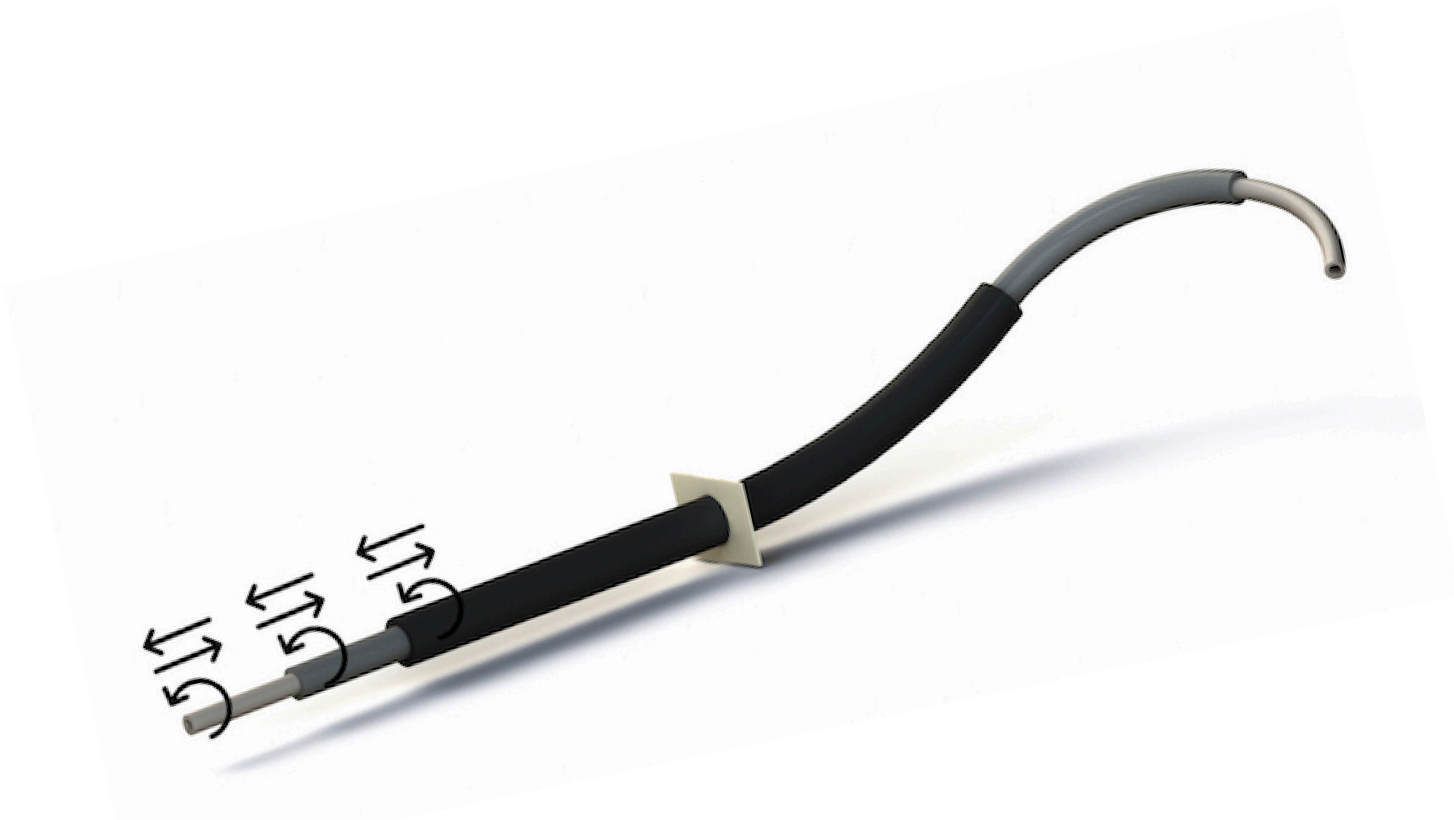
Technical Project: Medical Robotics

Concentric Tube Robot



STUDENTS

Catapano Serena

Gaudino Emanuele

Percuoco Paola

Varone Emanuela

PROFESSOR

Fanny Ficuciello

Goal: implementation of simulation scene in SOFA of a Concentric Tube Robot interacting with an elastic object, trough keyboard command

Workflow:

Model of a CTR (3 tubes) → **Keyboard Controller** → **Cartesian position of the tip** → **Volumetric mesh & Elastic object** → **Interaction & Force**

Development environments

a framework that implements a mathematical model describing the phisics around us

a high-level programming language and interactive environment for numerical computation, data analysis, and visualization

an interpreted, object-oriented, high-level programming language with dynamic semantics

Medical background

Concentric Tube Robots are well suited for minimally invasive surgery inside body cavities.

The design should ensure the capability, for the Robot, to perform a surgical procedure into the body.

Parameters like the lenght (for each of the 3 tubes) and the curvature highly determine the workspace.

The particular composition of concentrically arranged tubes allows for simple, yet dextrous, robotic manipulators.

Actuation is achieved mechanically by relative rotation and axial translation of all the tubes and the motion is a tentacle-like one.

In our case, the **design parameters** are the following:

	TUBE 1	TUBE 2	TUBE 3
Straight length	110 mm	135 mm	170 mm
Curved length	30 mm	40 mm	55 mm
Tube radius	0.95 mm	0.85 mm	0.75 mm
Radius Curvature	100 mm	115 mm	140 mm

Concentric Tube Robots are implemented into the SOFA scene using a .py script which allows to set all the relevant parameters for the simulation: the ones from the previous table, mostly important for the shaping of the tubes and others about the rigidity of the tubes and their mass density, which will be determinant in the interaction with a soft or rigid object.

