Preprocessing Algorithms in fMRI

Lecture Notes

Ahmet Ademoglu, *PhD*
Bogazici University
Institute of Biomedical Engineering

Some concepts and illustrations in this lecture are adapted from the textbooks,

Pattern Recognition and Machine Learning, C. M. Bishop, *Springer*, 2006.

Statistical Parametric Mapping: The Analysis of Functional Brain Images, Editors: K. Friston, J. Ashburner, S. Kiebel, T. Nichols and W. Penny, *Academic Press*, 2006.

RIGID BODY REGISTRATION

Some concepts and illustrations in this lecture are adapted from the textbooks,

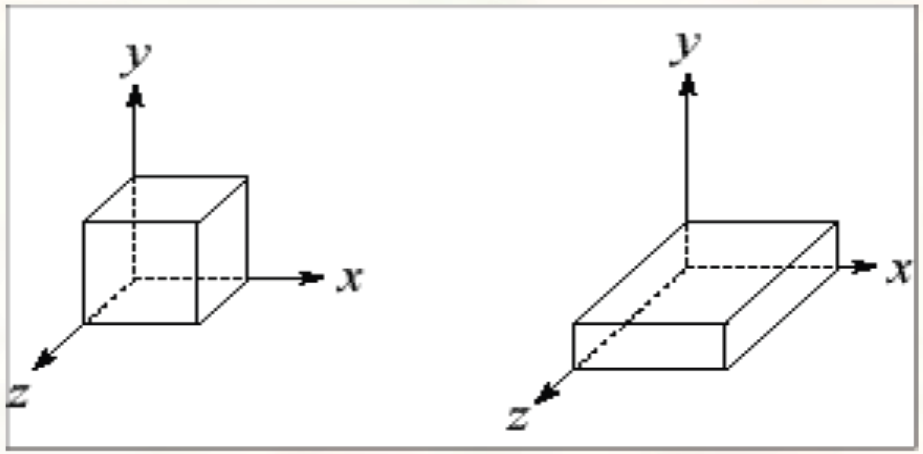
Pattern Recognition and Machine Learning, C. M. Bishop, *Springer*, 2006.

Statistical Paremetric Mapping: The Analysis of Functional Brain Images, Editors: K. Friston, J. Ashburner, S. Kiebel, T. Nichols and W. Penny, *Academic Press*, 2006.

Basic 3-D transformations: scaling

Some of the 3-D transformations are just like the 2-D ones.

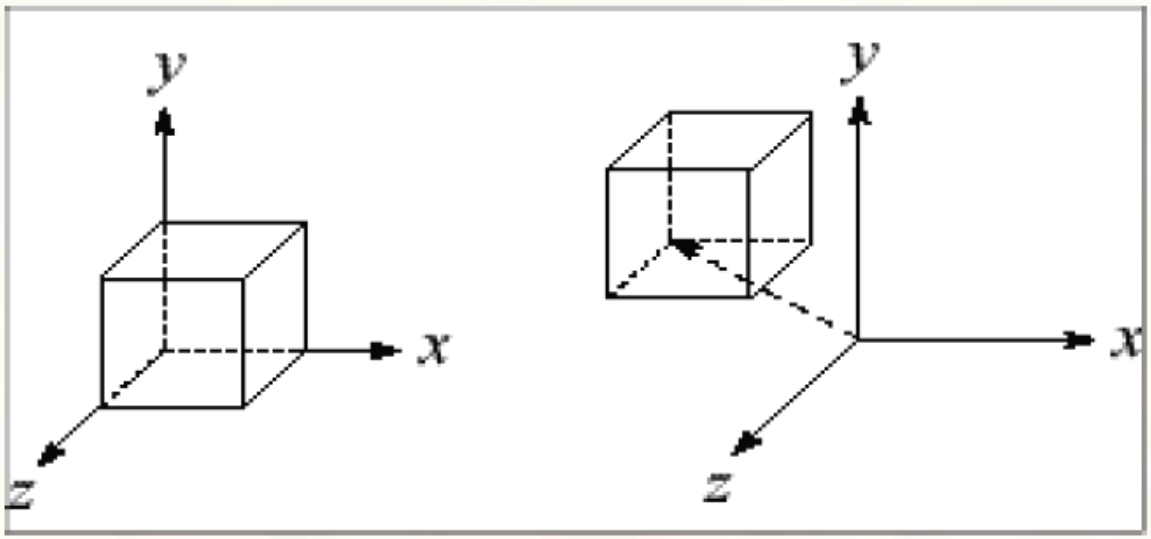
For example, scaling:



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Translation in 3D

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



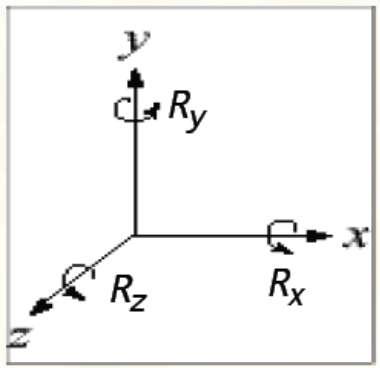
Rotation in 3D

Rotation now has more possibilities in 3D:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

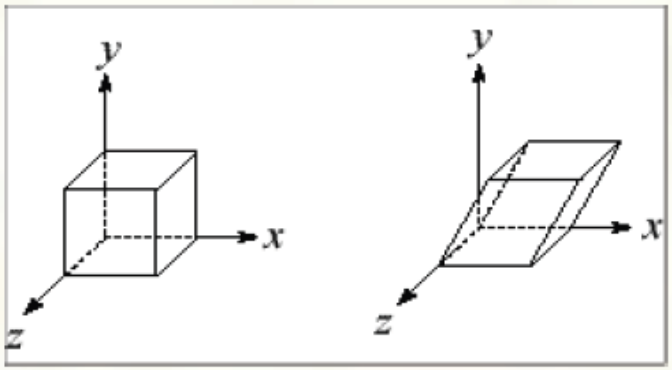


Use right hand rule

Shearing in 3D

- Shearing is also more complicated. Here is one example:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & b & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

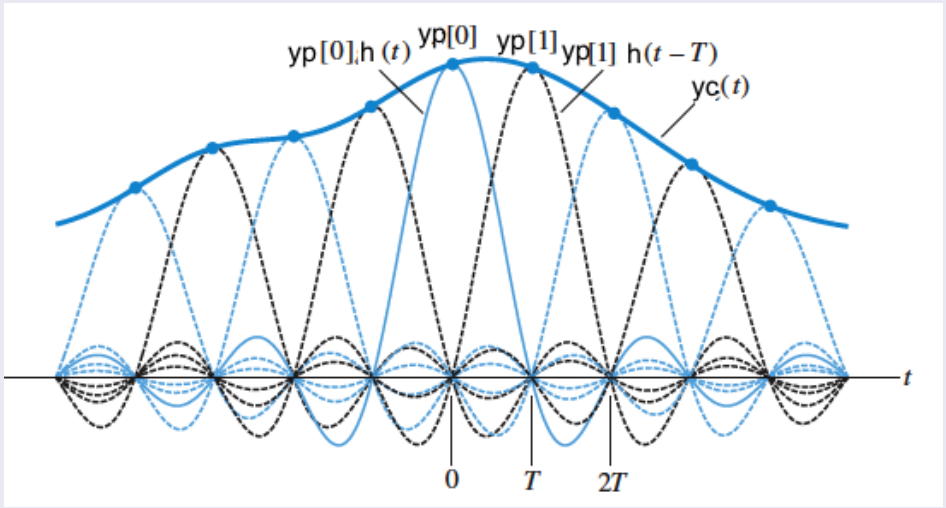


- We call this a shear with respect to the x-z plane.

