Evaluation of Plastimatch B-Spline Registration on the EMPIRE10 Data Set

Gregory Sharp[1], Marta Peroni[1,2,3], Rui Li[1],
James Shackleford[4], Nagarajan Kandasamy[4]

[1] Massachusetts General Hospital, Boston, MA
[2] Politecnico di Milano, Milan, Italy
[3] Massachusetts Institute of Technology, Cambridge, MA
[4] Drexel University, Philadelphia, PA

Abstract. In our open source software package "Plastimatch", we provide a B-spline based deformable image registration method with an efficient GPU and multicore implementation. We have participated in the EMPIRE10 grand challenge to evaluate our method on the task of registering a set of benchmark thoracic CT data sets. The results demonstrate that our method ranks 12 on the 34 methods evaluated. On the set of statistics we computed, we have shown that our registration methods can register the benchmark images at full resolution in $0.4 \sim 5.7$ minutes with good results based on the Dice and invertibility statistics.

Keywords: Deformable Image Registration, B-spline

1 Introduction

Deformable image registration is an important and challenging research area in medical imaging, and it has been successfully applied to various problems in the clinical setting. One such problem is pulmonary CT image registration as highlighted in this EMPIRE10 grand challenge.

Due to the elastic nature of lung tissue deformations, deformable image registration is needed to register a pair pulmonary CT images. We have developed a B-spline based deformable image registration method with efficient implementation as part of our open source software package "Plastimatch" [1].

By using a grid alignment scheme in our method, we can significantly accelerate the B-spline interpolation and gradient computation, thus speeding up the registration process. The improved efficiency of the registration process is reported as running time on the benchmark dataset provided by EMPIRE10. It takes $0.4 \sim 5.7$ minutes to register to a pair of 3D CT benchmark images at full resolution.

One of our goals to participate this grand challenge is see how accurate our method compared to other methods. Based on the evaluation metrics, we achieved an average ranking of 12 out of 34 methods submitted. Combined with our own evaluation metrics of Dice coefficients and root mean squared error (RMSE), our method is accurate and certainly demonstrate the potential of clinical applicability. However, as all these evaluation metrics are not conclusive

in terms of local registration error, visual inspection is still needed in the clinical setting.

2 Methods

This section provides an overview of B-spline registration and describes a grid-alignment scheme to significantly accelerate two major stages of the overall registration process: B-spline interpolation and gradient computation.

The B-spline registration method uses cubic B-spline curves to define a displacement field that maps the voxels in the moving image to those in a reference (or static) image. Since the vector field is defined in a parametric fashion (that is, in terms of coefficients provided by a set of control points), a cost function that quantifies the similarity between the fixed and moving images can be specified and registration can be posed as an optimization problem. This requires that we evaluate C, the cost function corresponding to a given set of spline coefficients, and $\partial C/\partial P$, the change in the cost function with respect to the coefficient values P at each individual control point. The registration process then becomes one of iteratively defining coefficients P, performing B-spline interpolation, evaluating the cost function C, calculating $\partial C/\partial P$ for each control point, and performing gradient-descent optimization to generate the next set of coefficients.

2.1 B-spline Interpolation

By aligning the voxel grid with a uniformly-spaced control grid, as shown in Fig. 1(a), the image volume becomes partitioned into many equally sized tiles. In the shown 2D example, the control grid partitions the voxel grid into tiles with 5×4 voxels each. The vector field at a voxel within a tile is influenced by the 16 control points in the tile's immediate vicinity, and the values of the B-spline

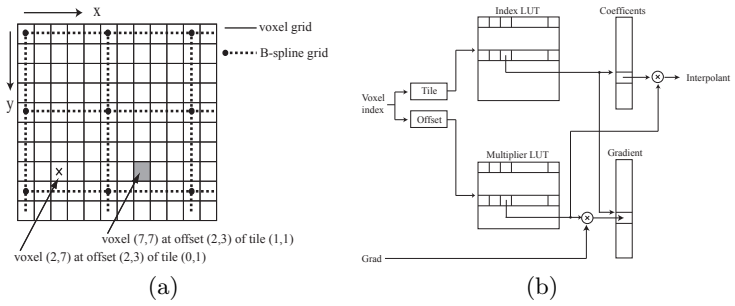

Fig. 1. (a) A portion of a 2D image upon which a 5×4 control-point grid is superimposed with the voxel grid aligned with the control grid. Since both the marked voxel and the grayed voxel are located at the same relative offset within their respective tiles, both voxels will use the same $\beta_l(u)\beta_m(v)$ value. (b) For aligned grids, lookup tables can accelerate the registration process by eliminating redundant computations.

basis functions depends upon the voxel's local coordinates, or offset, within the tile. Notice that the two marked voxels in Fig. 1(a), while residing at different locations within the image, both possess the same offsets within their respective tiles. This results in the B-spline basis function product yielding identical values when evaluated at these two voxels. This property allows us to pre-compute all relevant B-spline basis function products once instead of recomputing the evaluation for each individual tile.

