Package 'meshsimp'

June 13, 2017

Version 0.1.1

Date 2017-06-09

Title Simplification of Surface Triangular Meshes with Associated Distributed Data

Author Franco Dassi [aut],
Bree Ettinger [aut],
Marco Martinolli [aut],
Simona Perotto [aut],
Laura M. Sangalli [aut],
Stefano Ubbiali [aut, cre]

Maintainer Stefano Ubbiali <stefano.ubbiali@mail.polimi.it>

Depends R (>= 3.0.2), Rcpp (>= 0.9.11), plot3D

LinkingTo Rcpp (>= 0.9.11), RcppEigen

Imports methods

Description Iterative simplification strategy for surface triangular meshes (2.5D meshes) with associated data. Each iteration corresponds to an edge collapse where the selection of the edge to contract is driven by a cost functional that depends both on the geometry of the mesh than on the distribution of the data locations over the mesh. The library can handle both zero and higher genus surfaces. The package has been designed to be fully compatible with the R package 'fdaPDE', which implements regression models with partial differential regularizations, making use of the Finite Element Method. In the future, the functionalities provided by the current package may be directly integrated into 'fdaPDE'.

License GPL-3

NeedsCompilation yes

SystemRequirements C++11

RoxygenNote 6.0.1

Repository CRAN

Date/Publication 2017-06-13 06:53:42 UTC

R topics documented:

`get.data.locations` *Get the list of data locations from an object of class* `RcppSimplification`.

Description

Get the list of data locations from an object of class `RcppSimplification`.

Usage

```
get.data.locations(x)
```

Arguments

`x`	An object of class `RcppSimplification`.

Value

A #data-by-3 matrix storing the coordinates of data locations.

See Also

`simplify.mesh.2.5D`

get.edges *Get the list of edges from an object of class* RcppSimplification.

Description

Get the list of edges from an object of class RcppSimplification.

Usage

```
get.edges(x)
```

Arguments

x	An object of class RcppSimplification.

Value

A #edges-by-2 matrix, where the i-th row stores the identifiers of the vertices at the end-points of the i-th edge.

See Also

simplify.mesh.2.5D

get.mesh.2.5D *Get surface mesh from an object of class* RcppSimplification.

Description

Extract the mesh from an object of class RcppSimplification. The mesh is returned as an instance of class mesh.2.5D. The order of the Finite Elements is compliant with the input parameter order.

Usage

```
get.mesh.2.5D(x, order = 1)
```

Arguments

x	An object of class RcppSimplification.
order	Either '1' or '2'. It specifies whether each mesh triangle should be represented by 3 nodes (the triangle vertices) or by 6 nodes (the triangle vertices and mid-points of the triangle edges). These are respectively used for linear (order = 1) and quadratic (order = 2) Finite Elements. Default is order = 1.

Value

A `mesh.2.5D` object, endowed with the following attributes:

- `nnodes`: number of nodes in the mesh;
- `nodes`: `nnodes`-by-3 matrix collecting the coordinates of each vertex;
- `ntriangles`: number of triangles in the mesh;
- `triangles`: a `ntriangles`-by-3 (when `order` = 1) or `ntriangles`-by-6 (when `order` = 2) matrix. It specifies the triangles giving the row indices in `nodes` of the triangles vertices and (when `order` = 2) also of the triangles edges midpoints;
- `order`: either '1' or '2'. It specifies whether each mesh triangle should be represented by 3 nodes (the triangle vertices) or by 6 nodes (the triangle vertices and midpoints of the triangle edges). These are respectively used for linear (`order` = 1) and quadratic (`order` = 2) Finite Elements. Default is `order` = 1.

See Also

`simplify.mesh.2.5D`

`get.quantity.of.information`

Get the quantity of information associated with each element of the triangulation, from an object of class `RcppSimplification`.

Description

For each triangle T in the mesh, the associated quantity of information N_T is defined as:

$$N_T := n_f + \frac{1}{2} n_e + \frac{1}{\#\left(T_{v_1}\right)} n_1 + \frac{1}{\#\left(T_{v_2}\right)} n_2 + \frac{1}{\#\left(T_{v_3}\right)} n_3 \,,$$

where n_f and n_e denote the number of data points associated with the face and the edges of the triangle T, respectively. For $j = 1, 2, 3$, n_j is the number of data points associated with the j-th vertex v_j of T, T_{v_j} is the patch of elements associated with v_j (i.e., the elements sharing v_j), and $\#\left(T_{v_j}\right)$ denotes the cardinality of the patch T_{v_j}.