Rotating a 2-D Image by $\pi/3$ around z axis

```
B = zeros(64,64);
B(15:40,25:40) =1;
imagesc(B); colormap gray;
pause;
F = find(B>0);
[X,Y] = ind2sub(size(B),F);
M = [ 1 0  -32; 0 1 -32 ;0 0 1];
XY1 = M*([ X(:) Y(:) ones(size(X(:)))  ]');
R=[cos(pi/3) sin(pi/3) 0;-sin(pi/3) cos(pi/3) 0;0 0 1];
XY1 = round( inv(M)*(R*XY1) );
F1  =   sub2ind(size(B), XY1(1,:)',XY1(2,:)' );
BR = zeros(64,64);
BR(F1) = B(F);
imagesc(BR); colormap gray;
```

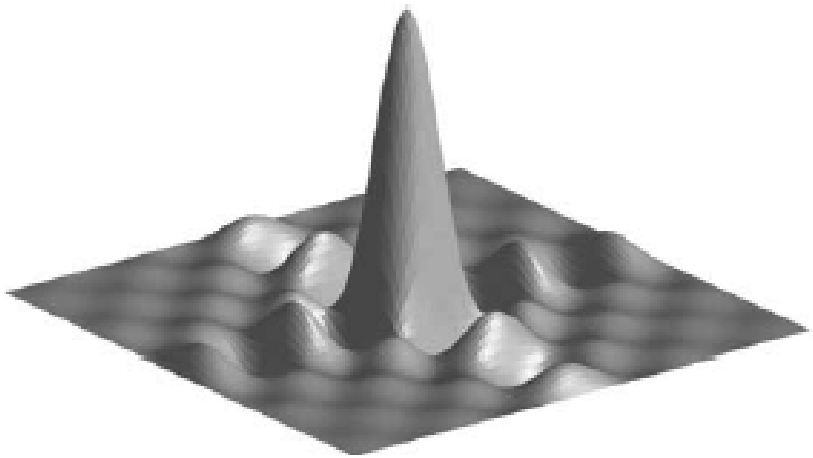
Sinc Interpolation



$$y_c(t) = \sum_{n=-\infty}^{\infty} y_p[nT]h(t - nT) \qquad h(t) = sinc(\frac{t}{T})$$

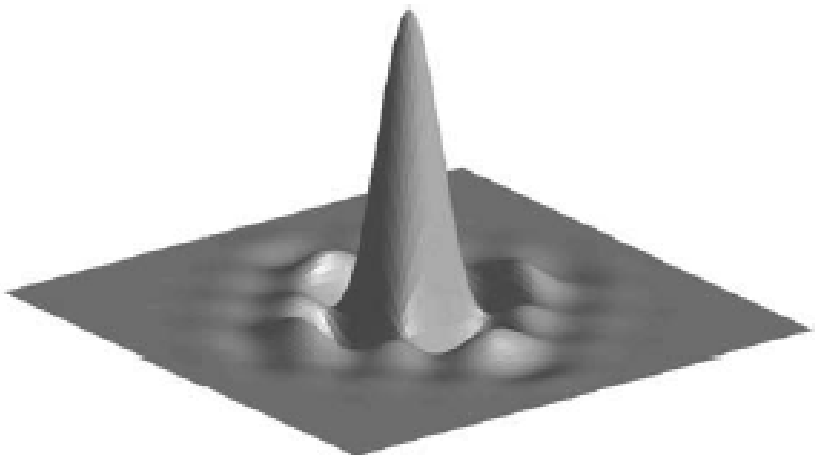
```
y= [1 3 -3 5 8 3 5]';
x= [ 1 3 4 5 6 8 10]';
dx= 0.01; X_min = min(x)-1; X_max = max(x)+1;
X = linspace(X_min,X_max,1/dx)';
Y=sinc(repmat(X,1,length(x))-repmat(x',length(X),1)).* repmat(y',size(X,1),1);
plot(X,Y);grid;hold;stem(x,y);plot(X,sum(Y,2);
```

2-D Interpolation and Resampling



Sinc Interpolation

$$I(x,y) = \sum_{i,j=1}^{N} v_{i,j} sinc(\frac{x-i\Delta x}{T_x}) sinc(\frac{y-i\Delta y}{T_y})$$



B-Spline with degree n Interpolation

$$I(x,y) = \sum_{i,j=1}^{N} v_{i,j} \beta^n(x) \beta^n(y)$$

$$\beta^n(x) = \sum_{i=0}^{n} \frac{(-1)^i (n+1)}{(n+1-i)!i!} \max\left(\frac{n+1}{2} + x - i, 0\right)^n$$

Rotating a 2-D Image by $\pi/3$ around z axis

```
B = zeros(64,64);
B(15:40,25:40) =1;
imagesc(B); colormap gray; pause;
F = find(B > 0);
[X,Y] = ind2sub(size(B),F);
M = [ 1 0  -32; 0 1 -32 ;0 0 1];
XY1 = M*([ X(:) Y(:) ones(size(X(:)))]');
R = [cos(pi/3) sin(pi/3) 0 ; -sin(pi/3) cos(pi/3) 0 ; 0 0 1 ];
XY1 =  (R*XY1);
XY2 = round(inv(M)*XY1);
F1 =  sub2ind(size(B),XY2(1,:), XY2(2,:));
BR = zeros(size(B));
BR(F1) = B(F);
imagesc(BR);colormap gray;
```

Sinc Interpolation on 2D

```
N=64;
B = zeros(N,N);
B(15:40,25:40) =1;
figure; imagesc(B) ; colormap gray;

Delta = 1;
F = find(B>0); % get nonzero voxel indices
[X,Y] = ind2sub(size(B),F); % get nonzero image coordinates
M = [ Delta  0  -32; 0 Delta -32 ;0 0 1]; % digital to cartesian
% transformation Matrix
XY = ([ X(:) Y(:) ones(size(X(:)))  ]');
R=[cos(pi/3) sin(pi/3) 0;-sin(pi/3) cos(pi/3) 0;0 0 1];
XY1 = inv(M)*R*M*XY; % transform digital coords to cartesian space
% coords and rotate them
XY1 =  round(XY1); % rotate cartesian space image coordinates
[F1] = sub2ind(size(B),XY1(1,:),XY1(2,:));
B_R = zeros(size(B));
B_R(F1') = B(F);
figure; imagesc(B_R) ; colormap gray;
```

