Assignment #5: 3D Steady Vector Field Visualization

Due on November 10th, before midnight

Goals and Requirements:

The goals of this assignment include (1) understanding the characteristics of 3D steady vector fields, (2) extending the arrow plots and streamline placement techniques implemented for 2D vector fields to 3D.

Use the **skeleton code you were given for Assignment 4** as the starting point for this assignment. You need to **remove** the following two lines in the init_vtk_widget() to enable 3D rotation

```
style = vtk.vtkInteractorStyleImage()
self.iren.SetInteractorStyle(style)
```

You should submit your source code and report **in a single .zip file** via Blackboard before the deadline.

Your report should include your answers to the writing questions **AND** high-quality images captured from your implemented program. In particular, **for each given 3D vector field**, please include the following in your report **for each data set**.

(1) An arrow plot (with clear arrow representation and little or no overlapping)

(2) A streamline placement result using **both** tube-based AND streamribbon based visualization. Please *provide the number of streamlines and the selected seeding strategy (uniform or random)* for each result.

A difference in this assignment when compared to the previous assignments is that I will NOT provide detailed, step-by-step instruction on how to complete the individual programming tasks. Instead, I will point you to a few example python codes that implement the similar functionality. You should now learn how to use those examples as reference to help you figure out the correct way to complete the individual tasks.

Tasks:

1. Writing questions (20 points)

1.1 What are the criteria of effective visualization of 3D vector fields? **(5pts)**

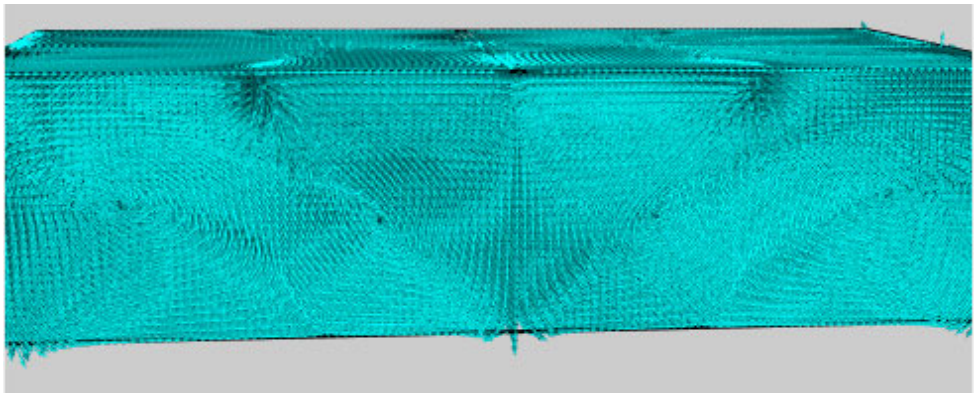
1.2 Why is it challenging to apply texture-based visualization methods for 3D vector fields? **(5 pts)**

1.3 What are the challenges faced by the geometric-based visualization for 3D vector fields? **(10 pts)**

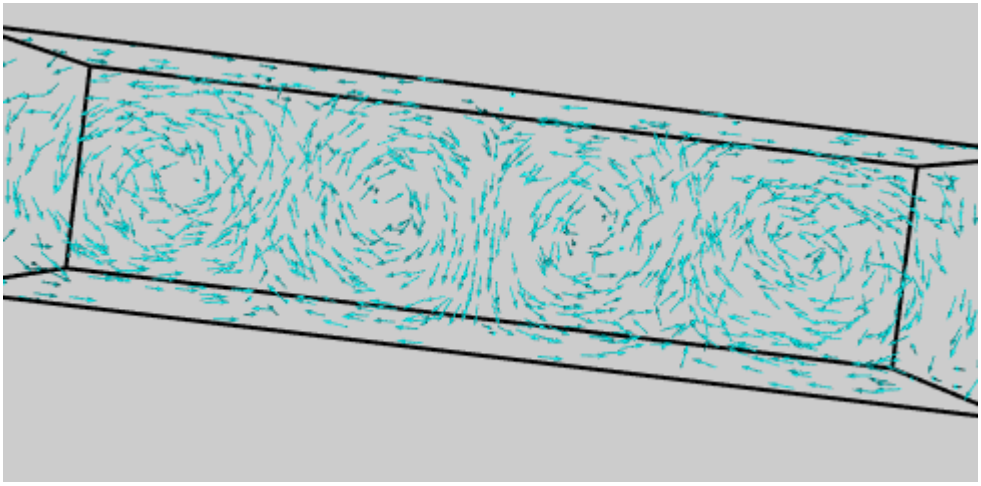
2. Generate arrow plots (30 points)

