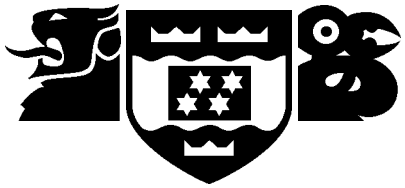
VICTORIA UNIVERSITY OF WELLINGTON



Department of Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 471 5328
Fax: +64 4 495 5232
Internet: Tech.Reports@comp.vuw.ac.nz

Simulating x-ray angiography, a user manual

Peter Hall

Technical Report CS-TR-94/17
October 1994

Abstract

This document describes how to use simulation of x-ray angiography software written by Peter Hall, at the Department of Computer Science, Victoria University of Wellington, Wellington, New Zealand. It does not provide detailed descriptions of the algorithms it uses, nor the internals of the software. Refer to technical reports (Hall 1994a, Hall 1994b, Hall 1994c) for a description of algorithms; there is currently no document that describes the software (C source code).

Author Information

Peter Hall is a lecturer at Victoria University. His principal research interests are scientific visualisation, and computer vision.

Contents

1 An overview of the simulation

The software was designed with a specific aim in mind; a system for evaluating reconstruction of three vasculature, in three dimensions, from x-ray angiograms. Accordingy, the system is a simulation of angiographic procedure. The simulation is divided (rather uncleanly) into three parts. These are:

(1) the vasculature and blood flow through it,
(2) the contrast agent, its injection, and flow though vasculature,
(3) and an x-ray device.

The system accepts descriptions of each part as a text file and yields a set of angiogram files as output. Input is on the standard input stream, output is on the standard output stream. Error messages are delivered to the standard error stream. There are difficulties attendent with this form of input, as described later, but text-file input does tend to make the system more portable - users can add their own 'front end' as desired. Output is in the form of a set of angiograms, currently only Siemen's format is supported.

1.1 Uses of the simulation

The simulation is intended to assist in the development of strategies for three dimensional reconstruction of vasculature from angiogram. Users of the simulation are able to construct arbitrarily complicated networks of *tubes*, so particular vascular structures can be modelled and then varied in terms of both topology and geometry. This should enable the result of a reconstruction to be tested by matching it against the input model. This simulation does not provide such matching functionality.

The flow of blood through the vasculature is (rather crudely) simulated, and is driven by pressure differentials. These pressure diferentials are cyclic when they arise from the beating heart. Blood flow is demonstrated on angiograms by means of an injected contrast agent. The syringe, the contrast agent, and the pressure with which the agent is forced into the blood stream all form part of the model; in this case the pressure differential is transient. In general, pressure differentials are functions of time that are described by a combination of cyclic and transient terms.

Users can model particluar x-rays devices. The x-ray device can be set up with one or more x-ray source and x-ray plate, and these *gantries* can be moved to a new position. This allows angiograms to be acquired from any position, and the relative position between angiograms is well defined (not so with real angiograms).

The x-ray device is capable of producing a single angiogram from each gantry, simultaneously. If the effects blood flow are not being considered then the whole vascular model will be projected inot evert view (though not all of it need be visible). If blood flow is being considered then the x-ray device can output a time-sequenced series of angiograms, from each point of view; time to the first frame and the time between frames are under user control. Given more than one injection event in an angiographic session then all aspects of injection can be modelled - so that *phase difference* between cine angiograms taken at different times becomes apparent.

The simulation does not model vascular lesions (such as *AVM's*) directly, nor does it model patient

movement, nor noise of any kind, nor image warping of the x-ray device,. AVM's can be modelled by a tangled web of vessels, but this is an expensive procedure.

1.2 Command line use

Currently each part is input to the simulator as a text file. I have a UNIX system so I use the simulator from the command line as follows

```
% cat vasculature xray_device injection_event | simulator > angio
```

where

`vasculature` is a file that contains a description of the vascular model to be used,
`xray_device` is a file that contains a description of the x-ray device
`injection_event` is a file that contains a description of the injection event,
`simulator` is the executable version of the software, and
`angio` is the prefix for each angiogram that is output, `angio.1`, `angio.2`, ...

Note that the order of the three input files is important. Modelling schemes and file formats are now discussed in some detail.

2 Vasculature

The modelling scheme for vasculature is first outlined, and the file format is then presented.

2.1 The modelling scheme

A vascular model is constructed from *furcations* and *vessel segments*. Here, a furcation is a place in the vasculature where a vessel segment branches into two or more vessel segments, and vessel segments connect furcations. In addition, furcations are used to limit the extent of the vascular model; that is, furcations may represent places in the model where a vessel segment terminates.