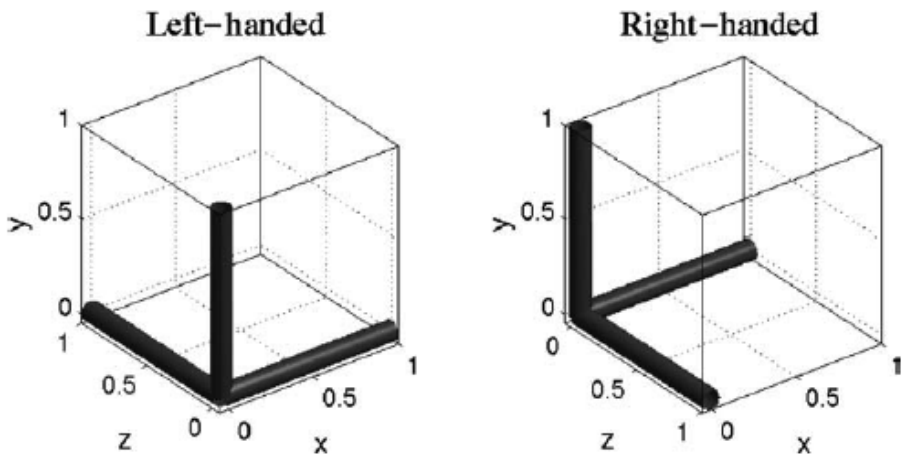
Sinc Interpolation on 2D

```
T =  10 ; % upsampling factor
Delta = (1/T); % pixel resolution in cartesian space
NT = N*T;
B_U= upsample((reshape(upsample(B_R(:),T),NT,N))',T)' ;
B_U = reshape(B_U,NT,NT); % upsampled image by a factor of T
figure;  imagesc(B_U) ; colormap gray;

SincX = sinc([-5:Delta:5]'/T);
SincX=hanning(length(SincX)).*SincX;
SincX = SincX*SincX'; % 2D sinc filter
SincX = SincX/sum(SincX(:))*(T*T); % normalize filter gain
SincF = fft2(SincX,NT,NT); % Fourier Transform of 2D sinc Filter
B_U = ifft2( fft2(B_U).*(SincF) ); % 2D Filtering
%B_U = B_U.*(B_U>0.1);  % Threshold
figure; imagesc(B_U) ; colormap gray;
B_I= downsample(downsample(B_U,T)',T)' ;
figure; imagesc(B_I) ; colormap gray;
```

Anisotropic voxel size $\mathbf{M_f} = \begin{bmatrix} 2.1 & 0 & 0 & -135.45 \\ 0 & 2.1 & 0 & -135.45 \\ 0 & 0 & 2.45 & -53.9 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

The transformation $\mathbf{M_f}$ maps the voxel coordinates in image $\mathbf{f}$ to Cartesian space coordinates in millimetres.



Talairach-Tournoux is right-handed *i.e.* the first dimension x increases from left to right, the second, y from posterior (back) to anterior (front), and the third, z from inferior (bottom) to superior (top).

Optimization

Let $b_i(\theta)$ be the error function describing the between the source and reference images at voxel i,

$$b_i(\theta - \Delta\theta) \approx b_i(\theta) - \frac{\partial b_i(\theta)}{\partial \theta_1}\Delta\theta_1 - \frac{\partial b_i(\theta)}{\partial \theta_2}\Delta\theta_2 + \dots$$

Mimimizing $\sum_i b_i(\theta - \Delta\theta)^2 \longrightarrow \mathbf{A}\Delta\theta = \mathbf{b}$

$$\begin{bmatrix} \frac{\partial b_1(\theta)}{\partial \theta_1} & \frac{\partial b_1(\theta)}{\partial \theta_2} & \dots \\ \frac{\partial b_2(\theta)}{\partial \theta_1} & \frac{\partial b_2(\theta)}{\partial \theta_2} & \dots \\ \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \vdots \\ \Delta\theta_7 \end{bmatrix} \approx \begin{bmatrix} b_1(\theta) \\ b_2(\theta) \\ \vdots \end{bmatrix}$$

Iterative Scheme: $\theta^{n+1} = \theta^n - (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$

If $\mathbf{M}_r$ is a transformation in cartesian space from image f to image g then $\mathbf{M}_f^{-1}\mathbf{M}_r^{-1}\mathbf{M}_g$ maps g to f.
$\mathbf{x}_i$ points in reference image g are compared with those in source image f by minimizing

$$\sum_i \left(f(\mathbf{M}\mathbf{x}_i) - \theta_7\ g(\mathbf{x}_i)\right)^2$$

where $\mathbf{M}$ is a rigid body transformation with 6 parameters $\{\theta_1, \theta_2 \ldots, \theta_6\}$ which maps voxels in $\mathbf{g}$ to those in $\mathbf{f}$ *i.e.* $\mathbf{M} = \mathbf{M}_f^{-1}\mathbf{M}_r^{-1}\mathbf{M}_g$ and θ_7 is the scaling factor that adjusts the source image to the reference.

Iterative Scheme: $\theta^{n+1} = \theta^n - (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$

$$\underbrace{\begin{bmatrix} \frac{\partial f(\mathbf{M}\mathbf{x}_1)}{\partial \theta_1} & \frac{\partial f(\mathbf{M}\mathbf{x}_1)}{\partial \theta_2} & \cdots & -g(\mathbf{x}_1) \\ \frac{\partial f(\mathbf{M}\mathbf{x}_2)}{\partial \theta_1} & \frac{\partial b_2(\mathbf{M}\mathbf{x}_2)}{\partial \theta_2} & \cdots & -g(\mathbf{x}_2) \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \vdots \end{bmatrix} \approx \underbrace{\begin{bmatrix} f(\mathbf{M}\mathbf{x}_1) - \theta_7 g(\mathbf{x}_1) \\ f(\mathbf{M}\mathbf{x}_2) - \theta_7 g(\mathbf{x}_2) \\ \vdots \end{bmatrix}}_{\mathbf{b}}$$

A More Robust Optimization Function

$$\lambda_1 \sum_i \left(f(\mathbf{M}\mathbf{x}_i) - \theta_7 g(\mathbf{x}_i)\right)^2 + \lambda_2 \sum_j \left(g(\mathbf{M}^{-1}\mathbf{y}_j) - \theta_7^{-1} f(\mathbf{y}_j)\right)^2$$

$$\mathbf{A} = \begin{bmatrix} \lambda_1^{\frac{1}{2}} \frac{\partial f(\mathbf{M}\mathbf{x}_1)}{\partial \theta_1} & \lambda_1^{\frac{1}{2}} \frac{\partial f(\mathbf{M}\mathbf{x}_1)}{\partial \theta_2} & \cdots & -\lambda_1^{\frac{1}{2}} g(\mathbf{x}_1) \\ \lambda_1^{\frac{1}{2}} \frac{\partial f(\mathbf{M}\mathbf{x}_2)}{\partial \theta_1} & \lambda_1^{\frac{1}{2}} \frac{\partial f_2(\mathbf{M}\mathbf{x}_2)}{\partial \theta_2} & \cdots & -\lambda_1^{\frac{1}{2}} g(\mathbf{x}_2) \\ \vdots & \vdots & \vdots & \vdots \\ \lambda_2^{\frac{1}{2}} \frac{\partial g(\mathbf{M}^{-1}\mathbf{y}_1)}{\partial \theta_1} & \lambda_2^{\frac{1}{2}} \frac{\partial g(\mathbf{M}_1\mathbf{y}_1)}{\partial \theta_2} & \cdots & \lambda_2^{\frac{1}{2}} \theta_7^{-2} f(\mathbf{y}_1) \\ \lambda_2^{\frac{1}{2}} \frac{\partial g(\mathbf{M}^{-1}\mathbf{y}_2)}{\partial \theta_1} & \lambda_2^{\frac{1}{2}} \frac{\partial g(\mathbf{M}_1\mathbf{y}_2)}{\partial \theta_2} & \cdots & \lambda_2^{\frac{1}{2}} \theta_7^{-2} f(\mathbf{y}_2) \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} \lambda_1^{\frac{1}{2}} \left(f(\mathbf{M}\mathbf{x}_1) - \theta_7 g(\mathbf{x}_1)\right) \\ \lambda_1^{\frac{1}{2}} \left(f(\mathbf{M}\mathbf{x}_2) - \theta_7 g(\mathbf{x}_2)\right) \\ \vdots \\ \lambda_2^{\frac{1}{2}} \left(g(\mathbf{M}^{-1}\mathbf{y}_1) - \theta_7^{-1} f(\mathbf{y}_1)\right) \\ \lambda_2^{\frac{1}{2}} \left(g(\mathbf{M}^{-1}\mathbf{y}_2) - \theta_7^{-1} f(\mathbf{y}_2)\right) \\ \vdots \end{bmatrix}$$