Based on the similar idea and process as 2D arrow plot and an example 3D arrow plot shown in "arrows3d_Ex.py", complete the 3D arrow plot for general 3D vector fields. Your arrow plots should reveal the flow configuration with little occlusion and clutter.

Suggestion: To produce an effective arrow plot with less occlusion and clutter, you should consider using the `vtkMaskPoints` filter to generate a sparse set of samples where the arrows will be placed. See the instruction provided in Assignment 4 on how to use this filter.
Your arrow plot will be turned on and off by toggling the "Arrow plot" checkbox on the interface.



Arrow plot of the bernard3D by placing an arrow at each grid point that leads to severe occlusion and clutter.



Arrow plot of the bernard3D data with 1000 arrows positioned at some uniform samples, using the "UNIFORM_SPATIAL_BOUNDS" random mode of the `vtkMaskPoints` filter.

3. Compute and visualize streamlines (50 points)

Next, you need to generate a number of seeds to compute a few streamlines for visualization. In this assignment, you are required to implement the following seeding and rendering strategies.

3.1 Seeds generation (30 points, 10 for each seeding strategy)

(1) **Extend the two seeding strategies** (i.e., uniform seeding and random seeding) that you have implemented in Assignment 4 for 3D streamline seeding.

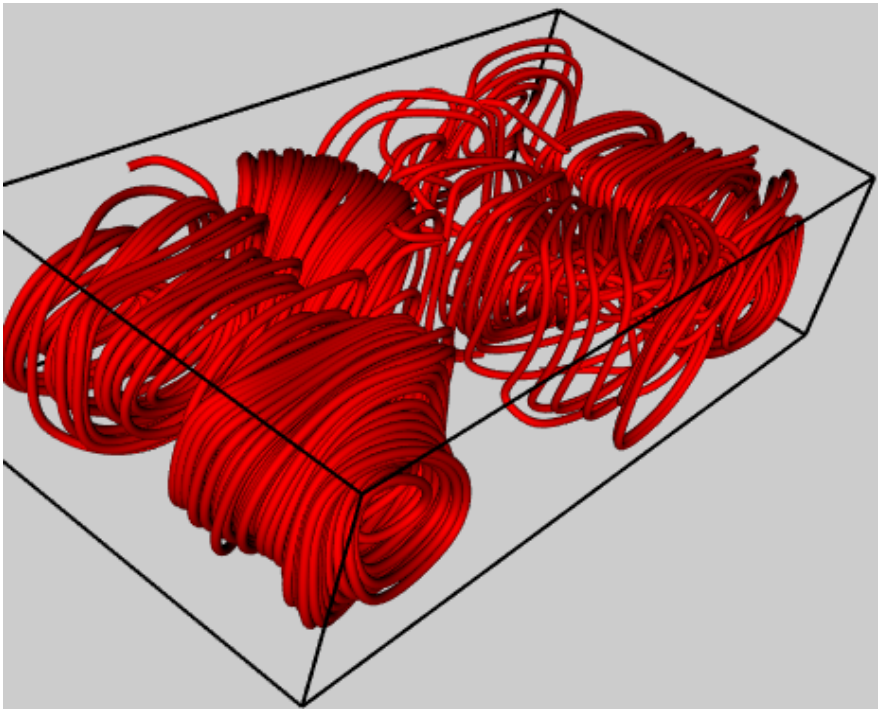
(2) Now, let us implement **a new seeding strategy by specifying a seeding curve/line** in the data domain and uniformly place seeds along it. You can look at the "StreamlinesWithLineWidget.py"

(https://lorensen.github.io/VTKExamples/site/Python/VisualizationAlgorithms/StreamlinesWithLineWidget/) on how to achieve that.

