Missing variable problems

- In many vision problems, if some variables were known the maximum likelihood inference problem would be easy
  - fitting; if we knew which line each token came from, it would be easy to determine line parameters
  - segmentation; if we knew the segment each pixel came from, it would be easy to determine the segment parameters
  - fundamental matrix estimation; if we knew which feature corresponded to which, it would be easy to determine the fundamental matrix
  - etc.
- This sort of thing happens in statistics, too

Bayes theorem

$$P(x, y) = P(x|y)\, P(y)$$

so

$$P(x|y)\, P(y) = P(y|x)\, P(x)$$

and

$$P(x|y) = P(y|x)\, P(x) \,/\, P(y)$$

The parameters you want to estimate

What you observe

Likelihood function

Prior probability

Constant w.r.t. parameters x.

Missing variable problems

General case:

- Complete space X (e.g., pixel values and labels)
- Incomplete space Y (e.g., pixel values)
- f: X->Y
- Parameters U (e.g., mixing weights cluster mean, covar.)

Complete log-likelihood: For Independent data samples

$$L_c(\boldsymbol{x};\boldsymbol{u}) = \log\{\prod_j p_c(\boldsymbol{x}_j;\boldsymbol{u})\}$$
$$= \sum_j \log\left(p_c(\boldsymbol{x}_j;\boldsymbol{u})\right)$$

Missing variables - strategy

- We have a problem with parameters, missing variables
- This suggests:
- Iterate until convergence
  - replace missing variable with expected values, given **fixed** values of parameters
  - fix missing variables, choose parameters to maximise likelihood given fixed values of missing variables
- e.g., iterate till convergence
  - allocate each point to a line with a weight, which is the probability of the point given the line
  - refit lines to the weighted set of points
- Converges to local extremum

EM for Mixture models

If log-likelihood is linear in missing variables we can replace missing variables with expectations. E.g.,

$$p(\boldsymbol{y}) = \sum_l \pi_l p(\boldsymbol{y}|\boldsymbol{a}_l)$$

mixture model

$$\sum_{j \in \text{observations}} \left(\sum_{l=1}^{g} z_{lj} \log p(\boldsymbol{y}_j|\boldsymbol{a}_l) \right)$$

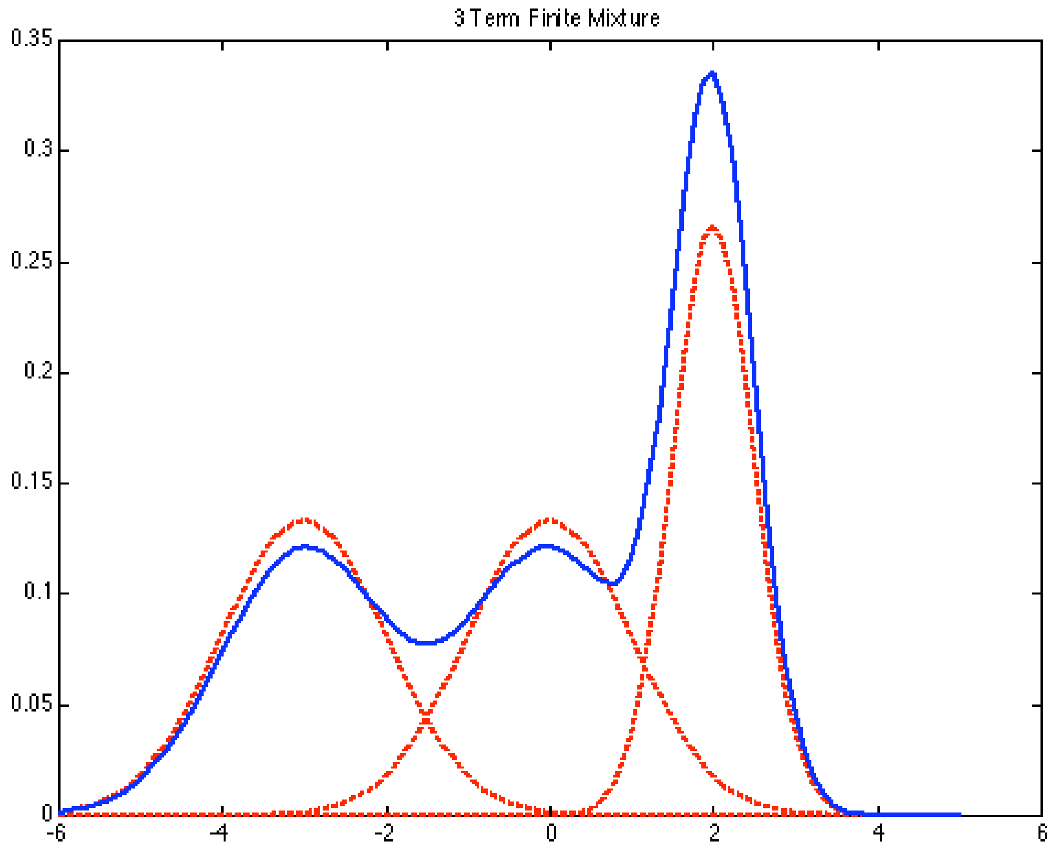
complete data log-likelihood

1. (E-step) estimate complete data (e.g, z_j's) using previous parameters
2. (M-step) maximize complete log-likelihood using estimated complete data

$$\begin{aligned} \boldsymbol{u}^{s+1} &= \arg\max_{\boldsymbol{u}} L_c(\overline{\boldsymbol{x}}^s; \boldsymbol{u}) \\ &= \arg\max_{\boldsymbol{u}} L_c([\boldsymbol{y}, \overline{\boldsymbol{z}}^s]; \boldsymbol{u}) \end{aligned}$$

Finite Mixtures

$$P(x) = \square_{i=1:3} a(i)\, g_i(x; \square)$$



Set: Probability in segmentation
Slides by D.A. Forsyth

Color segmentation with EM

- At each pixel in an image, we compute a *d*-dimensional feature vector x, which encapsulates position, colour and texture information.
- Pixel is generated by one of G segments, each Gaussian, chosen with probability π:

$$p(\boldsymbol{x}) = \sum_i p(\boldsymbol{x}|\theta_l)\pi_l$$

Color segmentation with EM

Parameters include mixing weights and means/covars:

$$\Theta = (\alpha_1, \ldots, \alpha_g, \theta_1, \ldots, \theta_g). \qquad \theta_l = (\boldsymbol{\mu}_l, \Sigma_l)$$

yielding

$$p(\boldsymbol{x}|\Theta) = \sum_{l=1}^{g} \alpha_l p_l(\boldsymbol{x}|\theta_l)$$

with

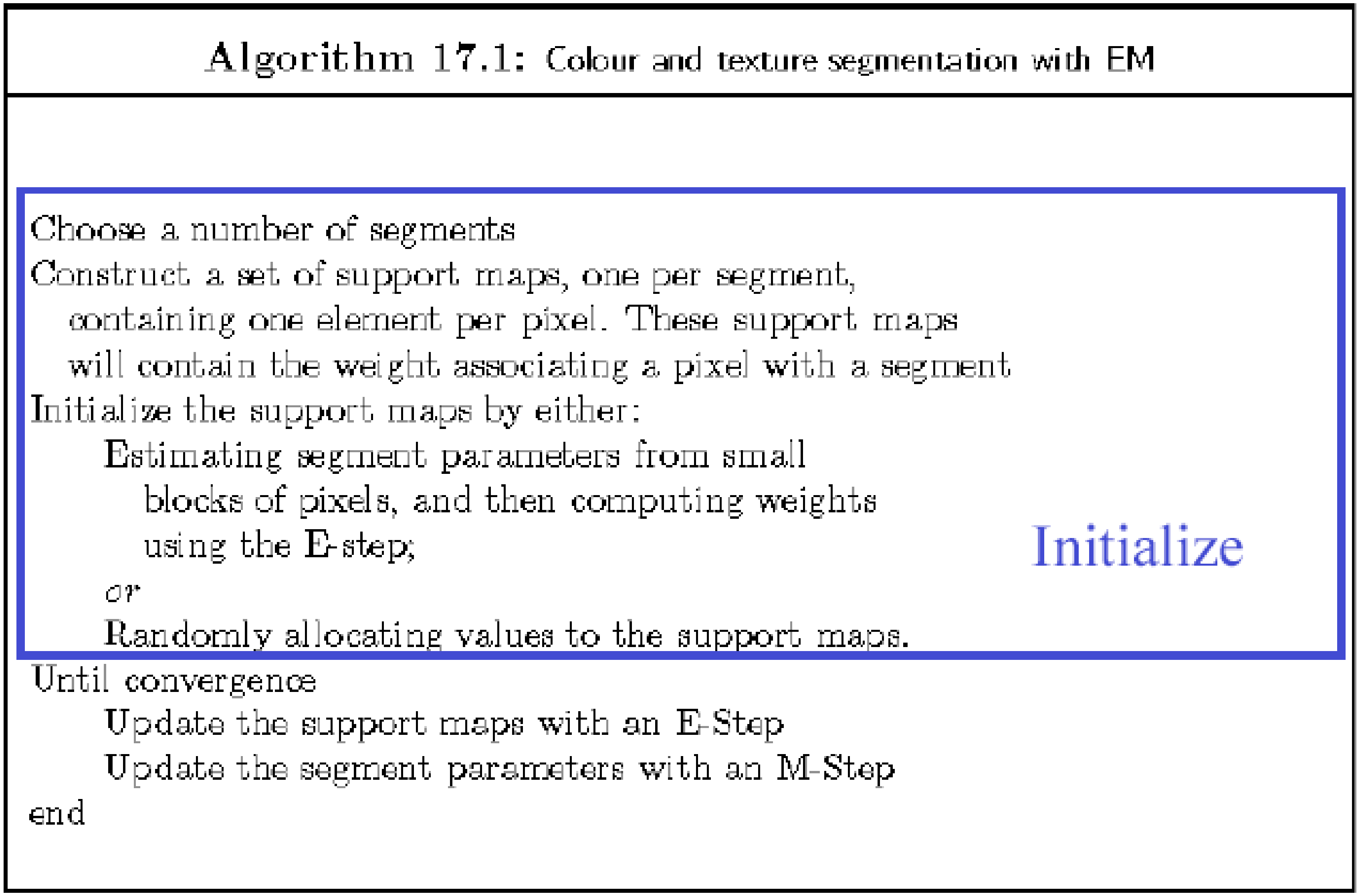
$$p_l(\boldsymbol{x}|\theta_l) = \frac{1}{(2\pi)^{d/2}\det(\Sigma_i)^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^T \Sigma_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)\right\}$$

Color segmentation with EM

Algorithm 17.1: Colour and texture segmentation with EM

```
Choose a number of segments
Construct a set of support maps, one per segment,
   containing one element per pixel. These support maps
   will contain the weight associating a pixel with a segment
Initialize the support maps by either:
      Estimating segment parameters from small
         blocks of pixels, and then computing weights
         using the E-step;
      or
      Randomly allocating values to the support maps.
Until convergence
      Update the support maps with an E-Step
      Update the segment parameters with an M-Step
end
```

Color segmentation with EM



Algorithm 17.1: Colour and texture segmentation with EM

```
Choose a number of segments
Construct a set of support maps, one per segment,
   containing one element per pixel. These support maps
   will contain the weight associating a pixel with a segment
Initialize the support maps by either:
      Estimating segment parameters from small
         blocks of pixels, and then computing weights
         using the E-step;
      or
      Randomly allocating values to the support maps.
Until convergence
      Update the support maps with an E-Step
      Update the segment parameters with an M-Step
end
```

Initialize

Expection on Indicator variables

The E-step: The l, m'th element of $\mathcal{I}$ is one if the l'th pixel comes from the m'th blob, and zero otherwise. This means that

$$\begin{aligned}\mathbf{E}(I_{lm}) = {} & 1P(l\text{'th pixel comes from the } m\text{'th blob}) + \\ & 0.P(l\text{'th pixel does not come from the } m\text{'th blob}) \\ = {} & P(l\text{'th pixel comes from the } m\text{'th blob})\end{aligned}$$

Assuming that the parameters are for the s'th iteration are $\Theta^{(s)}$, we have:

$$\overline{I}_{lm} = \frac{\alpha_m^{(s)} p_m(\boldsymbol{x}_l|\theta_l^{(s)})}{\sum_{k=1}^{K} \alpha_k^{(s)} p_k(\boldsymbol{x}_l|\theta_l^{(s)})}$$

(keeping in mind that $\alpha_m^{(s)}$ means the value of α_m on the s'th iteration!).

