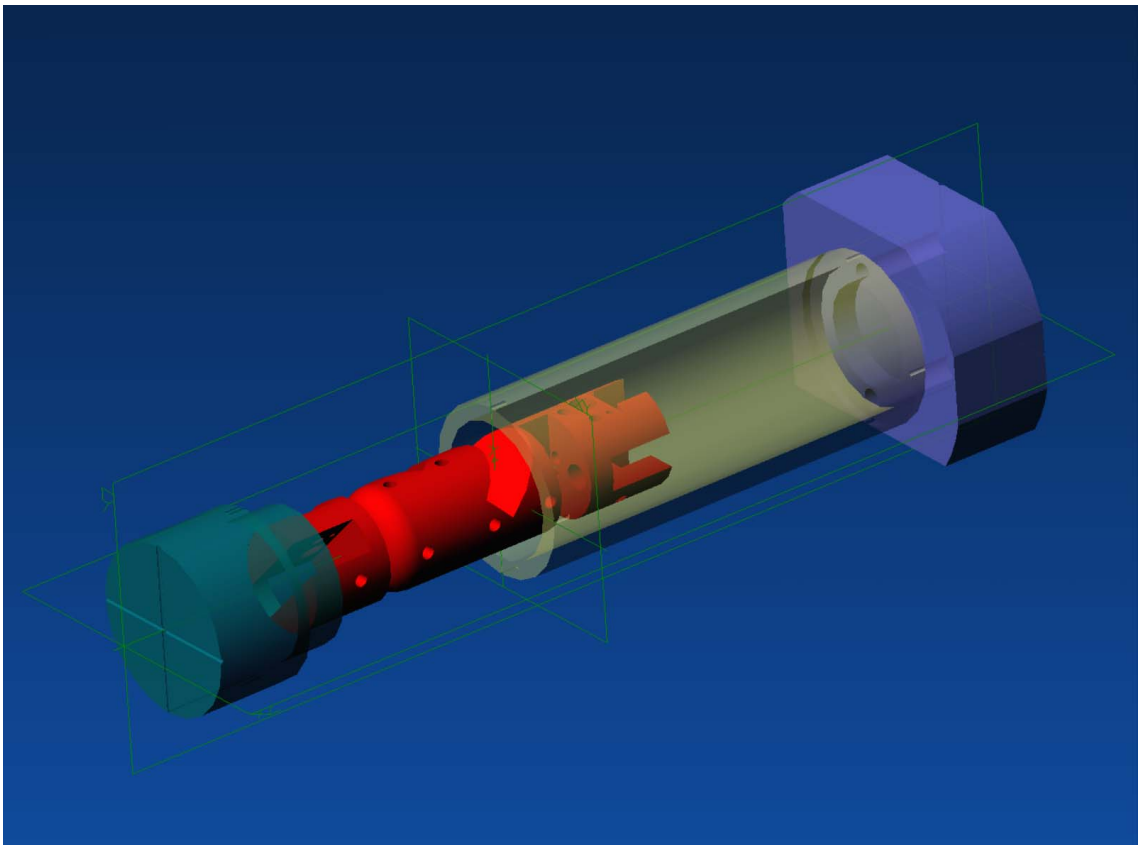
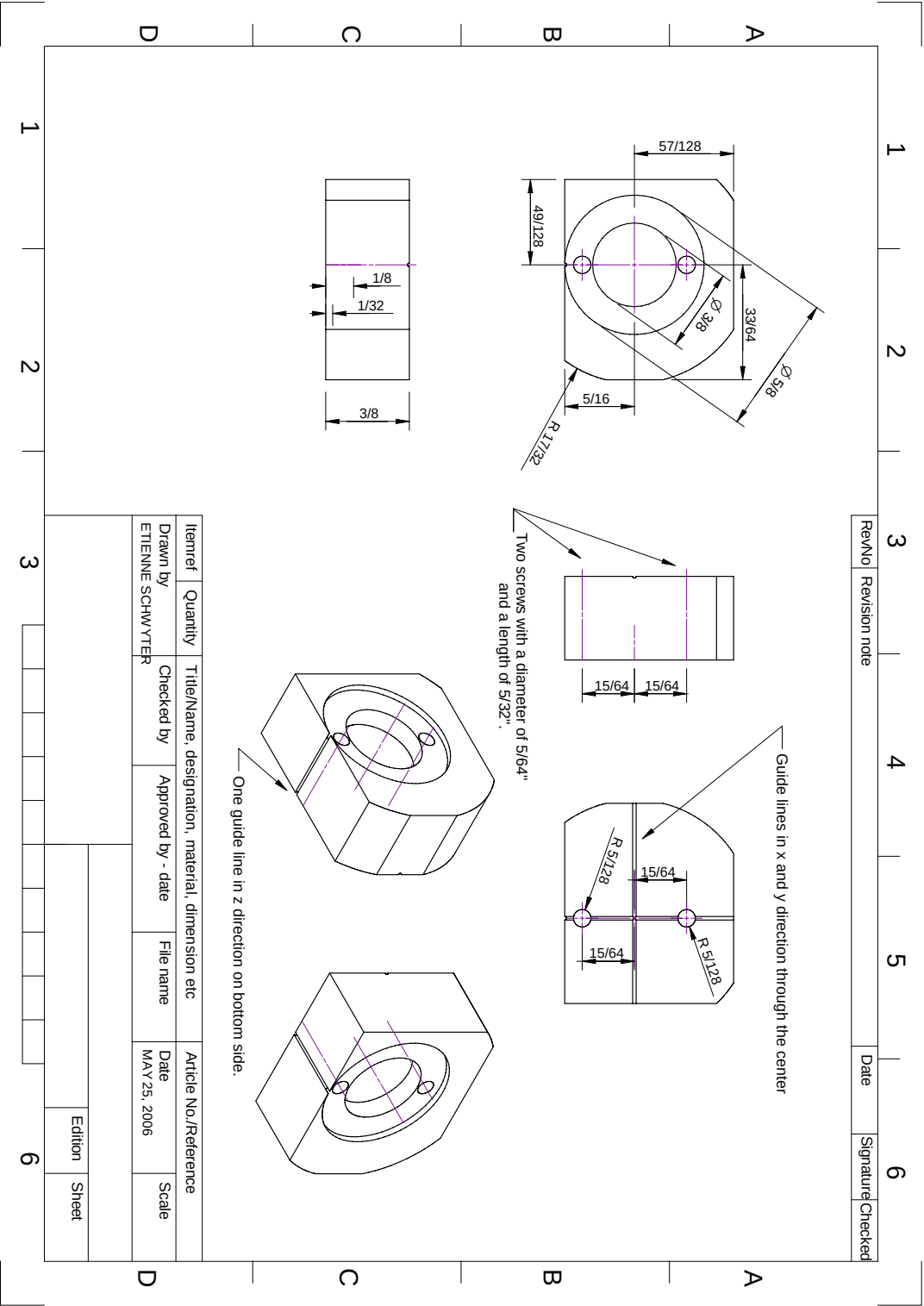


