Appendix B

Registration Software User Guide

The different programs used for registering images in this master's thesis are introduced in this appendix. They are either AIR [43] or ITK [44] based programs written specifically for this thesis. They constitute the set of tools all the experiments are based on. For the programs that require a specific file type (AIR / ITK) as input or generate a specific type as output, it is important to remark which are ITK-based and which ones AIR-based.

Both AIR and ITK programs work with Analyze files (a *.img / .hdr* pair) but, when two images are registered, ITK results in a meta-image deformation field (a $*.mha$ file) and AIR into a $*.warp$ or $*.air$ file containing the transformation coefficients. The appropriate program must then be run to apply the deformation: *warp* for ITK and *reslice/reslicepoly* for AIR. Options marked with [ ] are optional.

B.1 List of programs

B.1.1 Preprocessing filters

These filters are thought to be applied to the images before they are registered in order to try to improve the registration results. It has already been discussed that this should not be done because tuning the parameters automatically is a very complicated task, but one may want to experiment with them anyway. All of them are implemented in ITK.

Rescale filter

```
rescale input output [max_out]
```

Rescales the input image's values so that the minimum is 0 and the maximum *max_out*. If this constant is not specified, the default value is 9999 (which suits our CT images). A value higher than 32.767 should never be used.

Canny edge detection

```
canny input output variance
```

Uses the ITK canny edge detection filter defined in the ITK guide [26]. This filter is widely used for detecting edges because it is the optimal solution satisfying the constraints of good sensitivity, localization and noise robustness. It turns the resulting binary image into a 16-bit one, rescaling in from 0/1 to 0/9999. A suitable value for our CT data set for the filter variance parameter is 2.0.

Discrete gaussian filtering

```
gauss input output variance max_kernel_width
```

Applies a gaussian filter on the input image. Blurring the image can help to remove high frequency noise. The higher the variance, the prominent the blurring. The *max_kernel_width* parameter is an integer that limits the width of the cubic filter kernel (in voxels). A typical choice is to use a value of 5 - 7. It is also important to point out that the output is scaled to 0-9999.

Gradient anisotropic diffusion

```
gad input output no_iterations time_step conductance
```

This filters implements, as described in the ITK guide, an N-dimensional version of the classic Perona-Malik anisotropic diffusion equation for scalar-valued images. This filter has the property of blurring the image while keeping the edges, although it tends to enhance them (which may be undesirable). Typical values for the parameters are: 5 iterations, a time step of 0.05, and a conductance parameter of 3. This filter scales the output to 0-9999 too.

Curvature anisotropic diffusion

```
cad input output iterations time_step conductance
use_image_spacing
```

This filter performs anisotropic diffusion on an image using a modified curvature diffusion equation (MCDE). This filter is similar to the previous one but has the advantage of not enhancing edges. Typical values for the parameters are: 5 iterations, a time step of 0.02, , and $use\_image\_spacing = 1$ (which means YES, 0 is NO). This filter also scales the output to 0-9999.

Curvature flow

```
cf input output iterations time_step
```

Another filter that performs a blurring operation while preserving edges. In this filter, areas of high curvature will diffuse faster than areas of low curvature. Hence, small jagged noise artifacts will disappear quickly. The output is scaled to 0-9999, and typical values of number of iterations and time step for our application are 10 and 0.0625.

MinMaxCurvature flow

```
mmcf input output no_iterations time_step neigh_radius
```

This filter implements a variant of the curvature flow algorithm where diffusion is turned on or off depending of the scale of the noise that one wants to remove. Typical values for the parameters are 10 iterations, a time step of 0.0625 and a neighborhood radius of just 1. The output is scaled to 0-9999.

Bilateral filter

```
bf input output domain_sigma range_sigma (typ 6 5)
```

Still one more filter for smoothing while preserving edges. It uses both domain and range neighborhoods. Pixels that are close to a pixel in the image domain and similar to a pixel in the image range are used to calculate the filtered value. Two Gaussian kernels (one in the image domain and one in the image range) are used to smooth the image. The result is an image that is smoothed in homogeneous regions yet has edges preserved. The result is similar to anisotropic diffusion but the implementation in non-iterative. Typical values the standard deviations are 6 (domain) and 5(range). The output is scaled to 0-9999.

B.1.2 Resizing filters

These filters are used to shrink and expand images. This is especially useful if one wants to use a pyramidal scheme.

Shrinking filter

`smaller input output factor`

Shrinks an scalar image with the given factor, which must be an integer.

