. Documentation

Release 0.1.1

Author

Nov 13, 2017

Contents

1 Branch3D module 3

2 FractalTree module 7

3 Mesh module 9

4 parameters module 11

5 Indices and tables 13

Python Module Index 15

This code is to create a fractal tree over a surface discretized by triangles. It was developed to create a representation of the Purkinje network in the ventricles of the human heart.

The details of the algorithm are presented in this article. If you are going to use this code, please cite:

Generating Purkinje networks in the human heart.
F. Sahli Costabal, D. Hurtado and E. Kuhl.
Journal of Biomechanics, accepted for publication.

Pre-requisites:

- Numpy
- Scipy
- Mayavi, if you want to export Paraview files for visualization.

You will need .obj mesh file to create the tree. A very nice software to manipulate the mesh and export it to .obj is MeshLab. Please check if the mesh has duplicated vertex or faces before running the code. Also the orientation of the normals can change your results, because the angles will be fliped. To visualize the output, the best alternative is Paraview.

To define the mesh file and the parameters of the tree to use, edit the parameters.py file and then run:

```
from FractalTree import *
from parameters import Parameters

param=Parameters()

branches, nodes = Fractal_Tree_3D(param)
```

If you have questions you can contact me at francisco.sahli at gmail.com

Contents:

CHAPTER 1

Branch3D module

This module contains the Branch class (one branch of the tree) and the Nodes class

class Branch3D.**Branch**(*mesh*, *init_node*, *init_dir*, *init_tri*, *l*, *angle*, *w*, *nodes*, *brother_nodes*, *Nsegments*)

Class that contains a branch of the fractal tree.

Parameters

- **mesh** – an object of the mesh class, where the fractal tree will grow
- **init_node** (*int*) – initial node to grow the branch. This is an index that refers to a node in the nodes.nodes array.
- **init_dir** (*array*) – initial direction to grow the branch. In general, it refers to the direction of the last segment of the mother brach.
- **init_tri** (*int*) – the index of triangle of the mesh where the init_node sits.
- **l** (*float*) – total length of the branch
- **angle** (*float*) – angle (rad) with respect to the init_dir in the plane of the init_tri triangle
- **w** (*float*) – repulsitivity parameter. Controls how much the branches repel each other.
- **nodes** – the object of the class nodes that contains all the nodes of the existing branches.
- **brother_nodes** (*list*) – the nodes of the brother and mother branches, to be excluded from the collision detection between branches.
- **Nsegments** (*int*) – number of segments to divide the branch.

child

list – contains the indexes of the child branches. It is not assigned when created.

dir

array – vector direction of the last segment of the branch.

nodes

list – contains the node indices of the branch. The node coordinates can be retrieved using nodes.nodes[i]

triangles
list – contains the indices of the triangles from the mesh where every node of the branch lies.

tri
int – triangle index where last node sits.

growing
bool – False if the branch collide or is out of the surface. True otherwise.

add_node_to_queue(*mesh*, *init_node*, *dir*)
Functions that projects a node in the mesh surface and it to the queue is it lies in the surface.

Parameters

- **mesh** – an object of the mesh class, where the fractal tree will grow
- **init_node** (*array*) – vector that contains the coordinates of the last node added in the branch.
- **dir** (*array*) – vector that contains the direction from the init_node to the node to project.

Returns true if the new node is in the triangle.

Return type success (bool)

class Branch3D.**Nodes**(*init_node*)
A class containing the nodes of the branches plus some fuctions to compute distance related quantities.

Parameters **init_node** (*array*) – an array with the coordinates of the initial node of the first branch.

nodes
list – list of arrays containing the coordinates of the nodes

last_node
int – last added node.

end_nodes
list – a list containing the indices of all end nodes (nodes that are not connected) of the tree.

tree
scipy.spatial.cKDTree – a k-d tree to compute the distance from any point to the closest node in the tree. It is updated once a branch is finished.

collision_tree
scipy.spatial.cKDTree – a k-d tree to compute the distance from any point to the closest node in the tree, except from the brother and mother branches. It is used to check collision between branches.

add_nodes(*queue*)
This function stores a list of nodes of a branch and returns the node indices. It also updates the tree to compute distances.

