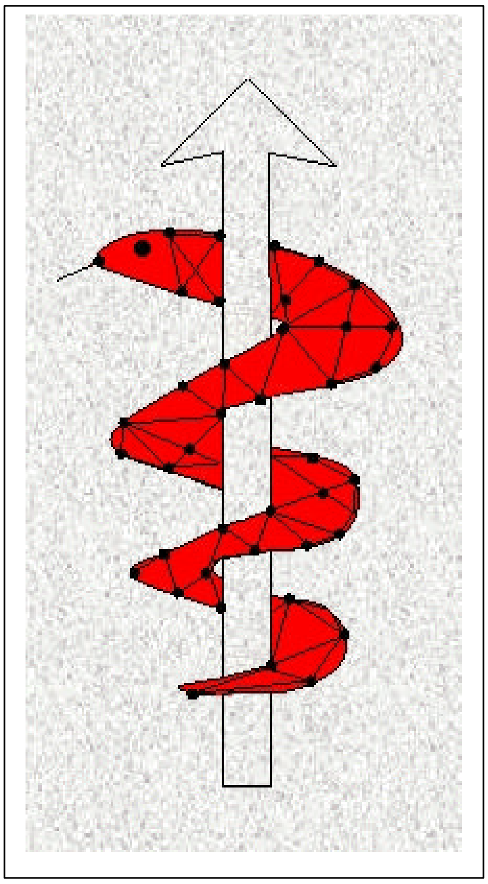
The IST Programme
Project No. 10378

SimBio

SimBio - A Generic Environment for Bio-numerical Simulation

http://www.simbio.de



Deliverable D1.1a
Image Processing Design Report

Status:	Final
Version:	1.0
Security:	Public
Responsible:	USFD
Authoring Partners:	USFD, MPI

Release History

Version	Date
0.1	01.09.00
1.0	29.09.00

1 Introduction

The first subtask of Workpackage 1 deals with image pre-processing and segmentation. Input to the SimBio applications is a volume dataset of a body-part of interest (BOI), most typically acquired on a Magnetic Resonance (MR) scanner. The output of the set of modules in this workpackage is a volume dataset, in which the BOI is aligned with some suitable pre-defined co-ordinate system, interpolated to an isotropical voxel size, corrected for imperfections of the scanner hardware (intensity inhomogeneities, intensity scaling), and finally segmented, so that each voxel receives a material label corresponding to the tissue type as revealed by the imaging technique. This volume dataset is used as input for the meshing process, which is the topic of WP1, Subtask 2.

Image segmentation invariably requires some a-priori information and typical segmentation tools include this information either explicitly or implicitly. Segmentation tools based on image intensity values implicitly assume that the intensity within an anatomically significant segment is uniform, or close to uniform. Segmentation tools based on boundary detection often assume that the boundaries of an anatomically significant segment are associated with relatively large changes in intensity. In the general case these assumptions may not be sufficient to segment the image without further a-priori information. Particularly useful information is likely to be a ‘fuzzy’ a-priori segmentation based on a previously defined exemplar segmentation which is then refined through the use of existing tools. Fuzzy segments can be superimposed on the image though the use of image registration techniques.

SimBio tools are meant to be generic in the sense that any body part of any individual subject could be used as input. It will not be possible within the SimBio programme to confirm that the segmentation tools are fully generic, but the two applications areas, the human head and the knee, represent two segmentation tasks sufficiently different to support the assertion of generic segmentation. In the case of the head, there is no need to supply additional explicit a-priori information to achieve reliable segmentation of the significant brain compartments, at least for the purposes of SimBio. In the case of the knee additional information will be required. Fortunately the knee can be reliably registered to an exemplar knee and fuzzy segments, already prepared for the exemplar image, superimposed on the subject knee image. These segments can be used to constrain the intensity segmentation tools in such a way as to allow accurate automatic segmentation of the knee. A similar approach could be used for the brain. However, this is not necessary in practice. Conceptually the a-priori segment covers the whole of the brain.

2 Toolkit Overview

Appropriate image processing routines will be provided for SimBio. These are listed below and described in more detail by their manual pages, collected in Appendix A:

- Images within SimBio will be oriented in a standard orientation and converted to an isotropic voxel size using a fourth-order b-spline interpolation in order to ease visualisation of these complex objects and facilitate the automatic use of the image processing modules. Image volumes may be converted to the standard orientation (*vstandard3d*). For the knee, we suggest following the conventions introduced by Winter [1] for the analysis of human gait: the x axis is parallel to the long axis of the femur, the y axis is parallel to the anterior-posterior direction, and the z axis parallel to the left-right direction. The origin of the co-ordinate system is in the centre of the joint. For the head, an additional orientation to the stereotactical co-ordinate system [2, 3] is desirable. A module is provided which aligns head datasets with this co-ordinate system (*valign3d*). This routine needs manual specification of the anterior and posterior commissure in the imaged volume, and the rotation angles of the mid-sagittal plane. These values may be obtained with the help of the SimBio visualisation module.

- Parts of an image volume may be cropped by the routine *vcrop3d*.

- Imperfections of the B1 field of the MR scanner lead to intensity inhomogeneities, which result in difficulties in segmentation. Thus, special care has been taken to estimate this background field and to correct for the inhomogeneities during the classification step. An approach proposed by Pham et al. [4] for segmentation with uniformity correction is used in this module (*vuniform3d*) to correct for B1 non-uniformities.

