Journal of Flow Visualization & Image Processing, 31(1):1–22 (2024)

VISUALIZATION OF TURBULENT EVENTS VIA VIRTUAL/AUGMENTED REALITY

David Paeres,[1] Christian Lagares,[1] Alan B. Craig,[2] & Guillermo Araya[3],*

[1]*Department of Mechanical Engineering, University of Puerto Rico, Mayaguez, PR 00681, USA*

[2]*University of Illinois, Urbana-Champaign, IL, USA*

[3]*Department of Mechanical Engineering, University of Texas at San Antonio, San Antonio, TX 78249, USA*

*Address all correspondence to: Guillermo Araya, Department of Mechanical Engineering, University of Texas at San Antonio, San Antonio, TX 78249, USA, E-mail: araya@mailaps.org

Original Manuscript Submitted: 1/21/2023; Final Draft Received: 4/10/2023

Mixed reality technology, i.e., virtual (VR) and augmented (AR) reality, has spread from research laboratories to enter the homes of many. Further, the widespread adoption of these technologies has caught the scientific community's attention, which is constantly researching potential applications. Backed by the continued enhancement of high-performance computing in hardware and software, we are applying mixed reality technologies as a scientific visualization tool for fluid dynamics purposes. In particular, we show a virtual wind tunnel (along with the simplified methodology to replicate it) that enables the user to visualize complex and intricate turbulent flow patterns within an immersive environment. Briefly, high spatial/temporal resolution numerical data over supersonic turbulent boundary layers subject to concave and convex wall curvature has been creatively "pipelined" for VR/AR visualization via several scripts, software, and apps, which are further explained and described along the manuscript. The intention is to present a technique of how to visualize fluid flows to be the most convenient for the user, especially if one is slightly unfamiliar with scientific visualization. Whereas VR/AR applications are principally discussed here for flow visualization, the lessons learned can be certainly extended to other disciplines involving three-dimensional time-dependent databases.

KEY WORDS: *VR, AR, flow visualization, CFD, DNS, Unity, ParaView, GLTF, USDZ*

1. INTRODUCTION

Data visualization is the general and all-encompassing term for everything related to the graphic portrayal of information and data. *Information visualization* denotes data in an organized visual way to provide meaning (Friendly and Denis, 2006). According to Friendly and Denis (2006), thematic cartography, statistical graphics, and data visualization histories are intertwined; their article lists 278 milestones of the concept of information visualization, starting with the supposed oldest known map, dated 6200 BC. Apart from the visuals created with purely artistic or religious motivations, it is recognizable that the first creations were intended to aid navigation

and exploration. These used geometric diagrams and tables to track the positions of stars and sky constellation elements. The birth of theories such as measurement errors, probability, and analytical geometry provoked the scientific community's mindset toward statistical theory and the systematic collection of empirical data. Inevitably, information visualization extended into new graphic forms, such as isolines, contours, and other techniques for thematic mapping of physical quantities such as geological, economic, and medical data (Friendly and Denis, 2006). Because of the technological development in the 20th century, data visualization received significant attention, causing its permanent presence in the scientific field due to its effective way of conveying information. Friendly and Denis (2006) define *scientific visualization* as the area "primarily concerned with the visualization of 3D+ phenomena (architectural, meteorological, medical, biological, etc.), where the emphasis is on realistic renderings of volumes, surfaces, illumination sources, and so forth, perhaps with a dynamic (time) component." In other words, in scientific visualization, the visual representations cannot be designed solely from the artistic inspiration of the creator but, by definition, must be based on scientific data identified as genuine and authentic in order to have scientific and academic applications (e.g., records, research, analyses, etc.). For Hansen and Johnson (2011), the visualization field focuses on creating images that tell significant information about underlying data and processes. In their work, they developed a handbook presenting the basic concepts, providing a snapshot of current visualization software systems, and examining research topics in the field. The objective of scientific visualization is also to enhance the interpretation of complex and large data sets. Since it is highly used in computing nowadays, it is considered a subset of computer graphics, making it part of computer science. More suggested literature includes *A Concise Introduction to Scientific Visualization: Past, Present, and Future* by Hollister and Pang (2022), while more specific articles in the computational aspect are Nielson et al. (1997) and McCormick (1987). Regarding flow visualization, a half-century ago, the only technique available to describe coherent structures in turbulent flows was smoke and dye injection (Brown and Roshko, 1974; Winant and Browand, 1974). According to Hesselink (1988), flow visualization results from the interaction between matter and light. The definition was based on the classical methods available in those years, such as shadow photography and Schlieren photography. In the early 1980s, other techniques were developed through numerical processing of flow visualization pictures for experimentally estimating some of the principal physical variables of fluid flows. That classical visualization relied on variations of the index of refraction or spatial derivatives of pressure, density, and temperature integrated along the light path through the fluid (Hesselink, 1988). With such technology, it was already possible to study phenomena like unsteady twin-vortex flow and Kármán vortices, as presented in Imaichi and Ohmi (1983). The *Visualization Handbook* [by Hansen and Johnson (2011)] describes *flow visualization* as a relevant topic of scientific visualization, intending to provide more understanding of fluid flows, where typically, nowadays, the data is originated from numerical simulations such as those of computational fluid dynamics (CFD).

Visualization techniques have substantially evolved along with the world's technology, spanning all disciplines (Sherman et al., 1997). One of the most used approaches in the experimental discipline of flow visualization is the incorporation of particle image velocimetry (PIV). As its name suggests, PIV is based on capturing a series of images with a known frame rate to measure the flow's instantaneous velocity. Much progress has been made in the research of fluid dynamics using PIV. For example, in recent studies, Mohd et al. (2022) has analyzed transient trailing jets (TJ) behind a compressible vortex ring generated by a shock of Mach 1.5. They were able to identify the dominant flow features of the present scenario by categorizing the development of TJ into three stages. Furthermore, Jha et al. (2022) attempted to visualize and measure a laminar

natural convection boundary layer formed over a heated vertical flat plate. The experimental results were compared with the analytical solutions, and they claimed broad agreement. Today visualization leans greatly on computers and numerical approaches because, in 20 years, the computing capacity has increased over three orders of magnitude (Paeres et al., 2021b).

The virtual wind tunnel (VWT) (Paeres et al., 2021b) is a virtual environment created with the Unity game-engine platform to enhance user's data visualization by immersively observing and interacting with any three-dimensional (3D) virtual object (i.e., static or animated), currently from direct numerical simulation (DNS) results, but expandable to other computer-aid designs (CAD) and other scientific fields. We unified two software rarely related to each other (i.e., Unity and ParaView), extending the applications offered by these software to create an immersive (i.e., virtual reality and augmented reality) scientific visualization methodology, more specifically for CFD simulations. The methodology is simple to replicate and uses open-source software, while the personal licenses are free. In the present article, we are expanding the capabilities of our VWT to work with GLTF files (see Section 3.4), a widely used standard file format for 3D scenes and models, especially for mixed reality, while maintaining the use of the HTC Vive VR kit and Microsoft HoloLens 1st gen. hardware, which are devices that many mixed reality (MR) enthusiasts have already acquired. Although not extensively tested, some preliminary findings suggest that using the HTC Vive Pro 2 to replicate the present pipeline yields comparable outcomes.

In MR, virtual objects can be invoked in the physical world with the ability of awareness and interactions, as was done in Paeres et al. (2021a, 2020). However, these further steps to obtain the virtual objects' interaction with the real world are out of the scope of this paper. In the Section 2, readers will be introduced to the concept, benefits, and applications of extended reality (XR).

2. VIRTUAL REALITY, AUGMENTED REALITY, AND MR

The concept of a VWT is not new, nor was it created in the 21st century. References to this term can be dated back to Bryson and Levit (1991). In that article, Bryson and Levit defined "*virtual environment*" as the approach to user interfaces in computer software, integrating various input and display devices, giving the illusion of being immersed in a computer-generated scene. The VWT idea was refined and presented again in 1993 (Bryson, 1993); at this time, the term "virtual reality" was also used. These early works were pioneering but held back by the limited technology of the time. Nonetheless, in the late 1980s and early 1990s, a debate between the "virtual reality" and "virtual environment" labels was known to be the "VR definition wars" (Bryson, 2013). In 1998, Bryson recognized that "VR" supporters surpassed the defenders of the term "VE" by sheer volume. Moreover, he proposed a modernized definition of "virtual reality" backed by his detailed discussion and the comparisons between the dictionary definitions of "virtual" and "reality," in which the definition stood the test of time (Bryson, 2013): "Virtual Reality is the use of computer technology to create the effect of an interactive three-dimensional world in which the objects have a sense of spatial presence."

Clarifying the difference between virtual reality (VR) and augmented reality (AR) requires an *a priori* explanation of the concept "virtuality continuum," introduced 29 years ago by Milgram and Kishino (1994). They declared the continuum with four regions and argued that the taxonomy should be based on three subconcepts, as follows: (i) "Extent of World Knowledge: The amount of information bleeding from the real environment into a virtual environment;" (ii) "Reproduction Fidelity: An attempt to quantify the image quality of the displays;" and (iii) "Extent of Presence Metaphor: A term to encapsulate the degree of immersion." In other words,

Milgram and Kishino (1994) described a continuum between the real world, which they call the "real environment," and the "virtual environment," which is a fully synthetic world in which computer generated signals are provided to the senses of a person in full replacement of the natural signals their senses normally perceive. This is what we are referring to as "virtual reality." Between those two end points are various levels of mixing, the natural signals and the synthetic signals. In short, the difference between virtual reality and augmented reality is how the real world is treated. In virtual reality, the real (natural) world is hidden and the participant experiences only computer generated stimuli to their senses, but in augmented reality, the participant fully experiences the real world with the addition of computer generated signals to augment their real-world experience. There are different ways to achieve this mixing; the primary distinction is whether the real world is mediated through some type of video or if the real world is perceived optically. Years later, Flavián et al. (2019) proposed a new taxonomy in five divisions, as shown in Fig. 1. Following the present interpretation, this continuum represents the classifications of the different modes in which a user can perceive sensory experiences and have their interactions.

