# GTV Reference Manual

Generated by Doxygen 1.5.2

Mon Feb 4 13:50:03 2008

# Contents

# Chapter 1

# GTV Module Index

## 1.1 GTV Modules

Here is a list of all modules:

# Chapter 2

# GTV Data Structure Index

## 2.1 GTV Data Structures

Here are the data structures with brief descriptions:

# Chapter 3

# GTV Module Documentation

## 3.1 Cell

### Data Structures

- struct **GtvCell**
- struct **GtvCellClass**

### Functions

- **GtvCellClass** ∗ **gtv_cell_class** (void)
- **GtvCell** ∗ **gtv_cell_new** (**GtvCellClass** ∗klass, **GtvFacet** ∗f1, **GtvFacet** ∗f2, **GtvFacet** ∗f3, **GtvFacet** ∗f4)
- GSList ∗ **gtv_cell_neighbours** (**GtvCell** ∗c, **GtvVolume** ∗v)
- **GtvCell** ∗ **gtv_cell_new_from_vertices** (**GtvCellClass** ∗klass, **GtvFacetClass** ∗facet_class, GtsEdgeClass ∗edge_class, GtsVertex ∗v1, GtsVertex ∗v2, GtsVertex ∗v3, GtsVertex ∗v4)
- GSList ∗ **gtv_edge_cells** (GtsEdge ∗e, **GtvVolume** ∗v)
- GSList ∗ **gtv_vertex_cells** (GtsVertex ∗p, **GtvVolume** ∗v)

### 3.1.1 Function Documentation

#### 3.1.1.1 GtvCellClass∗ gtv_cell_class (void)

The basic class for cells (volume elements) in GTV.

**Returns:**

the **GtvCellClass** (p. 34)

#### 3.1.1.2 GSList∗ gtv_cell_neighbours (GtvCell ∗ *c*, GtvVolume ∗ *v*)

Find the neighbours of a **GtvCell** (p. 33).

**Parameters:**

***c*** a **GtvCell** (p. 33);

*v* a **GtvVolume** (p. 39), or NULL.

**Returns:**

a GSList of the cells of *v*, or any cells if *v* is NULL, which share a facet with *c*.

#### 3.1.1.3 GtvCell∗ gtv_cell_new (GtvCellClass ∗ *klass*, GtvFacet ∗ *f1*, GtvFacet ∗ *f2*, GtvFacet ∗ *f3*, GtvFacet ∗ *f4*)

Make a new **GtvCell** (p. 33) from four facets which must define a proper tetrahedron.

**Parameters:**

*klass* a **GtvCellClass** (p. 34);

*f1* a **GtvFacet** (p. 35);

*f2* another **GtvFacet** (p. 35);

*f3* a further **GtvFacet** (p. 35);

*f4* one more **GtvFacet** (p. 35);

**Returns:**

the new **GtvCell** (p. 33).

#### 3.1.1.4 GtvCell∗ gtv_cell_new_from_vertices (GtvCellClass ∗ *klass*, GtvFacetClass ∗ *facet_class*, GtsEdgeClass ∗ *edge_class*, GtsVertex ∗ *v1*, GtsVertex ∗ *v2*, GtsVertex ∗ *v3*, GtsVertex ∗ *v4*)

Generate a new cell from four GtsVertex's, making use of any existing **GtvFacet** (p. 35)'s which connect them.

**Parameters:**

*klass* a **GtvCellClass** (p. 34)

*facet_class* a **GtvFacetClass** (p. 36)

*edge_class* a GtsEdgeClass

*v1* a GtsVertex

*v2* a GtsVertex

*v3* a GtsVertex

*v4* a GtsVertex

**Returns:**

the new **GtvTetrahedron** (p. 37)

#### 3.1.1.5 GSList∗ gtv_edge_cells (GtsEdge ∗ *e*, GtvVolume ∗ *v*)

Find the **GtvCell** (p. 33)'s which use a given GtsEdge.

**Parameters:**

*e* a GtsEdge

***v*** a **GtvVolume** (p. 39)

**Returns:**

a GSList of the **GtvCell** (p. 33)'s of *v* which contain *e*, NULL if *e* is not used by any cells.

#### 3.1.1.6 GSList∗ gtv_vertex_cells (GtsVertex ∗ *p*, GtvVolume ∗ *v*)

Find the **GtvCell** (p. 33)'s which use a given GtsVertex

**Parameters:**

***p*** a GtsVertex

***v*** a **GtvVolume** (p. 39)

**Returns:**

a GSList of the **GtvCell** (p. 33)'s of *v* which contain *e*, NULL if *e* is not used by any cells.

# 3.2 Delaunay tetrahedralization

## Functions

- **GtvCell** ∗ **gtv_delaunay_check** (**GtvVolume** ∗v)
- gint **gtv_delaunay_add_vertex_to_cell** (**GtvVolume** ∗v, GtsVertex ∗p, **GtvCell** ∗c)
- gboolean **gtv_facet_is_regular** (**GtvFacet** ∗f)
- gint **gtv_delaunay_remove_vertex** (**GtvVolume** ∗v, GtsVertex ∗p)

## 3.2.1 Function Documentation

### 3.2.1.1 gint gtv_delaunay_add_vertex_to_cell (GtvVolume ∗ *v*, GtsVertex ∗ *p*, GtvCell ∗ *c*)

Add a GtsVertex to a **GtvCell** (p. 33) of a **GtvVolume** (p. 39), restoring the Delaunay property of the volume, using the method of Edelsbrunner, H. and Shah, N. R., Algorithmica 15:223–241, 1996 and Lawson, C., Computer Aided Geometric Design, 3:231–246, 1986, as described in Ledoux, H., ‘Computing the 3D Voronoi diagram robustly: An easy explanation’, 4th International Symposium on Voronoi Diagrams in Science and Engineering, 2007.

