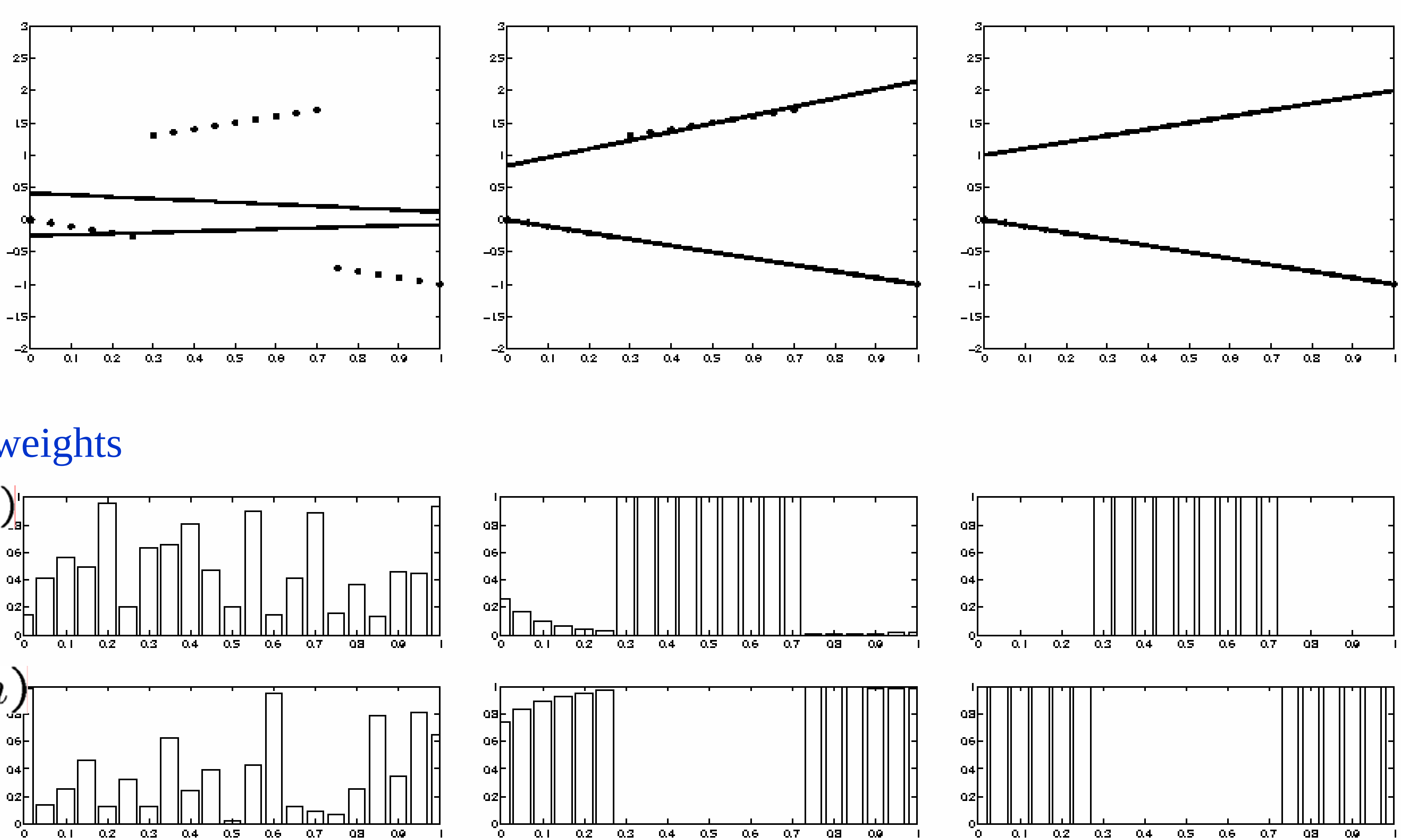
Regression becomes:

$$\hat{a}_i = \frac{\sum_n w_i(n) Y_n X_n}{\sum_n w_i(n) X_n^2}.$$

“M-step”

Experiments: EM fitting to two lines

(from a tutorial by Yair Weiss, http://www.cs.huji.ac.il/~yweiss/tutorials.html)



Line weights

$w_1(n)$ line 1

$w_2(n)$ line 2

Iteration 1 2 3

Applications of EM in computer vision

- Structure-from-motion with multiple moving objects
- Motion estimation combined with perceptual grouping
- Multiple layers/or sprites in an image.

Modeling outline

(a) So we want to look at high-dimensional visual data, and fit models to it; forming summaries of it that let us understand what we see.

(b) After that, we'll look at ways to modularize the joint probability distribution.

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?

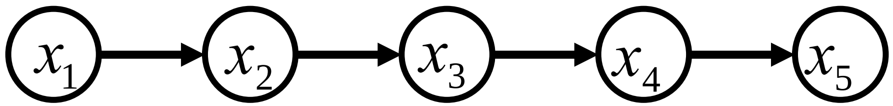
Two reasonable choices are: (a) find the value of x1 (and of all the other variables) that gives the maximum of P(x1, x2, x3, x4, x5); that's the MAP solution.
Or (b) marginalize over all the other variables and then take the mean or the maximum of the other variables. Marginalizing, then taking the 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:



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

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

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

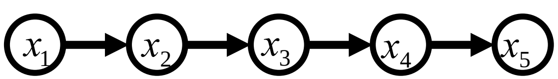
$$= P(x_1)P(x_2 \mid x_1)P(x_3, x_4, x_5 \mid x_1, x_2)$$

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

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

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

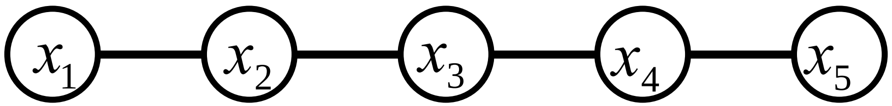
Performing the marginalization by doing the partial sums is called "belief propagation".

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

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:



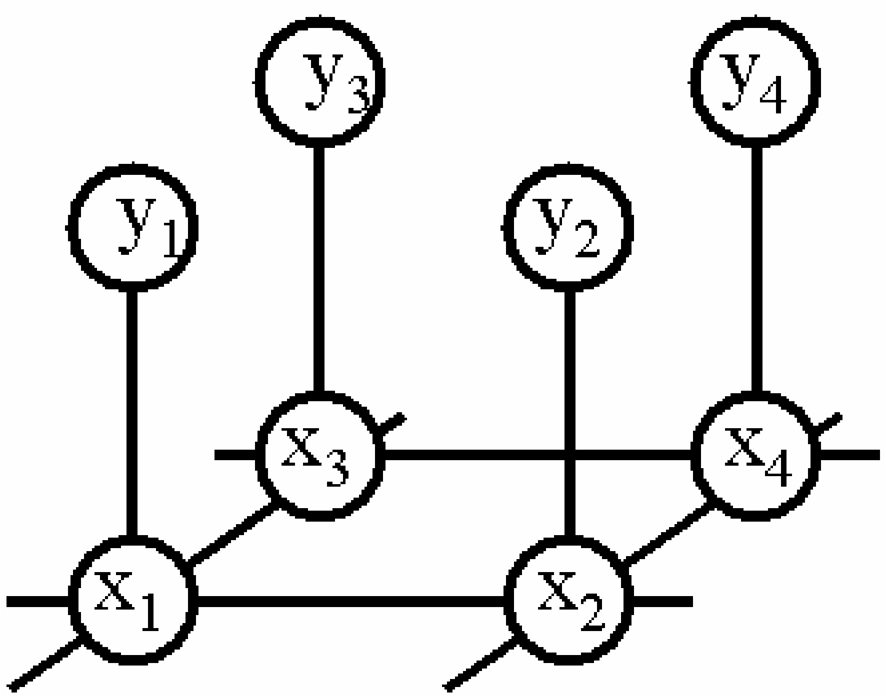
The joint probability for this graph is given by:

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

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

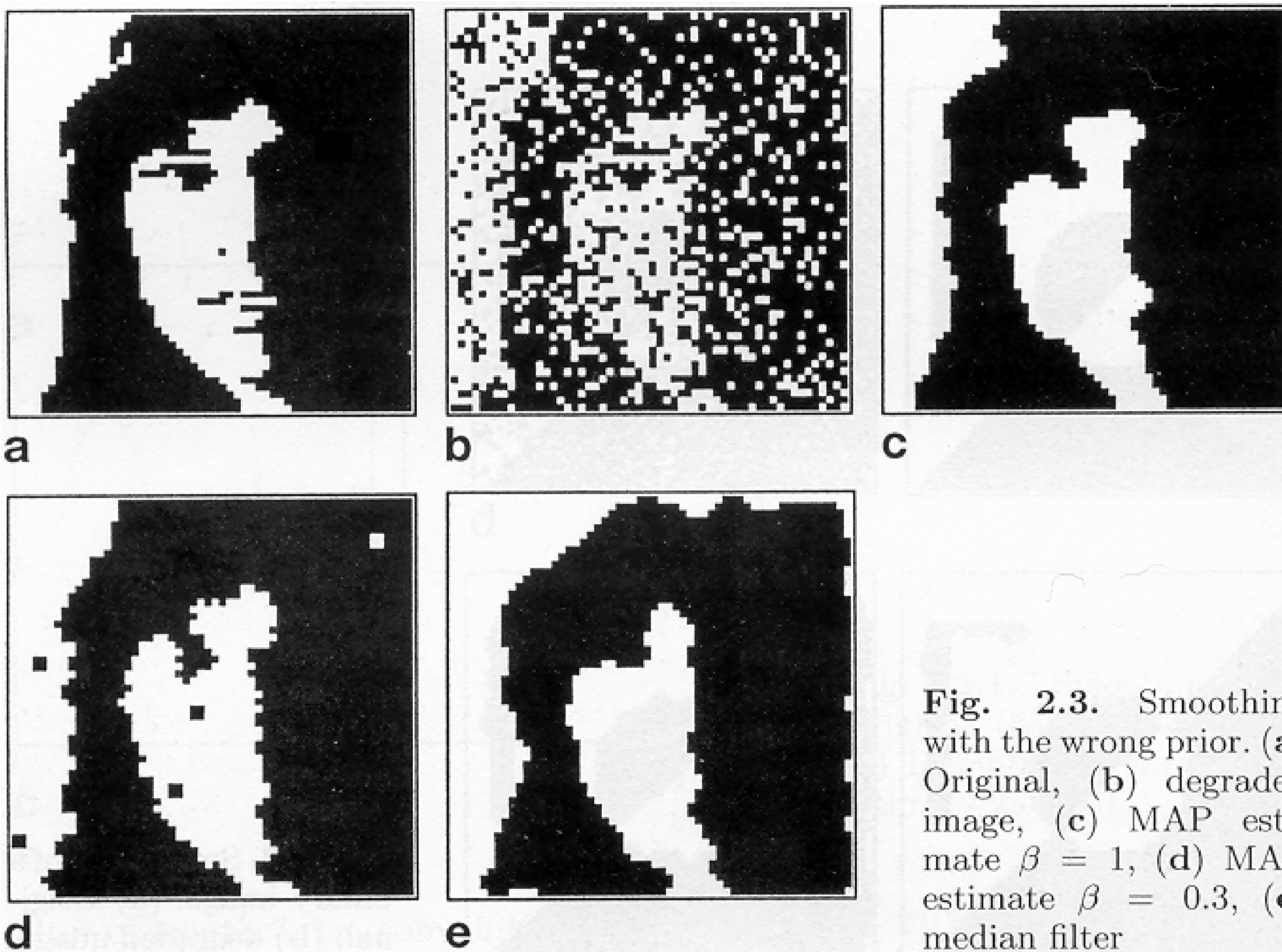
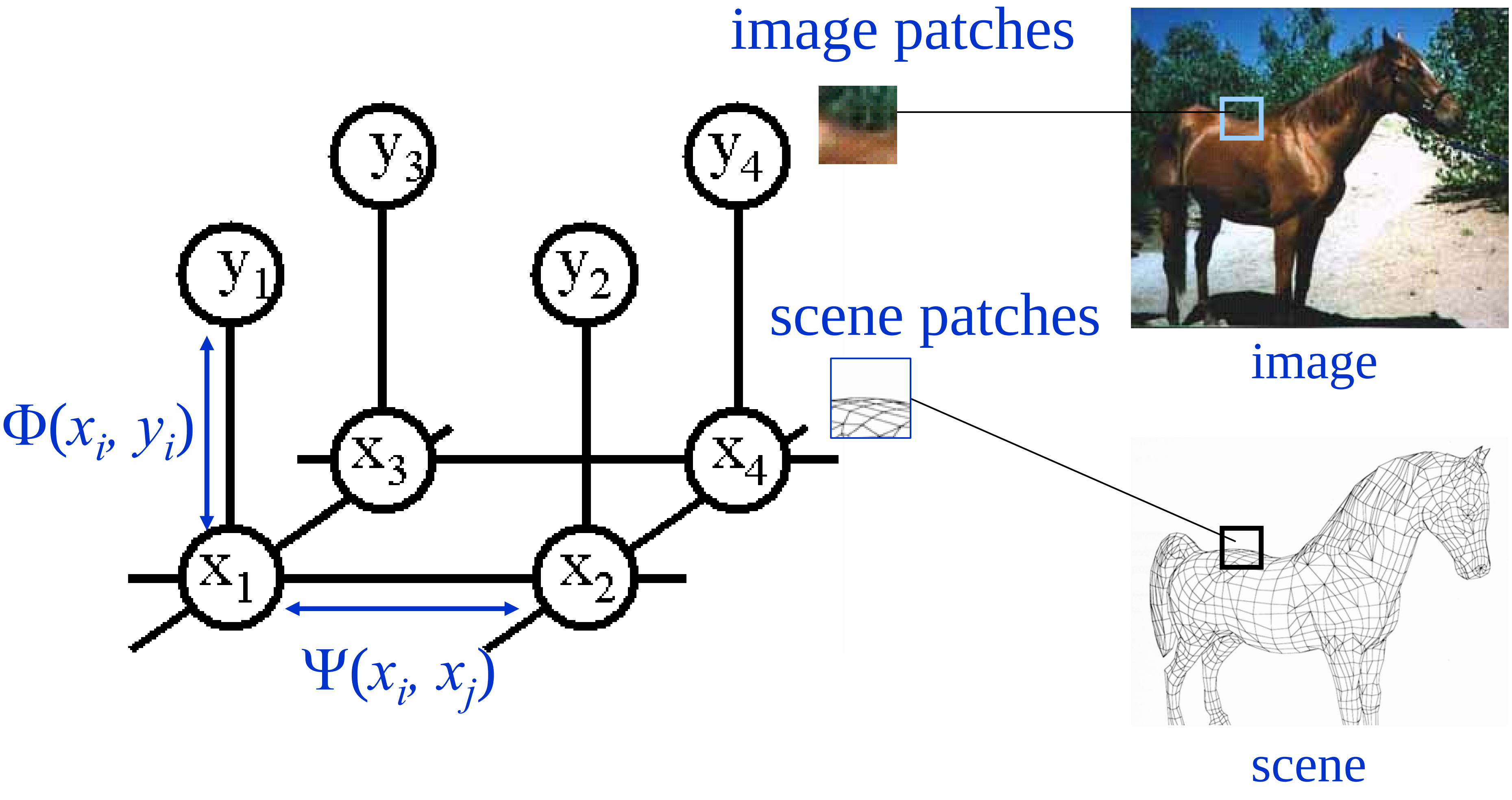


Fig. 2.3. Smoothing with the wrong prior. (**a**) Original, (**b**) degraded image, (**c**) MAP estimate $\beta = 1$, (**d**) MAP estimate $\beta = 0.3$, (**e**) median filter

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)

Gibbs Sampling and Simulated Annealing

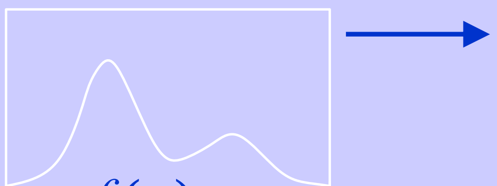
- Gibbs sampling:
  - A way to generate random samples from a (potentially very complicated) probability distribution.
- Simulated annealing:
  - A schedule for modifying the probability distribution so that, at "zero temperature", you draw samples only from the MAP solution.

$$P(x) = \frac{1}{Z} \exp(-E(x)/kT)$$

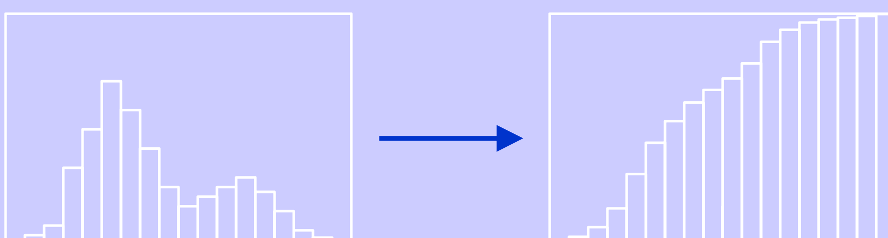
Reference: Geman and Geman, IEEE PAMI 1984.