From left to right in Fig. 1, the *real environment* is the physical world, where technology is not needed for its existence and interactions. Second is AR, where virtuality overlays reality yet does not interact with the whole physical world. AR usually requires a lens device so the user can visualize the virtual objects overlying the physical world. In the midway, it is the *pure mixed reality*, where virtuality and reality are fused, explained as the generation of virtual objects with complete awareness of the real environment. The fourth is *augmented virtuality* (AV), where reality overlaps with virtuality: for example, a computer-generated display crowded with digital images but controlled by physical commands. The last taxonomy is the *virtual environment*, where the complete environment is digital, and virtual reality is the medium used to access this mode (Bryson, 2013). Any input from the real world has to be translated into a digital expression. The junction from AR to AV is called MR. At the same time, extended reality (XR) covers from AR to virtual environment, also commonly used as an unspecified reality mix.

Regarding XR's benefits and applications, many studies conclude that XR enhances education and research (Boyles, 2017; Lee et al., 2010; Saidin et al., 2015). For example, Radianti et al. (2020) reviewed how to optimize VR applications for educational purposes. It was concluded that students tend to retain better information after VR exercises, and their performance was improved after applying what they learned in a more immersive experience. Consequently, one could hope for more development until reaching a fully immersive feeling in the XR technology. Since the beginning of the 21st century, limitations have been identified. Rosenblum (2000) outlined key areas where this technology's development should focus, from ultra-lightweight haptic feedback sensors to realistic interfaces that make it hard to distinguish between realities. Besides improving sensors for better virtual environment perspectives, the hardware and software need to be fast and smooth enough without having data memory size constraints (Albert et al., 2017; Elbamby et al., 2018).

Virtual reality applications fully immerse the user in a 3D virtual world (Craig et al., 2009). VR and MR technologies attempt to blur the line between the physical world and the digital

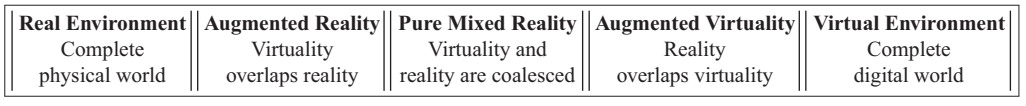

FIG. 1: Virtuality continuum, reproduced from Flavián et al. (2019)

environment, blossoming industries such as architecture, engineering, and construction (Delgado et al., 2020; Purushottam et al., 2021; Ververidis et al., 2022). Furthermore, it provides options in the research environment for difficult or costly experiments.

Video game engine platforms and VR technology merge well for scientific procedures. For example, Brookes et al. (2020) researched the development of an experimental platform for behavioral scientists with VR and the game engine software Unity 3D. Brookes et al. (2020) designed and proposed the Unity Experiment Framework (UXF) app providing the "nuts and bolts that work behind the scenes of an experiment developed within Unity ... that allow logical encoding of the common experimental features required by the behavioral scientist." UXF gave an optional graphical user interface (GUI), allowing the experimenter to design examinations easily, define dependent and independent variables, and edit the display interface, without requiring high-level knowledge of coding languages. A helpful feature of UXF was also the cloud-based platform; in experiments performed remotely from a laboratory, the software collected the data and stored it in servers. To confirm the UXF reliability, the authors performed a study between adults and children to examine their postural sway in response to an oscillating virtual room. The validation experiment resulted in the children presenting less postural stability than adults, as was expected from previous research done by others (Brookes et al., 2020), also endorsing the integration of XR and game engines into scientific methodologies.

Regarding flow visualization, such XR technologies benefit visual representation. It enables a researcher to perform scientific analysis and to gain a better insight into complex flows in complicated geometries that are hard to perceive with the naked eye at a fraction of the effort and cost required in a physical setup (Paeres et al., 2021b). For example, in the early 2000s, Aoki and Yamamoto (2000) proposed the design system for a three-dimensional blade involving VR in turbo-machinery. They showed how virtual reality could help a designer to visualize the three-dimensional flow fields and modify the blade configuration to further calculate the results in an iterative approach. In a different project, Eissele et al. (2008) presented a system able to track position, orientation, and other additional aspects from a scenario used to influence the visualization. They observed automatically seeded streamlines rendered with halos to enhance the contrast to the underlying real-world image captured by a camera. Their work was focused on flow visualization in AR. However, they claimed that their approach could also be applied to other visualization tasks. In a more recent project, Walcutt et al. (2019) used virtual reality as a scalable and cost-efficient visualization mechanism to interpret large volumes of ocean data. They summarized ways where VR could be applied to the oceanography field and demonstrated the utility of VR as a three-dimensional visualization tool for ocean scientists.

3. RESULTS AND DISCUSSION

In this section, we outline the required computational "pipeline" to perform virtual and augmented reality visualizations of turbulent flows via high spatial/temporal resolution numerical predictions. The major goal is to specifically describe the tasks, software, and conversion tools used in our proposed method for XR scientific visualization. In all cases, DNS databases generated by our research group are employed. This section is organized as follows: Section 3.1 provides the motivation for turbulent flows and its boundary layers simulations while summarizing the approach used in our DNS simulations, Section 3.2 gives details of the data post-processing approach regarding flow solutions via our Aquila library, Section 3.3 discusses some software and methods available to create and convert 3D models, Section 3.4 examines the data type selections for XR 3D models, Section 3.5 is focused on the process of creating the GTLF files with

ParaView, Section 3.6 addresses the process to recreate the VWT, and finally Section 3.7 refers to the process of creating the AR models with USDZ files.

3.1 Turbulent Flow Simulations

Boundary layer problems are challenging since the majority of transport phenomena (mass, momentum, and heat) occurs inside this thin shear layer. The problem becomes harder if one wants to model and simulate the effects of turbulence and compressibility, particularly at relevant Reynolds numbers. In addition, three-dimensionality and unsteadiness are "the rule not the exception," making the problem more arduous and resource-consuming. The DNS is a numerical tool that resolves all turbulence length and time scales; thus, it provides the highest spatial/temporal resolution possible of the flow within a computational domain. Furthermore, turbulent boundary layers that evolve along the flow direction are ubiquitous and show a nonhomogeneous condition along the streamwise direction due to turbulent entrainment (henceforth, spatially developing turbulent boundary layers, SDTBL).

From a computational point of view, these types of boundary layers (i.e., spatially developing boundary layers) involve enormous numerical challenges due to the need for precise and time-dependent inflow turbulence information. Furthermore, accounting for strong-wall curvature effects adds significant complexity to the problem since the boundary layer is subject to critical distortion due to the combined streamwise-streamline (adverse and favorable) pressure gradients. We are making use of the dynamic multi-scale approach (Araya et al., 2011), a method for prescribing realistic turbulent velocity inflow boundary conditions, later extended to high-speed turbulent boundary layers in Araya and Lagares (2022). It is based on the rescaling-recycling method proposed by Lund et al. (1998). Briefly, the methodology of our modified rescaling-recycling method is to prescribe time-dependent turbulent information at the "inlet" plane based on the transformed flow solution downstream from a plane called "recycle" by using scaling laws, as seen in Fig. 2. Furthermore, in our proposed new approach, there is no need to use an empirical correlation [as in Lund et al. (1998)] in order to compute the inlet friction velocity; such information is computed "on the fly" by involving an additional plane, the so-called test plane located between the inlet and recycle stations. Here, the friction velocity is defined as $u_\tau = \sqrt{\tau_w/\rho}$, where τ_w is the wall shear stress and ρ is the fluid density. Turbulence precursors are canonical flat-plate turbulent boundary layers, which are numerically resolved via DNS and

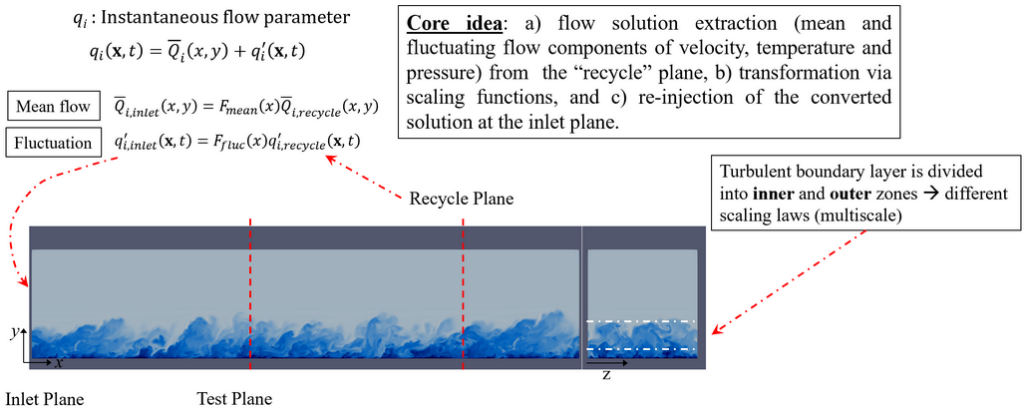

FIG. 2: The turbulence inflow generation methodology

the previously mentioned inflow generation technique. As part of the post-processing stage and once a statistically steady flow is obtained, a cross-sectional plane from downstream of the recycle plane is extracted and saved in a database in terms of the flow solution, i.e., the three velocity components, pressure, and temperature. Those instantaneous fields are used as time-dependent inlet boundary conditions on more complex domains.

