6.891

Computer Vision and Applications

Prof. Trevor. Darrell

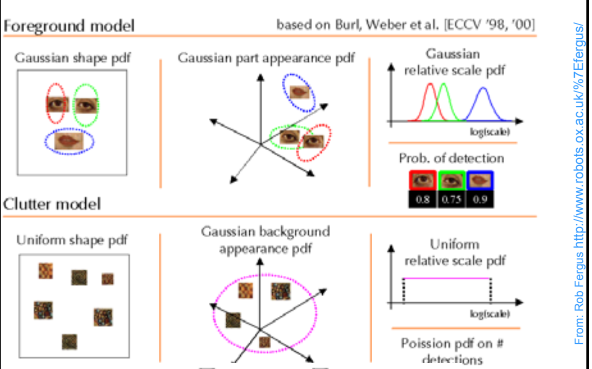
Lecture 15: Fitting and Segmentation

Readings: F&P Ch 15.3-15.5,16

Last time: "Segmentation and Clustering (Ch. 14)"

- Supervised->Unsupervised Category Learning needs segmentation
- K-Means
- Mean Shift
- Graph cuts
- Hough transform

Generative probabilistic model

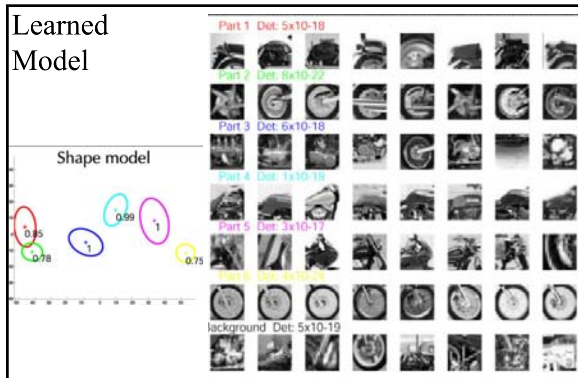


$$p(\mathcal{X},\mathcal{A}|\theta) = \sum_{\mathbf{h}\in H} p(\mathcal{X},\mathcal{A},\mathbf{h}|\theta) = \sum_{\mathbf{h}\in H} \underbrace{p(\mathcal{A}|\mathcal{X},\mathbf{h},\theta)}_{Appearance}\ \underbrace{p(\mathcal{X}|\mathbf{h},\theta)}_{Shape}$$

[Slide from Bradsky & Thrun, Stanford]

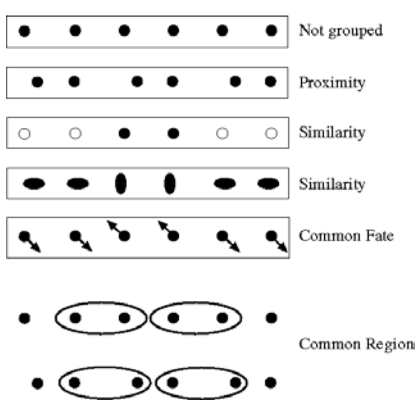
From: Rob Fergus http://www.robots.ox.ac.uk/%7Efergus/

Learned Model

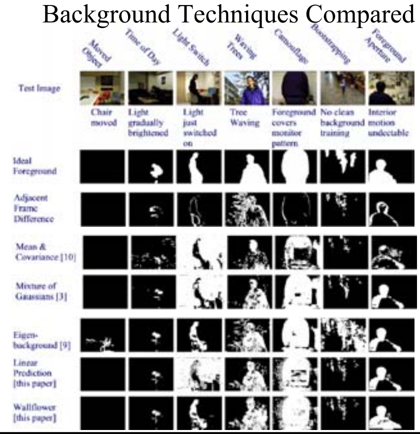


The shape model. The mean location is indicated by the cross, with the ellipse showing the uncertainty in location. The number by each part is the probability of that part being present.

From: Rob Fergus http://www.robots.ox.ac.uk/%7Efergus/



Background Techniques Compared



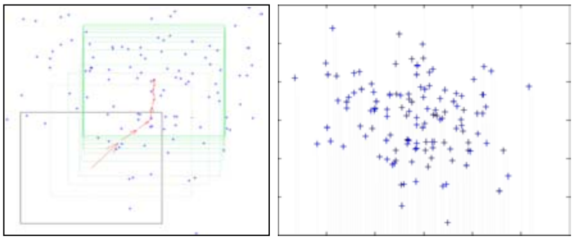
From the Wallflower Paper

Mean Shift Algorithm

Mean Shift Algorithm

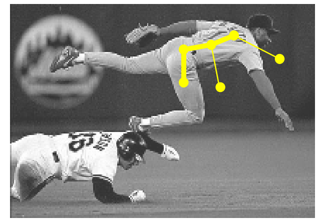
1. Choose a search window size.
2. Choose the initial location of the search window.
3. Compute the mean location (centroid of the data) in the search window.
4. Center the search window at the mean location computed in Step 3.
5. Repeat Steps 3 and 4 until convergence.