Algorithm 17.2: Colour and texture segmentation with EM: - the E-step

```
For each pixel location $l$
    For each segment $m$
        Insert $\alpha_m^{(s)} p_m(\boldsymbol{x}_l|\theta_l^{(s)})$
            in pixel location $l$ in the support map $m$
    end  Add the support map values to obtain
        $\sum_{k=1}^{K} \alpha_k^{(s)} p_k(\boldsymbol{x}_l|\theta_l^{(s)})$
        and divide the value in location $l$ in each support map by this term
end
```

E-step

Estimate support maps:

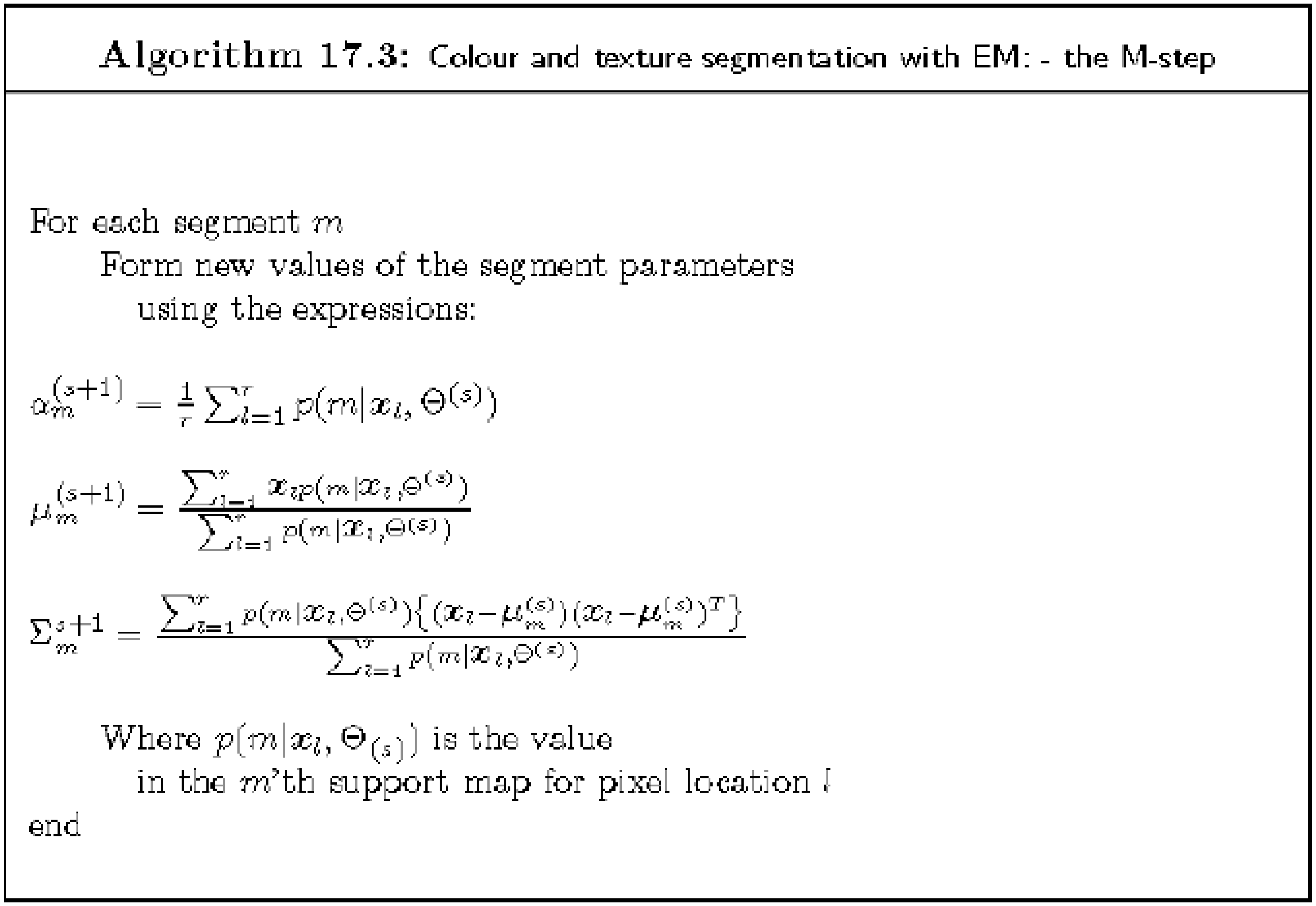
$$p(m|\boldsymbol{x}_l, \Theta_{(s)}) = \frac{\alpha_m^{(s)} p_m(\boldsymbol{x}_l|\theta_l^{(s)})}{\sum_{k=1}^{K} \alpha_k^{(s)} p_k(\boldsymbol{x}_l|\theta_l^{(s)})}$$

Algorithm 17.2: Colour and texture segmentation with EM: - the E-step

```
For each pixel location $l$
    For each segment $m$
      Insert $\alpha_m^{(s)} p_m(x_l|\theta_l^{(s)})$
        in pixel location $l$ in the support map $m$
    end  Add the support map values to obtain
      $\sum_{k=1}^{K} \alpha_k^{(s)} p_k(x_l|\theta_l^{(s)})$
      and divide the value in location $l$ in each support map by this term
end
```

M-step

Update mean's, covar's, and mixing coef.'s using support map:



Algorithm 17.3: Colour and texture segmentation with EM: - the M-step

```
For each segment $m$
    Form new values of the segment parameters
      using the expressions:

$\alpha_m^{(s+1)} = \frac{1}{r}\sum_{l=1}^{r} p(m|x_l, \Theta^{(s)})$

$\mu_m^{(s+1)} = \frac{\sum_{l=1}^{r} x_l p(m|x_l,\Theta^{(s)})}{\sum_{l=1}^{r} p(m|x_l,\Theta^{(s)})}$

$\Sigma_m^{s+1} = \frac{\sum_{l=1}^{r} p(m|x_l,\Theta^{(s)})\{(x_l-\mu_m^{(s)})(x_l-\mu_m^{(s)})^T\}}{\sum_{l=1}^{r} p(m|x_l,\Theta^{(s)})}$

    Where $p(m|x_l, \Theta_{(s)})$ is the value
      in the $m$'th support map for pixel location $l$
end
```

Segmentation with EM

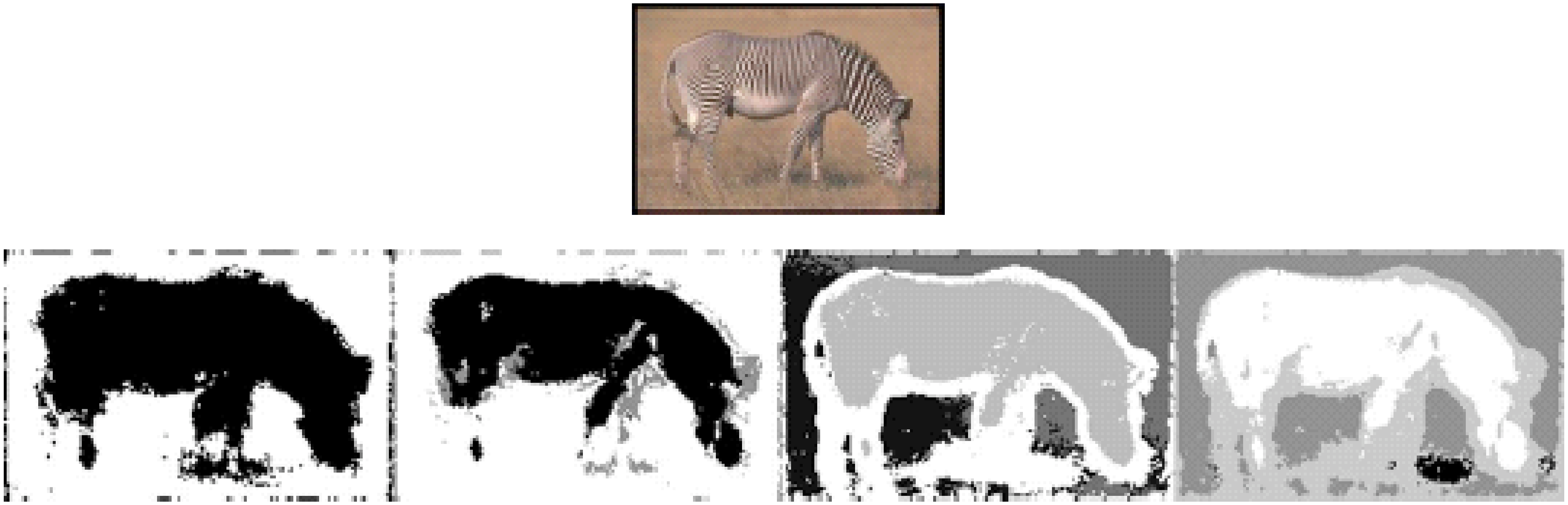


Figure from “Color and Texture Based Image Segmentation Using EM and Its Application to Content Based Image Retrieval”,S.J. Belongie et al., Proc. Int. Conf. Computer Vision, 1998, c1998, IEEE

Lines and robustness

- We have one line, and n points
- Some come from the line, some from “noise”
- This is a mixture model:

- We wish to determine
  - line parameters
  - p(comes from line)

$$
\begin{aligned}
P(\text{point} \mid \text{line and noise params}) &= P(\text{point} \mid \text{line})P(\text{comes from line}) + \\
&\quad P(\text{point} \mid \text{noise})P(\text{comes from noise}) \\
&= P(\text{point} \mid \text{line})\lambda + P(\text{point} \mid \text{noise})(1-\lambda)
\end{aligned}
$$

EM for line estimation

- We have a problem with parameters, missing variables

Iterate until convergence:

  - replace missing variable with expected values, given **fixed** values of parameters
  - fix missing variables, choose parameters to maximise likelihood given **fixed** values of missing variables

- e.g.,
  - allocate each point to a line with a weight, which is the probability of the point given the line
  - refit lines to the weighted set of points
- Converges to local extremum
- Somewhat more general form is available

Estimating the mixture model

- Introduce a set of hidden variables, δ, one for each point. They are one when the point is on the line, and zero when off.
- If these are known, the negative log-likelihood becomes (the line's parameters are θ, c):
- Here K is a normalising constant, kn is the noise intensity (we'll choose this later).

$$Q_c(x;\theta) = \sum_i \left(\delta_i \left(\frac{(x_i \cos\theta + y_i \sin\theta + c)^2}{2\sigma^2} \right) + (1 - \delta_i) k_n \right) + K$$

Substituting for delta

- We shall substitute the expected value of □, for a given □
- recall □=(□, c, □)
- E(□_i)=1. P(□_i=1|□)+0....
- Notice that if kn is small and positive, then if distance is small, this value is close to 1 and if it is large, close to zero

$$P\left(\square_i = 1 \mid \square, \mathbf{x}_i\right) = \frac{P\left(\mathbf{x}_i \mid \square_i = 1, \square\right)P\left(\square_i = 1\right)}{P\left(\mathbf{x}_i \mid \square_i = 1, \square\right)P\left(\square_i = 1\right) + P\left(\mathbf{x}_i \mid \square_i = 0, \square\right)P\left(\square_i = 0\right)}$$

$$= \frac{\exp\left(\square 1 / 2\square^2 \left[x_i \cos\square + y_i \sin\square + c\right]^2\right)\square}{\exp\left(\square 1 / 2\square^2 \left[x_i \cos\square + y_i \sin\square + c\right]^2\right)\square + \exp\left(\square k_n\right)\left(1 \square \square\right)}$$