The furcations and vessel segments are assembled in a graph structure, with furcations labelling nodes of the graph and vessel segments labelling arcs. The term vessel segment is used to indicate that they do not necessarily correspond to vessels as recognised in clinical texts (Newton and Potts 1974, Salamon and Huang 1976); such a vessel is a path of edges and nodes through the graph.

Furcations and vessels are themselves constructed from *tubes*. The geometry of a tube is a generalised cylinder; it is the generalised cylinder (see Ballard and Brown, 1982, Foley *et al*. 1990). Here, every generalised cylinders is restricted such that its medial axis is a cubic space curve and it cross section is a circle. Tubes have additional properties, such as conductance to flow. These additional properties are computed internally and are not of concern in this manual. Tubes are the modelling atom so far as vasculature are concerned. The difference between furcations and vessel segments is this:

to make a single vessel segment one or more tubes are placed end-to-end (in series),
to make a single furcation one or more tubes share a single common base (in parallel).

The position, tangent, and radii of a pair of tubes should be equivalent where they meet. Meeting points occur within a vessel segment (as suggested above), with a furcation (as suggested above), and between vessel segments and furcations.

2.1.1 Modelling vasculature using a text file

A problem arises when using text input, this is the difficulty of specifying a three dimensional vasculature with a numbers. Of the many solutions considered (and tried) the following was adopted. When input from a file the simulator will first position and orient each furcation in three dimensional space. Next, each vessel segment is fitted between the furcations that it connects.

Furcations are specified *locally* and then given some *global* position. Local means the positions, say, that are specified relative to a frame of reference fixed to the furcation. Global means position, say, relative to the world frame of reference. Locally, furcations are defined by the end position and end tangent of each tubes. Because all the tubes in a furcation share a common base there need only be n+1 ends specified for n tubes - if the position of the common base is well defined. The position and tangent at the ends of a particular tube are sufficient to define a unique parametric cubic space curve. Given a local definition for the furcation it is placed globally by rotating it around each of the three major axes and then translating the centre of its common base to a specified location in space.

Vessel segments are defined locally and placed globally. The geometry of each tube in a vessel segment is specified by a local position and tangent at each end. Because tubes in a vessel segment

appear in series there need only be n+1 such ends specified to defined n tubes. Because vessel segments are fitted into position between furcations it is, in general, not possible to preserve their shape and size. Placing a vessel occurs in a number of distinct stages. Consider placing a vessel segment between two furcations, the vessel segment connects an end of one the furcations with an end in the other furcation. There is a straight line that joins these ends, and this will be called the *connecting line*.With this definition the steps to place a vessel segment are:

(1) The length of the vessel segment is scaled so that the straight line that joins its extreme ends has the same length as the connecting line. This changes size but ensures zeroth order continuity at both ends.
(2) The vessel is rotated about the connecting line.
(3) The tangents at the end of the vessel are copied from the tangents in the corresponding furcation ends. This changes its shape, but ensures first-order continuity at both ends.
(4) The vessel segment is moved into position.

Notice that continuity of radii is not enforced, this is the responsibility of the user. The syntax of vessel segments in the text file is given below.

The modelling scheme is summarised in figure 1. Notice that the ends of furcations are called *vascular points*, these used to specify injection locations.