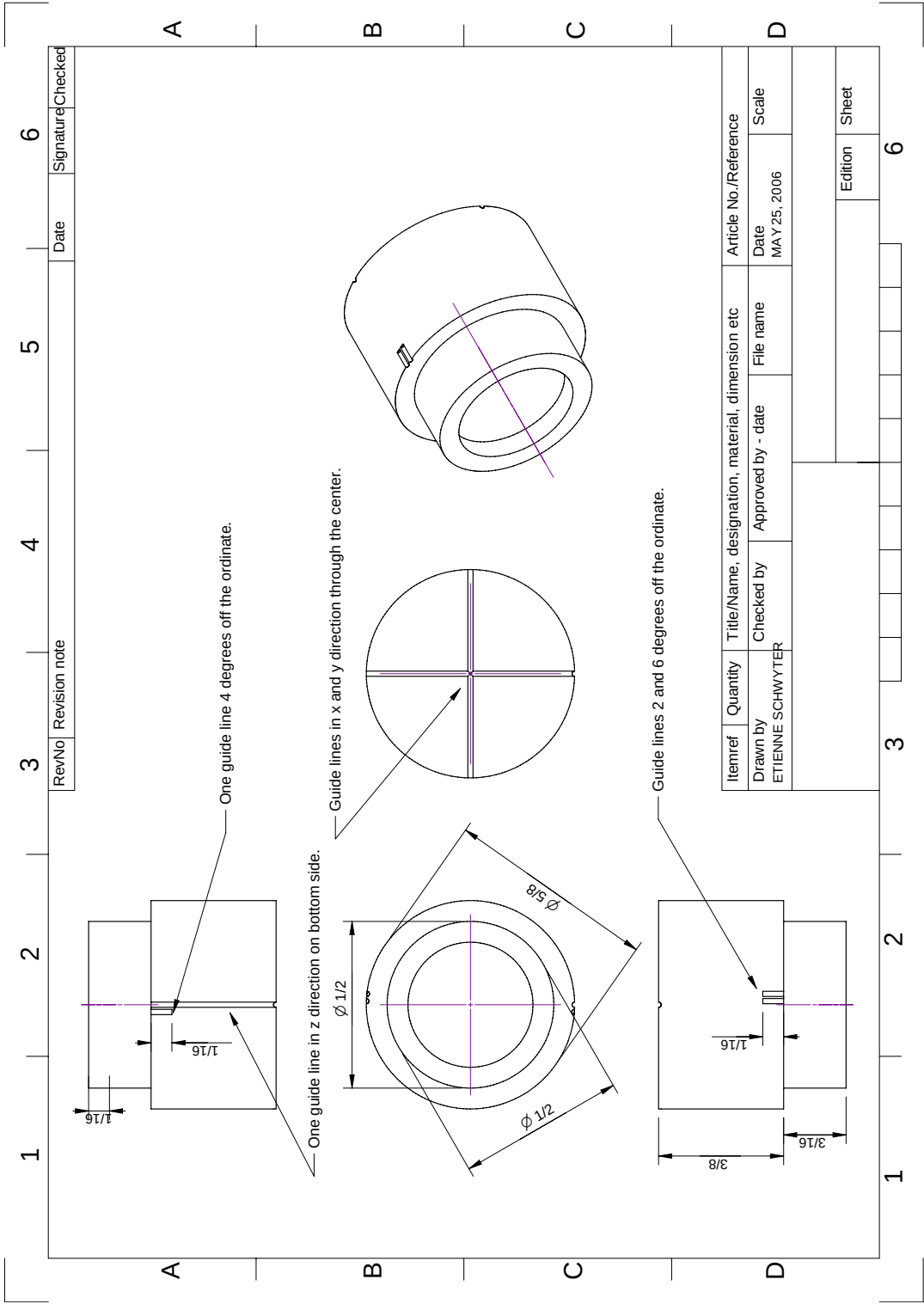
Appendix

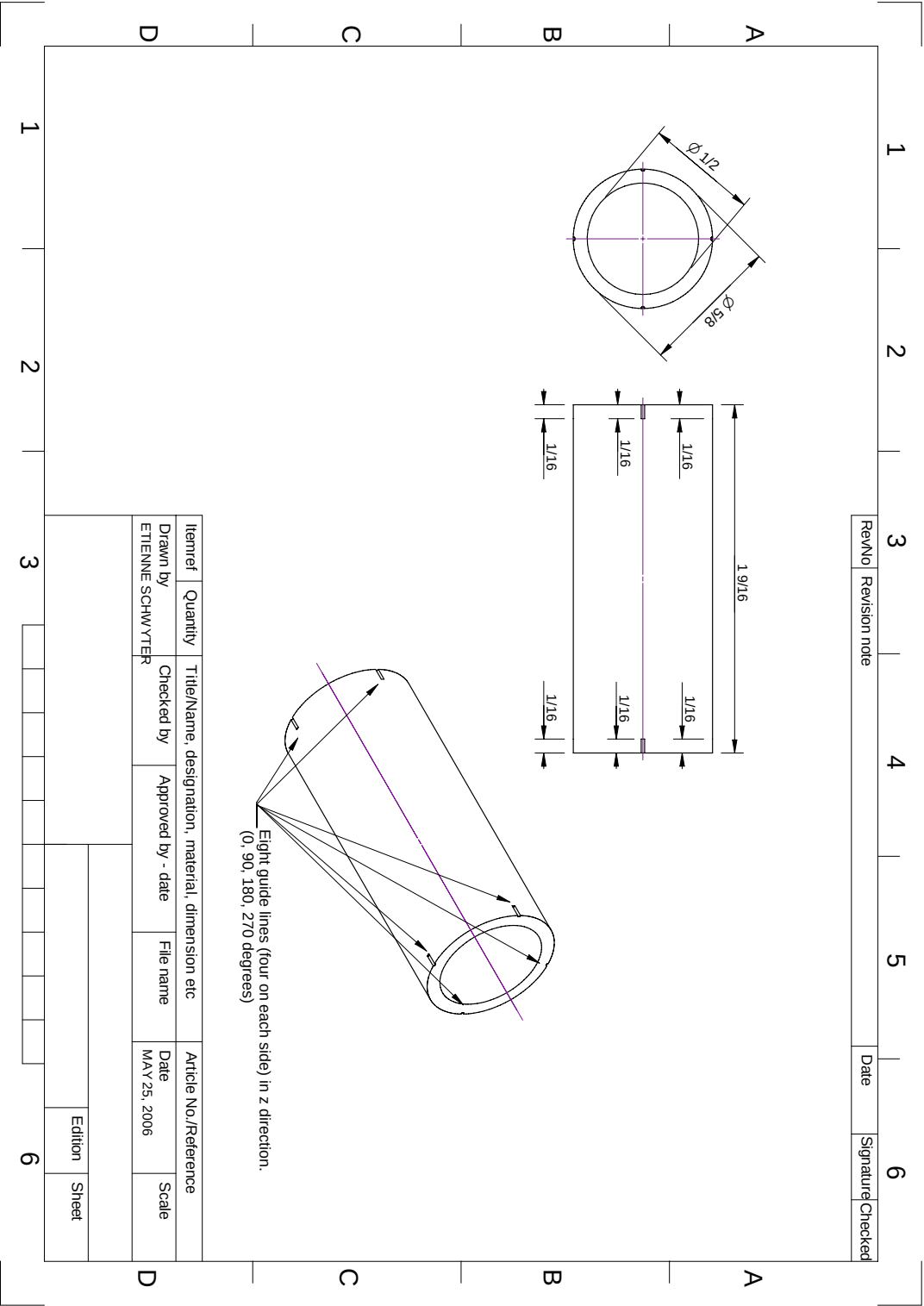
Appendix A

Phantom Drawings









Appendix B

Data Acquisition Settings

B.1 CT Scanner

Model:	vivaCT40
Manufacturer:	Scanco Medical AG, Switzerland
Energy [kVp]:	55
Intensity [μ A]:	109
Resolution:	Medium
No. Samples:	2048
No. Projections per 180:	511
Mode:	Conebeam
Diameter [μ m]:	25600
Increment [μ m]:	25
Integration Time [μ s]:	296000
Averages:	1
Calibration:	55kVp, 200mg HA wedge, 4096 scaling

Table B.1: CT scanner settings for all the acquired datasets.