Algorithm for line fitting

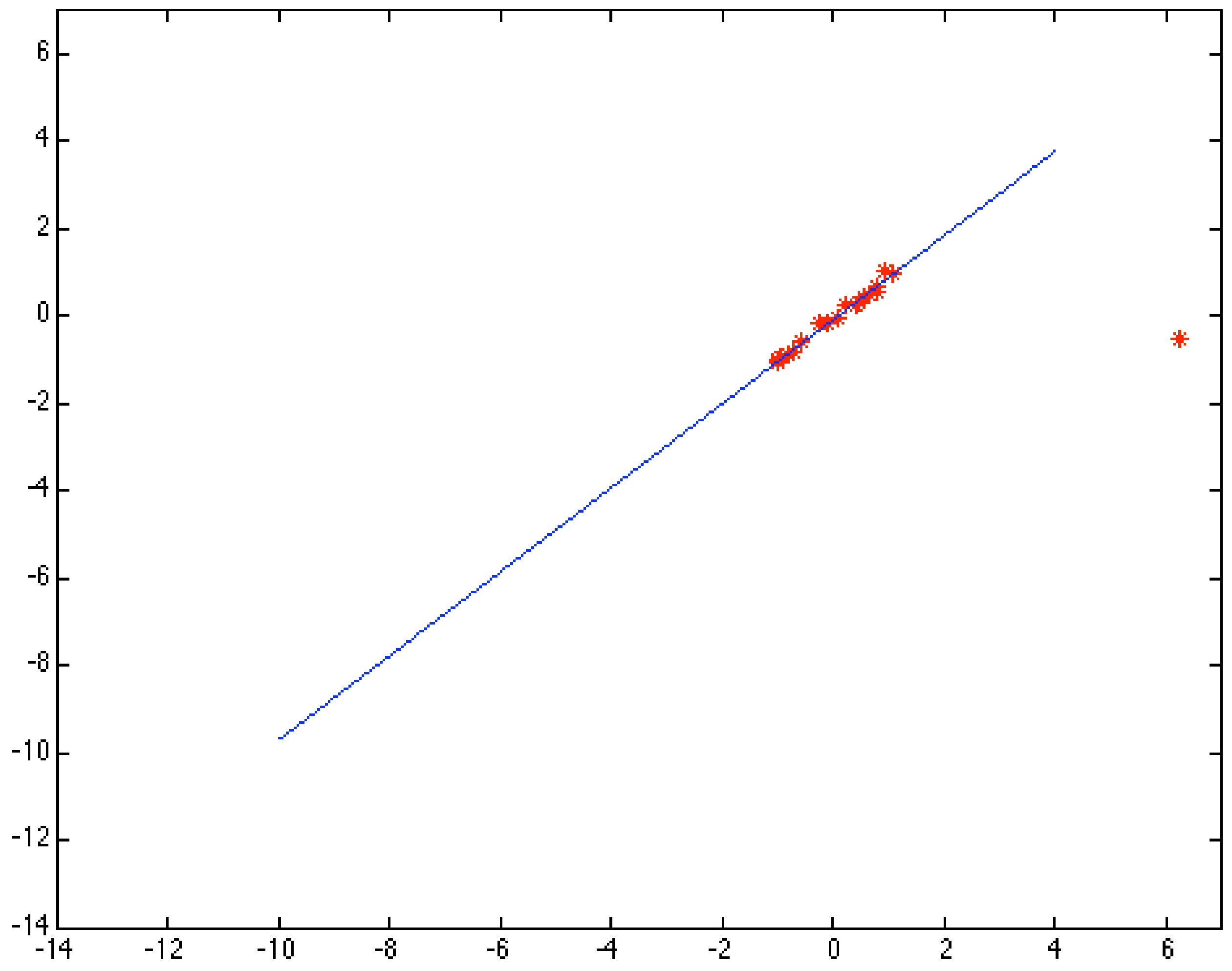
- Obtain some start point

$$\square^{(0)} = \left(\square^{(0)}, c^{(0)}, \square^{(0)}\right)$$

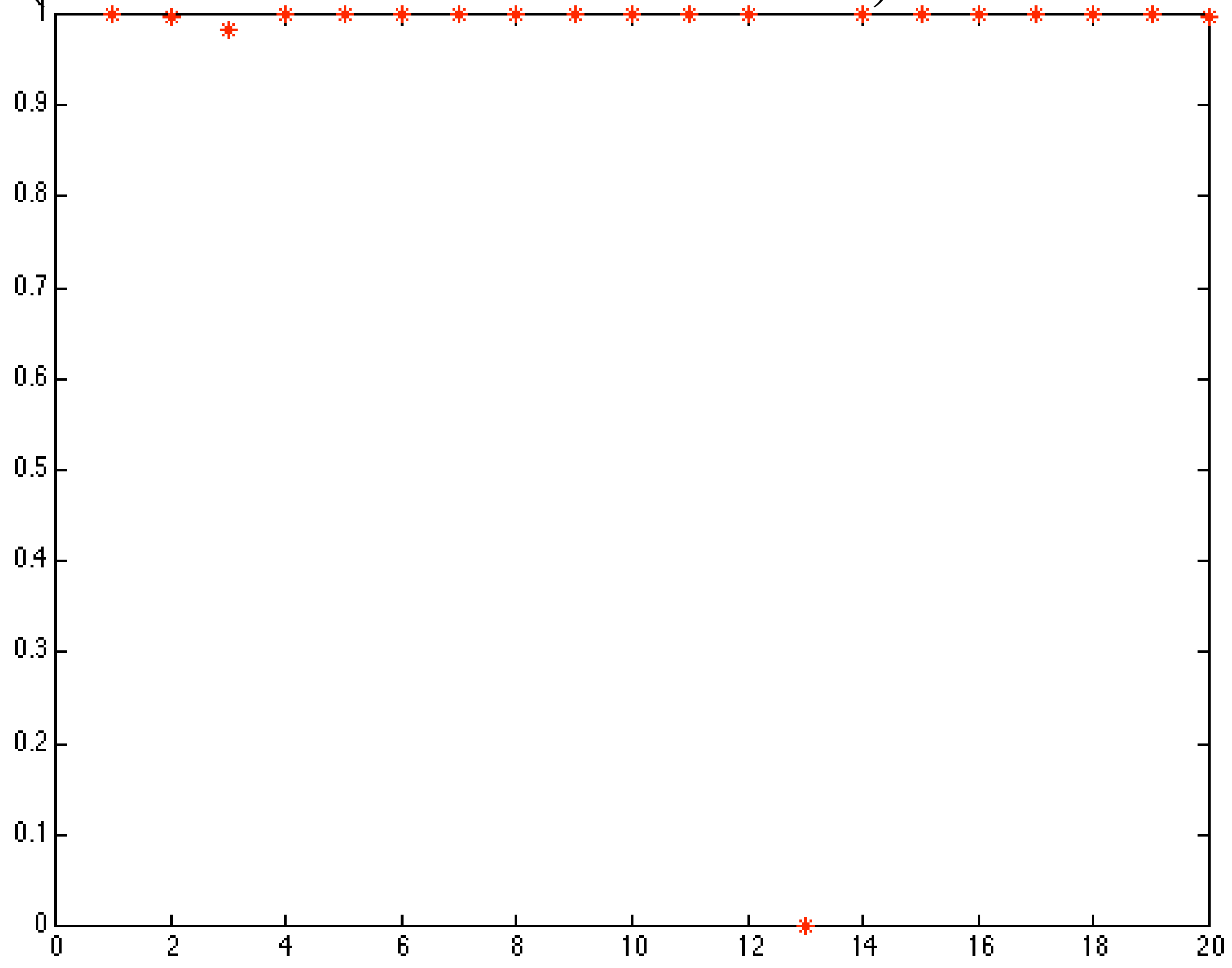
- Now compute □'s using formula above

- Now compute maximum likelihood estimate of $\square^{(1)}$
  - □, c come from fitting to weighted points
  - □ comes by counting

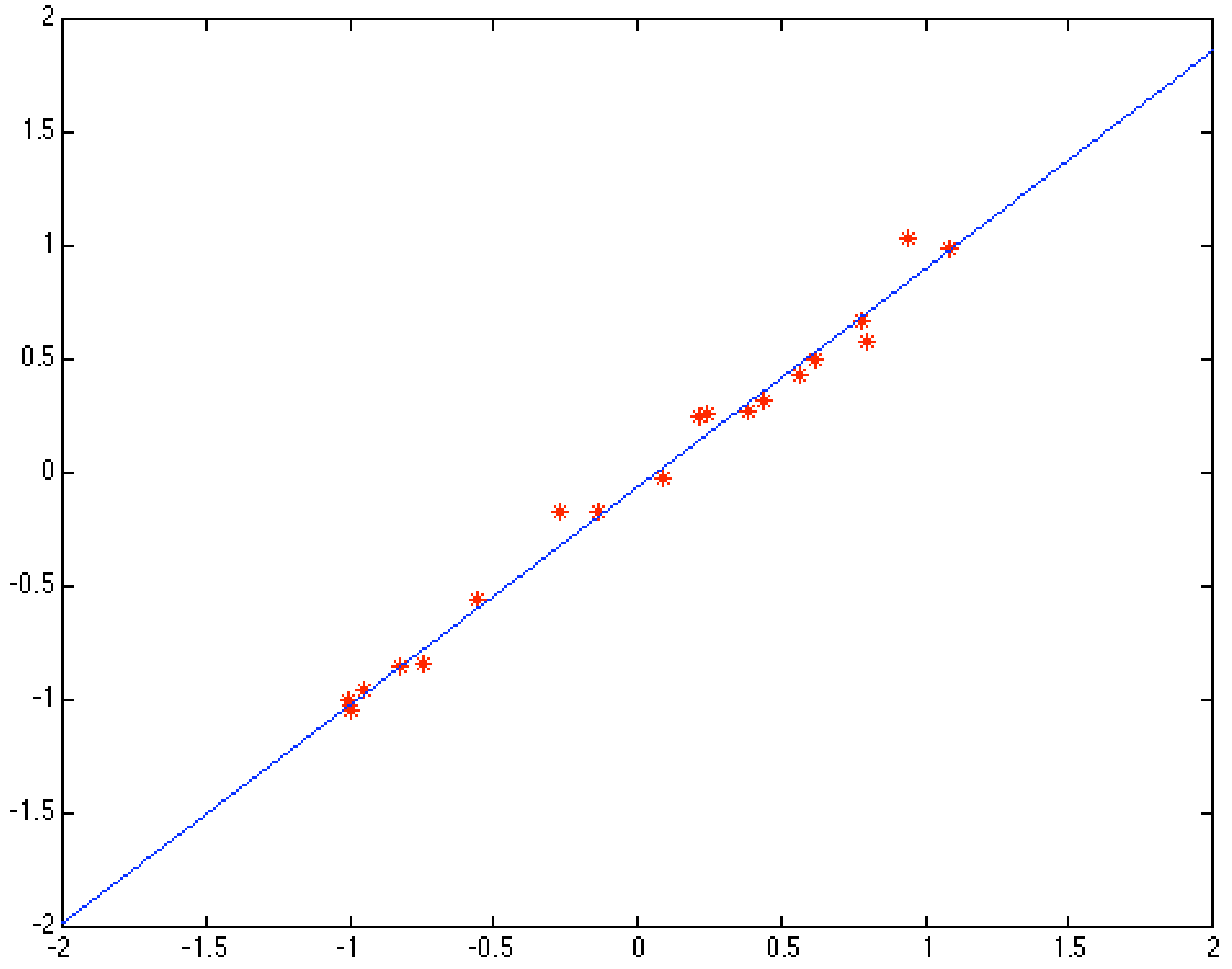
- Iterate to convergence


6
4
2
0
-2
-4
-6
-8
-10
-12
-14
-14
-12
-10
-8
-6
-4
-2
0
2
4
6

The expected values of the deltas at the maximum (notice the one value close to zero).



Closeup of the fit



Choosing parameters

- What about the noise parameter, and the sigma for the line?
  - several methods
    - from first principles knowledge of the problem (seldom really possible)
    - play around with a few examples and choose (usually quite effective, as precise choice doesn't matter much)
  - notice that if kn is large, this says that points very seldom come from noise, however far from the line they lie
    - usually biases the fit, by pushing outliers into the line
    - rule of thumb; its better to fit to the better fitting points, within reason; if this is hard to do, then the model could be a problem

Other examples

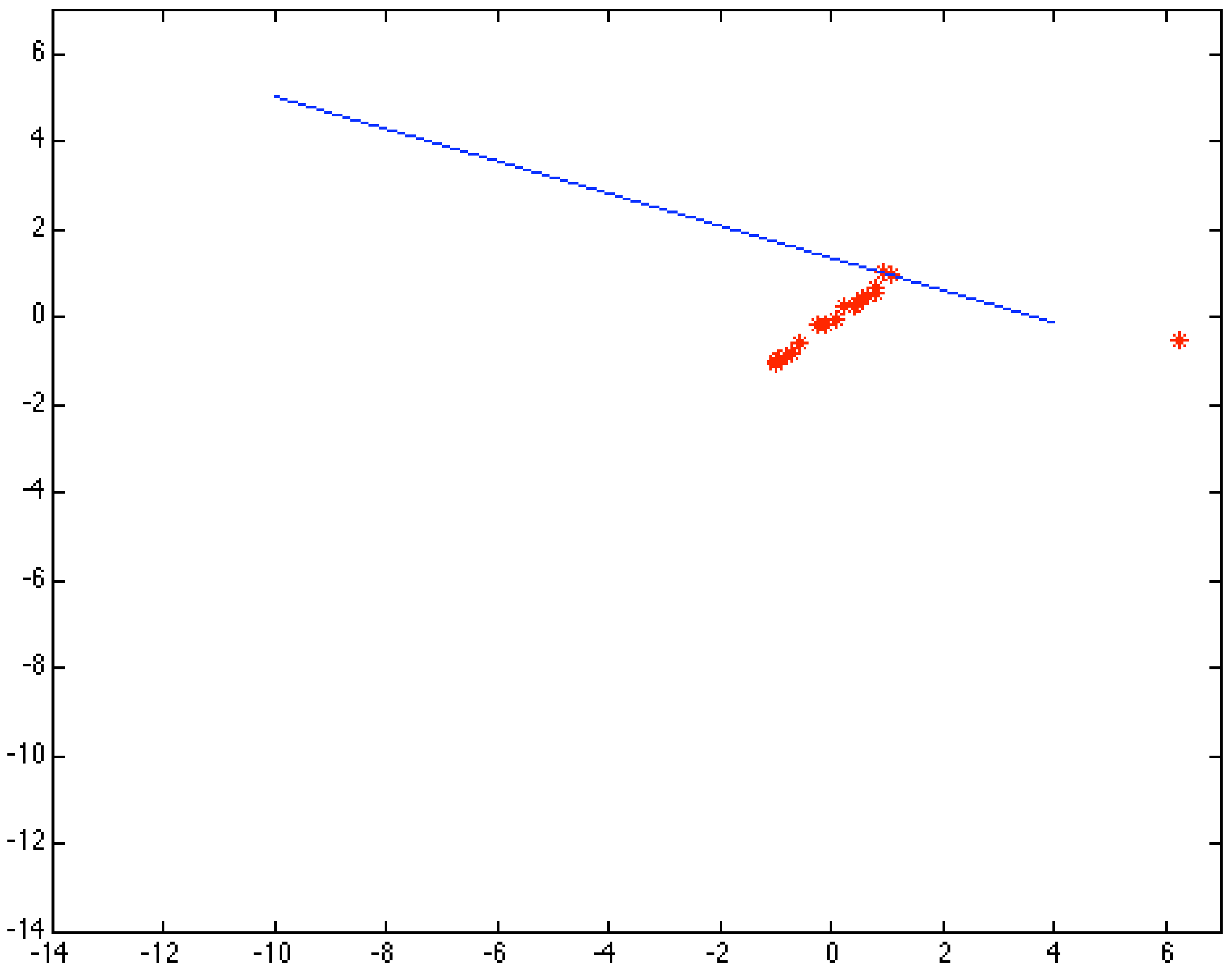
- Segmentation
  - a segment is a gaussian that emits feature vectors (which could contain colour; or colour and position; or colour, texture and position).
  - segment parameters are mean and (perhaps) covariance
  - if we knew which segment each point belonged to, estimating these parameters would be easy
  - rest is on same lines as fitting line