In the 3D case, the vector field at any given voxel is determined by the 64 control points in the immediate vicinity of the voxel's housing tile. This configuration forms the basis of an analytic expression for the continuous vector field $\boldsymbol{\nu}$. B-spline interpolation is performed for each voxel within a tile with respect to the 64 control point coefficients that form the local support for the operation. For example, the B-spline interpolation yielding the x-component of the displacement vector for a voxel located at coordinate (x, y, z) is

$$\nu_x(x,y,z) = \sum_{l=0}^{3}\sum_{m=0}^{3}\sum_{n=0}^{3} \beta_l(u)\beta_m(v)\beta_n(w)P_x(i+l, j+m, k+n), \qquad (1)$$

where P_x is the spline coefficient defining the x component of the displacement vector for one of the 64 control points that influence the voxel. We obtain the spline basis functions β as follows. Let N_x, N_y, and N_z denote the distance between control points, in terms of voxels, in the x, y, and z directions, respectively. The volume is therefore segmented by the B-spline control point grid into many equal-sized tiles of dimensions $N_x \times N_y \times N_z$. The three dimensional indices x_t, y_t, and z_t of the tile within which the voxel at (x, y, z) falls is given by

$$x_t = \left\lfloor \frac{x}{N_x} \right\rfloor - 1, y_t = \left\lfloor \frac{y}{N_y} \right\rfloor - 1, z_t = \left\lfloor \frac{z}{N_z} \right\rfloor - 1. \qquad (2)$$

The local coordinates of the voxel within its tile, normalized between $[0, 1]$, are

$$u = \frac{x}{N_x} - \left\lfloor \frac{x}{N_x} \right\rfloor, v = \frac{y}{N_y} - \left\lfloor \frac{y}{N_y} \right\rfloor, w = \frac{z}{N_z} - \left\lfloor \frac{z}{N_z} \right\rfloor. \qquad (3)$$

Finally, the uniform cubic B-spline basis function β_l along the x direction is given by

$$\beta_l(u) = \begin{cases} \frac{(1-u)^3}{6} & : l = 0 \\ \frac{3u^3-6u^2+4}{6} & : l = 1 \\ \frac{-3u^3+3u^2+3u+1}{6} & : l = 2 \\ \frac{u^3}{6} & : l = 3, \end{cases} \qquad (4)$$

and similarly for β_m and β_n along the y and z directions, respectively.

A straightforward implementation of (1) to compute the displacement vector $\boldsymbol{\nu}$ for a single voxel requires 192 computations of the cubic polynomial B-spline basis function β as well as 192 multiplications and 63 additions. However, many of these calculations can be eliminated by implementing a data structure that

exploits symmetrical features that emerge as a result of the grid alignment, making the implementation of (1) much faster.

- All voxels residing within a single tile use the same set of 64 control points to compute their respective displacement vectors. So, for each tile in the volume, the corresponding set of control point indices can be pre-computed and stored in a lookup table (LUT), called the *Index LUT*. These indices serve as pointers to a coefficient table.
- Eq. 3 indicates that for a tile of dimension $N_w = N_x \times N_y \times N_z$, the number of $\beta(u)\beta(v)\beta(w)$ combinations is limited to N_w values. Furthermore, as Fig. 1(a) shows, two voxels belonging to different tiles but possessing the same normalized coordinates (u, v, w) within their respective tiles will be subject to identical $\beta(u)\beta(v)\beta(w)$ products. Therefore, we pre-compute the $\beta_l(u)\beta_m(v)\beta_n(w)$ product for all valid normalized coordinate combinations (u, v, and w) and store the results into a LUT called the *Multiplier LUT*.

