entente Documentation

Metabolize

Apr 03, 2019

Contents

CHAPTER 1

entente package

1.1 Submodules

1.1.1 entente.cli module

1.1.2 entente.composite module

entente.composite.**composite_meshes**(*mesh_paths*)

Create a composite as a vertex-wise average of several meshes in correspondence. Faces, groups, and other attributes are loaded from the first mesh given.

Parameters **mesh_paths** (*list*) – Paths of the meshes to average.

Returns The composite mesh.

Return type lace.mesh.Mesh

1.1.3 entente.equality module

Utilities related to mesh equality.

entente.equality.**attr_has_same_shape**(*first_obj*, *second_obj*, *attr*)

Given two objects, check if the given arraylike attributes of those objects have the same shape. If one object has an attribute value of None, the other must too.

Parameters

- **first_obj** (*obj*) – A object with an arraylike attr attribute.
- **second_obj** (*obj*) – Another object with an arraylike attr attribute.
- **attr** (*str*) – The name of the attribute to test.

Returns *True* if attributes are the same shape

Return type bool

entente.equality.**attr_is_equal**(*first_obj*, *second_obj*, *attr*)

Given two objects, check if the given arraylike attributes of those objects are equal. If one object has an attribute value of `None`, the other must too.

Parameters

- **first_obj** (*obj*) – A object with an arraylike *attr* attribute.
- **second_obj** (*obj*) – Another object with an arraylike *attr* attribute.
- **attr** (*str*) – The name of the attribute to test.

Returns *True* if attributes are equal

Return type bool

entente.equality.**have_same_topology**(*first_mesh*, *second_mesh*)

Given two meshes, check if they have the same vertex count and same faces. In other words, check if they have the same topology.

Parameters

- **first_mesh** (*lace.mesh.Mesh*) – A mesh.
- **second_mesh** (*lace.mesh.Mesh*) – Another mesh.

Returns *True* if meshes have the same topology

Return type bool

1.1.4 entente.geometry module

Functions relating to mesh geometry.

entente.geometry.**compute_barycentric_coordinates**(*vertices_of_tris*, *points*)

Compute barycentric coordinates for the projection of a set of points to a given set of triangles specfied by their vertices.

These barycentric coordinates can refer to points outside the triangle. This happens when one of the coordinates is negative. However they can't specify points outside the triangle's plane. (That requires tetrahedral coordinates.)

The returned coordinates supply a linear combination which, applied to the vertices, returns the projection of the original point the plane of the triangle.

Parameters

- **vertices_of_tris** (*np.arraylike*) – A set of triangle vertices as *kx3x3*.
- **points** (*np.arraylike*) – Coordinates of points as *kx3*.

Returns Barycentric coordinates as *kx3*

Return type np.ndarray

See also:

https://en.wikipedia.org/wiki/Barycentric_coordinate_system

Note: A function with this signature probably belongs in *blmath*.

1.1.5 entente.landmarks module

1.1.6 entente.restore_correspondence module

entente.restore_correspondence.**find_correspondence**(*a*, *b*, *atol=0.0001*, *all_must_match=True*, *ret_unmatched_b=False*, *progress=True*)

Given *a[0], a[1], ..., a[k]* and *b[0], b[1], ..., b[j]*, match each element of *a* to the corresponding element of *b*.

When *all_must_match* is *True* *a* and *b* must contain the same set of elements. *b[find_correspondence(a, b)]* equals *a*. Otherwise, return *-1* for elements with no match in *b*.

Parameters

- **a** (*np.arraylike*) – *kxn* array.
- **b** (*np.arraylike*) – *jxn* array.
- **atol** (*float*) – Match tolerance.
- **all_must_match** (*bool*) – When *True*, *a* and *b* must contain the same elements.
- **ret_unmatched_b** (*bool*) – When *True*, return a tuple which also contains the indices of *b* which were not matched.
- **progress** (*bool*) – When *True*, show a progress bar.

Returns Indices of *b* as *kx1*

Return type np.ndarray

Note: This relies on a brute-force algorithm.

For the interpretation of *atol*, see documentation for *np.isclose*.