- Fitting multiple lines
  - rather like fitting one line, except there are more hidden variables
  - easiest is to encode as an array of hidden variables, which represent a table with a one where the i'th point comes from the j'th line, zeros otherwise
  - rest is on same lines as above

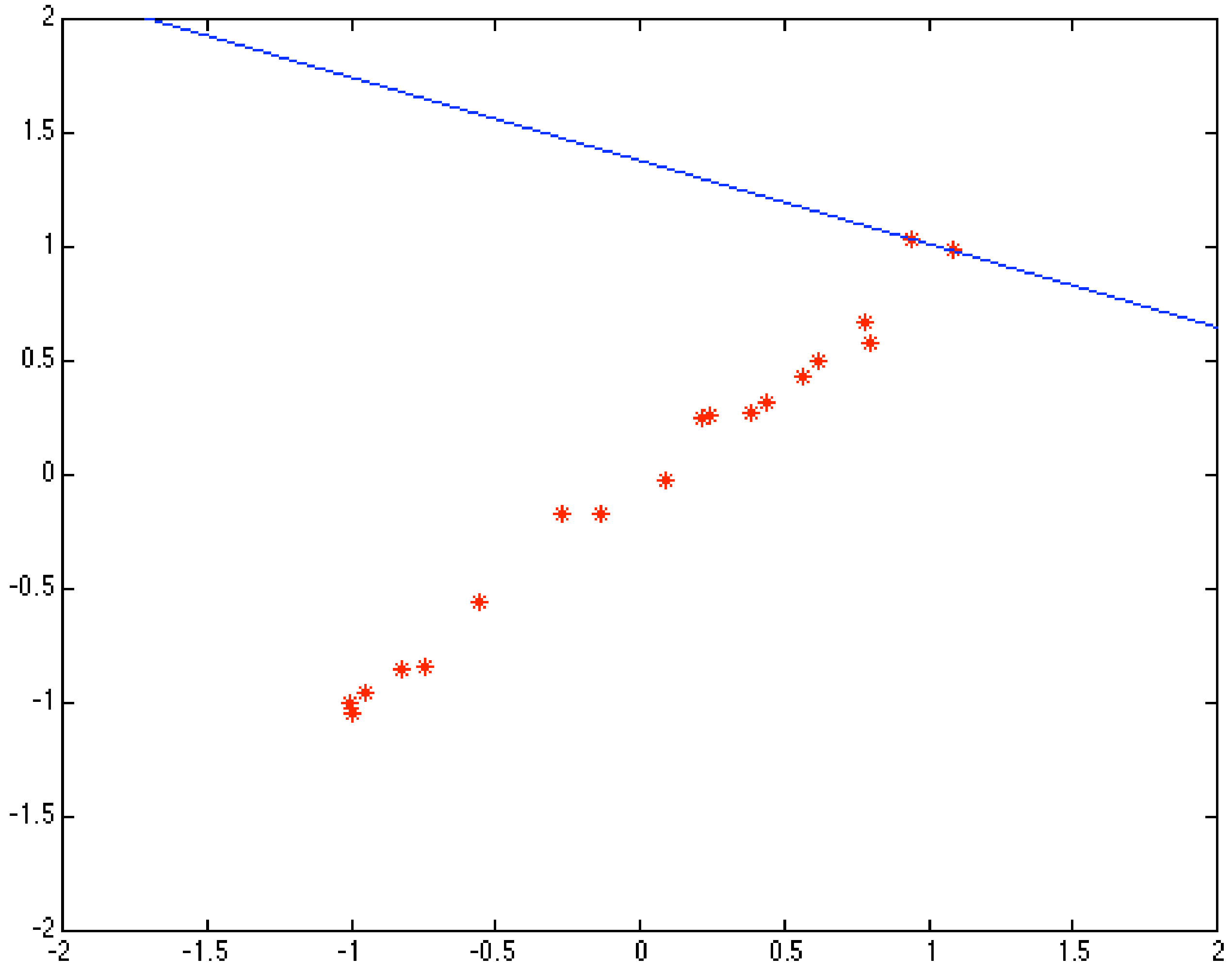
Issues with EM

- Local maxima
  - can be a serious nuisance in some problems
  - no guarantee that we have reached the “right” maximum
- Starting
  - k means to cluster the points is often a good idea

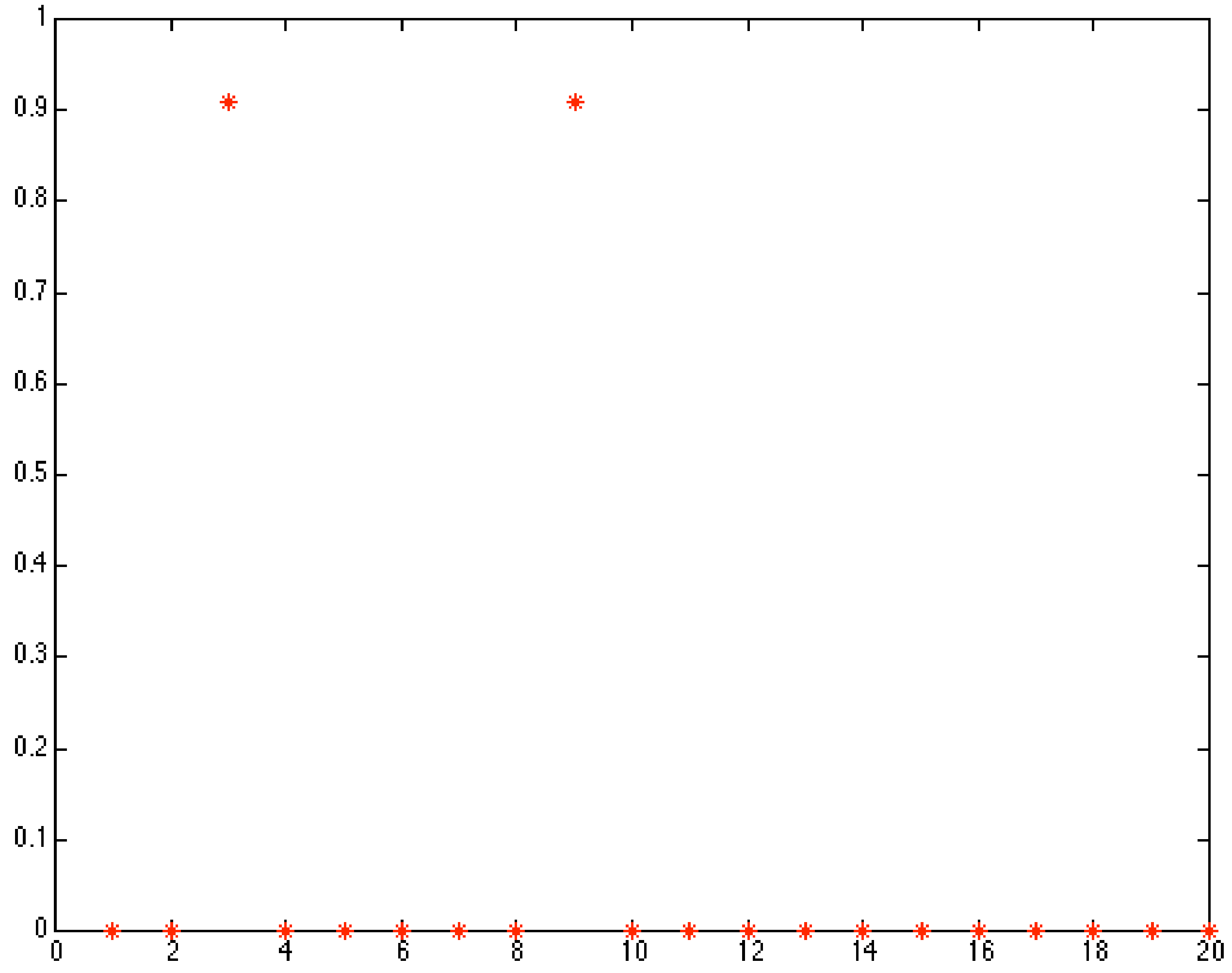
Local maximum



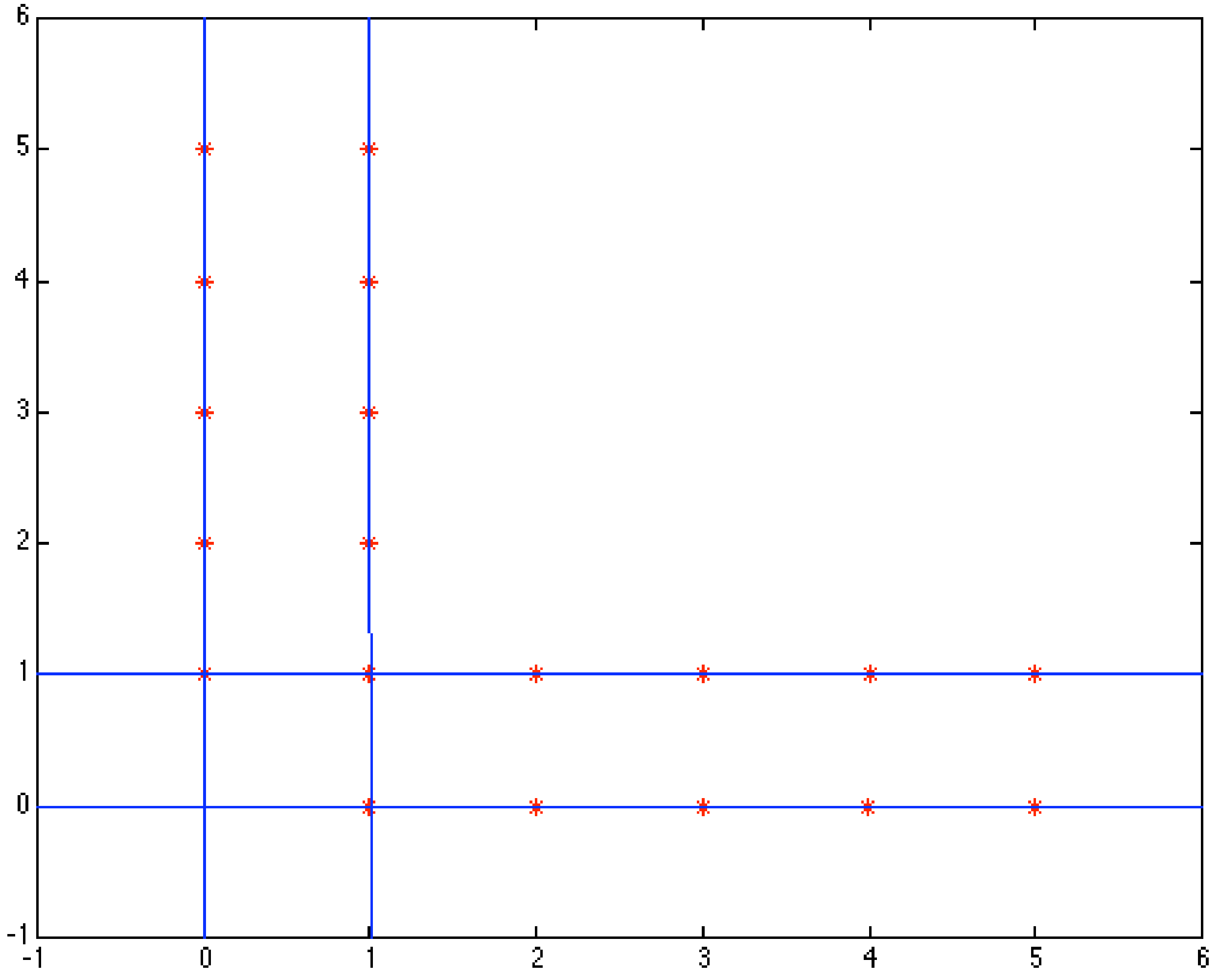
which is an excellent fit to some points



