

Journal of Visualized Experiments

Interactive and Immersive Visualization of Fluid Dynamics using Virtual Reality

--Manuscript Draft--

Article Type:	Invited Methods Article - JoVE Produced Video
Manuscript Number:	JoVE61151R1
Full Title:	Interactive and Immersive Visualization of Fluid Dynamics using Virtual Reality
Section/Category:	JoVE Engineering
Keywords:	Fluid dynamics; Virtual reality; water tunnel; Unity; visualization; immersive education
Corresponding Author:	Melissa Green UNITED STATES
Corresponding Author's Institution:	
Corresponding Author E-Mail:	greenma@syr.edu
Order of Authors:	Zackary Boone Amber Bartosh Melissa Green
Additional Information:	
Question	Response
Please indicate whether this article will be Standard Access or Open Access.	Standard Access (US\$2,400)
Please indicate the city, state/province, and country where this article will be filmed . Please do not use abbreviations.	Syracuse, NY USA

TITLE:

Interactive and Immersive Visualization of Fluid Dynamics using Virtual Reality

AUTHORS AND AFFILIATIONS:

Zackary Boone¹, Amber Bartosh², Melissa A. Green¹

¹Department of Mechanical and Aerospace Engineering, Syracuse University, Syracuse, NY, USA

²School of Architecture, Syracuse University, Syracuse, NY, USA

Email addresses of co-authors:

Zackary Boone (zrboone@syr.edu)

Amber Bartosh (abartosh@syr.edu)

Corresponding author:

Melissa A. Green (greenma@syr.edu)

KEYWORDS:

Fluid dynamics, virtual reality, water tunnel, Unity, visualization, immersive education

SUMMARY:

The article describes a method to visualize three-dimensional fluid flow data in virtual reality. The detailed protocol and shared data and scripts do this for a sample data set from water tunnel experiments, but it could be used for computational simulation results or 3D data from other fields as well.

ABSTRACT:

The last decade has seen a rise in both the technological capacity for data generation and the consumer availability of immersive visualization equipment, like Virtual Reality (VR). This paper outlines a method for visualizing the simulated behavior of fluids within an immersive and interactive virtual environment using an HTC Vive. This method integrates complex three-dimensional data sets as digital models within the video game engine, Unity, and allows for user interaction with these data sets using a VR headset and controllers. Custom scripts and a unique workflow have been developed to facilitate the export of processed data from Matlab and the programming of the controllers. The authors discuss the limitations of this particular protocol in its current manifestation, but also the potential for extending the process from this example to study other kinds of 3D data not limited to fluid dynamics, or for using different VR headsets or hardware.

INTRODUCTION:

Virtual reality (VR) is a tool that has seen increasing levels of popularity as it provides a new platform for collaboration, education, and research. According to Zhang et al., "VR is an immersive, interactive computer-simulated environment in which the users can interact with the virtual representations of the real world through various input/output devices and sensory channels¹." Fluid dynamics is a field of physics and engineering that attempts to describe the

45 motion of liquids or gases due to pressure gradients or applied forces. Because the study of fluid
46 dynamics is driven by the analysis of large data sets, a significant part of its research process
47 hinges on visualization. This visualization poses a difficult task due to the potential size of the
48 datasets and the four-dimensional (4D) character (3D in space + time) of the data. This complexity
49 can hinder commonly used methods of visualization that are typically constrained to flat screens,
50 limited in scale, and primarily allow the viewing of data from a removed perspective². To
51 overcome some of the challenges that impede effective visualization, this paper describes a
52 method for processing 3D fluid flow data sets for import and interaction in virtual reality.
53 Especially in the case of fluid dynamics, VR may give the user a more intuitive understanding of
54 data patterns and processes that could otherwise be difficult to detect when exploring the 3D
55 data via a 2D screen. Further, VR can pose as a novel educational tool for students because it
56 provides an opportunity to reinforce ideas that they are learning in a traditional classroom setting
57 in a new format^{3,4}.

58
59 To-date implementation of VR for fluid dynamics research has relied on performing
60 computational fluid dynamics (CFD) with in-house or off-the-shelf software packages and using
61 various techniques to import these simulations and results into VR environments. For example,
62 Kuester et al. created simulation code that continually injected particles into a flow field⁵. From
63 there, they calculated the velocity of the particles which allowed them to visualize the particles
64 in a VR toolkit called VirtualExplorer. The Matar research group at Imperial College London⁶ has
65 developed a broader range of tools to interrogate additional quantities such as pressure density
66 in the flow field, and they incorporate audio feedback as well. Still, their data is largely referenced
67 on particle points or trajectories. Use of VR in fluid dynamics tends to follow this pattern of
68 calculating particle streaks or streamlines in the highly resolved CFD of flow over immersed
69 objects, e.g. a car or a cylinder, to reveal the structure and organization of the fluid motion^{7,8,9}.

70
71 As an alternate to the particle-based modeling techniques, many 3D fluid dynamics results are
72 modeled as 3D isosurfaces. These models are digitally heavier than point-based systems and
73 require significant time and a number of steps to extract from the conventional software tools.
74 The advantage of integrating these volumetric visualizations into VR is that they enable a user to
75 virtually put themselves “inside” the measured or simulated domain. Isosurfaces in volumetric
76 data are commonly used in both computational, and increasingly, experimental results. With the
77 increasing prevalence of tomographic and holographic particle image velocimetry experiments
78 that are acquiring volumetric data sets in a single run, methods to visualize and interrogate these
79 fields will only grow in popularity.

80
81 Some of the previous implementations of VR with fluid dynamic data have relied on the creation
82 of a software package written in-house by the researchers, which can be limited in terms of
83 customization and user interaction^{8,10}. There has not been widespread implementation of these
84 packages in other research groups since their creation. In this paper, we describe a method and
85 share software that imports 3D experimental data that was generated by interpolating multiple
86 planes of stereoscopic PIV. Its conversion into digital objects in VR uses three popular software
87 packages: Matlab, Blender, and Unity. A flowchart of how the data objects are processed through
88 these software packages is shown in **Figure 1**.

Unity, a video-game engine that is free for personal use, offers interesting opportunities to envision and potentially to gamify fluid dynamic data sets with an abundance of resources that allow for representation of objects and customization of features that other packages lack. The method begins in Matlab, processing data from the standard PLOT3D format¹¹, computing three quantities (x -velocity u , y -velocity v , and z -vorticity ω_z), and then outputting the data into the .obj file format along with an associated .mtl file for each .obj file that contains the color information for the isosurfaces. These .obj files are then rendered and converted to the .fbx file format in Blender. This step can seem superfluous, but avoids difficulties that the authors encountered by ensuring that .mtl files correctly align with their associated .obj files, and that the colors of the isosurfaces are coordinated properly. These .fbx files are then imported into Unity, where the data is treated as an object that can be manipulated and toggled. Finally, the VR user interface has been designed to allow for the user to move the data in space and time, control visibility settings, and even create drawings of their own in and around the data without removing the VR headset.

[Place **Figure 1** Here]

Data set

The data set used to demonstrate the process of 3D data visualization in virtual reality is from water tunnel experiments that measured the wake of a simplified fish caudal fin geometry and kinematics¹². This work entailed experiments that used similar equipment and methods as described by King et al.¹³, in which a generic caudal fin was modeled as a rigid, trapezoidal panel that pitched about its leading edge. Stereoscopic particle image velocimetry was used to measure the three-component velocity fields in the wakes produced by the fin model, and these experiments were performed in a recirculating water tunnel located at the Syracuse Center of Excellence for Environmental and Energy Systems. Additional details about the experimental data acquisition can be found in Brooks & Green¹² or King et al.¹³.

The 3D surfaces used to visualize the structure of the fluid motion are first rendered in Matlab. The structure and organization of the wake is visualized using isosurfaces of the Q -criterion. The Q -criterion, also referred to as Q , is a scalar calculated at every point in 3D space from the three-component velocity fields, and is commonly used for vortex identification. These surfaces are then colored using spanwise vorticity (ω_z) to highlight and distinguish the large-scale structures of interest, mainly oriented in the spanwise (z) direction. Other quantities of interest include the streamwise (x -direction) velocity (u) and the transverse (y -direction) velocity (v). Visualization of isosurfaces of these quantities can often give context as to how the fluid has been accelerated by the motion of the fin model.

Example isosurfaces for each of these quantities, taken at the same instance of time, are shown in **Figure 2**. Isosurfaces of Q are shown in **Figure 2A**. The trapezoidal fin model is shown in black. The isosurface level was chosen to be 1% of the maximum value, which is commonly used in 3D vortex visualizations to reveal structures while avoiding noise that can be prevalent at lower levels. Blue coloration indicates negative rotation of the vortex tube, or clockwise rotation (when

viewed from above).

