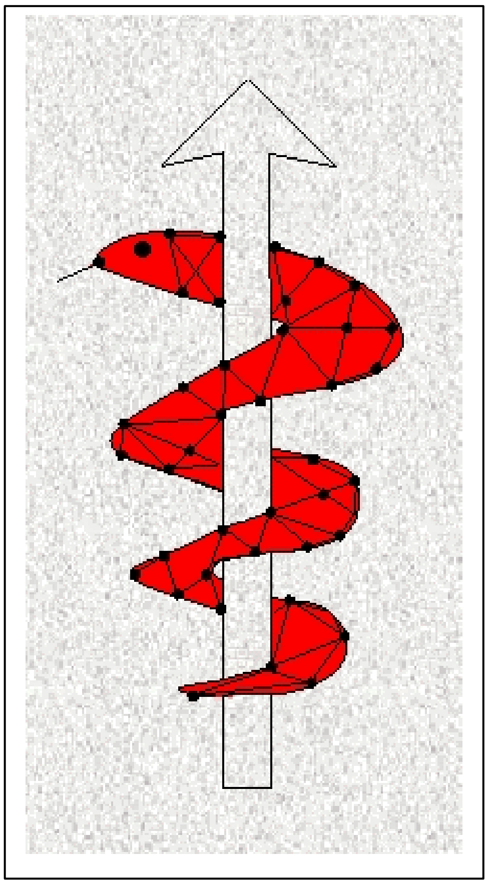
The IST Programme
Project No. 10378

SimBio

SimBio - A Generic Environment for Bio-numerical Simulation

http://www.simbio.de



Deliverable D4.2c
Inverse Field Reconstruction
Final Release
Release Notes v1.0.0

Status:	final
Version:	1.0
Security:	Public
Responsible:	MPI

Release History

Version	Date	
0.1	2002-10-10	initial draft
1.0	2002-11-10	release

1. Introduction

This document outlines the implementation of the algorithms designed for the **SimBio** bio-mechanical inverse field reconstruction tasks. As described in the deliverable D4.2a there is one major approach to inverse field reconstruction that is pursued within the **SimBio** project, i.e. a *consistent linear- or visco-elastic registration* and the analysis of the achieved vector fields.

These release notes describe the current state of the software including documentation for accessing, installing and using the tools. Options to the actual software that need to be specified by the **SimBio** user will be thoroughly explained.

This was planned to be the final release in **SimBio** subtask ST4.2. However, as we will outline in the final chapter of this notes, few features are still incomplete. These features will be revised or added soon.

2. Status of the Implementation

In accordance with the development plan, two approaches to bio-mechanical inverse field reconstruction have been followed within ST4.2, *consistent linear-elastic* and *fluid dynamic* image registration. Both approaches differentiate with respect to their applicability. The consistent linear elastic scheme allows for smooth, diffeomorphic transformation fields, but is restricted to small deformations. The fluid dynamic approach explicitly considers large deformations. To derive realistic models, both elasticity operators are driven by realistic material parameters, as provided in database WP2.

Both registration approaches work on 3D images and are computational expensive. Therefore, additional effort was spend on performance optimization (in terms of HPC and amendment of cost functions, as proposed by D4.2a and b). The gradient decent minimization within the linear-elastic approach is enhanced by a multi-resolution decomposition (Fourier series parameterization) of the displacement fields. In case of the fluid dynamic approach, the intrinsic minimization routine is promoted by Gauss-Southwell optimization. Both registration schemes apply multi-grid computation and facilitate multi-threading; they are implemented as programs vlet3d and vfluid3d. Additionally, the current ST4.2 release comprises tools to post-process the registration output. A program is provided to analyse vector fields for its critical points (vcpdetect). Another (vmaddvf), to overlay the registration results on a triangulated surface in order to achieve advanced visualization with the vm tool. The program vassignshift3d allows appling deformation fields to VISTA images.