The mean shift algorithm seeks the "mode" or point of highest density of a data distribution:



Graph-Theoretic Image Segmentation

Build a weighted graph G=(V,E) from image



V: image pixels

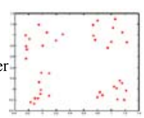
E: connections between pairs of nearby pixels

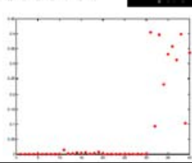
Eigenvectors and affinity clusters

- Simplest idea: we want a vector a giving the association between each element and a cluster
- We want elements within this cluster to, on the whole, have strong affinity with one another
- We could maximize
- But need the constraint

$$a^T A a \qquad a^T a = 1$$

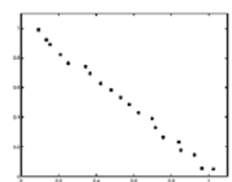
- This is an eigenvalue problem - choose the eigenvector of A with largest eigenvalue



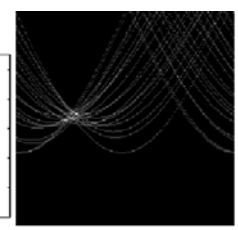


- Shi/Malik, Scott/Longuet-Higgens, Ng/Jordan/Weiss, etc.

Hough transform



tokens



votes

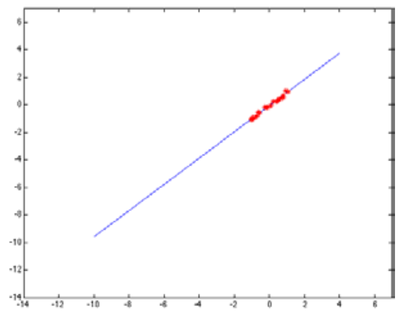
Today "Fitting and Segmentation (Ch. 15)"

- Robust estimation
- EM
- Model Selection
- RANSAC

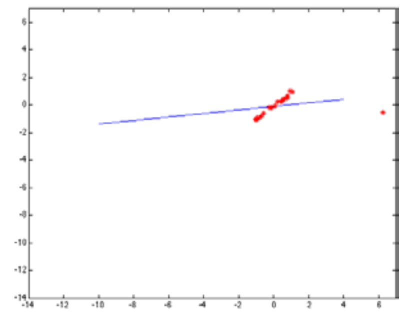
(Maybe "Segmentation I" and "Segmentation II" would be a better way to split these two lectures!)

Robustness

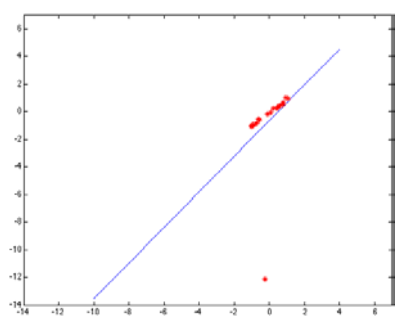
- Squared error can be a source of bias in the presence of noise points
  - One fix is EM - we'll do this shortly
  - Another is an M-estimator
    - Square nearby, threshold far away
  - A third is RANSAC
    - Search for good points



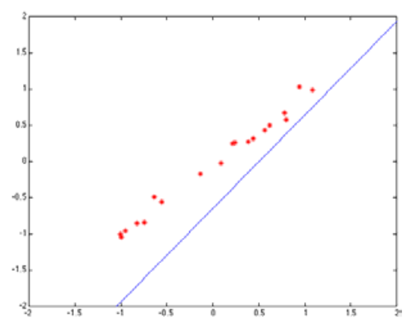
13



14



15



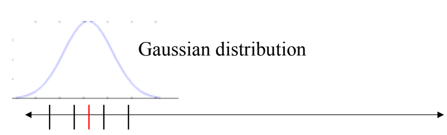
16

Robust Statistics

- Recover the best fit to the majority of the data.
- Detect and reject outliers.

17

Estimating the mean



$$\mu = \frac{1}{N}\sum_{i=1}^{N} d_i$$

Mean is the optimal solution to:

$$\min_{\mu} \sum_{i=1}^{N} \underbrace{(d_i - \mu)}_{\text{residual}}{}^2$$

18

Estimating the Mean

The mean maximizes this likelihood:

$$\max_{\mu} p(d_i \mid \mu) = \frac{1}{\sqrt{2\pi}\sigma} \prod_{i=1}^{N} \exp(-\frac{1}{2}(d_i - \mu)^2 / \sigma^2)$$

The negative log gives (with sigma=1):

$$\min_{\mu} \sum_{i=1}^{N} (d_i - \mu)^2$$

"least squares" estimate

Estimating the mean

Estimating the mean

What happens if we change just one measurement?

$$\mu' = \mu + \frac{\Delta}{N}$$

With a single "bad" data point I can move the mean arbitrarily far.

Influence

Breakdown point

* percentage of outliers required to make the solution arbitrarily bad.

Least squares:

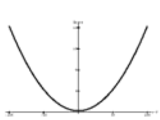
* influence of an outlier is linear (Δ/N)
* breakdown point is 0% -- not robust!

What about the median?

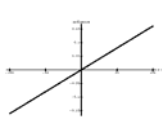
What's Wrong?

$$\min_{\mu} \sum_{i=1}^{N} (d_i - \mu)^2$$

Outliers (large residuals) have too much influence.



