SoftRobots Templates Documentation

Release 1.0

damien.marchal@univ-lille1.fr

Mar 08, 2019

Contents

Utility functions and scene templates for the real-time simulation framework SOFA and the SoftRobots plugin.

The library can be used with scenes written in python and PSL.

CHAPTER 1

Example:

```
from stlib.scene import MainHeader
from stlib.physics.rigid import Cube, Floor
from stlib.physics.deformable import ElasticMaterialObject

from softrobots.actuators import PullingCable
from softrobots.sensors import StringSensor

def createScene(rootNode):
    MainHeader(rootNode)
    DefaultSolver(rootNode)

    Cube(rootNode, translation=[5.0,0.0,0.0])
    Floor(rootNode, translation=[0.0,-1.0,0.0])

   target = ElasticMaterialObject(volumeMeshFileName="mesh/liver.msh",
                                  totalMass=0.5,
                                  attachedTo=node)

    PullingCable(target)
    StringSensor(target)
```

CHAPTER 2

Contents of the library

softrobots.actuators	Template for actuators.
softrobots.sensors	
softrobots.parts	Templates for robot parts.
softrobots.inverse	Inverse model.

2.1 softrobots.actuators

Template for actuators.

2.1.1 Content:

PullingCable([attachedTo, name, ...])	Creates and adds a cable constraint.
PneumaticCavity([surfaceMeshFileName, ...])	Creates and adds a pneumatic constraint.

softrobots.actuators.**PullingCable**(*attachedTo=None, name='Cable', cableGeometry=[[1.0, 0.0, 0.0], [0.0, 0.0, 0.0]], rotation=[0.0, 0.0, 0.0], translation=[0.0, 0.0, 0.0], uniformScale=1.0, pullPointLocation=None, initialValue=0.0, valueType='displacement'*)

Creates and adds a cable constraint.

The constraint apply to a parent mesh.

Args: name (str): Name of the created cable.

cableGeometry: (list vec3f): Location of the degree of freedom of the cable.

pullPointLocation (vec3f): Position from where the cable is pulled. If not specified the point will be considered in the structure.

valueType (str): either “force” or “displacement”. Default is displacement.

translation (vec3f): Apply a 3D translation to the object.

rotation (vec3f): Apply a 3D rotation to the object in Euler angles.

uniformScale (vec3f): Apply an uniform scaling to the object.

Structure:

```
Node : {
        name : "Cable"
        MechanicalObject,
        CableConstraint,
        BarycentricMapping
}
```

softrobots.actuators.**PneumaticCavity**(*surfaceMeshFileName=None, attachedAsAChildOf=None, attachedTo=None, name='PneumaticCavity', rotation=[0.0, 0.0, 0.0], translation=[0.0, 0.0, 0.0], uniformScale=1, initialValue=0, valueType='volumeGrowth'*)

Creates and adds a pneumatic constraint.

The constraint apply to a parent mesh.

Args: cavityMeshFile (string): path to the cavity mesh (the mesh should be a surfacic mesh, ie only triangles or quads).

name (string): name of the created node.

initialValue (real): value to apply, default is 0.

valueType (string): type of the parameter value (volumeGrowth or pressure), default is volumeGrowth.

Structure: .. sourcecode:: qml

Node [{] name : “PneumaticCavity” MeshTopology, MechanicalObject, SurfacePressureConstraint, BarycentricMapping

}

2.2 softrobots.parts

Templates for robot parts.

2.2.1 Content of the library

parts.finger

2.3 softrobots.inverse

Inverse model.

2.3.1 Content:

softrobots.inverse.actuators	Template for actuators.
softrobots.inverse.effectors	Template for effectors.

softrobots.inverse.actuators

Template for actuators.

Content:

PullingCable([attachedTo, name, ...])	Creates and adds a cable actuators.
PneumaticCavity([surfaceMeshFileName, ...])	Creates and adds a pneumatic actuation.

softrobots.inverse.actuators.**PullingCable**(*attachedTo=None, name='Cable', cableGeometry=[[1.0, 0.0, 0.0], [0.0, 0.0, 0.0]], pullPointLocation=None, minForce=None, maxForce=None, minDisplacement=None, maxDisplacement=None, maxDispVariation=None, translation=[0.0, 0.0, 0.0], rotation=[0.0, 0.0, 0.0], uniformScale=1.0*)

Creates and adds a cable actuators. Should be used in the context of the resolution of an inverse problem: find the actuation that leads to a desired deformation. See documentation at: https://project.inria.fr/softrobot/documentation/constraint/cable-actuator/

The constraint is applied to a parent mesh.