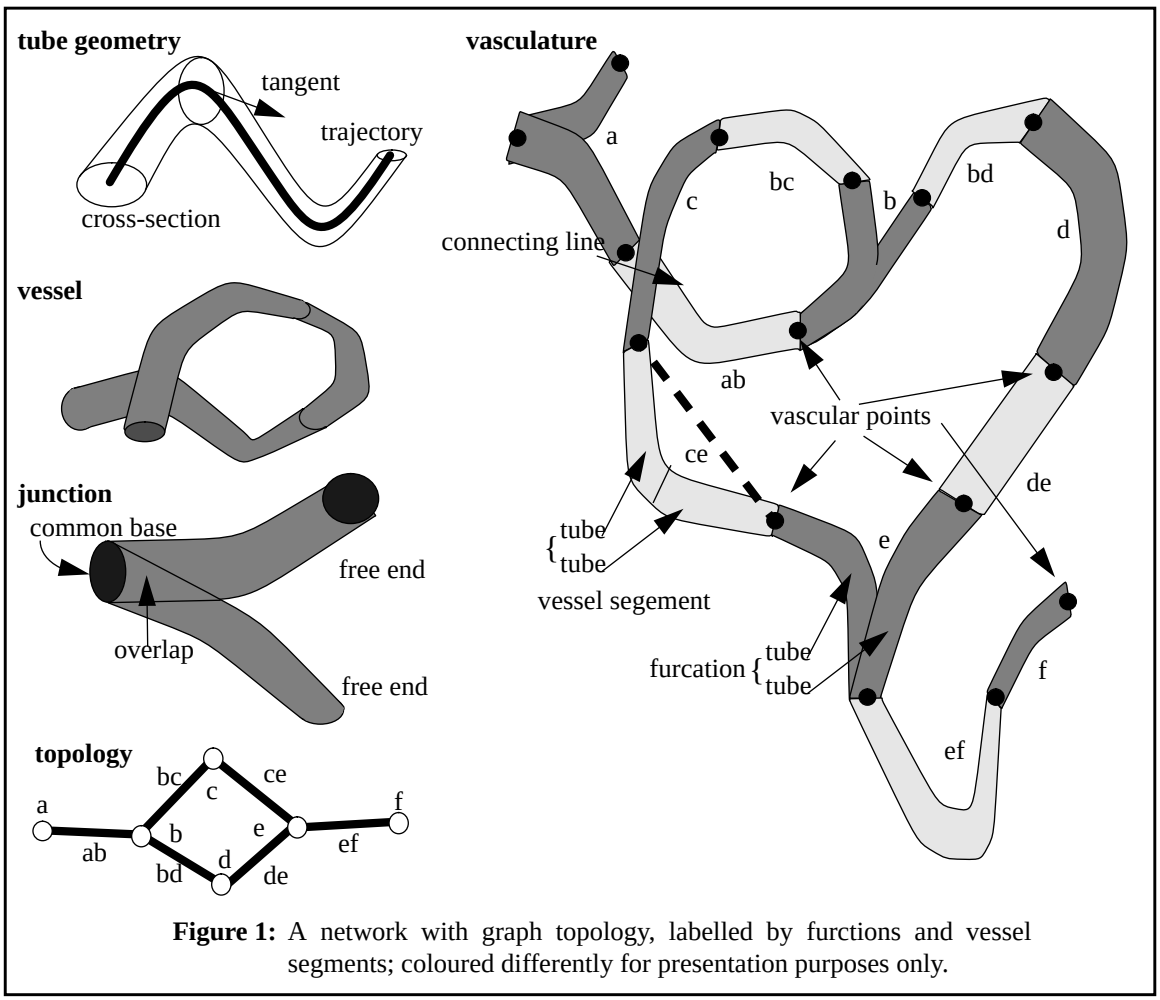


Figure 1: A network with graph topology, labelled by furctions and vessel segments; coloured differently for presentation purposes only.

Vascular points are use to specify injection points and pressure functions. Each vascular point can be identified by an integer pair `(furc_id, channel_id)` in which `furc_id` is the identifier of the furcation the vascular point in. If the vascular point has an abutting vessel segment then `channel_id` is the identifier of the furcation that the vessel segments connects to (so `( furc_id, channel_id )` is an edge). If the vascular point has no abutting vessel segment then the `channel_id` is an invalid identifier.

2.2 Vascular file format

The file format for a vascular model resembles is based on the syntax of a graph G = (V, E), where V is a set of nodes and E is a set of edges. The general form is

```
({nodes}, {edges})
```

where (and) are used to delimit the model, { and } are used to delimit sets, and the comma that separates sets is optional.

2.2.1 Nodes / furcations

Each node has the form

```
[ id:
        text_label
        number_of_ends
        position orientation
        ...
        pstn_i tngnt_i rds_i cnnctn_i
        ...
]
```

The node is delimited by [and], and a colon separates the identifier from the label. Fields in the node are as follows:

`id`	node identifier, it should be any unique, positive integer number.
`text_label`	text string, must contain no spaces, currently unused.

The remaining items in the label specify the geometry of the furcation.

`number_of_ends`	an integer that records the number of ends in the furcation.
`position`	three reals, (x,y,z) say, the global position of the common base
`orientation`	three reals, (ax,ay,az) say, the angle of rotation, in degrees, about the x, y, and z axes respectively, in that order (very important because rotation is non-commutative). The rotations are carried out before the translations to position.

Finally a list that describes each tube, there are `number_of_ends-1` elements in this list. The first element in the list is the common base of the furcation, the remaining elements are the *free-ends* of the furcation. The i^{th} element comprises

$pstn_i$	three reals, (x,y,z) say, the local position of the end
$tngnt_i$	three reals, (u,v,w) say, the local tangent, at the end
rds_i	one real, the radius of cross-section at the end
$cnnctn_i$	an integer to be explain below.