**Parameters:**

***v*** a **GtvVolume** (p. 39);

***p*** a GtsVertex to add to *p*;

***c*** a **GtvCell** (p. 33) to add *p* to.

**Returns:**

**GTV_SUCCESS** (p. 15) on success, non-zero if *p* is already part of *c* or coincides with a vertex of *c*.

### 3.2.1.2 GtvCell∗ gtv_delaunay_check (GtvVolume ∗ *v*)

Check whether a **GtvVolume** (p. 39) satisfies the Delaunay property.

**Parameters:**

***v*** volume to check

**Returns:**

a non-Delaunay **GtvCell** (p. 33) of *v* if *v* is non-Delaunay, NULL otherwise.

### 3.2.1.3 gint gtv_delaunay_remove_vertex (GtvVolume ∗ *v*, GtsVertex ∗ *p*)

Remove a GtsVertex from a **GtvVolume** (p. 39) and restore the Delaunay property.

**Parameters:**

***v*** a **GtvVolume** (p. 39);

***p*** a GtsVertex.

**Returns:**

**GTV_SUCCESS** (p. 15) if *p* has been successfully removed.

#### 3.2.1.4 gboolean gtv_facet_is_regular (GtvFacet ∗ *f*)

Check if a **GtvFacet** (p. 35) is regular. A facet is regular if neither of the tetrahedra using it has the apex of the opposite tetrahedron inside its circumsphere. Boundary facets and isolated facets are also considered regular.

**Parameters:**

*f* a **GtvFacet** (p. 35).

**Returns:**

TRUE if *f* is regular, FALSE otherwise.

## 3.3 Facets

### Data Structures

- struct **GtvFacet**
- struct **GtvFacetClass**

### Functions

- **GtvFacet** ∗ **gtv_facet_new** (**GtvFacetClass** ∗klass, GtsEdge ∗e1, GtsEdge ∗e2, GtsEdge ∗e3)
- **GtvCell** ∗ **gtv_facet_is_boundary** (**GtvFacet** ∗f, **GtvVolume** ∗v)
- gint **gtv_facet_tetrahedra_vertices** (**GtvFacet** ∗f, GtsVertex ∗∗v1, GtsVertex ∗∗v2, GtsVertex ∗∗v3, GtsVertex ∗∗v4, GtsVertex ∗∗v5)
- gint **gtv_facet_tetrahedra** (**GtvFacet** ∗f, **GtvTetrahedron** ∗∗t1, **GtvTetrahedron** ∗∗t2)
- **GtvFacet** ∗ **gtv_facet_from_vertices** (GtsVertex ∗v1, GtsVertex ∗v2, GtsVertex ∗v3)
- gboolean **gtv_facet_has_vertex** (**GtvFacet** ∗f, GtsVertex ∗v)
- gboolean **gtv_facet_has_edge** (**GtvFacet** ∗f, GtsEdge ∗e)
- **GtvCell** ∗ **gtv_edge_is_boundary** (GtsEdge ∗e, **GtvVolume** ∗v)

### 3.3.1 Function Documentation

#### 3.3.1.1 GtvCell∗ gtv_edge_is_boundary (GtsEdge ∗ *e*, GtvVolume ∗ *v*)

Check if a GtsEdge lies on the boundary of a **GtvVolume** (p. 39).

**Parameters:**

*e* a GtsEdge;

*v* a **GtvVolume** (p. 39) or NULL.

**Returns:**

a **GtvCell** (p. 33) which uses *e* and is on the boundary of *v*.

#### 3.3.1.2 GtvFacet∗ gtv_facet_from_vertices (GtsVertex ∗ *v1*, GtsVertex ∗ *v2*, GtsVertex ∗ *v3*)

Find a **GtvFacet** (p. 35) which uses three given GtsVertex's.

**Parameters:**

*v1* a GtsVertex;

*v2* a GtsVertex;

*v3* a GtsVertex.

**Returns:**

a **GtvFacet** (p. 35) which uses *v1*, *v2* and *v3*, if it exists, NULL otherwise.

#### 3.3.1.3 gboolean gtv_facet_has_edge (GtvFacet ∗ *f*, GtsEdge ∗ *e*)

Check if a **GtvFacet** (p. 35) uses a GtsEdge.

**Parameters:**

*f* a **GtvFacet** (p. 35);
*e* a GtsEdge;

**Returns:**

TRUE if *f* use *e*, FALSE otherwise.

#### 3.3.1.4 gboolean gtv_facet_has_vertex (GtvFacet ∗ *f*, GtsVertex ∗ *v*)

Check if a **GtvFacet** (p. 35) uses a GtsVertex.

**Parameters:**

*f* a **GtvFacet** (p. 35);
*v* a GtsVertex;

**Returns:**

TRUE if *f* use *v*, FALSE otherwise.

#### 3.3.1.5 GtvCell∗ gtv_facet_is_boundary (GtvFacet ∗ *f*, GtvVolume ∗ *v*)

Check if a **GtvFacet** (p. 35) lies on the boundary of a **GtvVolume** (p. 39).

**Parameters:**

*f* a **GtvFacet** (p. 35);
*v* a **GtvVolume** (p. 39) or NULL.

**Returns:**

a **GtvCell** (p. 33) which uses *f* and is on the boundary of *v*.

#### 3.3.1.6 GtvFacet∗ gtv_facet_new (GtvFacetClass ∗ *klass*, GtsEdge ∗ *e1*, GtsEdge ∗ *e2*, GtsEdge ∗ *e3*)

Make a new **GtvFacet** (p. 35) from three edges which must define a proper triangle.

**Parameters:**

*klass* a **GtvFacetClass** (p. 36);
*e1* a GtsEdge;
*e2* a GtsEdge;
*e3* a GtsEdge.

**Returns:**

a new **GtvFacet** (p. 35) or NULL if the edges do not define a valid triangle.

#### 3.3.1.7 gint gtv_facet_tetrahedra (GtvFacet $*$ *f*, GtvTetrahedron $**$ *t1*, GtvTetrahedron $**$ *t2*)

Find the tetrahedra which use a **GtvFacet** (p. 35).

