Medical Image Analysis - Point based registration without correspondences exercise

Mikkel Damgaard Olsen and Olivier Jais-Nielsen

November 14, 2009

1 Hungarian algorithm with euclidean distance based similarity measure

1.1 Metacarpal bones

We perform point to point matching between the contours of two metacarpal bones, using the euclidian distance as a disimilarity measure. Cf. figure 1.

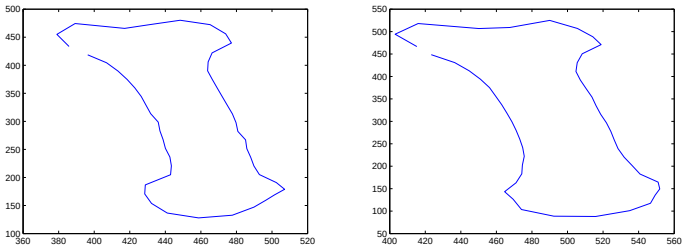

Figure 1: Two metacarpal bones contours.

1.1.1 Crude alignment of the shapes

We first center each shape on the origin and scale it so it has a unit size. Cf. figure 2.

```
1 % We load the data.
2 load meta.mat
3
4 % We extract the shapes.
5 c1=meta(:,1);
6 c2=meta(:,2);
7
8 % We compute the centroids of the shapes.
```

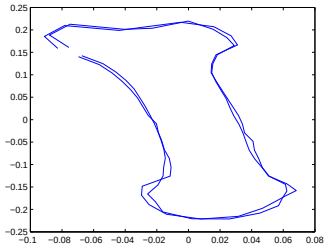

Figure 2: Crude alignment by centering and scaling the shapes

```
 9  meanc1=mean(c1);
10  meanc2=mean(c2);
11
12  % We compute the centered shapes.
13  c01=c1-meanc1;
14  c02=c2-meanc2;
15
16  % We compute the size of the shapes.
17  S1=norm(c01);
18  S2=norm(c02);
19
20  % We scale the shapes.
21  cs01=c01/S1;
22  cs02=c02/S2;
```

1.1.2 Correspondences computation

We use the Hungarian algorithm with the euclidian distance as a similarity measure to compute the correspondences between the two shapes. Cf. figure 3.

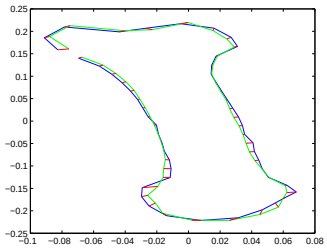

Figure 3: The two bones shapes with correpondences performed with Euclidian similarity.

```
% We extract the number of points in each shape.
N1=size(c1,1);
N2=size(c2,1);

% We create "e", wich (i, j)-th coeficient is the
    difference between the i-th coeficient of the first
    shape and the j-th coeficient of the second shape.
a=repmat(cs01,1,N2);
b=repmat(cs02.',N1,1);
e=a-b;

% We compute "A" as the matrix of the modulus of the
    coeficients of "e". It is our cost matrix.
A=e.*conj(e);

% We perform the Hungarian algorithm. "c"'s i-th
    coeficient is the index of the point of the second
    shape matching the i-th point of the first shape.
C=hungarian(A);
```

The results are rather good. However we can see that the transformation between the two original shapes was mostly a scaling and a translation.

1.2 Lungs annotated from a lung radiograph

We perform point to point matching between the contours of two lung radiographs (Cf. figure 4), using the euclidian distance as a disimilarity measure. We skip the centering and scaling part as both lungs are already centered and appear to have the same scale. Moreover, their shape being quite different, they don't have the same size thus scaling might alter the correspondences.