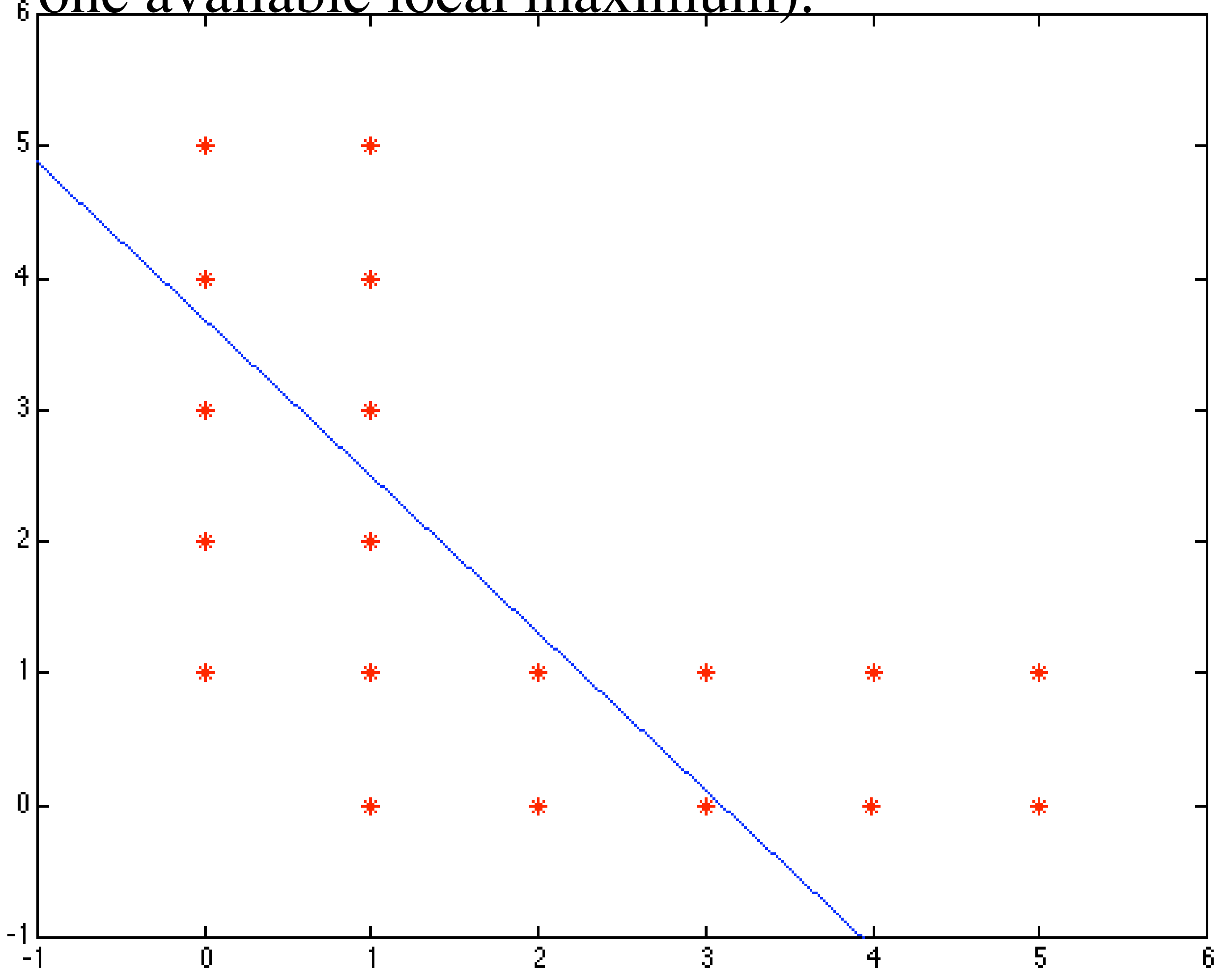
and the deltas for this maximum



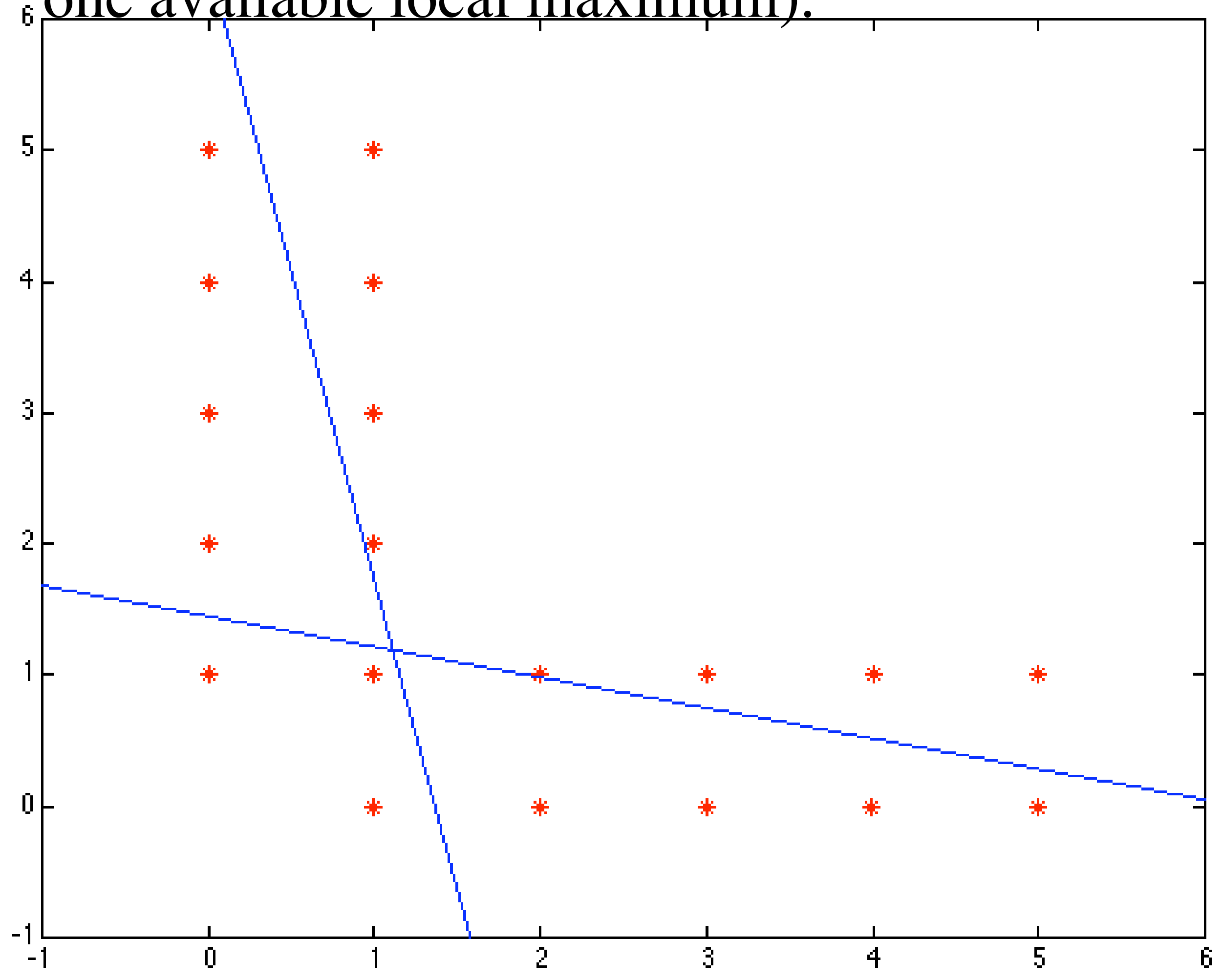
A dataset that is well fitted by four lines



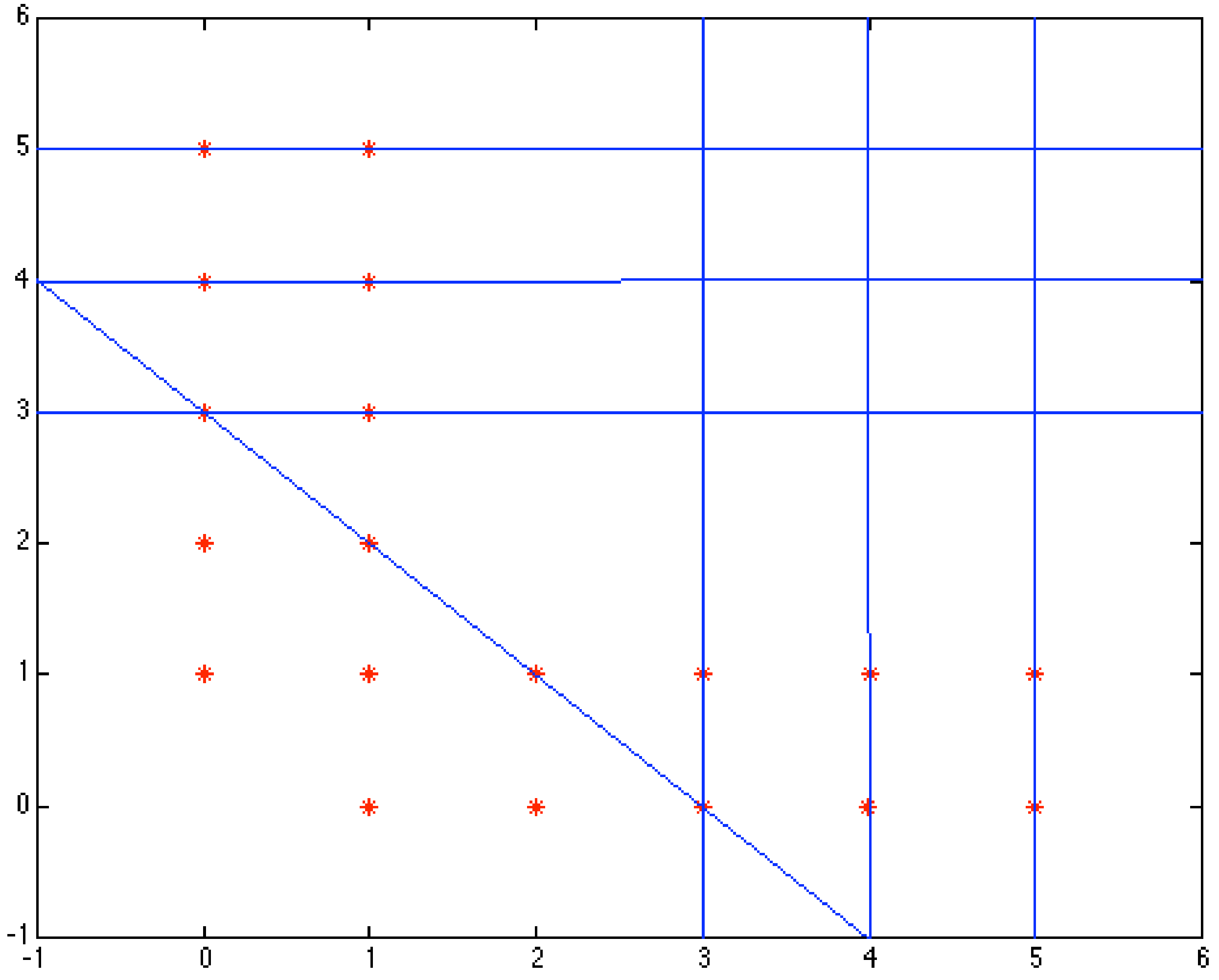
Result of EM fitting, with one line (or at least, one available local maximum).



Result of EM fitting, with two lines (or at least, one available local maximum).