We are employing a DNS database of supersonic concave-convex geometries for virtual/augmented reality purposes in the present study. The boundary conditions were selected as follows. At the wall, the classical no-slip condition is imposed for velocities. The lateral boundary conditions are handled via periodicity. For supersonic cases, the flow velocity (three components), temperature, and pressure are prescribed at the inlet based on precursor simulations at similar conditions. Dimensions of the complex computational domain (L_x, L_y, and L_z) are $27\delta_{inlet}$, $17\delta_{inlet}$, and $3\delta_{inlet}$ along the streamwise (x), wall-normal (y), and spanwise (z) directions, respectively. The number of cells in the finite-element mesh is approximately 10 million. The typical mesh resolution is $\Delta x^+ = 8$, $\Delta y^+_{\text{min}} = 0.2$, $\Delta y^+_{\text{max}} = 10$, and $\Delta z^+ = 8$, where superscript $+$ indicates wall units. The Courant-Friedrichs-Levy parameter is approximately 0.2 during the simulation, and the time step is fixed at $\Delta t^+ \approx 0.09$. The reader is referred to Araya et al. (2022) for more details and DNS validation. Figure 3 exhibits a flow visualization of DNS over concave-convex geometries at a freestream Mach number of 2.86, low Reynolds numbers ($\delta^+ = 239-676$), and wall Q-adiabatic conditions. In Fig. 3(a), iso-surfaces of instantaneous spanwise velocity fluctuations and iso-contours of instantaneous density can be observed. There is continuous flow turning and compression in the concave wall, which merge outside the turbulent boundary layer in a shockwave, as seen in Fig. 3(b). Turbulent conditions were generated via an asynchronous flat-plate precursor [iso-contours of instantaneous density at the inlet domain can be seen, according to a $y-z$ plane view in Fig. 3(c)]. The close-up of the concave wall in Fig. 3(d) shows a clear enhancement of spanwise velocity fluctuations at the turbulent concave wall, which is caused by large-scale roll cells. These roll cells are generated by unsteady "Taylor-Görtler-like" vortices via similar centrifugal instability mechanisms as in laminar concave flows, as stated by Barlow and Johnston (1988). No spanwise inhomogeneity has been observed in time-averaged flow statistics. The very strong favorable pressure gradient (FPG) induced by the convex surface stabilizes and quasi-laminarizes the flow, which remains

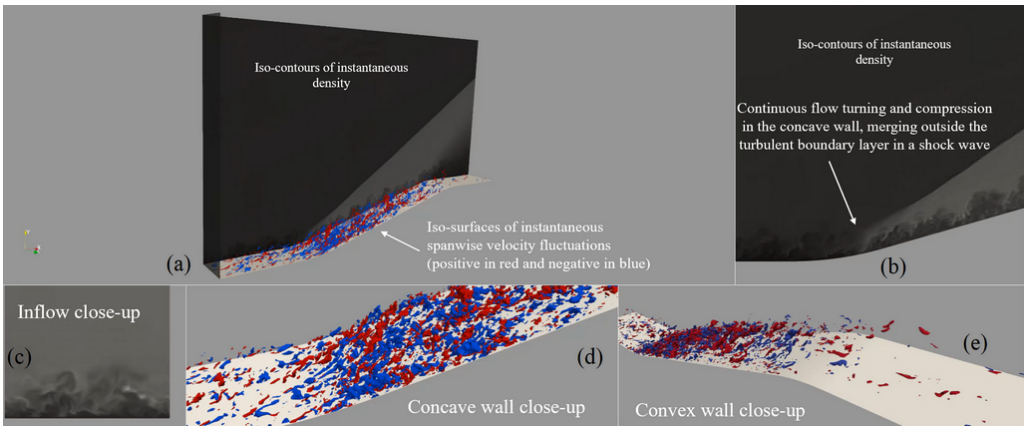

FIG. 3: Flow visualization of DNS at $\delta^+ = 239-676$ (low Reynolds numbers). Wall Q-adiabatic conditions with $T_w/T_r = 1.04$.

two-dimensional [see Fig. 3(e)]. Additionally, two extra wall thermal conditions were evaluated: wall cooling ($T_w/T_r = 0.4$) and wall heating ($T_w/T_r = 1.5$). Here, T_w is the wall temperature, and T_r is the recovery temperature.

3.2 DNS Post-Processing

Processing the large datasets generated via DNS requires careful attention to details, ensuring a scalable and efficient solution. For this work, we leveraged previous work in the Aquila library by Lagares et al. (2022, 2021b). Aquila is a modular library for post-processing large-scale simulations. It enables large datasets by operating out-of-core and providing the illusion of an in-memory dataset through asynchronous data pre-fetching. We currently target an MPI backend (Message Passing Forum, 1994) for distributed memory communication and HDF5 (The HDF Group, 2010) for our storage backend. However, storage backends can be swapped with ease by implementing reading/writing functions. To provide context for the size of a DNS simulation, the cases being presented in this work total 2.22 TBs[†]. Aquila serves as a computational platform and is currently not intended as a visualization engine. To enable visualization of the post-processed results, the library provides a modular interface into the VTK library (Schroeder et al., 2006), which enables visualization in VisIt (VisIt, 2022) and ParaView (Henderson, 2007). This enables a logical separation of concerns and also ensures a decoupled visualization and computation platform. What's more, Aquila supports writing to directly high-performance binary formats through HDF5, facilitating data archival and sharing across tools such as TecPlot (Tecplot, 2022).

To attain a fluid rate of animation, computationally intensive post-processing quantities are performed offline, leveraging all of the available computational resources. We have discussed scalable, time-averaged CFD post-processing in Lagares et al. (2022, 2021b). However, instantaneous parameters require special attention. This is especially true for visualizations such as Q-criterion. Visualizing the evolution of the Q-criterion (shown in Fig. 4) in time requires calculating the following expression,

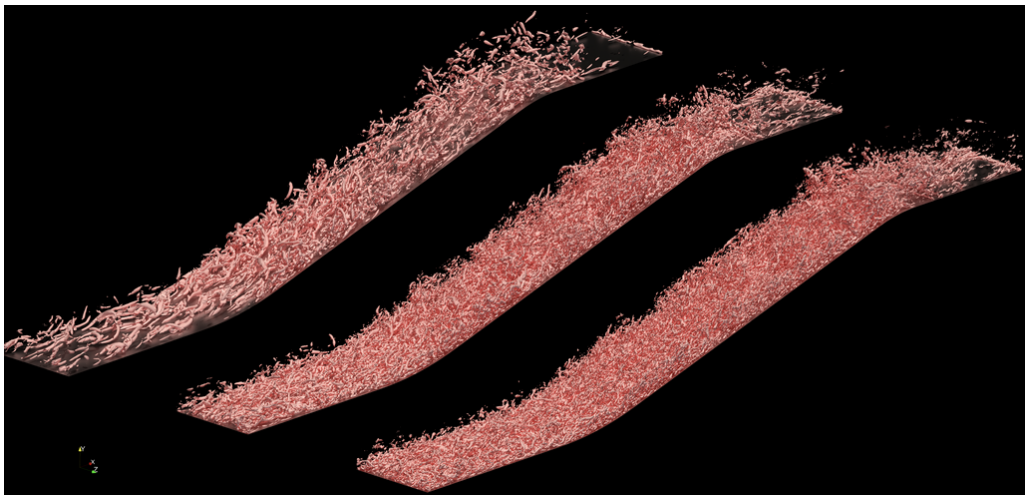

FIG. 4: Q-criterion for cold, adiabatic, and hot walls (left to right)

[†]1 TB = 1,099,511,627,776 bytes

$$Q = \frac{1}{2}\left(||\Omega||_2^2 - ||S||_2^2\right) \tag{1}$$

where $\Omega = (1/2)(\partial u_i/\partial x_j - \partial u_j/\partial x_i)$ (the antisymmetric strain tensor) and $S = (1/2)(\partial u_i/\partial x_j + \partial u_j/\partial x_i)$ (the symmetric strain tensor). Note that regions where $Q > 0$ indicate parcels of fluid being dominated by vorticity or flow rotation, whereas negative Q values highlight strain- (or viscous stress) dominated regions. The major drawback for the Q-criterion is its lack of objectivity (due to the necessary and arbitrary threshold required for visualization); however, that criticism does not hinder its utility in visualizing vortical structures. The benefits are directly appreciable in Figs. 5 and 6. Figure 5 highlights the enhancing effect of the adverse pressure gradient (APG) induced by the concave wall for the incoming supersonic boundary layer. Furthermore, the wall temperature effect is also clearly depicted, noting how the flow becomes much more isotropic as the wall is heated (or anisotropic when cooled). On the other hand, Fig. 6

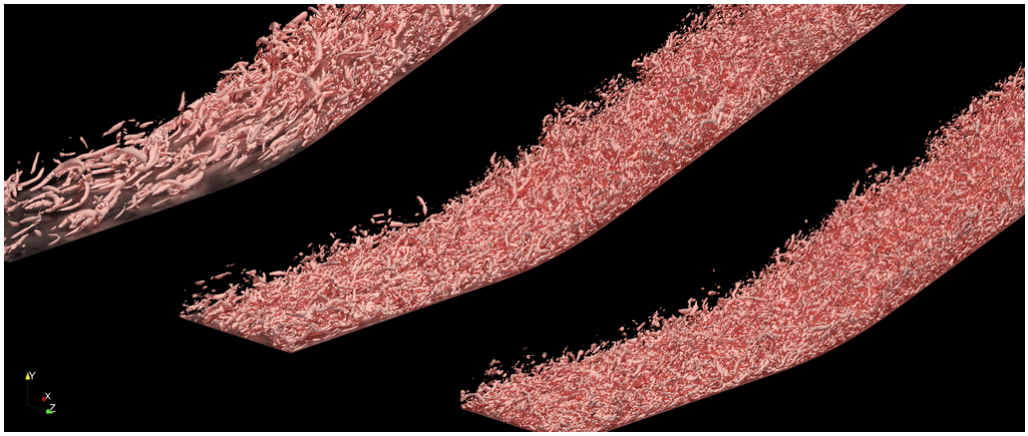

FIG. 5: Q-criterion for cold, adiabatic, and hot walls (left to right) focused on the concave wall, experiencing a strong APG enhancing the turbulent nature of the flow