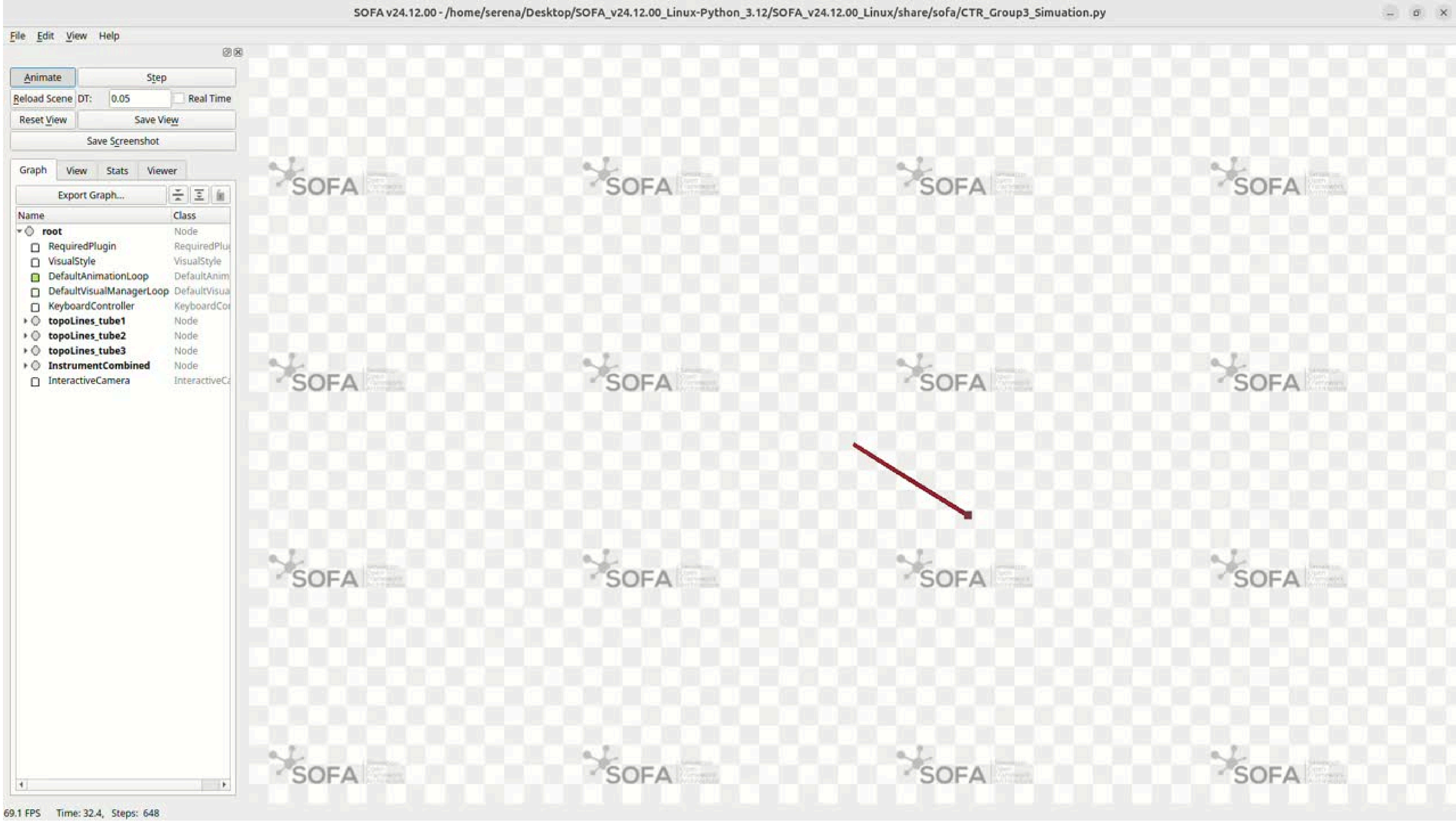
```python
# Topologia del tubo 1 (esterno)
topoLines_tube1 = rootNode.addChild('topoLines_tube1')
topoLines_tube1.addObject('RodStraightSection', name="StraightSection",
                          youngModulus="77000000000",
                          massDensity="6450",
                          radius=TUBE1_RADIUS,
                          nbEdgesCollis="40",
                          nbEdgesVisu="120",
                          length=TUBE1_STRAIGHT_LENGTH)

topoLines_tube1.addObject('RodSpireSection', name="SpireSection",
                          youngModulus="77000000000",
                          massDensity="6450",
                          radius=TUBE1_RADIUS,
                          nbEdgesCollis="20",
                          nbEdgesVisu="80",
                          length=TUBE1_CURVED_LENGTH,
                          spireDiameter=TUBE1_RADIUS_CURVATURE*2,
                          spireHeight="0.0")

topoLines_tube1.addObject('WireRestShape', template="Rigid3d", name="Tube1RestShape",
                          wireMaterials="@StraightSection @SpireSection")

topoLines_tube1.addObject('EdgeSetTopologyContainer', name="meshLinesTube1")
topoLines_tube1.addObject('EdgeSetTopologyModifier', name="Modifier")
topoLines_tube1.addObject('EdgeSetGeometryAlgorithms', name="GeomAlgo", template="Rigid3d")
topoLines_tube1.addObject('MechanicalObject', template="Rigid3d", name="dofTopo1")
```

Keyboard controller simulation

- The Keyboard Controller is a Python class that provides real-time interactive control of the CTR.

- Enables users to manually manipulate individual tubes through keyboard inputs: **keys 1-3** select which tube to control, **'u'/'j'** for translation (insertion/retraction), and **'h'/'k'** for rotation. Additionally, key **'t'** captures and logs the current tip position to a CSV file, while key **'f'** records contact forces between the CTR and surrounding tissues.

- The implementation inherits from SOFA's Controller class and overrides the **onKeypressedEvent()** method to capture keyboard inputs. When movement keys are pressed, it calls the **InterventionalRadiologyController's** methods (**deployControlledInstrument()** and **rotateControlledInstrument()**) to actuate the selected tube.



Cartesian position of the tip

To retrive the cartesian position of the tip, the getTipPosition() function has been employed.

Next, the results have been saved within a .csv file.

Finally, using MATLAB, the trajectory followed by the CTR has been poltted in a 3D graph.