**Parameters:**

*f* a **GtvFacet** (p. 35)

*t1* a **GtvTetrahedron** (p. 37) which uses *f* or NULL if *f* is isolated;

*t2* a **GtvTetrahedron** (p. 37) which uses *f* or NULL if *f* is a boundary facet.

**Returns:**

GTV_SUCCESS on success.

#### 3.3.1.8 gint gtv_facet_tetrahedra_vertices (GtvFacet $*$ *f*, GtsVertex $**$ *v1*, GtsVertex $**$ *v2*, GtsVertex $**$ *v3*, GtsVertex $**$ *v4*, GtsVertex $**$ *v5*)

Find the vertices of the tetrahedra sharing a particular facet. If *f* is used by only one tetrahedron, v5 will be NULL. If *f* is an isolated facet, i.e. not used by any tetrahedra, all the vertices will be NULL.

**Parameters:**

*f* a **GtvFacet** (p. 35)

*v1* a vertex of a tetrahedron using *f*;

*v2* a vertex of a tetrahedron using *f*;

*v3* a vertex of a tetrahedron using *f*;

*v4* a vertex of a tetrahedron using *f*;

*v5* a vertex of a tetrahedron using *f*.

**Returns:**

GTV_SUCCESS on success.

## 3.4 Geometric tests

### Functions

- gdouble **gtv_point_in_sphere** (GtsPoint ∗p, GtsPoint ∗p1, GtsPoint ∗p2, GtsPoint ∗p3, GtsPoint ∗p4)
- gboolean **gtv_points_are_collinear** (GtsPoint ∗p1, GtsPoint ∗p2, GtsPoint ∗p3)

### 3.4.1 Function Documentation

#### 3.4.1.1 gdouble gtv_point_in_sphere (GtsPoint ∗ *p*, GtsPoint ∗ *p1*, GtsPoint ∗ *p2*, GtsPoint ∗ *p3*, GtsPoint ∗ *p4*)

Check whether a point lies inside a sphere defined by four GtsPoint's.

**Parameters:**

***p*** a GtsPoint;

***p1*** a GtsPoint;

***p2*** a GtsPoint;

***p3*** a GtsPoint;

***p4*** a GtsPoint;

**Returns:**

positive value if *p* lies inside the sphere, zero if it lies on the sphere and a negative value if it lies outside.

#### 3.4.1.2 gboolean gtv_points_are_collinear (GtsPoint ∗ *p1*, GtsPoint ∗ *p2*, GtsPoint ∗ *p3*)

Check if three points are collinear by checking their orientation in three dimensions.

**Parameters:**

***p1*** a GtsPoint;

***p2*** a GtsPoint;

***p3*** a GtsPoint.

**Returns:**

TRUE if *p1*, *p2* and *p3* are collinear, FALSE otherswise.

# 3.5 Logging functions

## Functions

- gint **gtv_logging_init** (FILE ∗f, gchar ∗p, GLogLevelFlags log_level, gpointer exit_func)

## 3.5.1 Function Documentation

### 3.5.1.1 gint gtv_logging_init (FILE ∗ *f*, gchar ∗ *p*, GLogLevelFlags *log_level*, gpointer *exit_func*)

Initialize GTV logging

**Parameters:**

***f*** file stream for messages

***p*** string to prepend to messages

***log_level*** maximum logging level to handle (see g_log)

***exit_func*** function to call if exiting on an error

**Returns:**

GTV_SUCCESS on success

## 3.6 Status codes

### Enumerations

- enum **GtvStatus** {

  **GTV_FAILURE** = -1, **GTV_SUCCESS** = 0, **GTV_NULL_PARAMETER** = 1, **GTV_WRONG_TYPE** = 2,

  **GTV_VERTEX_PRESENT** = 3, **GTV_COINCIDENT_VERTEX** = 4, **GTV_VERTEX_NOT_IN_CELL** = 5, **GTV_REPEATED_PARAMETER** = 6,

  **GTV_VERTEX_ON_HULL** = 7 }
- enum **GtvIntersect** { ,

  **GTV_ON** = 0, **GTV_IN** = 1, **GTV_ON_FACET** = 2, **GTV_ON_EDGE** = 3,

  **GTV_ON_VERTEX** = 4 }

### 3.6.1 Enumeration Type Documentation

#### 3.6.1.1 enum GtvIntersect

Status codes returned by GTV functions which check if a vertex inside, outside or on the surface of a simplex, e.g. a tetrahedron.

**Enumerator:**

***GTV_ON*** vertex lies outside tetrahedron

***GTV_IN*** vertex lies on tetrahedron surface

***GTV_ON_FACET*** vertex lies strictly inside tetrahedron

***GTV_ON_EDGE*** vertex lies on a facet of tetrahedron

***GTV_ON_VERTEX*** vertex lies on an edge of tetrahedron

#### 3.6.1.2 enum GtvStatus

Status codes returned by GTV functions.

**Enumerator:**

***GTV_FAILURE*** unspecified failure

***GTV_SUCCESS*** function succeeded

***GTV_NULL_PARAMETER*** a parameter was NULL

***GTV_WRONG_TYPE*** a parameter was of the wrong type

***GTV_VERTEX_PRESENT*** vertex already present in tetrahedralization

***GTV_COINCIDENT_VERTEX*** vertex in tetrahedralization has the same coordinates

***GTV_VERTEX_NOT_IN_CELL*** the vertex to be added is not inside the cell

***GTV_REPEATED_PARAMETER*** two or more input parameters are identical

***GTV_VERTEX_ON_HULL*** vertex lies on the convex hull of a tetrahedralization

## 3.7 Point location in a volume

### Functions

- **GtvCell** ∗ **gtv_point_locate_slow** (GtsPoint ∗p, **GtvVolume** ∗volume, **GtvCell** ∗guess)

### 3.7.1 Function Documentation

#### 3.7.1.1 GtvCell∗ gtv_point_locate_slow (GtsPoint ∗ *p*, GtvVolume ∗ *volume*, GtvCell ∗ *guess*)

Find a **GtvCell** (p. 33) in a given volume which encloses a GtsPoint by testing all of the cells.

