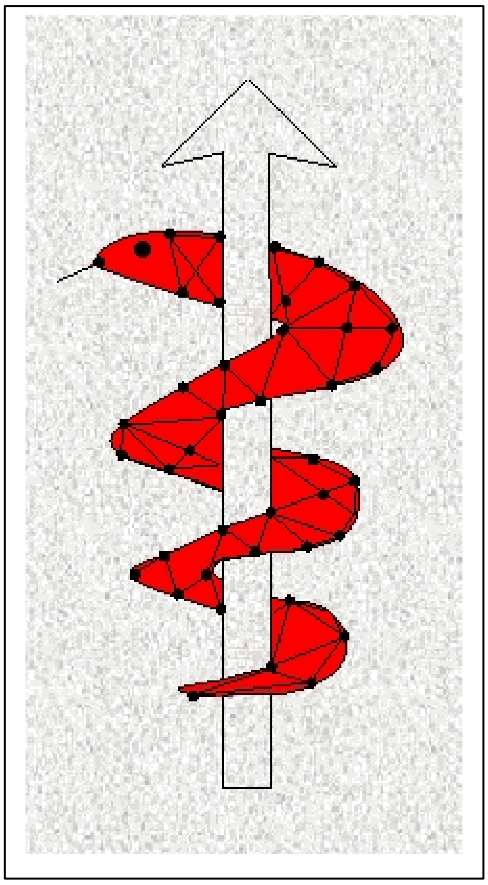
The IST Programme
Project No. 10378

SimBio

SimBio - A Generic Environment for Bio-numerical Simulation

http://www.simbio.de



Deliverable D4.2b
Inverse Field Reconstruction
Release Notes

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

Release History

Version	Date	
0.1	2001-04-27	initial draft
1.0	2001-04-30	release

The SimBio Consortium :

NEC Europe Ltd. – UK
A.N.T. Software – The Netherlands
CNRS-DR18 – France
ESI Group – France
MPI of Cognitive Neuroscience – Germany
Biomagnetisches Zentrum Jena – Germany
Sheffield University – UK
Smith & Nephew - UK

1. Introduction

This document outlines the status of 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 followed within the SimBio project, i.e. a *consistent linear- or visco-elastic registration*.

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. For an extensive discussion of the underlying theory please confer the deliverable D4.2a, or the manuscript *M. Tittgemeyer & F. Kruggel, Inverse field Reconstruction– An Approach Towards Consistent Image Registration: Part I, The Theoretical Framework* which will soon be available on the protected section of the SimBio home page. In the final chapter of this text a roadmap describing the way to completion of the software tools will be given by sketching the technical development plan.

2. Status of the Implementation

In accordance with the development plan we use in this first implementation a linear-elastic operator to constrain continuum mechanical plausibility of the transformation fields. For this reason, and sticking to the VISTA standard, the implemented release is called VLet (Vista contained program for Linear elastic transforms). After first successful testing we start with version 1.0.

2.1 The VLet Program

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 VLet (see 3.3.2 FFTW). It is assumed that 3D datasets in the format proposed by D1.1a is input for our program. The current version (1.0) of the program can only handle pixel (intensity) based images.

2.1.1 The Basic Algorithm

Currently the basic version of VLet is implemented as an initial version. It underwent extensive testing, but is worth to advance in terms of HPC and the amendment of cost functions and the elasticity operator (this has already been proposed in D4.2). 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 vastly 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 ~2 hours, for 256x256x256 images it takes ~1.5 Gbyte of memory and a computation time which is approximating a week. For performance tests we used a 500 MHz dual processor PENTIUM PC running LINUX. The Fourier transforms can already make use of multi-threading.

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.
10. Use forward displacement field to transform the template and reverse displacement field to transform the target.

As this algorithm is yet serially executed the performance is moderate. A significant speed-up can be expected by a parallel estimation of the forward and the reverse basis coefficients and the respective displacement fields (steps 2–7). Further, better performance could additionally be achieved by a multi-resolution approach directly on the template and reference images. This can, depending on the final resolution, also require significantly less memory. It is proposed in the deliverable D4.2a that performance could be increased by the use of more sophisticated optimisation algorithms. Some current alternative approaches have been evaluated, this implementation can, however, be regarded as optimal in that respect.