RM.m

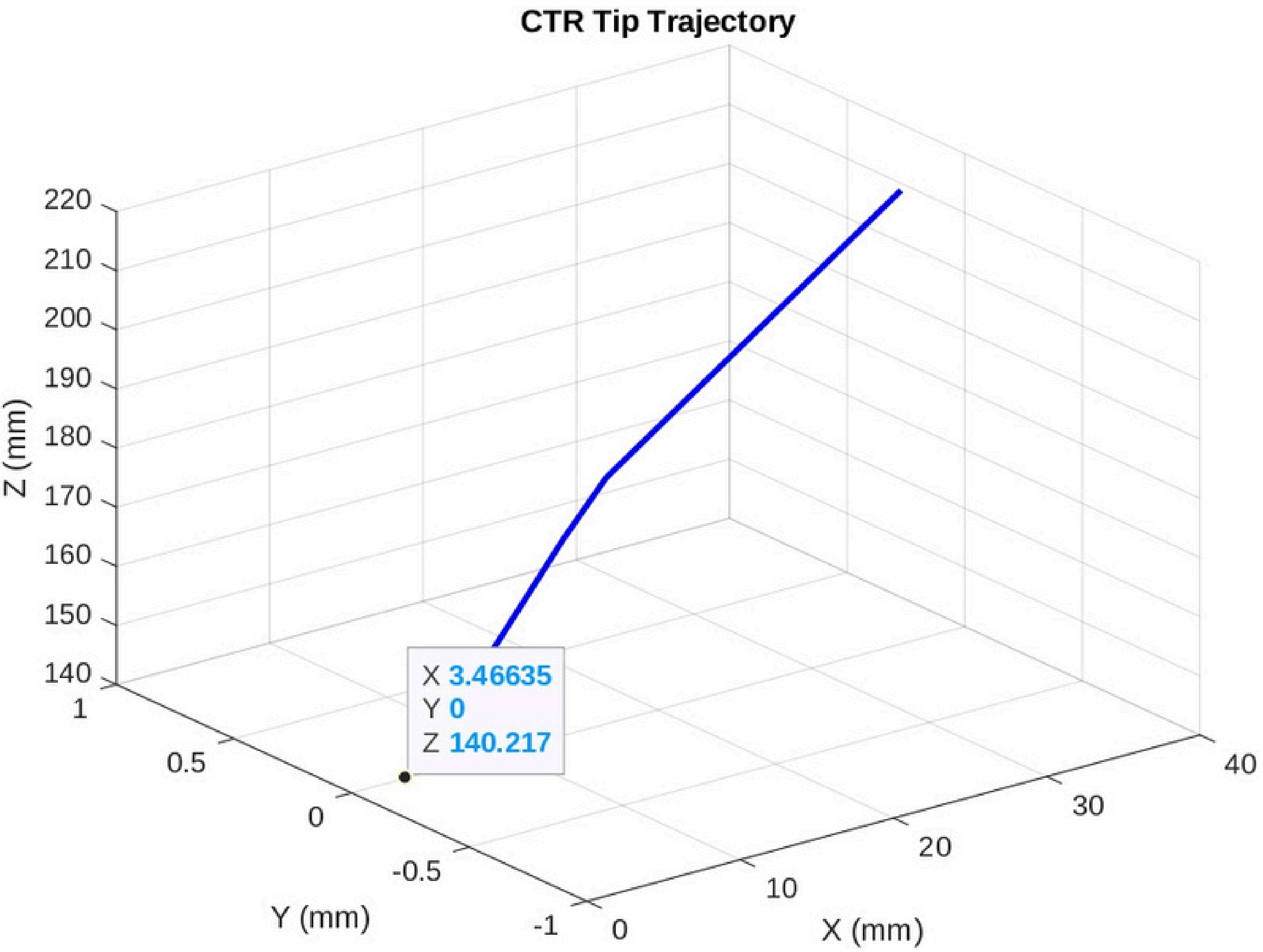
/MATLAB Drive/RM.m

```
% Import the data
data = readtable('tip_positions3.csv');

% Extract coordinates
x = data.X;
y = data.Y;
z = data.Z;

% Plot the 3D trajectory
figure;
plot3(x, y, z, 'b-', 'LineWidth', 2);
grid on;
xlabel('X (mm)');
ylabel('Y (mm)');
zlabel('Z (mm)');
title('CTR Tip Trajectory');
```

tip_positions.csv

	A	B	C	D
	Timestamp	X	Y	Z
	Number	Number	Number	Number
1	Timestamp	X	Y	Z
2	1746801179.5165	3.4664	0	140.2168
3	1746801194.2138	5.2686	0	144.4
4	1746801197.3221	13.7963	0	173.1608
5	1746801211.3592	16.5828	0	181.5749
6	1746801215.9224	35.8394	0	216.7019