[Place **Figure 2** Here]

Red indicates positive rotation of that vortex tube, or counter-clockwise. Green indicates rotation in a direction other than along the z-axis. In **Figure 2B**, blue isosurfaces indicate streamwise velocity that is 10% lower than the freestream value and orange indicates streamwise velocity that is 10% higher than the freestream value. In **Figure 2C**, red isosurfaces indicate transverse, or cross-stream velocity that has a magnitude 10% of the freestream value but negative (out of the page) and green indicates streamwise velocity has a magnitude 10% of the freestream value and positive (into the page). These velocity isosurfaces have been used to show how the vortex ring structure is consistent with the periodic acceleration and deceleration of the fluid due to the motion of the fin model.

In the following protocol, we detail the steps for converting these isosurface visualizations into surface objects in 3D space that can be processed and imported into Unity for visualization and interaction in virtual reality.

PROTOCOL:

1. Processing data files and exporting objects from Matlab

NOTE: The following steps will detail how to open the shared Matlab processing codes, which open PLOT3D data files and generate the isosurfaces as described. The necessary .obj and .mtl files are then also constructed and exported from Matlab.

1.1. From the supplementary files, download the scripts labeled **create_obj.m** and **write_wobj.m**. Open Matlab, then open the **create_obj.m** and the **write_wobj.m** files.

1.2. Customize the path **FileDir** to the folder where scripts and data files are saved. In the File Explorer where **FileDir** is located, create a new folder called '**FullSetOBJ**'.

1.3. Customize the path of **GridDir.vTo** make this code compatible with user data, make sure the grid information is in a folder that is a child of **FileDir**.

1.4. Customize the path variable **FileName**. This should be the folder where the data files of all 24 time steps are located.

1.5. Run the script.

NOTE: The code will take several hours to complete if all quantities at all time steps are solved for and converted in one execution of the script. Thus, it may be of interest to the user to only run one quantity at a time by commenting out the sections that create the .obj files for the other two quantities and allowing the code to run on only that single quantity. Also, the user can change

the number of time-steps created by changing how many iterations the for loop cycles through, which is located at the beginning of the script.

2. Rendering surfaces in Blender

NOTE: In this section, we detail the steps for importing the output object data from Matlab into Blender, where the objects are scaled and then exported in a file format that is compatible with Unity. The shared scripts execute these tasks.

2.1. Use Blender version 2.79 and download the provided Blender scripts in the supplementary files. Double click on the **exportAsFBX_XVel** Blender script, which will open Blender.

2.2. Customize the path of **file_path**. This path should be directed to where the .obj files are located. In **File Explorer**, where **file_path** is directed and under the **FullSetOBJ** folder, create a new folder called 'FBX'.

2.3. Press **Run Script** which is located just above the interactive console. Open the **exportAsFBX_YVel** Blender script, customize the **file_path**, and click **Run Script**. Open the **exportAsFBX_ZVort** Blender script, customize the **file_path**, and click **Run Script**.

3. Configure Unity, the SteamVR Plugin, and VRTK

NOTE: These steps will ensure that the necessary software is present and configured correctly to import the data and generate the related games. Please note the software versions recommended in these steps, as we cannot guarantee that the shared scripts will work with updated versions of these particular applications.

3.1. Use Unity version 2018.3.7. Create a new 3D scene by choosing 3D under the template option and pressing the Create Project button.

3.2. Do not download the new version of Unity if prompted. Use version 1.2.3. of the SteamVR plugin from the Unity Asset store.

3.3. Once downloaded, drag the **SteamVR folder** into **Assets** on the Unity UI. If prompted to update API for SteamVR, click the button labeled "**I Made A Backup. Go Ahead.**"

3.4. Use the current version of VRTK (version 3.3.0 was used for this manuscript) from the Unity Asset store and import into the game.

3.5. Drag the folder containing the .fbx files into **Assets** on the Unity UI.

3.6. Go to the **Asset Store** and type '**simple color picker**'¹⁴. Click on the option given with this name and click import. A pop-up screen will appear showing everything that will be imported.

Check all boxes except the box label **Draggable.cs**, and then click **import**.

3.7. Create a folder in the **Assets UI** under the **Project** tab, which is located at the bottom left corner of the screen, and call it **Scripts**.

3.8. From the supplementary files, download the scripts labeled **Left_Controller**, **Right_Controller**, **Set_Text_Position**, **MeshLineRenderer**, **DrawLineManager**, **ColorManager** and **Draggable** along with their associated meta files. Then place these in the **Scripts** folder under **Assets** in the **Project** tab.

4. Creating the data visualization game in Unity

NOTE: After successfully configuring Unity in step 3, these steps will use the provided scripts to build the game that can then be saved as a standalone executable launch file.

4.1. Disable the **Main Camera** by clicking on it in the section labeled **SampleScene** in the Hierarchy tab and unchecking the box that is next to the title, **Main Camera**, in the **Inspector**.

4.2. In the **Project** tab, go into **Assets** and open the **SteamVR** folder. Then go into **Prefabs** and drag the **SteamVR** and the **CameraRig** prefab into the **SampleScene**.

4.3. Click on **CameraRig**, and then click the **Add Component** button located at the bottom of the **Inspector** menu. Click **Scripts**, and finally click on the '**GameBuild**' script.

4.4. In the **Project** tab open the folder containing fbx files and drag each of them into the **SampleScene** section.

4.5. Add a plane by going to the upper tab, go to **GameObjects | 3D Object | Plane**.

4.6. Add a **Canvas** by going to **Component | Layout | Canvas**.

4.7. Highlight each .fbx file in the Hierarchy.

4.7.1. Click on **Component | Physics | Rigidbody**. Make sure **Use Gravity** and **Is Kinematic** do not have check marks next to them.

4.7.2. Click on **Component | Physics | Box Collider**. Make sure **Is Trigger** is checked.

4.8. Under **SampleScene**, click the drop-down arrow next to **CameraRig**. Then click the drop-down arrow next to **Controller (right)** and click on **Controller (right)**.

4.8.1. Click **Add Component | Mesh | Mesh Renderer**.

4.8.2. Click **Add Component | Scripts | Right_Controller**.

4.9. Under **SampleScene**, click the drop-down arrow next to **CameraRig**, then click the drop-down arrow next to **Controller (left)** and click on **Controller (left)**.

4.9.1. Click **Add Component** | **Physics** | **Rigidbody**. Uncheck **Use Gravity** and check **Is Kinematic**.

4.9.2. Click **Add Component** | **Mesh** | **Mesh Renderer**.

4.9.3. Click **Add Component** | **Physics** | **Sphere Collider**. Check **Is Trigger** and change the radius to **Infinity** by typing, **Infinity**, into the space next to radius.

4.9.4. Click **Add Component** | **Scripts** | **Left_Controller**.

4.9.5. Click **Add Component** | **Scripts** | **Steam VR_Laser Pointer**.

4.10. Click **Assets** under the **Project** tab and drag the **ColorPicker** into the **SampleScene**.

4.10.1. Click on the **ColorPicker** in the **SampleScene** and in the **Inspector**, change the scale to X = Y = Z = 0.003 and change the positions to X = 0, Y = 1, Z = 1.4

4.11. Add an empty game object by right clicking in the **SampleScene** Hierarchy and clicking **create new object**. Rename this object to **DrawLineManager**.

4.11.1. In the **Inspector** for the **DrawLineManager** object, go to **Add Component**, click on **Scripts**, then add the **DrawLineManager** script. Then drag **Controller (right)** into the **Tracked Obj** space in the **Draggable (Script)** section.

4.12. Under the **ColorPicker** object, choose **ColorHuePicker** and drag **Controller (left)** into the **Tracked Obj** space. Similarly, drag **Controller (left)** into the **Tracked Obj** space for **ColorSaturationBrightnessPicker** in the **Draggable (Script)** section.

4.13. In the **Project** section, click on **Materials**. Then add a new material by right clicking, then **Create**, then **Material** and rename this material to **DrawingMaterial**. Click on the white box, underneath the **Opaque** option and across from the **Albedo** option, and change A = B = G = R = 0. Drag this material into the **L Mat** position of the **DrawLineManager** object in the Hierarchy, which is located in the section labeled **Draw Line Manager (script)**.

4.14. Add a **TextMeshPro Text** by highlighting each fbx file, then **UI**, then **TextMeshPro Text**. If prompted to add **TextMesh Essentials**, then close out this window.

4.14.1. Highlight each **TextMeshPro Text** under all fbx files and change the position to Pos X = Pos Y = Pos Z = 0.5, change the Width to 10, the Height to 1, and the scale to X = Y = Z = 0.1.