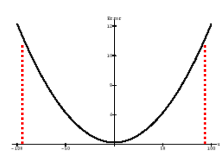
$\rho(x) = x^2$

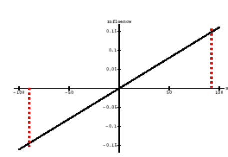


$\psi(x) = 2x$

Approach

Influence is proportional to the derivative of the ρ function.





Want to give less influence to points beyond some value.

Approach

$$\min_{\mu} \sum_{i=1}^{N} \rho(d_i - \mu, \sigma)$$

Robust error function

Scale parameter

Replace $\rho(x,\sigma) = \left(\frac{x}{\sigma}\right)^2$

with something that gives less influence to outliers.

Approach

$$\min_{\mu} \sum_{i=1}^{N} \rho(d_i - \mu, \sigma)$$

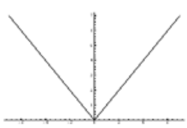
Robust error function

Scale parameter

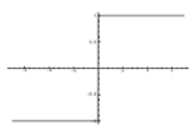
No closed form solutions!

- Iteratively Reweighted Least Squares
- Gradient Descent

L1 Norm

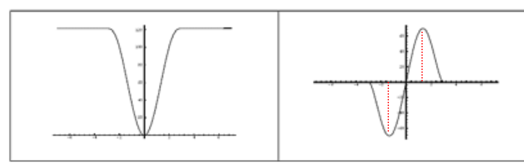


$\rho(x) = |x|$



$\psi(x) = \text{sign}(x)$

Redescending Function



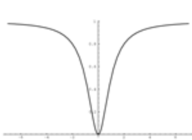
Tukey's biweight.

Beyond a point, the influence begins to decrease.

Beyond where the second derivative is zero – outlier points

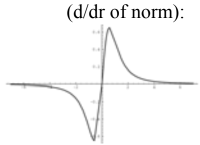
Robust Estimation

Geman-McClure function works well.
Twice differentiable, redescending.

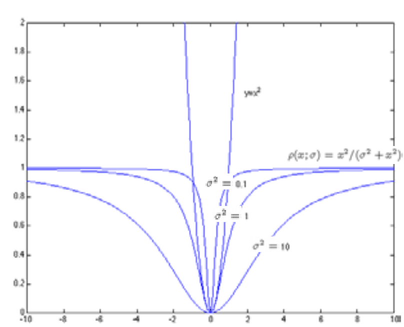


$$\rho(r,\sigma) = \frac{r^2}{\sigma^2 + r^2}$$

Influence function (d/dr of norm):



$$\psi(r,\sigma) = \frac{2r\sigma^2}{(\sigma^2 + r^2)^2}$$



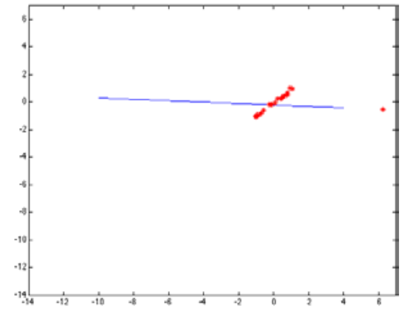
Robust scale

Scale is critical!

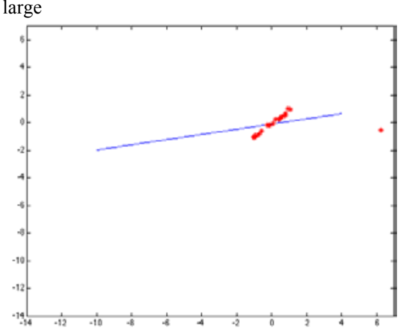
Popular choice:

$$\sigma^{(n)} = 1.4826 \text{ median}_i \, |r_i^{(n)}(x_i; \theta^{(n-1)})|$$

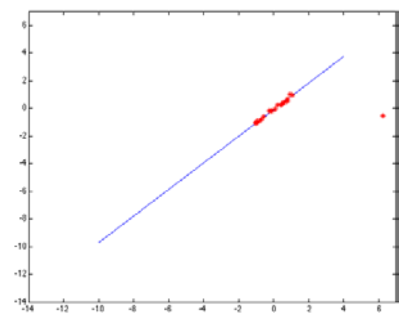
Too small



Too large



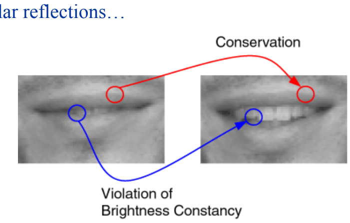
Just right



Example: Motion

Assumption: Within a finite image region, there is only a single motion present.

Violated by: motion discontinuities, shadows, transparency, specular reflections…



Violations of brightness constancy result in large residuals:

Estimating Flow

Minimize:

$$E(\mathbf{a}) = \sum_{\mathbf{x} \in R} \rho(\mathrm{I}_x u(\mathbf{x};\mathbf{a}) + \mathrm{I}_u v(\mathbf{x};\mathbf{a}) + \mathrm{I}_t, \sigma)$$

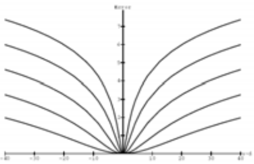
Parameterized models provide strong constraints:

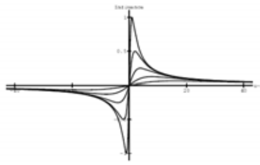
* Hundred, or thousands, of constraints.
* Handful (e.g. six) unknowns.