B.2 MR Scanner

Model:	9.4T 21cm bore MR system
Manufacturer:	Varian Inc., USA
Matrix size:	256x256x256
Mode:	3D Gradient Echo
Receiver:	35mm quadrature coil
Averages:	1-2
TR:	50ms
TE:	5ms
Scan time:	1h 4min

Table B.2: MR scanner settings for all the acquired datasets.

B.3 Laser Scanner

Model:	SLS 2000
Manufacturer:	LMI Selcom Inc., Sweden
Resolution:	0.0025% of measurement range
Inaccuracy:	0.2% of measurement range

Settings for small drilling hole:

Mode:	Plane, X-profile
Waiting time:	0ms
Scan speed:	0.5mm/s
Scan range X direction:	1mm
Scan spacing X direction:	1um
Scan range Y direction:	0.5mm
Scan spacing Y direction:	1um
Scan time:	35min 50sec

Settings for trench:

Mode:	Plane, Y-profile
Waiting time:	0ms
Scan speed:	0.5mm/s
Scan range X direction:	2mm
Scan spacing X direction:	20um
Scan range Y direction:	4mm
Scan spacing Y direction:	1um
Scan time:	33min 50sec

Table B.3: Laser scanner settings for all the acquired datasets.

Appendix C

Thickness Measurement Settings

C.1 Pre-Filter Settings

Filter1:	Erode with a kernel radius of 1
Filter2:	Dilate with a kernel radius of 1
Filter3:	Dilate with a kernel radius of 1
Filter4:	Erode with a kernel radius of 1

Table C.1: Pre-filter settings for the subsequent thickness measurement.

C.2 SiteSpecificThickness Measurement Settings

Settings for small drilling hole:

Threshold [-t]:	4500 (CT), 6000 (MRI)
Gaussian filter [-f]:	0.0
Boundary tolerance [-b]:	0.0
Initial offset [-o]:	0.3
Lookup table [-r]:	0.4-0.8
Histogram [-h]:	0.0-1.0

Settings for trench:

Threshold [-t]:	4500 (CT), 6000 (MRI)
Gaussian filter [-f]:	0.0
Boundary tolerance [-b]:	0.0
Initial offset [-o]:	0.3 (0.1)
Lookup table [-r]:	3.0-3.5
Histogram [-h]:	3.0-3.5

Table C.2: SiteSpecificThickness measurement settings for drilling hole and trench.

Appendix D

Thickness Measurement Results

D.1 Phantom Laser Scan

Measurement	# Points	Mean	Std. Dev.
DrillingHole1:	700	661.6	14.3
DrillingHole2:	760	657.8	13.1
DrillingHole3:	800	666.2	13.3
Trench1:	200	3271.0	35.2
Trench2:	200	3274.7	43.8
Trench3:	200	3265.2	44.3

Table D.1: Phantom Laser scan results of the individual measurements.

D.2 Phantom CT Scan

Measurement	# Points	Mean	Std. Dev.
DrillingHole1:	7034	617	53
DrillingHole2:	7081	629	51
DrillingHole3:	6999	632	51
Trench1:	4531	3239	103
Trench2:	4761	3253	57
Trench3:	4684	3249	56

Table D.2: Phantom CT scan results of the individual measurements.

D.3 Phantom MRI Scan

Measurement	# Points	Mean	Std. Dev.
DrillingHole1:	643	541	61
DrillingHole2:	641	560	67
DrillingHole3:	562	553	60
Trench1:	474	3240	54
Trench2:	539	3200	44
Trench3:	565	3208	41

Table D.3: Phantom MRI scan results of the individual measurements.

D.4 Acquisition Accuracy and Precision Results

Measurement	Mean	Std. Dev.	Accuracy	Precision
DrillingHole (Laser):	661.9	4.2		
Trench (Laser):	3270.3	4.8		
DrillingHole (CT):	626.0	7.9	35.9	7.9
Trench (CT):	3247.0	7.2	23.3	7.2
DrillingHole (MRI):	551.3	9.6	110.5	9.6
Trench (MRI):	3216.0	21.2	54.3	21.2

Table D.4: Phantom thickness measurement accuracy and precision results.

Appendix E

Registration Results

E.1 Fiducial Marker-Based Registration

Registration	Metric Value Diff.	Trafo Mean Sq. Diff.
MRI_01_C0002504_FM:	0.011672	0.000012
MRI_02_C0002504_FM:	0.028846	0.000120
MRI_03_C0002504_FM:	0.022050	0.000084

Metric:

Registration	Accuracy	Precision
MRI_C0002504_FM (all):	0.020856	0.002211

Transformation Parameter:

Registration	Accuracy	Precision
MRI_C0002504_FM (all):	0.000072	0.000006

Table E.1: Fiducial marker-based registration accuracy and precision results with the best intensity-based registration as a reference (goldstandard).

E.2 Intensity-Based Registration

Registration	Metric Value Diff.	Trafo Mean Sq. Diff.
MRL02_C0002504_OFFSET8:	0.015363	0.000309
MRL02_C0002504_OFFSET8_SET1:	0.000959	0.000238
MRL02_C0002504_OFFSET8_SET2:	0.003147	0.000216
MRL02_C0002504_OFFSET8_SET3:	0.001660	0.000477
MRL02_C0002504_OFFSET8_SET4:	0.014907	0.000312
MRL02_C0002504_OFFSET8_SET5:	0.000064	0.000218

Metric:

Registration	Accuracy	Precision
MRL02_C0002504_INT (all):	0.006017	0.007136

Transformation Parameter:

Registration	Accuracy	Precision
MRL02_C0002504_INT (all):	0.000295	0.000099

Table E.2: Intensity-based (Mattes Mutual Information) registration accuracy and precision results with the best intensity-based registration as a reference (goldstandard).

E.3 Intensity-Based Registration with Offset Variation

Registration	Metric Value Diff.	Trafo Mean Sq. Diff.
MRL02_C0002504_OFFSET2:	0.004836	0.000029
MRL02_C0002504_OFFSET4:	0.007981	0.000097
MRL02_C0002504_OFFSET8:	0.015363	0.000309
MRL02_C0002504_OFFSET16:	0.014055	0.000925
MRL02_C0002504_OFFSET32:	0.399062	0.006979

Table E.3: Intensity-based registration with an increasing initial guess offset ranging from 0.002 to 0.032.

E.4 Intra-Modality Registration Results

CT/CT registration:

Registration	Metric Value Diff.	Trafo Mean Sq. Diff.
C0002669_C0002670:	0.010293	0.000001
C0002669_C0002671:	0.000707	0.000001
C0002670_C0002671:	0.007714	0.000002

MRI/MRI registration:

Registration	Metric Value Diff.	Trafo Mean Sq. Diff.
PHANTOM_MRI_01_02:	0.128019	0.000058
PHANTOM_MRI_01_03:	0.008563	0.000000
PHANTOM_MRI_02_03:	0.041690	0.000039

Table E.4: Intra-modality registration difference between the fiducial marker-based and intensity-based registration method.