Seven lines can produce a rather logical answer



Motion segmentation with EM

- Model image pair (or video sequence) as consisting of regions of parametric motion
  - affine motion is popular

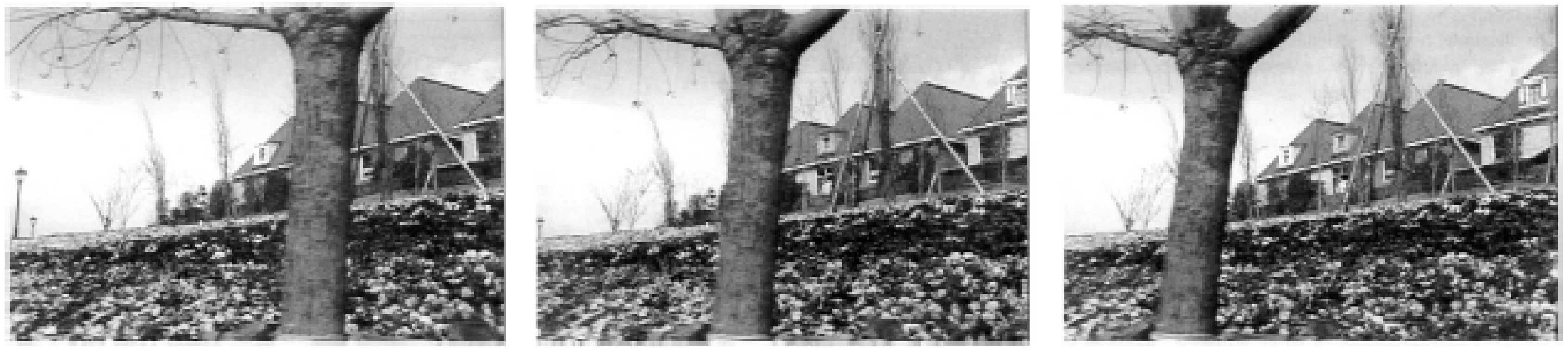
$$\begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

- Now we need to
  - determine which pixels belong to which region
  - estimate parameters

- Likelihood
  - assume

$$I(x, y, t) = I\left(x + v_x, y + v_y, t + 1\right) + noise$$

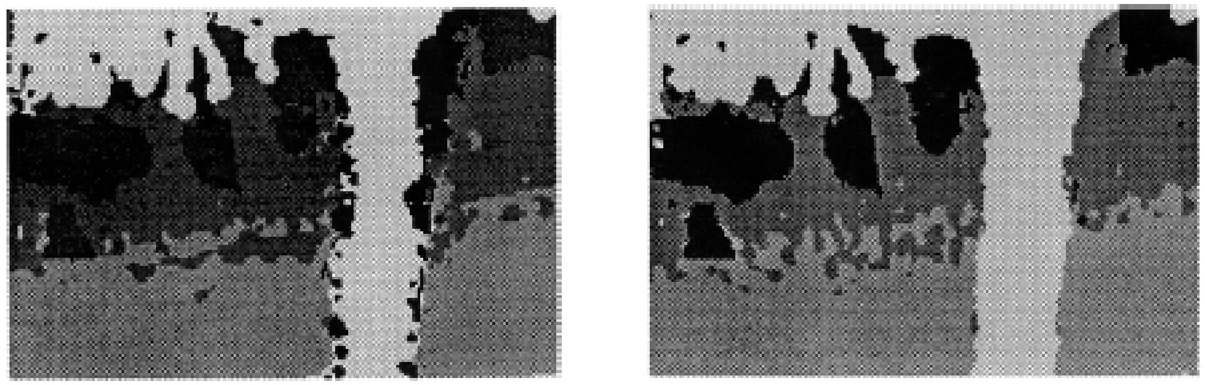
- Straightforward missing variable problem, rest is calculation

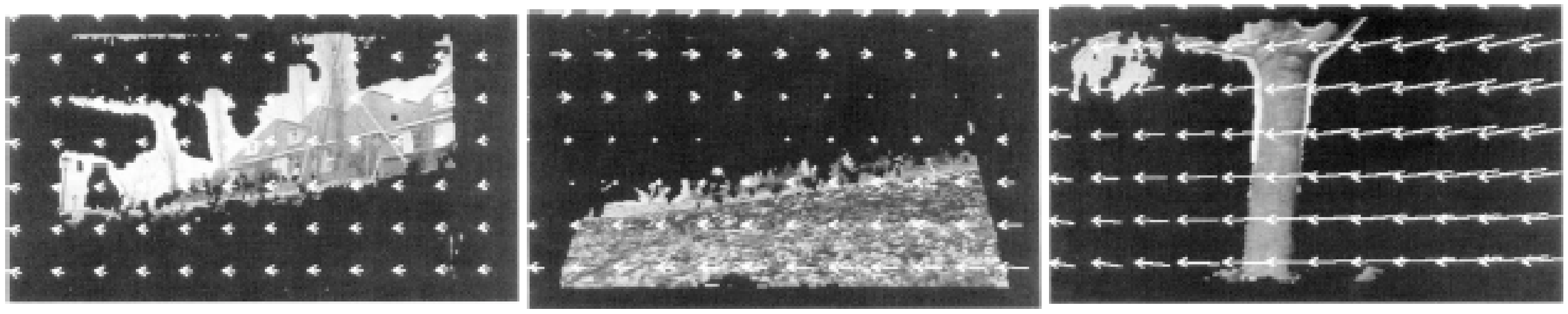


Three frames from the MPEG “flower garden” sequence

Figure from “Representing Images with layers,”, by J. Wang and E.H. Adelson, IEEE Transactions on Image Processing, 1994, c 1994, IEEE

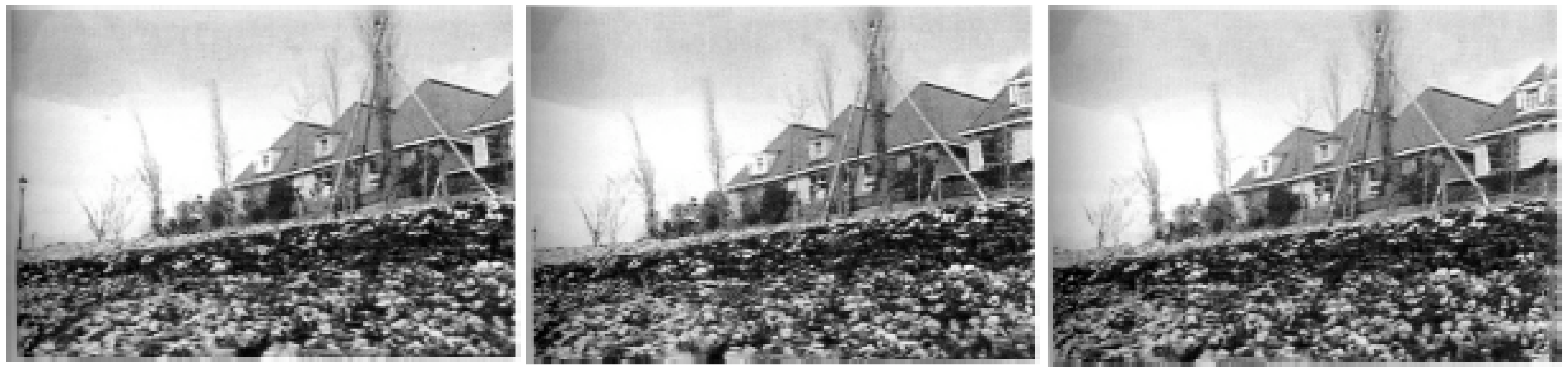
Grey level shows region no. with highest probability





Segments and motion fields associated with them

Figure from “Representing Images with layers,”, by J. Wang and E.H. Adelson, IEEE Transactions on Image Processing, 1994, c 1994, IEEE



If we use multiple frames to estimate the appearance of a segment, we can fill in occlusions; so we can re-render the sequence with some segments removed.

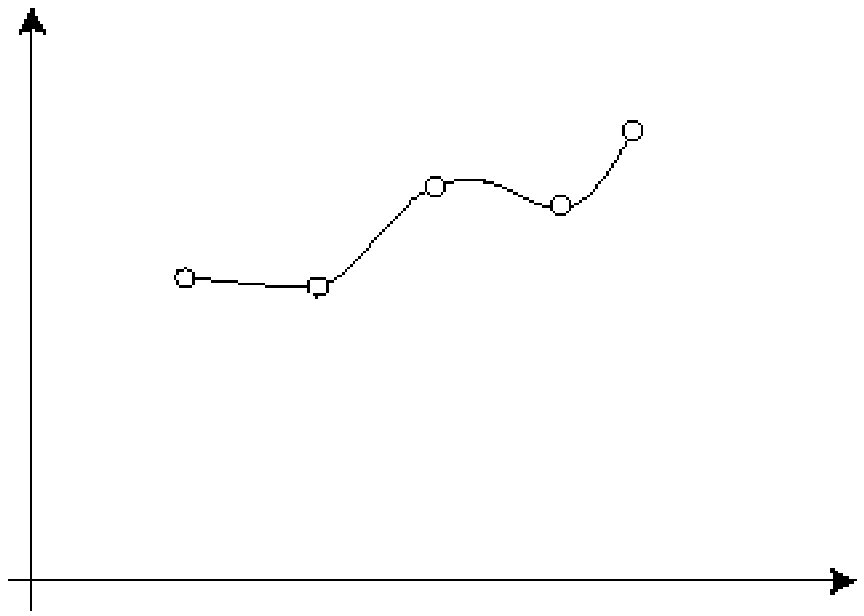
Figure from “Representing Images with layers,”, by J. Wang and E.H. Adelson, IEEE Transactions on Image Processing, 1994, c 1994, IEEE

Some generalities

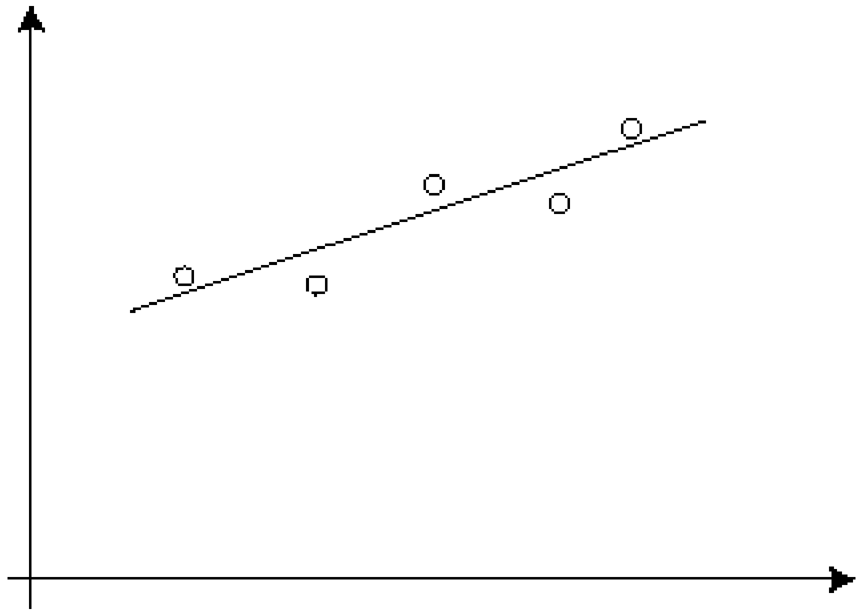
- Many, but not all problems that can be attacked with EM can also be attacked with RANSAC
  - need to be able to get a parameter estimate with a manageably small number of random choices.
  - RANSAC is usually better
- Didn’t present in the most general form
  - in the general form, the likelihood may not be a linear function of the missing variables
  - in this case, one takes an expectation of the likelihood, rather than substituting expected values of missing variables

