PyVista: A Python Library for Interactive 3D Data Visualization and Analysis

Tetsuo Koyama PyVista developer team

Abstract

Do you want to visualize 3D scientific data in a Pythonic way like matplotlib? If you want, this poster is for you. This poster is the introduction of PyVista. It is

- "VTK for humans" a high-level API to the Visualization Toolkit (VTK)
- 3D plotting made simple and built for large/complex data geometries
- mesh data structures and filtering methods for spatial datasets

Hello World!

In Code Listing 1, we demonstrate the "Hello World!" of PyVista. Basic step of PyVista script is the following. First, import PyVista. Then generate mesh and add it to Plotter object using add_mesh() method. And finally, we can check the render view (Figure 1) of PyVista using `show()` method.

Code Listing 1: Hello World!

```
import pyvista as pv

pv.set_plot_theme("document")

cyl = pv.Cylinder()
arrow = pv.Arrow()
sphere = pv.Sphere()

p = pv.Plotter(shape=(1, 3), window_size=[1000, 300], off_screen=True)
p.subplot(0, 0)
p.add_text("Cylinder")
p.add_mesh(cyl, color="tan", show_edges=True)
p.subplot(0, 1)
p.add_text("Arrow")
p.add_mesh(arrow, color="tan", show_edges=True)
p.subplot(0, 2)
p.add_text("Sphere")
p.add_mesh(sphere, color="tan", show_edges=True)

p.screenshot("hello_world.png")
```

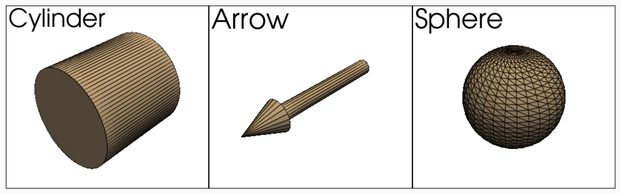


Figure 1: Hello World!

General filters to any data type

PyVista classes hold methods to apply general filters to any data type. A user can easily apply common filters in an intuitive manner. For example, Code Listing 2 shrink the individual faces of a mesh using shrink() method (Figure 2), and Code Listing 3 sweep polygonal data creating "skirt" from line using extrude_rotate() method (Figure 3).

Code Listing 2: Shrink Mesh

```
mesh = pv.Box()
shrunk_mesh = mesh.shrink(shrink_factor=0.8)
```

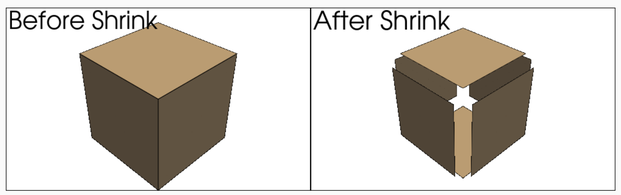
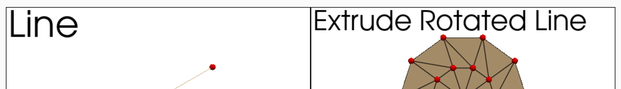


Figure 2: Shrink filter

Code Listing 3: Extrude Rotate

```
resolution = 10
line = pv.Line(pointa=(0, 0, 0), pointb=(1, 0, 0), resolution=2)
poly = line.extrude_rotate(resolution=resolution)
```



Load and plot from a files

Loading a mesh is trivial - if your data is in one of the many supported file formats, simply use pyvista.read() to load your spatially referenced dataset into a PyVista mesh object (Code Listing 4, Figure 4). Also note that we can export any PyVista mesh to any file format supported by meshio. To save a PyVista mesh using meshio, use pyvista.save_meshio()(Code Listing 5):

Code Listing 4: Load meshs from the many supported file formats

```
bunny = pv.read("bunny.ply")
cow = pv.read("cow.obj")
gears = pv.read("gears.stl")
human = pv.read("Human.vtp")
```