The last item of every element in the list, `cnnct`, is an integer used to resolve an ambiguity. Any node may connect to many other nodes, each connection is via a distinct vessel segment. Within a furcation every end of every tube represents the start, or end, of such a connecting vessel segment, but without the `cnnct` variable it is impossible to associate furcation ends with vessel segments. So, if there is to be a vessel segment connected to a particular furcation end then `cnnct` is the identifier of the node that the end is connected to (by that vessel segment). Of course, a reciprocating identifier exists in the connected-to node. If a particular end of a furcation has no connecting vessel segment then its `cnnct` field will contain a negative integer. This can be useful when constructing vasculature in a text file because it allows users to infer structure that will be put in place - by turning negative `cnnct` fields into positive connect fields. An example of nodes is given in figure 2.

```
[  1 : v1
        3
        10.0 14.0 15.0  45.0 180.0 0.0
        0.0 0.0 0.0     10.0 0.0 0.0    1.9     3
        5.0 5.0 0.0     0.0 20.0 0.0    1.9     8
        3.0 0.0 0.0     5.0 0.0 0.0     0.5     0 ]

[ 3 : v3
        4
        20.0 0.0 0.0    0.0 135.0 150.0
        0.0 0.0 0.0     20.0 0.0 0.0    3.0     10
        -2.0 5.0 0.0    -5.0 5.0 0.0    0.5     5
        2.0 -5.0 0.0    0.0  -5.0 0.0   1.9     1
        5.0 0.0 0.0     5.0 0.0 0.0     2.3     9 ]

[ 8: v8
        2
        9.0 40.0 0.0    0.0 -95.0 -90.0
        0.0 0.0 0.0     15.0 0.0 0.0    1.9     1
        5.0 0.0 0.0     5.0 0.0 0.0     1.9     -2 ]
```

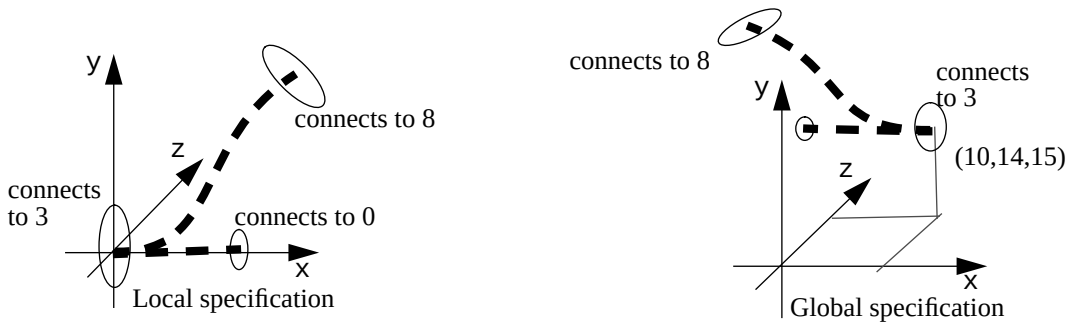


Figure 2: Three nodes. Notice the relationship between the `cnnct` fields; the common base of node 1 is conected to the second free end of node 3. The first free end in node 1 is connected to the common base of node 8. The free end in node 8 has no connecting vessel segment. A picture of how node 1 is specified is shown.

2.2.2 Edges / Vessel segments

Each edge has the form

```
[ id1 id2 :
            text
            n_ends
            angle
            pstn_i      tngnt_i      rds_i
]
```

The edge is delimited by [and], and a colon separates if edge identifier from the edge label. Fields in an edge are as follows:

`id1 id2` a pair of integers the identifier for the edge, the integers should be valid node identifiers - they specify which nodes the edge connects.

`text_label` a string, no spaces, currently unused

The remaining fields describe the geometry of the vessel segment

`n_ends` an integer, the number of ends, `n_ends-1` tubes are in the vessel segment

`angle` a real, the angle of rotation, in degrees, of the vessel segment about its connecting line

A list of ends in the vessel segments follows. There are `n_end` entries, each of the form:

$pstn_i$ three reals, (x,y,z) say, the local position of the end

$tngnt_i$ three reals, (u,v,w) say, the local tangent, at the end

rds_i one real, the radius of cross-section at the end

Recall that this definition is local; it is scaled and moved, and tangents at extreme ends ($tngnt_0$ and $tngnt_{n\_end-1}$) are changed. So, within a vessel segment specification positions represent fractional movements over the connecting line, and tangents give hints at shape.