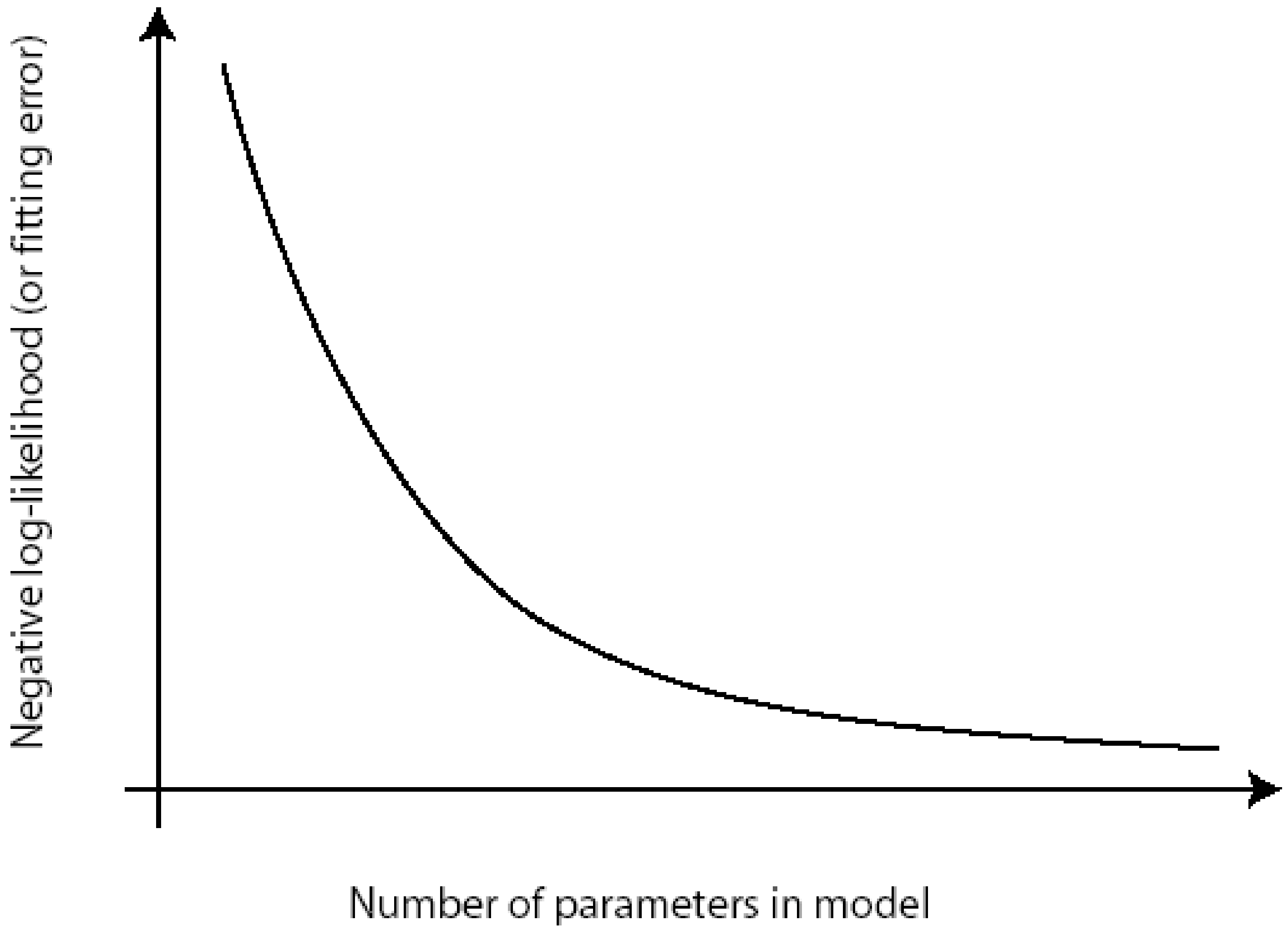
Model Selection

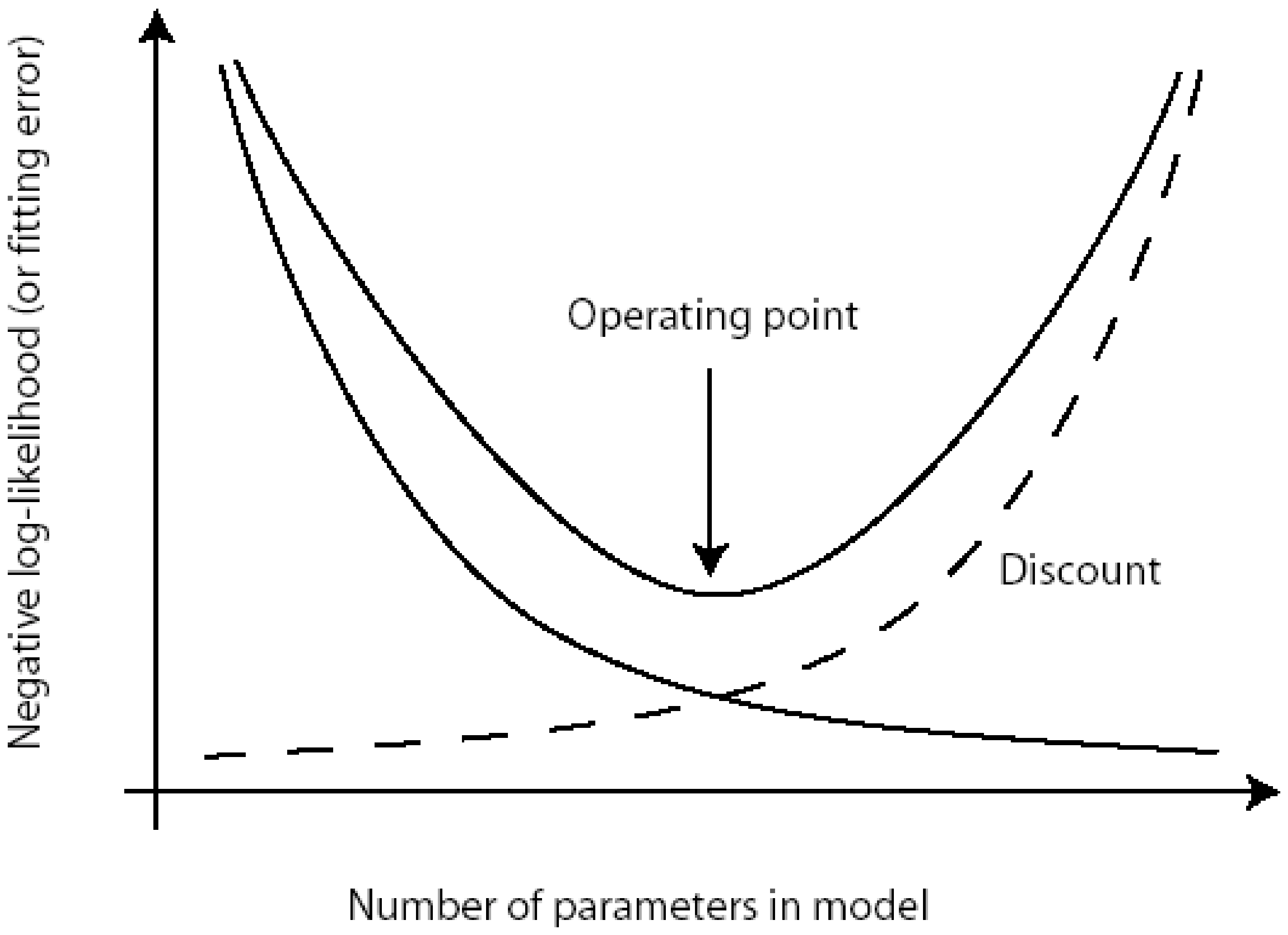
- We wish to choose a model to fit to data
  - e.g. is it a line or a circle?
  - e.g is this a perspective or orthographic camera?
  - e.g. is there an aeroplane there or is it noise?
- Issue
  - In general, models with more parameters will fit a dataset better, but are poorer at prediction
  - This means we can't simply look at the negative log-likelihood (or fitting error)



Top is not necessarily a better fit than bottom
(actually, almost always worse)




Negative log-likelihood (or fitting error)
Number of parameters in model



We can discount the fitting error with some term in the number of parameters in the model.

Discounts

- AIC (an information criterion)
  - choose model with smallest value of

$$-2L\left(D;\theta^*\right)+2p$$

  - p is the number of parameters

- BIC (Bayes information criterion)
  - choose model with smallest value of

$$-2L\left(D;\theta^*\right)+p\log N$$

  - N is the number of data points
- Minimum description length
  - same criterion as BIC, but derived in a completely different way

Cross-validation

- Split data set into two pieces, fit to one, and compute negative log-likelihood on the other
- Average over multiple different splits
- Choose the model with the smallest value of this average
- The difference in averages for two different models is an estimate of the difference in KL divergence of the models from the source of the data

Model averaging

- Very often, it is smarter to use multiple models for prediction than just one
- e.g. motion capture data
  - there are a small number of schemes that are used to put markers on the body
  - given we know the scheme S and the measurements D, we can estimate the configuration of the body X

- We want

$$P(X \mid D) = P(X \mid S_1, D)P(S_1 \mid D) + P(X \mid S_2, D)P(S_2 \mid D) + P(X \mid S_3, D)P(S_3 \mid D)$$

- If it is obvious what the scheme is from the data, then averaging makes little difference
- If it isn't, then not averaging underestimates the variance of X --- we think we have a more precise estimate than we do.

Learning as Model fitting

Model Fitting, Learning

- Model: Mapping between observations and world properties.
  - P(o,s)
  - Parametric:

    P(o,s;ø) where ø are parameters that need to be fit (learned) from observations (data)
  - Density Estimation:

    Learn P(o,s) from examples. E.g. Histogram, KDE

    Mixture of parametric densities.

What’s the problem?

- Two sets of hidden parameters
  - Goal, find *argmax*(s) P(s|o,ø)
  - But P(s|o;ø) depends on ø, thus different ø, different inference of s.
  - Optimize ø? Yes, but optimal ø only good for one s.
  - Need to optimize both at the same time.
  - One Solution: EM algorithm

Learning templates for categories

Assume a set of labelled examples for stimuli from categories C = 1, 2, etc

Stimuli are fixed images in gaussian noise

$$s = I1 + N$$

Then labelled examples can be used to learn a parametric model

P(s|C) = N(□,K). Let K = □^{2}I. How do we estimate □ across trials?

$$p(\mu|\mathcal{D}) = \frac{p(\mathcal{D}|\mu)\,p(\mu)}{\int p(\mathcal{D}|\mu)\,p(\mu)\,d\mu}$$

$$= \alpha \prod_{k=1}^{n} p(x_k|\mu)\,p(\mu),$$

$$\hat{\mu}_n = \frac{1}{n}\sum_{k=1}^{n} x_k.$$

$$\mu_n = \left(\frac{n\sigma_0^2}{n\sigma_0^2+\sigma^2}\right)\hat{\mu}_n + \frac{\sigma^2}{n\sigma_0^2+\sigma^2}\mu_0$$

$$p(\mu|\mathcal{D}) = \frac{1}{\sqrt{2\pi}\,\sigma_n}\exp\left[-\frac{1}{2}\left(\frac{\mu-\mu_n}{\sigma_n}\right)^2\right].$$

$$\sigma_n^2 = \frac{\sigma_0^2\sigma^2}{n\sigma_0^2+\sigma^2}.$$

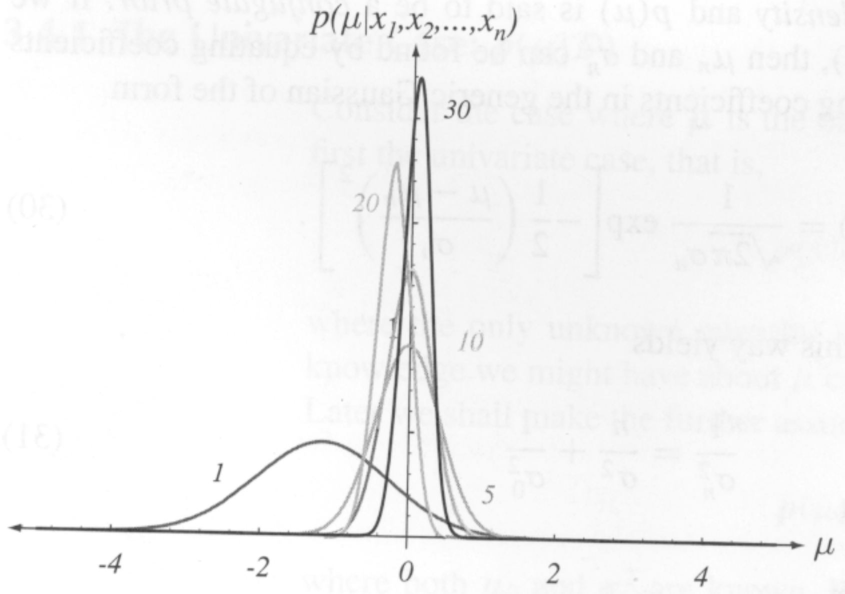
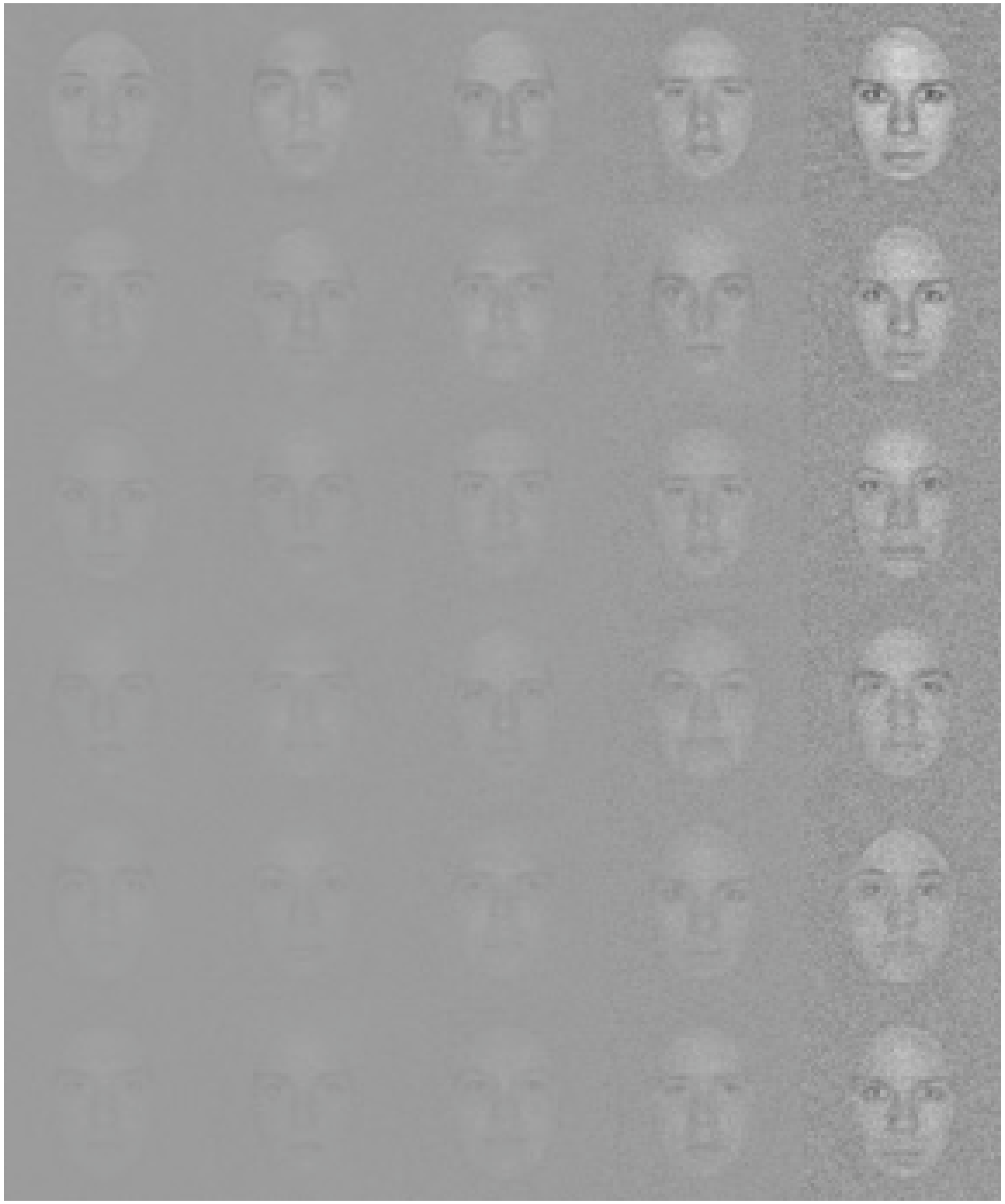


FIGURE 3.2. Bayesian learning of the mean of normal distributior pproach



EM algorithm

Normal maximum likelihood

$$p(\mathcal{X}|\Theta) = \prod_{i=1}^{N} p(\mathbf{x}_i|\Theta) = \mathcal{L}(\Theta|\mathcal{X}).$$

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}}\ \mathcal{L}(\Theta|\mathcal{X}).$$

What to do with $p(\mathcal{X},\mathcal{Y}|\Theta)$ Y unknown?