After generating the seeds, pass them to the `vtkStreamTracer` as how you did it in Assignment 4.

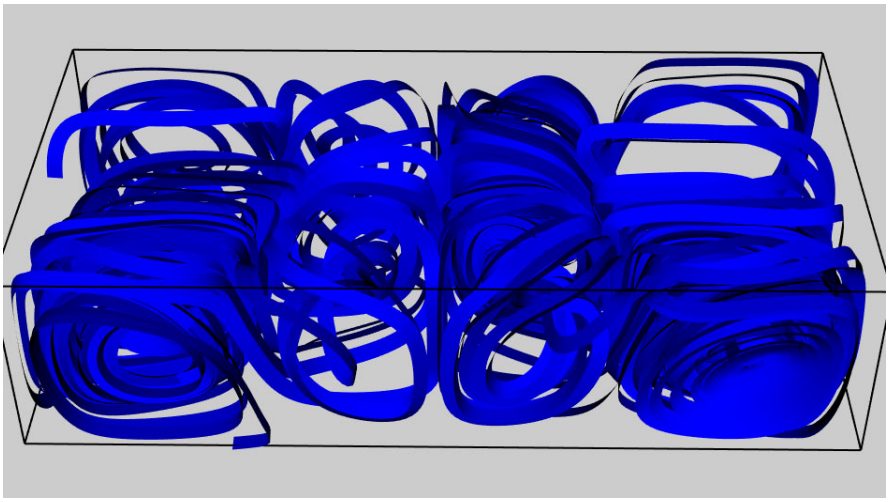
3.2 Rendering style (20 points, 10 for each of the following rendering styles)

(1) Visualize the computed streamlines using a tube-based representation. See the "**officetube.py**" (https://gitlab.kitware.com/vtk/vtk/blob/cff62a106f99c9bac3d1bc4a4e449d28b7d94285/Examples/VisualizationAlgorithms/Python/officeTube.py) for an example on how to enable the tube-based rendering of the streamlines.



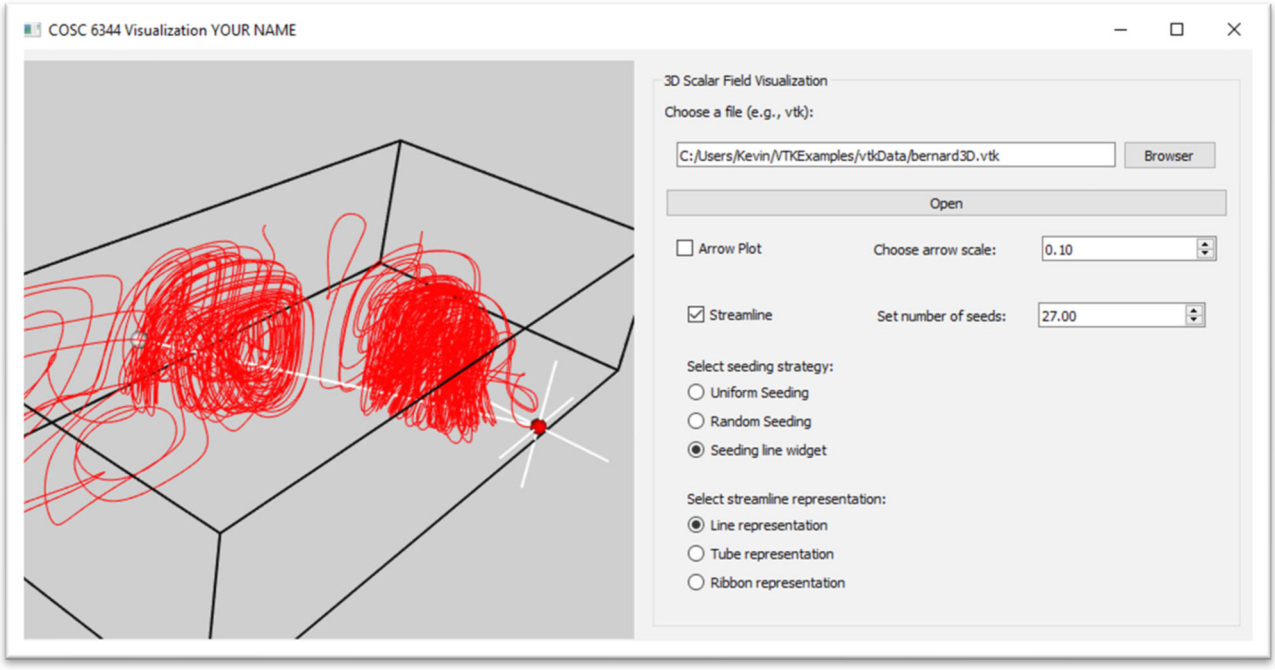
An example tube-based representation of 10 randomly seeded streamlines for the bernard3D data