4.14.2. Delete any text that is in the Text box, change the font size to 14, and change the vertex

color, which is the white box located next to Vertex Color, to R = G = B = 0 and A = 255.

4.14.3. Add a **Mesh Renderer** by going to **Add Component | Mesh | Mesh Renderer**.

4.15. Go to **VRTK** in the **Project** tab, go into the **Prefabs** folder, then the **ControllerTooltips** folder, and drag the **ControllerTooltips** prefab onto **Controller (left)** and **Controller (right)** in the **Hierarchy**.

4.15.1. For each controller, click on **ControllerTooltips** in the **Hierarchy** and change the **Button Text Settings** to the controller scheme given in **Figure 4**.

REPRESENTATIVE RESULTS:

Using the method described above, the results are created from the example dataset, which was split into 24 time steps and saved in the PLOT3D standard data format. Since 3 quantities were saved, 72 .obj files were created in Matlab and converted into 72 .fbx files in Blender and imported into Unity. The Matlab code can be adjusted according to the number of time steps of the user's data by changing the number of times that Matlab iterates through the main for loop, which can be found at the very beginning of the script. We note here that the results displayed in **Figure 3** show isosurfaces that have been trimmed in the Matlab code to exclude noise at the domain boundaries that were the results of edge effects.

The controller scheme is displayed in **Figure 4**, however the button functions can be changed by modifying the code in the Right_Controller and Left_Controller scripts. The functions programmed in the example game include:

- Manipulate the data object, similar to click and drag with a computer mouse (trigger 1)
- Scale the data object size (triggers 1 and 5 together)
- Change the time step of the displayed data object (button 7)
- Toggle through displayed quantity (ω_z , u , v) (button 8). Note that when toggling through either time or quantity, position, orientation, and scale else will stay the same. Moreover, toggling through time will not reset quantity, and toggling through quantity will not reset time step.
- Draw different color line objects on top of the data objects (buttons 4 and 9). When using this function, the drawings can be erased (button 2) and/or the data objects can be made invisible (button 6)

In **Figure 5**, a simple 3D drawing is made in Unity superimposed on the data objects, and the figure shows the color picker (horizontal spectrum bar) and the saturation picker (left square box). The right box shows what color the ribbon will be once the drawing function is engaged. The width of the ribbons can be changed in the DrawLineManager script.

When the scene is started in Unity, **Figure 3A** shows how it should look when the isosurfaces are properly imported and the controllers have been properly read into Unity. **Figure 6** shows what happens when Unity is not able to properly sense the controllers. This occurs because the cameras used for HTC Vive does not see the controllers or the scene was started before the user

turned on the controllers. Simply restart the scene in Unity to fix this issue.

FIGURE AND TABLE LEGENDS:

Figure 1: Process flow chart

Figure 2: Example images from visualization software Fieldview (A) Q -criterion isosurfaces, shaded from red to blue by spanwise (z) vorticity; (B) x -direction velocity (u) isosurfaces; and (C) y -direction velocity (v) isosurfaces

Figure 3: Analysis of isosurfaces in Unity (A) A representation of the z -vorticity isosurface using the example dataset (B) Representation of the x -velocity isosurface (C) Representation of the y -velocity isosurface.

Figure 4: Controller scheme. The scheme presented is the default setup that is given in the Unity scripts.

Figure 5: Drawing in Unity. An example of how the drawing will appear when in the VR environment.

Figure 6: Error when reading in controllers. Example image of the inaccurate data representations that occurs in Unity when the controllers are out of view of the base stations. Only the blue/green/red Q isosurfaces for a single time step should appear if the import scripts have run successfully

DISCUSSION:

In this article, we described a method for representing fluid dynamics data in VR. This method is designed to start with data in the format of a PLOT3D structured grid, and therefore can be used to represent analytic, computational, or experimental data. The protocol describes the necessary steps to implement the current method, which creates a VR “scene” that allows the user to move and scale the data objects with the use of hand movements, to cycle through time steps and quantities, and to create drawings around and within the data objects in the built. All of these functions can be customized or removed, and therefore not all steps in the protocol need to be followed depending on the user’s needs. Customization of the controls is primarily controlled by the Right_Controller and Left_Controller scripts. The protocol can be customized, and the important process steps could be adjusted depending on the user’s preferences. However, a critical step in the protocol for a new user with their own data will be making sure that it is in the PLOT3D structure so that the Matlab script, as written, can import it properly.

The protocol presented involves the use of three software programs: Matlab, Blender, and Unity. Matlab is a programming language commonly used to analyze large datasets and is a familiar platform for scientists and engineers. Blender is a 3D modeling and rendering software that can be used to manually render 3D objects or, similar to what is done in this article, automatically render 3D objects through the use of python scripts. Finally, Unity is a video game engine that is at the forefront of game design and allows users to customize their VR experience. This protocol

also uses the SteamVR and VRTK plugin for Unity. SteamVR and VRTK help to ease the design of the user interface and smooth the programming gap. For the purposes of the protocol, the SteamVR plugin gives the necessary scripts to program the controllers, while the VRTK plugin gives access to the ControllerTooltips which allows the user to see the controls in-game.

To establish that the protocol was written clearly, the authors asked a collaborator, unaffiliated with the work of developing this method, to test it. The collaborator did not run all the supplied Matlab scripts (the largest time investment, as described in the next paragraph), but did run protocol steps 2-4, finishing with a completed interactive game. This was done in under 2 hours with minimal intervention from the authors. Based on this, it is inferred that the protocol is reasonable for researchers with intermediate experience or expertise in coding or video game development.

There are multiple limitations to the method presented. First, there are three heavy software packages involved, all of which operate with different programming languages, as Matlab has its own scripting language, Blender uses Python, and Unity uses C#. Further, each package has vastly different user interfaces which, along with using different programming languages, makes the current process have a high learning curve to effectively manage. A second limitation is the issue of computational cost, specifically the section of the process that converts the data into .obj files in Matlab. The computer used for this protocol used an Intel Core i7-6700K CPU with 32 GB of RAM. Processing the data into .obj files using Matlab took about 2 hours with the computer specifications. This produced x-velocity and y-velocity isosurface object files that were about 20 MB in size and z-vorticity isosurface object files that were about 10 MB in size. In total, the FullSetOBJ folder was around 1 GB in size. Further, processing the three Blender scripts took approximately 15 minutes and produced the FBX folder that was also about 1 GB in size. Lastly, another limitation is that there is currently no way to compute new objects for a given scene (e.g. change isovalues) on-the-fly, as Unity requires all imported objects to be pre-rendered. Thus, anything additional the user wants to visualize will need to be rendered in Matlab and/or Blender and imported, which may take considerable time.

Various directions can be taken for further research. One is reducing the amount of time needed to render and import the objects into Unity. Currently, substantial time is taken to perform this process, but most of this time is taken during the Matlab section. Future research could convert the Matlab code into another programming language that handles loops and writing subroutines more effectively, such as C or C++. Another direction is the potential of executing computational fluid dynamics simulations within Unity. Blender and Unity are well-equipped to handle the process of animation, thus, CFD simulations could be rendered in both software packages. Blender, however, contains extensive features relating to animation that Unity does not, so users looking for a more thorough animation process may look to Blender.

Another point of future research may be to incorporate these tools and methods in the pursuit of an optimal learning experience through gamification for students. An interesting approach may be to capture user data in order to determine times of high user engagement similar to what is found in Krietemeyer et al.¹⁵ Integrating this approach with the current method could reveal

aspects of the VR experience that most engage students, allowing future researchers to expand on them to create a more immersive experience. Lastly, adopters of this method may want to visualize other researchers' experimental results. Integrating a database of experimental results such as in Garcia-Hernandez and Kranzlmüller¹⁶ may prove useful to future researchers. This will allow users to compare multiple variables of interest such as the outcome of different initial conditions, observations of the wake made by various geometries, or visualization of various quantities in a VR experience.

ACKNOWLEDGMENTS:

This work was supported by the Office of Naval Research under ONR Award No. N00014-17-1-2759. The authors also wish to thank the Syracuse Center of Excellence for Environmental and Energy Systems for providing funds used towards the purchase of lasers and related equipment as well as the HTC Vive hardware. Some images were created using FieldView as provided by Intelligent Light through its University Partners Program. Lastly, we would like to thank Dr. Minghao Rostami of the Math Department at Syracuse University, for her help in testing and refining the written protocol.