Sampling from a 1-d function

1. Discretize the density function



$f(x)$ $\quad f(k)$



$f(k)$ $\quad F(k)$

2. Compute distribution function from density function

3. Sampling

draw $\alpha \sim$ U(0,1);
for k = 1 to n
  if $F(k) \geq \alpha$
    break;
  $x = x_0 + k\tau$;

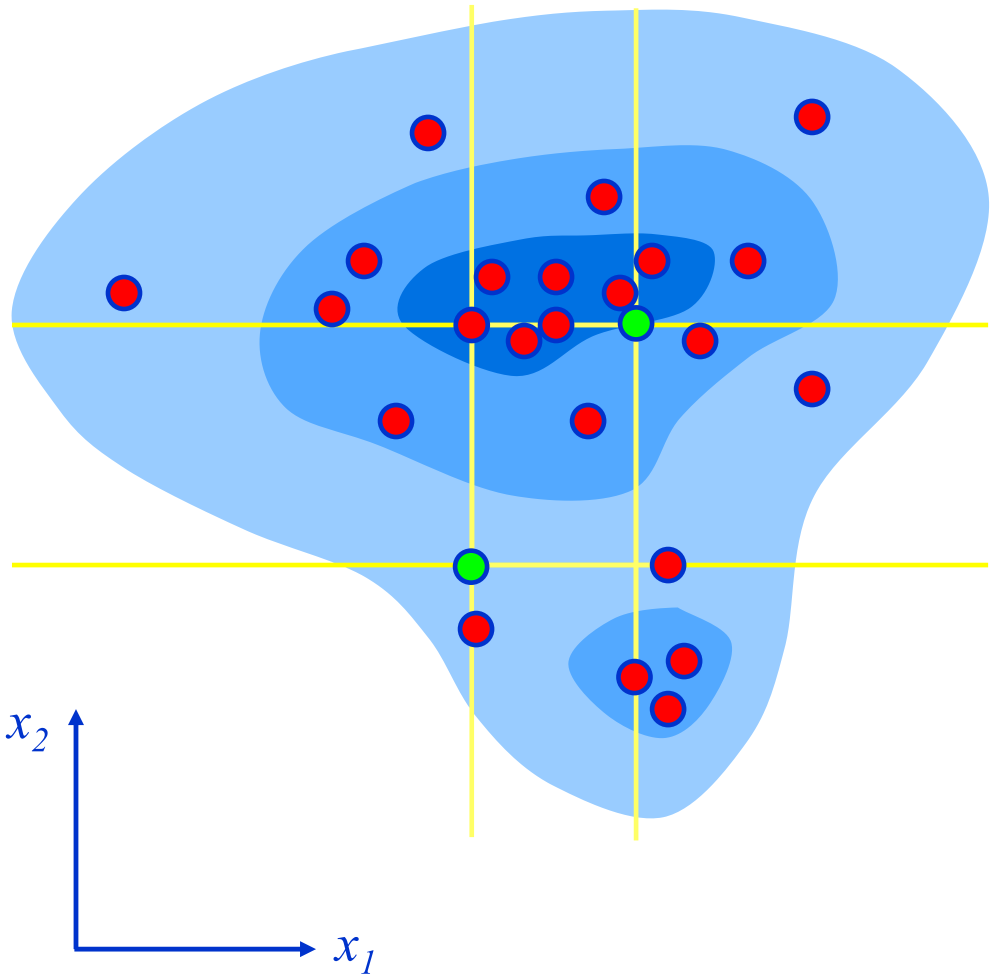
Gibbs Sampling

$$x_1^{(t+1)} \sim \pi(x_1 \mid x_2^{(t)}, x_3^{(t)}, \cdots, x_K^{(t)})$$

$$x_2^{(t+1)} \sim \pi(x_2 \mid x_1^{(t+1)}, x_3^{(t)}, \cdots, x_K^{(t)})$$

$$\vdots$$

$$x_K^{(t+1)} \sim \pi(x_K \mid x_1^{(t+1)}, \cdots, x_{K-1}^{(t+1)})$$



Slide by Ce Liu

Gibbs sampling and simulated annealing

Simulated annealing as you gradually lower the “temperature” of the probability distribution ultimately giving zero probability to all but the MAP estimate.

<u>What’s good about it</u>: finds global MAP solution.

<u>What’s bad about it</u>: takes forever. Gibbs sampling is in the inner loop…

Gibbs sampling and simulated annealing

So you can find the mean value (MMSE estimate) of a variable by doing Gibbs sampling and averaging over the values that come out of your sampler.

You can find the MAP value of a variable by doing Gibbs sampling and gradually lowering the temperature parameter to zero.

Iterated conditional modes

- For each node:
  - Condition on all the neighbors
  - Find the mode
  - Repeat.

Described in: Winkler, 1995. Introduced by Besag in 1986.

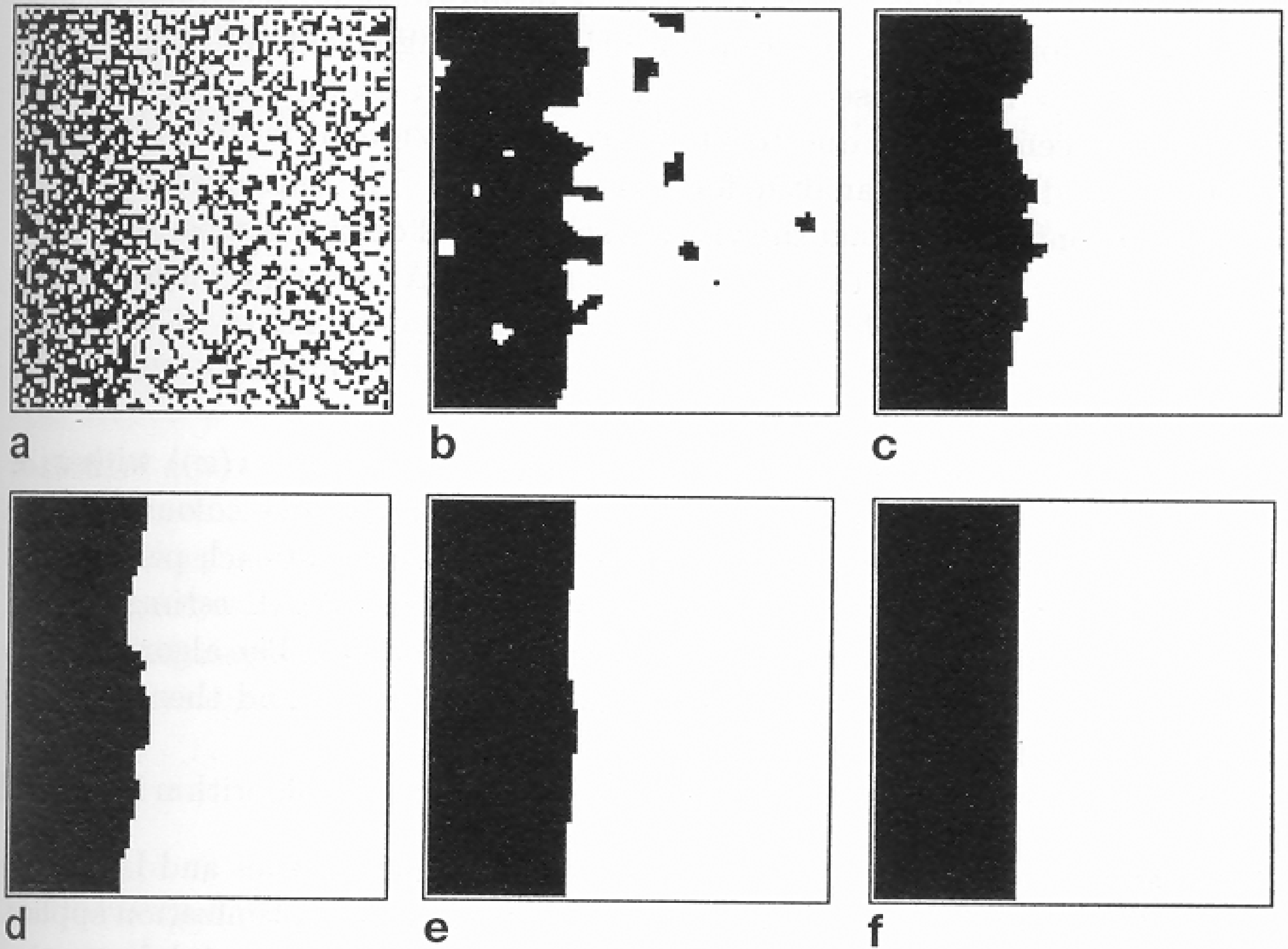


Fig. 6.2. Various steps of ICM

Winkler, 1995

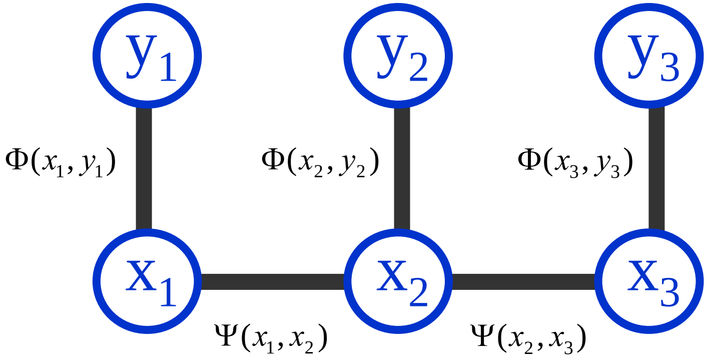
Variational methods

- 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

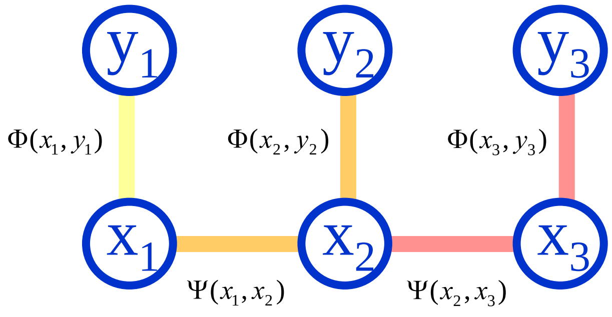


$$x_{1MMSE} = \underset{x_1}{\text{mean}} \; \underset{x_2}{\text{sum}} \; \underset{x_3}{\text{sum}} \; P(x_1, x_2, x_3, y_1, y_2, y_3)$$

The posterior factorizes

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

$$= \operatorname*{mean}_{x_1} \operatorname*{sum}_{x_2} \operatorname*{sum}_{x_3} \Phi(x_1, y_1) \Phi(x_2, y_2) \Psi(x_1, x_2) \Phi(x_3, y_3) \Psi(x_2, x_3)$$

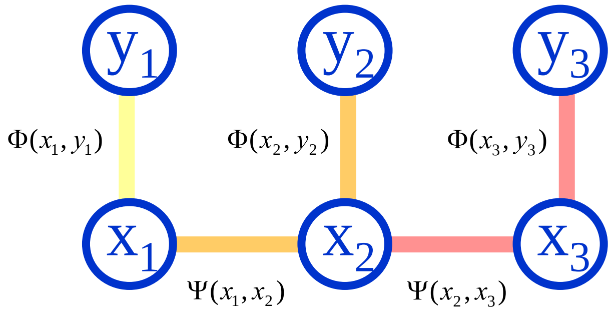


Propagation rules

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

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

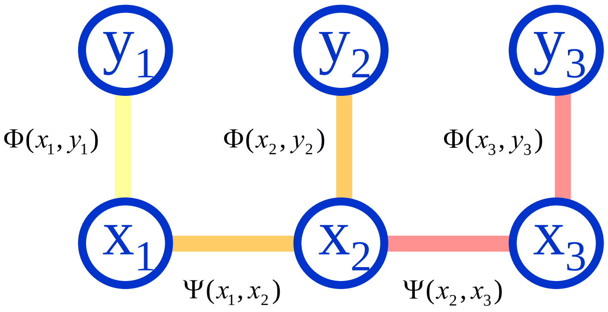
$$x_{1MMSE} = \underset{x_1}{\text{mean}} \; \Phi(x_1, y_1) \; \underset{x_2}{\text{sum}} \; \Phi(x_2, y_2) \, \Psi(x_1, x_2) \; \underset{x_3}{\text{sum}} \; \Phi(x_3, y_3) \, \Psi(x_2, x_3)$$



Propagation rules

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

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



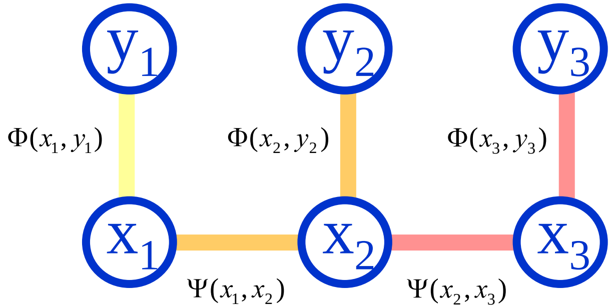
Propagation rules

$$x_{1MMSE} = \underset{x_1}{\text{mean}}\, \Phi(x_1, y_1)$$

$$\underset{x_2}{\text{sum}}\, \Phi(x_2, y_2)\, \Psi(x_1, x_2)$$

$$\underset{x_3}{\text{sum}}\, \Phi(x_3, y_3)\, \Psi(x_2, x_3)$$

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



Belief propagation: the nosey neighbor rule

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

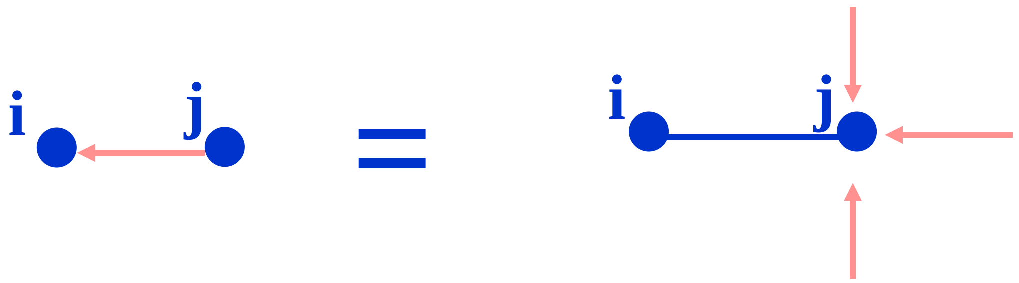
(Given the probabilities of my being in different states, and how my states relate to your states, here's what I think the probabilities of your states should be)

Belief propagation messages

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

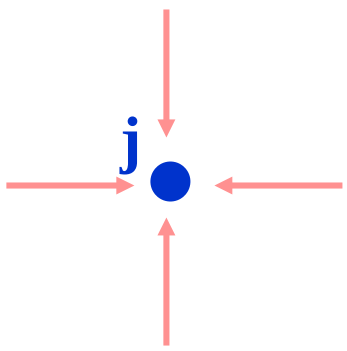
To send a message: Multiply together all the incoming messages, except from the node you're sending to, then multiply by the compatibility matrix and marginalize over the sender's states.