(2) Visualize the computed streamlines using a ribbon-based representation. Use the `vtkRibbonFilter` (https://python.hotexamples.com/examples/vtk/-/vtkRibbonFilter/python-vtkribbonfilter-function-examples.html)



An example ribbon-based representation of 10 randomly seeded streamlines for the bernard3D data

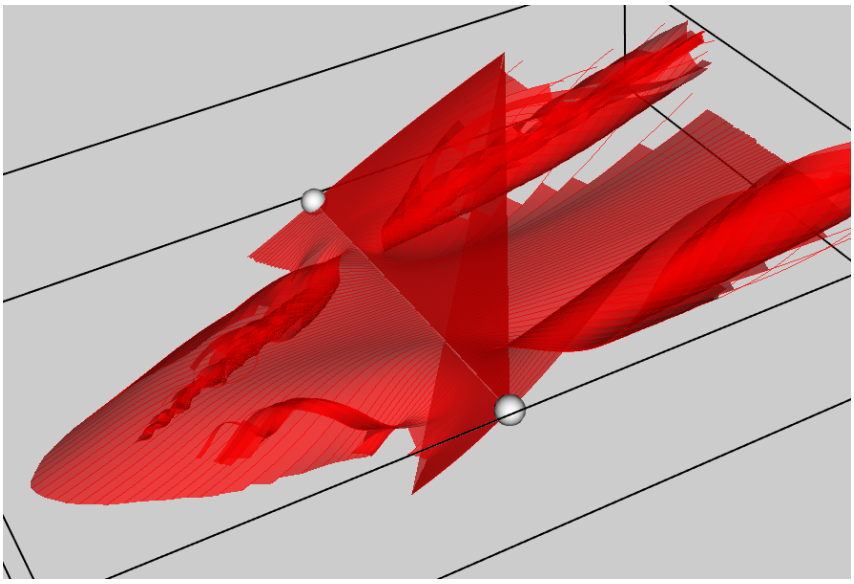
Include an interface to allow the user to select among different seeding strategies and streamline rendering style.



An example interface design for your reference. Feel free to design your own interface.

4. Stream surface placement and construction (Extra credits: 10)

Use the above line widget and `vtkRuledSurfaceFilter` following the example (https://vtk.org/gitweb?p=VTK.git;a=blob;f=Examples/VisualizationAlgorithms/Python/streamSurface.py) to construct a good stream surface to depict each of the given 3D vector field.



An example surface for the delta wing data

Add an interface (e.g., a checkbox) to enable the interaction and construction of the stream surface.

Note that using the `vtkRuledSurfaceFilter` to stitch the neighboring streamlines to form a stream surface **does not always produce the correct surface** (as shown in the above example). This is fine for this assignment.

Grading rubric:

Tasks	*Total points*
1	20
2	30
3	50
4(bonus)	***10***