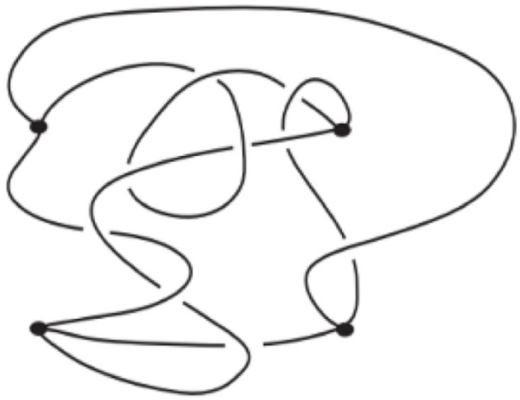
Engineering System Design Laboratory (ESDL)

# Optimal 3D Component Placement and Routing for Interconnected Engineering Systems

Gaurang Dada, Industrial Engineering, Class of 2023

---



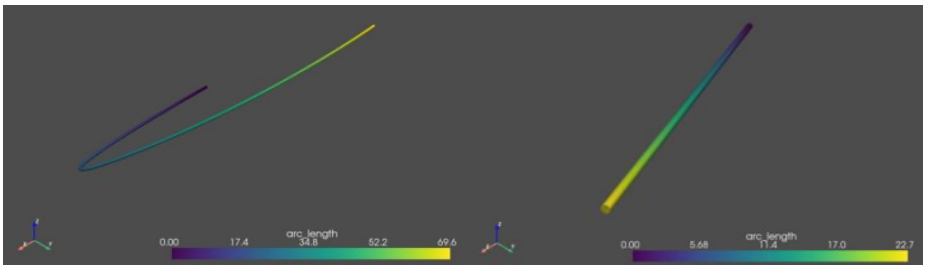
## I. Abstract

Engineering systems with spatially packaged and interconnected components interacting physically are called SPI2. From electronic chips to automobiles, such systems can be seen all around us in numerous applications. SPI2 have complex geometric interconnects, mathematically referred to as splines. The placement of components and splines in a system is a crucial engineering decision because of system interactions. Owing to multiple physical interactions and non-linear constraints in SPI2, optimal placement of components and calculated interpolation of splines are paramount to ensure SPI2 systems perform as effectively as possible. 3-D visualization of SPI2 is a constructive way of understanding and improving these engineering systems.

This paper explains all the necessary tools needed to create new or existing 3-D visualizations of SPI2 systems and discusses various practical applications of these visualizations.

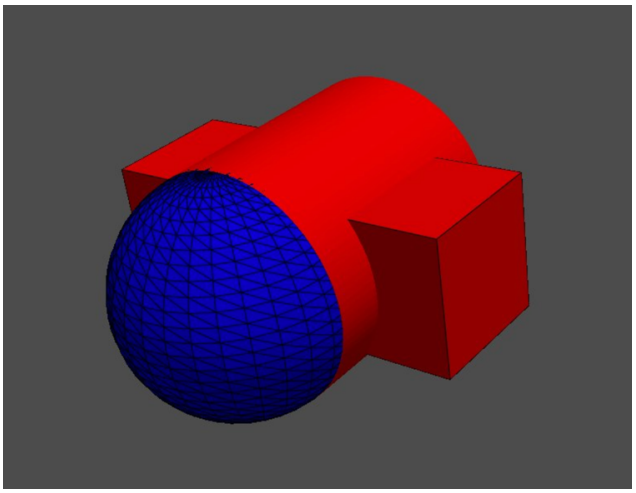
# II. Python & PyVista

Python's high-level API, PyVista, was used to create the 3-D visualizations described in this paper. PyVista is a suitable interface for obtaining 3-Dimensional models and figures. An essential factor to consider about SPI2 visualizations through PyVista is the accessibility it provides to information about the internal features of the system. Pyvista allows you to create splines using python's Numpy column stack; PyVista also provides axial information along with the length of the spline.