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



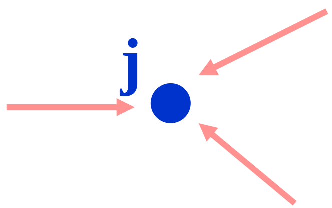
Beliefs

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



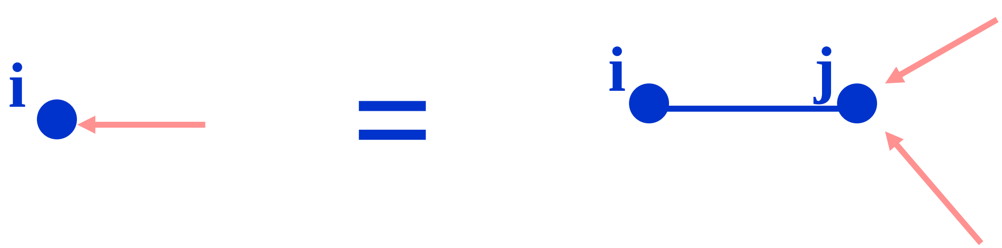
$$b_j(x_j) = \prod_{k \in N(j)} M_j^k(x_j)$$

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

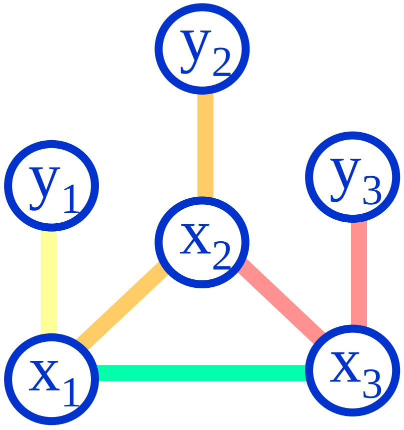
- “Do the right thing” Bayesian algorithm.
- For Gaussian random variables over time: Kalman filter.
- For hidden Markov models: forward/backward algorithm (and MAP variant is Viterbi).

No factorization with loops!

$$x_{1MMSE} = \underset{x_1}{\text{mean}}\, \Phi(x_1, y_1)$$

$$\underset{x_2}{\text{sum}}\, \Phi(x_2, y_2)\, \Psi(x_1, x_2)$$

$$\underset{x_3}{\text{sum}}\, \Phi(x_3, y_3)\, \Psi(x_2, x_3) \quad \Psi(x_1, x_3)$$



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: $F[p(x_1, x_2, ..., x_N)]$

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

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

Kikuchi Approximations:

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

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 $\lambda_{ij}(x_j)$

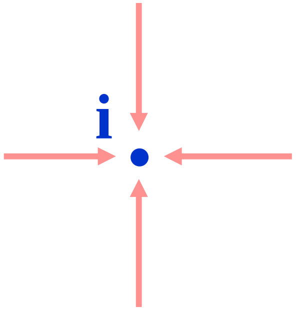
enforce the constraints $b_j(x_j) = \sum_{x_i} b_{ij}(x_i, x_j)$

Bethe stationary conditions = message update rules

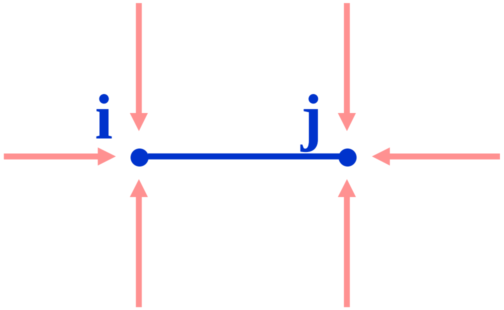
with $$\lambda_{ij}(x_j) = T \ln \prod_{k \in N(j) \backslash i} M_j^k(x_j)$$

Region marginal probabilities

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

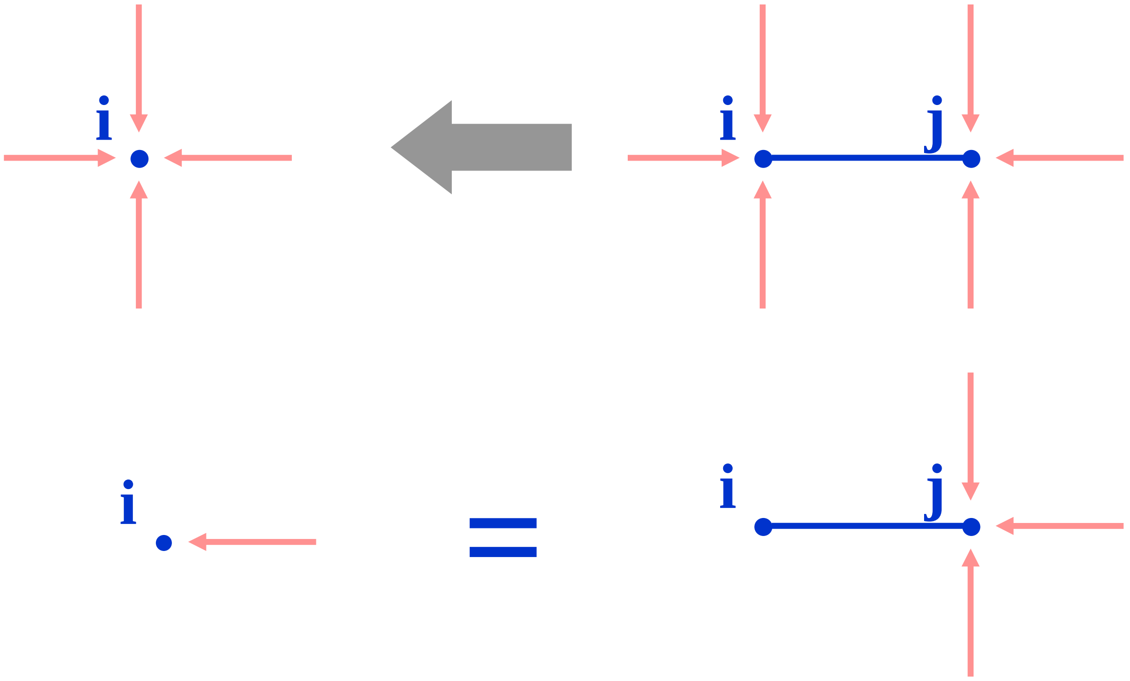


$$b_{ij}(x_i,x_j) = k\,\Psi(x_i,x_j)\prod_{k\in N(i)\backslash j} M_i^k(x_i)\prod_{k\in N(j)\backslash i} M_j^k(x_j)$$



Belief propagation equations

Belief propagation equations come from the marginalization constraints.



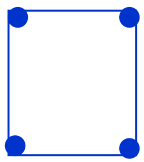
$$M_i^j(x_i) = \sum_{x_j} \psi_{\mathrm{ij}}(x_i, x_j) \prod_{k \in N(j) \setminus i} M_j^k(x_j)$$

Results from Bethe free energy analysis

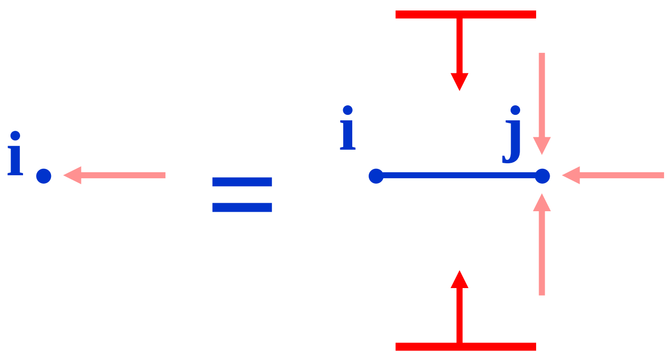
- Fixed point of belief propagation equations iff. Bethe approximation stationary point.
- Belief propagation always has a fixed point.
- Connection with variational methods for inference: both minimize approximations to Free Energy,
  - **variational**: usually use primal variables.
  - **belief propagation**: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

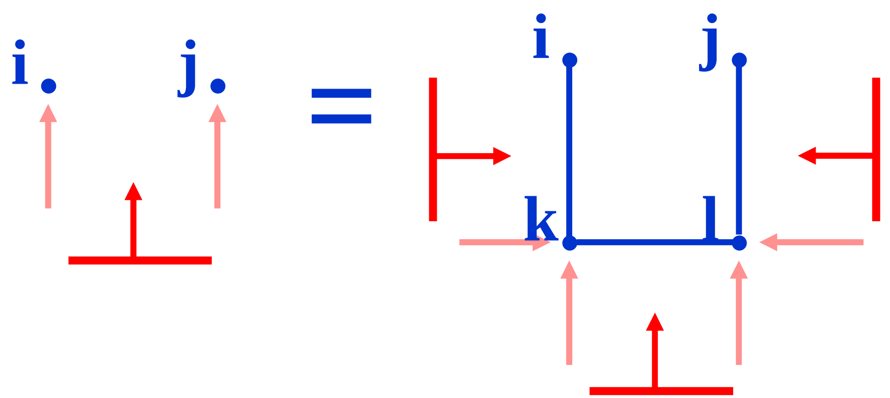
Kikuchi message-update rules

Groups of nodes send messages to other groups of nodes.

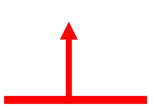


Typical choice for Kikuchi cluster.

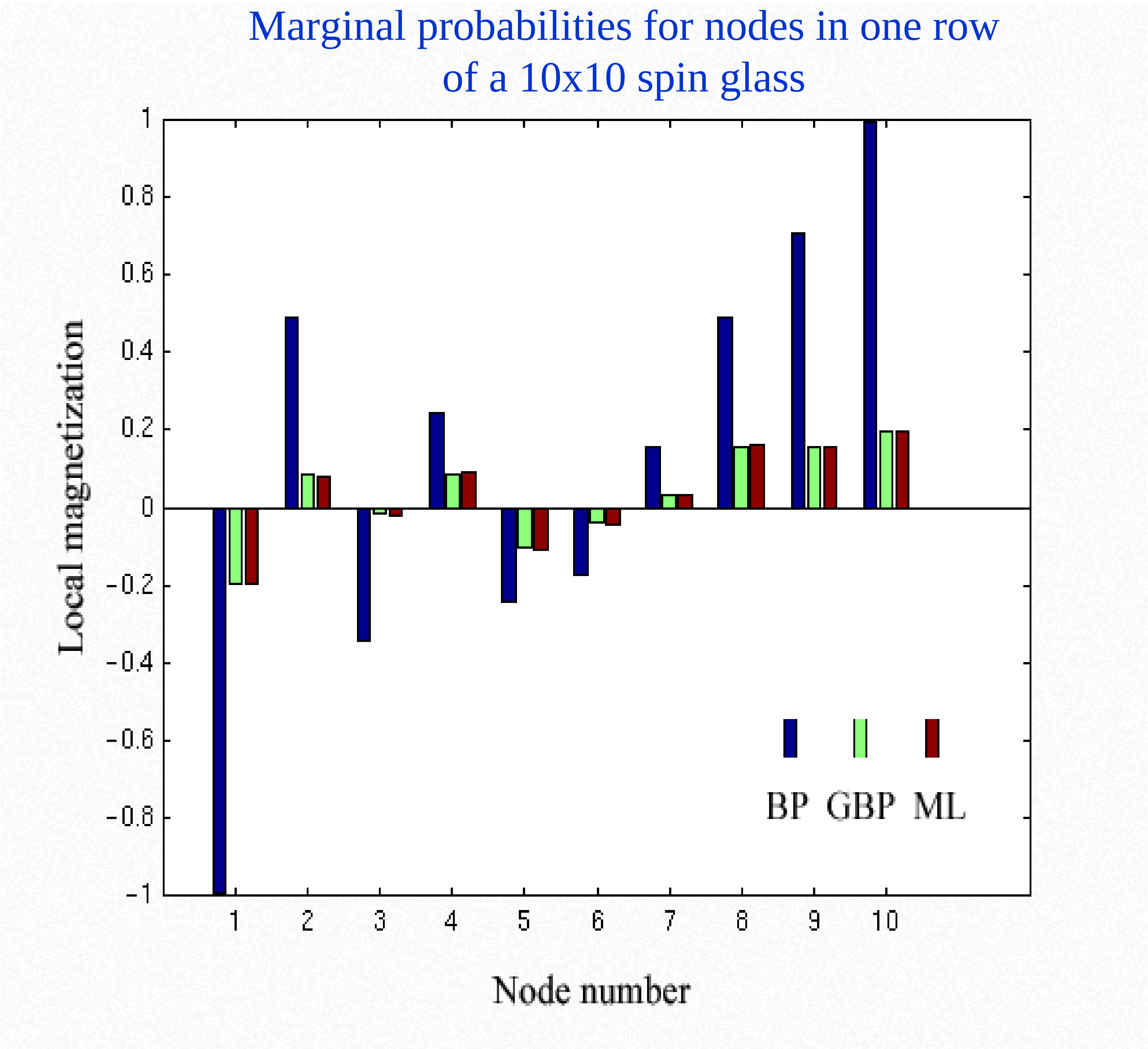




Update for messages 

Update for messages 

Generalized belief propagation



References on BP and GBP

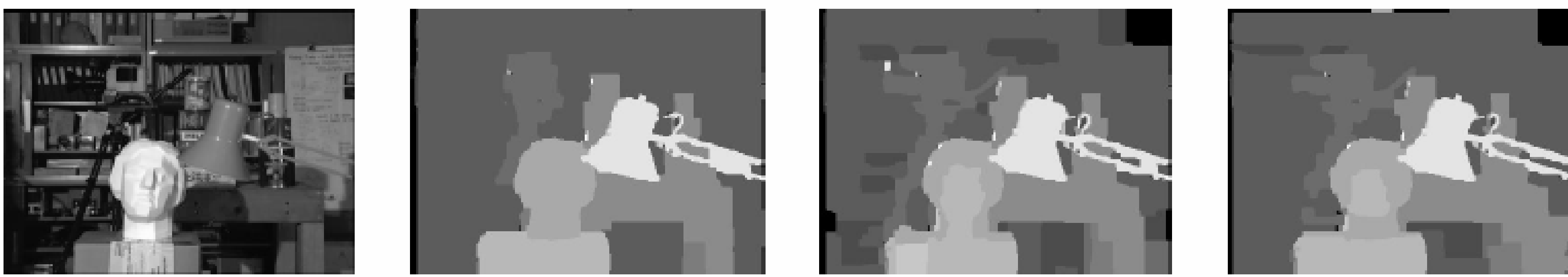
- J. Pearl, 1985
  - classic
- Y. Weiss, NIPS 1998
  - Inspires application of BP to vision
- W. Freeman et al learning low-level vision, IJCV 1999
  - Applications in super-resolution, motion, shading/paint discrimination
- H. Shum et al, ECCV 2002
  - Application to stereo
- M. Wainwright, T. Jaakkola, 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 ($\times 10^3$)			% Energy from Occluded Matching Costs
	Ground-Truth	Graph Cuts	Synchronous Belief Prop	
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)