To reiterate, this modelling scheme is not entirely satisfactory. It represents a balance between ease of definition (so that parts can be moved) and enforcing sensible continuity conditions between modelling parts (especially where a furcation and vessel segment abut). Given an interactive modeller with a graphics interface such problems will be invisible to the user.

The specification of a complete vascular is presented in figure 3

```
( {      [ 4 :      v4
                    2
                    -70.0 0.0 0.0   0.0     0.0     0.0
                    0.0 0.0 0.0     10.0    0.0     0.0     4.0     -1
                    10.0 0.0 0.0    10.0    0.0     0.0     4.0      0 ]

         [ 0 :      v0
                    3
                    -50.0 0.0 0.0   0.0     0.0     0.0
                    0.0 0.0 0.0     40.0    0.0     0.0     4.0     4
                    20.0 40.0 0.0   40.0    0.0     0.0     4.0     1
                    20.0 0.0 0.0    40.0    0.0     0.0     4.0     2 ]

         [ 1 :      v1
                    2
                    -10.0 40.0 0.0  0.0     0.0     0.0
                    0.0 0.0 0.0     20.0    0.0     0.0     4.0     0
                    0.0 0.0 0.0    20.0    0.0     0.0     4.0     3 ]

         [ 2 :      v2
                    2
                    10.0 0.0 0.0    0.0     0.0     180.0
                    0.0 0.0 0.0     20.0    0.0     0.0     4.0     3
                    20.0 0.0 0.0    20.0    0.0     0.0     4.0     0 ]

         [ 3 :      v3
                    3
                    50.0 0.0 0.0    0.0     180.0   0.0
                    0.0 0.0 0.0     40.0    0.0     0.0     4.0     5
                    20.0 40.0 0.0   40.0    0.0     0.0     4.0     1
                    20.0 0.0 0.0    40.0    0.0     0.0     4.0     2 ]

         [ 5 :      v5
                    2
                    60.0 0.0 0.0    0.0     0.0     0.0
                    0.0 0.0 0.0     10.0    0.0     0.0     4.0     3
                    10.0 0.0 0.0    10.0    0.0     0.0     4.0      -1 ]

},{

         [ 0 1: e0-1
                    2
                    0.0
                    0.0 0.0 0.0     10.0 0.0 0.0    4.0
                    10.0 0.0 0.0    10.0 0.0 0.0    4.0 ]

         [ 0 2 : e0-3
                    2
                    0.0
                    0.0 0.0 0.0     10.0 0.0 0.0    4.0
                    10.0 0.0 0.0    10.0 0.0 0.0    4.0 ]

         [ 1 3 : e1-3
                    2
                    0.0
                    0.0 0.0 0.0     10.0 0.0 0.0    4.0
                    10.0 0.0 0.0    10.0 0.0 0.0    4.0 ]

         [ 2 3 : e2-3
                    2
                    0.0
                    0.0 0.0 0.0     10.0 0.0 0.0    4.0
                    10.0 0.0 0.0    10.0 0.0 0.0    4.0 ]

         [  4 0 : e4-0
                    2
                    0.0
                    0.0 0.0 0.0     10.0 0.0 0.0    4.0
                    10.0 0.0 0.0    10.0 0.0 0.0    4.0 ]

         [  3 5 : e3-5
                    2
                    0.0
                    0.0 0.0 0.0     10.0 0.0 0.0    4.0
                    10.0 0.0 0.0    10.0 0.0 0.0    4.0 ]

})
```

Figure 3: A complete vascular specifiation.

3 X-ray device

3.1 X-ray device modelling scheme

The x-ray device consists of one or more gantries, as in figure 4. Each gantry consists of an x-ray source and an image plane. An x-ray source is modelled as a point emitter, radiating isotropically. The image plane has particular position, orientation, and size. The image plane square, and is decomposed into pixels, 512 pixels along each edge.