NONLINEAR REGISTRATION

Objective Functions

The aim of the nonlinear registration is to find the most probable deformation θ given the data $\mathbf{Y}$.

$$P(\theta|\mathbf{Y}) = \frac{P(\mathbf{Y}|\theta)P(\theta)}{P(\mathbf{Y})}$$

Likelihood Term

If f_i is the intensity of the i^{th} voxel of the source image and θ is the vector of parameters then the i^{th} voxel of the deformed template image $g_i(\theta)$ is

$$f_i = g_i(\theta) + \epsilon_i \text{ where } \epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

$$P(f_i|\theta) = (2\pi\sigma^2)^{-\frac{1}{2}} \exp(-\frac{(f_i - g_i(\theta))^2}{2\sigma^2})$$

The probability of the whole image $\mathbf{f}$ is

$$P(\mathbf{f}|\theta) = \prod_{i=1}^{M} (2\pi\sigma^2)^{-\frac{1}{2}} \exp(-\frac{(f_i - g_i(\theta))^2}{2\sigma^2})$$

The negative log-likelihood

$$-\log P(\mathbf{f}|\theta) = \frac{M}{2}\log(2\pi\sigma^2) + \sum_{i=1}^{I}\frac{(f_i - g_i(\theta))^2}{2\sigma^2}$$

is maximized over M voxels and with constant σ^2 by minimizing

$$\mathcal{E}(\theta) = \sum_{i=1}^{M}\frac{(f_i - g_i(\theta))^2}{2\sigma^2}$$

Prior Term for Regularization

$$P(\theta) = \frac{1}{(2\pi)^{\frac{N}{2}}}|\mathbf{C}_\theta|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}(\theta-\theta_0)^T\mathbf{C}_\theta^{-1}(\theta-\theta_0)\right)$$

The negative log-likelihood

$$-\log P(\theta) = \frac{N}{2}\log(2\pi) + \frac{1}{2}\log|\mathbf{C}_\theta| + \frac{1}{2}\left((\theta-\theta_0)^T\mathbf{C}_\theta^{-1}(\theta-\theta_0)\right)^2$$

Linear Regularization

If $\theta_0 = 0$ then $\frac{1}{2}\theta^T \mathbf{C}_\theta^{-1}\theta$ is called energy density and it is defined as the Membrane Energy of a vector function $\mathbf{u}(\mathbf{x}, \theta)$. If u is a linear function of θ as $\mathbf{u} = \mathbf{L}^{-1}\theta$ then
$\frac{1}{2}\theta^T|\mathbf{C}_\theta^{-1}|\theta = \mathbf{u}^T\mathbf{L}^T\mathbf{L}\mathbf{u}$ where $\mathbf{L}$ is a differential operator.

Membrane Energy (Laplacian) Model

$$\mathcal{H}(\theta) = \frac{\beta}{2}\int_{x\in\Omega}\sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3}\left(\frac{\partial u_i(\mathbf{x},\theta)}{\partial x_j}\right)^2 d\mathbf{x},$$

The overall optimization function is

$$\mathcal{L} = \mathcal{E}(\theta) + \mathcal{H}(\theta)$$

$$\mathcal{L} = \frac{1}{2\sigma^2}\sum_{i=1}^{M}(f_i - g_i(\theta))^2 + \frac{\beta}{2}\int_{x\in\Omega}\sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3}\left(\frac{\partial u_i(\mathbf{x},\theta)}{\partial x_j}\right)^2 d\mathbf{x},$$

which can be optimized for hyperparameters $1/\sigma^2$ and β using *ReML* algorithm.

Small Deformation Models

Let $\mathbf{x}_i$ be the coordinate of the i^{th} voxel in image f, a warping is defined as $\mathbf{x}_i \longrightarrow \mathbf{y}(\mathbf{x}_i, \theta)$ where
$\mathbf{y}(\mathbf{x}_i, \theta)$ is the coordinate of the i^{th} voxel after nonlinear mapping, then the i^{th} voxel value $f(\mathbf{x_i})$ or f_i is mapped to an image h such that

$$f(\mathbf{x}_i) \longrightarrow h(\theta) = h(\mathbf{y}(\mathbf{x}_i, \theta))$$

Sometimes deformation $\mathbf{y}(\mathbf{x}_i, \theta)$ can be parametrized linearly by using spatial basis functions

$$y_i(\mathbf{x}_i, \theta) = \sum_{m=1}^{M} \theta_m \phi_{mi}(\mathbf{x}_i) = \Phi_i(\mathbf{x}_i)\theta$$

where $\Phi_i(\mathbf{x}_i)$ is the matrix whose columns are the m spatial basis vectors in direction i.
A displacement can also be added so that

$$y_i(\mathbf{x}_i, \theta) = \mathbf{x}_i + \Phi_i(\mathbf{x}_i)\theta$$

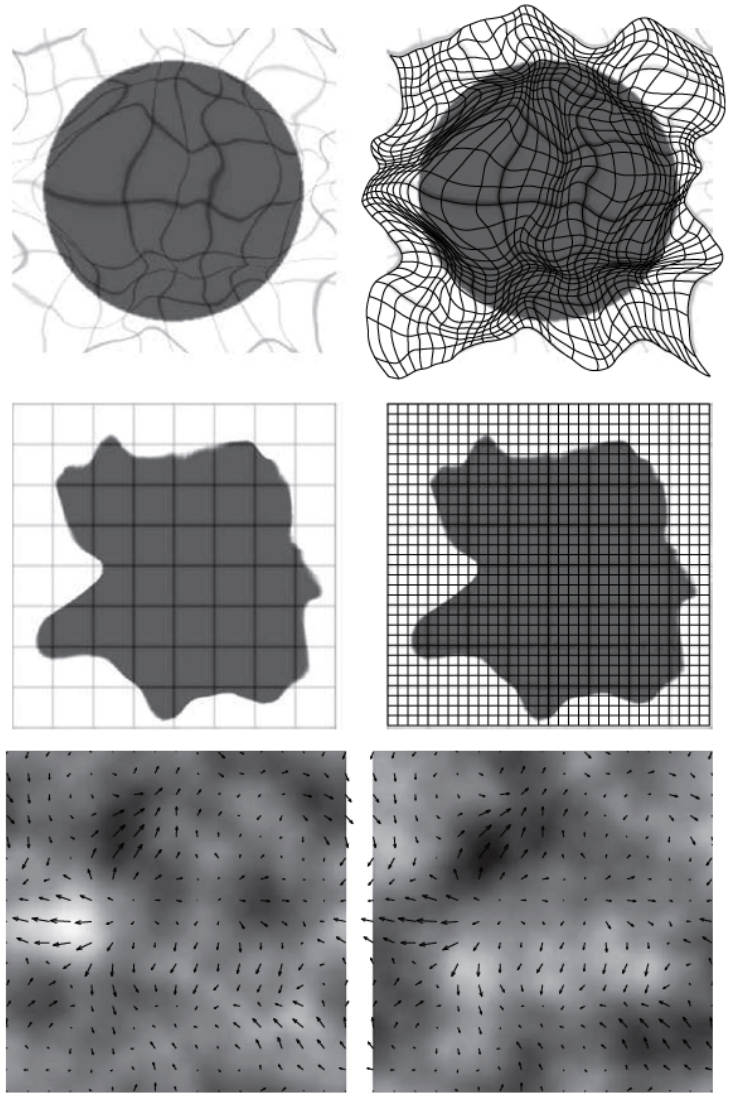
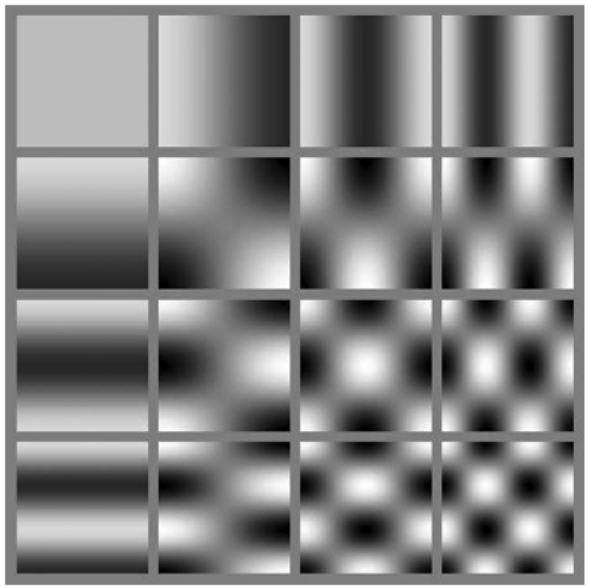