DISCLOSURES:

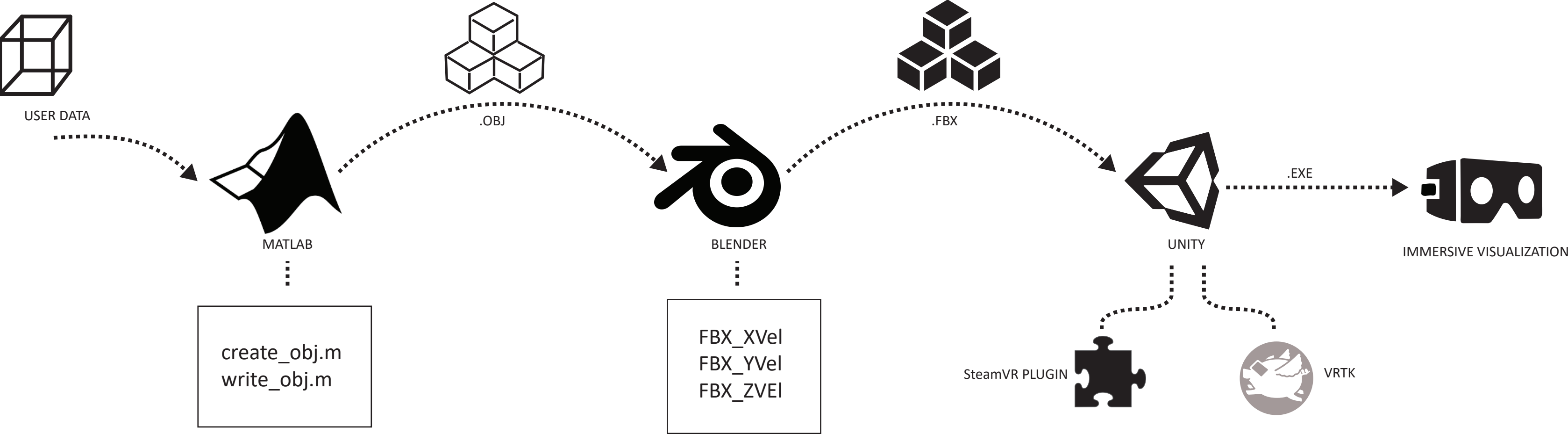
The authors have nothing to disclose.

REFERENCES:

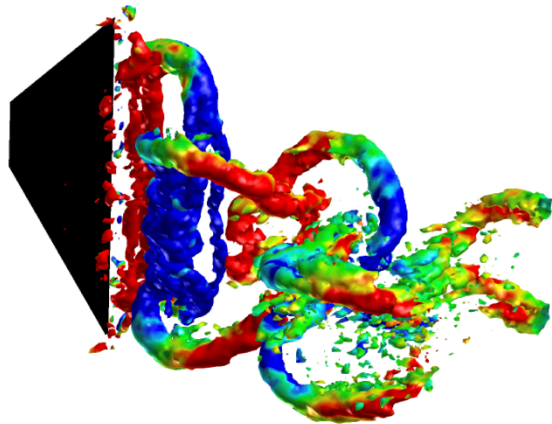
1. Zhang, Z., Zhang, M., Chang, Y., Aziz, E.S., Esche, S.K. and Chassapis, C. Collaborative virtual laboratory environments with hardware in the loop. In *Cyber-Physical Laboratories in Engineering and Science Education*. 363-402 (2018).
2. Reshetnikov, V. et al. Big Data Approach to Fluid Dynamics Visualization Problem. *Lecture Notes in Computer Science Computational Science – ICCS 2019*, 461–467 (2019).
3. Merchant, Z., Goetz, E. T., Cifuentes, L., Keeney-Kennicutt, W, Davis, T. J. Effectiveness of virtual reality-based instruction on students learning outcomes in K-12 and higher education: A meta-analysis. *Computers & Education*. **70**, 29–40 (2014).
4. Chang, Y., Aziz, E.S., Zhang, Z., Zhang, M. and Esche, S.K. Evaluation of a video game adaptation for mechanical engineering educational laboratories. in 2016 IEEE Frontiers in Education Conf (FIE) (2016).
5. Kuester, F., Bruckschen, R., Hamann, B, Joy, K. I. Visualization of particle traces in virtual environments. *Proceedings of the ACM Symposium on Virtual Reality Software and Technology - VRST 01* (2001).
6. West, S. "Innovative Virtual Reality Software Developed to Enhance Fluid Dynamics Lectures: Imperial News: Imperial College London." *Imperial News*, <https://www.imperial.ac.uk/news/189866/innovative-virtual-reality-software-developed-enhance/> (2019).
7. Huang, G, Bryden, K. Introducing Virtual Engineering Technology Into Interactive Design Process with High-fidelity Models. *Proceedings of the Winter Simulation Conference, 2005* (2005).
8. Shahnawaz, V. J., Vance, J. M, Kutti, S. V. Visualization and approximation of post processed computational fluid dynamics data in a virtual environment. *1999 ASME Design Engineering Technical Conferences* (1999).

9. Schulz, M., Reck, F., Bertelheimer, W, Ertl, T. Interactive visualization of fluid dynamics simulations in locally refined cartesian grids. *Proceedings Visualization 99 (Cat. No.99CB37067)* (1999).
10. Wu, B., Chen, G. H., Fu, D., Moreland, J, Zhou, C. Q. Integration of Virtual Reality with Computational Fluid Dynamics for Process Optimization. *AIP Conference Proceedings*. **1207**, 1017 (2010).
11. NPARC Alliance CFD Verification and Validation. "Plot3d File Format for Grid and Solution Files." Retrieved 20 December 2019, from <https://www.grc.nasa.gov/WWW/wind/valid/plot3d.html> (2008).
12. Brooks, S., Green, M. A. Effects of Upstream Body on Pitching Trapezoidal Panel. *AIAA Aviation 2019 Forum*. AIAA 2019-3429 (2019).
13. King, J., Kumar, R, Green, M. A. Experimental observations of the three-dimensional wake structures and dynamics generated by a rigid, bioinspired pitching panel. *Physical Review Fluids*. **3**, 034701 (2018).
14. Unity Asset Store. "Simple color picker." Retrieved 20 December 2019 from <https://assetstore.unity.com/packages/tools/gui/simple-color-picker-7353> (2013).
15. Krietemeyer, B., Bartosh, A, Covington, L. A shared realities workflow for interactive design using virtual reality and three-dimensional depth sensing. *International Journal of Architectural Computing*. **17** (2), 220–235 (2019).
16. García-Hernández, R. J, Kranzlmüller, D. NOMAD VR: Multiplatform virtual reality viewer for chemistry simulations. *Computer Physics Communication*. **237**, 230–237 (2019).