**Parameters:**

***p*** a GtsPoint;

***volume*** a **GtvVolume** (p. 39) to search for the location of *p*;

***guess*** ignored, included for compatibility with other location functions.

**Returns:**

a **GtvCell** (p. 33) containing *p* or NULL if *p* is not in *v*.

## 3.8 Parent entities

### Functions

- **GtvFacet** ∗ **gtv_edge_has_parent_volume** (GtsEdge ∗e, **GtvVolume** ∗v)
- **GtvCell** ∗ **gtv_facet_has_parent_volume** (**GtvFacet** ∗f, **GtvVolume** ∗v)
- gboolean **gtv_cell_has_parent_volume** (**GtvCell** ∗c, **GtvVolume** ∗v)
- guint **gtv_edge_facet_number** (GtsEdge ∗e, **GtvVolume** ∗v)
- guint **gtv_facet_cell_number** (**GtvFacet** ∗f, **GtvVolume** ∗v)

### 3.8.1 Function Documentation

#### 3.8.1.1 gboolean gtv_cell_has_parent_volume (GtvCell ∗ *c*, GtvVolume ∗ *v*)

Check if a **GtvCell** (p. 33) belongs to a given **GtvVolume** (p. 39).

**Parameters:**

*c* a **GtvCell** (p. 33);

*v* a **GtvVolume** (p. 39).

**Returns:**

TRUE if *c* belongs to *v*, FALSE otherwise.

#### 3.8.1.2 guint gtv_edge_facet_number (GtsEdge ∗ *e*, GtvVolume ∗ *v*)

Count the number of facets using an edge.

**Parameters:**

*e* a GtsEdge;

*v* a **GtvVolume** (p. 39).

**Returns:**

the number of facets of *v* which contain *e*.

#### 3.8.1.3 GtvFacet∗ gtv_edge_has_parent_volume (GtsEdge ∗ *e*, GtvVolume ∗ *v*)

Check if a GtsEdge has a given parent volume.

**Parameters:**

*e* a GtsEdge;

*v* a **GtvVolume** (p. 39).

**Returns:**

a **GtvFacet** (p. 35) of *v* containing *e*, NULL otherwise.

#### 3.8.1.4 guint gtv_facet_cell_number (GtvFacet ∗ *f*, GtvVolume ∗ *v*)

Count the number of cells using a facet.

**Parameters:**

*f* a **GtvFacet** (p. 35);

*v* a **GtvVolume** (p. 39).

**Returns:**

the number of cells of *v* which contain *f*.

#### 3.8.1.5 GtvCell∗ gtv_facet_has_parent_volume (GtvFacet ∗ *f*, GtvVolume ∗ *v*)

Check if a **GtvFacet** (p. 35) has a given parent volume.

**Parameters:**

*f* a **GtvFacet** (p. 35);

*v* a **GtvVolume** (p. 39).

**Returns:**

a **GtvCell** (p. 33) of *v* containing *f*, NULL otherwise.

## 3.9 GTV tetrahedra

### Data Structures

- struct **GtvTetrahedron**
- struct **GtvTetrahedronClass**

### Functions

- **GtvTetrahedronClass** ∗ **gtv_tetrahedron_class** (void)
- gint **gtv_tetrahedron_set** (**GtvTetrahedron** ∗tetrahedron, **GtvFacet** ∗f1, **GtvFacet** ∗f2, **GtvFacet** ∗f3, **GtvFacet** ∗f4)
- **GtvTetrahedron** ∗ **gtv_tetrahedron_new** (**GtvTetrahedronClass** ∗klass, **GtvFacet** ∗f1, **GtvFacet** ∗f2, **GtvFacet** ∗f3, **GtvFacet** ∗f4)
- gint **gtv_tetrahedron_vertices** (**GtvTetrahedron** ∗t, GtsVertex ∗∗v1, GtsVertex ∗∗v2, GtsVertex ∗∗v3, GtsVertex ∗∗v4)
- gdouble **gtv_tetrahedron_volume** (**GtvTetrahedron** ∗t)
- gboolean **gtv_tetrahedron_has_facet** (**GtvTetrahedron** ∗t, **GtvFacet** ∗f)
- **GtvTetrahedron** ∗ **gtv_tetrahedron_from_facets** (**GtvFacet** ∗f1, **GtvFacet** ∗f2, **GtvFacet** ∗f3, **GtvFacet** ∗f4)
- **GtvTetrahedron** ∗ **gtv_tetrahedron_is_duplicate** (**GtvTetrahedron** ∗t)
- gdouble **gtv_point_in_tetrahedron_sphere** (GtsPoint ∗p, **GtvTetrahedron** ∗t)
- **GtvIntersect gtv_point_in_tetrahedron** (GtsPoint ∗p, **GtvTetrahedron** ∗t, gpointer ∗s)
- **GtvTetrahedron** ∗ **gtv_tetrahedron_large** (**GtvTetrahedronClass** ∗klass, gdouble len)
- **GtvFacet** ∗ **gtv_tetrahedra_common_facet** (**GtvTetrahedron** ∗t1, **GtvTetrahedron** ∗t2)
- **GtvTetrahedron** ∗ **gtv_tetrahedron_opposite** (**GtvTetrahedron** ∗t, **GtvFacet** ∗f)
- gdouble **gtv_tetrahedron_orientation** (**GtvTetrahedron** ∗t)
- gint **gtv_tetrahedron_facets** (**GtvTetrahedron** ∗t, **GtvFacet** ∗∗f1, **GtvFacet** ∗∗f2, **GtvFacet** ∗∗f3, **GtvFacet** ∗∗f4)
- gint **gtv_tetrahedron_invert** (**GtvTetrahedron** ∗t)
- GtsVertex ∗ **gtv_tetrahedron_vertex_opposite** (**GtvTetrahedron** ∗t, **GtvFacet** ∗f)
- **GtvFacet** ∗ **gtv_tetrahedron_facet_opposite** (**GtvTetrahedron** ∗t, GtsVertex ∗v)
- gboolean **gtv_tetrahedron_is_okay** (**GtvTetrahedron** ∗t)
- **GtvTetrahedron** ∗ **gtv_tetrahedron_new_from_vertices** (**GtvTetrahedronClass** ∗klass, **GtvFacetClass** ∗facet_class, GtsEdgeClass ∗edge_class, GtsVertex ∗v1, GtsVertex ∗v2, GtsVertex ∗v3, GtsVertex ∗v4)
- **GtvFacet** ∗ **gtv_point_in_tetrahedron_facet** (**GtvTetrahedron** ∗t, GtsPoint ∗p)
- gint **gtv_tetrahedron_opposite_vertices** (**GtvTetrahedron** ∗t, GtsVertex ∗v, GtsVertex ∗∗v1, GtsVertex ∗∗v2, GtsVertex ∗∗v3)
- gint **gtv_tetrahedron_orient** (**GtvTetrahedron** ∗t)