Parameters **queue** (*list*) – a list of arrays containing the coordinates of the nodes of one branch.

Returns the indices of the added nodes.

Return type nodes_id (list)

collision(*point*)
This function returns the distance between one point and the closest node in the tree and the index of the closest node using the collision_tree.

Parameters **point** (*array*) – the coordinates of the point to calculate the distance from.

Returns (distance to the closest node, index of the closest node)

Return type *collision* (tuple)

distance_from_node(*node*)
This function returns the distance from any node to the closest node in the tree.

Parameters **node** (*int*) – the index of the node to calculate the distance from.

Returns the distance between specified node and the closest node in the tree.

Return type d (float)

distance_from_point(*point*)
This function returns the distance from any point to the closest node in the tree.

Parameters **point** (*array*) – the coordinates of the point to calculate the distance from.

Returns the distance between point and the closest node in the tree.

Return type d (float)

gradient(*point*)
This function returns the gradient of the distance from the existing points of the tree from any point. It uses a central finite difference approximation.

Parameters **point** (*array*) – the coordinates of the point to calculate the gradient of the distance from.

Returns (x,y,z) components of gradient of the distance.

Return type grad (array)

update_collision_tree(*nodes_to_exclude*)
This function updates the collision_tree excluding a list of nodes from all the nodes in the tree. If all the existing nodes are excluded, one distant node is added.

Parameters **nodes_to_exclude** (*list*) – contains the nodes to exclude from the tree. Usually it should be the mother and the brother branch nodes.

Returns none

CHAPTER 2

FractalTree module

This module contains the function that creates the fractal tree.

FractalTree.**Fractal_Tree_3D**(*param*)

This fuction creates the fractal tree. :param param: this object contains all the parameters that define the tree. See the parameters module documentation for details: :type param: Parameters object

Returns A dictionary that contains all the branches objects. nodes (nodes object): the object that contains all the nodes of the tree.

Return type branches (dict)

CHAPTER 3

Mesh module

This module contains the mesh class. This class is the triangular surface where the fractal tree is grown.

class Mesh.**Mesh**(*filename*)

Class that contains the mesh where fractal tree is grown. It must be Wavefront .obj file. Be careful on how the normals are defined. It can change where an specified angle will go.

Parameters **filename** (*str*) – the path and filename of the .obj file with the mesh.

verts

array – a numpy array that contains all the nodes of the mesh. verts[i,j], where i is the node index and j=[0,1,2] is the coordinate (x,y,z).

connectivity

array – a numpy array that contains all the connectivity of the triangles of the mesh. connectivity[i,j], where i is the triangle index and j=[0,1,2] is node index.

normals

array – a numpy array that contains all the normals of the triangles of the mesh. normals[i,j], where i is the triangle index and j=[0,1,2] is normal coordinate (x,y,z).

node_to_tri

dict – a dictionary that relates a node to the triangles that it is connected. It is the inverse relation of connectivity. The triangles are stored as a list for each node.

tree

scipy.spatial.cKDTree – a k-d tree to compute the distance from any point to the closest node in the mesh.

loadOBJ(*filename*)

This function reads a .obj mesh file

Parameters **filename** (*str*) – the path and filename of the .obj file.

Returns a numpy array that contains all the nodes of the mesh. verts[i,j], where i is the node index and j=[0,1,2] is the coordinate (x,y,z). connectivity (array): a numpy array that contains all the connectivity of the triangles of the mesh. connectivity[i,j], where i is the triangle index and j=[0,1,2] is node index.

Return type *verts* (array)

project_new_point(*point*)

This function projects any point to the surface defined by the mesh.

Parameters **point** (*array*) – coordinates of the point to project.

Returns the coordinates of the projected point that lies in the surface. intriangle (int): the index of the triangle where the projected point lies. If the point is outside surface, intriangle=-1.