Comparison of the results

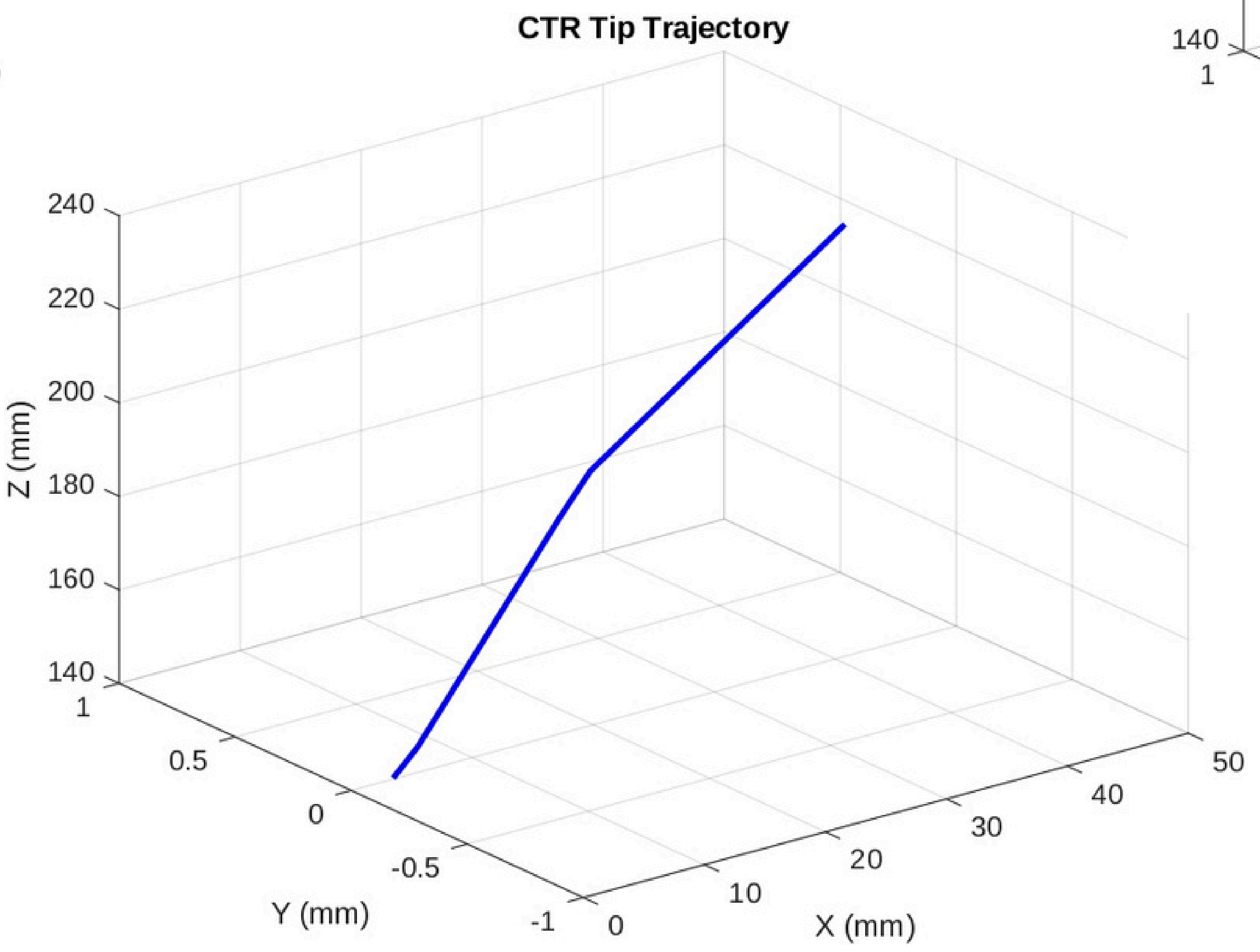


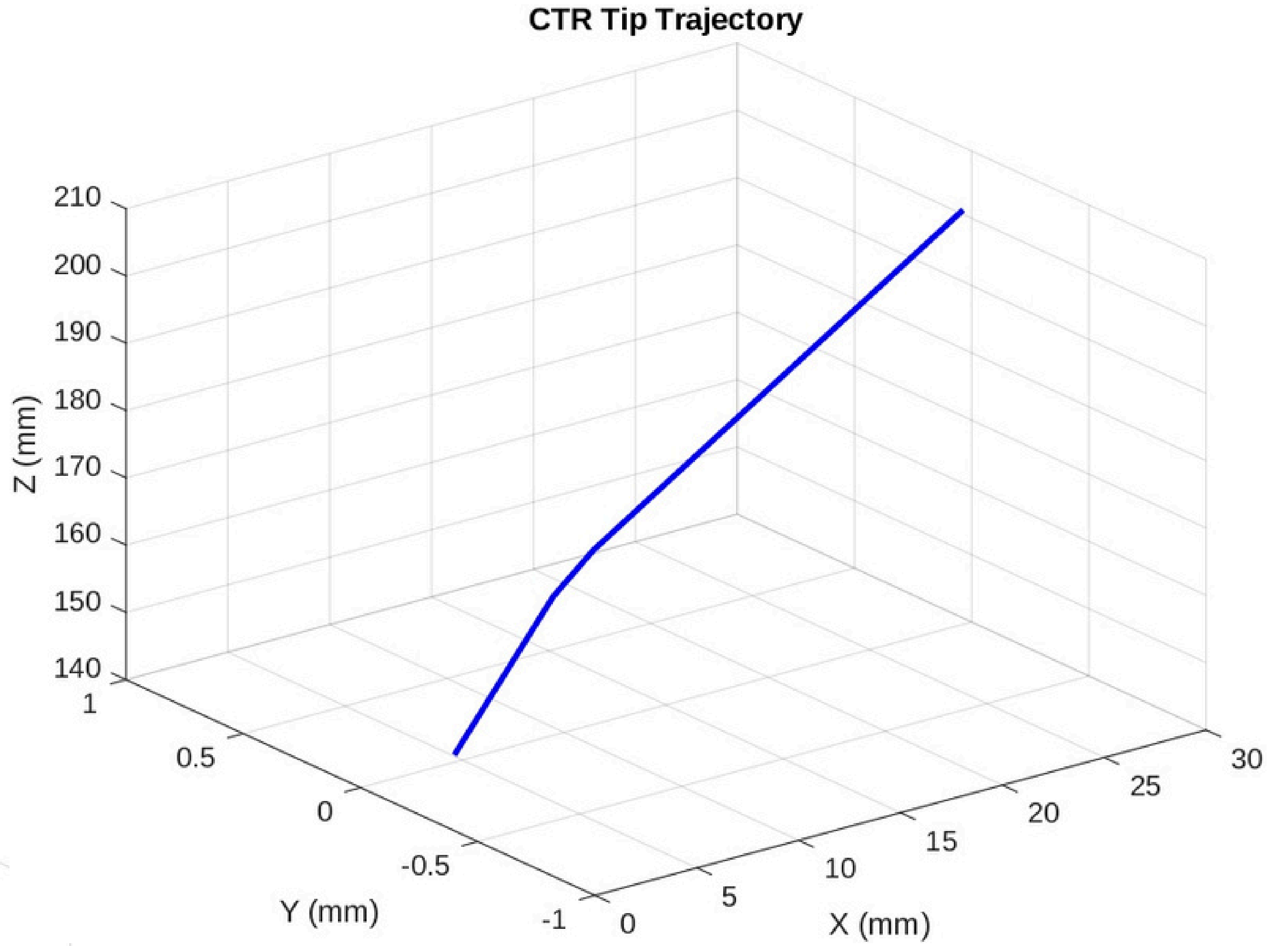
NOMINAL CASE:

- length of the innermost: **170mm**
- length of the medium tube: **135mm**

+10% CASE:

- length of the innermost: **187mm**
- length of the medium tube: **148.5mm**





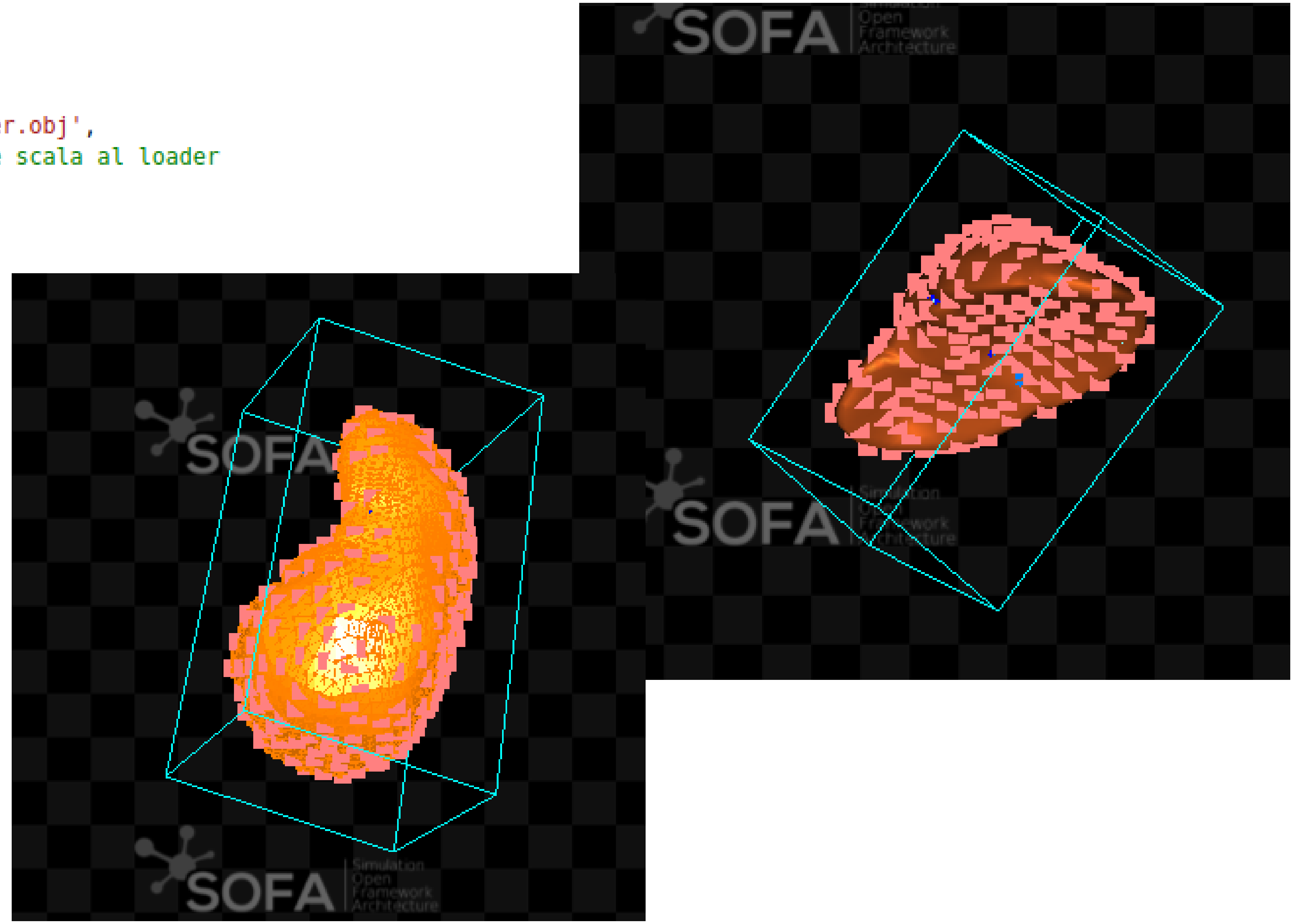
-10% CASE:

- length of the innermost: **153mm**
- length of the medium tube: **121.5mm**

Volumetric mesh & elastic object

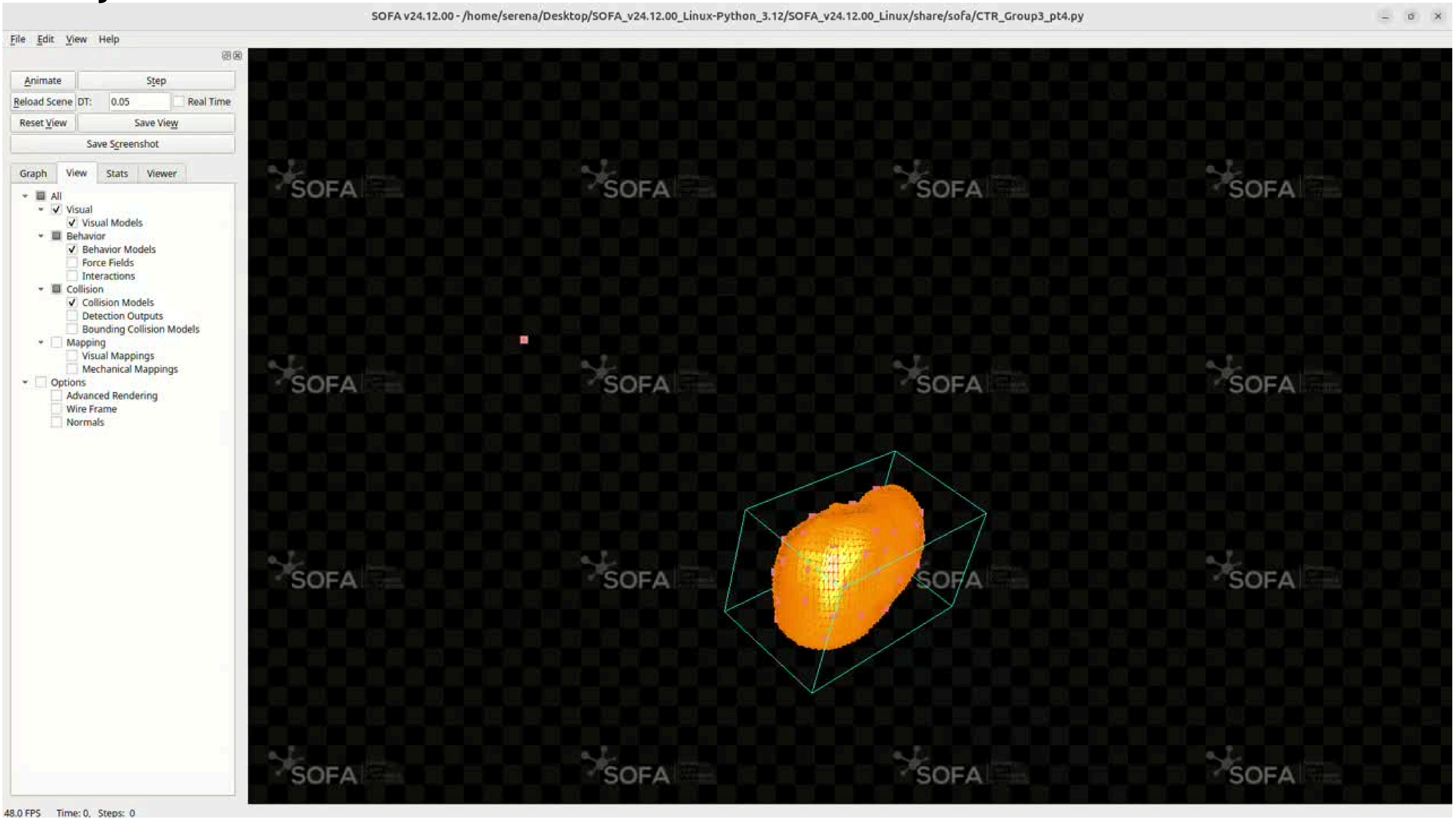
The volumetric mesh has been imported in the scene using the provided Python script. It has been employed to generate an elastic object.

```python
# Load the liver mesh
meshpath = get_mesh_path()
liver.addObject('MeshOBJLoader', name="liverLoader", filename=meshpath+'liver.obj',
                translation="5 30 200", scale="10")  # Applica traslazione e scala al loader

# Generate volumetric mesh using CGAL
liver.addObject('MeshGenerationFromPolyhedron', name="tetraGenerator",

            inputPoints="@liverLoader.position",
            inputTriangles="@liverLoader.triangles",
            inputQuads="@liverLoader.quads",
            drawTetras="1",
            facetSize="10",
            facetApproximation="10",
            cellRatio="10",
            cellSize="10")

# Create mechanical object for liver simulation
liver.addObject('MechanicalObject', name="dofs",
            position="@tetraGenerator.outputPoints")
```



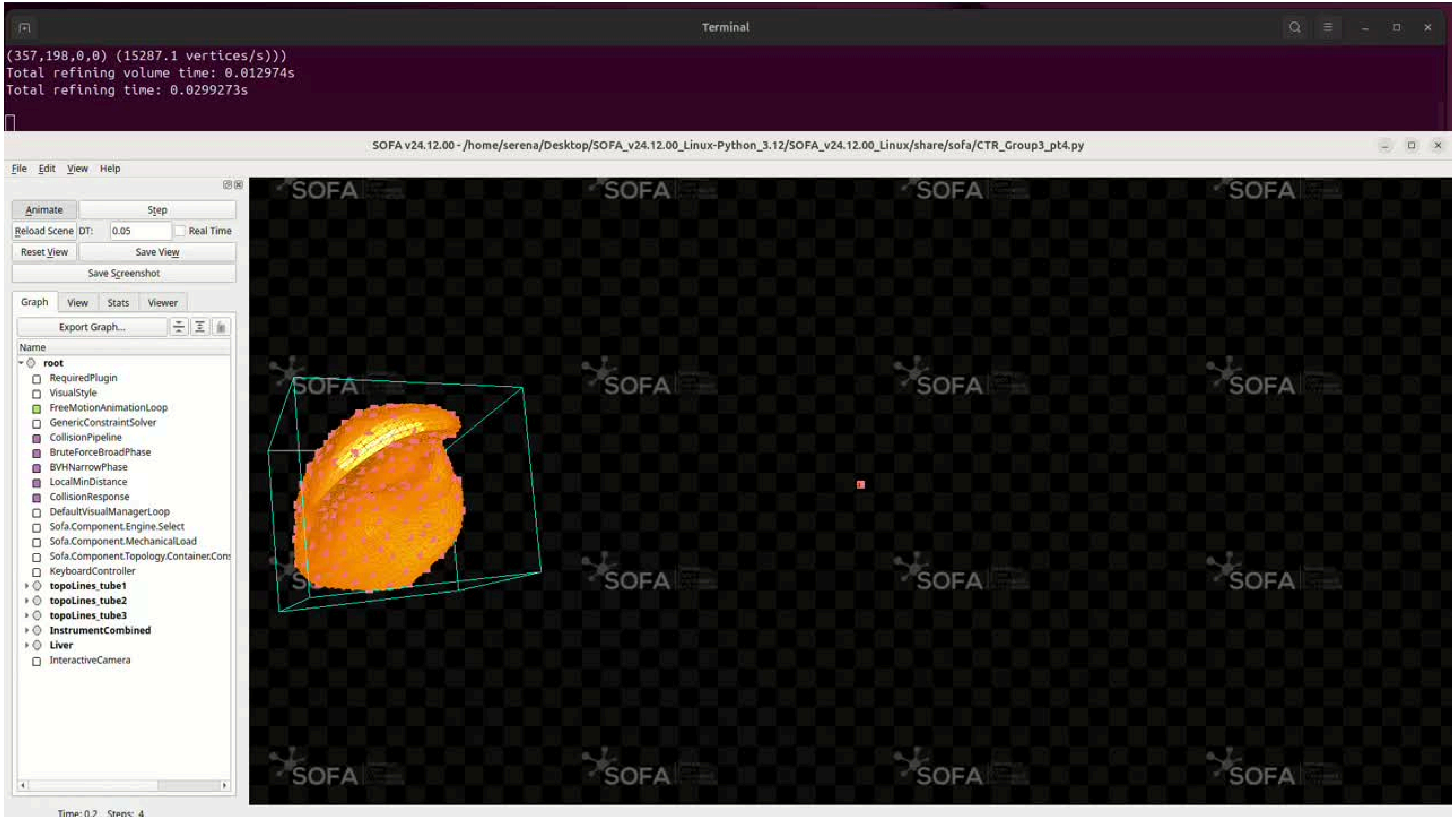
Simulation of the CTR interaction

CASE 1: stiffer object (50e6 Pa)