Quadratic Linear

As for components, the shape and orientation of a component are unique and defined by the system it exists in. All components were constructed as cylinders, spheres, or polygons. Such simple shapes can represent complex component shapes by placing them at desired spatial coordinates. We can create more complicated and realistic geometric shapes by combining these simple shapes

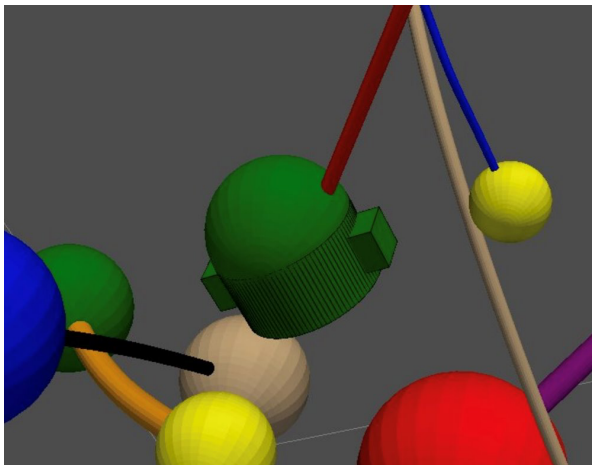


Component

Since PyVista allows users to spatially organize objects (splines and components) in a bounded space, it is extremely helpful in assessing and modelling SPI2.

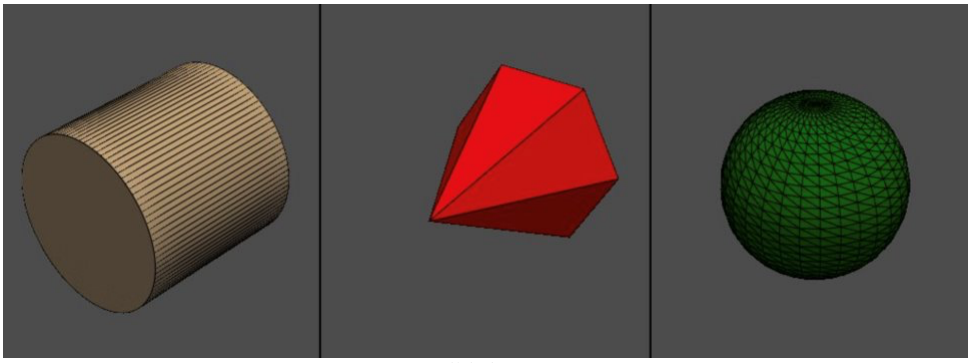
# III. Component Visualization

Component visualization was achieved using simple shapes to represent SPI2 components positionally. Just like spline objects, components take calculated geometric positions to ensure SPI2 systems perform efficiently. Since there are multiple physical interactions in SPI2, the optimal placement of components is highly consequential.



Components

The solid shapes used to represent SPI2 components were created using a component class to construct components easily. All solid shapes visible in the picture above and this paper have the following parameters: shape type, radius, centre, and colour. As mentioned, any engineering system with geometric positioning of components and splines is SPI2; covering the range of all possible engineering components is a near impossible task. Therefore, all components are shown as simple solid shapes for representational purposes.