Args: name (str): Name of the created cable.

cableGeometry: (list vec3f): Location of the degree of freedom of the cable.

pullPointLocation (vec3f): Position from where the cable is pulled. If not specified the point will be considered in the structure.

minForce:

maxForce:

minDisplacement:

maxDisplacement:

maxDispVariation:

translation (vec3f): Apply a 3D translation to the object.

rotation (vec3f): Apply a 3D rotation to the object in Euler angles.

uniformScale (vec3f): Apply an uniform scaling to the object.

Structure:

```
Node : {
        name : "Cable"
```

(continues on next page)

(continued from previous page)

```
        MechanicalObject,
        CableActuator,
        BarycentricMapping
}
```

softrobots.inverse.actuators.**PneumaticCavity**(*surfaceMeshFileName=None, attachedTo=None, name='PneumaticCavity', minPressure=None, maxPressure=None, minVolumeGrowth=None, maxVolumeGrowth=None, maxVolumeGrowthVariation=None, rotation=[0.0, 0.0, 0.0], translation=[0.0, 0.0, 0.0], uniformScale=1*)

Creates and adds a pneumatic actuation. Should be used in the context of the resolution of an inverse problem: find the actuation that leads to a desired deformation. See documentation at: https://project.inria.fr/softrobot/documentation/constraint/surface-pressure-actuator/

The constraint is applied to a parent mesh.

Args: cavityMeshFile (string): path to the cavity mesh (the mesh should be a surfacic mesh, ie only triangles or quads).

name (string): name of the created node.

minPressure:

maxPressure:

minVolumeGrowth:

maxVolumeGrowth:

maxVolumeGrowthVariation:

Structure: .. sourcecode:: qml

Node [{] name : "PneumaticCavity" MeshTopology, MechanicalObject, SurfacePressureConstraint, BarycentricMapping

}

softrobots.inverse.effectors

Template for effectors.

Content:

PositionEffector([attachedTo, name, ...])	Creates and adds a position effector.
EffectorGoal([attachedTo, name, position, ...])	Creates and adds a effector goal.

softrobots.inverse.effectors.**PositionEffector**(*attachedTo=None, name='effector', position=None, effectorGoal=None, template='Vec3d', directions=None, useDirections=None, translation=[0.0, 0.0, 0.0], rotation=[0.0, 0.0, 0.0], uniformScale=1.0*)

Creates and adds a position effector. Should be used in the context of the resolution of an inverse problem: find the actuation that leads to a desired position of the effector. See documentation at: https://project.inria.fr/softrobot/documentation/constraint/position-effector/

The constraint apply to a parent mesh.

Args: name (str): Name of the created effector.

position: Location of the degree of freedom of the effector.

effectorGoal: Location of the effector target

template:

translation (vec3f): Apply a 3D translation to the object.

rotation (vec3f): Apply a 3D rotation to the object in Euler angles.

uniformScale (vec3f): Apply an uniform scaling to the object.

Structure:

```
Node : {
        name : "effector"
        MechanicalObject,
        PositionEffector,
        BarycentricMapping or RigidMapping
}
```

softrobots.inverse.effectors.**EffectorGoal**(*attachedTo=None, name='Goal', position=None, template='Vec3d', translation=[0.0, 0.0, 0.0], rotation=[0.0, 0.0, 0.0], uniformScale=1.0, visuScale=0.1*)

Creates and adds a effector goal. Should be used in the context of the resolution of an inverse problem: find the actuation that leads to the given desired position. See examples in SoftRobots/docs/tutorials

Args: name (str): Name of the effector goal.

position: Location of the target position(s) of the effector(s).

template:

translation (vec3f): Apply a 3D translation to the object.

rotation (vec3f): Apply a 3D rotation to the object in Euler angles.

uniformScale (vec3f): Apply an uniform scaling to the object.

Structure:

```
Node : {
        name : "Goal"
        EulerImplicit,
        CGLinearSolver,
        MechanicalObject
}
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index