Iterate,

1) set value for Θ; average over Y
2) maximize Θ; use as next value

$$Q(\Theta,\Theta^{(i-1)}) = E\left[\log p(\mathcal{X},\mathcal{Y}|\Theta)|\mathcal{X},\Theta^{(i-1)}\right]$$

1) $$E\left[\log p(\mathcal{X},\mathcal{Y}|\Theta)|\mathcal{X},\Theta^{(i-1)}\right] = \int_{\mathbf{y}\in\Upsilon} \log p(\mathcal{X},\mathbf{y}|\Theta) f(\mathbf{y}|\mathcal{X},\Theta^{(i-1)})\,d\mathbf{y}.$$

2) $$\Theta^{(i)} = \underset{\Theta}{\operatorname{argmax}}\ Q(\Theta,\Theta^{(i-1)}).$$

Density Est: Mixture of Densities

$$p(\mathbf{x}|\Theta) = \sum_{i=1}^{M} \alpha_i p_i(\mathbf{x}|\theta_i)$$

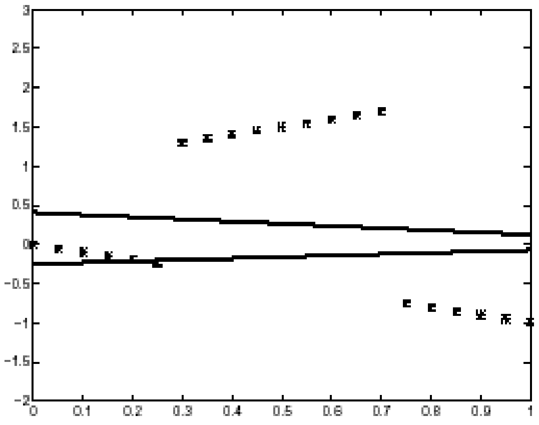
$$\Theta = (\alpha_1, \ldots, \alpha_M, \theta_1, \ldots, \theta_M)$$

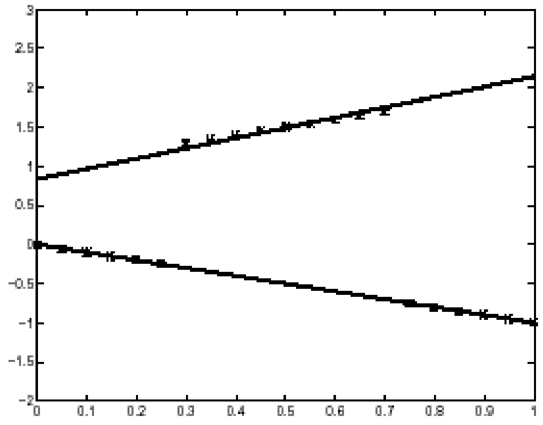
e.g. □ = { □, □ }

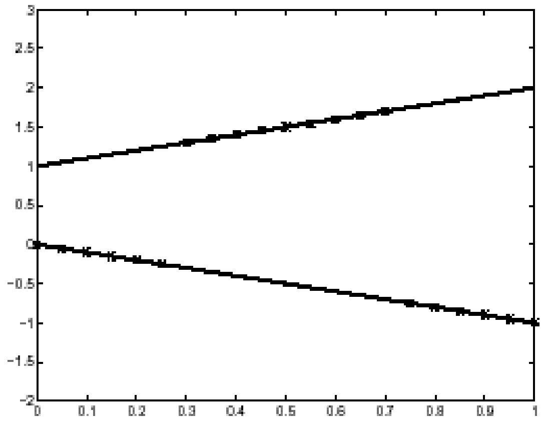
$$\log(\mathcal{L}(\Theta|\mathcal{X})) = \log \prod_{i=1}^{N} p(x_i|\Theta) = \sum_{i=1}^{N} \log \left(\sum_{j=1}^{M} \alpha_j p_j(x_i|\theta_j) \right)$$

$$= \sum_{i=1}^{N} \log \left(P(x_i|y_i) P(y) \right) = \sum_{i=1}^{N} \log \left(\alpha_{y_i} p_{y_i}(x_i|\theta_{y_i}) \right)$$

Mixture Model Applications







Motion Estimation

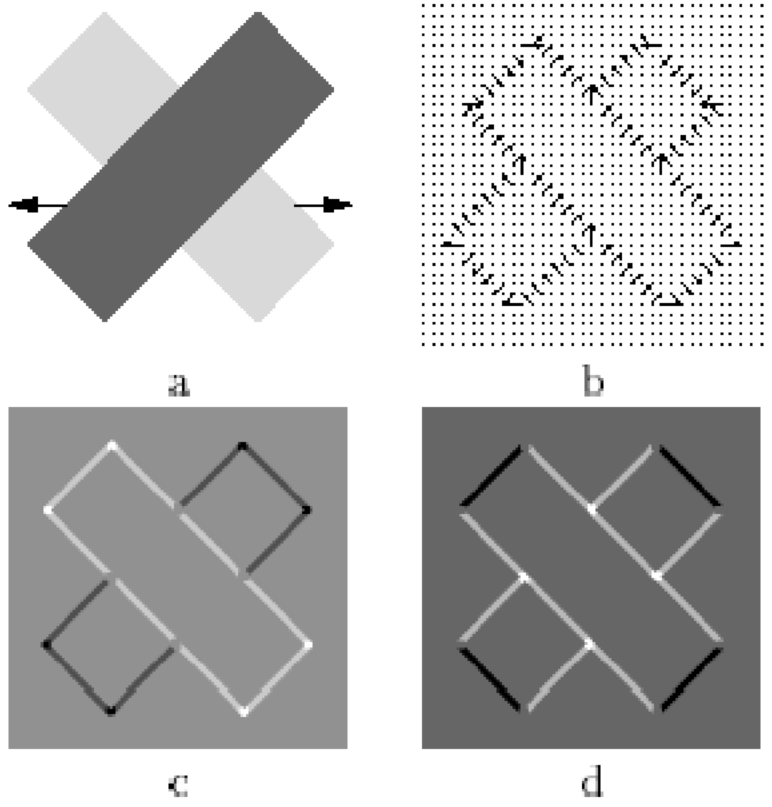
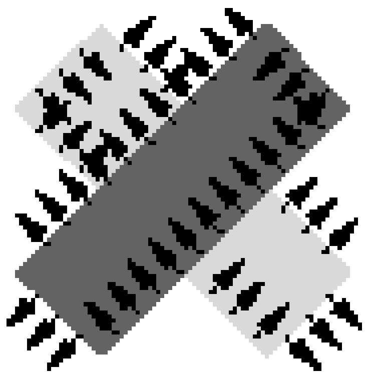
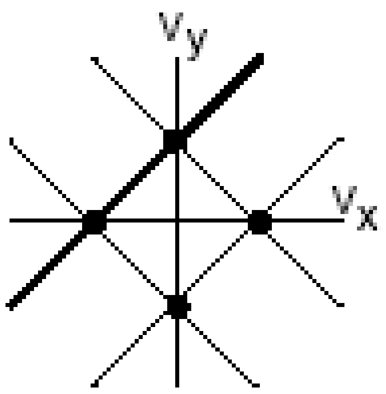


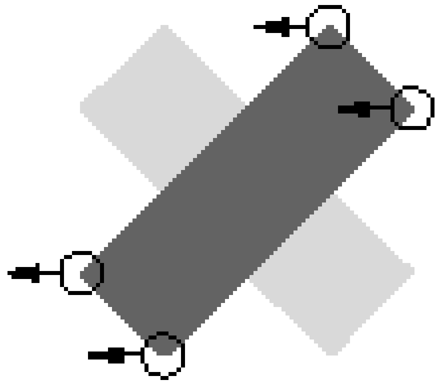
Figure 1: **a** A simple image sequence which causes problems for traditional motion estimation algorithms. **b** Least squares optical flow shown as an arrow plot **c** Least squares optical flow horizontal component. **d** Least squares optical flow vertical component.

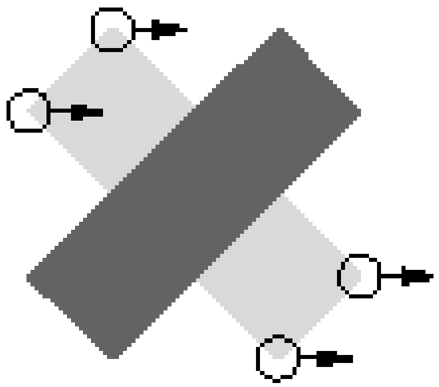
Motion Estimation

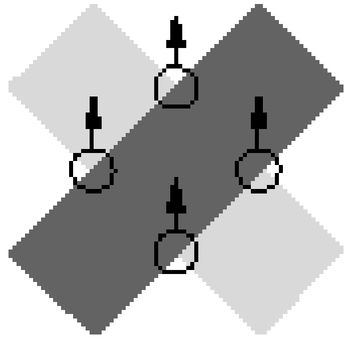

a

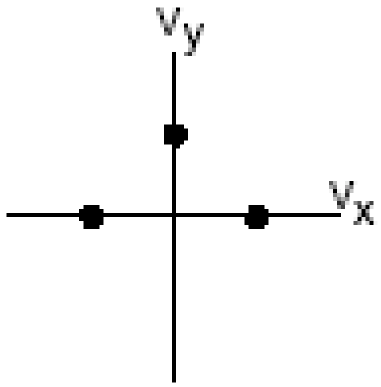


b


c


d


e



f

Motion Estimation

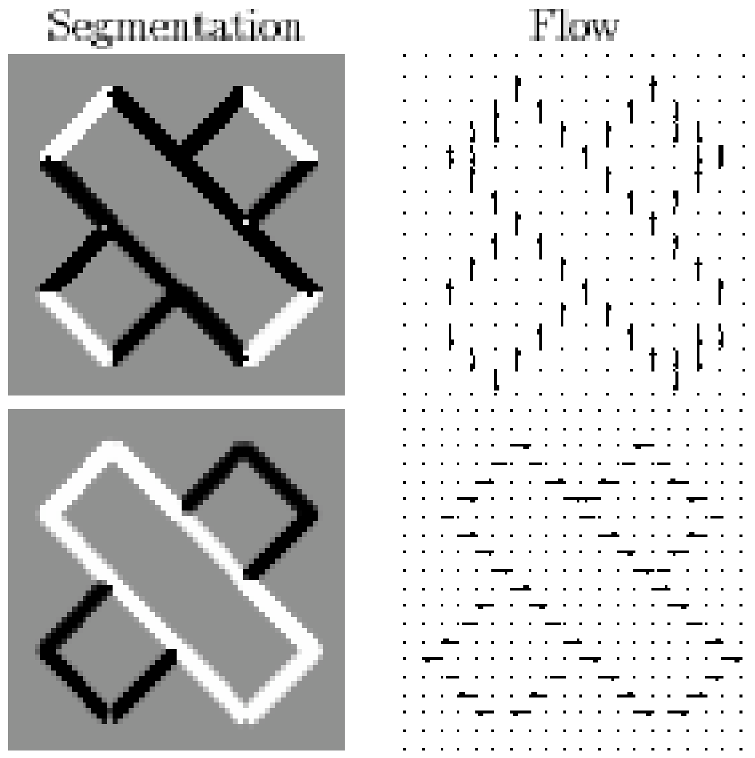


Figure 8: The two interpretations arrived at by the classical EM algorithm randomly on alternate runs. **Top:** The incorrect interpretation. Segmentation on the left and motion field on the right. With the exception of the corners, this interpretation explains all the data. **Bottom:** The correct interpretation. In both cases all untextured regions are ambiguous.