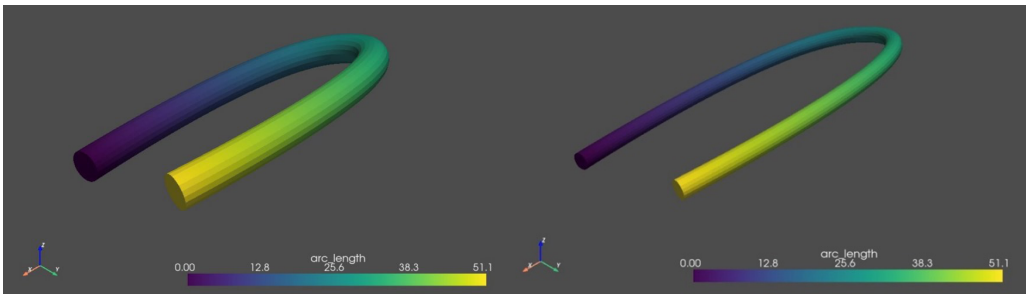


Solid shapes

# IV. Spline Visualization

## IV.1 Radius of Splines

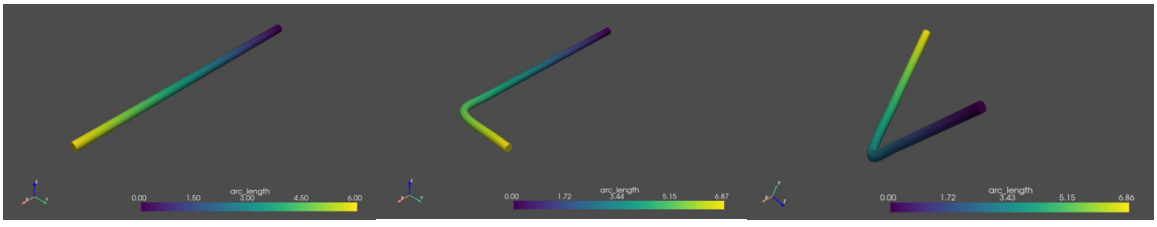
PyVista gives you the option to manipulate splines by changing the spline radius. The opportunity to create splines with the same orientation but different radii is a helpful tool for various applications. Since a radius change implies a change in volumetric flow through the spline, having access to similar splines with different radii can help assess how system interactions would change if we used thinner/thicker splines.




radius_a = 1.2 radius_b = 0.6

## IV.2 90º bend in Splines

A common feature across multiple engineering systems is a 90º bend in linearly oriented structures. Interestingly, splines can be used very efficiently to model these 90º bends seen in underground pipes, engine splines, air ducts, steel beams etc. Manipulation of control points of a spline can achieve these curves. PyVista allows you to choose a desired number of interpolation points, which helps achieve smoother 90º bends in splines.

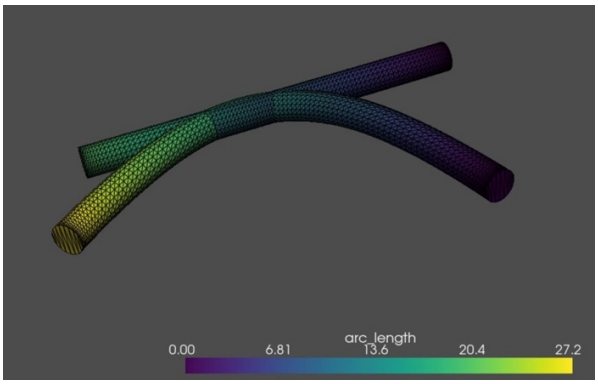


(a) (b) (c)

As shown above, PyVista also prints the length of the two splines. The 90º bends occur at different points in the initially linear spline in (a). The two splines, in (b) and (c), are achieved by bending the spline in (a) by 90º in different directions at different points along its straight-line

path. Since the arrangement of control points manipulates splines, splines in (b) and (c) have the same number of control points as (a). Instead of being arranged linearly, control points are manipulated by a function that accepts a linear spline and curves it by 90º at a specified position along its length. The algorithm perpendicularly positions the part of the spline beyond a chosen input point on the linear spline. Splines in (b) and (c) differ in length by 0.86 units, contributed by the stretch in the outer periphery of both splines. The "arc_length" description on the bottom of (b) and (c) shows the effectiveness of using splines to model angular bends.