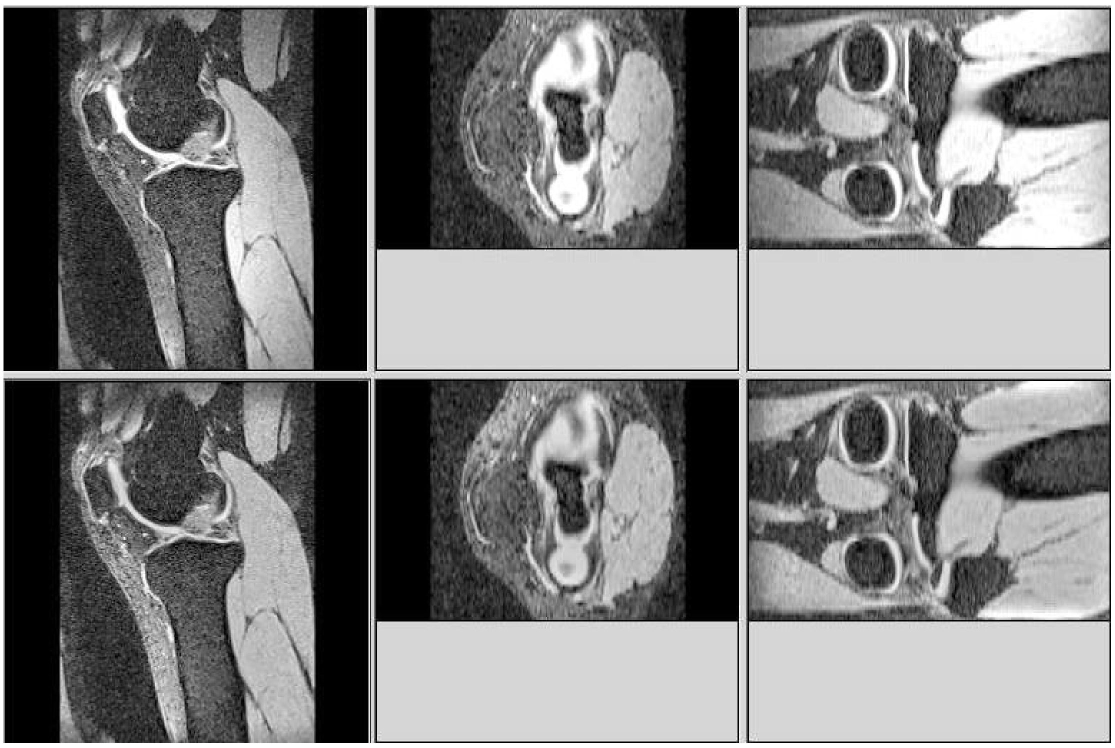


Figure 1. B1 uniformity correction. The upper images represent a knee image before correction and the lower images after correction (Kruggel, MPI).

- One of the SimBio validation application (ST7.2) deals with monitoring changes in time-series images due to disease processes or surgical interventions. Because the BOI is expected not to be oriented in exactly the same position in all examinations, a module for rigid registration of 3D volume datasets is provided (*vreg3d*). This routine uses up-to-date image processing techniques (voxel-wise registration using an entropy-based cost-function, combined genetic and downhill-simplex optimisation) to achieve a highly accurate rigid registration (registration error less than 0.5 voxel) in a moderate amount of computation time (10-40 minutes on standard PCs). This module is also capable of dealing rigid registration of images from different modalities or with different MR sequences, although this option may not be required if MR images are acquired using a standard sequence.

- Generic segmentation will require the use of prior fuzzy segments to guide the segmentation process. To achieve this an exemplar image will be registered to the subject image using a non-linear registration module (*vreglocal3d*). This module will produce a mapping function which will then be used to map pre-defined exemplar segments to the subject image (*vsegmap3d*). These will be used as priors to the local segmentation module.

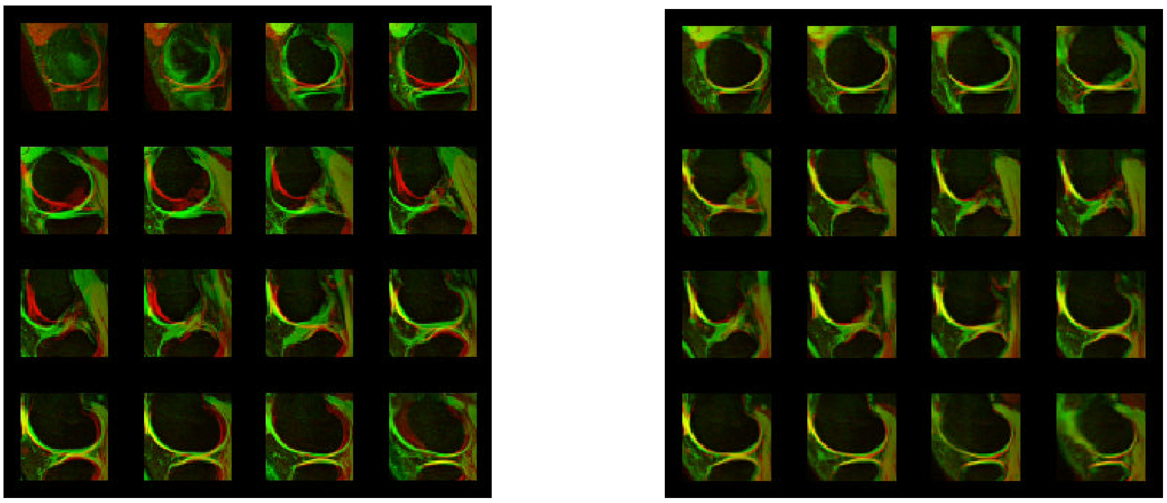


Figure 2. Exemplar image (red) and subject image green before (a) and after (b) non-linear registration.

- Because of the wide range of imaging protocols available, a standard intensity scale does not exist for MR images. In order to make time-series examinations more robust against intensity scaling artefacts in the segmentation process, a module (*vintens3d*) was implemented to automatically adjust the intensity given a spatially registered reference volume.

- Image volumes need to be segmented, i.e. voxels are labelled for their tissue types. For single-channel MR datasets of the head, this is achieved by classification of the voxel intensity using a k-means algorithm. Two modules will be provided for segmentation. The first will perform a global segmentation using an approach proposed by Pham et al. [8] which includes correction for B1 inhomogeneities (*vsegment3d*). This module is optimised to reduce segmentation to 30 min on a single processor system. A new approach for generating a first estimate of the bias field is implemented which uses a spatial homogeneity constraint and provides a much better segmentation result. The second module will perform segmentation (*vsegmentlocal3d*) using prior fuzzy segmentations.