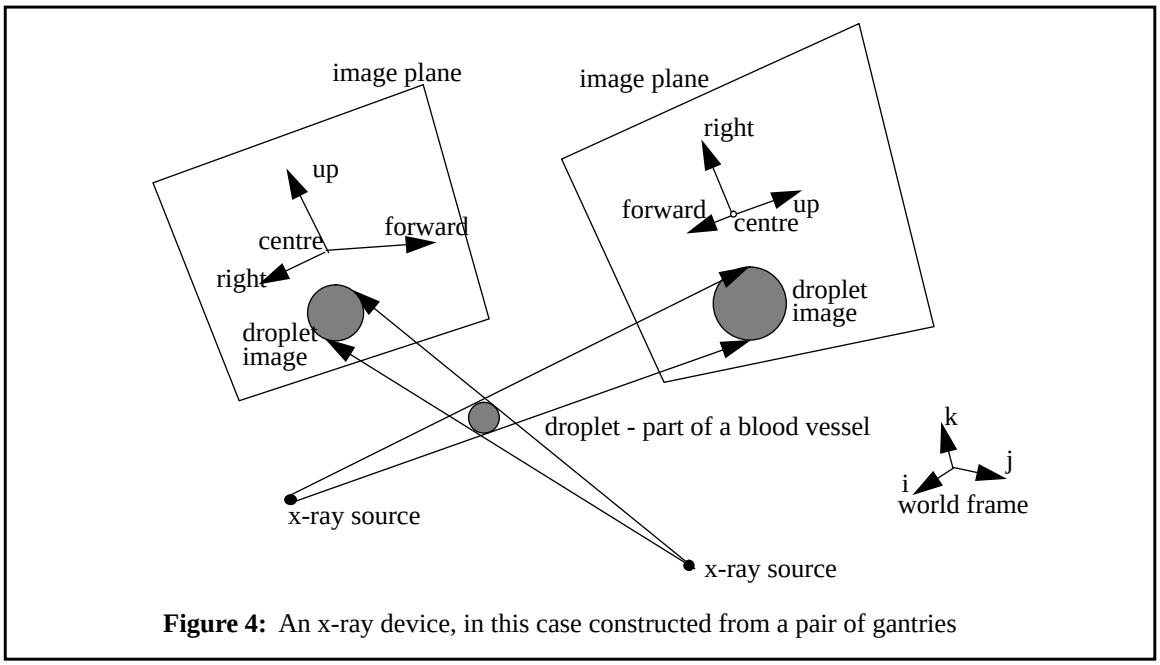


Figure 4: An x-ray device, in this case constructed from a pair of gantries

The geometry of each gantry is fixed, and the geometrical relation between gantries is also fixed. Should the x-ray device be moved then each of gantries moves also; the x-ray device is a rigid object. The x-ray device simultaneously acquires images from each gantry.

3.2 X-ray device file format

The file format for an x-ray device is as follows

`n_gantries` n integer specifying the number of gantries

For each gantry the following information is supplied

$\texttt{pstn}_{\texttt{src}}$	three reals, (x_0,y_0,z_0) say, the position of the x-ray source in the world
$\texttt{pstn}_{\texttt{pln}}$	three reals, (x,y,z) say, the position of the centre of the x-ray plane
$\texttt{nrml}_{\texttt{pln}}$	three reals, (a,b,c) say, a vector normal to the x-ray plane
$\texttt{up}_{\texttt{pln}}$	three reals, (u_u,v_u,w_u) say, a vector specifying ‘up’ in the x-ray plane,
$\texttt{rght}_{\texttt{pln}}$	three reals, (u_r,v_r,w_r) say, a vector specifying ‘right’ in the x-ray plane
`vendor`	integer the manufacture format (only Siemens is supported, vendor = 1)
`dtime`	one real, the time between frames in a cine-angiogram
intensity	one real, the intensity of the x-ray source

```
3

0.0 0.0 -1000.0
-256.0 256.0 1000.0
0.0 0.0 1.0
0.0 -1.0 0.0
1.0 0.0 0.0

-1000.0 0.0 0.0
1000.0 256.0 256.0
1.0 0.0 0.
0.0 -1.0 0.0
0.0 0.0 -1.0

0.0 -1000.0 0.0
-256.0 1000.0 256.0
0.0 -1.0 0.0
0.0 0.0 1.0
1.0 0.0 0.0

1

2.0

1023.0
```

Figure 5: An example input file that specifies an x-ray device.

4 Injection events / Acquisition file

Injection event specification appears in two flavours, the *static* and the *dynamic*. These are differentiated by a field at the very top of the input file. In the static case the blood flow is 'frozen' in the vasculature. Only one angiogram is acquired from each point of view (gantry), any part of the vasculature that is visible will appear in at lest one angiogram.