Can be very accurate (when the model is good)!

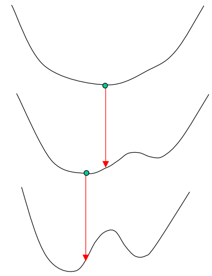
Deterministic Annealing

Start with a "quadratic" optimization problem and gradually reduce outliers.



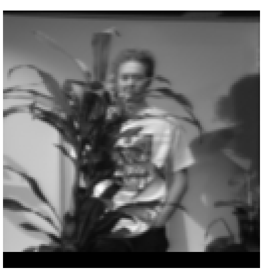


Continuation method

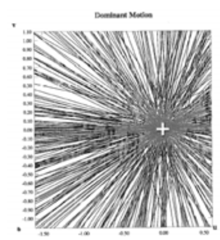


GNC: Graduated Non-Convexity

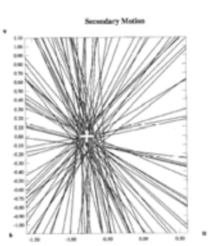
Fragmented Occlusion

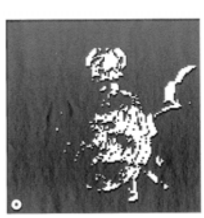


Results

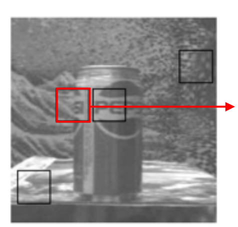


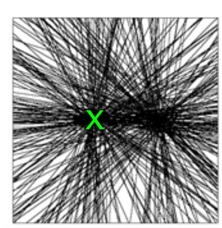
Results





Multiple Motions, again





Find the dominant motion while rejecting outliers.

Black & Anandan; Black & Jepson

Robust estimation models only a single process explicitly

Robust norm:

$$E(\mathbf{a}) = \sum_{x,y \in R} \rho(\nabla I^T \mathbf{u}(\mathbf{x};\mathbf{a}) + I_t; \sigma)$$

Assumption:

Constraints that don't fit the dominant motion are treated as "outliers" (noise).

Problem?

They aren't noise!

Alternative View

* There are two things going on simultaneously.
* We don't know which constraint lines correspond to which motion.
* If we knew this we could estimate the multiple motions.
  - a type of "segmentation" problem
* If we knew the segmentation then estimating the motion would be easy.

EM General framework

Estimate parameters from *segmented data.*

Consider segmentation labels to be *missing data.*

Missing variable problems

A missing data problem is a statistical problem where some data is missing

There are two natural contexts in which missing data are important:

- terms in a data vector are missing for some instances and present for other (perhaps someone responding to a survey was embarrassed by a question)
- an inference problem can be made very much simpler by rewriting it using some variables whose values are unknown.

Missing variable problems

A missing data problem is a statistical problem where some data is missing

There are two natural contexts in which missing data are important:

- terms in a data vector are missing for some instances and present for other (perhaps someone responding to a survey was embarrassed by a question)
- an inference problem can be made very much simpler by rewriting it using some variables whose values are unknown.

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.

Strategy

For each of our examples, if we knew the missing data we could estimate the parameters effectively.

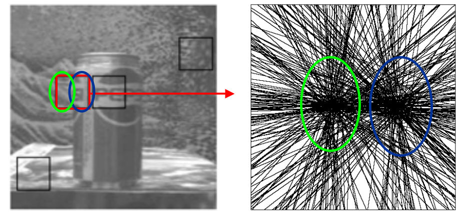
If we knew the parameters, the missing data would follow.

This suggests an iterative algorithm:

1. obtain some estimate of the missing data, using a guess at the parameters;
2. now form a maximum likelihood estimate of the free parameters using the estimate of the missing data.

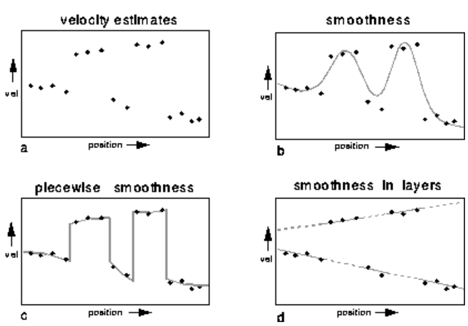
Motion Segmentation

"What goes with what?"

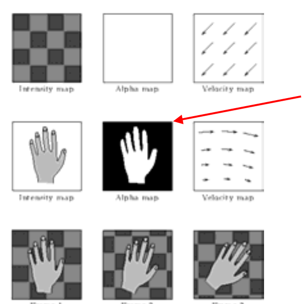


The constraints at these pixels all "go together."

Smoothness in layers



Layered Representation

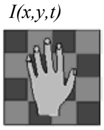


segmentation

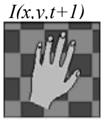
[Adelson]

EM in Pictures

I(x,y,t)



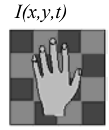
I(x,y,t+1)



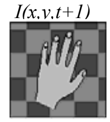
Given images at times t and t+1 containing two motions.