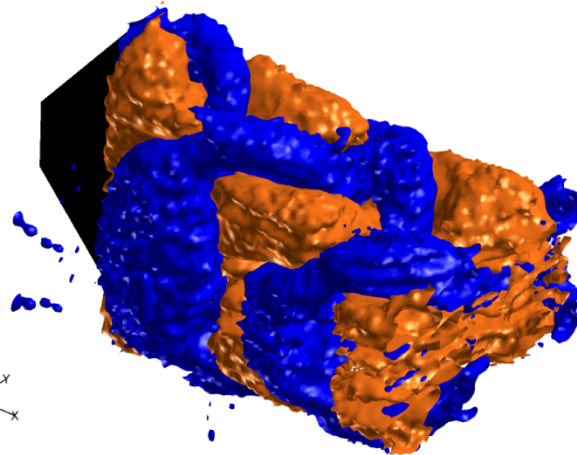
Figure 1



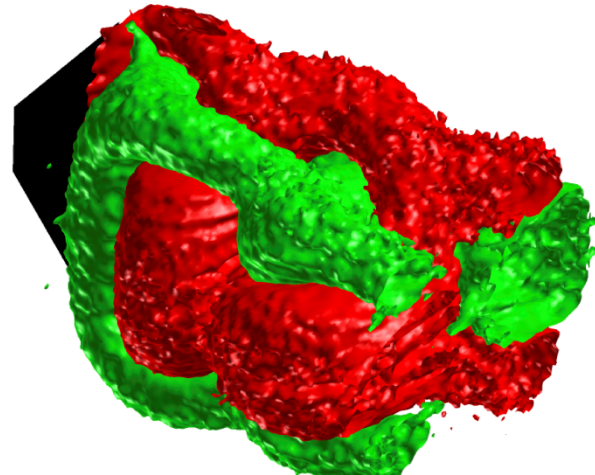
A



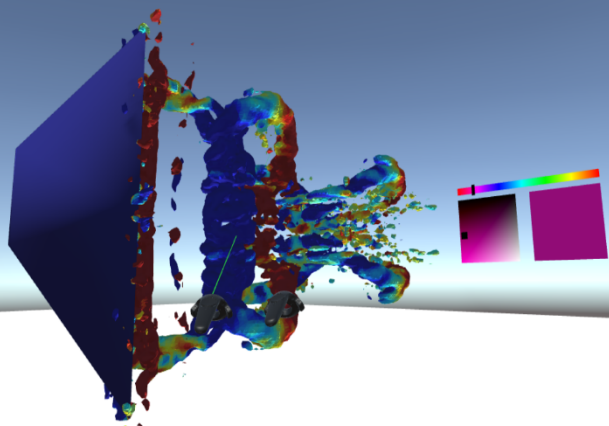
B



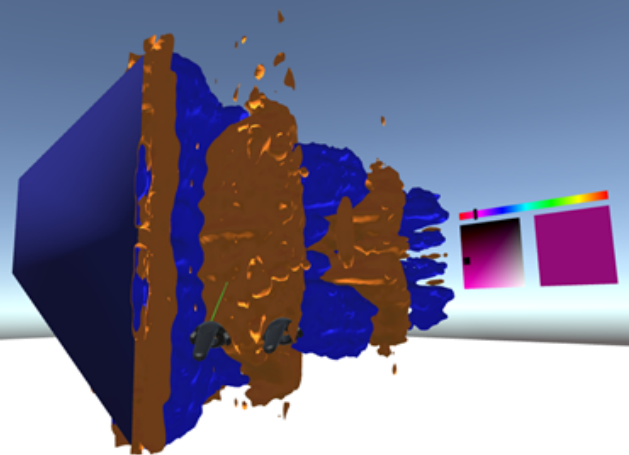
C



A



B



C

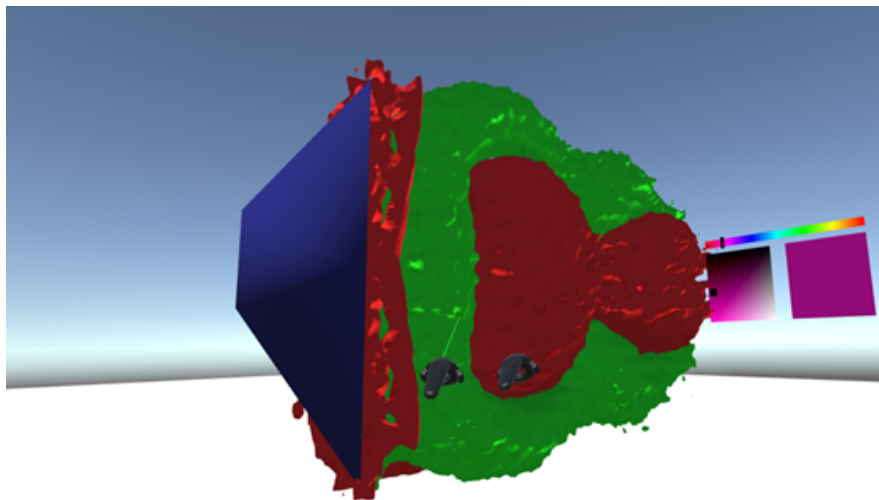


Figure 4



- [Click here to access/download:Figure;Figure_4 \(1\).pdf](#)
- Holding down 1 (the left trigger) while moving the controller will move the isosurface. Holding down 1 and 5 (right trigger) while moving the controllers away or bringing them closer together scale the isosurface.
 - 2 will erase the drawings made and 6 will deactivate/reactivate the isosurface.
 - 3 will deactivate the color picker. 7 will cycle through the time-steps and holding down 7 will cycle through the time-steps continuously. 8 will change the quantity, e.g. change to x-velocity.
 - Use 4 to select the color and saturation on the color picker. Use 9 to draw.

Figure 5

[Click here to access/download;Figure;Figure_5 \(1\).pdf](#)

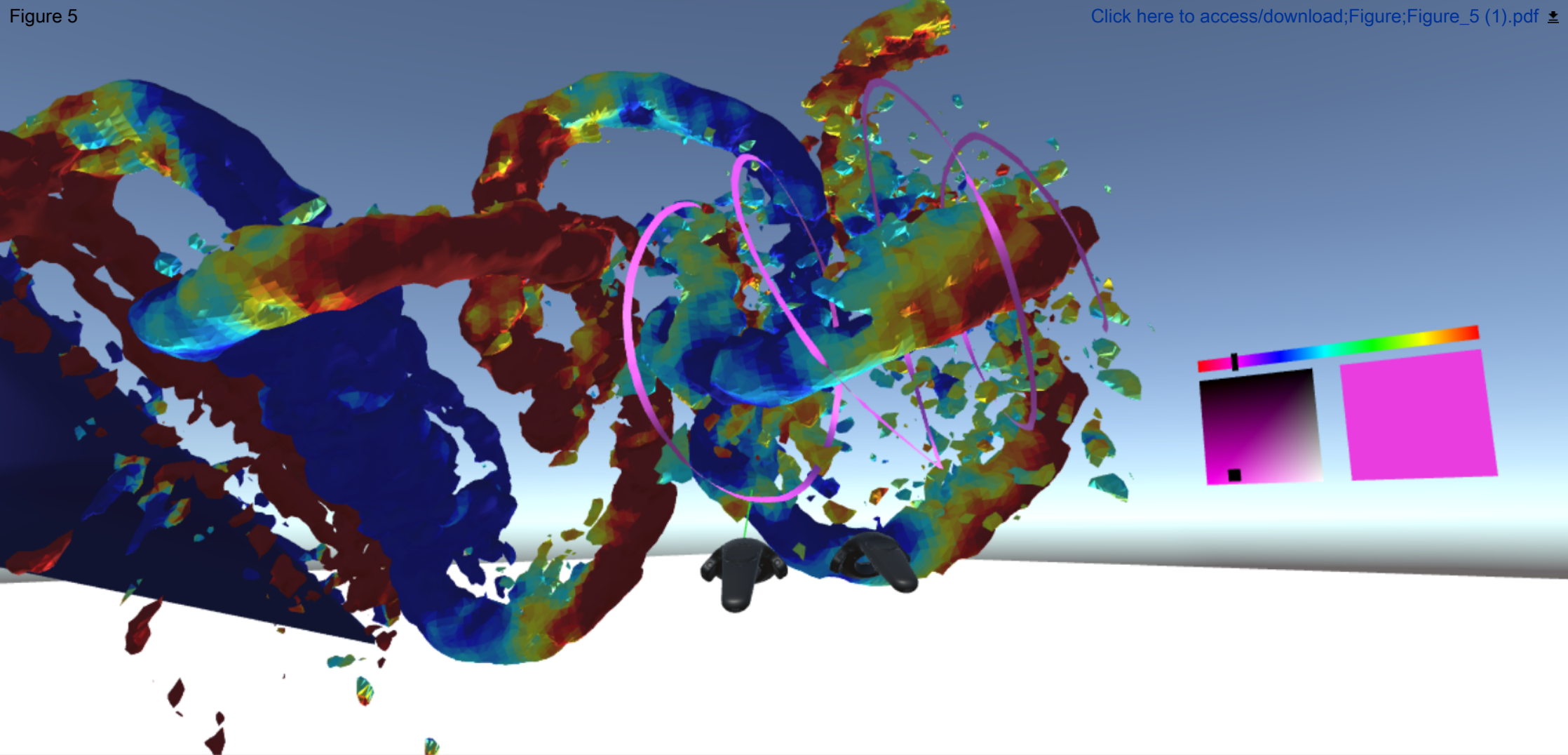
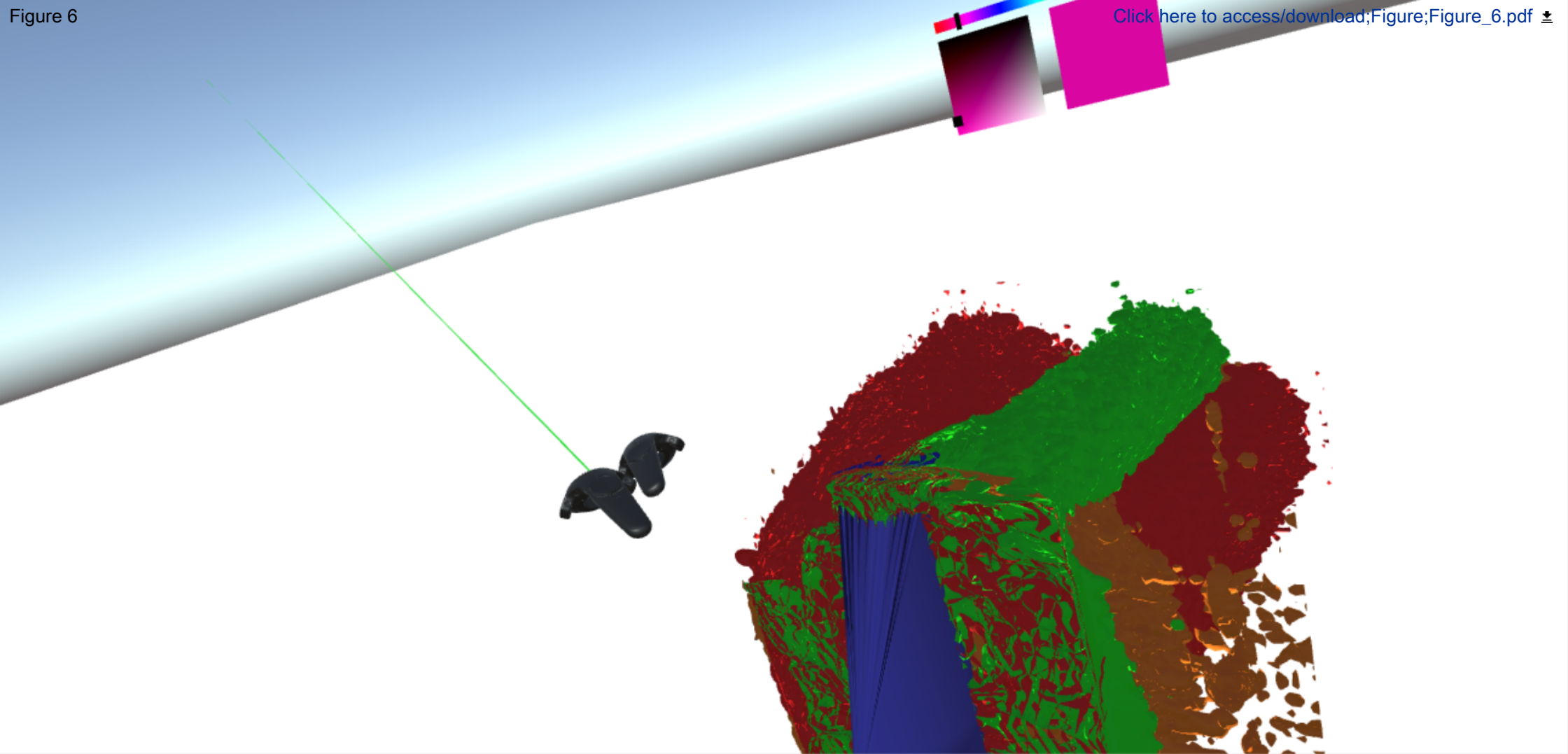