### 3.9.1 Function Documentation

#### 3.9.1.1 GtvIntersect gtv_point_in_tetrahedron (GtsPoint ∗ *p*, GtvTetrahedron ∗ *t*, gpointer ∗ *s*)

Check if a GtsPoint lies in a **GtvTetrahedron** (p. 37).

**Parameters:**

***p*** a GtsPoint;

*t* a **GtvTetrahedron** (p. 37);

*s* if NULL, ignored; if not NULL, set to the vertex, edge or facet of *t* which *p* intersects, or NULL if *p* lies strictly inside or strictly outside *t*.

**Returns:**

GTV_OUT if *p* is strictly outside *t* and GTV_IN if it lies strictly inside *t*. If *p* lies on the surface of *t*, returns GTV_IN if *s* is NULL, or GTV_ON_FACET, GTV_ON_EDGE or GTV_ON_VERTEX if it lies on a facet, edge or vertex of *t* respectively.

#### 3.9.1.2 GtvFacet∗ gtv_point_in_tetrahedron_facet (GtvTetrahedron ∗ *t*, GtsPoint ∗ *p*)

Find the facet of a **GtvTetrahedron** (p. 37) on which lies a GtsPoint.

**Parameters:**

*t* a **GtvTetrahedron** (p. 37);

*p* a GtsPoint.

**Returns:**

the facet of *t* which contains *p*, i.e. *p* lies in the plane of the facet and inside or on its boundary.

#### 3.9.1.3 gdouble gtv_point_in_tetrahedron_sphere (GtsPoint ∗ *p*, GtvTetrahedron ∗ *t*)

Check whether a point lies inside or outside the circumsphere of a tetrahedron.

**Parameters:**

*p* a GtsPoint;

*t* a **GtvTetrahedron** (p. 37).

**Returns:**

positive value if *p* lies inside *t*, zero if lies on *t* and a negative value if it lies outside *t*.

#### 3.9.1.4 GtvFacet∗ gtv_tetrahedra_common_facet (GtvTetrahedron ∗ *t1*, GtvTetrahedron ∗ *t2*)

Find the facet shared by two tetrahedra.

**Parameters:**

*t1* a **GtvTetrahedron** (p. 37);

*t2* another **GtvTetrahedron** (p. 37).

**Returns:**

the **GtvFacet** (p. 35) between *t1* and *t2*, if they neighbour each other, NULL otherwise.

#### 3.9.1.5 GtvTetrahedronClass∗ gtv_tetrahedron_class (void)

The basic class for tetrahedra in GTV.

**Returns:**

the **GtvTetrahedronClass** (p. 38)

#### 3.9.1.6 GtvFacet∗ gtv_tetrahedron_facet_opposite (GtvTetrahedron ∗ *t*, GtsVertex ∗ *v*)

Find the face of a tetrahedron which is opposite a specified vertex.

**Parameters:**

*t* **GtvTetrahedron** (p. 37);

*v* GtsVertex opposite to which a **GtvFacet** (p. 35) is to be found.

**Returns:**

**GtvFacet** (p. 35) of *t* which is opposite *v*, if *v* is on *t*, NULL otherwise.

#### 3.9.1.7 gint gtv_tetrahedron_facets (GtvTetrahedron ∗ *t*, GtvFacet ∗∗ *f1*, GtvFacet ∗∗ *f2*, GtvFacet ∗∗ *f3*, GtvFacet ∗∗ *f4*)

Find the facets of a tetrahedron.

**Parameters:**

*t* **GtvTetrahedron** (p. 37);

*f1* **GtvFacet** (p. 35) of first face;

*f2* **GtvFacet** (p. 35) of second face;

*f3* **GtvFacet** (p. 35) of third face;

*f4* **GtvFacet** (p. 35) of fourth face;

**Returns:**

GTV_SUCCESS on success.

#### 3.9.1.8 GtvTetrahedron∗ gtv_tetrahedron_from_facets (GtvFacet ∗ *f1*, GtvFacet ∗ *f2*, GtvFacet ∗ *f3*, GtvFacet ∗ *f4*)

Find a tetrahedron which uses a set of facets.

**Parameters:**

*f1* a **GtvFacet** (p. 35);

*f2* a **GtvFacet** (p. 35);

*f3* a **GtvFacet** (p. 35);

*f4* a **GtvFacet** (p. 35).

**Returns:**

a **GtvTetrahedron** (p. 37) which uses *f1*, *f2*, *f3* and *f4*, if one exists, NULL otherwise.

#### 3.9.1.9 gboolean gtv_tetrahedron_has_facet (GtvTetrahedron ∗ *t*, GtvFacet ∗ *f*)

Check if a tetrahedron has a given facet.

**Parameters:**

*t* a **GtvTetrahedron** (p. 37);

*f* a **GtvFacet** (p. 35).