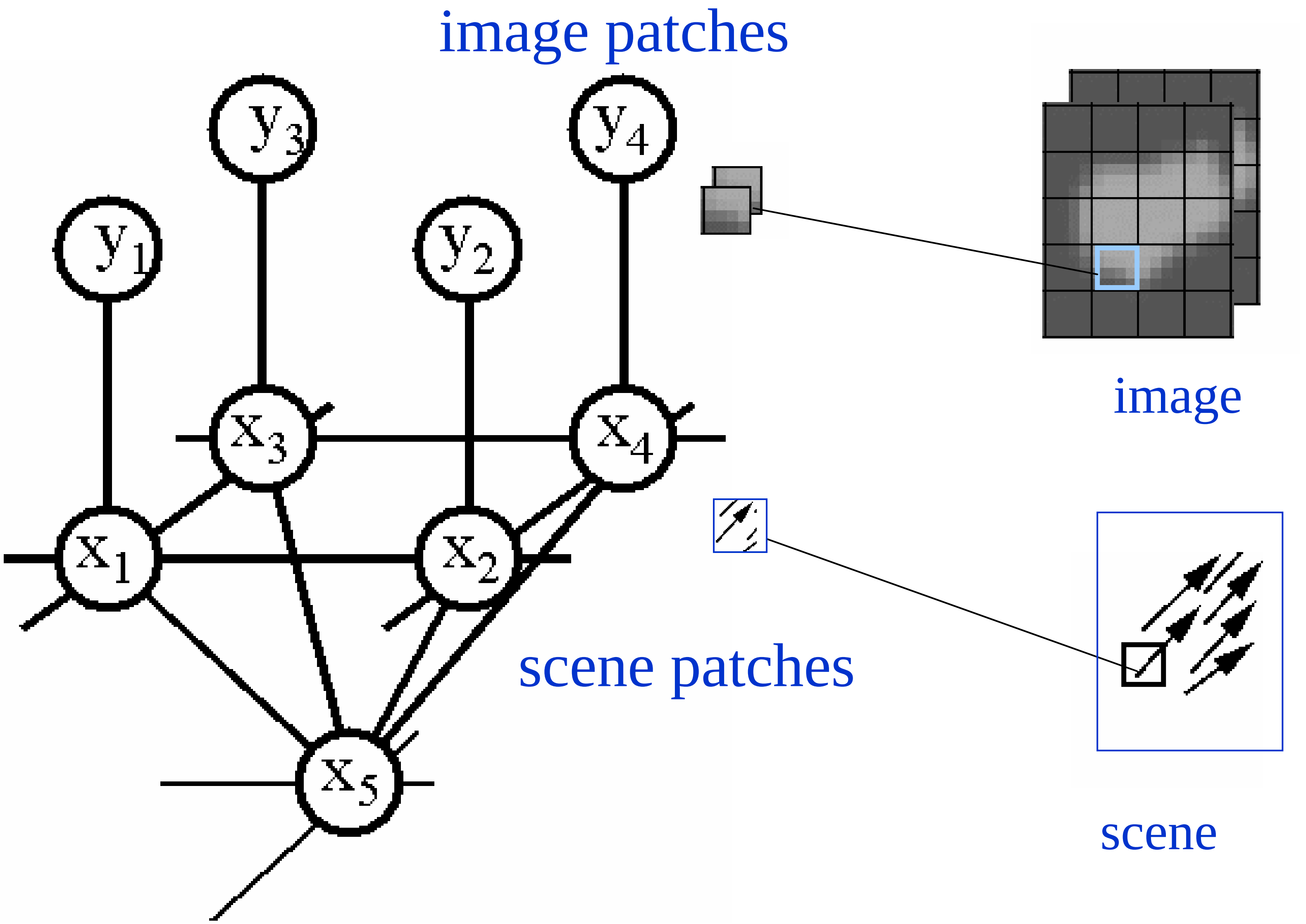
Vision applications of MRF’s

- Stereo
- Motion estimation
- Labelling shading and reflectance
- Many others…

Vision applications of MRF's

- Stereo
- Motion estimation
- Labelling shading and reflectance
- Many others…

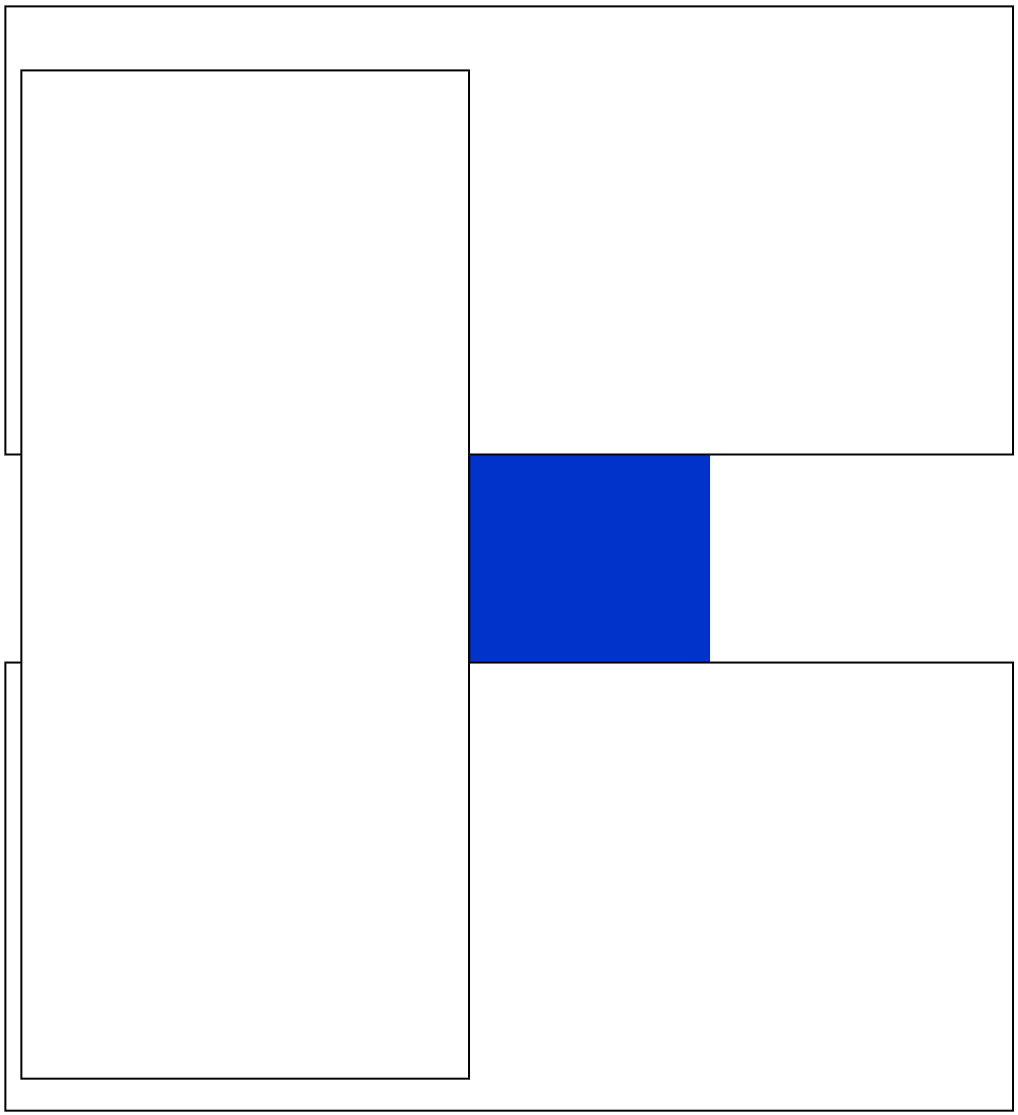
Motion application



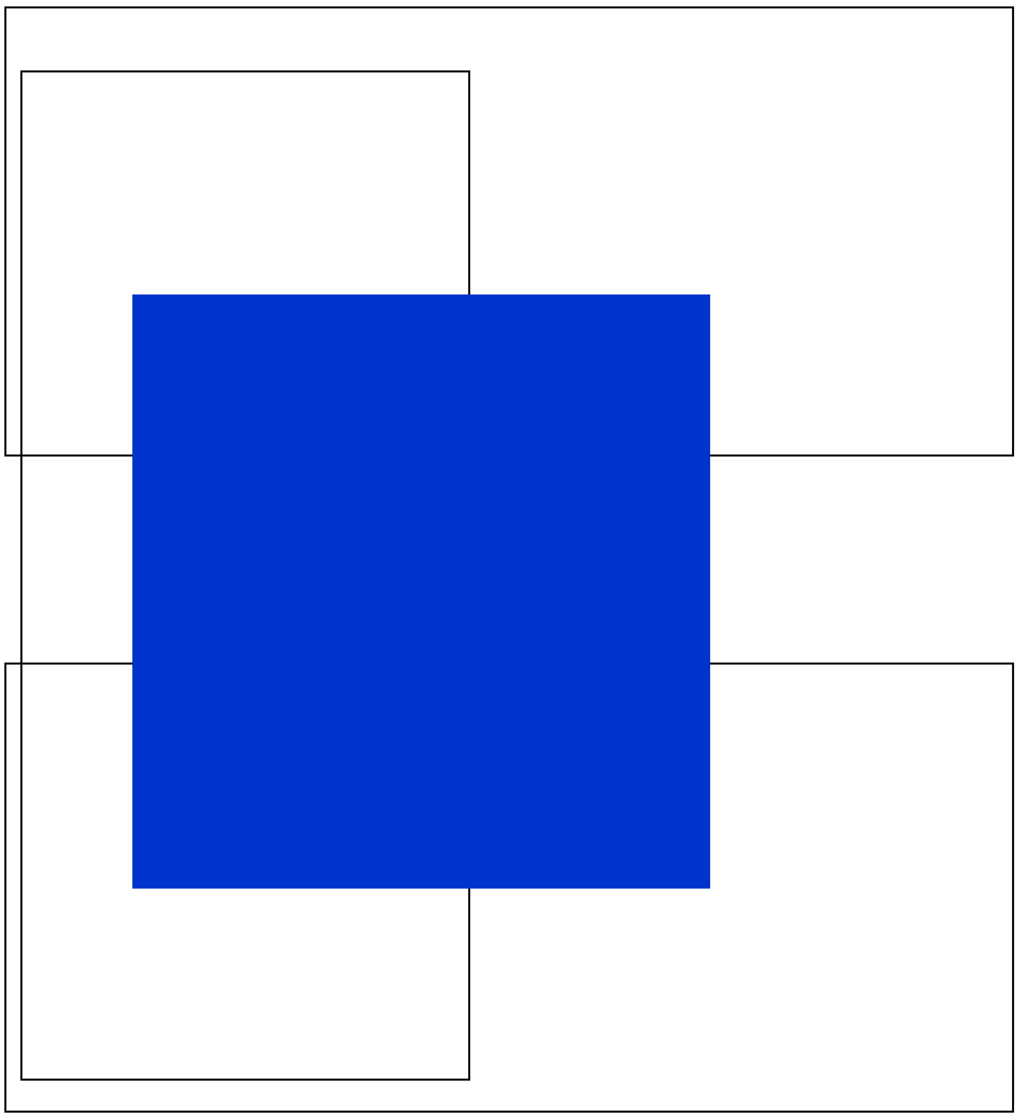
What behavior should we see in a motion algorithm?

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

The aperture problem



The aperture problem



Program demo

Motion analysis: related work

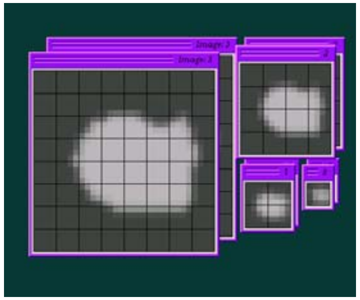
- **Markov network**
  - Luettgen, Karl, Willsky and collaborators.
- **Neural network or learning-based**
  - Nowlan & T. J. Senjowski; Sereno.
- **Optical flow analysis**
  - Weiss & Adelson; Darrell & Pentland; Ju, Black & Jepson; Simoncelli; Grzywacz & Yuille; Hildreth; Horn & Schunk; etc.

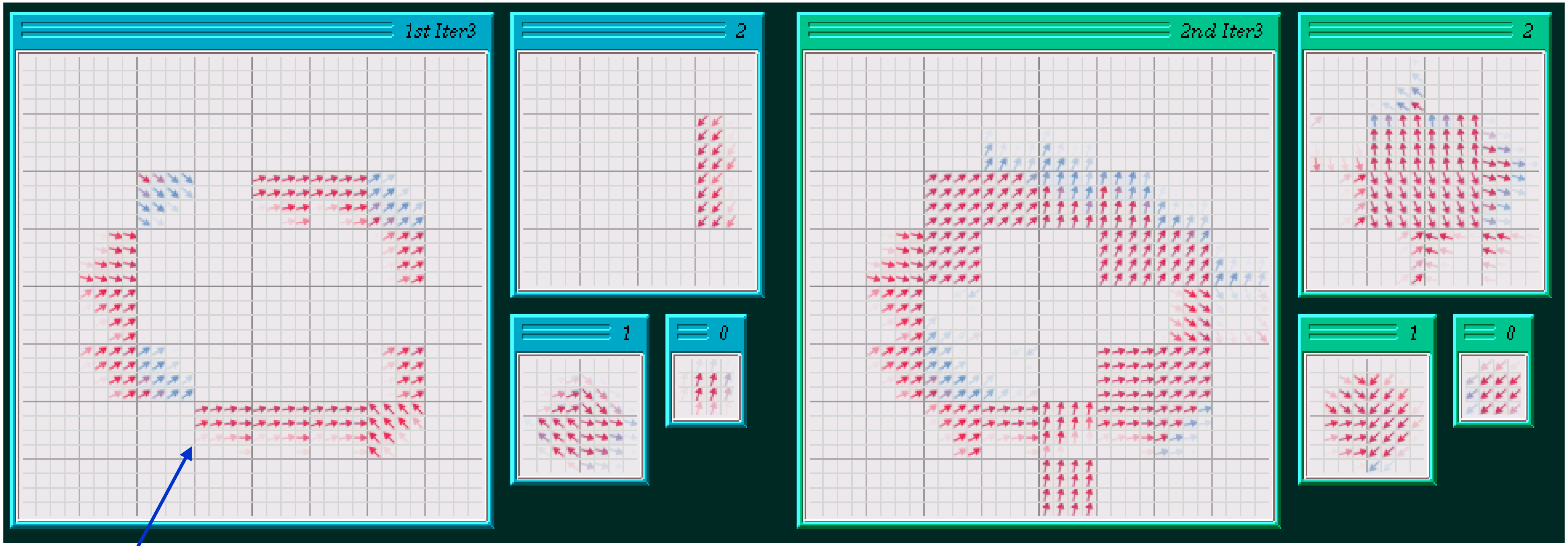
Inference:

Motion estimation results

(maxima of scene probability distributions displayed)

Image data



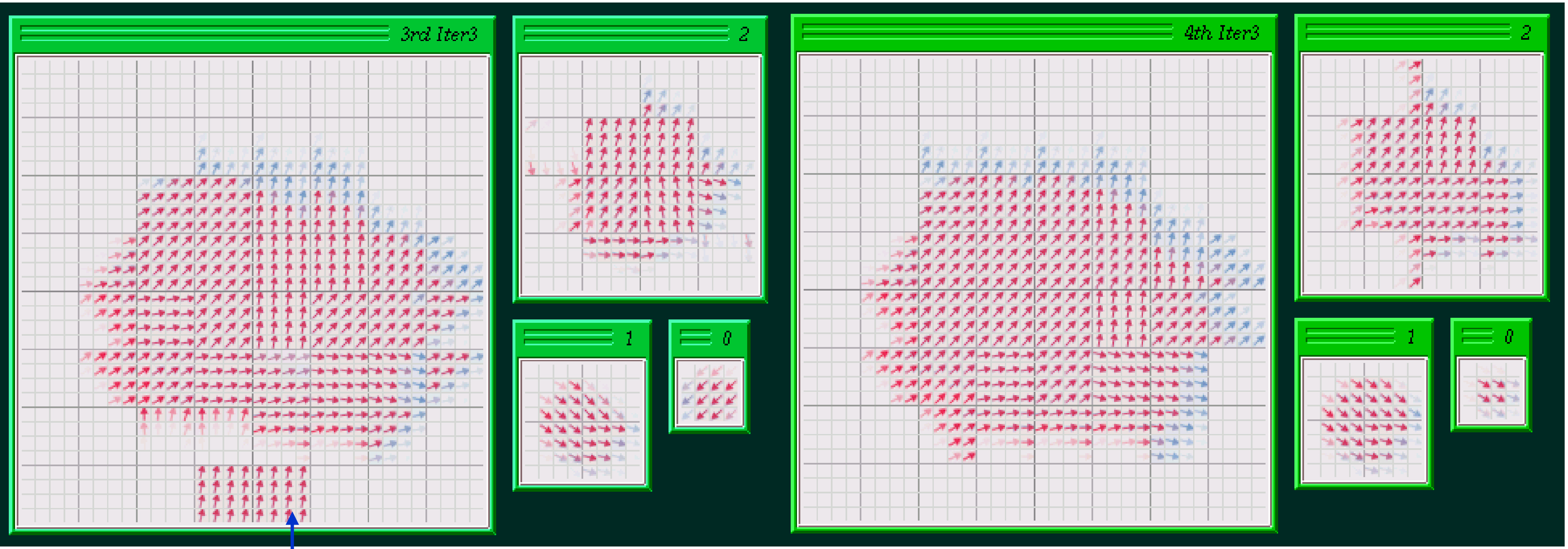


Iterations 0 and 1

Initial guesses only
show motion at edges.

Motion estimation results

(maxima of scene probability distributions displayed)

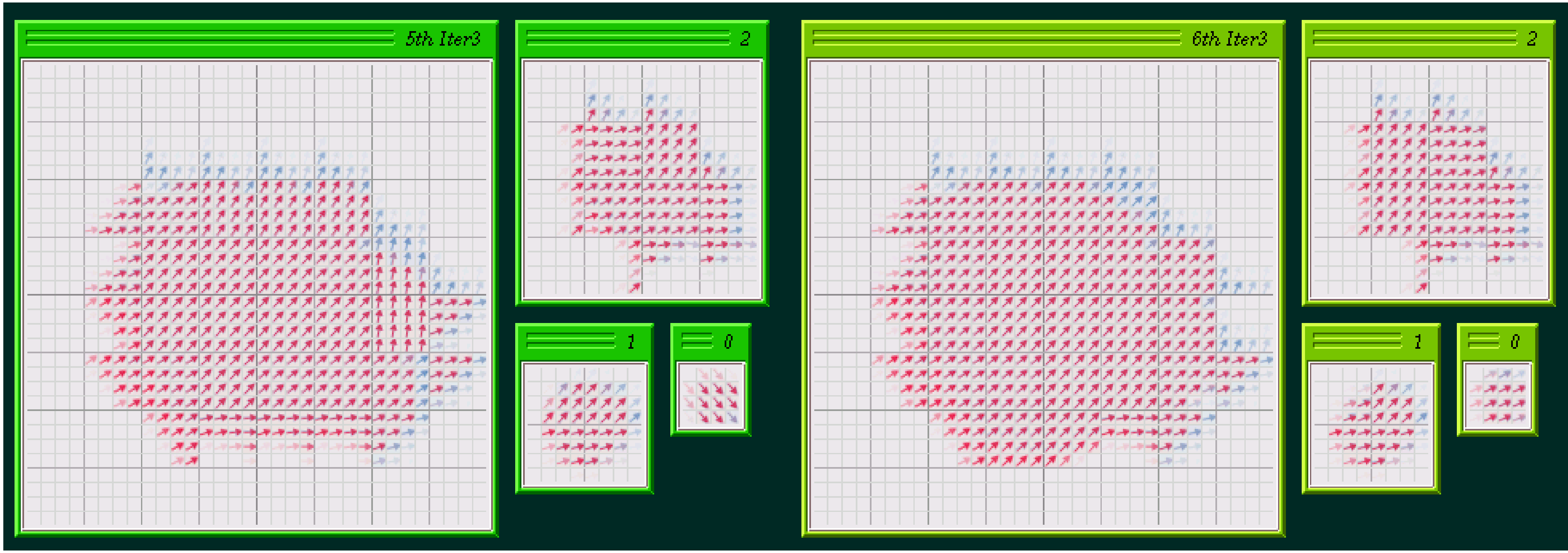


Iterations 2 and 3

Figure/ground still
unresolved here.

Motion estimation results

(maxima of scene probability distributions displayed)



Iterations 4 and 5

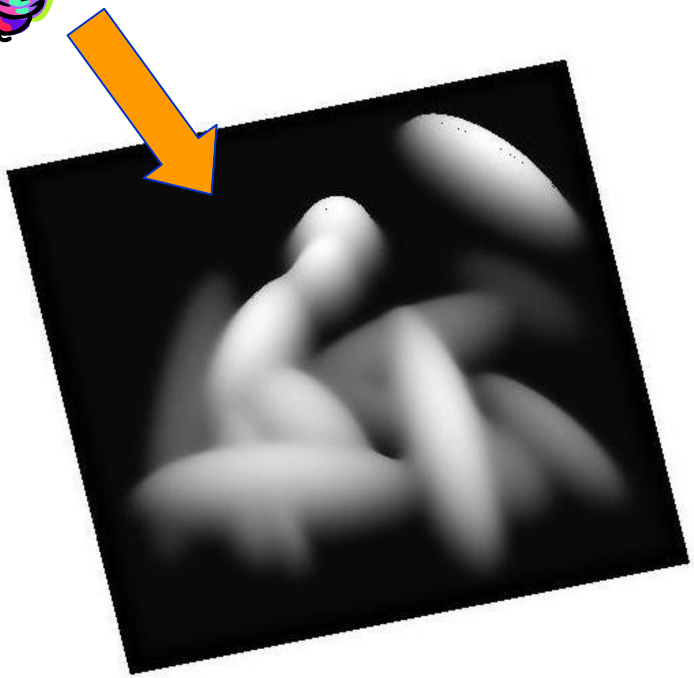
Final result compares well with vector quantized true (uniform) velocities.

Vision applications of MRF's

- Stereo
- Motion estimation
- Labelling shading and reflectance
- Many others…

Forming an Image

Illuminate the surface to get:

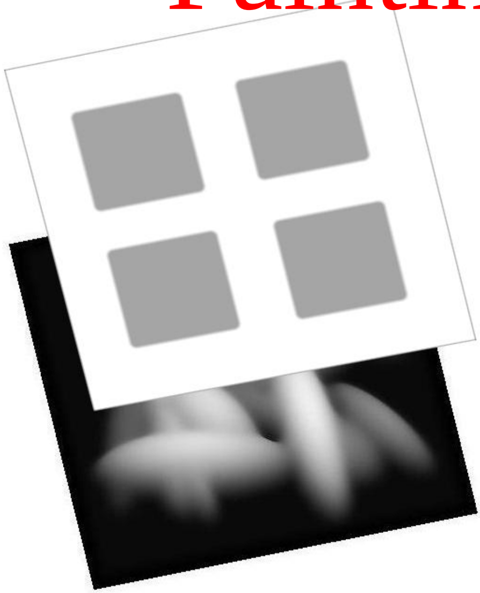


Surface (Height Map)

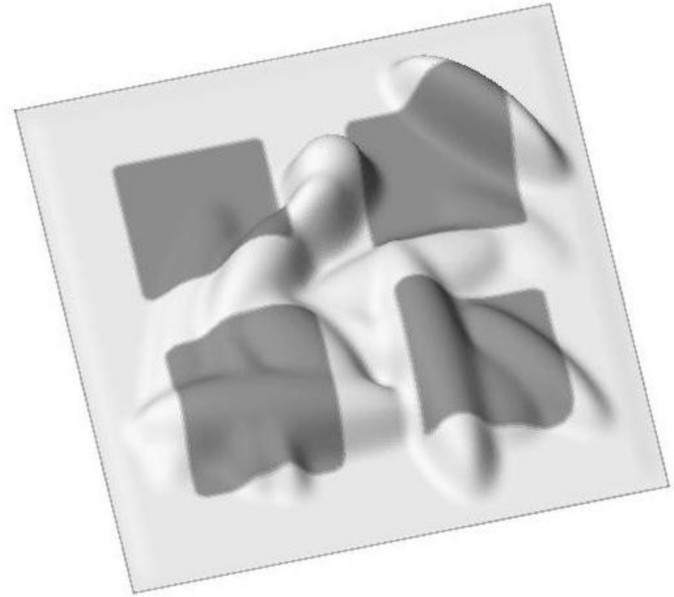
Shading Image

The shading image is the interaction of the shape of the surface and the illumination

Painting the Surface



Scene



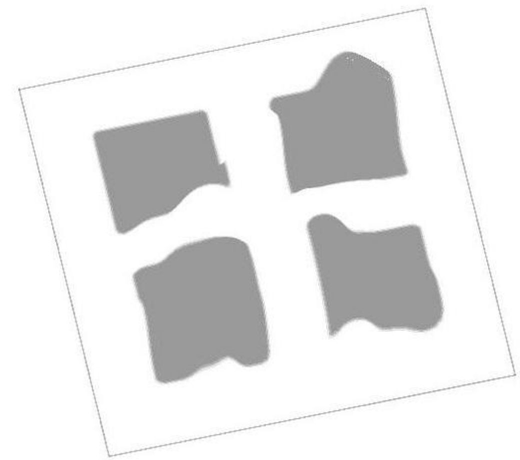
Image

Add a reflectance pattern to the surface. Points inside the squares should reflect less light

Goal

Image

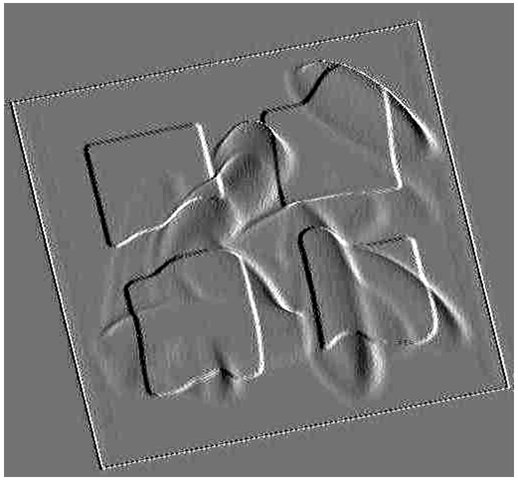
Shading Image



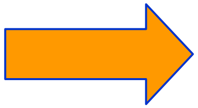
Reflectance Image

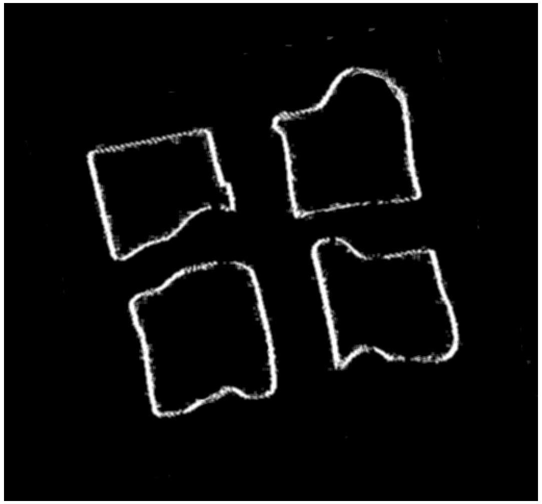
Basic Steps

1. Compute the *x* and *y* image derivatives
2. Classify each derivative as being caused by *either* shading or a reflectance change
3. Set derivatives with the wrong label to zero.
4. Recover the intrinsic images by finding the least-squares solution of the derivatives.



