entente

Nov 06, 2019

Contents

CHAPTER 1

entente package

1.1 Subpackages

1.1.1 entente.landmarks package

Submodules

entente.landmarks.landmark_composite_recipe module

entente.landmarks.landmark_compositor module

class entente.landmarks.landmark_compositor.**LandmarkCompositor**(*base_mesh*, *landmark_names*)

Bases: object

A tool for compositing landmarks from several examples in relation to a base mesh. Each example is projected onto the base mesh, then the points are averaged.

The tool takes as input:

- A base mesh
- **Several examples**
  - Mesh (in correspondence with the base mesh)
  - xyz coordinates for one or more landmarks

And will output:

- The xyz coordinates of the comosite landmark on the base mesh

add_example(*mesh*, *landmarks*)

result

entente.landmarks.landmarker module

Functions for transferring landmarks from one mesh to another.

This module requires libspatialindex and rtree. See note in *trimesh_search.py*.

class entente.landmarks.landmarker.**Landmarker**(*source_mesh*, *landmarks*)
Bases: object

An object which encapsulates a source mesh and a set of landmarks on that mesh. Its function is to transfer those landmarks onto a new mesh.

The resultant landmarks will always be on or near the surface of the mesh.

Parameters

- **source_mesh** (*lace.mesh.Mesh*) – The source mesh
- **landmarks** (*dict*) – A mapping of landmark names to the points, which are *3x1* arraylike objects.

classmethod load(*source_mesh_path*, *landmark_path*)
Create a landmarker using the given paths to a source mesh and landmarks.

Parameters

- **source_mesh_path** (*str*) – File path to the source mesh.
- **landmark_path** (*str*) – File path to a meshlab `.pp` file containing the landmark points.

transfer_landmarks_onto(*target*)
Transfer landmarks onto the given target mesh, which must be in the same topology as the source mesh.

Parameters **target** (*lace.mesh.Mesh*) – Target mesh

Returns A mapping of landmark names to a np.ndarray with shape *3x1*.

Return type dict

1.2 Submodules

1.2.1 entente.cli module

1.2.2 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

1.2.3 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

1.2.4 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.

1.2.5 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

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

e

Index