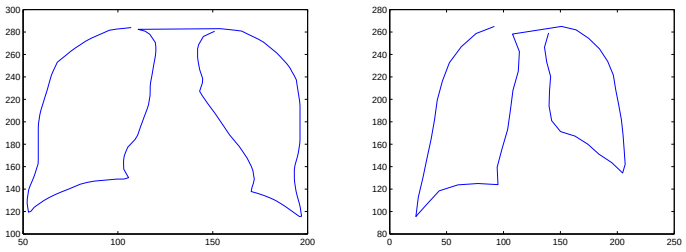

Figure 4: Two lungs contours.

```
% We load the data.
load lung.mat

```

```
4  % We extract the shapes. The second shape is subsampled
       to 50 points instead of 200.
5  c1=lung1;
6  c2=lung2(1:4:200);
7
8  % We extract the number of points in each shape.
9  N1=size(c1,1);
10 N2=size(c2,1);
11
12 % We create "e", wich (i, j)-th coeficient is the
       difference between the i-th coeficient of the first
       shape and the j-th coeficient of the second shape.
13 a=repmat(c1,1,N2);
14 b=repmat(c2.',N1,1);
15 e=a-b;
16
17 % We compute "A" as the matrix of the modulus of the
       coeficients of "e". It is our cost matrix.
18 A=e.*conj(e);
19
20 % However, because of the subsampling, both shapes have
       different numbers of points thus "A" is not square. We
        had dummy cost values (arbitrarily equal to 0.5) to
       make "A" a square matrix.
21 B=[A 0.5*ones(200,150)];
22
23 % We perform the Hungarian algorithm. "c"'s i-th
       coeficient is the index of the point of the second
       shape matching the i-th point of the first shape.
24 C=hungarian(B);
```

The result is quite bad: both contours have very different shapes thus the closest match is not always the most relevant. Cf figure 5.

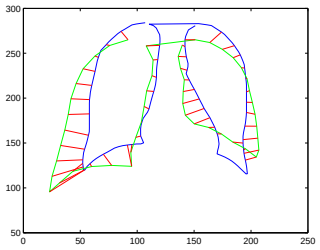

Figure 5: The two lung shapes with correpondences performed with Euclidian similarity.

2 Hungarian algorithm with shape context similarity measure

To get better correspondences, we use shape context as similarity measure.

2.1 Direct correspondences computation

2.1.1 Shape contexts computation

First, we compute the shape contexts of each shape. The size of the contexts are set to 25% of the maximum range of the coordinates of the shape's points.

```
1 rfrac=0.25;
2 ctx1=shapecontext(c1,rfrac);
3 ctx2=shapecontext(c2,rfrac);
```

We can check wether the shapecontexts previously matching points are similar. For some pairs this is the case, but not for all of them which is coherent with the fact some correspondences were incorrect. Cf figure 6.

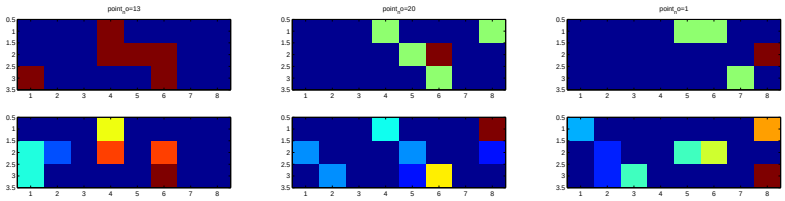

Figure 6: Shape contexts of previously determined pairs of points. The first two pairs are coherent while for the third the shape contexts are very dissimilar.

2.1.2 Cost matrix computation

From the shape contexts, we compute the cost matrix. Cf figure 7.

```
1 % We compute the cost matrix with no additional dummy-
      points. The cost of necessary dummy points is 0.5
2 cost=computecostmatrix(ctx1,ctx2,0,0.5);
```

The right part of the cost matrix corresponds to the dummy points. On the real part, we can see a "valley" surrounding the diagonal which as sensibly lower values than the rest. As the numbering of the points is distributed in the same way on the two shapes, it shows that most points have their corresponding point at approximately the same position, on the other shape.

2.1.3 Resulting correspondences

We compute the correspondences with the hungarian algorithm. Cf. figure 10.