Appendix F

Segmentation Results

F.1 Segmentation Accuracy and Precision Results

Denoise Filter	Feature Filter	Iterations	Voxel Diff.	Voxel Diff. %
Gauss	Gradient	1482	721	1.035
Gauss	Zero	2259	9461	13.587
Gauss	Canny	1749	3965	5.694
Gauss	Threshold	228	2	0.003
GAD	Gradient	1506	482	0.692
GAD	Zero	1277	5274	7.574
GAD	Canny	1497	4273	6.137
GAD	Threshold	270	123	0.177
ErodeDilate	Gradient	1495	1990	2.858
ErodeDilate	Zero	*	*	*
ErodeDilate	Canny	1712	8281	11.893
ErodeDilate	Threshold	220	307	0.441
MinMax	Gradient	1601	1002	1.439
MinMax	Zero	*	*	*
MinMax	Canny	2132	3960	5.687
MinMax	Threshold	208	498	0.715

Table F.1: Segmentation results from all the possible filter combinations with fixed segmentation parameters (Balloon Force: 1.5, Curvature Force: 0.1, Advection Force: 2.0). *) Unsuccessful segmentation.

F.2 Segmentation Results with Decreasing CNR

Denoise Filter	Feature Filter	SNR/CNR	Voxel Diff.	Voxel Diff. %
Gauss	Gradient	2.0/4.0	728	1.045
Gauss	Canny	2.0/4.0	3543	5.088
Gauss	Threshold	2.0/4.0	103	0.148
GAD	Gradient	2.0/4.0	406	0.583
GAD	Canny	2.0/4.0	2488	3.573
GAD	Threshold	2.0/4.0	16	0.023
Gauss	Gradient	2.0/2.0	739	1.035
Gauss	Canny	2.0/2.0	1749	3.965
Gauss	Threshold	2.0/2.0	2	0.003
GAD	Gradient	2.0/2.0	482	0.692
GAD	Canny	2.0/2.0	4273	6.137
GAD	Threshold	2.0/2.0	123	0.177
Gauss	Gradient	2.0/1.0	827	1.188
Gauss	Canny	2.0/1.0	4021	5.775
Gauss	Threshold	2.0/1.0	1906	2.737
GAD	Gradient	2.0/1.0	1119	1.607
GAD	Canny	2.0/1.0	4409	6.332
GAD	Threshold	2.0/1.0	521	0.748
Gauss	Gradient	2.0/0.5	2764	3.969
Gauss	Canny	2.0/0.5	4925	7.073
Gauss	Threshold	2.0/0.5	3068	4.406
GAD	Gradient	2.0/0.5	2296	3.297
GAD	Canny	2.0/0.5	4413	6.338
GAD	Threshold	2.0/0.5	2680	3.849

Table F.2: Segmentation results with the best six filter combinations with fixed segmentation parameters (Balloon Force: 1.5, Curvature Force: 0.1, Advection Force: 2.0) and a decreasing contrast-to-noise-ratio (CNR).

F.3 Segmentation Results with Increasing Number of Seed Points

Denoise Filter	Feature Filter	Seed Points	Iterations	Voxel Diff.
Gauss	Gradient	1	4632	721
Gauss	Gradient	3	1482	721
Gauss	Gradient	9	739	721
Gauss	Threshold	1	476	2
Gauss	Threshold	3	228	2
Gauss	Threshold	9	146	2

Table F.3: Segmentation results and number of iterations dependant on the number of initial seed points.

F.4 Segmentation Results with Changing Segmentation Parameters

Segmentation Parameter			Iterations	Voxel Diff.	Voxel Diff. %
Balloon	Curvature	Advection			
0.0	0.0	5.0	*	*	*
3.0	0.0	0.0	*	*	*
3.0	1.0	0.0	*	*	*
1.5	1.0	0.0	*	*	*
1.5	0.5	5.0	2295	2040	2.930
1.5	0.0	5.0	1975	1707	2.451
1.5	0.5	2.5	1180	2434	3.496
3.0	1.0	5.0	1451	1495	2.147

Table F.4: Segmentation results of a Gauss/Gradient filtered image with changing segmentation parameters. *) Unsuccessful segmentation.

Appendix G

Segmented Rat Knee Joint

Label	Tissue Part
1000	Muscular Belly
2000	Soleus Muscle
3000	Cranial Tibia Muscle
4000	Lateral Great Muscle Left
5000	Lateral Great Muscle Right
6000	Tensor Fascia Lata
7000	Sesamoid Bone Left
8000	Sesamoid Bone Right
9000	Meniscus Left
10000	Meniscus Right
11000	Front Bone Remanent
12000	Patella
13000	Cruciate Ligament 2
14000	Cruciate Ligament 1
15000	Fibula Growth Plate
16000	Fibula
17000	Bone Marrow Tibia
18000	Tibia Growth Plate
19000	Tibia
20000	Cartilage Femur
21000	Bone Marrow Femur
22000	Femur Growth Plate
23000	Femur

Table G.1: Segmented and relabeled rat knee joint tissue parts.

Appendix H

Libraries and Programming Languages

H.1 Visualization Toolkit (VTK)

VTK [41, 46] is an open source package maintained by Kitware [20], Inc. It consists of a set of C++ class libraries for 3D computer graphics, image processing and visualization. VTK supports a wide variety of visualization algorithms including scalar, vector, texture and volumetric methods. Many functions have been implemented such as polygon reduction, mesh smoothing, image translation and slicing. The toolkit is platform independent.

VTK 5.0.0 was used to develop most of the applications and classes listed below.

H.2 Insight Toolkit (ITK)

ITK [18, 49] is an open source, platform independent, C++ segmentation and registration library, founded by the National Library of Medicine. The toolkit contains numerous image manipulation functions such as image filters, registration algorithms and segmentation approaches.

Within this project ITK 2.6.0 was used in various implemented programs, described below.

H.3 Fast Light Toolkit (FLTK)

FLTK [12] is a cross-platform C++ graphical user interface (GUI) toolkit. It provides modern GUI functionalities and is designed to be statically linked (memory friendly).

FLUID, an FLTK internal program, is simplifying the creation of user interfaces by providing the option to add or change interface objects directly in the window, instead of changing them in the source code.

This library was used to create and change the graphical user interfaces in mySNAP.

H.4 CMake

CMake [20], developed by Kitware, is used to control the software compilation process, using simple platform and compiler independent configuration files. The generated make-files link the libraries used in the compiler environment.

CMake was used to link VTK, ITK, FLTK and other libraries for all the implemented applications.

H.5 Microsoft Visual C++

Microsoft Visual C++ 6.0 [29] provides a powerful and flexible development environment for creating Microsoft Windows based applications. Visual C++ is the main code writing environment, and compiles the written programs to obtain executables (exe-files).

All the implemented programs and classes are Microsoft Visual C++ based.

Appendix I

Applications and Classes

I.1 Implemented

I.1.1 DrillingHoleRawDataFilter

This Matlab program reads in the raw data from the phantom laser scan (saved as a data file (*.dat)), and finds the minimum values in a user defined range close to the expected drilling hole edges.

The function takes the file name of the raw data, as well as the names of the output Matlab files (*.mat) as an input. The two sets of evaluated minima are stored in separate output files (Output1.mat and Output2.mat). For a visual control, the result is displayed in a 2D graph, showing all the minima found in the defined range.

The data is now ready to be used by the DrillingHoleCircleFitting Matlab program described in [I.1.3](#). Due to the rather big raw data files (32MB), the processing time might take up to 20 minutes on a standard 3.4 GHz Pentium 4 workstation.