2.1.2 Classes for matrix, vector, and vector field handling

Along with the development of the registration software, templated classes for matrices and vector field operations were implemented. The respective interfaces can be found in the *include* directory of the release package: *st4_2/include* (*matrix.h*, *vector.h*, *vectorfield.h*).

3. How to Access, Install and Use the Software

3.1 Installation and Customisation

The VLet program is part of the Workpackage 4 (Subtask 4.2) software. Its 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 st4_2.tar.gz*
2. Change to the directory vlet: *cd st4_2/pgms/vlet*
3. Check whether everything is available for a successful conpilation: *./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 VLet, using double precission and running on a single processor. 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 preveous compilation attempts.

The *configure* script knows good *CFLAGS* (C compiler flags) for a few systems. If your computer is not known, the script will print a warning. In this case, you can compile VLet with the command

make CFLAGS=<write your *CFLAGS* here>

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

- --*enable-float* Produces a single precision (float) version of VLet instead of the default double-precision (double). This can be use to reduce memory requirements, but possibly by the cost of diminished accuracy.
- --*enable-threads* Enables the use multi-threading (with version 1.0 this only provides a simple interface to parallel Fourier transforms for SMP systems.
- --*path-to-vista* Allows to specify an individual path to the VISTA library
- --*path-to-fftw* Allows to specify an individual path to the FFTW library

3.2 Command line parameters

To get an idea of the options available in the current release of VLet one can simply type

```
./vlet
```

and as a result one obtains a list of command line switches together with their possible parameters. The following lines show the output of the ./vlet command:

```
-help
   Prints usage information.

-src <string>
   template image. Default: (none)

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

-c1 intensity | gradient magnitude | mutual information
   basis for initial cost. Default: intensity

-c3 linear elastic | visco elasic | jacobian based
   diffeomorphic constraint. Default: linear elastic

-data_path <string>
   path to output files. Default: ./data

-delta <number>
   initial step size for gradient decent. Default: 1

-format simbio | ipe
   file format convention to store vector field. Default: simbio

-inc <integer>
   step size to increase no. of harmonics. Default: 40

-j [ true | false ]
   protocol Jacobian of transformations after each iteration step. Default: true

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

-lag <integer>
   window lag used in search for incverse transforms. Default: 3
```

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

-mapfkn [true | false]
store displacements (u and w) to files. Default: true

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

-version [true | false]
display version and exit. Default: false

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

```
./vlet -src <template_image> -ref <reference_image>.
```

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. 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).

In accordance with D4.2a the cost function C1 and C3 can be redefined (**-c1** and **-c3**, respectively). However, the recent version (1.0) of VLet has been tested for an intensity-based C1 cost function and for a linear-elastic operator contributing to C3. This validation find its expression in the default values of **-c1** and **-c3** which shall not be changed within this program release.

For Unix-based operating systems, the path to output files (**-data_path**) can alternatively be set by defining the environment variable DATAPTH, e.g. within C-shell type or include to your .cshrc file

```
setenv  DATAPATH  <data_path>,
```

within bash-, borne, or korn-shell type or include to your .bashrc or .profile file

```
export DATAPATH=<data_path>.
```

The parameter **-threats** is a conditional feature which does only appear if VLet is customized for using multi-threaded routines. With the current version of VLet this does only apply for using multi-threaded Fourier transform.

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 is to be found on the SimBio home page (http://www.simbio.de).

3.3.2 FFTW

To conduct the multi-resolution parameterisations, VLet 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 VLet 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-type-prefix --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.

4. Future Planning

For the explicit development plan please see deliverable D4.2a (design report).

As it turned out that the algorithm is usually computationally demanding, the software requires exceptionally for high performance computing (HPC) and needs some advance in that respect (see 4.1.1). The proposed parallelisation of the code and the multi-resolution approach to further sophisticate the algorithm will be undertaken in addition to the original development plan.

5. Acknowledgement

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

6. Feedback

For questions, comments, and suggestions please contact

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

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

We are grateful for any feedback and interst!