Expanding filter

`bigger input output factor`

Expands a scalar image with the given factor (which must be an integer) using linear interpolation.

Expanding filter for vector fields

`bigger3d input output factor`

Expands a vector field with the given factor (which must be an integer). Linear interpolation is used.

B.1.3 Registration

These programs calculate the transformation parameters between two images.

AIR linear registration

`alignlinear fixed moving output_transform -m model_no [options]`

This program uses different linear transformations (depending on the model number provided) to generate a *.air* file that can be applied on the moving image afterwards. The different models are:

- 6. rigid body 6 parameter model
- 7. Global rescale 7 parameter model
- 9. Traditional 9 parameter model
- 12. Affine 12 parameter model
- 15.Perspective 15 parameter model

The rest of the options are described with high detail in the AIR manual. A typical setting is "-h 4 -r 30 -x 2", which sets the cost function to least squares, stops after 4 iterations with no improvement, and sets a total limit of 30 iterations.

AIR polynomial registration

```
alignpoly fixed moving output_transform  -m model_no [options]
```

This program uses different polynomial transformations (depending on the model number provided) to generate a *.warp* file that can be applied on the moving image afterwards. The different models are:

- 1. first order linear 12 parameter model
- 2. second order nonlinear 30 parameter model
- 3. third order nonlinear 60 parameter model
- 4. fourth order nonlinear 105 parameter model
- 5. fifth order nonlinear 168 parameter model
- 6. sixth order nonlinear 252 parameter model
- 7. seventh order nonlinear 360 parameter model
- 8. eighth order nonlinear 495 parameter model
- 9. ninth order nonlinear 660 parameter model
- 10. tenth order nonlinear 858 parameter model
- 11. eleventh order nonlinear 1092 parameter model
- 12. twelfth order nonlinear 1365 parameter model

The rest of the options are almost identical to those for the previous program, and are of course described in the AIR manual.

Demons algorithm (ITK)

```
demons1 fixed moving histo_no match_no...
           ... iterations sigma field [initial_field]
```

This program uses the "demons" deformable registration algorithm proposed at [24] in order to estimate the deformation field that turns the moving image into the reference one. As the algorithm is based on that equivalent voxels have the same value in the two images, a histogram matching operation can be performed as a preprocessing step.

An image's histogram is just a function of the number of counts against the different pixel values. In this program, *histo_no* represents the number of histogram levels (the number of histogram regions in which the pixels will be grouped) and *match_no* tells how many levels are to be matched. A way of skipping this step is to set them to 1 and 0 respectively. Otherwise, 1024 and 7 work quite well on our images.

The *iterations* parameter is usually around 50, and *sigma* is the standard deviation of the gaussian filter that smoothes the deformation field after each iteration in order to keep a certain coherence between adjacent vectors. A typical value is 1. The $field$ parameter is the name of the *.mha* output file and $inital\_field$ is the initial value for the solution in the iterative process. If not provided, a zero field is used.

Fast demons algorithm (ITK)

```
demons2 fixed moving histo_no match_no...
           ... iterations sigma field [initial_field]
```

This program works exactly in the same way as the previous one but uses a faster version of the algorithm [25]. By applying $field$ to the input image one gets the registered one. Voxels at non-grid positions are linearly interpolated, which is quite fast. The parameters are exactly the same ones, and take the same typical values. The number of iterations can be reduced, though, as the algorithm converges faster.

B.1.4 Operations

These filters are used to apply previously calculated transformations or to combine them.

Field compose (ITK)

```
compose_field field_1 field_2 output
```

Calculates the composition of two fields, that is, the equivalent one to applying $field\_1$ first and then $field\_2$. At every point P, the $v1$ vector defined by the first field is taken and added to P in order to get $P2$. Then it is possible to evaluate the value of the second field vector $v2$ at $P2$ (interpolating, if necessary) and add it to this point to find out the final point $P3$. This is where the pixel at P would end up. The equivalent field at the original point is the difference vector between $P3$ and P.

Warping (ITK)

```
warp input field output [interpolation_method]
```

Applies a deformation field to an image in order to generate the output. Voxels at non-grid positions are interpolated. Bsplines of the given order are used for this, with a maximum order of 5. Order 0 is equivalent to nearest neighbor, and order 1 (which is the default value) to linear interpolation. If higher orders are used, the first derivative is kept continuous, but it takes longer time to calculate.