Fig. 1(b) shows the data structure needed to support the above-described optimizations. For each voxel, its absolute coordinate (x, y, z) within the volume is used to calculate the tile number that the voxel falls within as well as the voxel's relative coordinates within that tile using (2) and (3), respectively. The tile number is used to query the *Index LUT*, which provides the coefficient values associated with the 64 control points influencing the voxel's interpolation calculation. The voxel's relative coordinates (u, v, w) within the tile determine its index within [0, N_w], which is used to retrieve the appropriate pre-computed $\beta(u)\beta(v)\beta(w)$ product from the *Multiplier LUT*. Computing ν_x, the x component of the displacement vector for the voxel, therefore, requires looping through the 64 entries of each LUT, fetching the associated values, multiplying, and accumulating. Similar computations are required to obtain ν_y and ν_z. The LUTs are stored in CPU cache, thereby achieving extremely fast lookup times.

Once the displacement vector field is generated, the moving image is deformed and compared to the static image in terms of the mean squared error (MSE) cost function, computed once per iteration by accumulating the square of the intensity difference between the fixed image S and the deformed moving image M as

$$C = \frac{1}{N} \sum_{z} \sum_{y} \sum_{x} (S(x, y, z) - M(x + \nu_x, y + \nu_y, z + \nu_z))^2, \tag{5}$$

where C is the cost function and N is the total number of voxels in the volume.

2.2 Gradient Computation and Optimization

Gradient descent optimization requires computing the partial derivatives of the cost function with respect to each control-point coefficient value. Recall that the cost function gradient $\partial C/\partial P$ quantifies the change in the cost function with respect to the coefficient values P at each individual control point. We decompose the cost function gradient for any given control point as

$$\frac{\partial C}{\partial P} = \sum_{(x,y,z)} \frac{\partial C}{\partial \boldsymbol{v}(x,y,z)} \frac{\partial \boldsymbol{v}(x,y,z)}{\partial P}. \tag{6}$$

This allows us to evaluate the cost function gradient's dependencies on the cost function and spline coefficients independently. The first term, $\partial C/\partial \boldsymbol{\nu}$, depends only on the cost function and is independent of the type of spline parameterization employed. The second term describes how the deformation field changes with respect to the control-point coefficients and can be found by simply taking the derivative of (1) with respect to P. This term is dependent only on the B-spline parameterization and is computed as

$$\frac{\partial \boldsymbol{v}(x,y,z)}{\partial P} = \sum_{l=0}^{3} \sum_{m=0}^{3} \sum_{n=0}^{3} \beta_l(u)\beta_m(v)\beta_n(w). \tag{7}$$

Since (7) depends only on the B-spline parameterization it will remain constant over all optimization iterations. This allows us to pre-compute and store (7) for each voxel prior to the optimization process. Note also, that the values generated by (7) are already available via the Multiplier LUT.

If MSE is used as the cost function, the first term in (6) can be re-written in terms of the moving image's spatial gradient $\nabla M(x,y,z)$ as

$$\frac{\partial C}{\partial \boldsymbol{\nu}(x,y,z)} = [S(x,y,z) - M(x+\nu_x, y+\nu_y, z+\nu_z)]\nabla M(x,y,z). \tag{8}$$

The above expression depends on the intensity values of the static and moving images, S and M, respectively, as well as the current value of the vector field $\boldsymbol{\nu}$. During each iteration, the vector field will change, modifying the correspondences between the static and moving images. So, unlike $\partial \boldsymbol{\nu}/\partial P$, $\partial C/\partial \boldsymbol{\nu}$ must be recomputed during each iteration of the optimization problem. Once both terms are computed, they are combined using the chain rule in (6).

2.3 Optimization

The coefficient values that minimize the registration cost function are found using L-BFGS-B, a quasi-Newton optimizer suitable for either bounded or unbounded problems [2]. During each iteration, the optimizer chooses a set of coefficient values; for these coefficient values (1)-(5) and (6)-(8) are used to compute the cost and gradient, respectively. The gradient values are transmitted back to the optimizer and the process is repeated for a set number of iterations or until the cost function converges to a local (or global) minimum.

3 Implementation

3.1 Synopsis of EMPIRE10 Data

The EMPIRE10 challenge data consists of 20 pairs of images, which are to be pairwise registered. A complete description of the data is described in the main

workshop article by Murphy et al [3]. Briefly, there are 18 human cases and 2 non-human (sheep) cases. Of the 18 human cases, two are synthetically warped, three are from 4D-CT, and 13 are breathhold scans. Only minor artifacts were visible in the 4D scans, and no obvious pathology changes were visible between scans on different day.