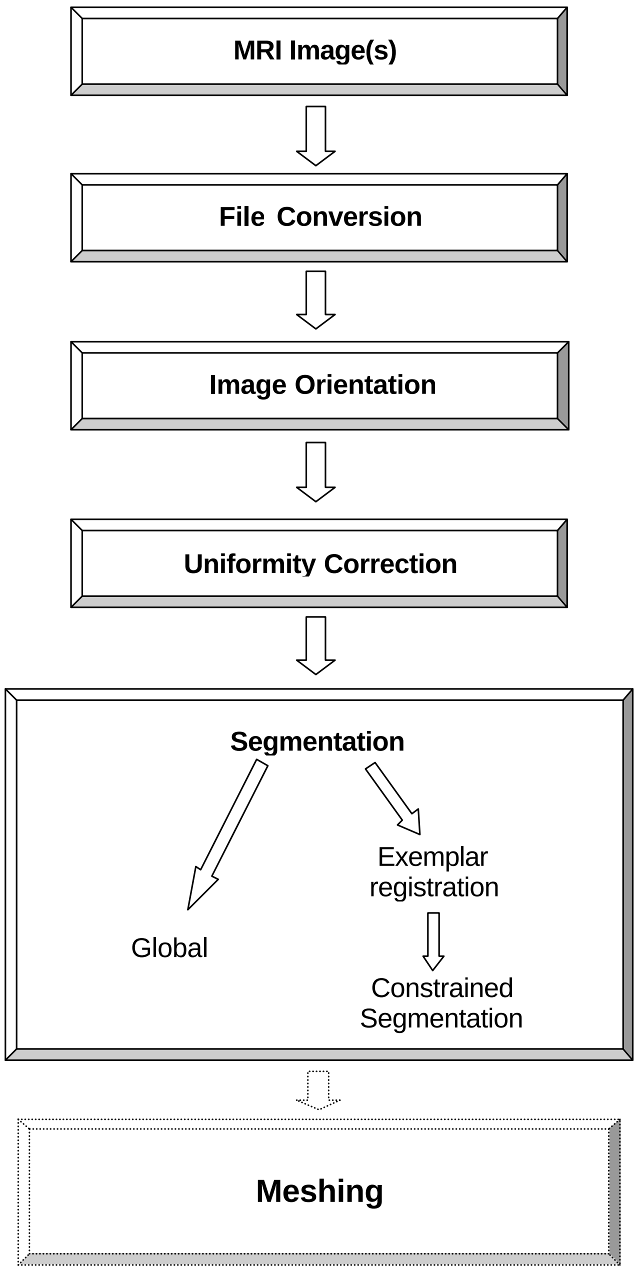
A preliminary subset of these modules has already been released in source code form to the consortium, targeted for Linux workstations, and adhering to the file format definitions as set forth in Appendix B.

The set of tools described above are sufficient to provide quality segmentations for the meshing process, thus allowing the computation of first simulations. However, to provide a sound basis for highly realistic simulations of the biomechanical and electromagnetic properties of the head, two special tissue types must be segmented with high accuracy:

- The skull has a rather low electrical conductivity and a rather high stiffness in relation to other tissues of the head. The thickness of the skull varies between 1 and 8 mm depending on the region, which is expected to influence simulation results strongly. Image processing tools will be developed prior to the first IP-tool release (project month 12) to provide a precise segmentation of the bone in MR datasets of the head. To be able to define this compartment, T1- and PD-weighted datasets must be acquired.

- The meninges consist of a thin layer of tough skin which encapsulates the brain and separates it into mechanically partially decoupled compartments, the hemispheres and the cerebellum. This sheath is at most 1 mm thick, but relevant for a precise modelling of mass shifts in the intracranial compartment. Image processing tools will be developed for a precise segmentation of the meninges in MR datasets of the head and are expected to be available with the second software release for this subtask, i.e., for Milestone 1 at project month 18.

Segmentation of the ligaments within the knee represents a particular challenge. Accurate identification and positioning of the ligaments is critical to simulating the correct mechanical motion of the knee joint. Development of accurate ligament modelling will form a key activity for the WP1.1 group (USFD) prior to Milestone 1 at project month 18.



3 Deliverables

The first IP-tool release (D1.1b, month 12) will contain preliminary but usable versions of the modules described in section 2. Feedback from the users of the tools will be used to improve the robustness of operation where required and iron out any other residual problems leading to the final IP-tool release (D1.1c) at month 30. It is not anticipated that any other significantly novel algorithms will be required to achieve segmentation for the tasks within SimBio, although this position will be reviewed from time to time between D1.1b and D1.1c.

4 File Format

It is of paramount importance for medical imaging applications such as SimBio that data are exchanged smoothly (i.e., without unnecessary conversion steps) and completely (i.e., without loss of concomitant information). Except for the DICOM format, which is unwieldy for volume data and does not provide a format definition for meshes, no storage format for medical data is defined. Because three consortium members already had experience with the Vista toolkit [5,6], this data format was revised for the requirements of SimBio and chosen as the information interchange format in SimBio. A document describing this format and SimBio-specific modifications is included as Appendix B. All the modules described in section 2 pass image and other data in the Vista file format.

- A conversion module from the proprietary format of the Bruker MR scanner to the Vista format (*brutov*).

- A conversion module from the DICOM format (which is supported as an export format on most MR and CT scanners (*dcmtov*).

A file input/output library (*libSimBio.a*) was released to the consortium as C and C++ source code for the Linux platform.

5 References

1) Winter D.A., (1979), Biomechanics of Human Movement, Wiley, New York, ISBN 0-471-03476-2, 11
2) Kruggel F., von Cramon D.Y. (1999) Alignment of magnetic-resonance brain datasets with the stereotactical co-ordinate system. *Medical Image Analysis* 3, 1-11.
3) Talairach J., Tournoux P. (1988) *Co-Planar Stereotactic Atlas of the Human Brain.* Thieme, Stuttgart.
4) Pham DL, Prince JL (1999) An adaptive fuzzy C-means algorithm for image segmentation in the presence of intensity inhomogeneities. *Pattern Recognition Letters* 20, 57-68.
5) Pope A.R., Lowe D.G. (1994) Vista: A software environment for computer vision research. In: *Computer Vision and Pattern Recognition (CVPR'94),* pp. 768-772. IEEE Press, Los Alamitos.
6) Pope A.R., Lowe D.G. (1998) The Vista toolkit: http://www.cs.ubc.ca/nest/lci/vista/vista.html.

6 Appendix A:

6.1 Manpages for ST1.1 Modules

6.1.1 brutov - converts Bruker to Vista format

SYNOPSIS

brutov [-option ...] [infile] [outfile]

DESCRIPTION

brutov converts a group of Bruker data files into a Vista data file.

COMMAND LINE OPTIONS

brutov accepts the following options:

-help	Prints a message describing options.
-in *infile*	Specifies a directory or a tar file that contains the source data.
-out *outfile*	Specifies the output file, which will be a Vista data file.
-white *number*	Specifies the percentage of voxels that will be mapped to white (default 0.5%).
-black *number*	Specifies the percentage of voxels that will be mapped to black (default 10%).
-ds 1 2 ...	Select datasets 1, 2, etc. from an experiment.