Deformation field inversion (ITK)

```
inverse_field input output [no_iterations]
```

Tries to invert a deformation field, saving a registration process if one wants the fixed and moving images' roles to be exchanged. There is not any analytical solution to this problem: this program is based on a iterative process that will not lead in general to an exact solution. The default number of iterations is 64.

AIR Linear warping

```
reslice air_file output [options]
```

Applies a given transformation file ($*.air$) to generate the registered output image. Please note that the input image name is embedded in the transformation file and hence not required. The options are described in the AIR manual. A typical choice may be"-n 1" to use trilinear interpolation to find out the values for the pixels at non-grid positions, or "-n 5" for a 3D windowed *sinc* function.

AIR Polynomial warping

```
reslicepoly warp_file output. [options]
```

Applies a given transformation file ($*.warp$) to generate the registered output image. As in the previous one, the input image name is not required. The options are almost the same than in the previous program, and can be checked in the AIR manual. "-n 1" or "-n 5" are still typical choices.

AIR Linear transformation inversion

```
invert_air input output
```

This program inverts the linear transformation matrix stored in the input *.air* file and stores the result in the output file. It saves running another linear registration if one wants the fixed and moving images' roles to be exchanged.

B.2 Examples

For illustrating the use of the above described software, an example of batch file will be examined. The improved demons algorithm and pyramidal processing are used in order to register two images (the reference is $pos1.img$, the moving image is $pos2.img$) and also generate the resulting one. Pyramidal processing is a very important technique that can reduce the total computation time dramatically. It consists of registering shrunk versions of the image in order to use the result as a starting point for the full image deformation field. In this example, a three level pyramid will be used.

The first step is to calculate the shrunk versions:

```
smaller pos1.img pos1_4.img 4
smaller pos2.img pos2_4.img 4
smaller pos1.img pos1_2.img 2
smaller pos2.img pos2_2.img 2
```

One can then register the quarter-sized images:

```
demons2 pos1_4.img pos2_4.img  1 0 5 1 field_4.mha
```

The resulting field is expanded so that it corresponds to a half-sized image:

```
bigger3d field_4.mha field_2b.mha 2
```

And now it is possible to register the half-sized images using the recently calculated field as starting solution:

```
demons2 pos1_2.img pos2_2.img  1 0 3 1 field_2.mha field_2b.mha
```

The process is repeated once more in order to get the full-size solution field:

```
bigger3d field_2.mha field_b.mha 2
demons2 pos1.img pos2.img  1 0 2 1 field.mha field_b.mha
```

The field can finally be applied to the moving input to get the registered image:

```
warp pos2.img field.mha res.img
```

A last step consists of removing the temporary files:

```
del field_4.*
del field_2b.*
del field_2.*
del field_b.*
del pos1_2.*
del pos1_4.*
del pos2_2.*
del pos2_4.*
```

B.3 Image Samples

Some samples are shown in order to illustrate the effects that the different programs have on images. All the images are slices extracted from a CT 3D volume. In figure B.1 the results from applying different filters on a CT slice are shown. The typical values mentioned in the previous section have been used for every filter.

In figure B.2, the deformable registration process with the improved demons algorithm is shown. The resulting registered image is displayed after one, two, four and eight iterations. It can be appreciated how it becomes more and more similar to the reference one, being almost identical after eight iterations. It is important to remark that the "hole" that is created on the left side does not come from nowhere, but from a different slice located at a different z position.