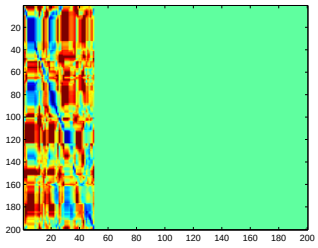

Figure 7: Cost matrix from shape contexts

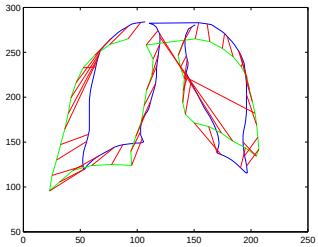

Figure 8: Correspondences resulting from shape contexts

The result is bad in many places. Some correspondences are completely incorrect, like for instance those that go from the right lung to the left.

2.2 Correspondences with a constrained cost matrix

2.2.1 Constraining the cost matrix

Some of the bad correspondences of the previous attempt can be avoided by specifiyng that points of one lung can only correspond to points of a lung of the same side. Moreover, it is safe to assume that points form one lung datasaet cannot shift more than 20% along the outline. As half the points describe the left lung and the other half, the other lung, and as the numbering is coherent between the two shapes, these constraints can be implemented in the shape of the cost matrix itself, by setting a high cost to the zones corresponding to impossible correspondences. Cf. figure 9. The value of this cost is chosen as 1.2, as the maximum value for the other costs is 1.

```
1 % The function "Context" takes as input two shapes and
      the maximum shift allowed represented as a percentage
      of the outline. It returns a cost matrix obtained from
       shape contexts and constrained.
2
3 function [C] = Context(c1,c2,percent)
4
```

```
5  % We compute the shape contexts.
6  rfrac=0.25;
7  ctx1=shapecontext(c1,rfrac);
8  ctx2=shapecontext(c2,rfrac);
9
10 % We compute the cost matrix.
11 cost_new=computecostmatrix(ctx1,ctx2,0,0.5);
12
13 % We stretch the cost matrix so it becomes square. To
       actually stretche, we add three columns of zeros
       between each column of the initial matrix.
14 cost_streched=zeros(200,200);
15 for c=1:50
16         cost_streched(:,(c-1)*4+1)=cost_new(1:200,c);
17 end
18
19
20 % The length of the outline is evaluated as the initial
       number of points that compose it. It is 200 points for
        one of the shapes, thus 100 points for one lung.
21 length_outline=100;
22
23 % We define "k_size", a size in the matrix equivalent to
       the specified percentage.
24 percent=round(length_outline*percent);
25 k_size=percent;
26 k_size_inv=100-k_size;
27
28 % We create a constraints mask: a matrix filled with
       zeros where we want to constrain and with ones
       anywhere else. The matrix is devided in 4 quarters.
29 % We start by selecting the coefficients of the global
       diagonal and their neighbours in a radius of "k_size"
       to implement the maximum shift constraint.
30 costmaskdiag=triu(tril(ones(size(cost_streched)),k_size)
       ,-k_size);
31
32 % However due to cyclic numbering, we have to add
       portions situated at the corners of the quarters. (For
        instance, on a lung of one of the shapes, the point
       no 0 is neighbour with the point no 100)
33 costmaskcorner=[[triu(ones(100,100),k_size_inv) + ...
34                  tril(ones(100,100),-k_size_inv)] zeros
                       (100,100); ...
35                 zeros(100,100) [triu(ones(100,100),
                       k_size_inv) + ...
```

```
36                      tril(ones(100,100),-k_size_inv)];
37
38 % We combine the first two masks.
39 costmaskPercent=costmaskdiag+costmaskcorner;
40
41 % Only the portions included in the top-left and the
       bottom-right quarters are kept (correspondences
       between the right lung and the left lung are not
       allowed).
42 costmaskRL=[ones(100,100) zeros(100,100); zeros(100,100)
       ones(100,100)];
43
44 % We combine this last mask with the previous two.
45 costmask=costmaskPercent.*costmaskRL;
46
47 % We apply the mask to the stretched cost matrix.
48 cost_streched_masked=cost_streched.*costmask+(costmask
       ==0)*1.2;
49
50 % We unstretch the matrix.
51 cost_unstreched=cost_streched_masked(:,1:4:200); cost_new
       (:,1:50)=cost_unstreched;
52
53 % We apply the Hungarian algorithm to the constrained
       cost matrix.
54 C=hungarian(cost_new);
```

With the previous function, the maximum allowed shift is actually near the given percentage but not exactly equal to it, because of the stretching of the matrix.