In the dynamic case blood flow through the vasculature is simulated; flow is driven by pressure differentials. Then, a series of time-sequence angiograms will be acquired from each distinct point of view, not all of the potentially visible vasculature will appear in each time instant. In addition, the number of injection events must be specified, and the x-ray device is moved into a new position for each distinct injection event. Because the blood flow is driven by pressure differentials the user must input time-varying pressure functions. These functions comprise a non-cyclic (non-periodic / transient) part, and a cyclic (periodic part). Each function part is specified by a series sample of pressure values that are distributed at regular intervals in time. The number of samples and the time duration of the function part must be specified by the user. Functions parts are additively combined to produce the final function.

All function parts begin at time zero. The injection of agent and the time at which the first frame is acquired can be any time after that. Usually, the user will want only one function with a transient part - this will represent the pressure profile caused by injecting contrast agent. hence the time delay to injection will then be zero. The cyclic parts of pressure functions are intended to represent heart beat.

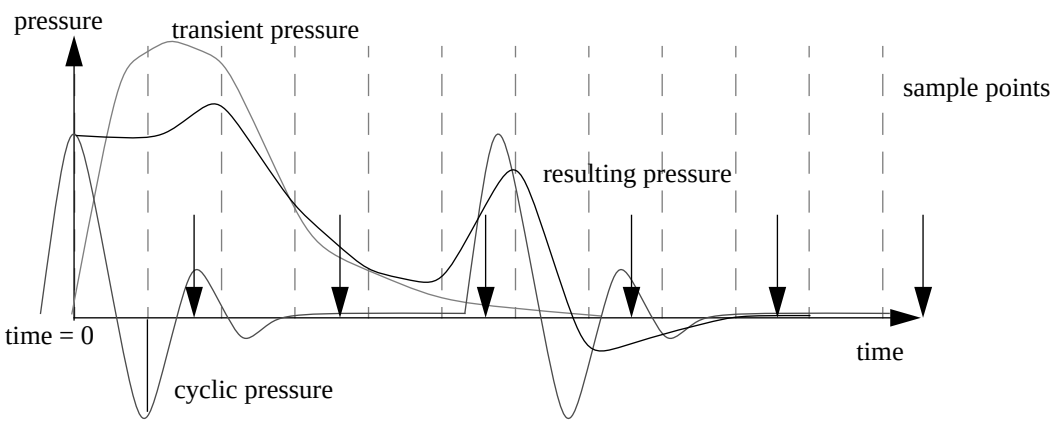


Figure 6: A pressure function composed from transient and cyclic parts. Here, each part is sampled at the same rate, for simplicity. Arrows show times at which frames are captured.

File formats are now discussed.

4.1 Static acquisition file format

The fields in the input file for static acquisition as follows:

`0`	integer zero - a flag to identify static acquisition
`conc`	a real, the concentration of the contrast agent (number of particles per unit volume)
`map`	a 4x4 matrix of reals, a map that moves the x-ray device into place
`atten_coeff`	a real, the x-ray attenuation coefficient per unit distance at unit through contrast agent at concentration.

The path-length is the distance that a x-ray travels though some medium such that its intensity is reduced to a fraction 1/e of its original value. This distance is related to the attenuation coefficient and concentration.

4.2 Dynamic acquisition file format

The fields in the input file for dynamic acquisition as follows:

`1`	integer one- a flag to identify dynamic acquisition
`n_event`	the number of acquisition events
`injection_event`	a list of n_event injection events, details explained below
`atten_coeff`	a real, the x-ray attenuation coefficient per unit distance at unit through contrast agent at unit concentration.

I like to use shell scripts to specify files for dynamic acquisition, figure 6 provides an example.

```
# the anim_flag: 1 = animate, 0 =static

        echo “1”

# the number of acquisition events

        echo “2”
         injection1.build
         injection2.build

# xray attenuattion coefficient - higher values mean darker images

        echo “0.05”
```

Figure 7: A shell script acquistion file.

In figure 6 the files `injection1.build` and `injection2.build` are shell scripts that specify particular injection events. These correspond to the `injecton_event` field mentioned above. The format of these files is as follows:

the first few fields describe the ‘syringe’

`what`	an integer, what is being injected, 1 = furcation, 2 = vessel segment
`where`	two integers, specifying a vascular point
`distance`	a real, the distance from the vascular point, currently unused
`volume`	a real, the amount of contrast agent being injected
`conc`	a real, concentration of contrast agent (particles per unit volume)
$\texttt{time}_0$	a non-negative real, the time after time zero the injection occurs
`dtime`	a real, the length of time the injection lasts

the next two fields describe the x-ray device for a particular injection event

`map`	a 4x4 matrix of reals, a map to move the x-ray gantry into place
$\texttt{time}_1$	a non-negative real, time delay to first frame from time zero

pressure functions are needed to drive the blood flow

`n_func`	the number of pressure functions

A list of pressure function specifications follow. For each pressure function input:

$\texttt{where}_i$	two integers, specifying a vascular point
`ncyc_dt`	a real, the non-cyclic time duration of the pressure function
`ncyc_pnt`	an integer, the number of points defining the non-cyclic pressure function
`ncyc_val`	`n_cyc` reals, each a pressure value (regularly distributed samples)
`cyc_dt`	a real, the cyclic time duration of the pressure function
`cyc_pnt`	an integer, the number of points defining the cyclic pressure function
`cyc_val`	`n_cyc` reals, each a pressure value (regularly distributed samples)

Finally, the least field is

`n_frame`	an integer, the maximum number of frames to be acquired from any point of view

An example shell script is shown in figure 7.

```
#!/bin/csh

### THE SYRINGE ###

# what is being injected? 1 = furcation, 2 = vessel
        echo “1”
# where the injection occurs, the vascular point id
        echo “10 -1”
# distance from vascular point (currently redundant)
        echo “0.0”
# volume, concentration, time of injection, duration
        echo “20000.0 1.0 0.7 0.8”

### THE X-RAY DEVICE ###

# move gantry
        map44_id   # this is a command level function
# set time to first frame (from start of injection)
        echo “0.5”

### PRESSURE FUNCTIONS ###
# how many pressure points are given?
        echo “3”
# the place in the vasculature that is fixed
# a non-cyclic and cylcic part of the function each comprising
#    the time duration of the function
#    the number of points in the pressure function at the point
#    the pressure function

# the place:  end of  int-carotid-left
        echo “10 -1”
# non_cyclic duration
        echo “4.0”
# number of points in non_cylic function
        echo “2”
# the non-cyclic point values”
        echo “0.0 0.0”
# cyclic duration
        echo “4.0”
# number of points in cylic function
        echo “5”
# the cyclic point values”
        echo “100.0 100.0 100.0 100.0 100.0”

# fix pressure at end of basilar
        echo “12 -1”
        echo “4.0 2 0.0 0.0”
        echo “4.0 5 100.0 100.0 100.0 100.0 100.0”

# fix pressure at end of int-carotid-right
        echo “14 -1”
        echo “4.0 2 0.0 0.0”
        echo “4.0 5 100.0 100.0 100.0 100.0 100.0”

# the maximum number of frames for the injection event
        echo “10”
```

Figure 8: A shell script specifying a particular injection event.

5 Using the system as a reconstruction test bed

The simulator should provide sufficient versatility to be used as a test bed for reconstruction, despite the fact its output is limited to Siemen's format angiograms. The static images are intended to permit investigation of reconstruction without worrying about temporal aspects such as phase difference between angiograms from different points of view; cine-angiograms can be acquired with complete phase matching between angiograms from different points of view. The user is invited to extend the range of outputs possible to suit themselves.

6 Errors

At this stage I decline to present extensive documentation on error messages. The simulator attempts to detect errors in the input file and give a helpful message should the need arise. An error-free input should result in an error-free running of the program (no guarantees, especially if the source has been modified by the user). Any problems should be reported to the author of the software (if satisfied tell others, if not tell us).

Other errors include, for example, the density profile of a cross-section; such errors represent algorithmic problems that will require further research.

7 Conclusion

I hope the simulator is useful to you. Please report any errors as they are found, including ant erros or deficiencies in this document. If you should upgrade or extend the functionality of the system then please report that too; the success of this shareware depends upon it.

8 References

Ballard, D.H., and Brown, C.M. (1982) Computer Vision. *Prentice-Hall, Englewood Cliffs, NJ, USA.*

Foley, J., van Dam, A., Feiner, S., and Hughes, J. (1990) Computer graphics principles and practice, second edition. *Addison-Wesley, Reading, MA, USA.*

Hall, P.M. (1994a) Implicit volume rendering for generalised cylinders. *CS-TR-94-10*

Hall, P.M. (1994b) Simulating and animating flow through a network of tubes, *CS-TR-94-13*

Hall, P.M. (1994c) Simulating angiography, *CS-TR-94-15*

Newton, T.H., and Potts D.G. (1974) Radiology of the skull and brain: V*olume 2 / Book 2. Mosby, Saint Louis*

Salamon, G. and Huang, Y.P. (1976) A radiological anatomy of the brain. *Springer-Verlag, Berlin*