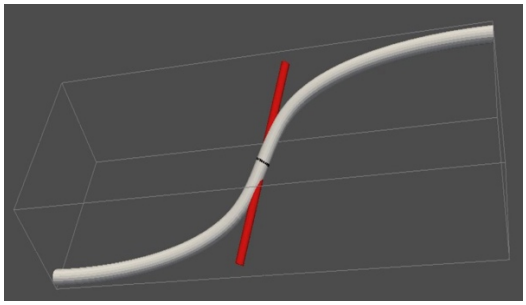
## IV.3 Intersections




(d)

SPI2 systems are characterized by the presence of complex, geometric splines. The spatial coordinates of these interconnecting splines are an essential consideration for designing optimal SPI2 systems. The complexity and geometric patterns that splines can take in SPI2 systems can expose splines to the risk of collisions. While designing an optimal SPI2 system, intersections between interconnects must be averted for the system to function.

### IV.3.1 Achieving Collision Detection

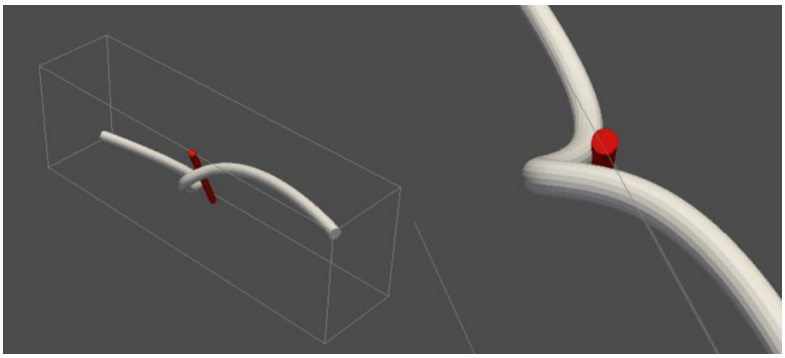


(e)

The shared spatial point, where the intersection occurs between the two column stacks of the splines, is used to centre the arc plotted cross-sectionally along one intersecting spline. Collision detection is just an intermediate step. It is necessary to avert collisions to ensure SPI2 systems work as intended.

### IV.3.2 Achieving Collision Aversion

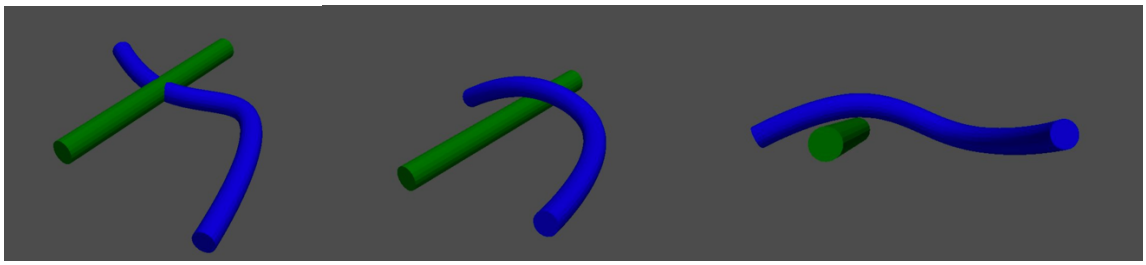
We must see SPI2 systems as they exist: collisions between interconnects must be averted when visualizing SPI2 systems. As pictorially shown in this paper, Collision aversion was achieved using an implemented function that shifts the coordinates of the detected point in a splines' column stack.



(e) (f)

Collision aversions are a vital display feature when visualizing SPI2. As shown in (e) and (f), detecting the intersecting point in space allows us to access it. Collision aversion is implemented by a function that manipulates a spline by considering the radius of the other intersecting spline. Making a radius consideration allows for effective aversion of collisions. For collisions that occur in any 2-D plane, radius considerations help to bend splines just enough to avoid the collision. The radius is used to shift the control point at the intersection just as much as the diameter of the other intersecting spline. We want to use space as efficiently as we can when designing engineering systems. Hence collision aversion by the minimum movement of intersecting splines is crucial.