EM in Pictures

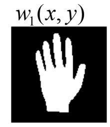
I(x,y,t)



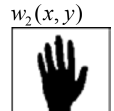
I(x,y,t+1)



$w_1(x,y)$



$w_2(x,y)$

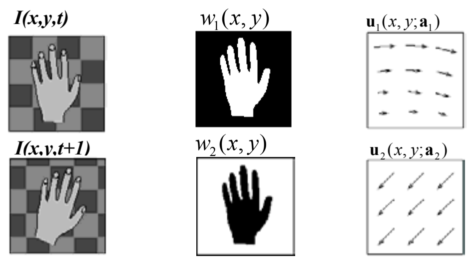


Assume we know the segmentation of pixels into "layers"

$$0 \le w_i(x,y) \le 1$$

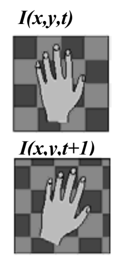
$$\sum_i w_i(x,y) = 1$$

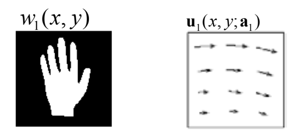
EM in Pictures



Then estimating the motion of each "layer" is easy.

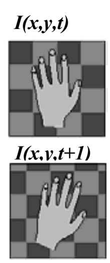
EM in Equations



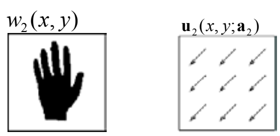


$$E(\mathbf{a}_1) = \sum_{x,y \in R} w_1(\mathbf{x})(\nabla I^T \mathbf{u}(\mathbf{x};\mathbf{a}_1) + I_t)^2$$

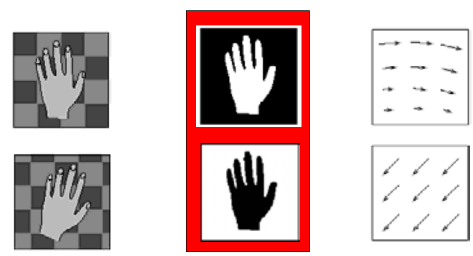
EM in Equations



$$E(\mathbf{a}_2) = \sum_{x,y \in R} w_2(\mathbf{x})(\nabla I^T \mathbf{u}(\mathbf{x};\mathbf{a}_2) + I_t)^2$$

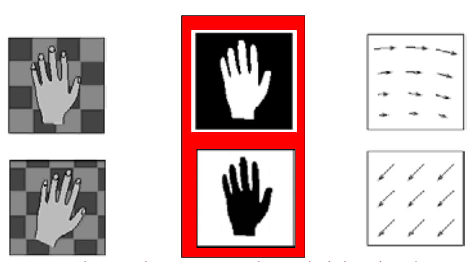


EM in Pictures



Ok. So where do we get the weights?

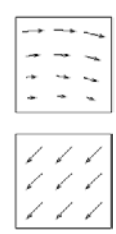
EM in Pictures



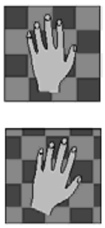
The weights represent the probability that the constraint "belongs" to a particular layer.

EM in Pictures

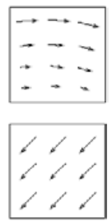
Assume we know the motion of the layers but not the ownership probabilities of the pixels (weights).



EM in Pictures



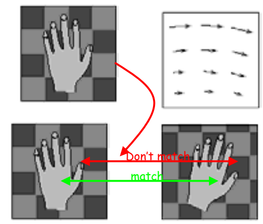
Assume we know the motion of the layers but not the ownership probabilities of the pixels (weights).



Also assume we have a likelihood at each pixel:

$$p(I(t), I(t+1) \mid \mathbf{a}) \approx \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{1}{2}(\nabla I^T \mathbf{u}(\mathbf{a}) + I_t)^2 / \sigma^2)$$

EM in Pictures

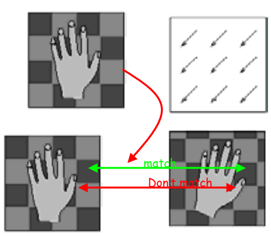


Given the flow, warp the first image towards the second.

Look at the residual error (I_t) (since the flow is now zero).

$$p(W(I(t), \mathbf{a}_1), I(t+1) \mid 0) \approx \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{1}{2}(I_t)^2 / \sigma^2)$$

EM in Pictures

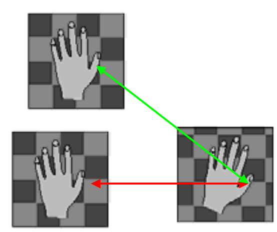


Given the flow, warp the first image towards the second.

Look at the residual error (I_t) (since the flow is now zero).

$$p(W(I(t), \mathbf{a}_2), I(t+1) \mid 0) \approx \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{1}{2}(I_t)^2 / \sigma^2)$$

EM in Pictures



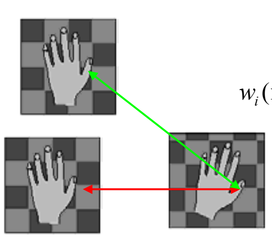
Two "explanations" for each pixel.

