Medical Image Analysis - Tensor B-Spline based warp exercise

Mikkel Damgaard Olsen and Olivier Jais-Nielsen

November 14, 2009

1 Introduction

The purpose of this exercise is to evaluate the transformation between two mid-saggittal brain MR slices (cf. figure 1) as a tensor B-spline. We will use a Gauss-Newton optimization to perform the registration.

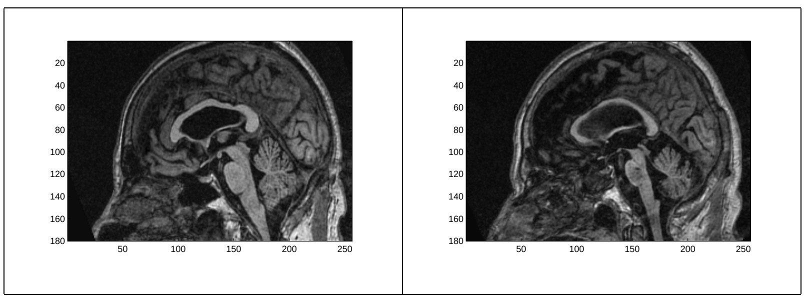


Figure 1: Two mid-saggittal brain MR slices (the left one is used as reference and the right one is the template)

The algorithm minimizes the dissimilarity (defined as the sum-of-squared differences) between the template and the reference as well as a regularizer (defined as the sum-of-squared B-Spline weights, to keep the transformation small between two iterations). The initial dissimilarity is shown in figure 2.

2 Algorithm

The algorithm minimizes the objective function $J(w) = \|T(x+Qw) - R\|^2 + \alpha \|w\|^2$ where w is the spline weights vector, x represents the spatial identity, Q contains the values of the bidimensional spline at each knot, T is the template, R is the reference and α is a parameter. The first term is the dissimilarity and the second the regularizer.

The number of iterations necessary for convergence is set experimentally. We found that, for the values of α and the number of knots used, after 50 iterations the template hardly evolves any more.

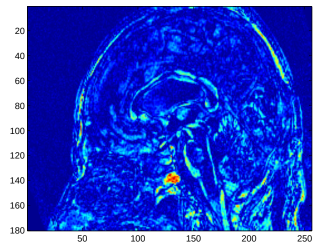


Figure 2: Initial dissimilarity

2.1 Initialization

```
% Load MR scans. "mr1" is the reference and "mr2" the template.
load mr.mat

% Size of images.
[R C]=size(mr1);

% Number of knots (total numper is "p*p").
p=3;

% Parameter "alpha"
alpha=100;

% Number of iterations.
k_max=50;

% Definition of the unidimensional sets of knots. "s" is for the vertical and "t" for the horizontal. The knots are equidistantly distributed along their direction. 3 additional knots are placed on top of the boundary knots so the spline will be equal to zero outside of the image.
s=augknt(linspace(0,R,p),3);
t=augknt(linspace(0,C,p),3);

% Construction of the splines, "B1" and "B2". We define "Q1" and "Q2" containing the values of the splines at each unidimensional knot. "Q" contains the values of the bidimensional spline at each knot.
B1=spmak(s,eye(p));
B2=spmak(t,eye(p));
```

```
Q1=fnval(B1,1:R)';
Q2=fnval(B2,1:C)';
Q=kron(speye(2),kron(Q2,Q1));

% "x1" and "x2" contain all the image coordinates with no transformation.
[x1 x2]=ndgrid(1:R,1:C);

% The initial value of the spline weights vector "w" is 0.
w=zeros(size(Q,2),1);

% Main loop. The loop keeps going on until "stop" equals "true". The number of iterations is stored in "k".
stop=false;
k=0;
while ~(stop)
```

2.2 Transformation update

The coordinates transformation is defined as the identity plus the displacement: $y(x) = x + Qw$ where Qw is the displacement given by the spline for the current weights.

```
% Computation of the transformation
y=[x1(:); x2(:)]+Q*w;

% "y1" contains the horizontal component of the transformation and "y2" the vertical component.
y1=reshape(y(1:end/2),R,C);
y2=reshape(y(end/2+1:end),R,C);
```

2.3 Transformed image and its spatial derivatives computation

```
% "Ty" is the transformed image.
Ty=interp2(mr1,y2,y1,'linear',0);

% "GradTy" is a matrix composed of two blocks. The left one is a diagonal matrix which diagonal is the vertical gradient. The right one is the equivalent for the horizontal gradient.
[GradTy2 GradTy1]=gradient(Ty);
GradTy=[spdiags(GradTy1(:),0,R*C,R*C) spdiags(GradTy2(:),0,R*C,R*C)];
```

2.4 Minimization of the objective function

We compute the new value of w as $w + \Delta w$ with Δw minimizing $J(w + \Delta w)$ by solving the equation $(A^T A + \alpha I)\Delta w = A^T(R - T(y)) - \alpha w$ where $A = \nabla T(y)Q$.

```
% Computation of "A".
A=GradTy*Q;

% Minimization.
AtA=A'*A;
delta_w=(AtA+alpha*eye(size(AtA)))\(A'*(mr2(:)-Ty(:))-
    alpha*w);

% Update of the value of "w"
w=w+delta_w;
```

2.5 End

```
% Update number of iteration
k=k+1;

% If the total number of iterations is reached, we stop
    the loop.
if k==k_max
    stop=true;
end

% End of the main loop.
end
```

3 Results

We tested the algorithm with 3×3 knots, 5×5 knots, 7×7 knots for $\alpha \in \{0.1; 1; 10; 100\}$. Cf figure 3.

Although very similar, the best results are obtained for the higher number of knots, which seems logical as it implies a better adaptation to the complexity of the transformation. The results for higher values of α seem slightly better as well: the zones of high dissimilarity (the bright zones) are smaller.

3.1 Conclusion

A tensor B-spline model for this non linear transform computed with a Gauss-Newton method is very efficient: the zones of strong dissimilarity are solved and the computing time is limited. However, the fact that the best results were obtained with a strong regularizer shows that in the present case, the initial dissimilarity was quite low and thus more important transform might not perform that well.

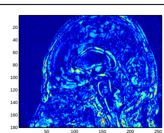
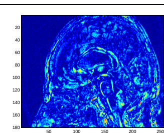
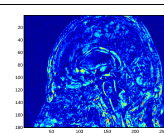
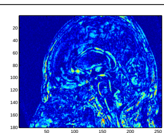
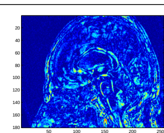
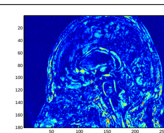
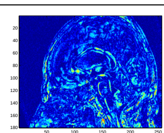
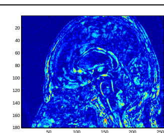
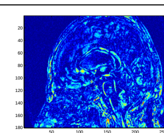
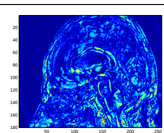
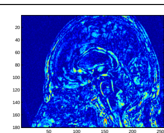
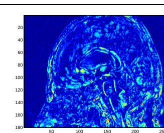
	3×3 knots	5×5 knots	7×7 knots
$\alpha = 0.1$			
$\alpha = 1$			
$\alpha = 10$			
$\alpha = 100$			

Figure 3: Resulting dissimilarities