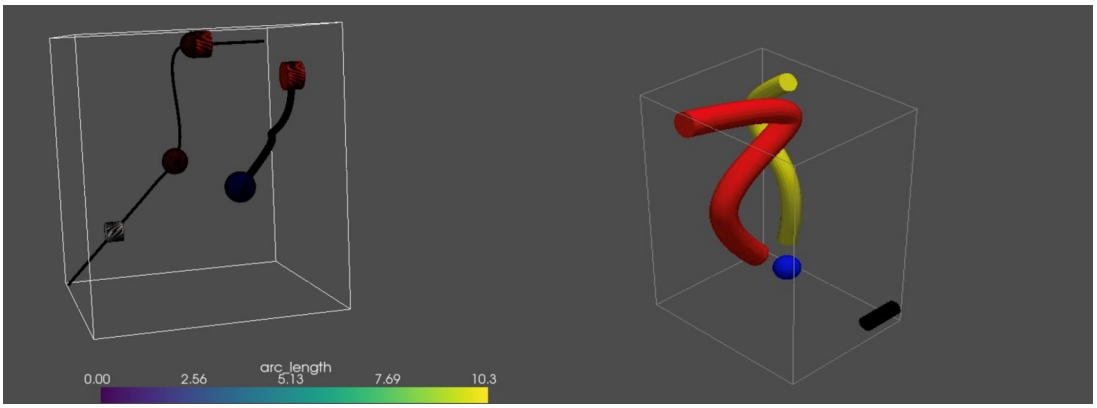
Pictures (h) and (i) below show collision aversion in 2-D



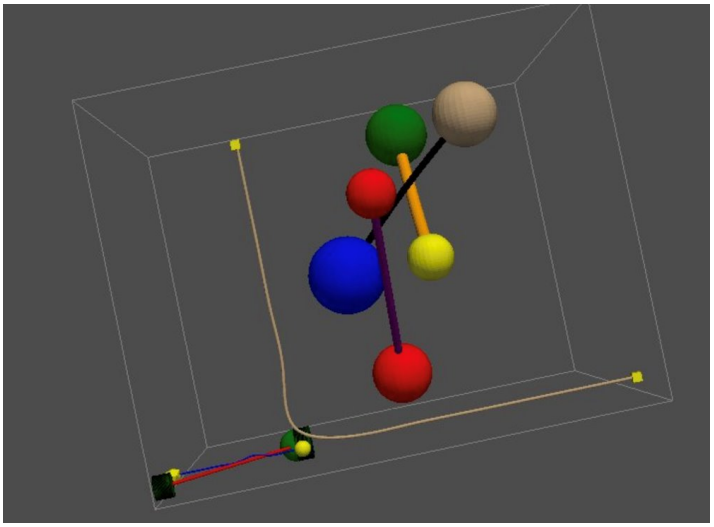
(g) (h) (i)

# V. SPI2 Visualization & GitHub

The SPI2 visualizations were created using a “scene” function that accepts object arrays of splines and components.




(j) (k)



(l)

The images (j), (k) and (l) above are bounded regions with a packing of components with interconnects. More specifically, (j) and (l) feature splines connecting solid shapes that represent components and (k) is just a random placement of two components and geometric splines (yellow and red). Such SPI2 can be created using the SPI2-Visualization public repository on GitHub.

(https://github.com/gaurangdada/SPI2-visualization.git)

The repository has a spline class and a component class. The spline class has functions that can create splines, bend by 90º, detect spline collisions, avoid spline collisions, and plot splines. Likewise, the component class has functions that let you place spheres and cylinders with chosen physical properties inside a bounded region. The component class has functions that create components, plot them, and allow us to characterize them with a shape type, centre, and colour.

# VI. References

Peddada, S. R. T., Zeidner, L. E., James, K., & Allison, J. T. (2021). An introduction to 3D SPI2 (spatial packaging of interconnected systems with physics interactions) design problems: A review of related work, existing gaps, challenges, and opportunities. In *47th Design Automation Conference (DAC)* [V03BT03A034] (Proceedings of the ASME Design Engineering Technical Conference; Vol. 3B-2021). American Society of Mechanical Engineers (ASME). https://doi.org/10.1115/DETC2021-72106