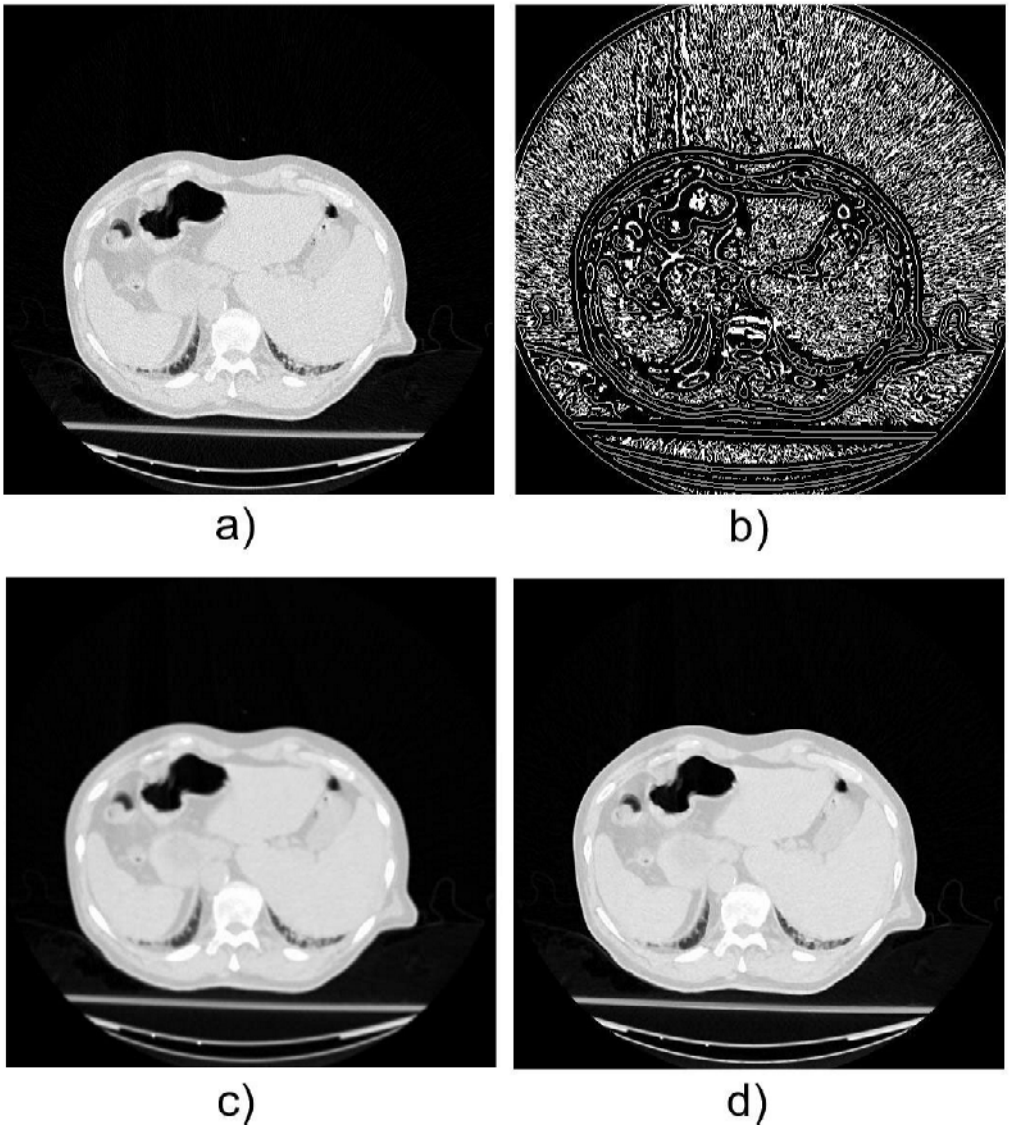

Figure B.1: a) Reference CT slice b) Edge detection filter c) Gaussian filter d) Blurring filter with edge preservation (MinMax Curvature Flow)

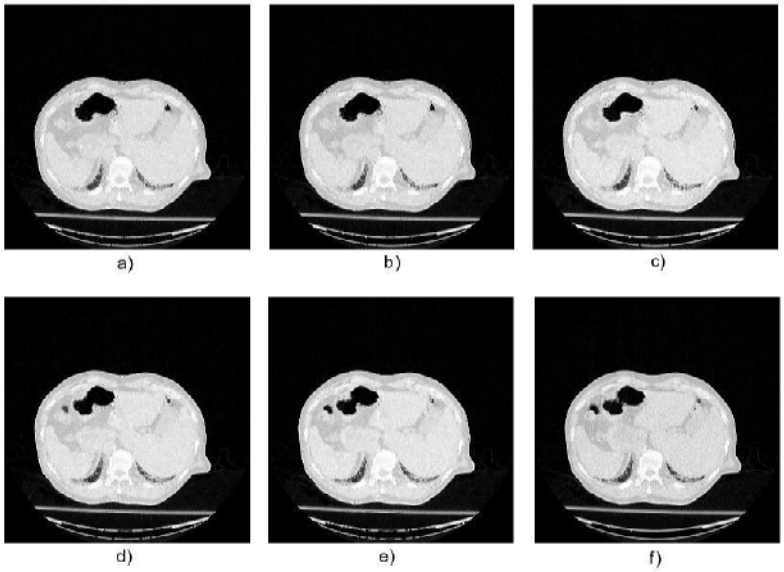

Figure B.2: Improved demons registration process a) Original moving image b) Moving image after 1 iteration c) 2 iterations d) 4 iterations e) 8 iterations f) Fixed image

Bibliography

[1] Hoyert DL; Kung H-C; Smith BL. Deaths: Preliminary data for 2003. *National Vital Statistics Report*, 53(15), February 2005.