Usage: DrillingHoleRawDataFilter Input.dat Output1.mat Output2.mat

I.1.2 TrenchRawDataFilter

The TrenchRawDataFilter program does the same as the DrillingHoleRawDataFilter, with the only difference of finding the maxima instead of the minima in a user defined range.

Usage: TrenchRawDataFilter Input.dat Output1.mat Output2.mat

I.1.3 DrillingHoleCircleFitting

This function is fitting a circle to the previously filtered data set. The algorithm (described in subchapter 4.3.4.2) is a least square approach and minimizes the distance between the data points and the fitted circle.

The input parameters of this function are the two Matlab files with the minima, a statistics Matlab file, where all the parameters and results will be listed, an initial guess for the centroid and the radius, as well as a tolerance for the step length and the gradient.

Beside the centroid, the radius and the corresponding standard variation, many other values, like the convergence criteria, the covariance matrix of the circle center and the uncertainty in the circle radius, are saved in the statistics file. The data points with the fitted circle are displayed in a 2D graph.

Usage: DrillingHoleCircleFitting Input1.mat Input2.mat Output.mat ...
 ... [CentroidInitialGuess][RadiusInitialGuess][ToleranceStepLength][ToleranceGradient]
Example: DrillingHoleCircleFitting Input1.mat Input2.mat Output.mat ...
 ... [320, 340] 300 1 1

I.1.4 TrenchLineFitting

Instead of the circle, two lines are fitted to the maxima values obtained from the TrenchRawDataFilter function. The algorithm is therefore slightly changed (see subchapter 4.3.4.2).

Due to the lower complexity of the algorithm, the function only needs the two sets of maxima and the file name for the statistics as an input. No initial guess is needed.

If the scan axis is not absolutely perpendicular to the trench walls, the thickness will be overestimated. Therefore, the thickness is corrected by the angle induced error.

TrenchLineFitting saves the mean thickness with and without angle correction, the thickness standard deviation, the line angle and angle difference, as well as the residuals and the norm of the residuals. The data points with the fitted lines are displayed in a 2D graph.

Usage: TrenchLineFitting Input1.mat Input2.mat Output.mat

I.1.5 AppAIMDenoiseFilter

In this application an open and close filter, was implemented. Both filters are based on the erosion and dilate basic filters. An erosion filter checks all the neighborhood

pixels in the defined kernel radius and sets the center value to the lowest intensity of all the voxels taken into concern. The dilate filter oppositely sets the center value to the highest intensity in the kernel range.

The open filter applies first an erosion followed by a dilation. In case of the closing filter, the filters are swapped. If open and close filters are combined, structures smaller than the kernel size will be eliminated and small gaps will be filled.

This filter is ideally suited to denoise the CT and MR images for the accuracy and precision thickness measurements, with as little information loss and changes as possible. Noise free images are absolutely crucial to obtain accurate thickness measurements. For more details see subchapter [4.3.4.1](#).

The application has a manual and an automatic mode. In the manual mode, the program guides the user through all the filtering steps and displays each filter result for a visual control. For the automatic mode, all the filters that want to be applied have to be added in the command line after the input and output file name (see below).

Usage: AppAIMDenoiseFilter Input.aim Output.aim [Arguments]

Example: AppAIMDenoiseFilter Input.aim Output.aim t 5500 e 1 d 1 d 1 e 1 s

The filter thresholds the input image at an intensity level of 5500, applies the erode and dilate filters with a kernel radius of 1, and finally saves the denoised image.

Arguments:

g [arg] : Gaussian filter with a user defined sigma
 t [arg] : Threshold filter at the [arg] level
 e [arg] : Erode filter with a kernel radius of [arg] voxel
 d [arg] : Dilate filter with a kernel radius of [arg] voxel
 s : Saves the denoised image
 q : Quits the program without saving the image

I.1.6 itkSimilarityRegistration

The original multi-resolution registration program including a scaling factor, was implemented by J. Owoc [\[33\]](#).

After running several test samples, several weak points of the software had to be corrected. The list below shows all the necessary changes and extensions to make the program compatible for multi-modal registrations.

- **Change Optimizer:** Based on the findings of Moser and Kuhn [\[31\]](#), their modified gradient descent optimizer performs best for various registration experiments. Though, for multi-modal image registrations, their algorithm often diverged or got trapped in local minima. Therefore, the optimizer was replaced

by the close related regular step gradient descent optimizer, which performed well for all the example data sets.

- **Implemented the normalized Mattes mutual information metric:** According to diverse papers, this metric is supposed to return as good or even better registrations results than the mattes mutual information metric. A comprehensive description of the algorithm is given in chapter 7.
- **Extended transformation properties:** So far the registered images were resampled and cropped to the size of the fixed image. This might be satisfying for most intra-modality registrations. In the particular case of multi-modality registration, the image- and voxel-sizes usually are significantly different. CT images focus on bone structures, MR images show good contrast for soft tissues. Therefore, the two imaging methods have completely different volumes of interest.
Now, the user has the option to preserve all the image information, even if the input image is translated and rotated. The transformed image edge points are taken to create a bounding box with the appropriate size (see figure I.1). Furthermore, the image is resampled with the original moving image voxel-size. This minimizes the information loss caused by image translation.
- **Change config file parser:** The AppSimilarityRegistration program takes a config file as an input. In this file, all the registration parameters are defined. Due to the extensions and changes in the application, the config file had to be updated. In the new version, each parameter is shortly described and all the options are listed.
A parser function extracts the parameters from the config file and passes them on to the registration program.
- **Extension and change of logfile:** The new optimizer returns different values than the previous one and therefore the logfile had to be adjusted. All the registration information can optionally be saved in the output AIM log, an external text file or as a logfile extension file which can easily be imported in Matlab or Excel.
- **ITK class:** The original similarity registration application was transformed into an ITK class by Dr. Steven Boyd. The registration program can now easily be included in any other ITK or VTK program.

This class is also available as a ready-to-use application (see "AppSimilarityRegistration.exe" on the enclosed DVD).

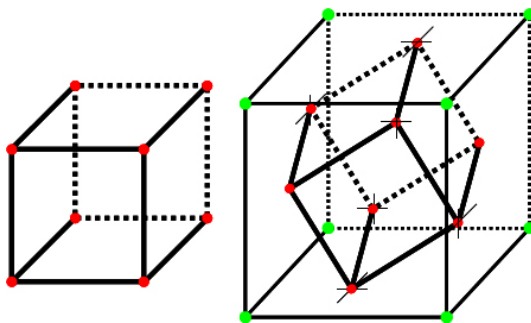


Figure I.1: In the largest region approach, the 8 image corners (red) are transformed and form the new extent (green) of the image.

Example:

```
vtkFileParser *cfile = vtkFileParser::New();
cfile->SetFileName(configFileName);
cfile->SetCommentCharacterModeToPound();

itkSimilarityRegistration* reg = itkSimilarityRegistration::New();
reg->SetFileParser(cfile);
reg->Execute();
```

I.1.7 itkSimilarityRegistrationFloatingPoint

The Viola-Wells metric only runs with floating point data. The change from short (16bit word type) to float (32bit word type) implicates a duplication of the memory requirements for the registration. Therefore, the metric was debugged and implemented in a separate registration program to compare the performances of the different metrics for multi-modal registrations. More information on this topic is provided in subchapter [7.1.1.3](#).

This class can be included exactly the same way as `itkSimilarityRegistration` and is also available as a ready-to-use application (see "AppSimilarityRegistrationFloatingPoint.exe" on the enclosed DVD).