2.1 The Program vlet3d

The registration algorithm described in D4.2a has been implemented in the C++ programming language using the VISTA library for image data handling and for the generation of graphical output. As described in D4.2a, to gain efficiency in solving the optimisation process a multi-resolution decomposition of the displacement fields is done. This decomposition is realised by a Fourier series parameterisation of the displacements. For the respective Fourier transforms the program uses the FFTW library which is freely available (http://www.fftw.org) under the GNU software license agreement and has to be installed before compiling vlet3d (see 3.3.2 FFTW). The current version (1.3) supports intensity-based registration of images with homogenous and inhomogenous material parameters.

For an extensive discussion of the underlying theory please confer the deliverable D4.2a, or the accompanying manuscript *M. Tittgemeyer & F. Kruggel, Inverse field Reconstruction–An Approach Towards Consistent Image Registration: Part I, The Theoretical Framework* which is available on the protected section of the **SimBio** home page.

2.1.1 The Basic Algorithm

The current version (1.3) of the vlet3d program underwent extensive testing and optimisation with respect to performance enhancement, amendment of cost functions and the elasticity operator (as it has been proposed in D4.2a and b). However, different sets of simulated images were developed for testing of correctness of the implementation and to optimise computation speed. However, even the optimised algorithm turns out to be computational demanding, in terms of memory usage as well as with respect to computation time, e.g. for 64x64x64 images the program will use ~60 Mbyte of memory and run ~1 hours, for 256x256x256 images it takes ~1.5 Gbyte of memory and a computation time of ~8 hours.[1] Therefore, we provide full multi-threading (pthreads) support and the possibility to further enhance performance by multi-grid computation. The decision to use this options is done at compile time; the Makefile can be configured respectively.

The basic version of our algorithm evolves ten steps:

1. Initialise starting set of forward and reverse basis coefficients.
2. Compute the inverse displacement field of the reverse transform.
3. Compute new forward basis coefficients.
4. Compute the forward displacement field.
5. Compute the inverse displacement field of the forward transform.
6. Compute the new reverse basis coefficients.
7. Compute the reverse displacement field.
8. If a particular criteria (**-inc**, see below) is met to increase the basis coefficients, increase the number of harmonics and clear coefficient fields.
9. If the algorithms has not converged or reached the maximum number of iterations (**-nit**, see below) return from step 2.

2.1.2 Classes for matrix, vector, and vector field handling

Along with the development of the registration software, fully templated classes for 3D fields, matrices, vectors and vector field operations were implemented. The respective interfaces can be found in the *include* directory of the release package: *st42-1.0.0/vlet3d/include* (*field3d.h*, *matrix.h*, *vector3d.h*, *vectorfield3d.h*).

2.2 The Program **vfluid3d**

The registration algorithm is based on a fluid dynamic approach and has been implemented in the C++ programming language using the VISTA library for image data handling and for the generation of graphical output. As vlet3d, the program utilises a multi-resolution approach to reduce computation times and to avoid local minima in the solution space. It makes use of a Gauss-Southwell optimisation scheme and works on a multi-grid pyramid. The release version supports intensity-based registration of images with homogenous and inhomogenous material parameters. The current version is fully capable of multi-threading.

Note that the memory requirement for vfluid3d are the same as for vlet3d. The computation time for vfluid3d is significantly less than for vlet3d, e.g. for 64x64x64 images the program will run ~8 min, for 256x256x256 images the computation time is ~20 min.

For an extensive discussion of the underlying theory as well as of performance issues please confer *G. Wollny and F. Kruggel, Computational cost of non-rigid registration algorithms based on fluid dynamics, Trans. Med. Imag., 8, 2002.*

[1] For performance tests we used a 1800 MHz dual processor Athlon PC running LINUX.

2.3 The Program vcpdetect

Vector fields, as are obtained by image registration are usually large, and hard to interpret. Critical points provide a sparse description of this vector field and inherently of the underlying dynamic of the mapped process.
The program vcpdetect allows to detect and analyse vector field singularities of any order as critical points. In particular, contraction mapping, is used to find differentiate attractors, repellors and unbalanced saddle points. The implemented approach for rotation centres and balanced saddle points is rather heuristic and needs improvement in the future.