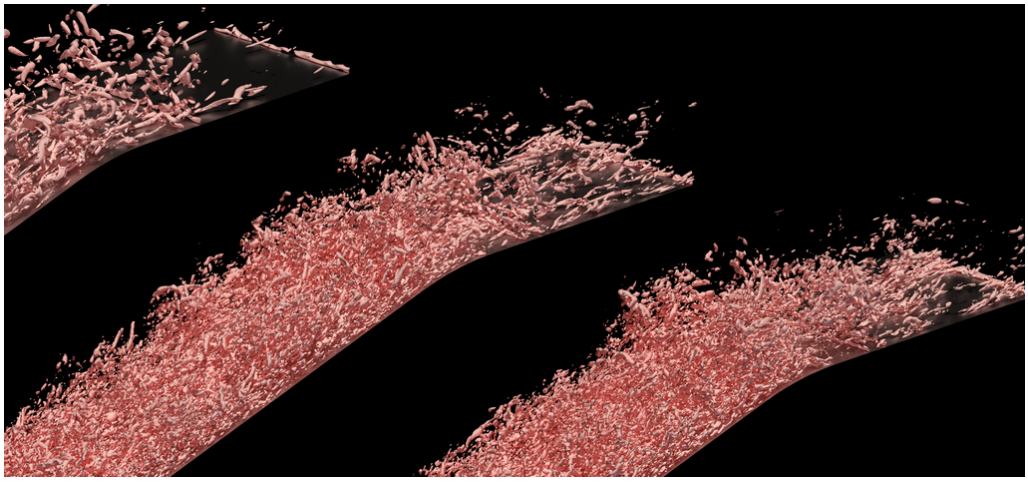

FIG. 6: Q-criterion for cold, adiabatic, and hot walls (left to right) focused on the supersonic expansion experiencing a strong FPG, inducing a quasi-laminar state

showcases the effect of the strong FPG induced by the convex wall where a supersonic expansion occurs. The acceleration is strong enough to force a laminarescent state. The effect is stronger in the cooling wall, which could be directly related to the stronger anisotropic character of the boundary layer along the streamwise direction for the cooling wall.

Our approach to high-performance calculations requiring instantaneous flow parameters is an asynchronous dispatch engine that handles scheduling without input from the domain expert. Currently, we target threading building blocks (TBB) as our threading layer (shared across kernel executions and higher-level kernel orchestration) (Malakhov et al., 2018; Pheatt, 2008). TBB incorporates a noncentralized, highly scalable work-stealing scheduler amenable to unbalanced workloads. Our implementation also approaches issues such as conditional masking of flow variables via Boolean algebra, which avoids excessive memory allocations in favor of additional integer calculations. Optimizing compilers are capable of generating machine instruction, accounting for the wide backends in modern CPUs and concurrent floating point/integer units in modern GPUs and taking advantage of the added integer computations. All in all, our post-processing infrastructure is scalable from platforms with constrained performance to large-scale supercomputers with thousands of CPU/GPU cores.

Although we currently conduct computationally expensive post-processing offline, there is still work being done to enable near-real-time post-processing by tapping into large-scale resources and streaming results as needed. This is analogous to work done so far by Atzori et al. (2022) and Rasquin et al. (2011). Also, we currently employ lossless and physically accurate techniques, whereas graphics workloads often tap visually lossless techniques and include numerical tricks that preserve the visual characteristics of a visualization, while enabling real-time computing in-tandem with graphics. This is the ever-standing issue of low latency vs. high throughput. Currently, we tackle high-throughput post-processing, whereas a near-real-time processing requires low-latency processing of individual flow fields. Enabling such techniques in juncture with AR/VR and high-resolution DNS is beyond the scope of our work, but the knowledge gap is very much worth highlighting.

3.3 CFD Data Post-Processing for XR Visualization

This subsection introduces software and methods to convert data and sampling to eventually create 3D models. While Section 3.2 is more general, covering all aspects on data processing of the flow solution (statistical analysis of turbulence), it is also responsible for generating .vtk or .vts files readable in ParaView. In CFD, post-processing is a set of custom steps for a specific investigation performed after the simulations are completed. Generally, the results of CFD simulation contain basic fields like velocity, pressure, and temperature in their solutions, and to properly execute an assessment, the data needs to be post-processed. Trying to mention all the possible steps for a fluid flow visualization, post-processing would be absurd; it is more convenient to show examples directly involved in assessing the presented work. The most straightforward visualization is presenting the whole simulation's domain with one primary field: for example, the temperature, the velocity's magnitude, or only one of its components. A slightly more advanced step would be to extract an iso-surface or the entire colored domain using a custom colormap or contour to the range of values of the displayed field. An iso-surface is a set of surfaces with a constant value that has been specified, while an iso-contour would be a set of iso-surfaces having several ranges of values, where these ranges can be differentiated by applying different colors to each iso-surface. From the authors' experience, a common and efficient approach to extracting an iso-surface from 3D volumetric data is the marching cubes algorithm (Lorensen and Cline,

1987), since it can be applied on the same scripting platforms (e.g., MatLab and Python), where one can read and manipulate the file data that contain the CFD solutions. 3D volumetric data refers to a three-dimensional array structured in the form (x, y, z), where each element in this array contains the value of the field of interest, and the array indices of the element are understood as the reference to its spatial position in the CFD domain. It is worth mentioning that before our methodology was developed, an attempt was made in Python to implement the marching cubes Lewiner version (Lewiner et al., 2003) to export iso-surfaces in Wavefront OBJ (.obj) format. The option to explicitly implement the Marching Cubes Lewiner algorithm was discarded because its most straightforward implementation was to extract iso-surface to rectangular 3D volumes. For nonrectangular domains, more calibration and complex techniques were required for the extraction of iso-surface for each specific CFD case. Because the goal is scientific visualization, the virtual objects must have true shapes and constructions and not fictional designs. Thus, the real appearance validation part occurs in CFD simulations, where the results are from governing equations and models.

In the search for ways to visualize CFD results without many complications and for the methodology to be capable of being automated, the option of the ParaView visualization toolkit arose. ParaView is a free and open-source multiple-platform software for scientific visualization developed by Kitware Inc., Sandia, and Los Alamos National Laboratories (Henderson, 2007). An alternative open-source tool is VisIt, originally developed by the Department of Energy (DOE) Advanced Simulation and Computing Initiative (ASCI) to visualize and analyze the results of terascale simulations (VisIt, 2022). Moreover, for 3D computational graphics (i.e., 3D-printed models, animated films, visual effects, motion graphics, interactive 3D applications, virtual reality, and more), there is Blender, another highly used open-source tool. However, Blender moves away a bit from scientific visualization and leans more toward object modeling, color and texture rendering, and animation, which is a whole other profession apart from what a CFD analyst can be. It is more convenient to use ParaView since it has been expanding as a preference for use in academia over the years, and the tools focus more on scientific visualization than 3D animation. We use an essential ParaView tool to automate the methodology presented here and facilitate scripting in Python. From version 4.2 of ParaView, there is an option called Trace tool that records every modification made in the ParaView user interface (UI) and generates a Python script corresponding to all the actions in chronological order to facilitate and teach the programming of commands in ParaView.

Regarding flow visualization and frames per second, it is important to differentiate the following terms: *playback* vs. *recording*. In slow motion, the playback frames per second (fps) is much lower than the recording fps. Slow motion permits the user to better identify turbulent events and coherent structures, particularly in turbulent boundary layer problems. The *recording* frequency is directly proportional to the sampling frequency in the numerical database collection. The physical timestep in a typical DNS case is in the order of 1×10^{-4} sec. (or 10,000 Hz), which is prescribed much lower than the limiting Kolmogorov time scale. In this study, we have collected enough volumetric frames at a sampling frequency (i.e., the *recording* frequency), which was ten times smaller than the physical DNS frequency (i.e., 1000 Hz). In other words, volumetric frames were saved every 10 timesteps. Unity game engine software permits the user to adjust the *playback* frequency according to the visualization needs and the upper limitation. In our case, this is given by a maximum of 1000 fps; however, the very small prescribed DNS timestep would allow us to significantly increase it (up to 10,000 fps) if the sampling frequency is increased in the collected database as well. Other important fps limitations are the device's display maximum fps and the mind's ability to interpret the display.

3.4 Virtual/Augmented Reality in Flow Visualization

The procedure of flow visualization with extended reality starts with the information post-processing of CFD results (generally ASCII or binary format). The data is transformed via programming (i.e., Python or MatLab scripts) or visualization toolkit (i.e., ParaView) into a file type of 3D models. During this data transformation, it is expected that the data might require manipulation and filtering such as iso-contour for the desired parameter field. 3D models have wide options of file formats, to name a few: Alembic (.abc), Autodesk FBX (.fbx), Graphics Library Transmission Format (.gltf), Wavefront OBJ (.obj), and CAD (computer-aided design) software extensions. The correct process depends on compatibility with the software and devices used for the visualization. Unfortunately, the options for immersive visualization like VR and AR are limited. The present study uses Graphics Library Transmission Format (GLTF) extension for VR and USD (Universal Scene Description) for AR with iOS (iPhone Operating System) devices. The USD type of file (i.e., .usdz) was developed by Apple and Pixar and launched in 2018. While USDZ files were originally designed for AR applications, it is also possible to employ them under VR environments. In terms of GLTF vs. USDZ, our lessons learned have dictated the following "pros and cons": (i) USDZ files have the perk of being a binary file format, meaning a reduction in the file's storage size ($\sim$ 50%), and could be obtained by GLTF conversion; (ii) GLTF format has its binary counterpart (i.e., GLB), which will be explored as future work in the present study; (iii) USDZ files are naivety compatible with iOS devices, opening up the possibility for AR applications on those devices; (iv) GLTF files can be generated directly from ParaView, making the process much simpler; and (v) Unity game engine can read both USDZ and GLTF files with add-on packages, but the compatibility process with GLTF files are far simplified compared to USDZ files. Furthermore, the virtual environment for the present work was developed with the Unity game engine.