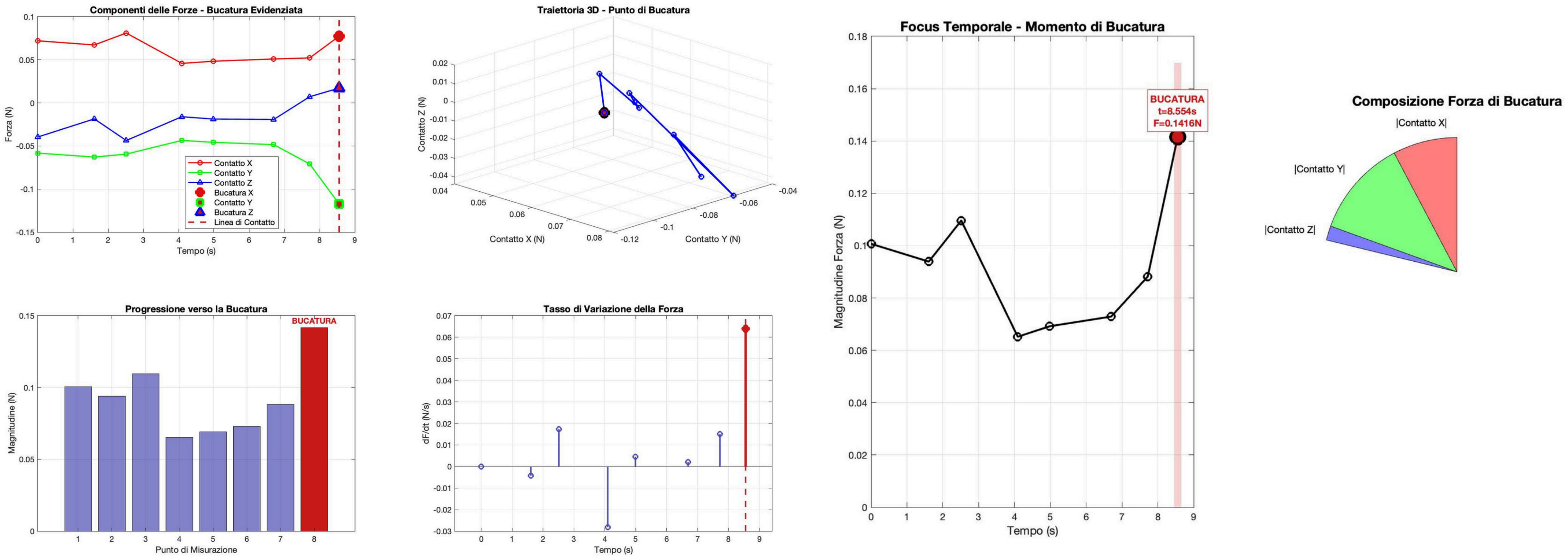


Simulation of the CTR interaction

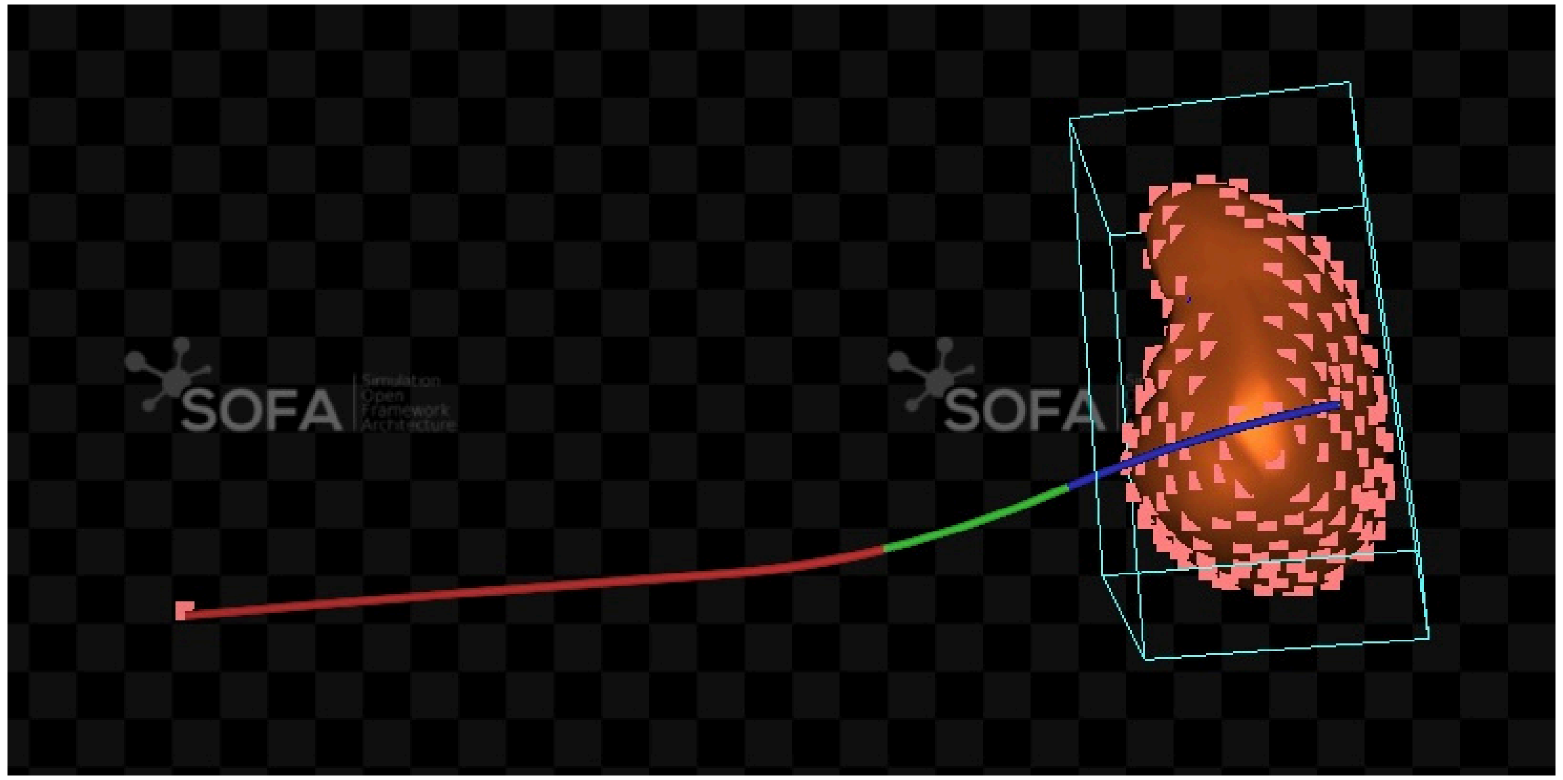
CASE 2: soft object (2e3 Pa)