The keyword ``to" can be used to specify a range of indices, as in ``-ds 1 to 2". Axial and coronal slices are flipped from the radiologic convention into the natural convention, i.e. the left image side corresponds to the left body side. An attribute "convention: natural" is appended to the list to document this orientation. Note that sagittal slices are not flipped, i.e. they are found "nose left".

EXAMPLES

The following command line

```
    brutov -in WA1T961101.GB1 -out vista-file.v -rep 4
```

expects a directory tree with root WA1T961101.GB1 containing Bruker parameter and data files.

To select only a subset of the datasets in an experiment, use

```
    brutov -in WA1T961101.GB1 -out vista-file.v -ds 4 5 7 to 14
```

This will convert datasets 4, 5, and 7 to 14 only.

NOTES

The attribute *convention* is always set to *natural*, and the attribute *component_interp* set to *intensity*. Conversion of diffusion tensor images is unimplemented, so the attribute *component_repn* is always set to *scalar*.

AUTHOR

Frithjof Kruggel.

6.1.2 dcmtov - convert DICOM to Vista format

SYNOPSIS

dcmtov [infile] [-option ...] [outfile]

DESCRIPTION

dcmtov converts a series of DICOM files into a Vista file. Input filenames are expected to be in printf format (see below), with a basename and indices ranging from first to last. On output, a Vista volume dataset is generated. Individual series within a session are represented as separate Vista image objects within the Vista file. Each image object has its own set of information regarding the patient name, image parameters etc.

COMMAND LINE OPTIONS

dcmtov accepts the following options:

-help	Prints a message describing options.
-in *infile*	Specifies the basename for a DICOM file series. This name should be given as printf format string, so that a specific filename may be generated from this format string and the values of first and last below.
-out *outfile*	Specifies the output file, which will be a Vista data file.
-first *index*	Specifies the index of the first input file, which is a DICOM file.
-last *index*	Specifies the index of the last input file, which is a DICOM file.
-swap *true* | *false*	Specifies if swapping of data words should be performed (default: false).

The message "dcmtov: input file XXX.001 not in DICOM format" notifies that either the filename was misspelled or byte swapping is necessary to convert the file. Try the option -swap. The output file can be specified on the command line or allowed to default to the standard output stream.

EXAMPLE

The following command line

```
    dcmtov -in %03d.ima -out vista-file.v -first 1 -last 203
```

will convert the sequence of DICOM files "001.ima" to "203.ima" into a Vista file.

AUTHOR

Frithjof Kruggel.

6.1.3 vtranspose3d - transpose a 3D dataset

SYNOPSIS

vtranspose3d [-option ...] [infile] [outfile]

DESCRIPTION

vtranspose3d transposes a Vista image file according to the specified co-ordinate changes and returns a re-dimensioned Vista volume dataset.

COMMAND LINE OPTIONS

vtranspose3d accepts the following options:

-help	Prints a message describing options.
-in *infile*	Specifies a Vista data file containing the input images.
-out *outfile*	Specifies where to write the output image.
-xyz *abc*	Specifies the co-ordinate transpositions. Available transpositions: xyz (default), xzy, zyx, zxy, yxz, yzx

Input and output files can be specified on the command line or allowed to default to the standard input and output streams.

EXAMPLE

To convert a sagittal 3D Vista dataset ("nose left") into a standard axial dataset ("nose up") use:

```
vtranspose3d -in xx.v -out xx1.v -xyz zxy
```

AUTHOR

Christian Uhl, Frithjof Kruggel.

6.1.4 vstandard3d - transpose a 3D dataset with isotropic voxels

SYNOPSIS

vstandard3d [-option ...] [infile] [outfile]

DESCRIPTION

vstandard3d transposes a Vista image file according to the specified co-ordinate changes and returns a re-dimensioned Vista volume dataset with isotropic voxels.

COMMAND LINE OPTIONS

vstandard3d accepts the following options:

-help	Prints a message describing options.
-in *infile*	Specifies a Vista data file containing the input images.
-out *outfile*	Specifies where to write the output image.
-xyz *abc*	Specifies the co-ordinate transpositions. Available transpositions: xyz (default), xzy, zyx, zxy, yxz, yzx

Input and output files can be specified on the command line or allowed to default to the standard input and output streams.

EXAMPLE

To convert a sagittal 3D Vista dataset ("nose left") into a standard axial dataset ("nose up") use:

```
vstandard3d -in xx.v -out xx1.v -xyz zxy
```

AUTHORS

David Barber, Christian Uhl, Frithjof Kruggel.

6.1.5 vcrop3d - crop a 3D dataset

SYNOPSIS

vcrop3d [-option ...] [infile] [outfile]

DESCRIPTION

vcrop3d is used to crop a 3D dataset, given a left upper corner and an extent. Both corner point and extent are expected to be given in voxels. The corner point may have negative components. Regions of the destination volume which were not present in the input are filled with the value 0.

COMMAND LINE OPTIONS

vcrop3d accepts the following options:

-help	Prints a message describing options.
-in *infile*	Specifies the source image dataset in Vista format.
-out *outfile*	Specifies the output file, which will be a Vista image.
-corner *x y z*	Specifies the left upper corner (default: 0 0 0).
-extent *x y z*	Specifies the extent of the destination volume (default: 0 0 0).

Input and output files can be specified on the command line or allowed to default to the standard input and output streams.

AUTHOR

Frithjof Kruggel.

6.1.6 valign3d - align a 3D dataset with the stereotactical co-ordinate system

SYNOPSIS

valign3d [-option ...] [infile] [outfile]

DESCRIPTION

valign3d is used to align a 3D dataset with the stereotactical co-ordinate system. Relevant information, i.e., the position of the anterior (CA) and posterior commissure (CP) and the rotation angle must be specified manually.

COMMAND LINE OPTIONS

valign3d accepts the following options:

-help	Prints a message describing options.
-in infile	Specifies the source image dataset in Vista format.
-out outfile	Specifies the output file, which will be a Vista data file.
-ca *x y z*	Specifies the position of the commissura anterior.
-cp *x y z*	Specifies the position of the commissura posterior.
-angle *x y z*	Specifies the rotation around the x, y and z axis.
-transpose *true* | *false*	Transpose a set of sagittal slices into axial slices before adapting the co-ordinate system.

Input and output files can be specified on the command line or allowed to default to the standard input and output streams. CA and CP position are expected in mm, measured from the left upper corner of the image volume. The convention for brain datasets denotes the x axis as parallel to the ear-to-ear direction (from left to right), the y axis to the nose-to-back direction, and the z axis to the body axis (from top to bottom). A positive y rotation (i.e. -angle 0 3 0) nods the head 3 degrees right, and a positive z rotation (i.e. -angle 0 0 3) turns the head along the body axis 3 degrees right. If CA and CP are specified, the x rotation is ignored and determined from these co-ordinates.