For an extensive discussion of the underlying theory please confer *M. Tittgemeyer and F. Kruggel, Visualising deformation fields computed by non-linear image registration, Comput. Visual. Sci., 5, 45-51, 2002.*

2.4 The Program vassignshift3d

The program is used to apply a deformation field to a VISTA image. Deformation fields may be obtained as output of the non-linear image registration software, i.e. vlet3d or vfluid3d.

2.5 The Program vmaddvf

The program is designed to overlay a vector field image to a triangulated surface mesh. This tool becomes necessary to communicate the registration results of vlet3d or vfluid3d to the visualisation tool (vm) in WP 5. With the current release, the tool is only capable to overly the unprocessed vector field. In a further revision it is foreseen to decompose the vectors, to optionally visualise tangential or orthogonal field components against a surface.

3. How to Access, Install and Use the Software

3.1 Installation and Customisation

The software is downloadable via the protected part of the **SimBio** home page (http://www.simbio.de) and comes together with a configure program in the GNU style. The following steps have to be followed in order to properly install the software:

1. Unpack the release: *tar zxvf st42-1.0.0.tar.gz*
2. Change to the directory st42-1.0.0: *cd st42-1.0.0*
3. Check whether everything is available for a successful compilation: *./configure*
4. Build the system: *make*
5. Install program: *make install* (you may need to have 'root' permission)

This will build the default configuration of the st42 tools, using double single precision, running on a single processor and applying multi-grid routines for the registration tools. We strongly recommend that you use GNU *make* if it is available; on some systems it is called *gmake*. The "*make install*" command installs the program in a standard place, and typically requires root privileges (unless you specify a different install directory with the --*prefix* flag to *configure*). If you have problems with the configuration or compilation, you may want to run "*make distclean*" before trying again; this ensures that you don't have any stale files left aver from previous compilation attempts.

The *configure* script knows sufficient *CFLAGS* and CXXFLAGS (C and C++ compiler flags) for a few systems. If your computer is not known, the script will print a warning. In this case, you can configure the compilation of the software with the command

CFLAGS=<write your *CFLAGS* here> CXXFLAGS=<write your *CXXFLAGS* here> *./configure*

The configure program supports all standard flags defined by the GNU Coding Standards; see the INSTALL file in the st42-1.0.0 directory; note especially --*help* to list all flags. In order to customise the software configure also accepts some st42 specific flags, particularly:

- *--enable-double* Produces a double precision (double) version of vlet3d and vfluid3d instead of the default single precision (float). This can be used to enhance registration accuracy, but will drastically increase memory requirements.
- *--disable-pthreads* Disables the use of multi-threading. Once *configure* finds libpthread on your system, it automatically compiles the registration tools for pthread support. However, on a single-processor machine you may want to disable multi-threading support.
- *--with-wp1* Allows to specify an individual path to the WP1 root.
- *--with-fftw* Allows to specify an individual path to the FFTW library.

3.2 Command line parameters

To get an idea of individual program options available in the current release simply type the respective program name and obtain a list of command line switches together with a brief parameter description. Further help is provided in the man pages to each program.

3.2.1 vlet3d

The following lines reflects the output of the vlet3d command:

-out <string>
Output file. Default: (none)

-in <string>
source image. Default: (none)

-ref <string>
reference image. Default: (none)

-mat <string1> <string2>
material parameter files. Default: (none) (none)

-delta <number>
initial step size for gradient decent. Default: 0.04

-l <number1> <number2> <number3>
Lagrange multipliers for C1, C2, and C3. Default: 1.0 0.0 0.0

-mltgrd true | false
registration on a multi-grid pyramid. Default: true

-inc <integer>
step size to increase no. of harmonics. Default: 100

-nit <integer>
total # of iterations. Default: 200

-threads <integer>
no. of threads being used. Default: 2

-verbose fatal | err | warn | mesg
verbosity level. Default: mesg

To run the program it is sufficient to call ./vlet giving just the template and reference image as input, e.g.:

```
./vlet3d -in <template_image> -ref <reference_image> -out <output vf_file>
```