Return type projected_point (array)

CHAPTER 4

parameters module

This module contains the Parameters class that is used to specify the input parameters of the tree.

class parameters.**Parameters**

Class to specify the parameters of the fractal tree.

meshfile
str – path and filename to obj file name.

filename
str – name of the output files.

init_node
numpy array – the first node of the tree.

second_node
numpy array – this point is only used to calculate the initial direction of the tree and is not included in the tree. Please avoid selecting nodes that are connected to the init_node by a single edge in the mesh, because it causes numerical issues.

init_length
float – length of the first branch.

N_it
int – number of generations of branches.

length
float – average lenght of the branches in the tree.

std_length
float – standard deviation of the length. Set to zero to avoid random lengths.

min_length
float – minimum length of the branches. To avoid randomly generated negative lengths.

branch_angle
float – angle with respect to the direction of the previous branch and the new branch.

w
float – repulsivity parameter.

l_segment
float – length of the segments that compose one branch (approximately, because the lenght of the branch is random). It can be interpreted as the element length in a finite element mesh.

Fascicles
bool – include one or more straigth branches with different lengths and angles from the initial branch. It is motivated by the fascicles of the left ventricle.

fascicles_angles
list – angles with respect to the initial branches of the fascicles. Include one per fascicle to include.

fascicles_length
list – length of the fascicles. Include one per fascicle to include. The size must match the size of fascicles_angles.

save
bool – save text files containing the nodes, the connectivity and end nodes of the tree.

save_paraview
bool – save a .vtu paraview file. The tvtk module must be installed.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

b

Branch3D, 3

f

FractalTree, 7

m

Mesh, 9

p

parameters, 11

Index

A

add_node_to_queue() (Branch3D.Branch method), 4
add_nodes() (Branch3D.Nodes method), 4

B

Branch (class in Branch3D), 3
Branch3D (module), 3
branch_angle (parameters.Parameters attribute), 11

C

child (Branch3D.Branch attribute), 3
collision() (Branch3D.Nodes method), 4
collision_tree (Branch3D.Nodes attribute), 4
connectivity (Mesh.Mesh attribute), 9

D

dir (Branch3D.Branch attribute), 3
distance_from_node() (Branch3D.Nodes method), 5
distance_from_point() (Branch3D.Nodes method), 5

E

end_nodes (Branch3D.Nodes attribute), 4

F

Fascicles (parameters.Parameters attribute), 12
fascicles_angles (parameters.Parameters attribute), 12
fascicles_length (parameters.Parameters attribute), 12
filename (parameters.Parameters attribute), 11
Fractal_Tree_3D() (in module FractalTree), 7
FractalTree (module), 7

G

gradient() (Branch3D.Nodes method), 5
growing (Branch3D.Branch attribute), 4

I

init_length (parameters.Parameters attribute), 11
init_node (parameters.Parameters attribute), 11

L

l_segment (parameters.Parameters attribute), 12
last_node (Branch3D.Nodes attribute), 4
length (parameters.Parameters attribute), 11
loadOBJ() (Mesh.Mesh method), 9

M

Mesh (class in Mesh), 9
Mesh (module), 9
meshfile (parameters.Parameters attribute), 11
min_length (parameters.Parameters attribute), 11

N

N_it (parameters.Parameters attribute), 11
node_to_tri (Mesh.Mesh attribute), 9
nodes (Branch3D.Branch attribute), 3
nodes (Branch3D.Nodes attribute), 4
Nodes (class in Branch3D), 4
normals (Mesh.Mesh attribute), 9

P

Parameters (class in parameters), 11
parameters (module), 11
project_new_point() (Mesh.Mesh method), 10

S

save (parameters.Parameters attribute), 12
save_paraview (parameters.Parameters attribute), 12
second_node (parameters.Parameters attribute), 11
std_length (parameters.Parameters attribute), 11

T

tree (Branch3D.Nodes attribute), 4
tree (Mesh.Mesh attribute), 9
tri (Branch3D.Branch attribute), 4
triangles (Branch3D.Branch attribute), 3

U

update_collision_tree() (Branch3D.Nodes method), 5

V

verts (Mesh.Mesh attribute), 9

W

w (parameters.Parameters attribute), 11