Image with deformation field overlayed

Warped Image

2D scalar fields
in horizontal (left) and
vertical (right) displacements

Discrete Cosine Basis as Deformation Field



$$y_1(\mathbf{x},\theta) = x_1 + u_1 = x_1 + \sum_{m=1}^{M} \theta_{m1}\phi_m(\mathbf{x})$$
$$y_2(\mathbf{x},\theta) = x_2 + u_2 = x_2 + \sum_{m=1}^{M} \theta_{m2}\phi_m(\mathbf{x})$$
$$y_3(\mathbf{x},\theta) = x_3 + u_3 = x_3 + \sum_{m=1}^{M} \theta_{m3}\phi_m(\mathbf{x})$$

θ_{mk}: m^{th} coefficient for dimension k
$\phi_m(\mathbf{x})$: m^{th} basis function at position $\mathbf{x}$
m is usually around 1000

Here an identity transformation is added to the displacement field

$$\mathbf{y}(\mathbf{x}_i,\theta) = \mathbf{x}_i + \Phi(\mathbf{x}_i)\theta$$

3-D Discrete Cosine Transform Basis

$$\phi_1(i) = \frac{1}{\sqrt{I}}, \quad i = 1 \dots I \qquad \phi_m(i) = \sqrt{\frac{2}{I}}\cos\left(\frac{\pi(2i-1)(m-1)}{2I}\right), \quad i = 1 \dots I,\ m = 2 \dots M.$$

$$\phi_m(\mathbf{x}) = \phi_{m1}(x_1)\phi_{m2}(x_2)\phi_{m3}(x_3)$$

Estimation of The Mappings

Nonlinear Registration searches for a maximum *a posteriori* MAP estimate of the parameters defining the warps.

The overall optimization function is $\mathcal{L} = \mathcal{E}(\theta) + \mathcal{H}(\theta)$

Using *Levenberg-Marquardt* optimization

$$\theta^{(n+1)} = \theta^{(n)} - \left(\frac{\partial^2 \mathcal{L}(\theta)}{\partial \theta^2}\bigg|_{\theta^{(n)}} + \zeta \mathbf{I} \right)^{-1} \frac{\partial \mathcal{L}(\theta)}{\partial \theta}\bigg|_{\theta^{(n)}}$$

When the prior is multivariate normal,

$$-\tfrac{1}{2}(\theta - \theta_0)^T \mathbf{C}_\theta^{-1}(\theta - \theta_0)$$

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta} = \frac{\partial \mathcal{E}(\theta)}{\partial \theta} + \mathbf{C}_\theta^{-1}(\theta - \theta_0) \text{ and}$$
$$\frac{\partial^2 \mathcal{L}(\theta)}{\partial \theta^2} = \frac{\partial^2 \mathcal{E}(\theta)}{\partial \theta^2} + \mathbf{C}_\theta^{-1}$$

$\mathcal{E}(\theta) = \frac{1}{2\sigma^2} \sum_{i=1}^{I} (wf(\mathbf{x_i}) - g(\mathbf{y}(\mathbf{x}_i, \theta)))^2$ w being a scaling parameter.

the first derivatives are

$$\frac{\partial \mathcal{E}}{\partial \theta_m} = -\frac{1}{\sigma^2} \sum_{i=1}^{I} \frac{\partial g(\mathbf{y}(\mathbf{x}_i, \theta))}{\partial \theta_m} (wf(\mathbf{x_i}) - g(\mathbf{y}(\mathbf{x}_i, \theta)))$$

$$\frac{\partial \mathcal{E}}{\partial w} = \frac{1}{\sigma^2} \sum_{i=1}^{I} f(\mathbf{x}_i)(wf(\mathbf{x}_i) - g(\mathbf{y}(\mathbf{x}_i, \theta)))$$

The second derivatives are

$$\frac{\partial \mathcal{E}}{\partial \theta_m \partial \theta_n} =$$

$$\frac{1}{\sigma^2} \sum_{i=1}^{I} \left(\frac{\partial gg(\mathbf{y}(\mathbf{x}_i, \theta))}{\partial \theta_m} \frac{\partial g(\mathbf{y}(\mathbf{x}_i, \theta))}{\partial \theta_n} - \underbrace{\frac{\partial^2 g(\mathbf{y}(\mathbf{x}_i, \theta))}{\partial \theta_m \partial \theta_n} (wf(\mathbf{x_i}) - g(\mathbf{y}(\mathbf{x}_i, \theta)))}_{\text{usually ignored}} \right)$$

$$\frac{\partial^2 \mathcal{E}}{\partial w^2} = \frac{1}{\sigma^2} \sum_{i=1}^{I} f(\mathbf{x}_i)^2$$

$$\frac{\partial^2 \mathcal{E}}{\partial \theta_m \partial w} = -\frac{1}{\sigma^2} \sum_{i=1}^{I} \frac{\partial g(\mathbf{y}(\mathbf{x}_i, \theta))}{\partial \theta_m} f(\mathbf{x}_i)$$

Image Warping using Bayesian Approach

- Image warping can be formulated as a regularised maximum likelihood problem and involves finding a set of parameters that maximise the likelihood of the observed data.
- We need a set of parameters θ that are maximally likely given the data which can be expressed something like:
  $-\log P(\theta|\mathbf{Y}) = -\log P(\mathbf{Y}|\theta) - \log P(\theta) + const$

Generative Model

Assuming that f_i and g_i are the source and the reference (template) image values at their i^{th} voxels, respectively and

$$g_i = g(\theta) = g(\mathbf{y}(\mathbf{x}_i, \theta)) = g(\mathbf{x}_i + \Phi(\mathbf{x}_i)\theta)$$

is the interpolated value of the deformed template image at point $\mathbf{y}_i$ where $\Phi(\mathbf{x}_i)$ is the matrix whose columns are the m spatial basis vectors and then the observation at iteration (n) in voxel i with coordinate $\mathbf{x}_i$ is $\mathbf{Y}_i^{(n)} = f_i - g_i$

and the optimization function $\mathcal{L}$ is

$$\mathcal{L} = \mathcal{E} + \mathcal{H} = \lambda_1 \frac{1}{2} \mathbf{Y}^{(n)T} \mathbf{Y}^{(n)} + \lambda_2 \frac{1}{2} \theta^{(n)T} \mathbf{C}_\theta^{-1} \theta^{(n)}$$

where $\lambda = [1/\sigma^2 \ \beta]^T$ are the hyperparameters.

