DOI: 10.2352/ISSN.2470-1173.2016.4.ERVR-417

Implementing Native Support for Oculus and Leap Motion in a Commercial Engineering Visualization and Analysis Platform

Anastacia MacAllister[a], Tsung-Pin Yeh[b], Eliot Winer[a]; [a]Iowa State University – Ames, Iowa; [b]Siemens PLM Software – Ames, Iowa

Abstract

While previous research in academia points to the ability of Natural User Interfaces (NUIs) and low-cost display devices to help users better understand a design, there does not exist much research on how these devices can be integrated into existing legacy code used by engineering and design firms. The lack of commercial engineering software that integrates NUIs and low-cost display devices, like the Oculus Rift, can be attributed to the fast changing device market and the lack of awareness many engineering software makers show in emerging interaction paradigms. The lack of work in the area of integrating low-cost immersion devices into commercial software creates a barrier for adoption of these new devices and interaction paradigms. The work presented in this paper details a proof of concept system integrating the Leap Motion and Oculus Rift, into a commercial engineering visualization and analysis package called Siemens' Teamcenter® Lifecycle Visualization Mockup (Mockup). Based on the recorded performance data, hooking up both the Leap and the Oculus results in a frame rate of around 30 frame per second. Indicating that these two devices together can provide real time, fluid interaction in a commercial engineering platform.

Introduction

Research in academia points to the ability of low-cost Virtual Reality (VR) immersion devices to improve a user's ability to understand designs [1-3]. With today's fast passed globalized economy, interdisciplinary design teams need these tools that help bridge the gap between varying backgrounds, allowing them to make more informed decisions faster [4-6].

While previous research in academia points to the ability of NUIs and low-cost display devices to better aid the user in developing a mental model there does not exist much research on how these devices can be integrated into existing legacy code bases that many engineering and design firms use. The lack of work in the area of integrating low-cost immersion devices into commercial software creates a barrier for adoption for these new devices and interaction paradigms. Work presented in this paper aims to accelerate the adoption of these devices for the general working population by looking at how low-cost immersion devices can be integrated in to commercial engineering visualization software.

The purpose of the work presented in this paper is to demonstrate a proof of concept system that integrates low-cost immersion devices into a commercial engineering visualization and analysis package. The problem the proof of concept system aims to address is the lack of commercially available solutions that fuse together research and academic benefits associated with low-cost immersion devices and commercial software. For this work a low-cost immersion device is considered a device that is priced at a consumer level, plus facilitates more natural intuitive interaction between the user and the computer when compared with the traditional mouse and keyboard. Specifically, device wise, the research presented in this paper looks at brining together NUIs and low-cost HMD devices to a commercial software package. The aim is help users develop a better design understanding and allow them to intuitively manipulate and interact with the design. For the proof of concept, the low-cost immersion devices used are an interaction device called the Leap Motion and a head-mounted display (HMD) called the Oculus Rift. The engineering software package used is *Teamcenter® Lifecycle Visualization Mockup (Mockup)*. The work presented in this paper summarizes the process of integrating such devices into commercial code and how well these devices provide a fluid user experience.

Background

Recently low cost immersion devices have burst onto the scene and grabbed headlines. Examples include Facebook's acquisition of Oculus and the release of motion capture devices like the Leap Motion. The capabilities of these new devices have garnered not only the publics' attention, but also the attention of both academia and industry. That attention is due to the devices ability to impart novel user experiences and facilitate increased design understanding. All of this potential comes at a fraction of the cost of traditional VR systems like CAVEs™.

Much research has been focused on exploring the potential these devices hold. Existing research touches on everything from aiding a physically impaired user in their interactions with computers to creating simplified interfaces for design review. While the devices intrigue industry, like academia, commercial applications lag behind in investigating these promising new mediums of interaction. This is due to the fast changing landscape of these new devices. Many traditional engineering software companies have yet to integrate any of these products directly into their commercial offerings. This creates a barrier to acceptance and adoption in the professional realm. If these promising devices continue to only exist in research code the professional cannot gain expertise with or evaluate these devices. Thus they cannot contribute valuable input to the creation of these new interfaces and display modes.