Figure 6





Name of Material/Equipment	Company	Catalog Number
Base station	HTC and Valve Corporation	
Base station	HTC and Valve Corporation	
Link box	HTC and Valve Corporation	
Vive controller	HTC and Valve Corporation	
Vive controller	HTC and Valve Corporation	
Vive headset	HTC and Valve Corporation	

Comments/Description

Used to track the user's position in the play area.

Used to track the user's position in the play area.

Used to connect the headset with the computer being used.

One of the controllers that can be used.

One of the controllers that can be used.

Headset the user wears.

JOVE-S-19-02255

Reviewer Response

We would like to thank the reviewers for their comments. Across the board, we found that we were able to improve the manuscript by addressing them.

Below, we have copied most of the review text. Those comments are in bold face, with our comments below. We have started a new page for each reviewer.

Reviewer #1

Manuscript Summary:

The authors report a protocol for an immersive visualization technique for fluid dynamics using virtual reality that is based on already available commercial software. The combination of using commercial software and this protocol will aid in customization and user interaction for researchers that are getting started in using virtual reality for visualization purposes. The wake of a caudal fin geometry that is modeled as a trapezoidal panel is visualized here. The detailed description of the protocol and the transfer of data from computing environment to rendering surfaces to the final visualization is a valuable contribution.

I strongly recommend this paper for publication, as this protocol can be used for similar immersive visualization in fluid dynamics that can be useful as an educational tool for students.

We appreciate the reviewer's recommendation!

Major Concerns:

Line 42 - It states that VR can reveal interesting patterns and processes that can be difficult to detect in traditional 2D format. Why is the comparison being made against 2D? It is intuitive that VR will reveal more features compared to 2D but what about in comparison to 3D data that can be visualized using Matlab, tecplot. 3D data visualization is more than sufficient to reveal all the patterns, so VR has no edge over it except that the use of VR is crucial for educational purposes. Please revise the statements to reflect that.

In fact, we did mean to compare with visualizations of 3D surfaces (rendered on 2D screens). While there are not types of patterns that are accessible in VR that are not accessible in 3D rendering software like Matlab, Tecplot, Fieldview, or Paraview, the immersive environment (object manipulation + physically moving the viewer's body through the space) does add to the research visualization experience. We are not social psychologists, however, and will not venture to detail what that difference is in this paper, nor do we think that would be appropriate for this journal.

We do think the reviewer is right that we should not be pushing the fact that this visualization is "better" as a *result* of this work, even if we unintentionally did so in the original submission. The result of this work is that interaction with the 3D surface data that fluid dynamicists commonly do using the mentioned software is *possible* in VR as an immersive experience, and to detail the protocol that results in that. We have edited the text to reflect that, as the reviewer suggested.

Minor Concerns:

Line 229 (Step 4.15.4) Repeat this process for each of the .fbx flies. So for the current data set, you'd have to repeat it 72 times, right? Is there a way to automate this process in the future?

The reviewer is absolutely correct. This is something that we have continued to work on since the original submission. That process is more concrete now, and we have added text to the manuscript to incorporate it.

Line 301 - What does the plugin VRTK provide the user with?

The VRTK value is in the eased design of user interface - SteamVR and other apps have since helped smooth the programming gap, but initially the controls and head display were obtuse to access for non-programmers. In the revision of the manuscript, we have actually added some explicit tools from VRTK, and its use is now explicit in the protocol.

Reviewer #2:**Manuscript Summary:**

The manuscript described an interesting topic in interactive and immersive visualization of fluid dynamics using virtual reality. The topic itself is not innovative. Researchers have been applying virtual reality to education for many years, but each researcher may have his/her own methodology to create innovative solutions for a specific topic. The authors generated a protocol using Matlab, Blender, and Unity to generate the protocol, and the protocol was defined in a general form. In addition, the authors conducted sufficient research in the problem background. The protocol was illustrated clearly with details, and the figures demonstrated the feasibility of the protocol.

We thank the reviewer for these comments. We agree - we are not the leaders in VR technology and visualization. We have, however, come up with a protocol that makes visualization of 3D fluid dynamics data something that is relatively inexpensive and accessible to researchers who may not otherwise have pursued it!

Major Concerns:

However, this manuscript is lack of discussion of the protocol testing and evaluation. As a journal paper, a whole package including introduction, protocol description, testing and evaluation, and discussion should all be included and demonstrated to readers. Without user's testing and evaluation, it is difficult to justify whether the product the protocol generated is useful to improve user's perception and understanding of a topic such as fluid dynamics. It is strongly recommended that authors add one separate session of testing and verification.

Our understanding of a JoVE manuscript is taken from their website: "The journal publishes experimental techniques in a visual format with detailed text protocols to increase scientific reproducibility and productivity." In light of JoVE's mission, our manuscript provides a method that results in the visualization of the experimental data in VR. The sample results demonstrate that this is the case. We absolutely understand and agree with the reviewer that there is an entire field of social psychologists who study how people interact and engage with VR environments, whether understanding is increased or not, etc. - we consider that to be outside the scope of this manuscript/journal.

That being said, in the spirit of establishing reproducibility, we did engage a collaborator who did not participate in the development of this protocol, and established that the protocol is readable, relatively easy to follow for someone with intermediate coding and/or video game experience and expertise, and does produce the expected results. This process did illuminate some places in the protocol in which clarification was needed. Text reflecting this has been added to the manuscript.

Reviewer #3:

Major Concerns:

in my opinion the app should be tested among users to get the feedback

Minor Concerns:

this is rather the presentation of the application, not a research paper

From JoVE's website, "The journal publishes experimental techniques in a visual format with detailed text protocols to increase scientific reproducibility and productivity." It is with this motivation in mind that we wrote the protocol and sample results.

That being said, in the spirit of establishing reproducibility, we did engage a collaborator who did not participate in the development of this protocol, and established that the protocol is readable, relatively easy to follow for someone with intermediate coding and/or video game experience and expertise, and does produce the expected results. This process did illuminate some places in the protocol in which clarification was needed. Some text reflecting this has been added to the manuscript.

Reviewer #4:

Manuscript Summary:

This paper talks about the application of the virtual reality in the visualization of fluid dynamics. The structure, method, data and experiments are great, but the expression of English should be improved greatly.

Major Concerns:

The expression of English.

In revising the manuscript for resubmission in response to all the reviewers, we closely checked it for grammar and readability.