[2] American Cancer Society. Cancer fact and figures 2005. *American Cancer Society*, 2005.

[3] American Cancer Society. http://www.cancer.org/.

[4] Krestel E. *Imaging Systems for Medical Diagnosis*. Siemens Publishing, 1990.

[5] Suetens P. *Fundamentals of Medical Imaging*. Cambridge University Press, March 2002.

[6] Radon J. Uber die bestimmung von funktionen durch ihre integralwerte langs gewisser mannigfaltigkeiten. *Ber. Verh. Sachs. Akad. Wiss. Leipzig*, 69:262–277, 1917.

[7] Macovski A. Physical problems of computerized tomography. *Procedures of the IEEE Medical Imaging Conference*, 71(3):373–378, March 1983.

[8] Kontaxakis G; Strauss LG. Maximum likelihood algorithms for image reconstruction in positron emission tomography. In Bender HF Limouris GS, Shukla SK and Biersack H-J, editors, *Radionuclides for Oncology - Current Status and Future Aspects*, pages 73–106. MEDITERRA Publishers, 1998.

[9] Hudson HM; Larkin RS. Accelerated image reconstruction using ordered subsets of projection data. *IEEE Transactions on Medical Imaging*, XX:100–108, 1994.

[10] Roux S; Desbat L; Koenig A; Grangeat P. Exact reconstruction in 2d dynamic ct: compensation of time-dependent affine deformations. *Physics in Medicine and Biology*, 49(11):2169–2182, June 2004.

[11] Udupa JK; Herman GT. *3-D Imaging in Medicine*. CRC Press, 1991.

[12] Valk PE; Bailey DL; Townsend DW; Maisey MN. *Positron Emission Tomography : Basic Science and Clinical Practice*. Springer, May 2003.

[13] Townsend DW; Beyer T; Blodgett TM. A hardware approach to image fusion. *Seminars in Nuclear Medicine*, XXXIII(3):193–204, July 2003.

[14] Rosenfeld A; Kak AC. *Digital Picture Processing, vol I and II*. Academic Press, Orlando,Fla, 1982.

[15] Karlsson J; Iglesias JE; Sommerfeld J; Vigilante N; Athari K. Project lasergame, http://www.s3.kth.se/signal/project_course/2005/blue/. This is work of group blue in the Project Course in Signal Processing and Digital Communications at KTH university, Stockholm, 2005.

[16] Zitova B; Flusser J. Image registration methods: a survey. *Image and Vision Computing*, 21(11):977–1000, 2003.

[17] Brown LG. A survey of image registration techniques. *ACM Computing Surveys*, 24(4):326–376, December 1992.

[18] Goshtasby A; Stockman GC. Point pattern matching using convex hull edges. *IEEE Transactions on Systems, Man and Cybernetics*, 15(5):631–637, 1985.

[19] Lavine D; Lambird B; Kanal L. Recognition of spatial point patterns. *Pattern Recognition*, 16(3):289–295, 1983.

[20] Ehlers M. Region-based matching for image registration in remote sensing databases. *Proceedings of the International Geoscience and Remote Sensing Symposium IGARSS'91, Espoo, Finland*, pages 2231–2234, 1991.

[21] Mohr R; Pavlidis T; Sanfeliu A. *Structural Pattern Analysis*. World Scientific, Teaneck, NJ, 1990.

[22] Bunke H; Sanfeliu A. *Syntactic and Structural Pattern Recognition, Theory and Applications*. World Scientific, Teaneck, NJ, 1990.

[23] Whaba G. *Spline models for observational data*. Society for Industrial and Applied Mathematics, 1990.

[24] Thirion JP. Image matching as a diffusion process: an analogy with maxwell's demons. *Medical Image Analysis*, 2(3):243–260, September 1998.

[25] Wang H; Dong L; O'Daniel J; Mohan R; Garden AS; Ang KK; Kuban DA; Bonnen M; Chang JY; Cheung R. Validation of an accelerated demons algorithm for deformable image registration in radiation therapy. *Physics in Medicine and Biology*, 50(12):2887–2905, June 2005.

[26] Ibanez L; Schroeder W; Ng L; Cates J. The itk software guide second edition, draft version, http://www.itk.org/itksoftwareguide.pdf. This is the documentation of the Insight Segmentation and Registration Toolkit, 2005.