Teams and Technology

Product design in today's globalized world requires quick turnarounds to maintain profitability. As a result, many design teams are becoming an amalgamation of diverse interdisciplinary backgrounds in order to take advantage of concurrent design. This shift to the concurrent design approach allows companies to push products out in a faster more accurate manner [4]. However, this push towards interdisciplinary design teams means that these diverse teams need tools to communicate. There needs to be a way for the engineering and the marketing professional to be on the same page. In addition, not only are teams made up of different background, like engineers and marketers, but also they can often be distributed across the country or the world. Effective work tools such as visualization or product life-cycle management software can mitigate the challenges of diverse working backgrounds and distributed work environments [5, 6].

DOI: 10.2352/ISSN.2470-1173.2016.4.ERVR-417

While software tools can help facilitate communication between members and streamline the design process, developing a platform that all members of the team are comfortable using is a challenging task. When creating a system to aid the design process researchers need to take special time to consider all team members involved. A team member's technical skill level can vary greatly from the engineer all the way to upper management. In order to reap the benefits of concurrent design all members need to be involved in the design process from the beginning. This ensures they are able to understand the design enough to contribute time and ideas to shape the final product.

To try and maximize the effectiveness of the software tools for distributed and interdisciplinary teams much research has been done on how teams collaborate and the role software can play in that process [7-9]. From previous research it is apparent how important these tools are for aiding communication. However, while these tools can be invaluable, it is difficult to make them suitable for all team members' skill levels. Technical programs like computer-aided design (CAD) usually drive the design process. This makes the goal of letting every team member, especially those who lack a technical background, interact with the design especially challenging [10]. This interaction challenge is due to the large amount of learning required to master technical complex engineering software. These new interaction devices have the potential to lower the barrier of entry and greatly aid the user's design understanding at a fraction of the cost of traditional installations.

Virtual Reality

VR principles can greatly improve users experiences with engineering software programs like CAD because VR principles, like head-tracking, tend to be more user oriented than traditional keyboard based interaction models [11]. While VR interaction modes are more user-friendly, engineers and product designers still require the mathematically accurate model representations produced by engineering software. Berta theorized that if engineering software and VR were ever combined, users would be given the user centered principles that come with VR but the model based accuracy that comes with CAD [1]. By combining the two the aim is to make a program helpful to both the technical engineers and other team members. In addition to the usability aspect, VR provides the ability to integrate a sense of presence and immersion into the process of investigating designs. These senses can be invaluable when trying to make informed design decisions since it provides an increase in understanding and intent [2, 12].

While traditional large scale VR has the ability to provide numerous benefits, the implementation of such setups can be problematic. Challenges include the large costly nature of the installations, the purchasing of specialty software, and the necessary domain knowledge to run such a system. These barriers serve to create serious bottlenecks and do not allow the vast majority of worker's access to the technology. This lack of access serves to seriously blunt the potential benefits of its use [13].

While these large costly installations are the gold standard for immersion, they are not always necessary to reap the benefits of VR technology [14]. Lower cost devices such as the Oculus can produce experience on par with large costly installations for a fraction of the price [16]. Using these lower cost devices does come with a slight tradeoff in performance and immersion, but as pointed out in work by McMahan full immersion is not always necessary to produce enhanced design understanding [15]. In addition in some cases namely product design, immersion level and design understanding are more dependent on interaction paradigms than the technology used to display the information, meaning that these low-cost HMDs can provide a visual display of information similar to a costly walled system [3, 18].

Natural User Interfaces

As detailed above, previous work in VR suggests that it has the ability to help aid users' understanding of a design. In addition to display technology, user understanding can be aided through interaction with the on screen environment. Namely using Natural User Interfaces (NUIs), like gestures, to interact with entities instead of a mouse and keyboard [1, 19-32].

Tumkor et al. shows how low-cost interaction devices can improve interaction and collaboration [33]. They found through a user study that performing some tasks when using the modeling program SolidWorks are faster when using gestures. Overall, they found that gestures as an interaction paradigm hold considerable promise, but are hampered by the mixing of traditional mouse/keyboard based interaction metaphors and new NUI based interaction modes. They conclude that to move forward, applications that use gesture interfaces need to be designed from the ground up, using only NUI principles and interaction modes to avoid user confusion.

Song et al. developed an application that interacts with a CAD system using gaze and finger control [34]. Their research aimed to tackle the issue of user fatigue when using gestures. For their program they identified a subset of actions that are considered primary CAD tasks. The tasks they identified are translation, rotation, and zooming. In their study they interviewed users after testing their program. Results indicated that users found the gestures to be far more intuitive to use than the mouse and keyboard, but the users still reported the mouse and keyboard to be more comfortable than the gestures.