Regarding the VWT, a workstation [see Fig. 7(a)] gathers the information of all input devices. Fully immersed visualizations are performed on the HTC Vive VR kit from High Tech

(a)  (b) 

FIG. 7: (a) Workstation Dell tower 5810 and (b) HTC Vive VR toolkit

Computer corporation [see Fig. 7(b)]. Controllers, head-mounted display (HMD), and base stations sensors identify the user's spatial movement and field of view (FOV), as seen in Fig. 8, and render the virtual objects that belong inside the FOV. Images in the display must be rendered at a speed of at least 30 frames per second (fps) to achieve an immersive impression of the virtual environment. Once the frames are rendered, they are sent to the HMD, which outputs a stereoscopic image providing the user with the 3D illusion of depth. Furthermore, some researchers have performed parametric analyses on VR/AR experiences on users (Louis et al., 2019; Watson et al., 1997). According to Louis et al. (2019), the most influential aspects to establish pleasant VR/AR practices were (i) avoiding a perceivable motion jitter and (ii) a loss in frames per second. Watson et al. (1997) investigated the effects of frame time variation on VR tasks over 10 participants (a mixture of undergraduate and graduate students). At that time, it was concluded that average frame times of 50 ms (or 20 fps) were deemed acceptable for many VR applications. Obviously, the VR technology has substantially evolved in the last decades in terms of GPU capabilities, particularly for the "gaming" field. More recently, investigations have demonstrated that in practice, any VR setup generating frame rates below 90 frames per second (fps) may cause discomfort, upset stomach, disorientation, nausea, and other negative user effects (Antycip, 2022; Zhang, 2020). Therefore, the lower the frame rate, the worse the effects. In addition, a rate of 120 fps per eye seems adequate for delivering outstanding VR training (Linde, 2022).

Obtaining AR flow visualization starts with the same post-processing and data transformations as for VR. However, it differentiates depending on the final goal and interactive degree with the virtual objects. Suppose the idea is only to invoke virtual objects in the real world via iOS devices. In that case, the USDZ file extension is the best option because Apple devices have built-in apps capable of reading the USDZ files, and Unity game engine is not required anymore in our methodology. After converting the GLTF files to USDZ (see Section 3.7), the last step is to share the files with the iOS device to use. However, if the device does not have iOS (e.g., Android and HoloLens), a compatible app will be needed. Here, the Unity game engine comes to the rescue. With Unity, one can have a single development effort, yet the application (containing the virtual objects' files) can be deployed to many different devices. The benefits of designing the apps with Unity are extending the applicability to smartphones and XR devices such as the Microsoft HoloLens (see Fig. 9) and giving opportunities to develop more interactive manipulations such as image recognition and much more.

FIG. 8: User visualizing a flow in the VWT (iso-surface of instantaneous streamwise fluid velocity at 75% of the incoming freestream velocity, flow from right to left)

FIG. 9: Image of Microsoft HoloLens 1st gen., used for AR applications

3.5 Creating the GLTF Files

ParaView has a wide variety of file formats it can read. During each version update, they tend to include more compatible formats. The example presented here is based on ParaView 5.10.1. In the CFD field, simulation results can usually be saved in .vts, .vtk, or other extensions of the Visualization Toolkit data-type family. Operations and manipulations of the results often are carried out before opening the data in ParaView. However, several post-processing calculations can also be accomplished within ParaView. After developing the 3D visualization approach in ParaView, the next step consists of exporting it. There is an option to export files in GLTF format; it must be mentioned that this feature has been available since the ParaView 5.7 version. One should select *Export Scene*, located in the *File* tab, and specify the file type **.GLTF Files (*.gltf)*. As shown in Fig. 10, check all boxes in *Export Options* and *Save* it; this will export the whole scenario that is displayed and rendered in ParaView as GLTF format.

It is also possible and highly convenient to perform the exporting procedure automatically for multiple files (i.e., time step files) employing the *Trace* tool. The intention is to obtain a Python script containing all the actions' commands to automate the process, including reading the multiple files and applying manipulations. This example uses eleven files, each representing a different time step; these files correspond to the work of Lagares et al. (2021a), performing high-fidelity numerical results of supersonic spatially developing turbulent boundary layers (SDTBL) subject to strong concave and convex curvature at Mach = 2.86. The DNS time step was 1×10^{-4} seconds, resulting in a DNS frequency of 10,000 Hz or fps. However, because the sampling was every ten frames, the *recording* fps was 1000 Hz. It is only needed to export two files from the whole file group in the *Trace* stage and then to edit the Python script for looping the export command for all the desired files. For a detailed example guide of the GLTF files export process with ParaView, the reader is referred to Paeres Castaño (2022). The 11 VTS example files used in this subsection can be found in our web page (CTVLab, 2023).

3.6 Creation of the VWT

The VWT is a virtual environment created in the Unity platform to enhance CFD results' visualization using immersive VR and interactions with any virtual object (i.e., static or animated). Currently, the VWT runs in Unity version 2021.3.11f1 and is configured to be used with the HTC VIVE and a Windows-OS-based computer, as seen in Paeres et al. (2020, 2021b).

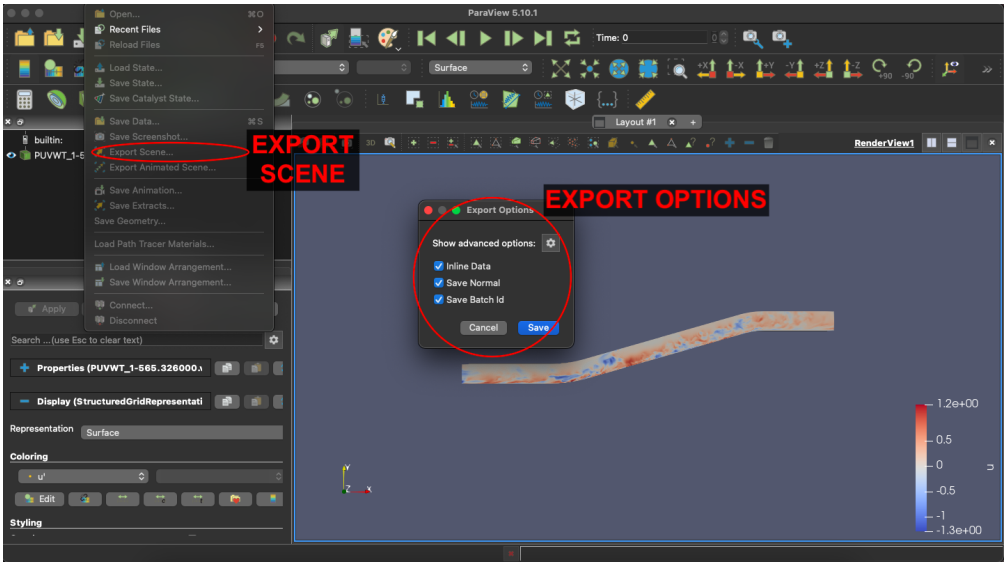

FIG. 10: How to export a scene as GLTF in ParaView 5.10.1. Note the "Export Scene..." option from the File tab; also "Inline Data," "Save Normal," and "Save Batch id" options checked when exporting.

Unity installation recommends Visual Studio along with its Unity add-on. In this study, version 2022 of Visual Studio is utilized. SteamVR is a beneficial tool for connectivity between VR devices (e.g., HTC VIVE) and Unity. It can be installed from the Steam platform, where the Steam software can be obtained from the official web page: http://store.steampowered.com/about/. Furthermore, SteamVR has a plugin registered as a commercial asset in Unity. Sign in with the Unity account on the following web page: http://assetstore.unity.com, and purchase (it is a free asset) the SteamVR Plugin asset. To obtain the VWT by replication, the steps are creating a Unity project with a 3D (core) template and further including more add-on packages, which will be mentioned in this subsection. Given that the present methodology will cause a high computational load with the file transfers and rendering, the project should be saved locally on the computer for optimal hard drive and graphic board performance. Make sure Unity editor is using Visual Studio as external script editor.

The virtual environment (where the user will be immersed) has to be previously created to implement VR properly. The virtual environment could be a simple room with walls and a floor. Many tutorials on the world wide web are on creating virtual buildings, facilities, etc. However, we will import our virtual warehouse created in Unity for this methodology. It will not be shown how the warehouse was built since it is out of the work's scope. Download the virtual warehouse asset from our Dropbox link: https://www.dropbox.com/scl/fo/h32mtibe5ow07vj6s3xzp/h?dl=0\&rlkey=griuoopk7c68pasi3whq70j8s, and locate it with the name *VWT_for_HTCvive.unitypackage*. The warehouse is imported as a custom Unity package. It includes the correct settings for other Unity packages needed, such as SteamVR and High Definition Render Pipeline. After importing the packages with all files and folders, reload the modified scene and accept all SteamVR settings recommendations. From the Scenes directory, open the *VWT* scene, and the virtual environment probably will be all colored in pink. To solve this, import the High Definition Render Pipeline package located in Unity's registry, which achieves a more realistic environment. From the Window tab, open the Package Manager window and change the

list's filter to *Unity Registry* and import the package named *High Definition RP* (version 12.1.7). HDRP Wizard will prompt, click *Fix All* configurations, create *HDRenderPipelineAsset* (with automatic assignation), and wait for the compiling compute variants, then restart Unity Editor. It is crucial to open the *VWT* scene first before fixing the configurations with the HDRP Wizard. If the steps were executed correctly, Unity should look similar as shown in Fig. 11.