**Returns:**

TRUE if *t* contains *f*, FALSE otherwise.

#### 3.9.1.10 gint gtv_tetrahedron_invert (GtvTetrahedron ∗ *t*)

Invert a tetrahedron by changing the order of two faces. This will change the sign of the tetrahedron volume.

**Parameters:**

*t* **GtvTetrahedron** (p. 37) to invert.

**Returns:**

GTV_SUCCESS on success.

#### 3.9.1.11 GtvTetrahedron∗ gtv_tetrahedron_is_duplicate (GtvTetrahedron ∗ *t*)

Check if a tetrahedron is duplicated.

**Parameters:**

*t* a **GtvTetrahedron** (p. 37);

**Returns:**

NULL if *t* is unique, otherwise the **GtvTetrahedron** (p. 37) which duplicates *t*.

#### 3.9.1.12 gboolean gtv_tetrahedron_is_okay (GtvTetrahedron ∗ *t*)

Check that a **GtvTetrahedron** (p. 37) has positive volume and is not a duplicate.

**Parameters:**

*t* **GtvTetrahedron** (p. 37) to check.

**Returns:**

TRUE if *t* is okay, FALSE otherwise.

#### 3.9.1.13 GtvTetrahedron∗ gtv_tetrahedron_large (GtvTetrahedronClass ∗ *klass*, gdouble *len*)

Generate a 'large' tetrahedron which can be used to enclose a Delaunay tetrahedralization under construction.

**Parameters:**

*klass* a **GtvTetrahedronClass** (p. 38);

*len* a length.

**Returns:**

a **GtvTetrahedron** (p. 37) with vertices at (0,0,*len*), (0,*len*, -*len*), (*len*, -*len*, -*len*) and (-*len*, -*len*, -*len*).

#### 3.9.1.14 GtvTetrahedron∗ gtv_tetrahedron_new (GtvTetrahedronClass ∗ *klass*, GtvFacet ∗ *f1*, GtvFacet ∗ *f2*, GtvFacet ∗ *f3*, GtvFacet ∗ *f4*)

Generate a new tetrahedron from four **GtvFacet** (p. 35)'s

**Parameters:**

*klass* a **GtvTetrahedronClass** (p. 38)

*f1* **GtvFacet** (p. 35)

*f2* **GtvFacet** (p. 35)

*f3* **GtvFacet** (p. 35)

*f4* **GtvFacet** (p. 35)

**Returns:**

the new **GtvTetrahedron** (p. 37)

#### 3.9.1.15 GtvTetrahedron∗ gtv_tetrahedron_new_from_vertices (GtvTetrahedronClass ∗ *klass*, GtvFacetClass ∗ *facet_class*, GtsEdgeClass ∗ *edge_class*, GtsVertex ∗ *v1*, GtsVertex ∗ *v2*, GtsVertex ∗ *v3*, GtsVertex ∗ *v4*)

Generate a new tetrahedron from four GtsVertex's, making use of any existing **GtvFacet** (p. 35)'s which connect them.

**Parameters:**

*klass* a **GtvTetrahedronClass** (p. 38)

*facet_class* a **GtvFacetClass** (p. 36)

*edge_class* a GtsEdgeClass

*v1* a GtsVertex

*v2* a GtsVertex

*v3* a GtsVertex

*v4* a GtsVertex

**Returns:**

the new **GtvTetrahedron** (p. 37)

#### 3.9.1.16 GtvTetrahedron∗ gtv_tetrahedron_opposite (GtvTetrahedron ∗ *t*, GtvFacet ∗ *f*)

Find the tetrahedron neighbouring a specified tetrahedron on a given side.

**Parameters:**

*t* a **GtvTetrahedron** (p. 37)

*f* a **GtvFacet** (p. 35) of *t*

**Returns:**

the **GtvTetrahedron** (p. 37) which neighbours *t* on the side *f*, if there is one, otherwise NULL.

#### 3.9.1.17 gint gtv_tetrahedron_opposite_vertices (GtvTetrahedron ∗ *t*, GtsVertex ∗ *v*, GtsVertex ∗∗ *v1*, GtsVertex ∗∗ *v2*, GtsVertex ∗∗ *v3*)

Find the three vertices of a **GtvTetrahedron** (p. 37) opposite a given GtsVertex of the tetrahedron, respecting the orientation of the tetrahedron. On exit, *v1*, *v2* and *v3* will be the vertices of the facet of *t* opposite *v*, such that the orientation of *v*, *v1*, *v2* and *v3* will be the same as that of the tetrahedron itself, including the sign.

**Parameters:**

*t* a **GtvTetrahedron** (p. 37);

*v* a GtsVertex;

*v1* a GtsVertex;

*v2* a GtsVertex;

*v3* a GtsVertex.

**Returns:**

GTV_SUCCESS or GTV_FAILURE if *v* is not a vertex of *t*.

#### 3.9.1.18 gint gtv_tetrahedron_orient (GtvTetrahedron ∗ *t*)

Orient the facets of a tetrahedron so that it has non-negative volume.

**Parameters:**

*t* a **GtvTetrahedron** (p. 37);

**Returns:**

GTV_SUCCESS on success, i.e. *t* has positive volume.

#### 3.9.1.19 gdouble gtv_tetrahedron_orientation (GtvTetrahedron ∗ *t*)

Find the orientation of the vertices of a tetrahedron. This is an approximation of six times the signed volume of the the tetrahedron.

**Parameters:**

*t* a **GtvTetrahedron** (p. 37).

**Returns:**

a positive value if the tetrahedron apex lies above the plane of the other three points, taken in cyclic order; a negative value if it lies below that plane and zero if it lies in the plane.

#### 3.9.1.20 gint gtv_tetrahedron_set (GtvTetrahedron ∗ *tetrahedron*, GtvFacet ∗ *f1*, GtvFacet ∗ *f2*, GtvFacet ∗ *f3*, GtvFacet ∗ *f4*)

Set the facets of a **GtvTetrahedron** (p. 37). A check is performed to ensure that the facets define a valid tetrahedron.