AUTHOR
Frithjof Kruggel.

6.1.7 vreg3d - register two 3D datasets

SYNOPSIS
vreg3d [-option ...] [infile] [outfile]

DESCRIPTION
vreg3d is used to register a 3D dataset with a reference dataset. Three different registration plans are implemented: one for intra-modal registration problems (such as T1-T1 registration of head datasets), one for cross-modal datasets (for registering a set of T2-weighted slices to a T1-weighted head dataset), and a "manual" plan, where the registration method, an initial rotation and translation should be specified.

COMMAND LINE OPTIONS
vreg3d accepts the following options:

-help	Prints a message describing options.
-in *infile*	Specifies the source image dataset in Vista format.
-out *outfile*	Specifies the output file, which will be a Vista data file.
-ref *file*	Specifies the reference image dataset in Vista format.
-plan *intra* | *cross* | *manual*	Specifies the registration plan (default: intra).
-resort *true* | *false*	Specifies whether the stacking order of the source dataset should be reversed (default: false).
-scaling *true* | *false*	Specifies whether re-scaling is allowed (default: false).
-angle *x y z*	Specifies the initial rotation angle (default: 0 0 0).
-trans *x y z*	Specifies the initial translation (default: 0 0 0).
-func *nmi* | *cc*	Specifies the type of cost function: normalised mutual information (nmi) or correlation coefficient (cc) (default: nmi).

Input and output files can be specified on the command line or allowed to default to the standard input and output streams. For plans *intra* and *cross*, any values given for *-rt*, *-tr*, or *-method* are ignored. Note that the translation is expected to be given in mm. Typical registration problems need 10-30 min computation time on a PC equipped with a 500 MHz Intel Pentium III processor.

AUTHOR
Frithjof Kruggel.

6.1.8 vreglocal3d – non-linear registration of two 3D datasets

SYNOPSIS

vreglocal3d [-option ...] [infile] [outfile]

DESCRIPTION

vreglocal3d is used to register a 3D dataset with a reference dataset. The mapping function is defined in terms of a trilinear interpolation on a three dimensional grid. The dimensions of the grid need to be specified as well as a registration mask. An optional output (needed for vsegmap) is the mapping function.

COMMAND LINE OPTIONS

vreglocal3d accepts the following options:

-help	Prints a message describing options.
-in *infile*	Specifies the source image dataset in Vista format.
-out *outfile*	Specifies the output file, which will be a Vista data file.
-ref *file*	Specifies the reference image dataset in Vista format.
-mask *file*	The registration region mask
-grid *xyz*	Specifies the registration grid size.
-func *file*	Specifies the file in which the mapping function is stored.

Input and output files can be specified on the command line or allowed to default to the standard input and output streams.

AUTHOR

David Barber.

6.1.9 vsegmap3d – mapping of segments

SYNOPSIS

vsegmap3d [-option ...] [infile] [outfile]

DESCRIPTION

vsegmap3d is used to map a set of 3d segments to a new position. The mapping function is produced by vreglocal3d.

COMMAND LINE OPTIONS

vsegmap3d accepts the following options:

-help	Prints a message describing options.
-in *infile*	Specifies the source segments dataset in Vista format.
-out *outfile*	Specifies the output segments file.
-func *file*	Specifies the file in which the mapping function is stored.

Input and output files can be specified on the command line or allowed to default to the standard input and output streams.

AUTHOR

David Barber.

6.1.10 vintens3d - correct intensities a 3D dataset according to a reference

SYNOPSIS

vintens3d [-option ...] [infile] [outfile]

DESCRIPTION

vintens3d corrects image intensities a 3D dataset, given a reference image. First, intensities in both images are scaled to fit in a 256x256 joint histogram. A regression line is fitted to the joint histogram, and intensities in the input dataset are transformed according to the regression parameters. A threshold may be specified to leave out the background during the computation of the regression line.

COMMAND LINE OPTIONS

vintens3d accepts the following options:

-help	Prints a message describing options.
-in *infile*	Specifies the source image dataset in Vista format.
-out *outfile*	Specifies the output file, which will be a Vista image.
-ref *file*	Specifies the reference image.
-threshold *t*	Specifies the minimum intensity bin for computing the regression (default: 30).

Input and output files can be specified on the command line or allowed to default to the standard input and output streams.

AUTHOR

Frithjof Kruggel.

6.1.11 vsegment3d - segment a 3D dataset

SYNOPSIS

vsegment3d [-option ...] [infile] [outfile]

DESCRIPTION

vsegment3d segments a 3D dataset based on intensity criteria into a set of classes. Optionally, it tries to correct for the intensity variations due to inhomogeneities of the B1 field of the MR scanner. For input data sets aligned with the stereotactical co-ordinate system, the argument -opt true specifies to correct within a minimal subvolume only, for a threefold reduction in computation time.

COMMAND LINE OPTIONS

vsegment3d accepts the following options:

-help	Prints a message describing options.
-in *infile*	Specifies the source image dataset in Vista format.
-out *outfile*	Specifies the output file, which will be a Vista image.
-nc *number*	Specifies the number of tissue classes. Default: 3.
-opt *true | false*	Specifies whether to use a minimum sub-volume (for aligned data sets only). Default: false.
-correct *true | false*	Specifies whether to correct for B1 inhomogeneities. Default: false.
-lambda1 *number*	Specifies the value of the first regularisation constant. Default: 200000.
-lambda2 *number*	Specifies the value of the second regularisation constant. Default: 2000000.

Input and output files can be specified on the command line or allowed to default to the standard input and output streams.

AUTHOR
Frithjof Kruggel.

6.1.12 vsegmentlocal3d - segment a 3D dataset using prior fuzzy segments

SYNOPSIS
vsegmentlocal3d [-option ...] [infile] [outfile]

DESCRIPTION
vsegmentlocal3d segments a 3D dataset based on intensity criteria and a set of prior fuzzy segments into a set of classes.

COMMAND LINE OPTIONS
vsegmentlocal3d accepts the following options:

-help	Prints a message describing options.
-in *infile*	Specifies the source image dataset in Vista format.
-out *outfile*	Specifies the output file, which will be a Vista image.
-seg *file*	Specifies the file of a-priori fuzzy segments.