To this point, the virtual environment is fully set up, including the teleport ability within the warehouse. The goal is to transform the virtual warehouse into a VWT laboratory. The next step is to enable the import automated process for the high number of GLTF files and develop a script to animate the flow visualization using a stop-motion approach. To use GLTF files in Unity, import another custom Unity asset named *GLTFUtiliy*, which can be obtained from Siccity's GitHub: https://github.com/Siccity/GLTFUtility (also included in our DropBox). After dragging and dropping *GLTFUtiliy-master* directory into the Unity project, right-click on the *Assets* panel to create a directory with the name *Resources*. Use the *Resources* directory to place the material created using the correct color Base map image and all GLTF imported files in one directory (e.g., *GLTFs*). Import the two scripts available in our Dropbox: *FlowAnimator* and *AnimationMeshesMap*. In the *Hierarchy* panel, create an Empty gameObject and add the last two scripts as components. Enter the *GLTFs* directory path and the material applied to the flow. The last step, update gameObject's Transform values (position, rotation, and scale). If the instructions have been followed properly, the VWT should look similar, as shown in Fig. 12. For a video demonstration of the VWT, the reader is referred to Paeres et al. (2022).

3.7 Augmented Reality with USDZ Files

The steps for AR flow visualization depend on the final goal and the interactive degree of virtual objects in the real world. Suppose the idea is only to invoke virtual objects in the real world via

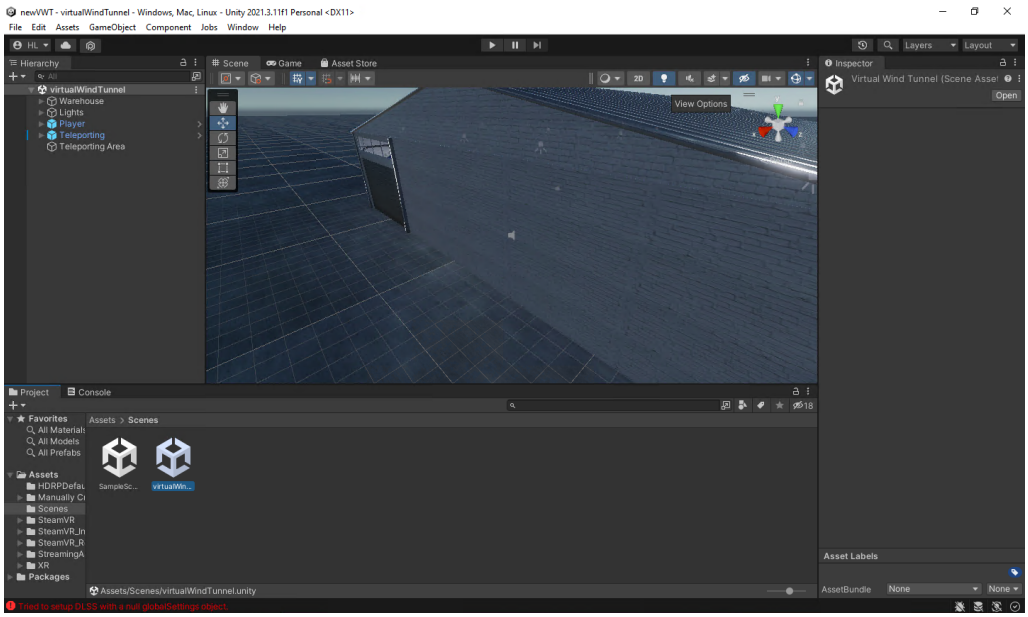

FIG. 11: How the virtual warehouse should look after properly rendering the materials

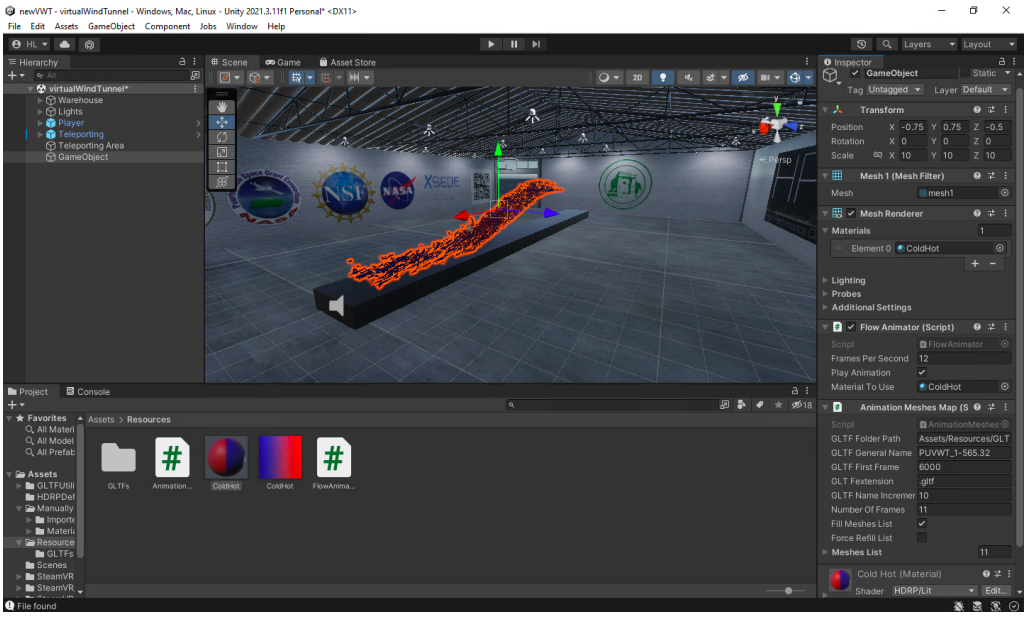

FIG. 12: Correct setup of GLTF files in Unity. Note that inside the *Resources* directory are the *GLTFs* files directory, the material file using an image file as color Base map, and two scripts.

iOS devices; in that case, the USDZ file format is optimal because Apple devices have built-in apps capable of reading the USDZ files. On the other hand, if the device to use is not iOS (e.g., Android and HoloLens), a compatible app will be needed to deploy the USDZ files. Regardless of the final usage of the AR object, in this methodology, it is necessary to transform the GLTF files to USDZ format. On Apple's webpage, there are several tools available to convert files from GLTF to USDZ: https://developer.apple.com/augmented-reality/tools/. This site mentions four software as downloading options; however, this work will be only interested in *Reality Converter* and *USDZ Tools*. Reality Converter is the most intuitive option to convert, edit, and view customized USDZ. This software allows file importing in OBJ, GLTF, and FBX formats by just dragging and dropping to convert them to USDZ. It must be mentioned that currently, ParaView does not export GLTF files preserving the colors used in the visualization. Nonetheless, it does conserve the gradient or distribution of the coloring applied. By default, the *Base color* in Reality Converter is usually a yellowish gradient. Nevertheless, the desired color map can be added with a simple drag and drop of the correct image (for example, see Fig. 13).

Reality Converter is very intuitive due to its GUI (graphical user interface), but for converting many files it is not the most viable option. For multiple files, it is better to use USDZ Tools. USDZ Tools or USD Python Tools is a Python-based pre-compilation with more tools than Reality Converter for modifying USDZ files. It means converting multiple GLTF files to USDZ can be automated with a single Python script. The software installation is simple but has a minor issue since the pre-compilation was done with Python 3.7.9, requiring this specific version of Python to be installed for compatibility in the libraries. To launch the tool, double-click the file called *USD.command*; this file opens a new terminal window for command lines with the proper environment. Checking if the USDZ Tools' terminal window is running with Python 3.7.9 is recommended. Usually, bash or zsh profiles automatically load other versions of Python,

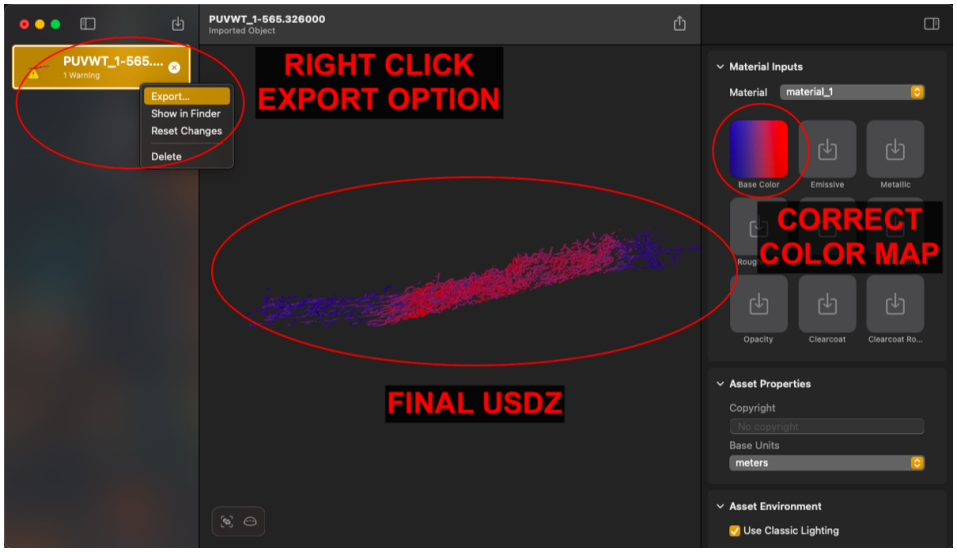

FIG. 13: Exporting USDZ using Reality Converter. Note the image used as color map is placed in the Base Color slot.

and it may be necessary to temporarily remove these commands from the profiles while using USDZ Tools. Also, it is worth noting the storage-size reduction by approximately 50%; this is because the USDZ files are binary data files. The benefits of designing the apps with Unity 3D are extending the AR applicability to Android and AR goggles devices (e.g., HoloLens) and increasing opportunities to develop interactive manipulations, such as image recognition, as shown in Paeres et al. (2021a) (see Fig. 14). Furthermore, Fig. 15 shows the final result of an AR object invoked and placed over a table; this visualization was done through an iOS device, displaying iso-surfaces of thermal fluctuations of supersonic turbulent boundary layers subject to strong concave-convex curvatures (positive values in red, i.e., hot fluid; and negative values in blue, i.e., cold fluid). The AR object was created following the methodology explained in this work.