[27] Desmedt P; Hill DLG; Hwakes DJ Penney GP; Weese J; Little JA. A comparison of similarity measures for use in 2-d-3-d medical image registration. *IEEE transactions on medical imaging*, 17(4):586–595, August 1998.

[28] Wolthaus JWH; van Herk M; Muller SH; Belderbos JSA; Lebesque JV; de Bois JA; Rossi MMG; Damen EMF. Fusion of respiration-correlated pet and ct scans: correlated lung tumour motion in anatomical and functional scans. *Physics in Medicine and Biology*, 50(7):1569–1583, April 2005.

[29] Tada T; Minakuchi K; Fujioka T; Sakurai M; Koda M; Kawase I; Nakajima T; Nishioka M; Tonai T; Kozuka T. Lung cancer: intermittent irradiation synchronized with respiratory motion - results of a pilot study. *Radiology*, 207(3):779–783, June 1998.

[30] Nehmeh SA; Erdi YE; Ling CC; Rosenzweig KE; Squire OD; Braban LE; Ford E; Sidhu K; Mageras GS; Larson SM; Humm JL. Effect of respiratory gating on reducing lung motion artifacts in pet imaging of lung cancer. *Medical Physics*, 29(3):366–371, March 2002.

[31] Boucher L; Rodrigue S; Lecomte R; Benard F. Respiratory gating for 3-dimensional pet of the thorax: Feasibility and initial results. *Journal of Nuclear Medicine*, 45(2):214–219, February 2004.

[32] UCLA Department of medicine. http://www.med.ucla.edu.

[33] Livieratos L; Stegger L; Bloomfield PM; Schafers K; Bailey DL; Camici PG. Rigid-body transformation of list-mode projection data for respiratory motion correction in cardiac pet. *Physics in Medicine and Biology*, 50(14):3313–3322, July 2005.

[34] van Herk M; Lebesque JV; Miyasaka K Seppenwoolde Y; Shirato H; Kitamura K; Shimizu S. Precise and real-time measurement of 3d tumor

motion in lung due to breathing and heartbeat, measured during radiotherapy. *International Journal of Radiation Oncology Biology Physics*, 53(4):822–834, July 2002.

[35] George R; Vedam SS; Chung TD; Ramakrishnan V; Keall PJ. The application of the sinusoidal model to lung cancer patient respiratory motion. *Medical Physics*, 32(9):2850–2861, September 2005.

[36] Low DA; Nystrom M; Kalinin E; Parikh P; Dempsey JF; Bradley JD; Mutic S; Wahab SH; Islam T; Christensen G; Politte DG; Whiting BR. A method for the reconstruction of four-dimensional synchronized ct scans acquired during free breathing. *Medical Physics*, 30(6):1254–1263, June 2003.

[37] Radiology Info. Radiation exposure in x-ray examinations. *Radiology Info*, January 2005.

[38] Nehmeh SA; Erdi YE; Pan T; Yorke E; Mageras GS; Rosenzweig KE; Schoder H; Mostafavi H; Squire O; Pevsner A; Larson SM; Humm JL. Quantitation of respiratory motion during 4d-pet/ct acquisition. *Medical Physics*, 31(6):1333–1338, June 2004.

[39] Pan T; Lee T-Y; Rietzel E; Chen GTY. 4d-ct imaging of a volume influenced by respiratory motion on multi-slice ct. *Medical Physics*, 31(2):333–340, January 2004.

[40] Segars WP. *Development of a new dynamic NURBS-based cardiac-torso (NCAT) phantom*. PhD thesis, The University of North Carolina, May 2001.

[41] Rueckert D; Sonoda LI; Hayes C; Hill DLG; Leach MO; Hawkes DJ. Nonrigid registration using free-form deformations: application to breast mr images. *IEEE transactions on medical imaging*, 18(8):712–721, August 1999.

[42] Riedel M; Navab N; Moller A. Respiratory motion estimation: tests and comparison of different sensors. SEP or IDP Project together with the Nuclear Medicine Department, Klinikum Rechts der Isar der TU Mnchen, 2005.

[43] UCLA Department of medicine. Automated image registration, http://bishopw.loni.ucla.edu/air5/.

[44] Insight Software Consortium. Insight segmentation and registration toolkit, http://www.itk.org/.

[45] Trolltech. http://www.trolltech.com/products/qt/index.html.

[46] GNU: GNU's Not UNIX. http://www.gnu.org/copyleft/gpl.html.

[47] Minimalist GNU for Windows. http://www.mingw.org.