Original *x* derivative image





Classify each derivative
(White is reflectance)

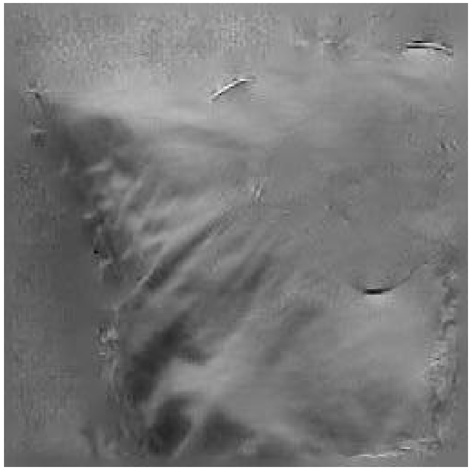
Learning the Classifiers

- Combine multiple classifiers into a strong classifier using AdaBoost (Freund and Schapire)
- Choose weak classifiers greedily similar to (Tieu and Viola 2000)
- Train on synthetic images
- Assume the light direction is from the right

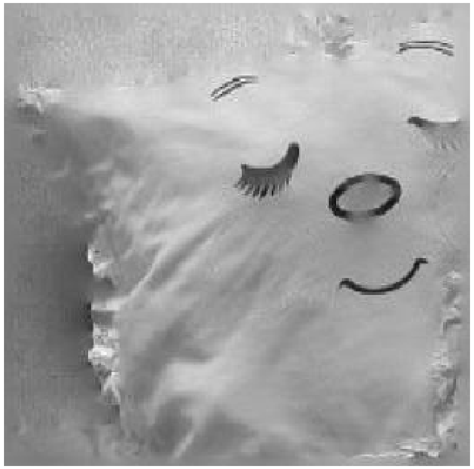
Shading Training Set

Reflectance Change Training Set

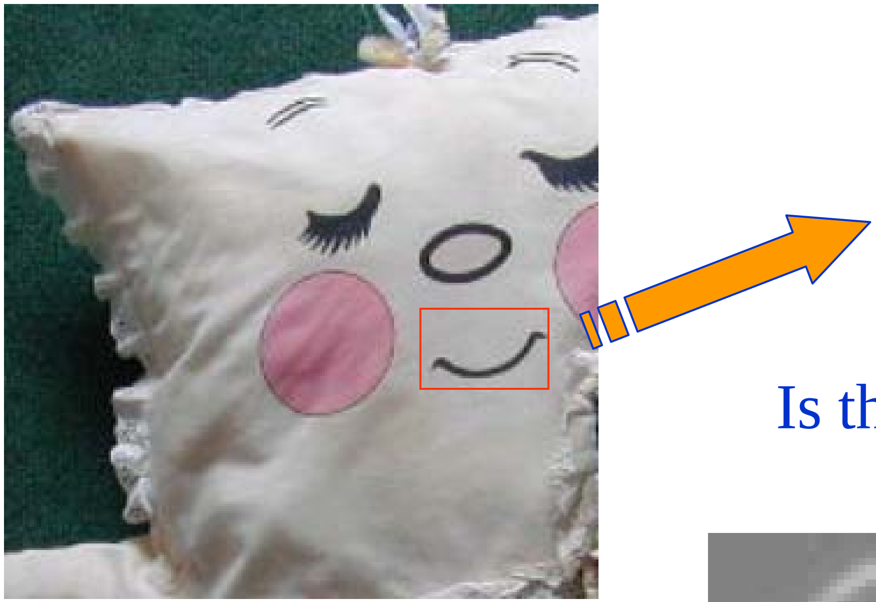
Using Both Color and Gray-Scale Information



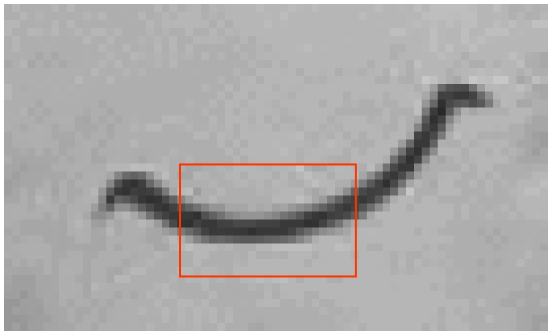
Results without considering gray-scale



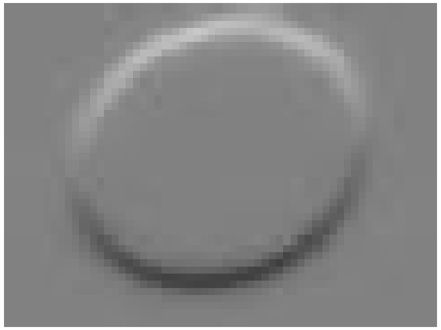
Some Areas of the Image Are Locally Ambiguous



Input



Is the change here better explained as



Shading

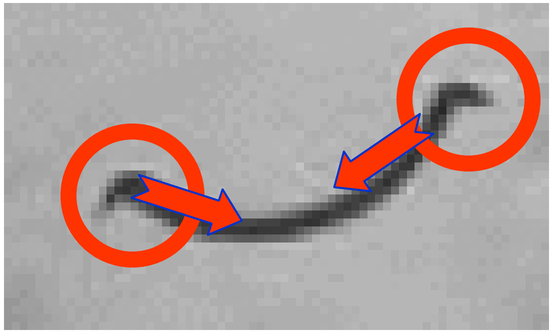
or

Reflectance

?

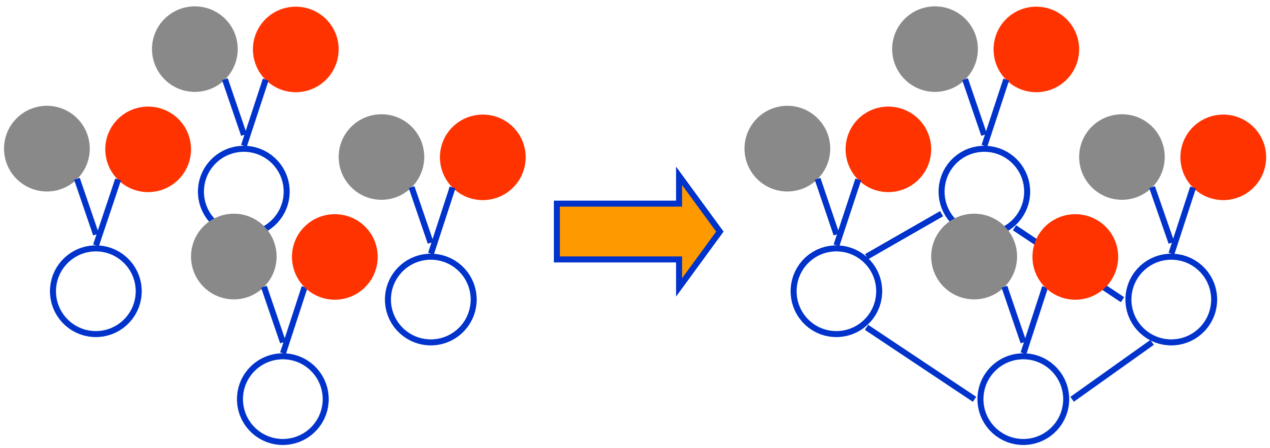
Propagating Information

- Can disambiguate areas by propagating information from reliable areas of the image into ambiguous areas of the image



Propagating Information

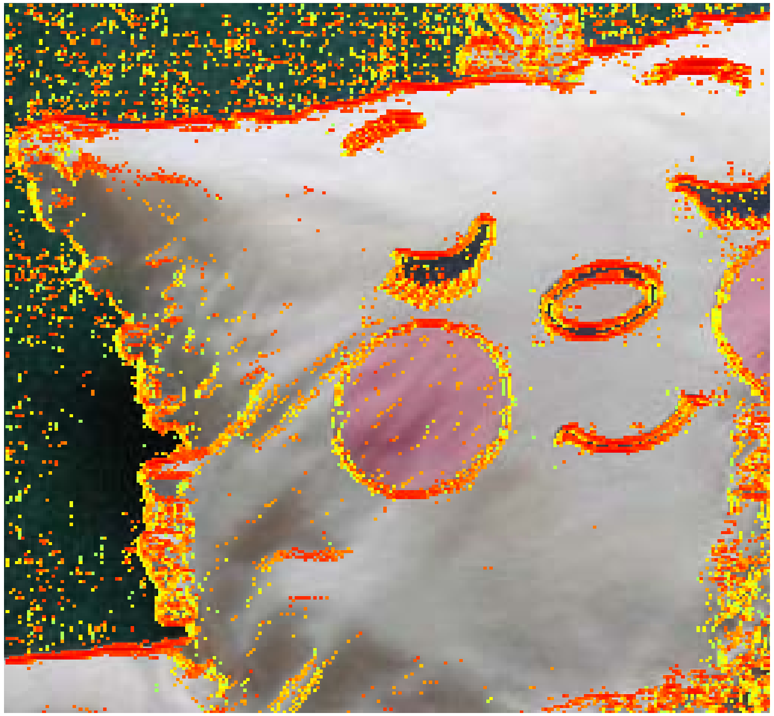
- Consider relationship between neighboring derivatives



- Use Generalized Belief Propagation to infer labels

Setting Compatibilities

- Set compatibilities according to image contours
  - All derivatives along a contour should have the same label
- Derivatives along an image contour strongly influence each other

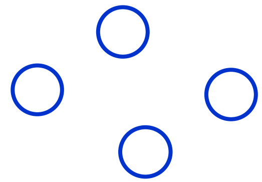


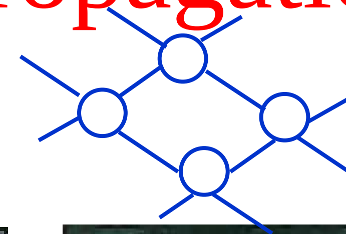
$\beta =$ 

0.5 1.0

$$\psi(x_i, x_j) = \begin{bmatrix} 1-\beta & \beta \\ \beta & 1-\beta \end{bmatrix}$$

Improvements Using Propagation





Input Image

Reflectance Image
Without Propagation

Reflectance Image
With Propagation

Combining: local evidence from shape and color, and GBP for propagation

(a) Original Image　　(b) Shading Image　　(c) Reflectance Image

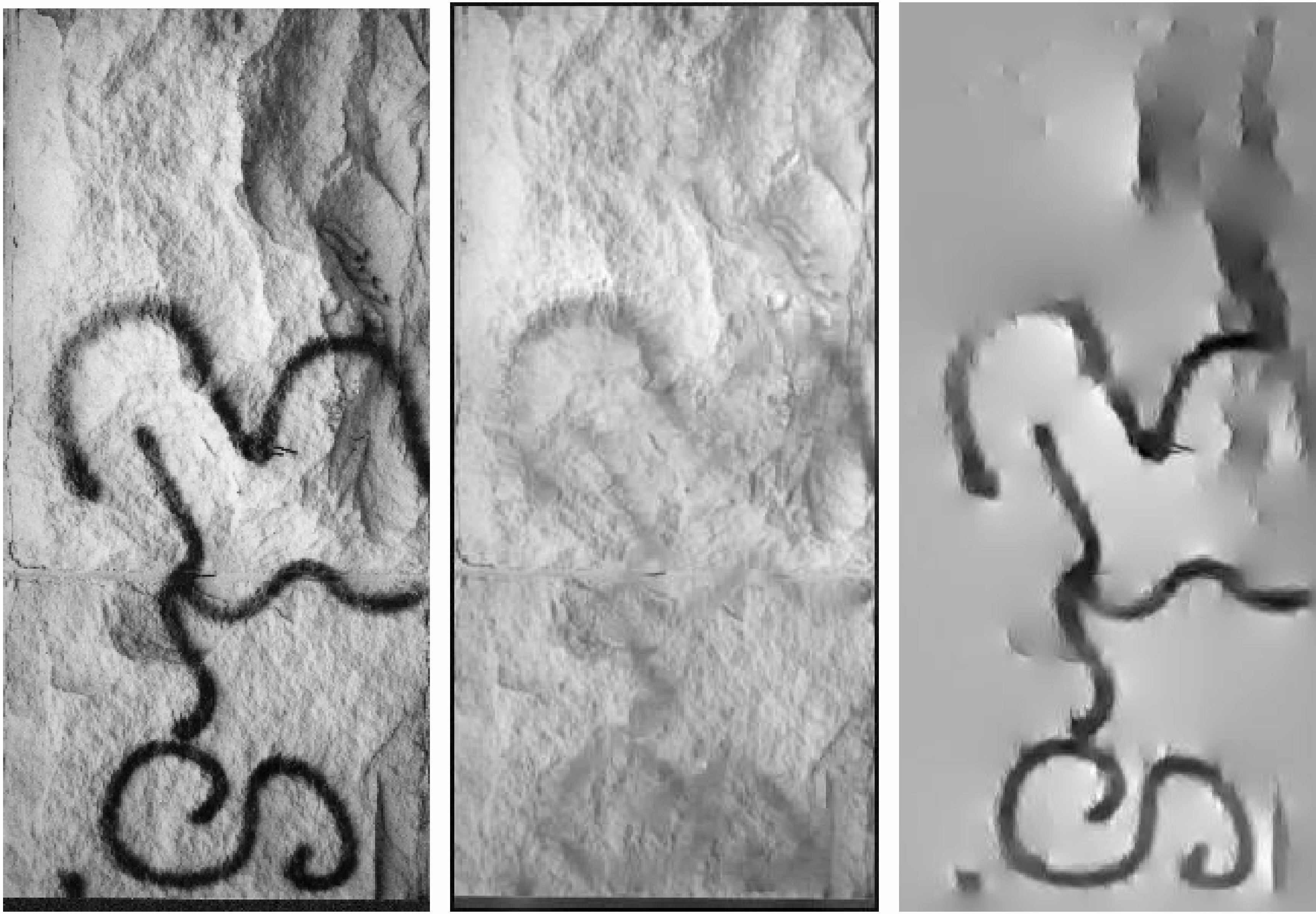
(More Results)