Input and output files can be specified on the command line or allowed to default to the standard input and output streams. Stored with each fuzzy segment is the number of distinct tissue expected to fall within that segment.

AUTHOR
David Barber.

Pham DL, Prince JL (1999) An adaptive fuzzy C-means algorithm for image segmentation in the presence of intensity inhomogeneities. Pattern Recognition Letters 20, 57-68.

7 Appendix B:

File Format Conventions for the SimBio Project

Frithjof Kruggel
Max-Planck-Institute of Cognitive Neuroscience
Stephanstraße 1, 04103 Leipzig, Germany
e-mail: kruggel@cns.mpg.de

The SimBio environment provides tools for bio-numerical simulations using finite-element modelling techniques. To ensure an efficient data flow across SimBio tools, a versatile file format is required which allows a mixed storage of volumetric images and meshes in a single file. This document provides a description of a suitable format and suggests conventions required for a successful application in the target domain.

7.1 Introduction

This document describes file format conventions for SimBio tools. It is assumed to be relevant for SimBio software developers involved in work packages 1 and 3-6. Because SimBio tools will operate on medical data, some additional conventions are required to ensure a reliable use of the tools and to ease communication with users in the target domain.

It is agreed that most SimBio tools use the Vista file format for data storage to provide a maximum compatibility between program modules. Providers of other SimBio tools may convert from their data format to Vista (or vice versa), based on the description in this document. Example Vista data sets conforming to these conventions will be provided on the SimBio website.

Vista was developed as a toolkit for computer vision research by the University of British Columbia in 1994 and released to the public domain [1]. The Vista data format is suggested for the following reasons: The C source code is available [2], well documented, easily portable, tested and stable. The file format is machine-independent, very efficient, versatile and easily extendible. Three SimBio partners have significant amount of experience with using these tools.

A Vista data file can contain a variety of different sorts of objects, images and graphs being the most interesting for SimBio. A file's contents are organised as a list of attributes, each comprising both an attribute name and an attribute value. The attribute name is an alphanumeric string. The attribute value may have one of several forms depending on whether the value is a number, a string, a keyword, a nested list of attributes, an image, or some other object.

7.2 Image Format

The Vista image format is most efficient for storing data defined on regular grids. An entry on this grid is called a voxel and may contain a scalar-, a vector- or tensor-valued quantity in different pre-defined number representations. Image attributes which provide a minimal description of the grid and the data representation are generated automatically by the Vista library routines (and thus called "automatic attributes").

Medical imaging data must be further qualified by additional attributes, which are listed in the section "Compulsory Attributes". Although none of the SimBio tools should rely on the presence of these attributes (i.e., provide reasonable defaults), their inclusion is necessary to allow a reliable use of medical imaging data. Any number of optional attributes may be added. Note that this mechanism may be used to communicate information from one routine to another (such as a transformation matrix, material constants etc.).

7.2.1 Automatic Attributes

The three voxel dimensions of an image are called *band*, *row*, and *column*. The band dimension can serve a variety of purposes, including representing such things as the colour channels of an RGB image, vector- or tensor-valued data. The remaining dimensions, row and column, index pixels by their vertical and horizontal co-ordinates. Vista's convention is to number rows and columns from the upper left pixel, which is at row 0, column 0.

The following Vista attributes are automatically generated when allocating a Vista image structure:

- *data:* an unsigned integer, points to the first byte of the data of this image, as counted from the end of the header.
- *length:* an unsigned integer, denotes the number of image data bytes.
- *nbands:* a non-zero unsigned integer, denotes the total number of 2D slices (the z co-ordinate). If *nbands == 1*, this attribute may be left out.
- *nframes:* a non-zero unsigned integer, denotes the number of 2D slice packages. If voxels represent scalar quantitites, the number of frames equals the number of bands. For vector- or tensor-valued quantities, a set of consecutive bands are collected as a single frame.

  Example: an RGB image is stored as a set of 2D frames, where each frame consists of three slices, representing the red, green and blue components. Thus, *nbands = nframes * 3*.
- *nrows:* a non-zero unsigned integer, denotes image dimension in the y direction.
- *ncolumns:* a non-zero unsigned integer, denotes image dimension in the x direction
- *repn:* a string containing the voxel representation of the image:

`bit`	represents an unsigned integer in the range [0,1]
`ubyte`	represents an unsigned integer in the range [0,255]
`sbyte`	represents a signed integer in the range [-128,127]
`short`	represents a signed integer in 16 bits
`long`	represents a signed integer in 32 bits
`float`	represents a floating point number in 32 bits
`double`	represents a floating point number in 64 bits

- *component_interp:* a string defining the interpretation of a voxel quantity. For SimBio, the following values are of interest:

`gradient`	a single gradient component
`intensity`	a single component reprensenting intensity information
`rgb`	three color channels in the sequence red-green-blue
`complex`	two components (real and imaginary part)
`vector3`	any 3D vector valued component in the sequence x-y-z
`tensor6`	any 3D symmetric tensor valued component in the sequence xx-xy-xz-yy-yz-zz

  If there is only a single band per frame (i.e. *nbands == nframes*), this attribute may be omitted. In this case, *component_interp: intensity* is assumed.

Example: the following header describes an image of dimensions 256 * 256 * 128 voxels in unsigned byte format:

```
      image: image {
     data: 0
     length: 8388608
     nbands: 256
     nframes: 256
     nrows: 256
     ncolumns: 128
     repn: ubyte
      }
```

7.2.2 Compulsory Attributes

The following Vista attributes are compulsory for image data in the SimBio environment. These attributes are required to describe medical image data:

- *voxel:* a string containing three floating point numbers denoting the real world dimensions of a voxel in *mm* along the x-y-z axes.
- *orientation:* a string describing the relation of an image slice (the x-y plane) to the body plane:

`axial`	a slice is perpendicular to the body axis
`coronal`	a slice is perpendicular to the fronto-occipital (i.e., nose-to-back) axis
`sagittal`	a slice is perpendicular to the left-right (i.e., ear-to-ear) axis

For head data sets aligned with the stereotactical co-ordinate system, the orientation is axial by definition.

- *convention:* a string describing the relation of image and body w.r.t. the body symmetry axis:

natural	the left image side corresponds to the left body side
radiologic	the left image side corresponds to the right body side

- *patient:* a string referencing a patient code. Note that the use of patient names is deprecated here. This information is only used to relate image data at a certain processing level to a specific subject.
- *date:* a string containing the date and time of the examination. This information is only used to identify time-series examinations of subjects.