Two likelihoods:

$$p(I(\mathbf{x}, t+1) \mid \mathbf{u}(\mathbf{a}_1))$$

$$p(I(\mathbf{x}, t+1) \mid \mathbf{u}(\mathbf{a}_2))$$

EM in Pictures



Compute total likelihood and normalize:

$$w_i(\mathbf{x}) = \frac{p(I(\mathbf{x}, t+1) \mid \mathbf{u}(\mathbf{a}_i))}{\sum_k p(I(\mathbf{x}, t+1) \mid \mathbf{u}(\mathbf{a}_k))}$$

Motion segmentation Example

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

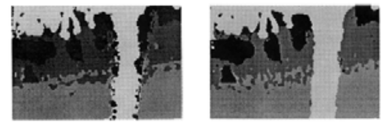
$$\begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

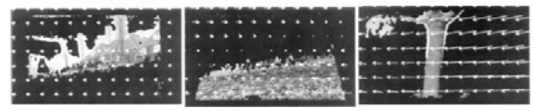
- iterate E/M…
  - determine which pixels belong to which region
  - estimate parameters

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

Lines

- Simple case: 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)

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

- 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

Line fitting review

- In case of single line and normal i.i.d. errors, *maximum likelihood* estimation reduces to least-squares:

$$\min_{a,b} \sum_i (ax_i + b - y_i)^2 = \min_{a,b} \sum_i r_i^2$$

- The line parameters (a,b) are solutions to the system:

$$\begin{pmatrix} \sum_i x_i^2 & \sum_i x_i \\ \sum_i x_i & \sum_i 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_i x_i y_i \\ \sum_i y_i \end{pmatrix}$$

The E Step

- Compute residuals:

$r_1(i) = a_1 x_i + b_1 - y_i$

$r_2(i) =$ k *(uniform noise model)*

- Compute soft assignments:

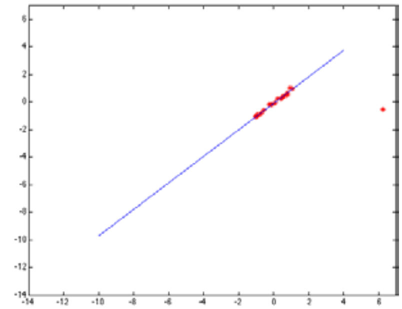
$$w_1(i) = \frac{e^{-r_1^2(i)/\sigma^2}}{e^{-r_1^2(i)/\sigma^2} + e^{-r_2^2(i)/\sigma^2}}$$

$$w_2(i) = \frac{e^{-r_2^2(i)/\sigma^2}}{e^{-r_1^2(i)/\sigma^2} + e^{-r_2^2(i)/\sigma^2}}$$

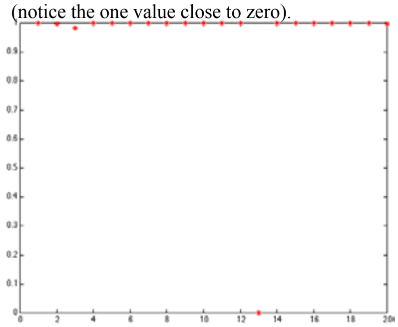
The M Step

Weighted least squares system is solved for (a_1,b_1)

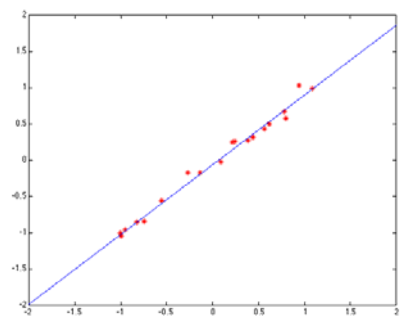
$$\begin{pmatrix} \sum_i w_1(i)x_i^2 & \sum_i w_1(i)x_i \\ \sum_i w_1(i)x_i & \sum_i w_1(i) \end{pmatrix} \begin{pmatrix} a_1 \\ b_1 \end{pmatrix} = \begin{pmatrix} \sum_i w_1(i)x_i y_i \\ \sum_i w_1(i) y_i \end{pmatrix}$$