**Parameters:**

***tetrahedron,:*** a **GtvTetrahedron** (p. 37);

***f1,:*** a **GtvFacet** (p. 35);

***f2,:*** a **GtvFacet** (p. 35);

***f3,:*** a **GtvFacet** (p. 35);

***f4,:*** a **GtvFacet** (p. 35).

**Returns:**

GTV_SUCCESS on success.

#### 3.9.1.21 GtsVertex∗ gtv_tetrahedron_vertex_opposite (GtvTetrahedron ∗ *t*, GtvFacet ∗ *f*)

Find the vertex of a tetrahedron which is opposite a specified face.

**Parameters:**

*t* **GtvTetrahedron** (p. 37);

*f* **GtvFacet** (p. 35) opposite to which a GtsVertex is to be found.

**Returns:**

GtsVertex of *t* which is opposite *f*, if *f* is on *t*, NULL otherwise.

#### 3.9.1.22 gint gtv_tetrahedron_vertices (GtvTetrahedron ∗ *t*, GtsVertex ∗∗ *v1*, GtsVertex ∗∗ *v2*, GtsVertex ∗∗ *v3*, GtsVertex ∗∗ *v4*)

Extract the vertices of a tetrahedron. These are ordered so that *v1* is opposite the first face of *t* and so on.

**Parameters:**

*t* a **GtvTetrahedron** (p. 37);

***v1*** a GtsVertex;

***v2*** a GtsVertex;

***v3*** a GtsVertex;

***v4*** a GtsVertex.

**Returns:**

GTV_SUCCESS on success.

#### 3.9.1.23 gdouble gtv_tetrahedron_volume (GtvTetrahedron $*$ *t*)

Find the signed volume of a tetrahedron.

**Parameters:**

***t*** a **GtvTetrahedron** (p. 37).

**Returns:**

the signed volume of *t*; if this is negative, you might want to use **gtv_tetrahedron_invert** (p. 22) or **gtv_tetrahedron_orient** (p. 24) to reorient the vertices.

# 3.10 Volume

## Data Structures

- struct **GtvVolume**
- struct **GtvVolumeClass**

## Functions

- **GtvVolumeClass** ∗ **gtv_volume_class** (void)
- **GtvVolume** ∗ **gtv_volume_new** (**GtvVolumeClass** ∗klass, **GtvCellClass** ∗cell_class, **GtvFacetClass** ∗facet_class, GtsEdgeClass ∗edge_class, GtsVertexClass ∗vertex_class)
- gint **gtv_volume_add_cell** (**GtvVolume** ∗v, **GtvCell** ∗c)
- gint **gtv_volume_remove_cell** (**GtvVolume** ∗v, **GtvCell** ∗c)
- gint **gtv_volume_write** (**GtvVolume** ∗v, FILE ∗f)
- guint **gtv_volume_read** (**GtvVolume** ∗v, GtsFile ∗f)
- gint **gtv_volume_foreach_cell** (**GtvVolume** ∗v, GtsFunc func, gpointer data)
- gint **gtv_volume_foreach_facet** (**GtvVolume** ∗v, GtsFunc func, gpointer data)
- gint **gtv_volume_foreach_edge** (**GtvVolume** ∗v, GtsFunc func, gpointer data)
- gint **gtv_volume_foreach_vertex** (**GtvVolume** ∗v, GtsFunc func, gpointer data)
- gint **gtv_volume_stats** (**GtvVolume** ∗v, GtvVolumeStats ∗s)
- gint **gtv_volume_print_stats** (**GtvVolume** ∗v, FILE ∗f)
- gint **gtv_volume_boundary** (**GtvVolume** ∗v, GtsSurface ∗s)
- gdouble **gtv_volume_volume** (**GtvVolume** ∗v)
- GtsVertex ∗ **gtv_volume_nearest_vertex** (**GtvVolume** ∗v, GtsPoint ∗p)
- gint **gtv_volume_write_tetgen** (**GtvVolume** ∗v, gchar ∗stub)

### 3.10.1 Function Documentation

#### 3.10.1.1 gint gtv_volume_add_cell (GtvVolume ∗ *v*, GtvCell ∗ *c*)

Add a **GtvCell** (p. 33) to a **GtvVolume** (p. 39).

**Parameters:**

*v* **GtvVolume** (p. 39)

*c* **GtvCell** (p. 33)

**Returns:**

GTV_SUCCESS on success.

#### 3.10.1.2 gint gtv_volume_boundary (GtvVolume ∗ *v*, GtsSurface ∗ *s*)

Add the boundary facets of a **GtvVolume** (p. 39) to a GtsSurface.

**Parameters:**

*v* **GtvVolume** (p. 39)

*s* GtsSurface to take boundary facets

**Returns:**

GTV_SUCCESS on success

#### 3.10.1.3 GtvVolumeClass∗ gtv_volume_class (void)

Generate the class definition for the **GtvVolume** (p. 39) type.

**Returns:**

**GtvVolumeClass** (p. 40).

#### 3.10.1.4 gint gtv_volume_foreach_cell (GtvVolume ∗ *v*, GtsFunc *func*, gpointer *data*)

Execute a function for each cell of a **GtvVolume** (p. 39).

**Parameters:**

***v*** **GtvVolume** (p. 39);
***func*** a GtsFunc to be evaluated for each cell;
***data*** data to be passed to function.

**Returns:**

GTV_SUCCESS on success.

#### 3.10.1.5 gint gtv_volume_foreach_edge (GtvVolume ∗ *v*, GtsFunc *func*, gpointer *data*)

Execute a function for each edge of a **GtvVolume** (p. 39).

**Parameters:**

***v*** **GtvVolume** (p. 39);
***func*** a GtsFunc to be evaluated for each edge;
***data*** data to be passed to function.

**Returns:**

GTV_SUCCESS on success.