I.1.8 AppFiducialMarkerBasedRegistration

The fiducial marker based registration is, due to its high accuracy and precision and its very low computation time, a good alternative to the intensity based registration method.

The application is running two sub-filter classes to calculate the centroids of the fiducial markers and to evaluate the transformation parameters. The program reads

all the parameters necessary for the registration from a config file and writes the seven transformation parameters optionally to a txt-file.

Usage: AppFiducialMarkerBasedRegistration -config [ConfigFile]

I.1.9 vtkIntensityWeightedCentroid

This class is calculating the intensity weighted centroid of an image respectively a fiducial marker (see subchapter 7.2.1). It takes a single AIM file as an input. Fiducial markers are assumed to be brighter then the background (default settings). If this is not the case, the image gray-values can be inverted (`IWC->InvertOn();`).

Example:

```
vtkIntensityWeightedCentroid *IWC = vtkIntensityWeightedCentroid::New();
IWC->SetInput(fixedReader->GetOutput());
IWC->InvertOff();
IWC->Update();
IWC->GetCentroid(centroid);
```

I.1.10 vtkPointBasedRegistration

After having evaluated all the centroids of the fiducial markers, the transformation parameters can be calculated with the orthogonal procrusted method as described in subchapter 7.2.2.2.

This is also a purely VTK based function, taking two `vnL_matrices` with the centroid coordinates as an input, and returning the rigid or similarity transform parameters and the fiducial registration error (FRE).

Example:

```
vnL_matrix<double> fixedPointSet(nFiducials,3);
vnL_matrix<double> movingPointSet(nFiducials,3);

vtkPointBasedRegistration *PBR = vtkPointBasedRegistration::New();
PBR->SetInputFixed(fixedPointSet);
PBR->SetInputMoving(movingPointSet);
PBR->ScalingOn();
PBR->Update();
transform = PBR->GetTransform();
fre = PBR->GetFRE();
```

I.1.11 vtkBrukerReader

The acquired MRI raw data is processed with the Bruker MRI software and is stored in the software-provider's own file format. The viva-CT 40 scans with the Scanco Medical scanner are stored in the company's own file format, called AIM.

vtkBrukerReader transforms the Bruker data to a VTK image which subsequently can be saved as a AIM file. Several aspects had to be taken into consideration for the format conversion:

- Bruker data can be saved as 16 or 32 bit integers. Both word types are common. AIM data is stored as 16 bit integers. The transformation from 16 bit Bruker to 16 bit AIM does not need any additional treatment. 32 bit Bruker data though needs to be downsized to 16 bit. This leads to a loss of information. Three approaches are implemented to face this problem:
 1. **Ignore:** The information loss is ignored and 2^{16} intensity levels are merged to form one new intensity level. This induces a big information loss.
 2. **Fit:** This function searches for the smallest and biggest intensity value used in the image. The intensity values within the boundaries are merged to fit the 16 bit word type. In the worst case, this method is no better than the ignore approach, even if only very little information is stored in the very low and very high intensity values.
 3. **Histogram:** The histogram approach tries to narrow the fitting range by cutting of 0.1% of all pixels from both sides.
- The byte order in Bruker raw data can be both, little- or big-endian. Aim files are stored in little-endian. Therefore, this application is capable of reading data in both byte-orders.
- MRI data can be acquired in 2D slices or 3D images. In either of these cases, different image parameters have to be read out from various files.
- The Bruker file format was recently updated, and therefore several files have new names and contents. Depending on the acquisition method, the old format is still used. The program therefore recognizes the format and reads in the correct files and parameters.
- When the data is read, the reconstructed image can optionally be mirrored. This feature does only have to be applied, if the read out direction in the MRI and CT scan are inverse to one another.

- As a second option, the y and z axis of the reconstructed image can be flipped. This add on is useful if the read out direction in the MRI scan was not parallel to the static magnetic field vector, but perpendicular to it.
- `vtkBruker2Aim` provides the user with the option to add all the Bruker config files to the AIM log.

`vtkBrukerReader` only needs the file path where the Bruker data is stored, and will automatically extract all the necessary information, to transform the data into a VTK image (short word-type).

This class is also available as a ready-to-use application which saves the Bruker data directly as a AIM file (see "AppBruker2AIM.exe" on the enclosed DVD).

Example:

```

vtkBrukerReader* bruker = vtkBrukerReader::New();
bruker->SetFilePath(filepath);
bruker->GenerateLogOn();
bruker->ReadIntoLogACQPOn();
bruker->ReadIntoLogIMNDOn();
bruker->ReadIntoLogMethodOn();
bruker->ReadIntoLogRecoOn();
bruker->ReadIntoLogGrdProgOff();
bruker->ReadIntoLogPulseProgramOff();
bruker->ReadIntoLogSpmam00ff();
bruker->ReadIntoLogSpmam10ff();
bruker->ReadIntoLogD3ProcOff();
bruker->ReadIntoLogMetaOff();
bruker->ReadIntoLogROIOff();

bruker->FlipYZAxisOff();
bruker->MirrorImageOff();

bruker->DebugOn();
bruker->SetDataConversionMethod(conversionMethod);
bruker->Update();

```

I.1.12 AppCheckerBoard

With this program, slices of two images can visually be compared by creating an image checker board. It is up to the user to choose the slice direction, slice number and number of checkers in each direction. The program takes account of any

differences in the position, size, spacing and range of intensity. Therefore, the field of view is limited to the region where both images are overlapping each other. The images are resliced and a color-map is applied to each image separately to optimize the contrast. The 2D checkerboard image is saved as a bitmap file. The size of the image can be changed by changing the scaling factor.

The CheckerBoard application takes an AIM file and the name of the output bitmap file as an input. To switch from default mode to manual mode, the command "-manual" has to be added after the two file names. The manual mode guides the user through all the steps.

The default values for the auto mode are: Two checkers in each direction of the xy-plane, where the middle slice is displayed with a magnification factor of 1.

Usage: AppCheckerBoard Input1.aim Input2.aim Output.bmp [-manual]

I.1.13 AppAIMCube

This application simply creates an AIM file and draws a cube with user defined dimensions inside the matrix. The cube has the foreground value 1 and all the other voxels are set to the background value 0. The user enters the image dimensions, the spacing, the cube position, the cube dimensions as well as the file name of the new image. This program was used to create the segmentation accuracy and precision image.

Usage: AppAIMCube Output.aim [Dimensions][CubePosition][CubeDimensions][Spacing]

Example: AppAIMCube Output.aim 64 64 64 6 12 6 32 26 40 0.035

I.1.14 AppAIMMerge

Two AIM files are merged. Voxel by voxel is compared and set to the value with the higher intensity respectively to the value with the lower intensity, if the option "-invert" is chosen. The images do not have to have the same properties (dimensions, position, spacing). The program automatically resamples the image (linear interpolation). The dimension is expanded to the smallest values, which include both images and the spacing is set to the image with the higher resolution. To run the function, two input file names and an output file name have to be set.

This program was used to create the segmentation accuracy and precision image and to fuse the 3D model partitions.