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



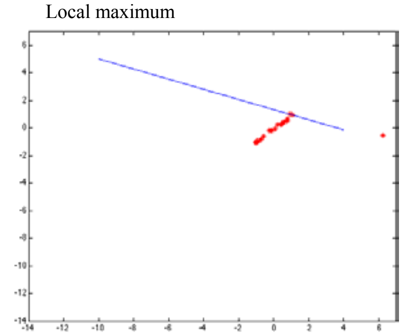
Closeup of the fit



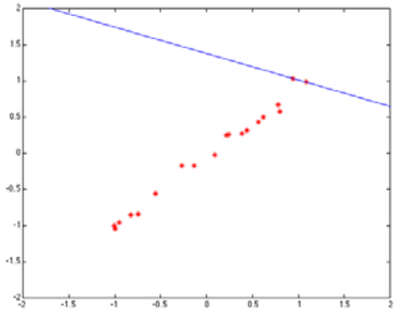
Issues with EM

- 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

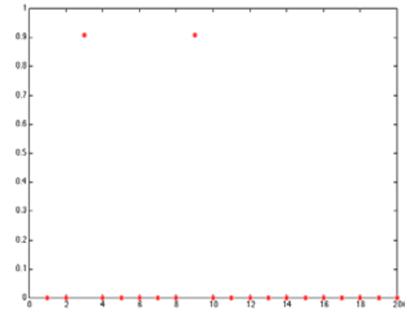
Local maximum



which is an excellent fit to some points



and the deltas for this maximum



Choosing parameters

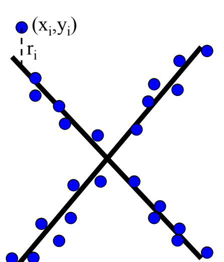
- What about the noise parameter, and the sigma for the line?
  - several methods
    - from first principles knowledge of the problem (seldom really possible)
    - play around with a few examples and choose (usually quite effective, as precise choice doesn't matter much)
  - notice that if k_n is large, 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

Estimating the number of models

- In weighted scenario, additional models will not necessarily reduce the total error.
- The optimal number of models is a function of the σ parameter – how well we expect the model to fit the data.
- Algorithm: start with many models. redundant models will collapse.

Fitting 2 lines to data points

- Input:
  - Data points that where generated by 2 lines with Gaussian noise.
- Output:
  - The parameters of the 2 lines.
  - The assignment of each point to its line.



$y=a_1x+b_1+\sigma v$ $y=a_2x+b_2+\sigma v$

$v \sim N(0,1)$

The E Step

- Compute residuals assuming known lines:

$$r_1(i) = a_1 x_i + b_1 - y_i$$
$$r_2(i) = a_2 x_i + b_2 - y_i$$

- Compute soft assignments:

$$w_1(i) = \frac{e^{-r_1^2(i)/\sigma^2}}{e^{-r_1^2(i)/\sigma^2} + e^{-r_2^2(i)/\sigma^2}}$$

$$w_2(i) = \frac{e^{-r_2^2(i)/\sigma^2}}{e^{-r_1^2(i)/\sigma^2} + e^{-r_2^2(i)/\sigma^2}}$$

The M Step

- In the weighted case we find

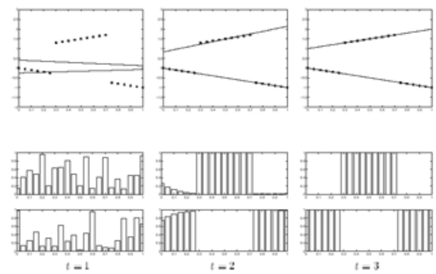
$$\min_{a,b}\left(\sum_i w_1(i) r_1^2(i) + \sum_i w_2(i) r_2^2(i)\right)$$

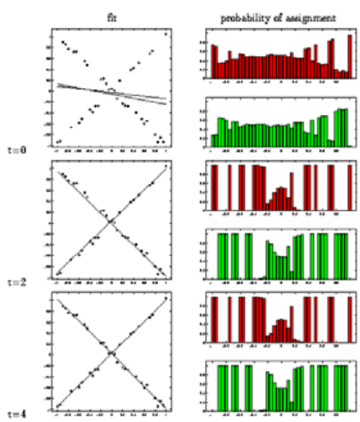
- Weighted least squares system is solved twice for (a_1,b_1) and (a_2,b_2).

$$\begin{pmatrix} \sum_i w_1(i)x_i^2 & \sum_i w_1(i)x_i \\ \sum_i w_1(i)x_i & \sum_i w_1(i) \end{pmatrix}\begin{pmatrix} a_1 \\ b_1 \end{pmatrix} = \begin{pmatrix} \sum_i w_1(i)x_i y_i \\ \sum_i w_1(i) y_i \end{pmatrix}$$

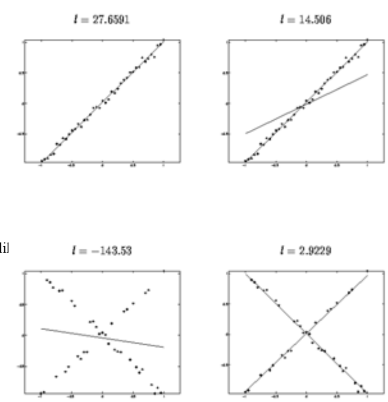
$$\begin{pmatrix} \sum_i w_2(i)x_i^2 & \sum_i w_2(i)x_i \\ \sum_i w_2(i)x_i & \sum_i w_2(i) \end{pmatrix}\begin{pmatrix} a_2 \\ b_2 \end{pmatrix} = \begin{pmatrix} \sum_i w_2(i)x_i y_i \\ \sum_i w_2(i) y_i \end{pmatrix}$$

Illustrations





l=log(likelil



Color segmentation Example

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

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

EM for Mixture models

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

$$\underbrace{p(\boldsymbol{y}) = \sum_l \pi_l p(\boldsymbol{y}|\boldsymbol{a}_l)}_{\text{mixture model}} \qquad \underbrace{\sum_{j \in \text{observations}} \left(\sum_{l=1}^{g} z_{lj} \log p(\boldsymbol{y}_j|\boldsymbol{a}_l)\right)}_{\text{complete data log-likelihood}}$$