Usage

```
get.quantity.of.information(x)
```

Arguments

`x`	An object of class `RcppSimplification`.

Value

A #elements-by-1 vector, storing the quantity of information for each element.

See Also

simplify.mesh.2.5D

`import.mesh.2.5D` *Instantiate a* `mesh.2.5D` *object from file.*

Description

This function reads a surface mesh from file, and returns an object of class `mesh.2.5D` holding the list of nodes and triangles in the mesh. The parsing of the file is carried out at the C++ level for the sake of efficiency.

Usage

```
import.mesh.2.5D(file)
```

Arguments

`file`	Absolute or relative path to the input mesh; the following file formats are supported: • AVS UCD ASCII (extension .inp); • text files (extension .txt); • Legacy VTK (extension .vtk). Text files are assumed to be structured as the AVS UCD ASCII files (.inp extension).

Value

An object of class `mesh.2.5D`, provided with the following attributes:

- `nnodes`: number of nodes in the mesh;
- `nodes`: `nnodes`-by-3 matrix collecting the coordinates of each vertex;
- `ntriangles`: number of triangles in the mesh;
- `triangles`: a `ntriangles`-by-3 (when `order` = 1) or `ntriangles`-by-6 (when `order` = 2) matrix. It specifies the triangles giving the row indices in `nodes` of the triangles vertices and (when `order` = 2) also of the triangles edges midpoints;
- `order`: either '1' or '2'. It specifies whether each mesh triangle should be represented by 3 nodes (the triangle vertices) or by 6 nodes (the triangle vertices and midpoints of the triangle edges). These are respectively used for linear (`order` = 1) and quadratic (`order` = 2) Finite Elements. Default is `order` = 1.

See Also

simplify.mesh.2.5D, plot.mesh.2.5D

Examples

```
## Import the mesh of a pawn
fpath <- system.file("extdata", "pawn_250.inp", package="meshsimp")
mesh <- import.mesh.2.5D(fpath)
## Simplify the mesh down to 200 nodes; assume the components of the
## edge cost functions are equally weighted and that the data locations
## coincide with the vertices of the mesh
out1 <- simplify.mesh.2.5D(mesh, 200)
## Resume the simplification procedure, reducing the mesh down to 150 nodes
out2 <- simplify.mesh.2.5D(out1$mesh, 150, out1$locations)
```

`mesh` *The mesh of a pawn.*

Description

The mesh of a pawn.

Usage

```
mesh
```

Format

An object of class `mesh.2.5D` of length 5.

`plot.mesh.2.5D` *Plot a mesh in a 3D perspective.*

Description

Plot a 2.5D surface mesh augmented with associated data locations. The package **plot3D** is used.

Usage

```
## S3 method for class 'mesh.2.5D'
plot(x, loc = NULL, phi = 40, theta = 40, ...)
```

Arguments

`x`	An object of class `mesh.2.5D`, representing the mesh to plot.
`loc`	#data-by-3 matrix collecting the locations of data points over the mesh. Default is NULL, i.e. the data points are assumed to coincide with the mesh vertices.
`phi`	Colatitude (in degrees) identifying the viewing direction; default is 40.
`theta`	Longitude (in degrees) identifying the viewing direction; default is 40.
`...`	Additional arguments passed to the plotting methods; these include: xlim, ylim, zlim, xlab, ylab, zlab, main, sub, r, d, scale, expand, box, axes, nticks, ticktype.

See Also

simplify.mesh.2.5D

Examples

```
## Import the mesh of a pawn
 data(pawn_250)
## Plot the original mesh
plot.mesh.2.5D(mesh, main = "Original mesh, 250 nodes")
## Simplify the mesh down to 200 nodes; assume the components of the
## edge cost functions are equally weighted and that the data locations
## coincide with the vertices of the mesh
out <- simplify.mesh.2.5D(mesh, 200)
## Plot the simplified mesh
plot.mesh.2.5D(out$mesh, loc = out$locations, main = "Simplified mesh, 200 nodes")
```

---

simplify.mesh.2.5D *Perform the simplification of a mesh, given as an object of class* mesh.2.5D.