Example: the following header describes an image of dimensions 256 * 256 * 128 voxels in unsigned byte format:

```
 image: image {
...
voxel: "1.5 0.976562 0.976562"
orientation: axial
convention: natural
patient: PS1T000410
date: "11:56:34 10 Apr 2000"
 }
```

7.2.3 Optional Attributes

Any number of optional Vista attributes may be added to further qualify the image. In order to allow compatibility between SimBio tools, their use should be communicated to us and included in a revision of this document.

Note that time-dependent information (such as EEG or MEG data) may also be stored as a Vista image. Most typically, this will be a 2D image, where each row represents a single channel, i.e., the columns represent the time points.

The following optional attributes characterise a Vista signal data set:

- *nChannels:* the number of measurement channels (i.e., rows of the image). The presence of this attribute discriminates images from signal data sets. For signal data sets, the presence of this attribute is considered to be compulsory.
- *sampleInterval:* a number specifying the sampling interval in ms.
- origin: a number specifying the column corresponding to the trigger point.
- *xAxisLabel:* a string specifying the x co-ordinate label (e.g., ms).
- *yAxisLabel:* a string specifying the y co-ordinate label (e.g., μV).
- *chanDDD:* a string containing information regarding channel DDD: electrode label, biosignal type, AD converter range, lower and upper limit of the frequency band.

Example: the following header describes an image of dimensions 256 * 256 * 128 voxels in unsigned byte format:

```
 image: image {
```

```
...
nChannels: 20
sampleInterval: 2
origin: 0
xAxisLabel: ms
yAxisLabel: uV
chan00: " Fp1/G19   EEG   300 0.530 70.000 0.000 0.000"
chan01: " Fp2/G19   EEG   300 0.530 70.000 0.000 0.000"
...
 }
```

7.2.4 Working with Vista Images

This excursion is intended to give a brief overview of some Vista routines to work with the image format. Additional information about Vista images is compiled in the manpage `VImage`.

Images are created with the command *VCreateImage*:

```
      VImage im = VCreateImage(nz, ny, nx, VUByteRepn);
```

where *nx*, *ny* and *nz* denote the image dimensions (note the ordering!) and *VUByteRepn* corresponds to the `ubyte` voxel representation. In analogy, the line:

```
      VDestroyImage(im);
```

releases storage allocated by the previous call. An attribute is added to an existing image using:

```
      char pat[MAX_NAME_LENGTH];

      VSetAttr(VImageAttrList(im),   "patient",   NULL,   VStringRepn,
pat);
```

and, likewise, queried by:

```
      VGetAttr(VImageAttrList(im),   "patient",   NULL,   VStringRepn,
pat);
```

An attribute is destroyed by the following operations:

```
      VAttrListPosn posn;

   if (VLookupAttr(list, "condition", &posn) == True)
            VDeleteAttr(&posn);
```

Note that a newly created image may inherit a complete set of attributes by the call:

```
      VCopyImageAttrs(src, dst);
```

More information about Vista attributes may be found on the manpage `Vattribute`. A specific voxel may be accessed by one of the following mechanisms. A generic function retrieves a value at a given position:

```
      double v = VGetPixel(im, z, y, x);
```

The symmetric call `VSetPixel(im, z, y, x, v)` stores *v* at position *(x, y, z)*. Note that these functions are independent of the image representation, however, some overhead due to the function call must be expected. A more efficient method is to use a macro:

```
      double v = (double)VPixel(im, z, y, x, VUByteRepn);
```

or a pointer:

```
      VUByte ***p = VPixelArray(im, VUByteRepn);
   double v = (double)p[z][y][x];
```

7.3 Graph Format

The Vista graph format may be used to store a general graph structure in a file. It is the preferred format for storing data defined on irregular grids (such as surfaces in 3D or finite element nets). A Vista graph is comprised of a list of nodes and connections between nodes. Nodes and connections may have weights, connections may be uni- or bidirectional in a graph. As for images, a graph may have an arbitrary list of associated attributes. Attributes which provide a minimal description of the graph layout are generated automatically by the Vista library routines (and thus called "automatic attributes").

A node consists of some book-keeping fields and some user-defined fields. Customised node representations are instantiated by subclassing from the structure *VNode.* Two restrictions are enforced for Vista graphs: all nodes in a graph instance must have the same class (i.e., occupy the same amount of storage); all user fields in a node must have the same representation.

A node in a graph may be referenced either by sequencing operations (i.e., iterators), by walking along links, or by directly referencing entries in the node table. Entries in the node table may be empty (e.g., as a result of a node deletion). In order to distinguish between valid and invalid node references, the first entry (at table position 0) is empty by definition, i.e., 0 denotes an invalid or empty node reference.

In a sample application within SimBio, a single graph contains a set of vertices in 3D; links between nodes correspond to edges. Let us call such a structure a *vertex graph*. While this graph is sufficient to represent a surface or volumetric mesh, it is useful to add a second graph containing geometrical primitives. For surface meshes, this *primitive graph* contains polygons as nodes, for volumetric meshes, nodes represent cells (i.e., tetrahedra or hexahedra) of the finite element mesh. Links between nodes denote neighbourhood relationships between primitives: in the case of surfaces, neighbours share edges, in the case of volumetric meshes, neighbours share faces.

The correspondence between images and graphs is given by the following convention: Vertices represent points in the domain, primitives represent some sub-volume of the domain. .Thus, an image may be considered as a mesh containing hexahedral primitives with vertices on the corners of the hexahedra.

A third (optional) graph may be added to include material properties for a volumetric mesh. This graph parallels the primitive graph (i.e. has nodes at the same table positions) and is described below.

7.3.1 Automatic Attributes of General Graphs

Vista graphs have automatic attributes with keywords *data*, *length* and *repn* which have the same meaning as for Vista images. In addition, the following attributes are predefined for any graph:

- *useWeights:* a boolean variable denoting whether the nodes and links in the graph contain additional weight fields. For SimBio application, most likely, weights are not used.
- *nnodes:* an unsigned integer, denotes the number of nodes in a graph.
- *nfields:* a non-zero unsigned integer, denotes the number of fields of representation *repn* in the user portion of the node structure.

7.3.2 Compulsory Attributes of General Graphs

For SimBio applications, it is suggested to include the attributes *patient*, and *date*, as defined for the image format, in all graph structures. The attribute *component_interp* contains a string defining the interpretation of a node. For SimBio, the following values are of interest:

`vertex`	this graph contains vertices
`primitive`	this graph contains primitives
`material`	nodes contain material properties

If there is only a single graph in a file, it is assumed to contain vertices.