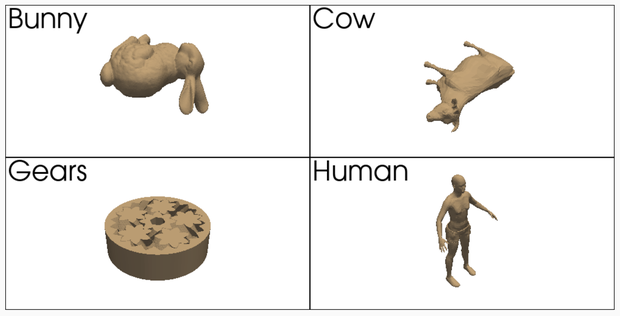


Figure 4: Meshs from the many supported file formats

Code Listing 5: Save meshs to the many supported file formats

```
# pv.save_meshio("bunny.vtk", bunny)
# pv.save_meshio("cow.stl", cow)
# pv.save_meshio("gears.obj", gears)
# pv.save_meshio("Human.ply", human)
```

Extracting and Contouring

Attributes are data values that live on either the nodes or cells of a mesh. In PyVista, we work with both point data and cell data and allow easy access to data dictionaries to hold arrays for attributes that live either on all nodes or on all cells of a mesh. Meshes can have a scalar field extracted using warp_by_scalar() method (Code List 6, Figure 5). Also can have a vector filed extracted using warp_by_vector() method (Code List 8, Figure 6). add_mesh() method can use a Matplotlib, Colorcet, cmocean, or custom colormap when plotting scalar values (Code List 7, Figure 6).

Code Listing 6: Extracted by scalar

```
warped_mesh = mesh.warp_by_scalar("Elevation")
```

Code Listing 7: Contouring by scalar

```
p.add_mesh(mesh, scalars="Elevation", cmap="hot", stitle="Matplotlib Hot")

p.subplot(0, 1)
```

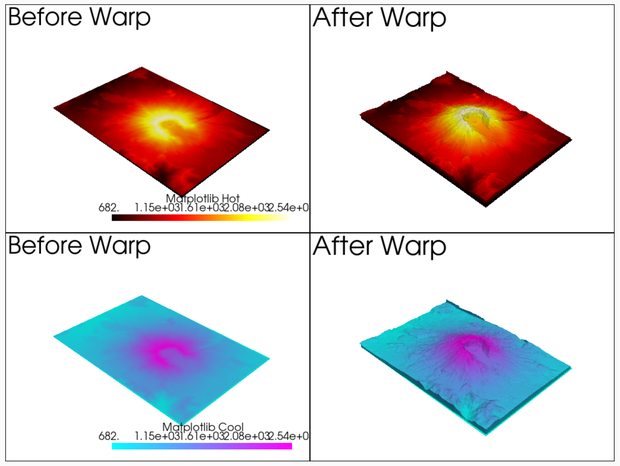


Figure 5: Contouring

Code Listing 8: Extracted by vector

```
p.add_mesh(warped, scalars=None)
```

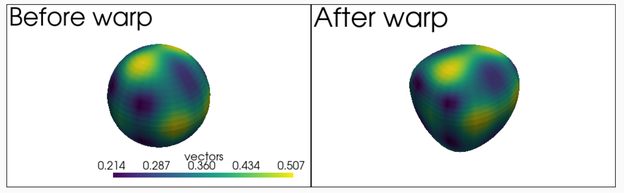


Figure 6: Warped sphere by vector

Camera class

Camera class is a virtual camera for 3D rendering. It provides methods to position and orient the view point and focal point. Convenience methods for moving about the focal point also are provided. More complex methods allow the manipulation of the computer graphics model including view up vector, clipping planes, and camera perspective (Figure 7). Code Listing 9 create a camera and frustum. Then create a scene of inside frustum adding Camera object to Plotter object (Code list 9 ,Figure 7).