Minor Concerns:

(1) Proper references. Line 32, introduction. Virtual reality is not a new concept, so there should be one reference to show where the definition of VR comes from. One referring source may be "Zhang, Z., Zhang, M., Chang, Y., Aziz, E.S., Esche, S.K. and Chassapis, C., 2018. Collaborative virtual laboratory environments with hardware in the loop. In Cyber-Physical Laboratories in Engineering and Science Education (pp. 363-402). Springer, Cham."

We thank the reviewer for the suggested reference. Using it as a starting point, we revisited some relevant literature and added more than one new citation.

(2) Line 107 data set. 'which is commonly'. I think that this method can only be used in the case of vortex surfaces. So, please make sure that the claim is accurate.

We are not sure entirely follow the reviewer on this point. Using isosurfaces of 3D scalar fields is common in fluid dynamics, and not restricted to vortex-dominated flow fields. If it is about specifically using an isosurface of 1% Q_{\max} - we don't disagree that this is really only used to show vortex surfaces, but in the manuscript as submitted the sentence was (emphasis here is ours for this reviewer response), "The isosurface level was chosen to be 1% of the maximum value, which is commonly used in 3D **vortex** visualizations to reveal structures while avoiding noise that can be prevalent at lower levels." For this reason, we think the text was already accurate.

(3) Protocol. Line 125 to line 229. I recommend to use four flowcharts to help the description of the protocol.

We really liked this suggestion from the reviewer and have done just that! Please see figure 1.

Reviewer #5:

Manuscript Summary:

The manuscript presents a novel protocol to post-process and visualize CFD data in Virtual Reality using Matlab, Blender and Unity. The authors have demonstrated the protocol using CFD data from one of their publications of water tunnel experiments. The 3D surfaces used to visualize the structure of the fluid motion are first rendered in Matlab. The data is visualized using isosurfaces of the Q-criterion. It is a scalar calculated at every point in 3D space from the velocity vectors. The resulting surfaces are then coloured using spanwise vorticity (ω_z) to highlight and distinguish the large-scale structures of interest, mainly oriented in the spanwise (z) direction. Quantities such as streamwise (x-direction) velocity (u) and the transverse (y-direction) velocity (v) can be processed similarly. The two provided Matlab scripts i.e. create_obj.m and the write_wobj.m are used for processing and converting the visualized data to 3D OBJ Files for each time slice. These files are converted to 3D FBX file using Blender. The resultant FBX files are rendered using Unity 3D running SteamVR plugin software for visualizing the data set in HTC Vive VR. Representative results are discussed.

We appreciate the summary from the reviewer. We are on the same page, with one small note - the data used as an example is not from CFD. It is experimentally-acquired. In revising the manuscript, though, we found 1-2 places in the text where that lack of clarity may have come from, and this gave us the impetus to check for and fix those!

Major Concerns:

The authors have discussed the computational cost as one of the limitations without discussing any benchmarks. The presented approach requires the pre-rendered data to be loaded in-memory for VR visualization. This can be a bottleneck for rendering large and complex temporal datasets. Benchmarking results can help the users in identifying the set of hardware required for visualizing such CFD data and conducting experiments. However, this can be beyond the scope of JOVE.

The reviewer is absolutely correct. We made sure to include machine details and reported some run times in this revision of the paper, for exactly the reason the reviewer mentioned here.

The paper mainly talks about visualizing CFD data in VR as a protocol. The paragraph between lines 321-331 is little out of context and can be omitted unless experiments related to education and learning experiences are presented as part of the protocol.

We can see what the reviewer is saying here. We do, however, want to push back slightly. The reviewer is correct that we are not presenting an educational module as part of this protocol. The paragraph in question, though, was meant to be part of a discussion of potential future work using this protocol as a baseline. We think it still has value there - to suggest other uses of this kind of data visualization to the reader. We have also, though, revised the paragraph slightly to make sure that it is clear that this is the intention, and not that we think the protocol, as-written, is classroom-ready.

Minor Concerns:

1. To my experience, 3D OBJ files can be visualized directly in Unity for VR. The expensive process of converting OBJ files to FBX files using Blender can be omitted.

We absolutely understand the reviewer's comments here, and in general don't disagree. However, we ourselves had issues with Unity referencing the .mtl files (that handle the object "material," which we use for surface color). If we convert to .fbx, though, that issue is resolved. We acknowledge that ours is **not** the most elegant process, but it **is** possible, and accessible for many who already handle three-dimensional fluid dynamics data. In particular, many of those researchers are already using Matlab to generate isosurface representations, so our protocol is a fairly simple add-on to enable VR visualization.

To make sure this was clear in the manuscript, we added some text to reflect that this step avoids other complications.

2. The authors have used Unity's "personal licence" (free) instead of "educational licence" that may have licence restrictions for the images used in figures.

We thank the reviewer for bringing this up! We clarified explicitly with those who handle our educational license, and are confident that our use of images in this manuscript is covered under those terms.



Click here to access/download
Supplemental Coding Files
exportAsFBX_XVel.blend






Click here to access/download
Supplemental Coding Files
exportAsFBX_YVel.blend



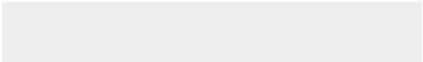



Click here to access/download
Supplemental Coding Files
exportAsFBX_ZVort.blend





Click here to access/download
Supplemental Coding Files
create_obj.m





Click here to access/download
Supplemental Coding Files
write_wobj.m



Click here to access/download
Supplemental Coding Files
ColorManager.cs





Click here to access/download
Supplemental Coding Files
Draggable.cs



Click here to access/download
Supplemental Coding Files
Draggable.cs.meta





Click here to access/download
Supplemental Coding Files
DrawLineManager.cs

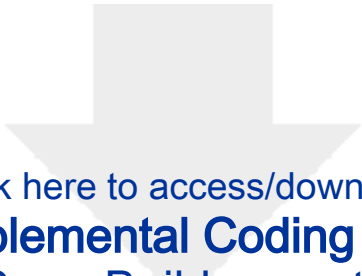


Click here to access/download
Supplemental Coding Files
DrawLineManager.cs.meta





Click here to access/download
Supplemental Coding Files
GameBuild.cs



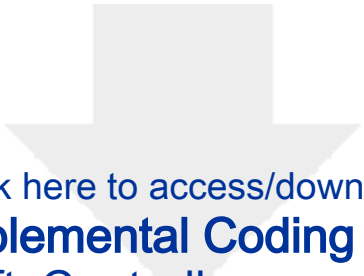
Click here to access/download
Supplemental Coding Files
GameBuild.cs.meta



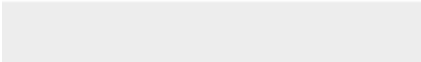



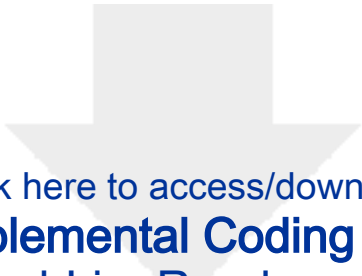
Click here to access/download
Supplemental Coding Files
Left_Controller.cs



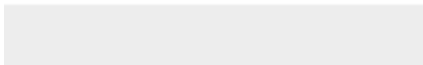



Click here to access/download
Supplemental Coding Files
Left_Controller.cs.meta





Click here to access/download
Supplemental Coding Files
MeshLineRenderer.cs





Click here to access/download
Supplemental Coding Files
MeshLineRenderer.cs.meta



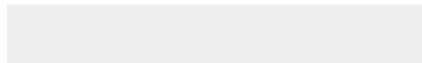



Click here to access/download
Supplemental Coding Files
Right_Controller.cs



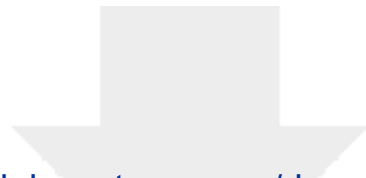


Click here to access/download
Supplemental Coding Files
Right_Controller.cs.meta



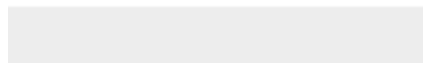
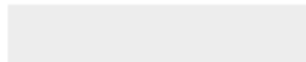


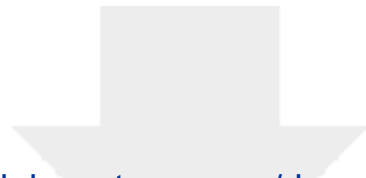
Click here to access/download
Supplemental Coding Files
Set_Text_Position.cs



[Click here to access/download](#)

Supplemental Coding Files
Set_Text_Position.cs.meta

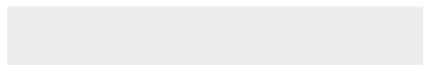




[Click here to access/download](#)