- In image registration, if we use too many parameters in a our model, then we will have the overfitting problem, whereas if we do not use enough parameters then we will not obtain a proper registeration.
- This problem can be addressed by obtaining an optimal balance between the cost function terms $\mathcal{E}$ and H by using the most optimal weight for each.
- These weights as called hyperparameters $\lambda_1 = 1/\sigma^2$ and and $\lambda_2 = \beta$ can be estimated using an *Empirical Bayes* approach, which employs an *Expectation Maximization* (EM) scheme.

Our aim is to find the hyperparameters λ which maximize the likelihood of observed data $\mathbf{Y}$ given the unobserved variables θ in the model:

$$p(\mathbf{Y}|\lambda) = \int_\theta p(\mathbf{Y}, \theta|\lambda) d\theta$$

We use a distribution $q(\theta)$ to approximate $p(\theta|\mathbf{Y}, \lambda)$ and maximize a lower bound on $p(\mathbf{Y}|\lambda)$ by making the Kullbeck-Leibler divergence between $q(\theta)$ and $p(\theta|\mathbf{Y}, \lambda)$ to be zero.

The objective is to maximize the free energy $\mathcal{L}(q, \lambda)$

$$\mathcal{L}(q, \lambda) = \log p(\mathbf{Y}|\lambda) - \int_\theta q(\theta) \log \left(\frac{q(\theta)}{p(\theta|\mathbf{Y}, \lambda)} \right) d\theta$$

$$= \int_\theta q(\theta) \log p(\mathbf{Y}, \theta|\lambda) d\theta - \int_\theta q(\theta) \log q(\theta) d\theta$$

EM algorithm maximizes $\mathcal{L}(q, \lambda)$ by

- E-step: maximizing $\mathcal{L}(q, \lambda)$ w.r.t. $q(\theta)$ keeping λ constant.
- M-step: maximizing $\mathcal{L}(q, \lambda)$ w.r.t. θ keeping $q(\theta)$ constant.

EM Algorithm for a Linear Generative Model

$\mathbf{y} = \mathbf{X}\theta + \epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_\lambda)$

If the prior distribution for θ is multivariate normal *i.e.*

$$\theta \sim \mathcal{N}(\eta_\theta, C_\theta)$$

$$\text{then } p(\mathbf{y}, \theta|\lambda) = p(\mathbf{y}|\theta, \lambda)p(\theta)$$

where

$$p(\mathbf{y}|\theta, \lambda) = (2\pi)^{-\frac{I}{2}}|\mathbf{C}_\lambda|^{-\frac{1}{2}} \exp(-\tfrac{1}{2}(\mathbf{y} - \mathbf{X}\theta)^T\mathbf{C}_\lambda^{-1}(\mathbf{y} - \mathbf{X}\theta))$$

and

$$p(\theta) = (2\pi)^{-\frac{M}{2}}|\mathbf{C}_\theta|^{-\frac{1}{2}} \exp(-\tfrac{1}{2}(\theta - \eta_\theta)^T\mathbf{C}_\theta^{-1}(\theta - \eta_\theta))$$

The posterior distribution of θ

$p(\theta|\mathbf{y}, \lambda) \propto p(\mathbf{y}|\theta, \lambda)p(\theta)$

becomes

$p(\theta|\mathbf{y}, \lambda) \sim \mathcal{N}(\eta_{\theta|\mathbf{y}}, \mathbf{C}_{\theta|\mathbf{y}})$

where

$$\mathbf{C}_{\theta|\mathbf{y}} = \left(\mathbf{X}^T\mathbf{C}_\lambda^{-1}\mathbf{X} + \mathbf{C}_\theta^{-1}\right)^{-1}$$

$$\eta_{\theta|\mathbf{y}} = \mathbf{C}_{\theta|\mathbf{y}}\left(\mathbf{X}^T\mathbf{C}_\lambda^{-1}\mathbf{y} + \mathbf{C}_\theta^{-1}\eta_\theta\right)$$

The cost function is

$\mathcal{L}(q,\lambda) = \int_\theta q(\theta) \log p(\mathbf{y},\theta|\lambda) d\theta - \int_\theta q(\theta) \log q(\theta) d\theta$

Setting the approximation distribution $q(\theta)$ as

$q(\theta) = p(\theta|\mathbf{y},\lambda) \sim \mathcal{N}(\eta_{\theta|\mathbf{y}}, \mathbf{C}_{\theta|\mathbf{y}})$

yields

$$\int_\theta q(\theta) \log q(\theta) d\theta = -\frac{M}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{C}_{\theta|\mathbf{y}}|$$

$$-\frac{1}{2}\underbrace{\int_\theta \left((\theta-\eta_{\theta|\mathbf{y}})^T \mathbf{C}_{\theta|\mathbf{y}}^{-1}(\theta-\eta_{\theta|\mathbf{y}})\right)\mathcal{N}(\eta_{\theta|\mathbf{y}}, \mathbf{C}_{\theta|\mathbf{y}})d\theta}_{M}$$

$$\int_{\theta} q(\theta) \log p(\mathbf{y}, \theta|\lambda) d\theta = \int_{\theta} q(\theta) \log \left[p(\mathbf{y}|\theta, \lambda) \cdot p(\theta) \right] d\theta$$

$$\int_{\theta} \mathcal{N}(\eta_{\theta|\mathbf{y}}, \mathbf{C}_{\theta|\mathbf{y}}) \left[\log \mathcal{N}(\mathbf{X}\theta, \mathbf{C}_{\lambda}) + \log \mathcal{N}(\eta_{\theta}, \mathbf{C}_{\theta}) \right] d\theta$$

$$= -\tfrac{1}{2} \int_{\theta} [(\mathbf{y} - \mathbf{X}\theta)^T \mathbf{C}_{\lambda}^{-1} (\mathbf{y} - \mathbf{X}\theta) + (\theta - \eta_{\theta})^T \mathbf{C}_{\theta}^{-1} (\theta - \eta_{\theta})] \mathcal{N}(\eta_{\theta|\mathbf{y}}, \mathbf{C}_{\theta|\mathbf{y}}) d\theta$$

$$-\tfrac{I}{2} \log(2\pi) - \tfrac{1}{2} \log |\mathbf{C}_{\lambda}| - \tfrac{M}{2} \log(2\pi) - \tfrac{1}{2} \log |\mathbf{C}_{\theta}|$$

If we use the transformations

$$(\mathbf{y}-\mathbf{X}\theta)^T\mathbf{C}_\lambda^{-1}(\mathbf{y}-\mathbf{X}\theta) = (\theta-\eta_{\theta|\mathbf{y}})^T\mathbf{X}^T\mathbf{C}_\lambda^{-1}\mathbf{X}(\theta-\eta_{\theta|\mathbf{y}}) + (\mathbf{y}-\mathbf{X}\eta_{\theta|\mathbf{y}})^T\mathbf{C}_\lambda^{-1}(\mathbf{y}-\mathbf{X}\eta_{\theta|\mathbf{y}})$$

$$(\theta-\eta_\theta)^T\mathbf{C}_\theta^{-1}(\theta-\eta_\theta) = (\theta-\eta_{\theta|\mathbf{y}})^T\mathbf{C}_\theta^{-1}(\theta-\eta_{\theta|\mathbf{y}}) + (\eta_\theta-\eta_{\theta|\mathbf{y}})^T\mathbf{C}_\theta^{-1}(\eta_\theta-\eta_{\theta|\mathbf{y}})$$

the exponential terms in $\int_\theta q(\theta)\log p(\mathbf{y},\theta|\lambda)d\theta$ become