Usage: AppAIMMerge Input1.aim Input2.aim Output.aim [-invert]

I.1.15 AppAIMNoiseFilter

This program changes the contrast and adds noise to a foreground/background based image. The noise and contrast are adjusted by the signal-to-noise-ratio (SNR) and the contrast-to-noise ratio (CNR). Beside these two values, the application asks for the desired object intensity and the input and output file names. If the applied noise is assumed to be random with a uniform distribution, equations 13.1 and 13.2 can be simplified to

$$SNR^* = \frac{\bar{I}_o}{\sqrt{\frac{\sum_{i=1}^N (random)^2}{N}}} \xrightarrow{N \gg 0} \frac{\bar{I}_o}{random} \quad (I.1)$$

$$CNR^* = \frac{\bar{I}_o - \bar{I}_b}{\sqrt{\frac{\sum_{i=1}^{M+N} (random)^2}{M+N}}} \xrightarrow{M, N \gg 0} \frac{\bar{I}_o - \bar{I}_b}{\sqrt{2} \cdot random} \quad (I.2)$$

If M and N are big, the *random* distribution can be rewritten as a geometric series, where R is the range of the *random* noise.

$$random \xrightarrow{M, N \gg 0} \frac{2 \cdot \sum_{i=1}^R R^2}{2R + 1} = 1/3 \cdot R \cdot (R + 1) \quad (I.3)$$

Equations I.1 and I.2 can be solved for the background intensity $\bar{I}_b$.

$$\bar{I}_b = \bar{I}_o - \frac{\sqrt{2} \cdot \bar{I}_o \cdot CNR^*}{SNR^*} \quad (I.4)$$

The range R still has to be calculated. Unfortunately either equation I.1 nor equation I.2 can be theoretically solved for the range R . Here, a simple numerical approach is used, whereby R is increased by 1 each iteration until the expression on the left is bigger than the expression on the right.

$$for (R \cdot (R + 1) < 3 \cdot \bar{I}_o^2 / (SNR^*)^2) R++ \quad (I.5)$$

The object or background intensity as well as the random noise are applied to each voxel in the image. As a control, the SNR and CNR are calculated for the final image. All the parameters are optionally stored in a log file.

Usage: AppAIMNoiseFilter Input.aim Output.aim [SNR][CNR][Color]

Example: AppAIMNoiseFilter Input.aim Output.aim 2 4 1000

I.1.16 AppAIMPos

Simple application to change the position of an AIM image. The program asks for the file name and the new position values.

Usage: AppAIMPos Input.aim [pos]

Example: AppAIMPos Input.aim 10 20 30

I.1.17 AppAIMRelabel

Simple application to relabel the segmented partitions saved as an AIM image. The program asks for the file name, the label number to replace and the label number to replace it with.

Usage: AppAIMRelabel Input.aim [OldLabel][NewLabel]

Example: AppAIMRelabel Input.aim 1 12000

I.1.18 mySNAP

mySNAP is a semiautomatic 3D segmentation tool which is based on the itkSNAP [50] open source program, available in the ITK-Application software package.

itkSNAP implements two well-known 3D active contour segmentation methods: Geodesic Active Contours by Caselles et al. [6] and Region Competition by Zhu and Yuille [5]. The two methods are explicitly described in chapter 11.

It furthermore includes tools for viewing and navigating 3D images, manual labeling of regions of interest, combining multiple segmentation results, and post-processing them in 2D and 3D.

In the mySNAP segmentation program all the features mentioned above are preserved. Furthermore, the user has the following additional options to make use of:

- **Read and write AIM files:** The ability to read and write AIM files was one of the major needs, since most of the images to be segmented are acquired with Scanco Medical CT scanners and therefore are stored in AIM format. mySNAP is capable of reading and writing AIM files of any size and word type (short or char).
- **Denoise pre-filter:** CT and especially MR images usually contain noise and intensity drifts. These distortions can lead to significant problems while trying to segment volumes of interest. In case of subsequent edge snake segmentation without pre-filtering, noise is falsely detected as edge borders. These artifacts

might make a proper segmentation impossible.

In mySNAP, five different filters are implemented, which all try to diminish distortions with different approaches. The filters are a simple Gaussian blur filter, an opening and closing filter, a gradient anisotropic diffusion filter, a minima maxima curvature flow filter and a bilateral filter. The bilateral filter is deactivated at the moment due to its very slow performance on 3D images. Chapter 12 gives a detailed description of the algorithms and functionality of the individual filters.

If the image contains only little distortion artifacts, the user has the option to skip the denoise filter.

- **Edge filter:** In mySNAP there are in general two methods to segment an image, the edge based and the region based segmentation. In the region based segmentation, the boundaries are set by intensity regions, in case of the edge based segmentation, the region edges are evaluated and used as a segmentation bounding condition.

The user is free to set the upper and lower threshold for the region method and has several filter options in the edge detection method to choose from. The options are a simple gradient filter, a zero crossing filter and a Canny edge detection filter. Chapter 12 provides more information on the functionality of these filters.

All the filters (inclusive denoise filters) have a preview option, where the effect of changing filter parameters can directly be observed. Therefore, the user can optimize the filter values with a direct visual response.

- **Elapsed time:** Time is usually an important factor in data processing. mySNAP calculates the elapsed time for each filter as well as for the segmentation. The timer takes only the actual processing time in account and skips moments where the segmentation is not running. This is especially useful if the segmentation tool is used in the "stop and go" mode.
- **Logfile:** The user has the option to save all the parameters of the used filters, the segmentation parameters and statistics as well as the elapsed time in a separate text file.
- **Save segmentation result as an AIM file:** The segmentation results can be saved in different file formats, among others in the AIM format. All the segmented voxels are saved with their label as an intensity value, ranging from 1 to a maximum of 256. The AIM file can be of char or short type. The content of the logfile described above, is written to the AIM log. If the input file was of AIM format, all the information about the previous image processing steps are added to the AIM log as well.

- **Load AIM file segmentation results:** To view the segmentation results or to use a previous segmentation result as an initial guess for subsequent segmentations, labeled AIM files can be loaded. This might be a useful and time saving option if many similar registered images have to be segmented.

mySNAP is based on a graphical user interface (GUI) and therefore does not need any command line input parameters.

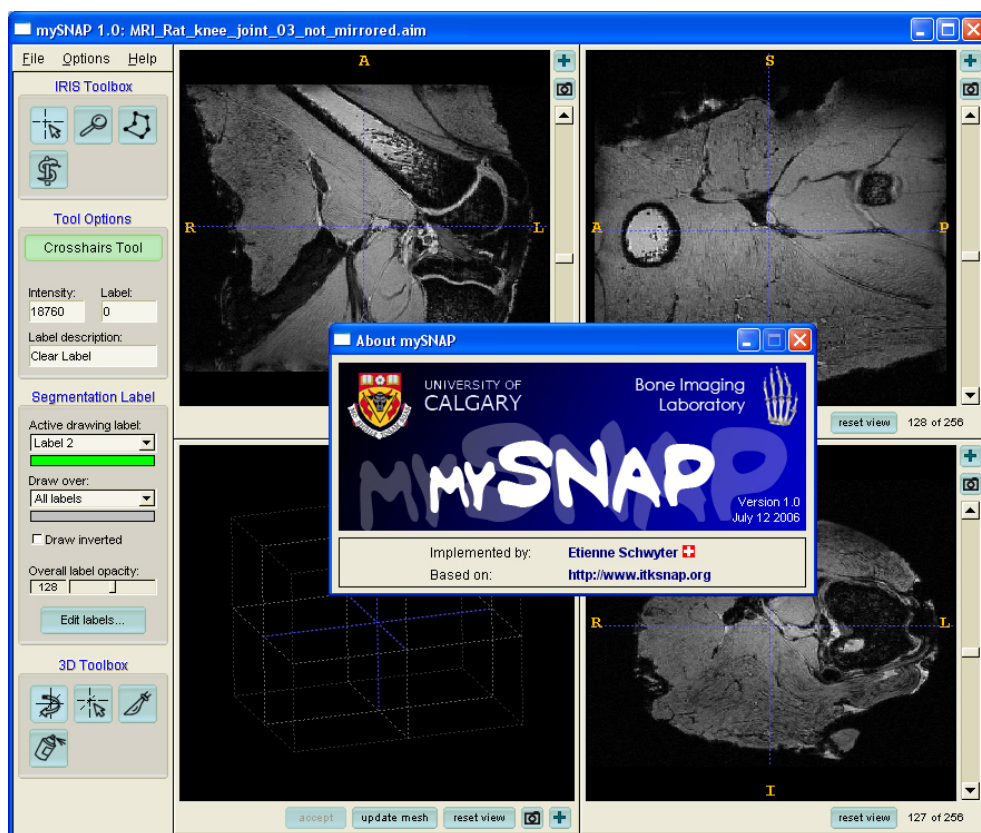


Figure I.2: PrintScreen of mySNAP in the image loading mode.