Each image was accompanied by a lung segmentation mask, which was constructed according to the method of van Rikxoort [4]. The masks were found to be of reasonably high quality, although some image pairs had small differences. These differences were most visible at the border between the lungs and great pulmonary vessels, where the segmentation choice is rather arbitrary.

3.2 Preprocessing

Our only preprocessing stage for the grand challenge was to perform lung masking. A similar strategy was evaluated by Wu et al. for whole thorax registration [5]. By performing segmentation and masking of moving (lung and mediastinum) and non-moving (ribcage) parts, and then registering each part separately, Wu demonstrated that both demons and B-spline based methods could achieve improved landmark accuracy.

During preprocessing, areas outside of the mask are filled with a constant value of -1200 Hounsfield Units prior to registration. Because this value is outside of the normal range of CT numbers, most registration algorithms will align the boundaries of the masked regions as part of their cost function. This method is particularly well suited for use with the MSE cost function.

3.3 Registration

Registration was performed using Plastimatch [6, 1], an open source software for deformable image registration. Plastimatch is designed for high-performance volumetric image registration, and features a pipelined, multi-stage registration framework and high-performance implementations of demons and B-spline based algorithms. The registration pipeline is controlled by a parameter file, which specifies the algorithm and settings to be used in each stage.

For the EMPIRE10 data, we used five processing stages. Two slightly different settings were used: one for the human images (Fig. 3.2 left) and one for the sheep images (Fig. 3.2 right) [1]. In the initial stage, the images are aligned by matching the geometric centers of the images. Next, a single rigid registration stage was performed on a $4 \times 4 \times 2$ subsampled image using the ITK versor optimizer. Finally, three stages of B-spline registration were performed using a multi-resolution method, where the image resolution increases and the B-spline grid spacing decreases at each stage. These parameters were selected empirically, based on past experience using this algorithm. Although Plastimatch supports GPU-acceleration, we used the multicore CPU implementation for this challenge.

[1] The parameter file also includes a preamble which specifies the input and output files. We have omitted the preamble for brevity.

```
[STAGE]
xform=align_center

[STAGE]
xform=rigid
optim=versor
impl=itk
metric=mse
max_its=100
res=4 4 2

[STAGE]
xform=bspline
optim=lbfgsb
impl=plastimatch
metric=mse
max_its=100
res=4 4 2
grid_spac=70 70 70

[STAGE]
max_its=70
res=2 2 1
grid_spac=40 40 40

[STAGE]
max_its=10
res=1 1 1
grid_spac=40 40 40
```

```
[STAGE]
xform=align_center

[STAGE]
xform=rigid
optim=versor
impl=itk
metric=mse
max_its=100
res=4 4 2

[STAGE]
xform=bspline
optim=lbfgsb
impl=plastimatch
metric=mse
max_its=100
res=4 4 2
grid_spac=60 60 60

[STAGE]
max_its=100
res=4 4 2
grid_spac=40 40 40

[STAGE]
max_its=50
res=2 2 1
grid_spac=20 20 20
```

Fig. 2. Parameter file used for human images (left), and sheep images (right).

4 Results

Evaluation of the performance of a deformable registration algorithm is a challenge without a unique and absolute winner so far. Depending on the application, the complete bijectivity of the deformation field, a coarse match of the structures boundaries or the precise alignment of the finer structures may or may not be requested. Evaluating a set of registration is eventually a complex task of balancing absolutely needed features with less crucial characteristics of the transformation. In the following section we present the results by means of a set of metrics that we used for internal evaluation purposes (Table 1) and of performance indices provided by the EMPIRE10 challenge organizing committee for ranking (Table 2). All the results are restricted to the lung area only.

4.1 Internal Evaluation of Results

In our internal evaluation, we first performed a visual inspection the registration results (Table 1, 2^{nd} column). An example of "very good" alignment is presented in the first row of Fig. 4, while a "fair" alignment is shown in the bottom row.