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

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

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

Color segmentation

- 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:

$$p(\boldsymbol{x}) = \sum_i p(\boldsymbol{x}|\theta_i)\pi_i$$

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

E

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

E

M

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:

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

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

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

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

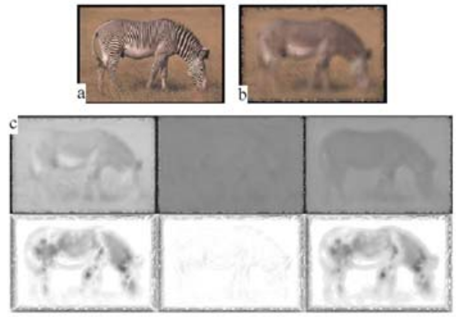


FIGURE 17.1: The image of the zebra in (a) is smoothed at varying scales to yield (b). This smoothing is done using local estimates of scale. These scale measurements essentially measure the scale of the change around a pixel; at edges, the scale is narrow, and in stripey regions it is broad, for example. The features that result are shown in (c); the top three images show the smoothed colour coordinates and the bottom three show the texture features (ac, pc and c — the scale and anisotropy features are weighted by contrast).

Segmentation with EM

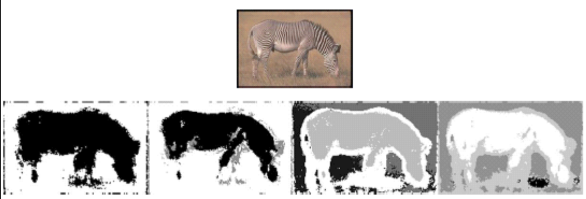
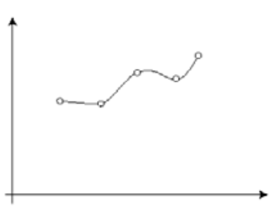


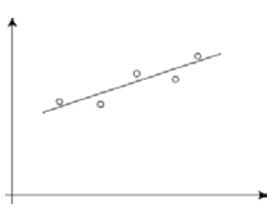
FIGURE 17.2: Each pixel of the zebra image (which is the same as that in figure 17.1) is labelled with the value of m for which $p(m|x_l, \Theta^s)$ is a maximum, to yield a segmentation. The images in show the result of this process for $K = 2, 3, 4, 5$. Each image has K grey-level values corresponding to the segment indexes. *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 © 1998 IEEE*

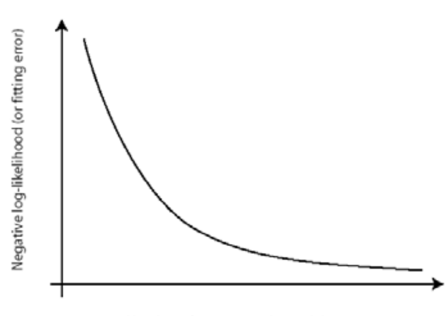
Model Selection

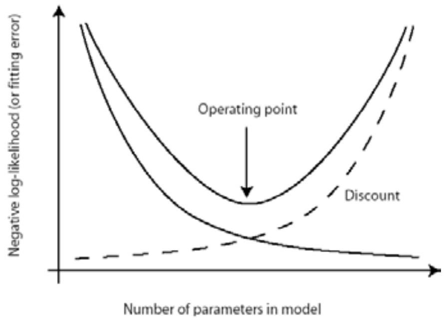
- We wish to choose a model to fit to data
  - e.g. is it a line or a circle?
  - e.g is this a perspective or orthographic camera?
  - e.g. is there an aeroplane there or is it noise?
- Issue
  - In general, models with more parameters will fit a dataset better, but are poorer at prediction
  - This 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)







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

Extreme segmentation

What if more than half the points are noise?

RANSAC

- Iterate:
  - Sample
  - Fit
  - Test
- Keep best estimate; refit on inliers

RANSAC

- Choose a small subset uniformly at random
- Fit to that
- Anything that is close to result is signal; all others are noise
- Refit
- Do this many times and choose the best
- Issues
  - How many times?
    - Often enough that we are likely to have a good line
  - How big a subset?
    - Smallest possible
  - What does close mean?
    - Depends on the problem
  - What is a good line?
    - One where the number of nearby points is so big it is unlikely to be all outliers

Algorithm 15.4: RANSAC: fitting lines using random sample consensus

Determine:
    n — the smallest number of points required
    k — the number of iterations required
    t — the threshold used to identify a point that fits well
    d — the number of nearby points required
        to assert a model fits well
Until k iterations have occurred
    Draw a sample of n points from the data
        uniformly and at random
    Fit to that set of n points
    For each data point outside the sample
        Test the distance from the point to the line
            against t; if the distance from the point to the line
            is less than t, the point is close
    end
    If there are d or more points close to the line
        then there is a good fit. Refit the line using all
        these points.
end
Use the best fit from this collection, using the
    fitting error as a criterion

RANSAC applications

- Fundamental Matricies
  - estimate F from 7 points
  - test agreement with all other points
- Direct motion
  - estimate affine (or rigid motion) from small match
  - see what other parts of image are consistent
- …

Fitting and Probabilistic Segmentation

- Robust estimation
- EM
- Model Selection
- RANSAC

[Slides from Micheal Black and F&P]