7.3.3 Nodes in a Vertex Graph

In order to agree on how user-defined fields in a node are used in SimBio applications, the first field contains a type code. Currently, the following type codes have been defined:

Type	Usage	Field #	Fields
1	simple vertex	4	x y z
2	vertex + normal	7	x y z nx ny nz
3	vertex + normal + curvature	10	x y z nx ny nz cn cg cm
4	vertex + scalar	5	x y z s
5	vertex + normal + scalar	8	x y z nx ny nz s
6	vertex + normal + curvature + scalar	11	x y z nx ny nz cn cg cm s

Note that this table is easily extendible by definition of additional type code and agreement on their usage. Because all fields must have the same representation, the preferred data representation for vertex graphs in SimBio applications is *VFloat.*

It is convenient to have vertex co-ordinates in real world dimensions (i.e., a voxel attribute is not necessary). In addition, if images are to be used together with vertex graphs, it is assumed that they live in the same co-ordinate frame.

7.3.4 Optional Attributes of Vertex Graphs

For SimBio applications, it is necessary to further qualify the information contained in a graph.

- *vertex_interp:* a string qualifying the vertex information. This attribute may indicate that vertices are to be interpreted as electrode or SQUID positions, current or force sources.
  Example: `vertex_interp: electrode`
- *scalar_interp:* a string qualifying the scalar information. This attribute should indicate the meaning and unit of the scalar.
  Example: `scalar_interp: "voltage uV"`

In order to interpret the information correctly (e.g., display electrodes by predefined glyphs), these attributes should be considered as compulsive.

7.3.5 Nodes in a Primitive Graph

A node in a primitive graph represents a geometrical object in a mesh. The first field contains the number of vertices belonging to this primitive, subsequent fields contain the references to the vertices in the table of the
vertex graph.

Example: The primitive node containing the fields `3 57 1 14` denotes a triangle referenced by nodes 57, 1, and 14 of the vertex graph.

It is possible to mix primitives (e.g., triangles and quadrilaterals) in a graph. However, since only a single node type is allowed in a graph, all nodes must provide the same number of fields. Because fields contain references to the vertex graph, the preferred data representation for primitive graphs is *VLong*.

For SimBio applications, surface meshes may contain triangles and quadrilaterals, volumetric meshes may consist of tetrahedra and hexahedra.

7.3.6 Compulsory Attributes of Primitive Graphs

In order to distinguish between surface and volumetric meshes, the presence of following attribute is required:

- *primitive_interp:* a string indicating the mesh type:

`surface`	a surface mesh
`volume`	a volumetric mesh

7.3.7 Nodes in a Material Graph

A third (optional) graph describes material properties for a volumetric mesh. It parallels the primitive graph (i.e. has nodes at the same table positions). Multiple material graphs may be present in a file, i.e. one describing the forces, a second one containing the displacements. In order to agree on how user-defined fields in a node are used in SimBio applications, the first field contains a type code. Currently, the following type codes have been defined:

Type	Usage	Field #	Fields
1	scalar	2	s
2	vector	4	x y z
3	tensor	7	x y z xx yy zz

Note that this table is easily extendible by definition of additional type code and agreement on their usage. Because all fields must have the same representation, the preferred data representation for material graphs in SimBio applications is *VFloat*.

7.3.8 Compulsive Attributes of Material Graphs

For SimBio applications, it is necessary to further qualify the information contained in a material graph.

- *scalar_interp:* a string qualifying the scalar information. This attribute should indicate the meaning and unit of the scalar.
  Example: `scalar_interp: "voltage uV"`
- *vector_interp:* a string qualifying the vector information.
  Example: `vector_interp: "displacement um"`
- *tensor_interp:* a string qualifying the tensor information.

7.3.9 Working with Vista Graphs

As for images, we now give a brief overview of some Vista routines to work with the graph format. Complete information about Vista graphs is compiled in the manpage `VGraph`. Graphs are created with the command *VCreateGraph*:

```
VGraph vtx = VCreateGraph(nnodes, nfields, VFloatRepn, False);
```

where *nnodes* corresponds to the initial table length, *nfields* to the number of fields of representation *VFloat*. The last argument indicates that weights are not used in this graph. Note that the table may grow automatically when nodes are added to the graph. The call `VDestroyGraph(vtx)` releases all storage allocated for this graph. Attributes are added in a similar fashion as described for images:

```
VSetAttr(VGraphAttrList(im), "scalar_interp", NULL, VStringRepn,
"voltage uV");
```

Nodes are added to the graph by the call:

```
VNode node;

unsigned int ref = VGraphAddNode(vtx, node);
```

Note that the information in *node* is copied in the graph, so re-using (or deallocating) *node* is safe. The routine `VGraphAddNode` checks if a node containing the same information was added before using a linear search through the table. For certain situations this performance penalty may be avoided by using the call `VGraphAddNodeAt(vtx, node, ref)`, which places a node at the specified position. Here, the specified position must be within the table. A link is added between two nodes by the command:

```
VGraphAddLink(vtx, ref1, ref2);
```

Note that this link is uni-directonal (i.e., from node *ref1* to *ref2*). The simplest way of traversing a graph is to use an iterator:

```
  for (VNode node = VGraphFirstNode(vtx); node;
node = VGraphNextNode(vtx)) { ....
```

which is equivalent to:

```
      for (unsigned int ref = 1; ref <= VGraphNNodes(vtx); ref++) {
            VNode node = VGraphGetNode(vtx, ref);
            if (node == 0) continue;
            ...
```

An alternative is to walk along the links of a node:

```
      VNode node;

   for (VAdjacency adj = node->adj; adj; adj = adj->next) {
     unsigned int ref = adj->id;
     VNode neighbor = VGraphGetNode(vtx, ref);
     if (neighbor == 0) continue;
     ...
```

7.4 Summary

The Vista toolkit defines a compact, efficient and machine-independent file format, which is suitable to map data structures within the SimBio environment. Additional conventions are described in this document which are necessary for the target application in the medical field.

7.5 References

1. Pope A.R., Lowe D.G. (1994) Vista: A software environment for computer vision research. In: *Computer Vision and Pattern Recognition (CVPR'94)*, pp. 768-772. IEEE Press, Los Alamitos.
2. Pope A.R., Lowe D.G. (1998) The Vista toolkit: http://www.cs.ubc.ca/nest/lci/vista/vista.html.