Code Listing 9: Create camera frustum

```
camera = pv.Camera()
near_range = 0.3
far_range = 0.8
camera.clipping_range = (near_range, far_range)

frustum = camera.view_frustum(1.0)
```

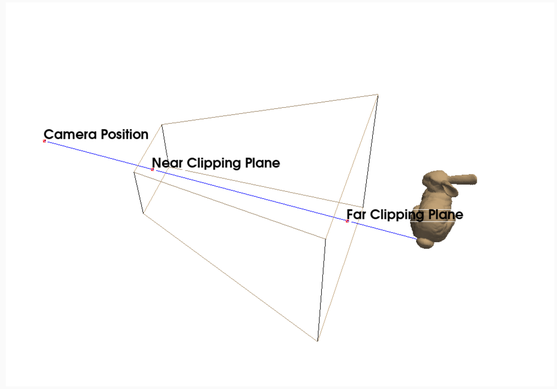


Figure 7: Frustum of camera

Code Listing 10: Add Camera to Plotter

```
camera = pv.Camera()
near_range = 0.3
far_range = 0.8
camera.clipping_range = (near_range, far_range)

p = pv.Plotter(window_size=[1000, 300], off_screen=True)
p.camera = camera
```

Figure 8: Camera view

Controlling Camera Rotation

In addition to directly controlling the camera position by setting it via the pyvista. Camera.position property, you can also directly control the pyvista.Camera.roll, pyvista.Camera.elevation, and pyvista.Camera.azimuth of the camera. (Code list 11 ,Figure 9).

Code Listing 11: Controlling Camera Rotation

```
p.camera.roll += 10

p.camera.azimuth = 45

p.camera.elevation = 10
```

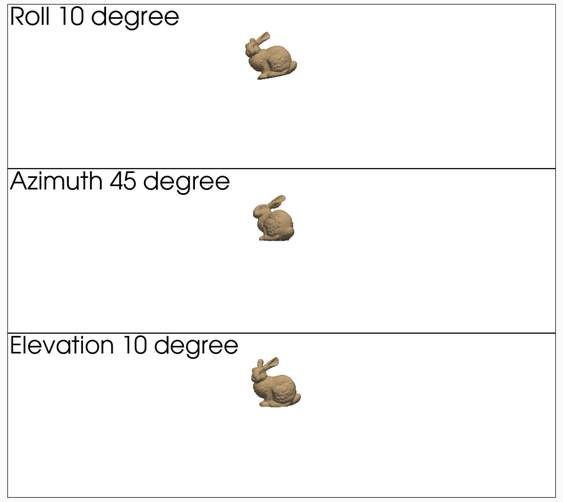


Figure 9: Controlling Camera Rotation

Plot data over circular arc

It can be plotting the values of a dataset over a circular arc through that dataset using plot_over_circular_arc_normal. (Code list 12, Figure 10 and 11).

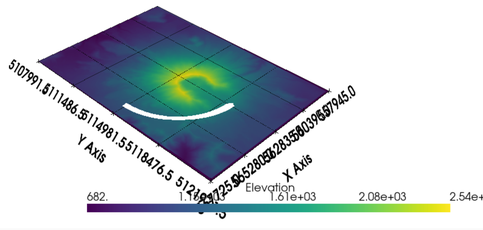


Figure 10: Circular arc to plot

Code Listing 12: Plotting over circular arc

```
normal = [0, 0, 1]
polar = [4000.0, 0.0, 0.0]
center = mesh.center
angle = 90.0
resolution = 10000

# Preview how this circular arc intersects this mesh
arc = pv.CircularArcFromNormal(center, resolution, normal, polar, angle)
```

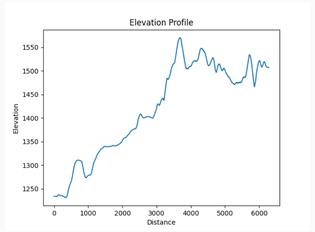


Figure 11: Plot over line

Acknowlegment

I would like to thank PyVista developer team for developing useful library.

References

C. Bane Sullivan and Alexander Kaszynski, (2019). PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK). Journal of Open Source Software, 4(37), 1450, https://doi.org/10.21105/joss.01450

Contact Information

If you want to know pyvista more, join Slack.