Araullo and Potter perform a small user study with commodity interaction devices, the Leap Motion and Oculus Rift, to test how users respond to interacting with these devices [35]. They found that the most important aspect when designing gesture interfaces is to keep things simple to avoid user confusion. A factor that especially creates confusion is the mixing of interface types. The work advocates designing systems around the NUI and strongly cautions against mixing mouse and keyboard mental models with NUI principles.

Commodity Hardware in Research and Industry

While NUIs play an important role in increasing a users understanding of the design by allowing intuitive interaction, the display technology is also a very important piece of the puzzle. As mentioned above HMDs can be a viable alternative to costly and bulky CAVE™ systems. The use of these HMDs over standard 2D desktop displays allows designers to create a better mental model of the design. When these low-cost display devices are coupled with NUIs they provide a natural cost effective way to see and interact with the design or environment. This interaction and viewing greatly aids a users understanding [16, 36, 38]. Increasing the likelihood that they can identify and correct any design defects earlier in the design process reducing change costs.

Researchers have taken notice of the potential of low cost VR. They have already started to leverage the combination of NUIs and emerging low-cost immersion devices. Research in the field shows the potential of NUI principles and low-cost display devices to transform user interaction [38-41].

DOI: 10.2352/ISSN.2470-1173.2016.4.ERVR-417

Work by Karolczak and Klepaczko is one such example of the pairing of NUIs and low-cost HMDs [42]. The researchers use an Oculus and a Leap Motion to allow a user to explore medical images in a 3D environment. In the paper the researchers tout the relatively low cost of the viewer and its ability to provide fine control for investigation for a fraction of the cost of other systems. The researchers theorize that at a relatively low cost the system could be widely deployed allowing a large number of doctors and students the ability to use the powerful learning tool.

Thompson, is an example of a Leap being used in an engineering design type environment [43]. The work looked at using the Leap Motion for manipulating a conceptual engineering design program. Results showed that the relatively accurate Leap provides a viable way to quickly manipulate CAD models [44]. The one detractor the researcher mentioned was that, in order for these types of interfaces to catch on, no matter how promising they are, they need to be integrated into a commercial offering to gain support and acceptance.

MacAllister et al. takes steps towards addressing this issue by exploring the use of an NUI in an existing engineering design package. The research looked at using a Kinect for large-scale design review in a commercial program called *Teamcenter® Lifecycle Visualization Mockup (Mockup)* [45]. The program was a proof of concept. It integrated a commercial low-cost immersion device, the Kinect, with the existing commercial code though a plug-in. The program allowed the user to interact with the model rotating it left or right, use the immersive wand and allowed the user to bring up Mockup's immersive menu. From the work and feedback the authors received, the proof of concept program was a step in the right direction. The concept program allowed the user to fluidly and intuitively interact with the design, but had a limited feature set.

Additional work by MacAllister focused on evaluating the NUI by seeing how it performed against the existing user interface in Mockup [46]. The analysis showed that the NUI program worked well for the user from a learning and execution time point of view and it improved upon the existing interface for everyday core tasks. However, the work found that the Kinect at times created user fatigue and the large throw distance (~5 ft.) for the Kinect limited its use. The space requirement limited the programs use to large areas, eliminating the possibility of letting the desktop engineer have a new tool at their personal workstation.

The two drawbacks of user fatigue and large space requirements drove the exploration of the Leap motion over the Kinect as a new interface device. In addition, in previous work the authors mainly focused on the NUI. Previous work did not incorporate the benefits to design understanding produced by portable low-cost HMDs.

While academics use of NUIs and low-cost immersion devices marks progress in developing new interaction models, true change will not happen until commercial offerings start to integrate these devices and principles. This work aims to jump start that process by producing a proof of concept system that integrates NUIs and low-cost immersion devices into commercial engineering software.

Method

Device Selection

Over the past few years the market has been flooded with new commodity low-cost immersion devices like the Kinect, WiiMote, Oculus Rift, and Leap Motion. With the consistently shifting market keeping track of the latest most promising devices can be a challenge. The authors selected the technology used in this work based on experience from former projects and though researching current market offerings [45, 46]. The selections for display devices and motion controlling devices are justified below.