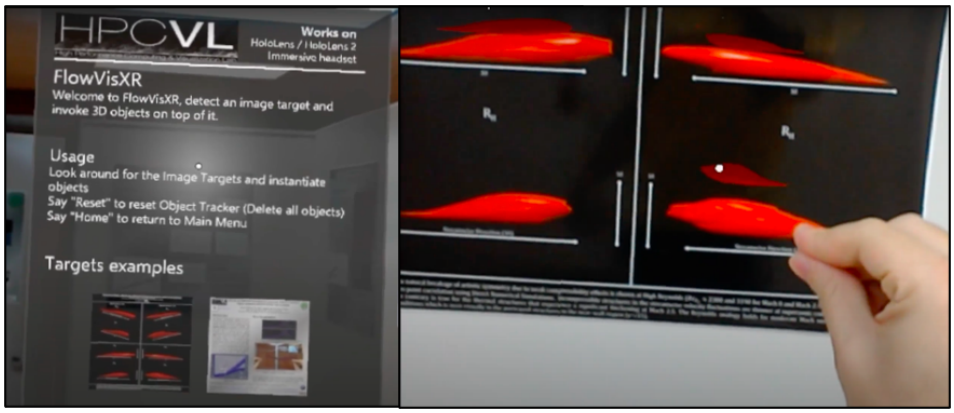

FIG. 14: Image recognition from a poster display and Microsoft HoloLens 1

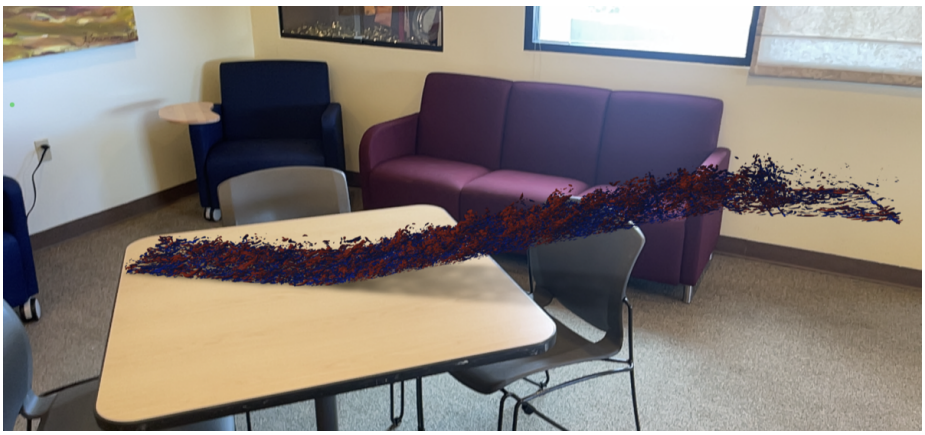

FIG. 15: AR object (iso-surfaces of thermal fluctuations) visualized through an iOS device

4. CONCLUSIONS

This work presents a state-of-the-art methodology for the immersive visualization of virtual objects with mixed reality (i.e., virtual and augmented reality). Whereas the specific application has been focused on DNS of spatially developing turbulent boundary layers, extension of the acquired knowledge to other disciplines is straightforward. This new methodology improves our previous VWT, combining a video game engine platform with scientific visualization software and utilizing one of the most common file formats for 3D scenes and models (i.e., GLTF). Unity and ParaView allow the data conversion, model creation, and scene animation to be automated, a big help for CFD simulations containing multiple files corresponding to each time step solution. The present work was executed with a workstation Dell Tower 5810, HTC Vive VR kit, Microsoft HoloLens 1st gen. hardware, and an iPhone 12 Pro-Max with iOS 15. All utilized software had a free version license or was open-source for public use. Along with the supplied user guide, the algorithms, scripts, and codes developed in this research can be accessed at our research group's webpage (CTVLab, 2023) for future replications by the reader. XR technology is amazingly growing in applications and methods, and the follow-up process represents a real challenge. For example, using additional libraries, AR devices can recognize images designated as triggers for the emergence of virtual objects from a database into the device's eyesight. In the author's opinion, the evolution of these technologies is so accelerated that a pitfall has been identified. The constant tools' creations and updates are now frequently causing compatibility issues, leading to the withdrawal of the development and support of potentially powerful tools. As part of future work, we plan to explore the potential inclusion of the Varjo XR-3, which is currently the most advanced XR headset available on the market, and the GLTF binary counterpart format (GLB) in our ongoing research. Finally, interested readers are invited to check out our VR/AR flow animation videos at Flow-Animation-Gallery (2023).

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation (Grant Nos. 2314303, 1847241, HRD-1906130, and DGE-2240397). GA acknowledges financial support from AFOSR #FA9550-23-1-0241.

REFERENCES

Albert, R., Patney, A., Luebke, D., and Kim, J., Latency Requirements for Foveated Rendering in Virtual Reality, *ACM Trans. Appl. Perception (TAP)*, vol. **14**, no. 4, pp. 1–13, 2017.

Antycip, ST Engineering Antycip, accessed from https://steantycip.com/, 2022.

Aoki, H. and Yamamoto, M., Development of Three-Dimensional Blade Design System Using Virtual Reality Technique, *J. Flow Vis. Image Process.*, vol. **7**, no. 1, 2000.

Araya, G., Castillo, L., Meneveau, C., and Jansen, K., A Dynamic Multi-Scale Approach for Turbulent Inflow Boundary Conditions in Spatially Evolving Flows, *J. Fluid Mech.*, vol. **670**, pp. 518–605, 2011.

Araya, G. and Lagares, C., Implicit Subgrid-Scale Modeling of a Mach-2.5 Spatially-Developing Turbulent Boundary Layer, *Entropy*, vol. **24**, no. 4, p. 555, 2022.

Araya, G., Lagares, C., and Jansen, K., AFOSR 2022: Effects of Wall Curvature on Hypersonic Turbulent Spatially-Developing Boundary Layers, *2022 AFOSR/ONR/HVSI Annual High-Speed Aerodynamics Portfolio Review*, p. 19, 2022.

Atzori, M., Köpp, W., Chien, S.W.D., Massaro, D., Mallor, F., Peplinski, A., Rezaei, M., Jansson, N., Markidis, S., Vinuesa, R., Laure, E., Schlatter, P., and Weinkauf, T., In Situ Visualization of Large-Scale Turbulence Simulations in Nek5000 with ParaView Catalyst, *J. Supercomput.*, vol. **78**, no. 3, pp. 3605–3620, 2022.

Barlow, R.S. and Johnston, J.P., Structure of a Turbulent Boundary Layer on a Concave Surface, *J. Fluid Mech.*, vol. **191**, pp. 137–176, 1988.

Boyles, B., Virtual Reality and Augmented Reality in Education, Center for Teaching Excellence, United States Military Academy, West Point, NY, 2017.

Brookes, J., Warburton, M., Alghadier, M., Mon-Williams, M., and Mushtaq, F., Studying Human Behavior with Virtual Reality: The Unity Experiment Framework, *Behav. Res. Methods*, vol. **52**, no. 2, pp. 455–463, 2020.

Brown, G.L. and Roshko, A., On Density Effects and Large Structure in Turbulent Mixing Layers, *J. Fluid Mech.*, vol. **64**, no. 4, pp. 775–816, 1974.

Bryson, S., The Virtual Windtunnel: Visualizing Modern CFD Datasets with a Virtual Environment, *Proc. of the 1993 Conf. on Intelligent Computer-Aided Training and Virtual Environment Technology*, Houston, TX, 1993.

Bryson, S., Virtual Reality: A Definition History—A Personal Essay, *Comput. Sci. Human-Comput. Inter.*, arXiv:1312.4322, 2013.

Bryson, S. and Levit, C., The Virtual Windtunnel: An Environment for the Exploration of Three-Dimensional Unsteady Flows, *Proc. Vis.*, vol. **91**, pp. 17–24, 1991.

Craig, A.B., Sherman, W.R., and Will, J.D., *Developing Virtual Reality Applications: Foundations of Effective Design*, Burlington, MA: Morgan Kaufmann, 2009.

CTVLab, Computational Turbulence and Visualization Lab, accessed from https://ceid.utsa.edu/garaya/, 2023.

Delgado, J.M.D., Oyedele, L., Demian, P., and Beach, T., A Research Agenda for Augmented and Virtual Reality in Architecture, Engineering and Construction, *Adv. Eng. Inform.*, vol. **45**, p. 101122, 2020.

Eissele, M., Kreiser, M., and Ertl, T., Context-Controlled Flow Visualization in Augmented Reality, *Graph. Interface*, pp. 89–96, 2008.

Elbamby, M.S., Perfecto, C., Bennis, M., and Doppler, K., Toward Low-Latency and Ultra-Reliable Virtual Reality, *IEEE Network*, vol. **32**, no. 2, pp. 78–84, 2018.