Supplemental Coding Files

Trapezoid_All_Interp_Plot3D_01.dat



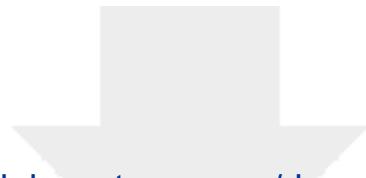


[Click here to access/download](#)

Supplemental Coding Files

Trapezoid_All_Interp_Plot3D_02.dat

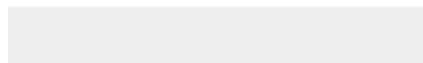
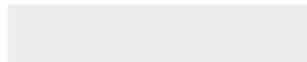


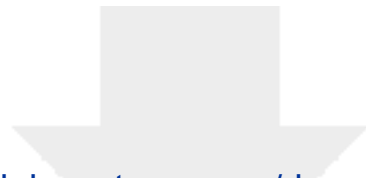


[Click here to access/download](#)

Supplemental Coding Files

Trapezoid_All_Interp_Plot3D_03.dat

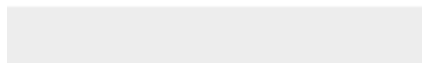




[Click here to access/download](#)

Supplemental Coding Files

Trapezoid_All_Interp_Plot3D_04.dat

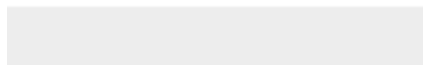


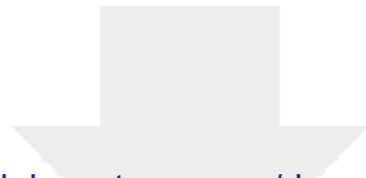


[Click here to access/download](#)

Supplemental Coding Files

Trapezoid_All_Interp_Plot3D_05.dat

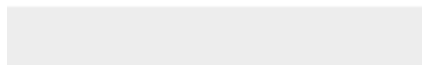


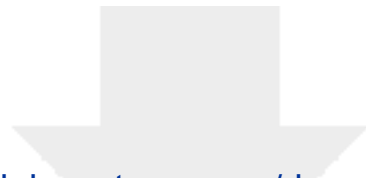


[Click here to access/download](#)

Supplemental Coding Files

Trapezoid_All_Interp_Plot3D_06.dat

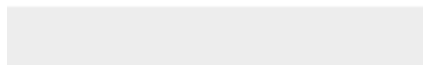


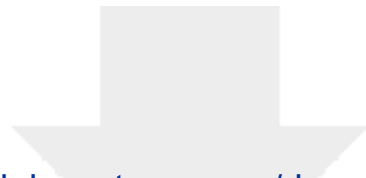


[Click here to access/download](#)

Supplemental Coding Files

Trapezoid_All_Interp_Plot3D_07.dat

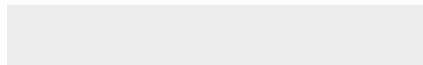




[Click here to access/download](#)

Supplemental Coding Files

Trapezoid_All_Interp_Plot3D_08.dat



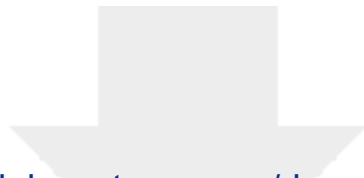


[Click here to access/download](#)

Supplemental Coding Files

Trapezoid_All_Interp_Plot3D_09.dat

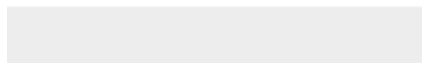
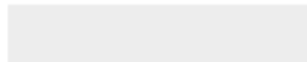


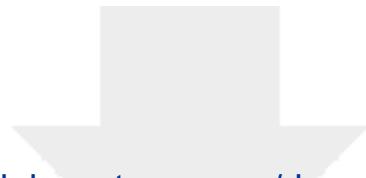


[Click here to access/download](#)

Supplemental Coding Files

Trapezoid_All_Interp_Plot3D_10.dat



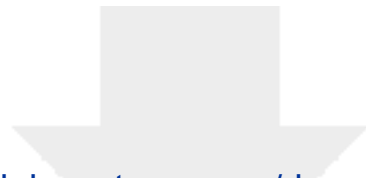


[Click here to access/download](#)

Supplemental Coding Files

Trapezoid_All_Interp_Plot3D_11.dat

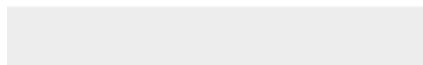




[Click here to access/download](#)

Supplemental Coding Files

Trapezoid_All_Interp_Plot3D_12.dat

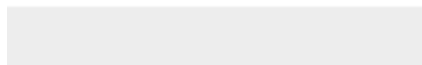


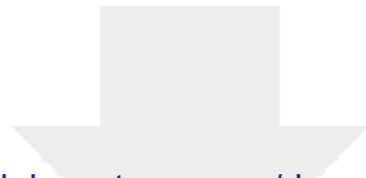


[Click here to access/download](#)

Supplemental Coding Files

Trapezoid_All_Interp_Plot3D_13.dat

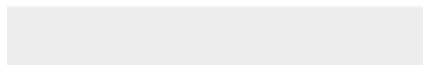


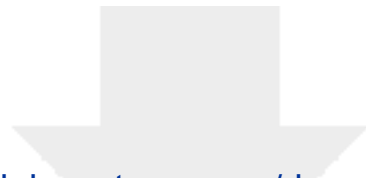


[Click here to access/download](#)

Supplemental Coding Files

Trapezoid_All_Interp_Plot3D_14.dat

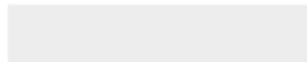




[Click here to access/download](#)

Supplemental Coding Files

Trapezoid_All_Interp_Plot3D_15.dat



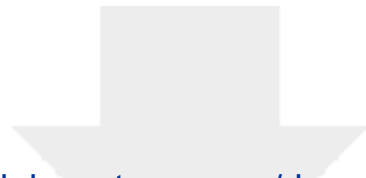


[Click here to access/download](#)

Supplemental Coding Files

Trapezoid_All_Interp_Plot3D_16.dat

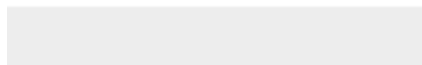




[Click here to access/download](#)

Supplemental Coding Files

Trapezoid_All_Interp_Plot3D_17.dat





[Click here to access/download](#)

Supplemental Coding Files

Trapezoid_All_Interp_Plot3D_18.dat





[Click here to access/download](#)

Supplemental Coding Files

Trapezoid_All_Interp_Plot3D_19.dat





[Click here to access/download](#)

Supplemental Coding Files

Trapezoid_All_Interp_Plot3D_20.dat

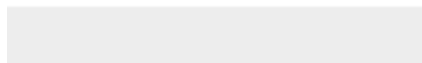
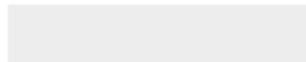




[Click here to access/download](#)

Supplemental Coding Files

Trapezoid_All_Interp_Plot3D_21.dat



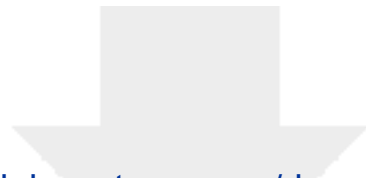


[Click here to access/download](#)

Supplemental Coding Files

Trapezoid_All_Interp_Plot3D_22.dat

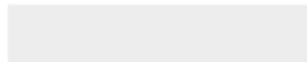


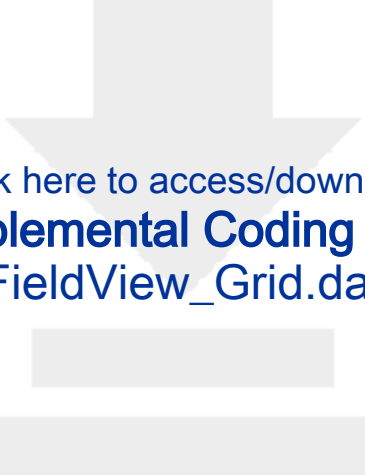


[Click here to access/download](#)

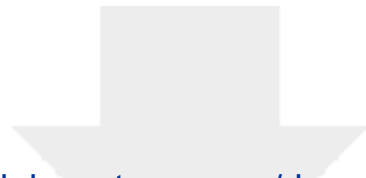
Supplemental Coding Files

Trapezoid_All_Interp_Plot3D_23.dat





Click here to access/download
Supplemental Coding Files
FieldView_Grid.dat



[Click here to access/download](#)

Supplemental Coding Files

Trapezoid_All_Interp_Plot3D_24.dat