Any other parameter is optional and will use a default value if not changed explicitly. Note that some parameters are crucial to a successful registration, the respective values strongly depend on the data being registered. The parameter **-delta**, for instance, controls the step size for the optimization algorithm. Experimenting with different values should be done with care. If the value for **-delta** is chosen too large the algorithm will be unstable, if it is chosen too small there will be no change. The criteria for increasing the number of basis functions (**-inc**) is to look at the graphs of the likelihood. When these graphs show convergence the number of basis functions can be increased. The default value of 40 iterations is very aggressive and most time the algorithm will not have converged at the particular resolution. The same holds for the total number of iterations (**-nit**). If this limit has been reached the program will halt, whether the algorithms has converged or not. Both of the latter parameters will be not available if a multi-grid approach is chosen (**-mltgrd** true). Then this parameters are estimated individually in each resolution step from the program.

Setting **-mltgrd** to false disables the multi-grid approach in vlet3d. For parameter calibration and accuracy test it is reasonable to have a program version running on maximum resolution, but it will drastically decrease the performance.

Other crucial parameters are the multipliers to the cost functions (**-l**). If a multiplier is too low the contribution of the respective cost function to the registration will be suppressed, if the value is too large the algorithm will be unstable. Default values are chosen that the registration will be unconstraint, i.e. the registration is solely based on image intensities which most certainly does not guarantee a consistent registration (see D4.2a).

The parameter **-threads** is a conditional feature which does only appear if vlet3d is customized for using multi-threaded routines.

3.2.2 vfluid3d

The following lines reflects the output of the vfluid3d command:

-in <string>
input image. Default: (none)

-ref <string>
reference image. Default: (none)

-out <string>
output vector field. Default: (none)

-method sor | sora | sorap
method for solving the underlying PDE. Default: (sora)

-maxiter <integer>
max. number of iterations for solving the PDE. Default: (20)

-epsilon <float>
rel. error to stop iteration iterations. Default: (0.01)

-startsize <integer>
start size in multi-grid. Default: (16)

-statlog <string>
file to write some statistics. Default: (true)

-maxthreads <integer>
max. number of threads. Default: (2)

-mu <float>

Lamè constant (shearing). Default: (1.0)

-lambda <float>
Lamè constant (compression). Default: (1.0)

-label <string>
material label image. Default: (none)
-material <string>
text file, material parameters. Default: (none)

To run the program it is sufficient to call vfluid3d giving just the template and reference image as input, e.g.:

```
./vfluid3d -in <template_image> -ref <reference_image> -out <output vf_file>
```

Any other parameter is optional and will use a default value if not changed explicitly. Note if a material label image is given, the corresponding file containing material parameters must be given as well. In that case, a inhomogeneous material distribution is anticipated and any given value for **-lambda** and **-mu** will neglected. The parameters "–**method** sorap**"** and **-maxthreads** are conditional feature which will only appear if vfluid3d is customized for using multi-threaded routines.

3.2.3 vcpdetect

The following lines reflects the output of the vcpdetect command:

-in <string>
input vector field. Default: (none)

-out <string>
list of critical points. Default: (none)

-lifetime <integer>
life time of voxels in iteration cycles. Default: (1000)

-nvoxels <5000>
no. of voxels being used. Default: (none)

-use-brownian true | false
field driven movement follows Brownian randomness. Default: (true)

-bweight <float>
weight of the Brownian movement. Default: (2.0)

-gfw <integer>
size of Gaussian filter to smooth random clusters image. Default: (3)

-thresh <float>
threshold for clustering the cumulated points. Default: (0.5)

-cluster true | false
cluster the results and create an array of critical points. Default: (true)

-growth <float>
growth factor for cluster boundary. Default: (1.0)

To run the program it is sufficient to call ./vcpdetect giving just the input (vectorfield) image and a file for the output, e.g.:

```
./vcpdetect -in <vectorfield_image> -out <cp_image>
```

Any other parameter is optional and will use a default value if not changed explicitly. Note, the parameter **–bweight** is obsolete if **–use-brownian** is set to false.