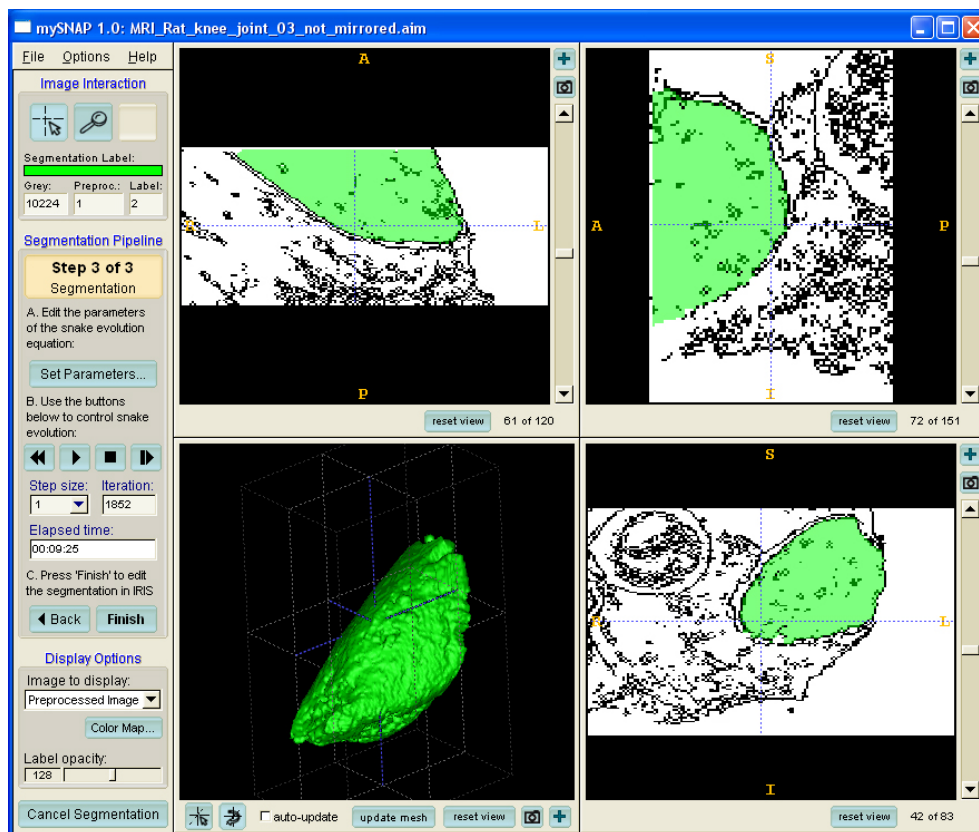


Figure I.3: PrintScreen of mySNAP in the segmentation mode.

I.2 Used

I.2.1 Alibre Design

Alibre Design [1] is a powerful 3D CAD software. It provides parametric modeling in a 3D environment, associative 2D drafting and detailing.

This software was used to design the CT/MRI compatible accuracy and precision phantom.

I.2.2 CUTOOUT

CUTOOUT is a program provided by Scanco Medical AG. CT scanner "raw" data (ISQ format) can be read in, segmented and cut. The output is a segmentation mask saved as a GOBJ file and the cut out AIM file.

This software is in general used to trim the images to the volume of interest and simultaneously reduce the data size. The program was used to cut out the CT scanned rat knee joints as well as the fiducial markers for the point based registration. The Scanco Medical user manual [40] describes the CUTOOUT functionalities into more detail.

I.2.3 IPL

This Scanco Medical AG software is manipulating AIM files. Among others, the program takes the GOBJ files and applies the segmentation to the AIM image. Rotation, mirroring, and cropping are other applicable functions of IPL (see also [40]).

I.2.4 SAP

Scanco Medical offers a third application to process scan data. SAP reads in AIM files and has diverse functions such as image filtering, segmenting and cropping. In this project only the crop function was used to cut out the MRI data which was transformed into AIM files by applying the AppBruker2AIM application, as previously described.

Check the Scanco Medical user manual [40] for a detailed description of the abilities of this program.

I.2.5 VIS and SHAPE

These programs are powerful QT based graphical user interface (GUI) applications for AIM and VTK data visualization and manipulation.

SHAPE [15], implemented by J. Gire, supplies the user with a broad range of different functions such as distance and thickness measurements, Gradient images, 2D slices, intensity histograms and lately the similarity registration program described in chapter 7. The registration initial guess pick-point function and most of the image visualization features in VIS implemented by Dr. S. Boyd, are also available in SHAPE.

These programs were mostly used to visually check the quality of the registration and segmentation results.

I.2.6 SiteSpecificThickness

The SiteSpecificThickness filter was implemented by Kuhn and Mattmann [27]. It tries to fit maximum spheres into the volume of interest to calculate a (volume weighted) mean thickness, as first introduced by Hildebrand and Rüegsegger in 1996 [17].

The iso-surface is colored accordingly to the thickness variation. All the thickness measurement values are included in the histogram and the mean value as well as the standard deviation are calculated.

This application was used to evaluate the thickness of the CT and MRI scanned phantom trenches and drilling holes for the data acquisition accuracy and precision analysis as described in chapter 4.

Usage: AppSiteSpecificThickness [Arguments] Input.aim

Example: AppSiteSpecificThickness -d 0 -t 5500 -o 0.5 -w Output.vtk Input.aim

This is an example where no Gaussian smoothing is applied to the thresholded image. The offset is set to 0.5mm and the result is saved as a VTK-file.

Arguments:

gaussian image smooth settings

-d arg : standard deviation (df. 1)

-f arg : radius factor (df. 0)

marching cubes settings

-t arg : threshold (df. 30)

padding options (close the boundary of the bone)

-c : close the boundaries (df. off)

site-specific thickness filter settings

-s : change method to spheres surface weighted (df. spheres)

-n : change method to normals (df. spheres)

-m : measure separation with set method (df. off)

-b arg : set boundary tolerance [mm] (df. 0.1)

-p : pass normals off (df. on)

settings only for the spheres and surface weighted spheres method

-o arg : set initial offset sphere radius [mm] (df. 0.005)

-a arg : change accuracy ($= 2^{arg}$ [um]) (df. 0)

settings only for normals method

-l arg : set length of normals [mm] (df. 2)

histogram

-h : do not show the histogram (df. on)

-h2 arg1 arg2 : set the range of the histogram (df. auto)

output

-w arg : write output of site-specific filter to .vtk file

lookup table

-r arg1 arg2 : set lookup table range [mm] (df. auto)

render window

-x : don't display axes (df. on)

-z : don't visualize the results in a render window (df. on)