Input Image

Shading Image

Reflectance Image

(a) Original Image



(a) Original Image (b) Shape Image (c) Reflectance Image

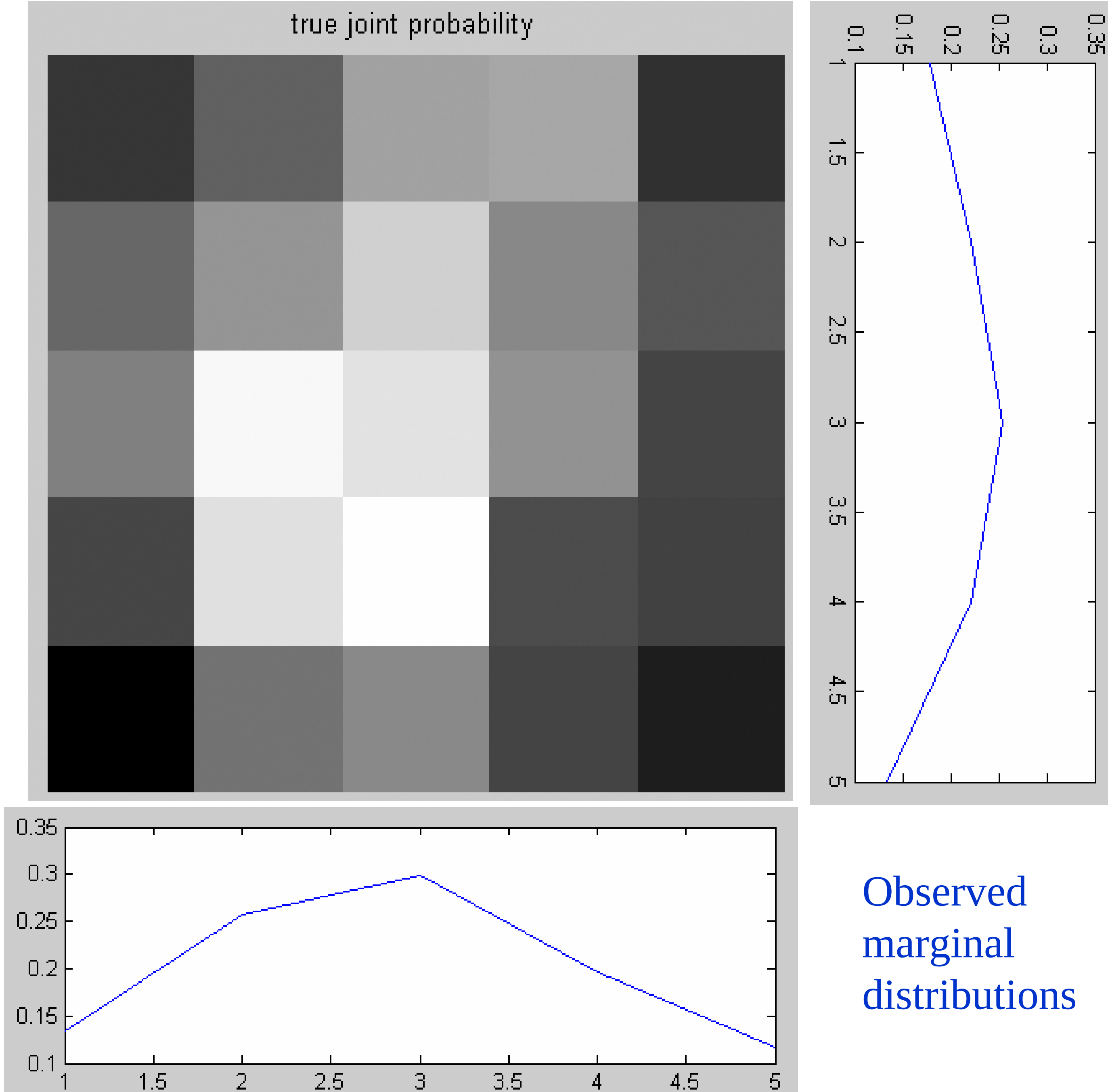
Outline of MRF section

- 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

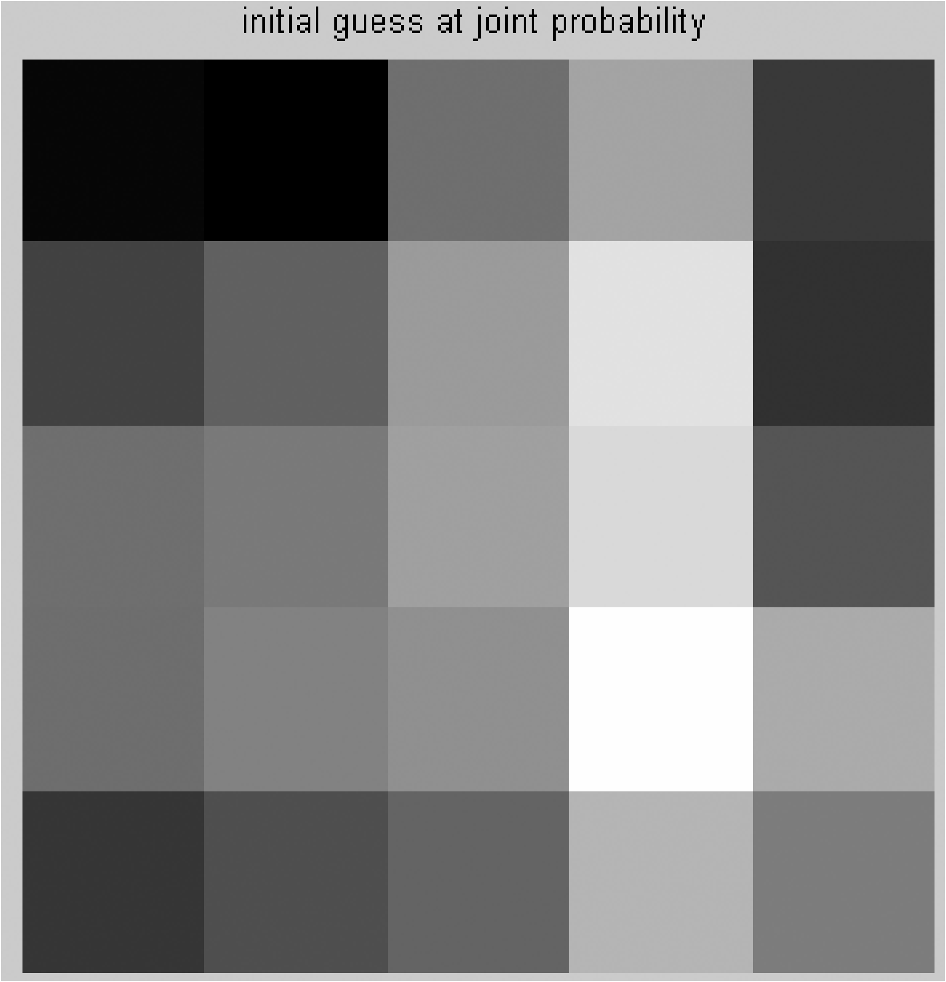
Iterative proportional fitting lets you make a maximum likelihood estimate a joint distribution from observations of various marginal distributions.

True joint probability



Observed marginal distributions

Initial guess at joint probability



IPF update equation

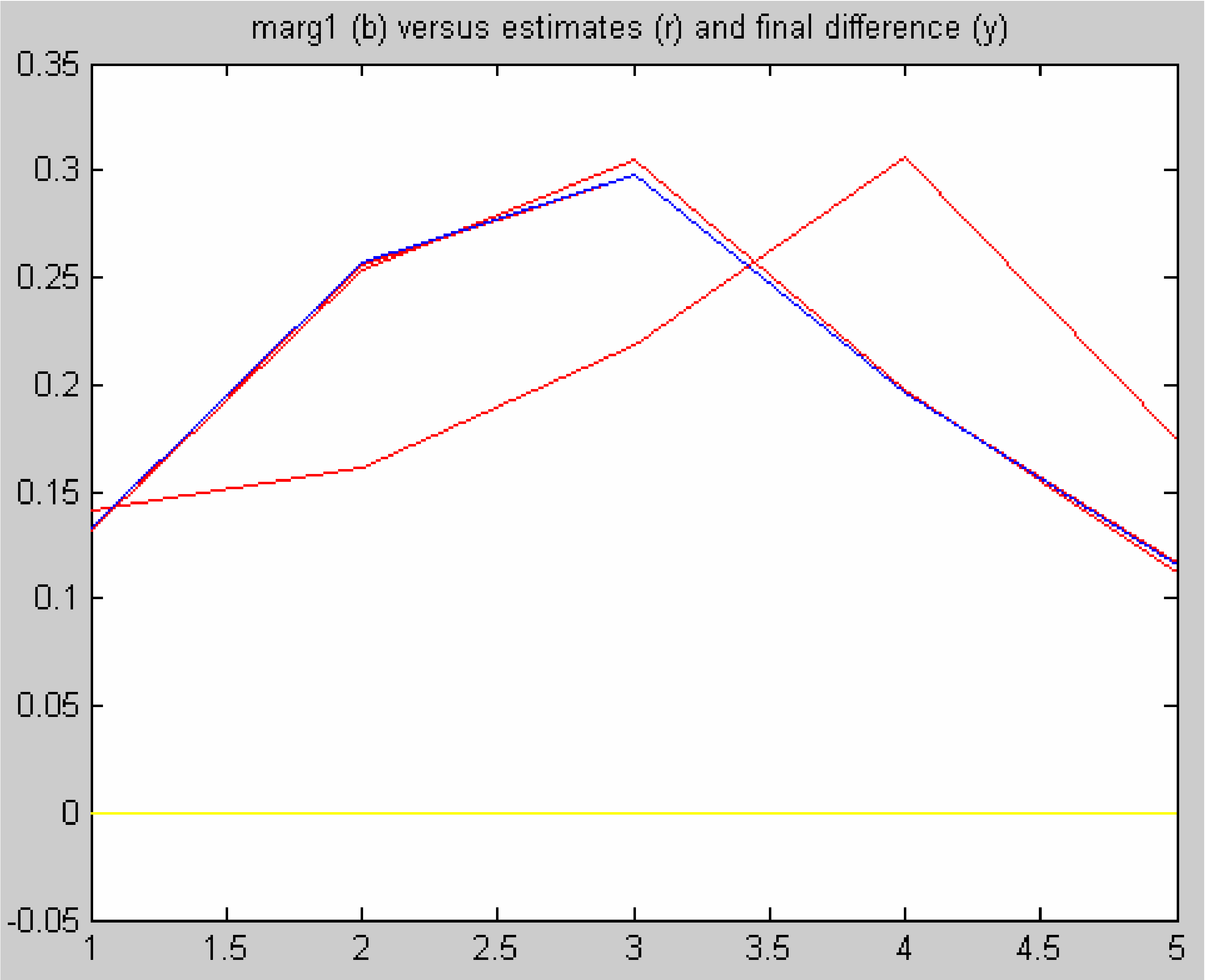
$$P(x_1, x_2, \ldots, x_d)^{(t+1)} = P(x_1, x_2, \ldots, x_d)^{(t)} \frac{P(x_i)^{\text{observed}}}{P(x_i)^{(t)}}$$

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

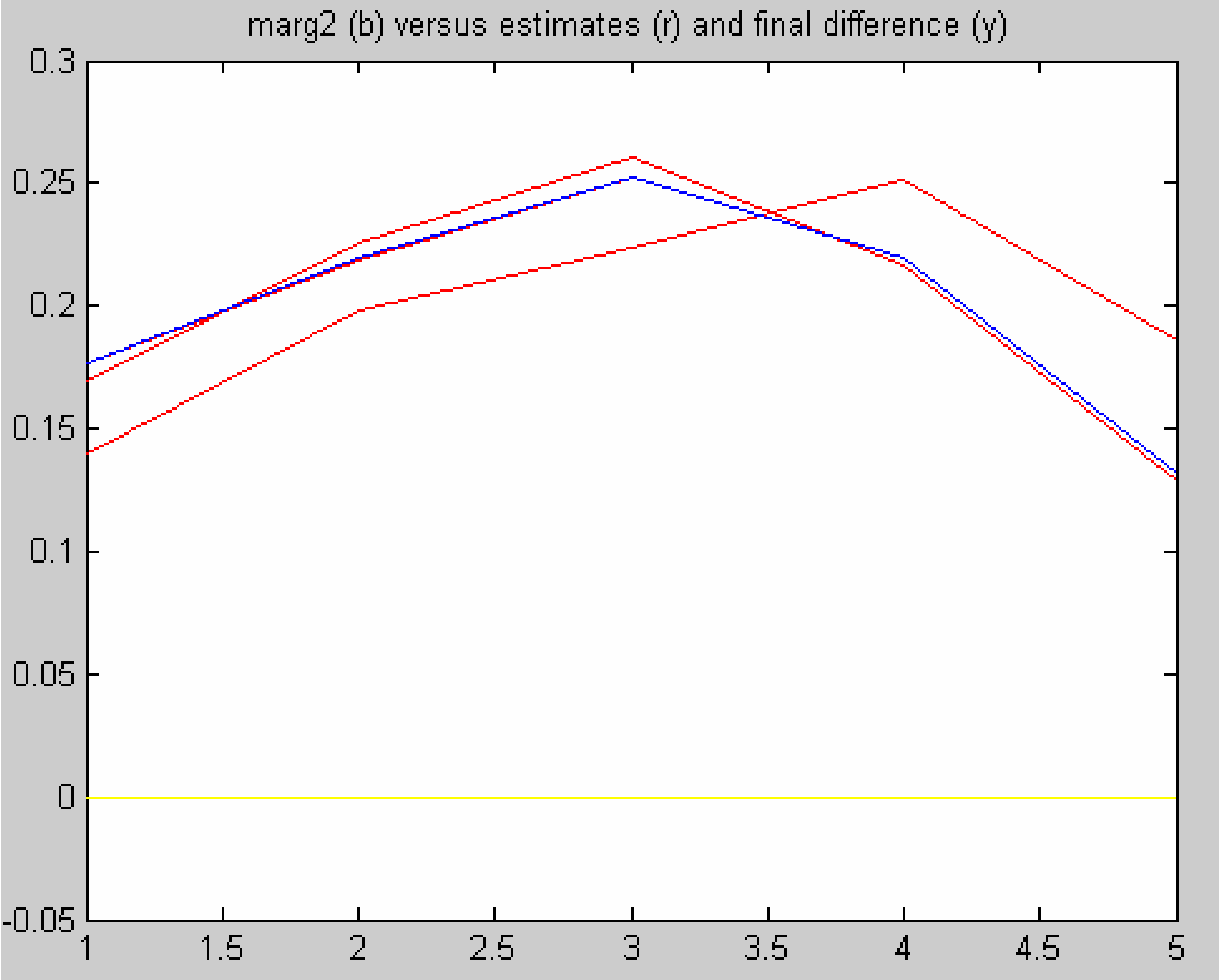
Gives gradient ascent in the likelihood of the joint probability, given the observations of the marginals.

See: Michael Jordan's book on graphical models

Convergence of to correct marginals by IPF algorithm

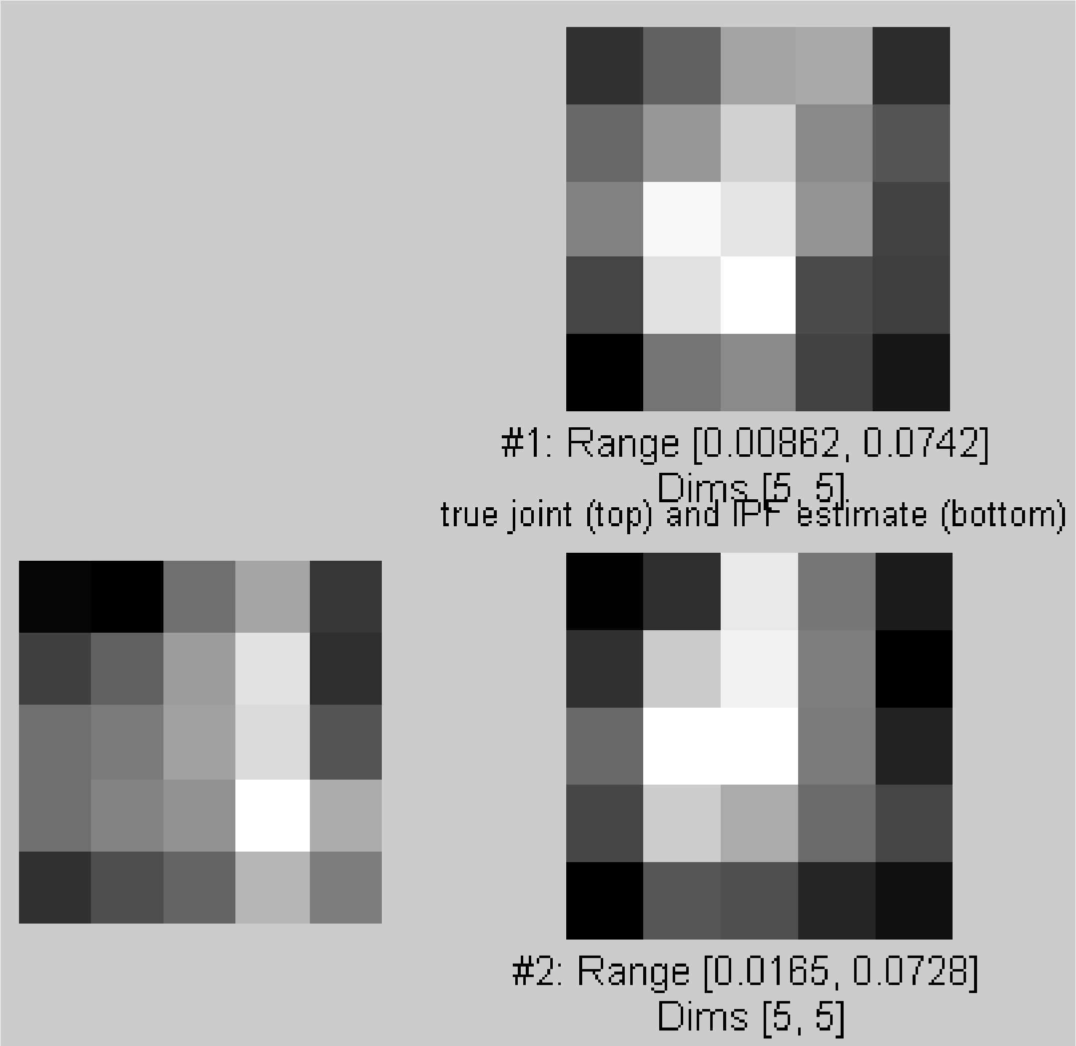


Convergence of to correct marginals by IPF algorithm



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.