3.2.4 vassignshift3d

The following lines reflects the output of the vassignshift3d command:

-in <string>
input image. Default: (none)

-defo <string>
vector field applied. Default: (none)

-out <string>
deformed image. Default: (none)

-start <integer1 integer2 integer3>
starting location to apply shift. Default: (0 0 0)

-nn true | false
use nearest neighbor interpolation. Default: (none)

To run the program it is sufficient to call vassignshift3d giving just the input image, the deformation field and a file for the output, e.g.:

```
vassignshift3d -in <input_image> -defo <deformation_field> -out <output-image>
```

Any other parameter is optional and will use a default value if not changed explicitly.

3.2.5 vmaddvf

The following lines reflects the output of the vmaddvf command:

-in <string>
input mesh. Default: (none)

-defo <string>
vector field applied. Default: (none)

-out <string>
deformed image. Default: (none)

To run the program call vassignshift3d by giving the input mesh, the deformation field which you want to overly and a file for the output, e.g.:

```
vassignshift3d -in <input_image> -defo <deformation_field> -out <output-image>
```

3.3 External Libraries

3.3.1 VISTA

The Vista format has been accepted as **SimBio** internal file format for medical images as well as for finite element meshes. More information about the Vista format and how to install the libraries (WP1) is to be found on the **SimBio** home page (http://www.simbio.de).

3.3.2 FFTW

To conduct the multi-resolution parameterisations, vlet3d makes extensive use of Fourier transforms which are performed by calling the library FFTW. FFTW is a comprehensive collection of fast C routines for computing a discrete Fourier transform (DFT) in one or more dimensions, of both real and complex data, and of arbitrary input size. FFTW also includes parallel transforms for both shared- and distributed-memory systems. The library is free software (under the GNU software agreement) and is usually faster than other freely-available Fourier transform programs; it can be downloaded from the FFTW home page (http://www.fftw.org). For a installation on non-Unix systems and for a specific customisation please confer the user's manual which is available at the home page. For Unix systems FFTW comes with a configure program in the GNU style. To link the library correctly into vlet3d it needs some special customisation with respect to using both, float or double precision, and to make use of its multi-threading features. However, installation can be as simple as:

```
./configure --enable-float --enable-type-prefix --enable-threads --enable-shared
make
make install

./configure --enable-threads --enable-shared
make
make install
```

Note that the compilation and installation have to be performed twice, with the *--enable_float* parameter being set and without (using double precision is default). To install the libraries correctly you need to have 'root' privilege. The current version of FFTW is 2.3.1.

4. Missing Features and Known Bugs

For the explicit development plan please confer deliverable D4.2a (design report) and TA (revision 16.06.01). Despite of landmark based regularisation in the registration tools, all proposed items have been successfully implemented. Both registration approached have been proven successful to invert for structural changes in medic al scan data, but appeared to computational expensive; its results are difficult to understand. Therefore, much effort has been spent in performance optimisation as well as in the analysis of the deformation fields. At the time being, the software in this release is applicable on a recent PC architecture, it performs reasonably well. Having reached this status, we will implement the landmark based routines.

Bugs in any of the tools are not yet known.

5. Acknowledgement

The initial ideas of the underlying theory to the registration software has been first formulated by Gary E. Christensen. We are especially thankful to him for providing partly unpublished work.

6. Feedback

For questions, comments, and suggestions please contact

M. TITTGEMEYER, G. WOLLNY or F. KRUGGEL
MAX-PLANCK INSTITUTE OF COGNITIVE NEUROSCIENCE
STEPHANSTR. 1A
04103 LEIPZIG, GERMANY

E-mail address: {tittge,wollny,kruggel}@cns.mpg.de

We are grateful for any feedback and interest!