#### 3.10.1.6 gint gtv_volume_foreach_facet (GtvVolume ∗ *v*, GtsFunc *func*, gpointer *data*)

Execute a function for each facet of a **GtvVolume** (p. 39).

**Parameters:**

***v*** **GtvVolume** (p. 39);
***func*** a GtsFunc to be evaluated for each facet;
***data*** data to be passed to function.

**Returns:**

GTV_SUCCESS on success.

#### 3.10.1.7 gint gtv_volume_foreach_vertex (GtvVolume * *v*, GtsFunc *func*, gpointer *data*)

Execute a function for each vertex of a **GtvVolume** (p. 39).

**Parameters:**

*v* **GtvVolume** (p. 39);

*func* a GtsFunc to be evaluated for each vertex;

*data* data to be passed to function.

**Returns:**

GTV_SUCCESS on success.

#### 3.10.1.8 GtsVertex* gtv_volume_nearest_vertex (GtvVolume * *v*, GtsPoint * *p*)

Find the vertex of a tetrahedralization which is nearest a GtsPoint.

**Parameters:**

*v* a **GtvVolume** (p. 39);

*p* a GtsPoint.

**Returns:**

the vertex of *v* which is closest to *p*, NULL in case of an error.

#### 3.10.1.9 GtvVolume* gtv_volume_new (GtvVolumeClass * *klass*, GtvCellClass * *cell_class*, GtvFacetClass * *facet_class*, GtsEdgeClass * *edge_class*, GtsVertexClass * *vertex_class*)

Generate a new **GtvVolume** (p. 39)

**Parameters:**

*klass* **GtvVolumeClass** (p. 40) for new volume

*cell_class* **GtvCellClass** (p. 34)

*facet_class* **GtvFacetClass** (p. 36)

*edge_class* GtsEdgeClass

*vertex_class* GtsVertexClass

**Returns:**

the new **GtvVolume** (p. 39)

#### 3.10.1.10 gint gtv_volume_print_stats (GtvVolume * *v*, FILE * *f*)

Print out basic statistics about a **GtvVolume** (p. 39)

**Parameters:**

*v* **GtvVolume** (p. 39)

***f*** file pointer

**Returns:**

GTV_SUCCESS on success

#### 3.10.1.11 guint gtv_volume_read (GtvVolume $*$ *v*, GtsFile $*$ *f*)

Read a volume from file, adding its cells to the **GtvVolume** (p. 39) *v*.

**Parameters:**

***v*** **GtvVolume** (p. 39) to add cells to;

***f*** GtsFile to read from.

**Returns:**

GTV_SUCCESS on success, otherwise the line number where the error occurred.

#### 3.10.1.12 gint gtv_volume_remove_cell (GtvVolume $*$ *v*, GtvCell $*$ *c*)

Remove a **GtvCell** (p. 33) from a **GtvVolume** (p. 39). If gtv_allow_floating_cells is FALSE, the cell will be destroyed if it is not used by any other **GtvVolume** (p. 39).

**Parameters:**

***v*** **GtvVolume** (p. 39);

***c*** **GtvCell** (p. 33) to remove.

**Returns:**

GTV_SUCCESS on success.

#### 3.10.1.13 gint gtv_volume_stats (GtvVolume $*$ *v*, GtvVolumeStats $*$ *s*)

Find basic statistics for a volume

**Parameters:**

***v*** **GtvVolume** (p. 39)

***s*** GtvVolumeStats to fill with data

**Returns:**

GTV_SUCCESS on success

#### 3.10.1.14 gdouble gtv_volume_volume (GtvVolume $*$ *v*)

Volume of a **GtvVolume** (p. 39), found as the sum of the cell volumes.

**Parameters:**

*v* **GtvVolume** (p. 39)

**Returns:**

volume of *v*

#### 3.10.1.15 gint gtv_volume_write (GtvVolume $*$ *v*, FILE $*$ *f*)

Write a **GtvVolume** (p. 39) to file. The file's first line is the number of vertices, edges, facets and cells respectively, followed by the class of each in the volume. There then follow the vertex coordinates, one vertex per line, and then the edges, facets and cells.

**Parameters:**

*v* **GtvVolume** (p. 39) to write;

*f* file pointer.

**Returns:**

GTV_SUCCESS on success.

#### 3.10.1.16 gint gtv_volume_write_tetgen (GtvVolume $*$ *v*, gchar $*$ *stub*)

Write a **GtvVolume** (p. 39) to .node and .ele files which can be read by tetgen, to allow checking and comparisons.

**Parameters:**

*v* a **GtvVolume** (p. 39);

***stub*** the stub file name. Files will be written to stub.node and stub.ele.

**Returns:**

GTV_SUCCESS on success.

# Chapter 4

# GTV Data Structure Documentation

## 4.1 GtvCell Struct Reference

### 4.1.1 Detailed Description

Basic tetrahedral cell derived from **GtvTetrahedron** (p. 37), used to build up **GtvVolume** (p. 39)'s.

## 4.2 GtvCellClass Struct Reference

### 4.2.1 Detailed Description

The basic cell class, derived from the **GtvTetrahedronClass** (p. 38).

## 4.3 GtvFacet Struct Reference

### 4.3.1 Detailed Description

Triangular facet used to form **GtvTetrahedron** (p. 37)'s.

## 4.4 GtvFacetClass Struct Reference

### 4.4.1 Detailed Description

The basic facet class, derived from the GtsTriangleClass.

## 4.5 GtvTetrahedron Struct Reference

### 4.5.1 Detailed Description

Basic tetrahedral cell made up of four **GtvFacet** (p. 35)'s.

## 4.6 GtvTetrahedronClass Struct Reference

### 4.6.1 Detailed Description

The basic tetrahedron class.

# 4.7 GtvVolume Struct Reference

## 4.7.1 Detailed Description

Opaque data structure for the GTV volume.

# 4.8 GtvVolumeClass Struct Reference

## 4.8.1 Detailed Description

The basic class for a **GtvVolume** (p. 39)

# Index