Flavián, C., Ibáñez-Sánchez, S., and Orús, C., The Impact of Virtual, Augmented and Mixed Reality Technologies on the Customer Experience, *J. Business Res.*, vol. **100**, pp. 547–560, 2019.

Flow-Animation-Gallery, Computational Turbulence and Visualization Lab, accessed from https://ceid.utsa.edu/garaya/research/, 2023.

Friendly, M. and Denis, D.J., Milestones in the History of Thematic Cartography, Statistical Graphics, and Data Visualization, York University, Toronto, 2006.

Hansen, C.D. and Johnson, C.R., *Visualization Handbook*, Amsterdam: Elsevier, 2011.

Henderson, A., ParaView Guide, A Parallel Visualization Application, Kitware Inc., 2007.

Hesselink, L., Digital Image Processing in Flow Visualization, *Ann. Rev. Fluid Mech.*, vol. **20**, no. 1, pp. 421–486, 1988.

Hollister, B.E. and Pang, A., *A Concise Introduction to Scientific Visualization: Past, Present, and Future*, Berlin: Springer Nature, 2022.

Imaichi, K. and Ohmi, K., Numerical Processing of Flow-Visualization Pictures–Measurement of Two-Dimensional Vortex Flow, *J. Fluid Mech.*, vol. **129**, pp. 283–311, 1983.

Jha, A.K., Shukla, P., Ghosh, P., Khisti, P., and Dubey, A., Ivsualization and Measurement of Natural Convection Boundary Layer by Particle Image Velocimetry, *J. Flow Vis. Image Process.*, vol. **30**, no. 2, 2022.

Lagares, C., Rivera, W., and Araya, G., Scalable Post-Processing of Large-Scale Numerical Simulations of Turbulent Fluid Flows, *Symmetry*, vol. **14**, no. 4, p. 823, 2022.

Lagares, C.J., Paeres, D., and Araya, G., Wall Temperature Effect on Thermal Coherent Structures over Supersonic Turbulent Boundary Layers Subject to Surface Curvature, *Proc. of the 74th Annual Meeting of the APS Division of Fluid Dynamics*, Phoenix, AZ, 2021a.

Lagares, C.J., Rivera, W., and Araya, G., Aquila: A Distributed and Portable Post-Processing Library for Large-Scale Computational Fluid Dynamics, *AIAA SciTech*, 2021b.

Lee, E.A.L., Wong, K.W., and Fung, C.C., How Does Desktop Virtual Reality Enhance Learning Outcomes? A Structural Equation Modeling Approach, *Comput. Ed.*, vol. **55**, no. 4, pp. 1424–1442, 2010.

Lewiner, T., Lopes, H., Vieira, A.W., and Tavares, G., Efficient Implementation of Marching Cubes' Cases with Topological Guarantees, *J. Graph. Tools*, vol. **8**, no. 2, pp. 1–15, 2003.

Linde, Linde Virtual Academy, accessed from https://vr.linde.com/, 2022.

Lorensen, W.E. and Cline, H.E., Marching Cubes: A High Resolution 3D Surface Construction Algorithm, *SIGGRAPH Comput. Graph.*, vol. **21**, no. 4, pp. 163–169, 1987.

Louis, T., Troccaz, J., Rochet-Capellan, A., and Bérard, F., Is It Real? Measuring the Effect of Resolution, Latency, Frame Rate and Jitter on the Presence of Virtual Entities, *Proc. of the 2019 ACM Int. Conf. on Interactive Surfaces and Spaces*, New York, NY, USA, pp. 5–16, 2019.

Lund, T., Wu, X., and Squires, K., Generation of Turbulent Inflow Data for Spatially-Developing Boundary Layer Simulations, *J. Comput. Phys.*, vol. **140**, no. 2, pp. 233–258, 1998.

Malakhov, A., Liu, D., Gorshkov, A., and Wilmarth, T., Composable Multi-Threading and Multi-Processing for Numeric Libraries, *Proc. of the 17th Python in Science Conf.*, Austin, TX, pp. 18–24, 2018.

McCormick, B.H., Visualization in Scientific Computing, *Comput. Graph.*, vol. **21**, no. 6, 1987.

Message Passing Forum, A Message-Passing Interface Standard, Tech. Rep., USA, 1994.

Milgram, P. and Kishino, F., A Taxonomy of Mixed Reality Visual Displays, *IEICE Trans. Inf. Syst.*, vol. **77**, no. 12, pp. 1321–1329, 1994.

Mohd, J., Murugan, T., and Das, D., Transient Characteristics of the Trailing Jet of a Compressible Vortex Ring at Mach 1.5, *J. Flow Vis. Image Process.*, vol. **29**, no. 4, 2022.

Nielson, G., Hagen, H., and Müller, H., Scientific Visualization, Tech. Rep., Institute of Electrical & Electronics Engineers, 1997.

Paeres, D., Lagares, C.J., and Araya, G., The Use of Augmented Reality (AR) in Flow Visualization, *Proc. of the 74th Annual Meeting of the APS Division of Fluid Dynamics*, Phoenix, AZ, 2021a.

Paeres, D., Lagares, C.J., and Araya, G., Dynamic Fully Immersive Virtual Reality of Supersonic Flows, *Proc. of the 75th Annual Meeting of the APS Division of Fluid Dynamics*, Indiannapolis, IN, 2022.

Paeres, D., Lagares, C.J., Santiago, J., Craig, A.B., Jansen, K., and Araya, G., Turbulent Coherent Structures via VR/AR, *Proc. of the 73th Annual Meeting of the APS Division of Fluid Dynamics*, Virtual, 2020.

Paeres, D., Santiago, J., Lagares, C.J., Rivera, W., Craig, A.B., and Araya, G., Design of a Virtual Wind Tunnel for CFD Visualization, *AIAA Scitech 2021 Forum*, p. 1600, Virtual, 2021b.

Paeres Castaño, D., Assessment of Turbulent Boundary Layer Detachment Due to Wall-Curvature-Driven Pressure Gradient, MSc, University of Puerto Rico-Mayaguez, 2022.

Pheatt, C., Intel® Threading Building Blocks, *J. Comput. Sci. Coll.*, vol. **23**, no. 4, p. 298, 2008.

Purushottam, K., Chandramouli, K., Sree Naga Chaitanya, J., and Gowreswari, B., A Review on Virtual Reality and Augmented Reality in Architecture, Engineering and Construction Industry, *Int. J. Modern Trends Sci. Technol.*, vol. **7**, pp. 28–33, 2021.

Radianti, J., Majchrzak, T.A., Fromm, J., and Wohlgenannt, I., A Systematic Review of Immersive Virtual Reality Applications for Higher Education: Design Elements, Lessons Learned, and Research Agenda, *Comput. Ed.*, vol. **147**, p. 103778, 2020.

Rasquin, M., Marion, P., Vishwanath, V., Matthews, B., Hereld, M., Jansen, K., Loy, R., Bauer, A., Zhou, M., Sahni, O., Fu, J., Liu, N., Carothers, C., Shephard, M., Papka, M., Kumaran, K., and Geveci, B., Co-Visualization of Full Data and in Situ Data Extracts from Unstructured Grid CFD at 160k Cores, *Proc. of the 2011 Companion on High Performance Computing Networking. Storage and Analysis Companion, SC 11 Companion.*, Seattle, WA, pp. 103–104, 2011.

Rosenblum, L., Virtual and Augmented Reality 2020, *IEEE Comput. Graph. Appl.*, vol. **20**, no. 1, pp. 38–39, 2000.

Saidin, N.F., Halim, N., and Yahaya, N., A Review of Research on Augmented Reality in Education: Advantages and Applications, *Int. Ed. Studies*, vol. **8**, no. 13, pp. 1–8, 2015.

Schroeder, W., Martin, K., and Lorensen, B., *The Visualization Toolkit*, Clifron Park: Kitware, 2006.

Sherman, W.R., Craig, A.B., Baker, M.P., and Bushell, C., Scientific Visualization, The Computer Science and Engineering Handbook, Chap. 35, A.B. Tucker Jr. (Ed.), Boca Raton, FL: CRC Press, 1997.

Tecplot 360, accessed February 12, 2022, from https://www.tecplot.com/products/tecplot-360/, 2022.

The HDF Group, Hierarchical Data Format Version 5, accessed from http://www.hdfgroup.org/HDF5, 2010.

Ververidis, D., Nikolopoulos, S., and Kompatsiaris, I., A Review of Collaborative Virtual Reality Systems for the Architecture, Engineering, and Construction Industry, *Architecture*, vol. **2**, no. 3, pp. 476–496, 2022.

VisIt, accessed December 24, 2022, from https://wci.llnl.gov/simulation/computer-codes/visit, 2022.

Walcutt, N.L., Knörlein, B., Sgouros, T., Cetinić, I., and Omand, M.M., Virtual Reality and Oceanography: Overview, Applications, and Perspective, *Front. Mar. Sci.*, vol. **6**, p. 644, 2019.

Watson, B., Spaulding, V., Walker, N., and Ribarsky, W., Evaluation of the Effects of Frame Time Variation on VR Task Performance, *Proc. of IEEE 1997 Annual International Symposium on Virtual Reality*, Albuquerque, NM, pp. 38–44, 1997.

Winant, C.D. and Browand, F.K., Vortex Pairing: The Mechanism of Turbulent Mixing-Layer Growth at Moderate Reynolds Number, *J. Fluid Mech.*, vol. **63**, no. 2, pp. 237–255, 1974.

Zhang, C., Investigation on Motion Sickness in Virtual Reality Environment from the Perspective of User Experience, *2020 IEEE 3rd Int. Conf. on Information Systems and Computer Aided Education (ICISCAE)*, pp. 393–396, 2020.