Critical analysis of interaction



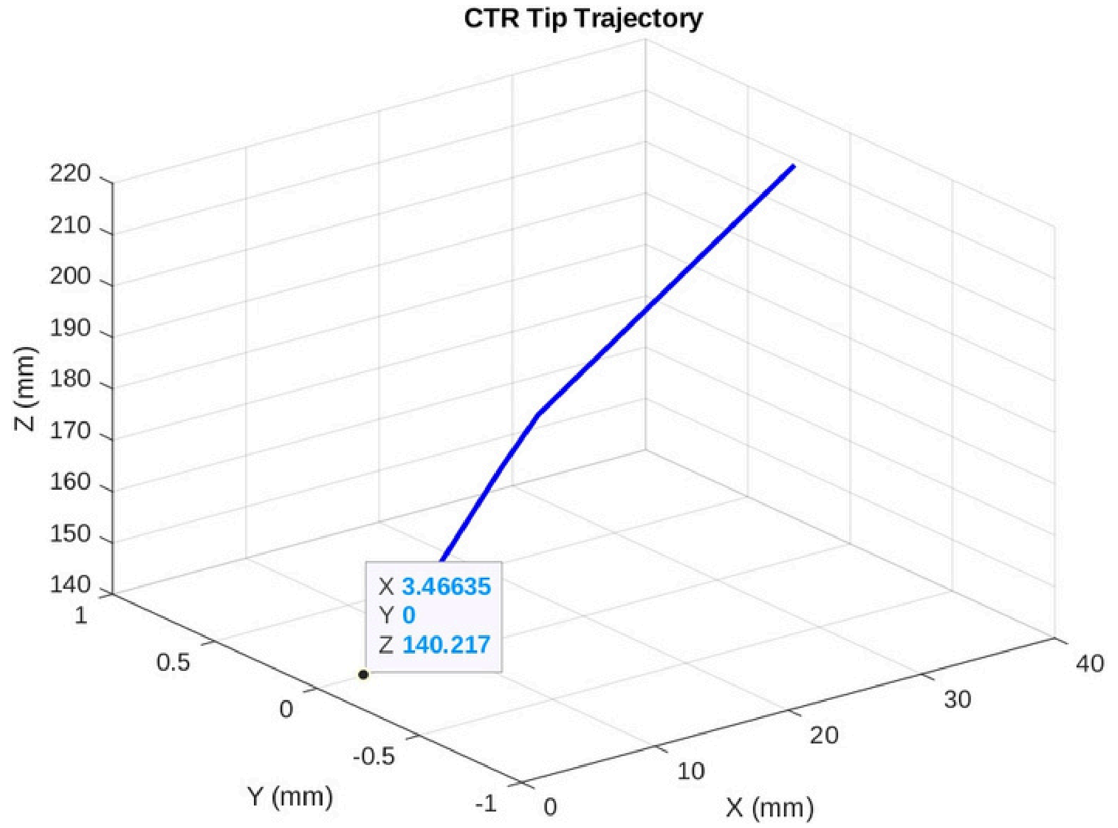
Force registration



	INNER TUBE	MEDIUM TUBE	OUTER TUBE
RADIUS [mm]	0.75	0.80 (-0.05 w.r.t. nominal case)	0.95
MASS DENSITY [kg/m^3]	6450	8000 (+1550 w.r.t. nominal case)	6450
YOUNG MODULUS [GPa]	77	77	77

Terminal

```
Registrazione forze di contatto...
Contact force at point 0: [-0.3530, 0.1943, -0.0196], Magnitude: 0.4034 N
Forze di contatto registrate nel file.
Registrazione forze di contatto...
Contact force at point 0: [-0.3528, 0.1942, -0.0196], Magnitude: 0.4032 N
Forze di contatto registrate nel file.
Registrazione forze di contatto...
Contact force at point 0: [-0.3527, 0.1942, -0.0196], Magnitude: 0.4031 N
Forze di contatto registrate nel file.
Registrazione forze di contatto...
Contact force at point 0: [-0.3526, 0.1941, -0.0196], Magnitude: 0.4030 N
Forze di contatto registrate nel file.
Registrazione forze di contatto...
Contact force at point 0: [-0.3526, 0.1941, -0.0194], Magnitude: 0.4030 N
Forze di contatto registrate nel file.
Registrazione forze di contatto...
Contact force at point 0: [-0.1573, 0.0796, 0.0617], Magnitude: 0.1867 N
Forze di contatto registrate nel file.
Registrazione forze di contatto...
Contact force at point 0: [-0.1748, 0.0891, 0.0672], Magnitude: 0.2074 N
Forze di contatto registrate nel file.
Registrazione forze di contatto...
Contact force at point 0: [-0.2798, 0.1254, -0.0158], Magnitude: 0.3071 N
Forze di contatto registrate nel file.
```



THANKS FOR THE ATTENTION