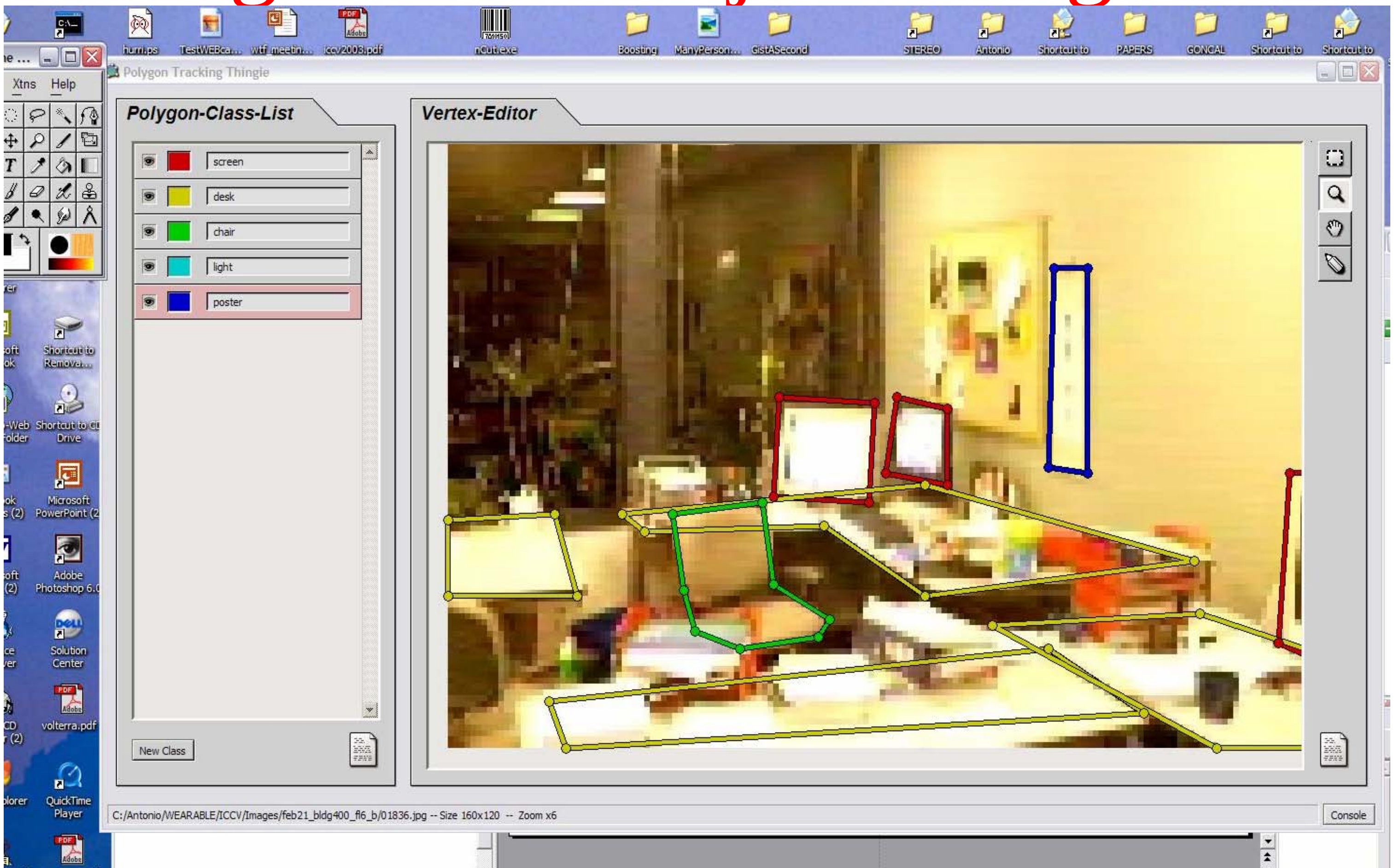
“Top-down” information: a representation for image context

Images

80-dimensional representation

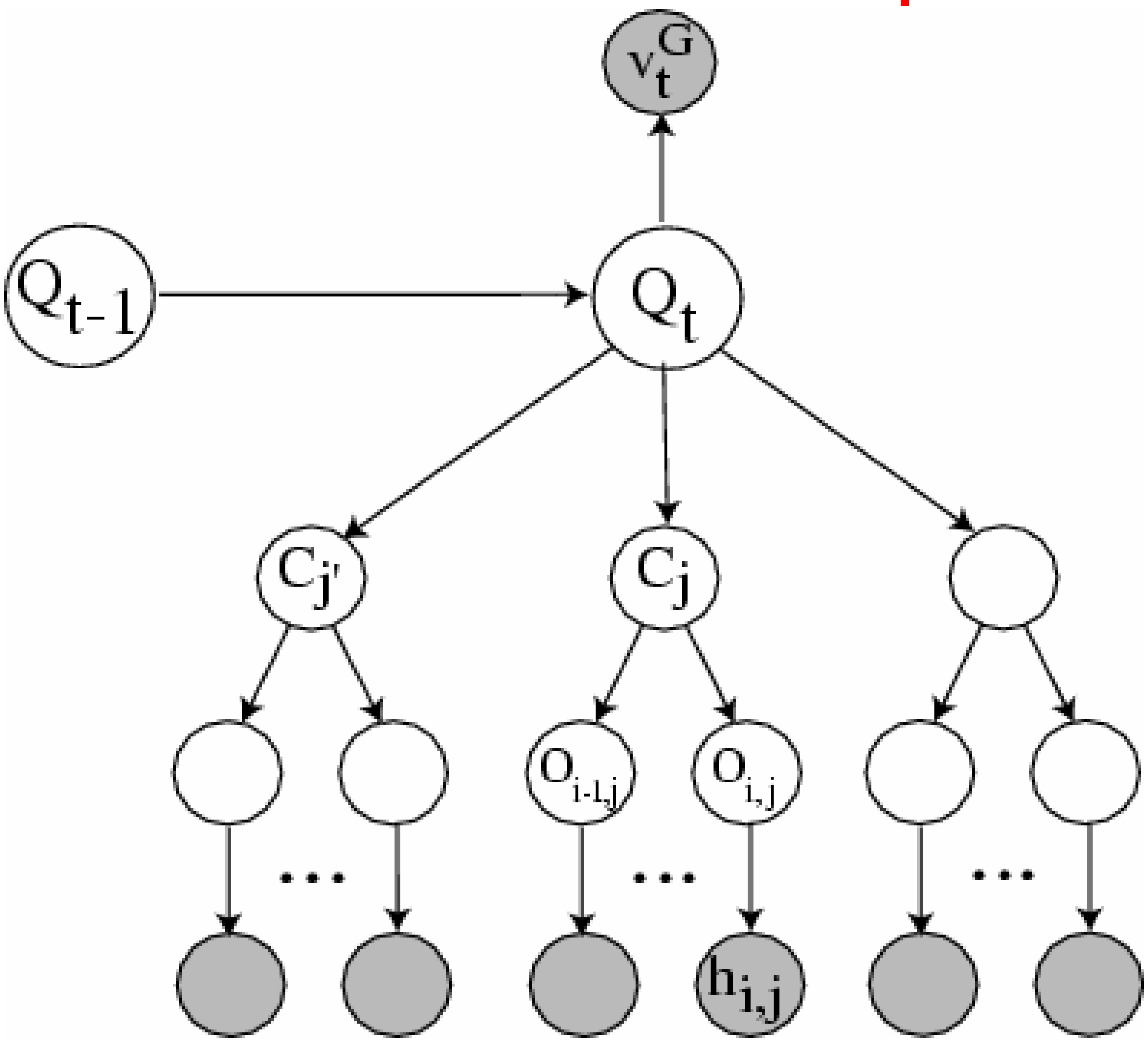
Credit: Antonio Torralba

"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



Visual "gist" observations

Scene category

kitchen, office, lab, conference room, open area, corridor, elevator and street.

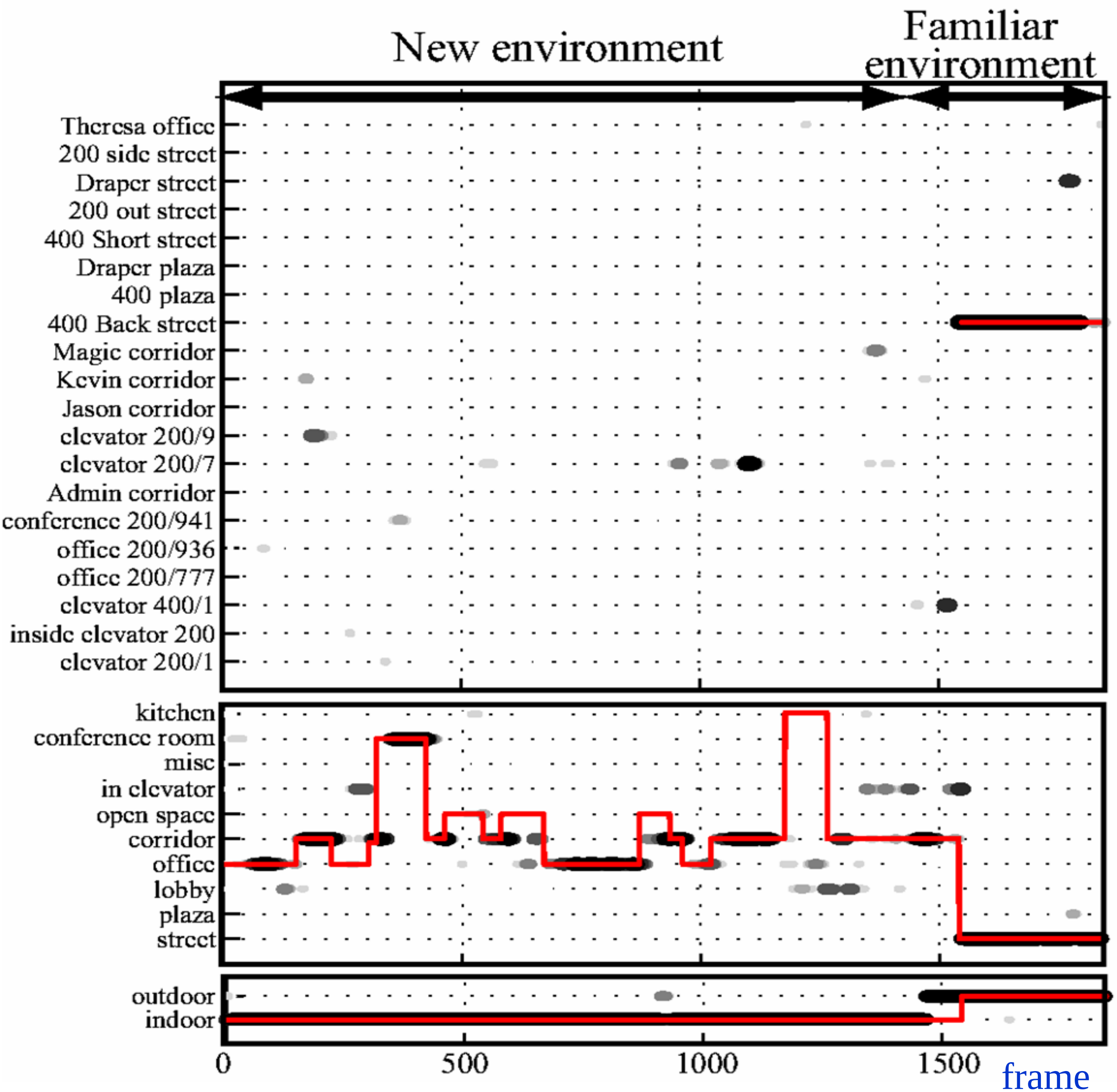
Object class

Particular objects

Local image features

Categorization of new places

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

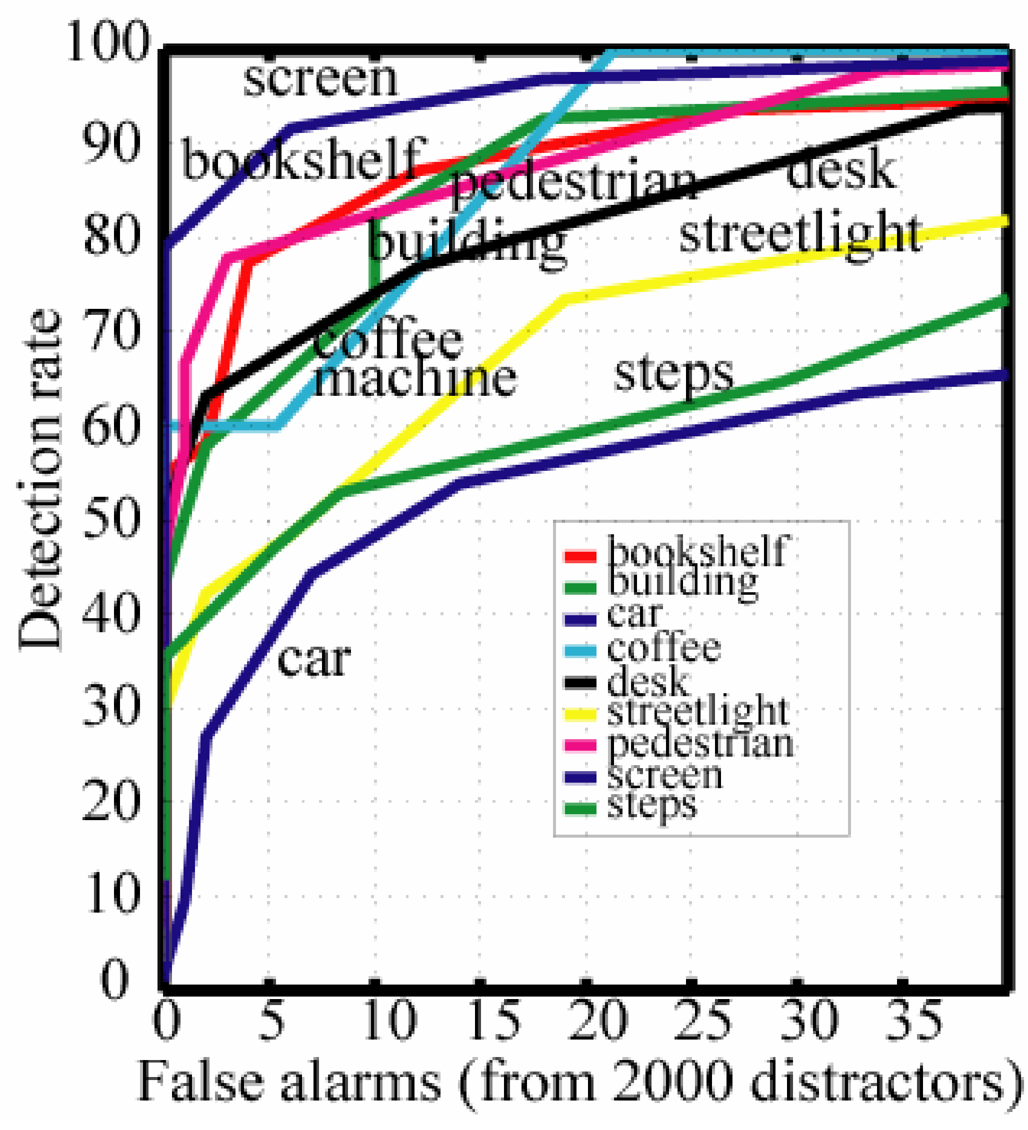


Specific location

Location category

Indoor/outdoor

Bottom-up detection: ROC curves



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