---

entente.restore_correspondence.**restore_correspondence**(*shuffled_mesh*, *reference_mesh*, *atol=0.0001*, *progress=True*)

Given a reference mesh, reorder the vertices of a shuffled copy to restore correspondence with the reference mesh. The vertex set of the shuffled mesh and reference mesh must be equal within *atol*. Mutate *reference_mesh*. Ignore faces but preserves their integrity.

Parameters

- **reference_mesh** (*lace.mesh.Mesh*) – A mesh with the vertices in the desired order.
- **shuffled_mesh** (*lace.mesh.Mesh*) – A mesh with the same vertex set as *reference_mesh*.
- **progress** (*bool*) – When *True*, show a progress bar.

Returns *vx1* which maps old vertices in *shuffled_mesh* to new.

Return type np.ndarray

Note: This was designed to assist in extracting face ordering and groups from a *shuffled_mesh* that "work" with *reference_mesh*, so the face ordering and groups can be used with different vertices.

It relies on a brute-force algorithm.

1.1.7 entente.shuffle module

entente.shuffle.**shuffle_faces**(*mesh*)
Shuffle the mesh's face ordering. The mesh is mutated.

Parameters **mesh** (*lace.mesh.Mesh*) – A mesh.

Returns *fx1* mapping of old face indices to new.

Return type np.ndarray

entente.shuffle.**shuffle_vertices**(*mesh*)
Shuffle the mesh's vertex ordering, preserving the integrity of the faces. The mesh is mutated.

Parameters **mesh** (*lace.mesh.Mesh*) – A mesh.

Returns *vx1* mapping of old vertex indices to new.

Return type np.ndarray

1.1.8 entente.testing module

entente.testing.**assert_same_face_set**(*a*, *b*)

entente.testing.**assert_same_vertex_set**(*a*, *b*)

entente.testing.**coord_set**(*a*)

entente.testing.**mesh_asset**(**components*)

entente.testing.**relative_to_project**(**components*)

entente.testing.**vitra_mesh**()

1.1.9 entente.trimesh_search module

On Mac OS:

```
brew install spatialindex
pip install rtree trimesh
```

entente.trimesh_search.**faces_nearest_to_points**(*mesh*, *query_points*, *ret_points=False*)
Find the triangular faces on a mesh which are nearest to the given query points.

Parameters

- **query_points** (*np.arraylike*) – The points to query, with shape *kx3*
- **ret_points** (*bool*) – When *True*, return both the indices of the nearest faces and the closest points to the query points, which are not necessarily vertices. When *False*, return only the face indices.

Returns face indices as *kx1 np.ndarray*, or when *ret_points* is *True*, a tuple also including the coordinates of the closest points as *kx3 np.ndarray*.

Return type object

entente.trimesh_search.**require_trimesh_with_rtree**()
Check that trimesh and rtree are installed and can be imported, and raise an error with a helpful error message if they are not.

1.1.10 entente.validation module

entente.validation.**validate_shape**(*a*, **shape*, ***kwargs*)

Check that the given argument has the expected shape. Shape dimensions can be ints or -1 for a wildcard. The wildcard dimensions are returned, which allows them to be used for subsequent validation or elsewhere in the function.

Parameters

- **a** (*np.arraylike*) – An array-like input.
- **shape** (*list*) – Shape to validate. To require 3 by 1, pass *3*. To require n by 3, pass *-1, 3*.
- **name** (*str*) – Variable name to embed in the error message.

Returns The wildcard dimension (if one) or a tuple of wildcard dimensions (if more than one).

Return type object

entente.validation.**validate_shape_from_ns**(*namespace*, *name*, **shape*)

Convenience function for invoking *validate_shape()* with a *locals()* dict.

Parameters

- **namespace** (*dict*) – A subscriptable object, typically *locals()*.
- **name** (*str*) – Key to pull from *namespace*.
- **shape** (*list*) – Shape to validate. To require 3 by 1, pass *3*. To require n by 3, pass *-1, 3*.

Returns The wildcard dimension (if one) or a tuple of wildcard dimensions (if more than one).

Return type object

Example

validate_shape_from_namespace(locals(), 'points', -1, 3)

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

e

Index