Table 1. Qualitative and quantitative results for each scan pair. In the 1^{st} column, a visual inspection categorization is presented. The 2^{nd} and 3^{rd} columns report the dilation and the running time for each scan pair respectively. Latter two columns show some statistics on the final output (i.e. RMSE at the last iteration and Dice coefficient)

Scan Pair	Visual Inspection	Invertibility	Running Time [s]	RMSE [HU]	Dice Coefficient
01	Poor	-0.28	257.8	11825.9	0.97
02	Very Good	-0.32	343.1	3737.2	0.99
03	Very Good	-0.38	84.7	5556.5	0.98
04	Fair	-0.72	51.3	5240.1	0.98
05	Good	-0.18	209.9	3386.3	0.99
06	Very Good	-0.07	111.8	4308.6	0.99
07	Good	-0.43	248.7	10971.9	0.97
08	Fair	-0.34	256.9	6353.2	0.98
09	Very Good	-0.27	250.1	4680.7	0.99
10	Fair	-0.87	41.0	6926.6	0.98
11	Poor	-0.28	246.4	7825.1	0.98
12	Very Good	-0.22	268.7	3791.9	0.99
13	Good	-0.39	63.5	6362.6	0.98
14	Fair	-0.91	272.8	14523.1	0.97
15	Very Good	-0.36	233.9	4344.2	0.99
16	Fair	-0.66	25.3	7821.5	0.97
17	Good	-0.65	51.1	4634.1	0.98
18	Fair	-0.70	237.5	12403.4	0.97
19	Very Good	-0.11	333.0	3712.3	0.99
20	Good	-0.59	280.6	12152.5	0.97

The invertibility of the transformation was computed using the dilation measure:

$$D_i = \frac{d^2 u_i}{dx^2} + \frac{d^2 u_i}{dy^2} + \frac{d^2 u_i}{dz^2} \tag{9}$$

where u_i is the i-th vector in the deformation field. We measured the minimum dilation over all images within the lung mask, which are shown in the 3^{rd} of Table 1. A dilation value below -1 indicates the non invertibility of the vector field, which we were successful in avoiding.

The running time is reported in the 4^{th} of Table 1. This time includes image reformatting, but excludes the time spent on image I/O. These results were achieved on a PC equipped with dual Intel Xeon E5540 @ 2.53GHz and 24GB of 1.6GHz RAM.

We also report the Root Mean Squared Error (RMSE) at the last iteration of the registration and the Dice Coefficient computed between the fixed and warped lung masks. This index gives an idea of the mean overlap between the two structure, although it cannot detect internal misalignments and finer errors.

4.2 EMPIRE10 Evaluation and Results

In Table 2, we report the results and ranking for Plastimatch in the EMPIRE10 challenge. The organizing committee looked in particular at the lung boundaries and fissures alignment, at the residual distance between landmarks in the two scans and at the presence of singularities in the vector field. The evaluation

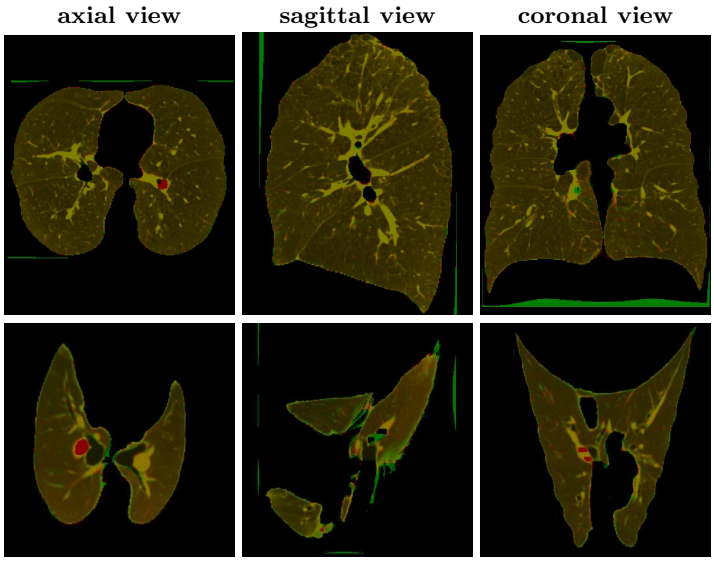

Fig. 3. Example of qualitative evaluation. In all the views, the fixed (target) scan is presented in red, while the warped (output) scan is in green. A good overlap between the two images will visually result in yellow. In the first row, an example of "very good" result is presented, while the bottom shows a "fair" result with relatively large misalignment in the images of a sheep.

metrics and the scoring procedure are presented in detail in the main workshop article by Murphy et al [3].