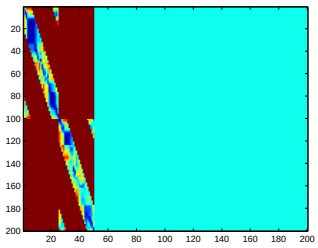

Figure 9: Constrained cost matrix from shape contexts

2.2.2 Resulting correspondences

```
1 % We compute the correspondences with the constrained
       matrix.
```

```
C_constraint=Context(c1,c2,0.2);
```

The resulting correspondences are much better. All the obvious erroneous correspondences are avoided. Cf. figure 10.

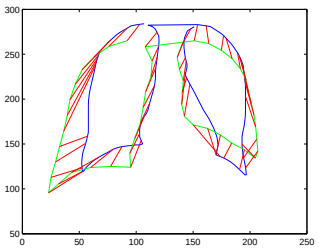

Figure 10: Correspondences resulting from shape contexts with constraints

3 Shape context and TPS based registration algorithm

The algorithm starts by performing the preceding matching. Then the first shape is warped in a smoothing TPS so that it fits the other shape's points that were matched. The algorithm starts again until it converges.

```
% The matching described before is computed.
T0=c1(C_constraint(1:50));

% We set the shapes in a format the "tps" function uses.
T0=[real(T0); imag(T0)];
S0=c2;
S0=[real(S0); imag(S0)];

% Main loop.
stop=0;
norm_old=100;
while ~stop

        % We warp the first shape in the TPS, with the
            corresponding points of the second shape.
        S0_warped=tps(T0,S0);

        % We perform the matching.
        k=size(S0,1)/2;
        c2=[S0_warped(1:k) + sqrt(-1)*S0_warped(k+1:end)
            ];
        C=Context(c1,c2,0.2);
        T0=c1(C(1:50));
```

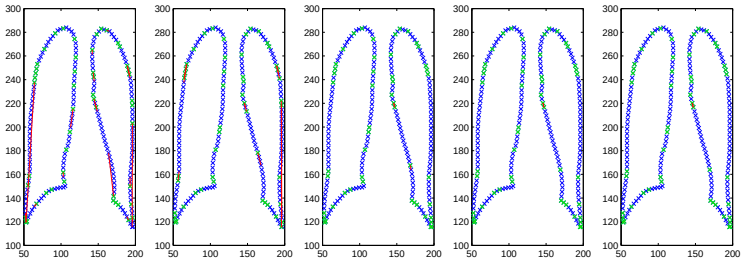

Figure 11: Shape context and TPS based registration algorithm - results at each iteration

```
22          T0=[real(T0); imag(T0)];
23          S0=c2;
24          S0=[real(S0); imag(S0)];
25
26          % We test to see if the new pair of shapes are
                significantly closer than before. If this isn'
                t the case, we stop the algorithm.
27          norm_new=norm(T0-S0);
28          if abs(norm_new-norm_old)<0.5
29                  stop=1;
30          end
31          norm_old=norm_new;
32  end
```

4 Conclusion

Shape context similarity is definitely a relevant measure for shapes as it puts a point in its context and thus allow substantial deformation. Its use with teh Hungarian algorithm is very efficient because of its flexibility: the ability to add extra reliable information that the algorithm cannot guess. Finally, the use of TPS is the perfect jonction between points matching and actual deformation. The results are all the more satisfactory that the initial shapes had strong differences in comparison to their global similarity. Moreover, the algorithm converges very quickly, once the first correspondences are established.