$$-\tfrac{1}{2}\int_\theta \left[(\theta-\eta_{\theta|\mathbf{y}})^T\mathbf{C}_{\theta|\mathbf{y}}^{-1}(\theta-\eta_{\theta|\mathbf{y}})\right]\mathcal{N}(\eta_{\theta|\mathbf{y}},\mathbf{C}_{\theta|\mathbf{y}})d\theta + (\mathbf{y}-\mathbf{X}\eta_{\theta|\mathbf{y}})^T\mathbf{C}_\lambda^{-1}(\mathbf{y}-\mathbf{X}\eta_{\theta|\mathbf{y}})$$
$$-\tfrac{1}{2}(\eta_\theta-\eta_{\theta|\mathbf{y}})^T\mathbf{C}_\theta^{-1}(\eta_\theta-\eta_{\theta|\mathbf{y}})$$

$$= -\tfrac{M}{2} - \tfrac{1}{2}(\mathbf{y}-\mathbf{X}\eta_{\theta|\mathbf{y}})^T\mathbf{C}_\lambda^{-1}(\mathbf{y}-\mathbf{X}\eta_{\theta|\mathbf{y}}) - \tfrac{1}{2}(\eta_\theta-\eta_{\theta|\mathbf{y}})^T\mathbf{C}_\theta^{-1}(\eta_\theta-\eta_{\theta|\mathbf{y}})$$

$$\int_\theta q(\theta)\log p(\mathbf{y},\theta|\lambda)d\theta =$$

$$-\tfrac{1}{2}\log|\mathbf{C}_\lambda| - \tfrac{1}{2}(\mathbf{y}-\mathbf{X}\eta_{\theta|\mathbf{y}})^T\mathbf{C}_\lambda^{-1}(\mathbf{y}-\mathbf{X}\eta_{\theta|\mathbf{y}}) + const$$

The overall cost function $\mathcal{L}(q,\lambda)$ then becomes

$$-\tfrac{1}{2}\log|\mathbf{C}_\lambda| - \tfrac{1}{2}\log|\mathbf{C}_\theta| - \tfrac{1}{2}(\mathbf{y}-\mathbf{X}\eta_{\theta|\mathbf{y}})^T\mathbf{C}_\lambda^{-1}(\mathbf{y}-\mathbf{X}\eta_{\theta|\mathbf{y}}) + \tfrac{1}{2}\log|\mathbf{C}_{\theta|\mathbf{y}}|$$
$$-\tfrac{1}{2}(\eta_\theta-\eta_{\theta|\mathbf{y}})^T\mathbf{C}_\theta^{-1}(\eta_\theta-\eta_{\theta|\mathbf{y}}) + const$$

If ϵ is assumed to have a distribution as $\epsilon \sim \mathcal{N}(\mathbf{0}, \lambda_1^{-1}\mathbf{I})$ and $\theta \sim \mathcal{N}(\mathbf{0}, \lambda_2^{-1}\mathbf{V})$
The objective function $\mathcal{L}(q, \lambda) = \log p(\mathbf{y}|\lambda)$ can be expressed as

$$\frac{I}{2}\log\lambda_1 + \frac{M}{2}\log\lambda_2 - \frac{1}{2}\lambda_1(\mathbf{y} - \mathbf{X}\eta_{\theta|\mathbf{y}})^T(\mathbf{y} - \mathbf{X}\eta_{\theta|\mathbf{y}}) - \frac{1}{2}\log|\lambda_1\mathbf{X}^T\mathbf{X} + \lambda_2\mathbf{V}^{-1}|$$
$$-\frac{1}{2}\lambda_2(\eta_\theta - \eta_{\theta|\mathbf{y}})^T\mathbf{V}^{-1}(\eta_\theta - \eta_{\theta|\mathbf{y}}) + const$$

The ReML algorithm for maximizing $\mathcal{L}(q, \lambda)$ is
repeat {
E-step: Compute the expectation of θ by;

$$\mathbf{C}_{\theta|\mathbf{y}} = (\lambda_1^{(n)}\mathbf{X}^T\mathbf{X} + \lambda_2^{(n)}\mathbf{V}^{-1})^{-1}$$
$$\eta_{\theta|\mathbf{y}} = \mathbf{C}_{\theta|\mathbf{y}}(\lambda_1^{(n)}\mathbf{X}^T\mathbf{y} + \lambda_2^{(n)}\mathbf{V}^{-1}\eta_{\theta|\mathbf{y}})$$

M-step: re-estimate λ

$$\lambda_1^{(n+1)} = (I - \lambda_1^{(n)}Tr[\mathbf{T}\mathbf{X}^T\mathbf{X}])/((\mathbf{y} - \mathbf{X}\eta_{\theta|\mathbf{y}})^T(\mathbf{y} - \mathbf{X}\eta_{\theta|\mathbf{y}}))$$
$$\lambda_2^{(n+1)} = (M - \lambda_2^{(n)}Tr[\mathbf{T}\mathbf{V}^{-1}])/(\eta_{\theta|\mathbf{y}}^T\mathbf{V}^{-1}\eta_{\theta|\mathbf{y}})$$

} *until* $\mathcal{L}(q, \lambda)$ *converges*

Generative Model

Assuming that f_i and g_i are the source and the reference (template) image values at their i^{th} voxels, respectively and

$$g_i = g(\theta) = g(\mathbf{y}(\mathbf{x}_i, \theta)) = g(\mathbf{x}_i + \Phi(\mathbf{x}_i)\theta)$$

is the interpolated value of the deformed template image at point $\mathbf{y}_i$ where $\Phi(\mathbf{x}_i)$ is the matrix whose columns are the m spatial basis vectors and then the observation at iteration (n) in voxel i with coordinate $\mathbf{x}_i$ is $\mathbf{Y}_i^{(n)} = f_i - g_i$

and the optimization function $\mathcal{L}$ is

$$\mathcal{L} = \mathcal{E} + \mathcal{H} = \lambda_1 \frac{1}{2} \mathbf{Y}^{(n)T} \mathbf{Y}^{(n)} + \lambda_2 \frac{1}{2} \theta^{(n)T} \mathbf{C}_\theta^{-1} \theta^{(n)}$$

where $\lambda = [1/\sigma^2 \ \beta]^T$ are the hyperparameters.