5 Discussion and Conclusion

We presented an efficient implementation of B-spline based deformable image registration algorithm. The improved efficiency of the registration process is reported as running time on the EMPIRE10 benchmark dataset. It takes 0.4 $\sim$ 5.7 minutes to register to a pair of 3D CT benchmark images.

Our method ranked 12 out of 34 methods that participated this challenge. The Dice coefficients and RMSE we computed show that our method provides good alignment between the registered lung CT images. However, these statistics maybe not sufficient to indicate accurate registration on local structures. As a result, visual inspection is still needed in the clinical setting.

We did not use any landmark points in the EMPIRE10 grand challenge as our algorithm is automatic. However, our method can certainly benefit from the use of the landmark points. We have already extended our method to incorporate landmark points as hard constraints to improve the registration of the local structures. We will test the extension in more benchmark datasets where landmark points are available.

Table 2. Results for each scan pair, per category and overall. Rankings and final placement are from a total of 34 competing algorithms.

	Lung Boundaries		Fissures		Landmarks		Singularities	
Scan Pair	**Score**	**Rank**	**Score**	**Rank**	**Score**	**Rank**	**Score**	**Rank**
01	0.00	15.00	0.03	7.00	2.21	10.00	0.00	11.50
02	0.00	11.00	0.00	15.00	0.53	18.00	0.00	12.50
03	0.00	27.00	0.00	12.50	0.70	27.00	0.00	12.00
04	0.00	2.50	0.00	16.50	2.33	24.00	0.00	14.00
05	0.00	13.00	0.00	16.00	0.39	26.00	0.00	13.50
06	0.00	16.00	0.00	7.00	0.45	24.00	0.00	14.00
07	0.00	11.00	0.54	8.00	1.91	8.00	0.00	10.00
08	0.00	12.00	0.00	3.50	1.19	18.00	0.00	12.50
09	0.00	15.00	0.00	13.00	0.75	23.00	0.00	13.00
10	0.00	4.00	0.00	31.00	4.65	25.00	0.00	13.50
11	0.00	10.00	0.11	17.00	1.25	15.00	0.00	11.50
12	0.00	10.00	0.00	13.50	0.46	23.00	0.00	14.50
13	0.00	8.00	0.14	24.00	1.13	19.00	0.00	13.00
14	0.04	17.00	2.23	6.00	2.05	8.00	0.00	9.50
15	0.00	20.00	0.00	18.00	0.73	18.00	0.00	12.50
16	0.00	13.00	0.11	20.00	1.51	22.00	0.00	13.50
17	0.00	6.50	0.04	9.50	1.15	22.00	0.00	14.00
18	0.01	12.00	1.41	13.00	2.18	9.00	0.00	10.50
19	0.00	14.00	0.00	12.00	0.54	17.00	0.00	14.50
20	0.00	15.00	0.70	3.00	1.97	13.00	0.00	10.50
Avg	0.00	12.60	0.26	13.27	1.40	18.45	0.00	12.52
Average Ranking Overall								14.21
Final Placement								12

References

1. Plastimatch. http://plastimatch.org.
2. C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778. L-BFGS-B: Fortran subroutines for Large-Scale bound constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997.
3. K. Murphy, B. van Ginneken, J.M. Reinhardt, S. Kabus, K. Ding, X. Deng, and J.P.W. Pluim. Evaluation of methods for pulmonary image registration: The EMPIRE10 study. In *Grand Challenges in Medical Image Analysis*, 2010.
4. E. M. van Rikxoort, B. de Hoop, M. A. Viergever, M. Prokop, and B. van Ginneken. Automatic lung segmentation from thoracic computed tomography scans using a hybrid approach with error detection. *Medical Physics*, 36(7).
5. Z. Wu, E. Rietzel, V. Boldea, D. Sarrut, and G. C. Sharp. Evaluation of deformable registration of patient lung 4DCT with subanatomical region segmentations. *Medical Physics*, 35(2).
6. G. C. Sharp, R. Li, J. Wolfgang, G. Chen, M. Peroni, M. F. Spadea, S. Mori, J. Zhang, J. Shackleford, and N. Kandasamy. Plastimatch - An open source software suite for radiotherapy image processing. In *Proceedings of the XVI'th International Conference on the use of Computers in Radiotherapy (ICCR)*, Amsterdam, Netherlands.