Description

Implementation of a mesh simplification strategy for a surface mesh with associated distributed data. The algorithm works by iteratively collapsing an edge into an internal point. The selection of the edge to contract at each iteration is driven by a cost functional, which measures the loss of geometrical accuracy and data information associated with the collapse. For a detailed description of the algorithm, please refer to:
Dassi, F., Ettinger, B., Perotto, S., Sangalli, L.M. (2015),
A mesh simplification strategy for a spatial regression analysis over the cortical surface of the brain,
Applied Numerical Mathematics, Vol. 90, pp. 111-131.

The whole (computing-intensive) procedure is carried out at the C++ level, thus ensuring high-performance. In detail, the function relies on the class `RcppSimplification` - a wrapper for the template class `simplification<Triangle, MeshType::DATA, DataGeo>` provided by the C++ library `meshsimplification`. Methods of class `RcppSimplification` are exposed to R through the API supplied by the **Rcpp** and **RcppEigen** packages.

Usage

```
simplify.mesh.2.5D(mesh,target,loc=NULL,val=NULL,wgeom=1/3,wdisp=1/3,wequi=1/3,file='')
```

Arguments

mesh	A mesh.2.5D object, endowed with the following attributes: • nnodes: number of nodes in the mesh; • nodes: nnodes-by-3 matrix collecting the coordinates of each vertex; • ntriangles: number of triangles in the mesh;

- `triangles`: a `ntriangles`-by-3 (when `order` = 1) or `ntriangles`-by-6 (when `order` = 2) matrix. It specifies the triangles giving the row indices in `nodes` of the triangles vertices and (when `order` = 2) also of the triangles edges midpoints;
- `order`: either '1' or '2'. It specifies whether each mesh triangle should be represented by 3 nodes (the triangle vertices) or by 6 nodes (the triangle vertices and midpoints of the triangle edges). These are respectively used for linear (`order` = 1) and quadratic (`order` = 2) Finite Elements. Default is `order` = 1.

`target` Number of nodes which the mesh should feature at the end of the simplification process. In other terms, `target` is used as stopping criterium: the algorithm stops when the number of nodes in the mesh matches `target`.

`loc` #data-by-3 vector with data locations; default is NULL, i.e. data locations are assumed to coincide with the mesh vertices.

`val` #data-by-1 vector with the observations associated with each data point; default is NULL.

`wgeom` Weight for the geometric component of the edge cost function; default is 1/3. Note that the all weights should be positive and sum up to one.

`wdisp` Weight for the data displacement component of the edge cost function; default is 1/3. Note that the all weights should be positive and sum up to one.

`wequi` Weight for the data equidistribution component of the edge cost function; default is 1/3. Note that the all weights should be positive and sum up to one.

`file` String specifying the path to the location where the simplified mesh will be stored; the following file formats are supported:

- AVS UCD ASCII (extension .inp);
- text files (extension .txt);
- Legacy VTK (extension .vtk).

Text files are assumed to be structured as the AVS UCD ASCII files (.inp extension). If `outfile` is not provided, the mesh will not get saved to file but just returned as `mesh.2.5D` object.

Value

A list equipped with the following fields:

- `mesh`: the simplified mesh as an instance of class `mesh.2.5D`; the order of the output mesh coincides with the order of the input mesh;
- `simplifier`: the object of class `RcppSimplification` used within the function to carry out the simplification procedure at the C++ level. This object is returned since eases (and speeds up) the extraction of useful information about the mesh, which are not directly made available by `mesh.2.5D` or which may rely on the connectivities of the mesh itself, as, e.g., the list of edges (see `get.edges`);
- `locations`: #data-by-3 matrix holding the location of each data point over the simplified mesh;
- `qoi`: vector listing the quantity of information associated with each triangle in the simplified mesh; see `get.quantity.of.information` for a rigorous definition of the quantity of information.

See Also

plot.mesh.2.5D, import.mesh.2.5D

Examples

```
## Import the mesh of a pawn
 data(pawn_250)
## Simplify the mesh down to 200 nodes; assume the components of the
## edge cost functions are equally weighted and that the data locations
## coincide with the vertices of the mesh
out1 <- simplify.mesh.2.5D(mesh, 200)
## Resume the simplification procedure, reducing the mesh down to 150 nodes
out2 <- simplify.mesh.2.5D(out1$mesh, 150, out1$locations)
```

Index