REML Algorithm for Non-linear Generative Model

1. Initialize the $\theta_{M\times 1}$ and λ ($\lambda_1 = 1/\sigma^2$ and $\lambda_2 = \beta$).
2. E-Step: compute the MAP estimate of θ by maximizing $\mathcal{L} = \mathcal{E} + \mathcal{H} = \lambda_1 \frac{1}{2}\mathbf{Y}^{(n)T}\mathbf{Y}^{(n)} + \lambda_2 \frac{1}{2}\theta^{(n)T}\mathbf{C}_\theta^{-1}\theta^{(n)}$
   repeat $\{$
   $\mathbf{Y}_i^{(n)} = (f_i - g_i)$ where $g_i = g(\mathbf{y}(\mathbf{x}_i, \theta)) = g(\mathbf{x}_i + \Phi(\mathbf{x}_i)\theta)$ for $i = 1, \ldots, I$
   $r_1 = \mathbf{Y}^{(n)T}\mathbf{Y}^{(n)}$ and $r_2 = \theta^{(n)}\mathbf{C}_\theta^{-1}\theta^{(n)}$
   $\theta^{(n+1)} = \theta^{(n)} - (\lambda_1 \frac{\partial \mathbf{Y}^{(n)T}}{\partial \theta}\frac{\partial \mathbf{Y}^{(n)}}{\partial \theta} + \lambda_2\mathbf{C}_\theta^{-1})^{-1}(\lambda_1 \frac{\partial \mathbf{Y}^{(n)}}{\partial \theta}^T\mathbf{Y}^{(n)} + \lambda_2\mathbf{C}_\theta^{-1}\theta^{(n)})$
   $\}$ until $\mathcal{L} = \lambda_1 r_1 + \lambda_2 r_2$ converges.
3. M-Step: Re-estimate hyperparameters λ by maximizing $\mathcal{L}(q, \lambda)$
   repeat $\{$
   $\lambda_1^{(n+1)} = \left(I - \lambda_1^{(n)} Tr\left[(\lambda_1^{(n)} \frac{\partial \mathbf{Y}^{(n)T}}{\partial \theta}\frac{\partial \mathbf{Y}^{(n)}}{\partial \theta} + \lambda_2\mathbf{C}_\theta^{-1})^{-1}\frac{\partial \mathbf{Y}^{(n)T}}{\partial \theta}\frac{\partial \mathbf{Y}^{(n)}}{\partial \theta}\right]\right)/r_1$
   $\lambda_2^{(n+1)} = \left(M - \lambda_2^{(n)} Tr\left[(\lambda_1^{(n)} \frac{\partial \mathbf{Y}^{(n)T}}{\partial \theta}\frac{\partial \mathbf{Y}^{(n)}}{\partial \theta} + \lambda_2\mathbf{C}_\theta^{-1})^{-1}\mathbf{C}_\theta^{-1}\right]\right)/r_2$
   $\}$ until
   $\mathcal{L}(q, \lambda) = \frac{1}{2}I \log\lambda_1 + \frac{1}{2}M\log\lambda_2 - \frac{1}{2}\lambda_1 r_1 - \frac{1}{2}\lambda_2 r_2 + \frac{1}{2}\log|\lambda_1 \frac{\partial \mathbf{Y}^{(n)T}}{\partial \theta}\frac{\partial \mathbf{Y}^{(n)}}{\partial \theta} + \lambda_2\mathbf{C}_\theta^{-1}|$
   converges.

Modified Algorithm Dealing with the Smoothness Problem

Images are normally smoothed before registering.
The EM modeling assuming that covariance of the error to be independent and identically distributed *i.e.* $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ is not valid any more. Therefore ϵ is modeled as $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{S})$ Since $\mathbf{S}$ is likely to change at the end of each iteration, FWHM for smoothing is re-estimated after each M-step from which the $\mathbf{S}$ is computed.

Repeat

1. E-Step: compute robust estimation of θ
   $$\mathbf{T} = (\lambda_1^{(n)} \frac{\partial \mathbf{Y}^{(n)T}}{\partial \theta} \frac{\partial \mathbf{Y}^{(n)}}{\partial \theta} + \lambda_2^{(n)} \mathbf{C}_\theta^{-1})^{-1}$$
   $$\theta = \mathbf{T}\lambda_1^{(n)} \frac{\partial \mathbf{Y}^{(n)T}}{\partial \theta} \mathbf{Y}^{(n)}$$
2. M-Step: re-estimate hyperparameters λ
   $$\lambda_1^{(n+1)} = Tr[\mathbf{S} - \lambda_1^{(n)} \mathbf{T} \frac{\partial \mathbf{Y}^{(n)T}}{\partial \theta} \mathbf{S} \frac{\partial \mathbf{Y}^{(n)}}{\partial \theta}] / ((\mathbf{Y}^{(n)} - \frac{\partial \mathbf{Y}^{(n)}}{\partial \theta}\theta)^T \mathbf{S} (\mathbf{Y}^{(n)} - \frac{\partial \mathbf{Y}^{(n)}}{\partial \theta}\theta))$$
   $$\lambda_2^{(n+1)} = (M - \lambda_2^{(n)} Tr[\mathbf{T}\mathbf{C}_\theta^{-1}]) / (\theta^T \mathbf{C}_\theta^{-1} \theta)$$

until convergence.

1-D Nonlinear Registration Using ReML

```
% Nonlinear REML
% The objective is to estimate the hyper-parameters used to
% simulate the data:
w = [10 5e-6];
m = 256; % 1-D Image size
n = 50; % Number of DCT basis functions
B    = dctmtx(m)'; B = B(:,1:n); % Basis functions
XX2  = diag((1:n).^3.0); % Form of priors
x    = (1:m)';
% Simulated images
% ----------------
f = zeros(m,1);
f(60:200) = exp(randn(length(60:200),1))*10;
f = conv2(f,exp(-0.5 * (-40:40)'.^2/50)/sqrt(2*pi*50),'same');
p = (XX2*w(2))^(-0.5)*randn(n,1);
t = x+B*p;
g = interp1(x,f,t,'*linear'); g(~isfinite(g))=0;
g = g +randn(size(t))*w(1)^(-0.5);
 B    = dctmtx(m)'; B = B(:,1:n); % Basis functions
 XX2  = diag((1:n).^3.0); % Form of priors
 x    = (1:m)';
 % Starting estimates
 beta = zeros(n,1);
 w    = [1 0.01];
 rml  = [];
```

```
 for it=1:100,
 % E-Step: obtain MAP solution using current hyper-parameters
 f1    = interp1(x,f,x+B*beta,'*linear'); f1(~isfinite(f1))=0;
 df1   = interp1(x,gradient(f),x+B*beta,'*linear'); df1(~isfinite(df1))=0;
 Y1    = f1 - g;
 r1    = Y1'*Y1;
 r2    = beta'*XX2*beta;
 r     = r1*w(1)+r2*w(2);
 for i=1:20,
 obeta = beta; oY1 = Y1;
 or1   = r1;   or2 = r2;or = r;
X1    = repmat(df1,[1 size(B,2)]).*B;
% chain rule df_i / dtheta . d (phi theta )/dtheta
XX1   = X1'*X1;
XY1   = X1'*Y1;
T     = inv(w(1)*XX1 + w(2)*XX2); % ??? matrix inevrsion lemma??
%beta  = T* (w(1)*XX1*beta - w(1)*XY1);
beta = beta - T*(w(1)*XY1 + w(2)*XX2*beta);
f1    = interp1(x,f,x+B*beta,'*linear'); f1(~isfinite(f1))=0;
df1   = interp1(x,gradient(f),x+B*beta,'*linear'); df1(~isfinite(df1))=0;
Y1    = f1 - g;
r1    = Y1'*Y1;
r2    = beta'*XX2*beta;
r     = r1*w(1)+r2*w(2);
fprintf('%g\t%g\t| %g\n', r1*w(1),r2*w(2),r1*w(1)+r2*w(2));
subplot(2,2,1);plot(x,g,'r',x,f1,'b',x,f,'b:');
title('"Images"');
subplot(2,2,2);plot(x,x+B*beta,'b',x,t,'r');axis([1 m 1 m]);
title('"Deformations"');
drawnow;
```

```
if r>or,
beta = obeta; Y1 = oY1;
r1   = or1;   r2 = or2; r = or;
end;
if (or-r)/(r+or)<0.00001, break; end;
end;
% REML cost function
rml=[rml (0.5*log(det(XX1+XX2))-...
0.5*r-0.5*(log(1/w(1))*m+log(1/w(2))*n))+0.5*log(det(T))];
fprintf('%g %g %g\n',rml(end),w)
% M-Step: Re-estimate hyper-parameters
w(1)  = (m-trace(T*XX1*w(1)))/(r1+eps);
w(2)  = (n-trace(T*XX2*w(2)))/(r2+eps);
subplot(2,1,2);
plot(rml(1:end));
xlabel('# Msteps');
ylabel('Objective function');
drawnow;
if it>1 & abs(rml(end)-rml(end-1))<0.001, break; end;
end;
```