Display Device – Oculus Rift DK2

Criteria for the display device were that it had to be low-cost commodity hardware, less than one-thousand dollars, and relatively close to a mature, released product. The maturity of the device was an important factor, with the changing low-cost immersion device market many promises are made, but few actually deliver. The Rift was selected in part because, it was relatively low-cost when compared with other HMDs on the market like the HTC Vive or Sony HMZ-T3. The authors were constrained, though, during HMD selection by the fact that Mockup runs on a desktop. As a result, phone compatible devices like the VROne were not included in the review. In addition to the cost requirement, the Oculus was also selected due to its maturity. The pre-released prototypes by Oculus and the early 2016 projected release date indicate that the device is closer to maturity than many of its peers. This near production state of the Rift integrates well with the stated goal of the work, to take steps to develop a proof of concept system that could actually be rolled out into production in the near future.

In the end, the Oculus Rift used for the proof of concept system was a second-generation version, shown below in Figure 1 with the Leap Motion attached. The resolution of the device was 960x1080 pixels per eye. The field of view of the device was one-hundred degrees.

FIGURE 1: OCULUS RIFT DK2 HEADSET WITH LEAP

Interaction Device – Leap Motion Controller

The low-cost commodity HMD market generates a lot of buzz but not many mature products, limiting the number of offerings to choose from at this point. In the interaction device market, however, there are many mature devices. Devices such as the WiiMote, Kinect, Leap Motion and Playstation Move are just a few such commodity interaction devices. As far as devices go, the majority such as the WiiMote, Kinect and PlayStation Move are suited to large gesture based actions not smaller desktop workstations. These devices, due to hardware limitations, are unable to register finer movements like the Leap.

The proof of concept system described in this paper is geared towards using finer hand movements captured at close distances, relative to the controller. The device selection was based on previous work indicating that while devices such as the Kinect could produce a valid proof of concept system, the space, fatigue and

DOI: 10.2352/ISSN.2470-1173.2016.4.ERVR-417

accuracy constraints hampered the system's potential reach [45, 46]. Based on the constraints and previous work the authors selected the Leap Motion due to a combination of its tracking accuracy, low-cost, and small space requirements [44].

Gesture Selection

During the conceptualization phase of the program gesture selection was guided by standards used in previous work [34, 45, 46]. Selected based on these guidelines were a subset of gestures that are representative of everyday CAD actions. Only a limited number of actions are integrated into the program, because the program is a proof of concept. The authors wanted to start out manageable and prove first that low-cost display and interaction devices could be integrated successfully into a commercial engineering visualization package.

The actions the authors decided to use for the system's interface were an on/off gesture for the hand display mode, direction based rotation, and a zoom in/out feature. These gestures were selected as a representative subset of CAD based actions common to everyday engineering tasks [34].

The first gesture is the on/off feature for turning on hand tracking. By making a closed right fist the user can turn on the virtual hands and gesture recording functionality. By turning on the hand-tracking mode the user can interact with the on screen geometry using the gestures detailed above. In addition, the user can see a virtual representation of their hands in the virtual environment. Figure 2 is an example of the hands a user sees when hand-tracking mode is active. To turn off hand tracking mode the user has to make a fist with their left hand. The authors decided to use these mirrored gestures since research in NUIs suggests users find mirrored gestures for on/off actions highly intuitive [47].

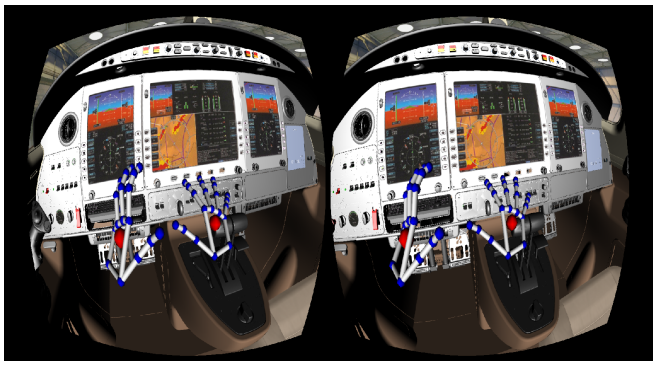

FIGURE 2: OCULUS VIEW OF HANDS IN MOCKUP WITH DISTORTION SHADER

The second gesture, direction based rotation shown in Figure 3, allows the user to use the Leap application programming interface (APIs) built in swipe gesture to rotate the model. The swipe gesture allows the user to rotate the model about any axis by recording the direction vector of the swipe gesture. This allows the user full freedom to explore any part of the model. Work in similar areas only allowed rotation of the model around a program specified axis [45]. This created frustration and the inability to fully investigate the model. The proof of concept system presented in this paper allows the user greater control over the model, thus giving them more ability to investigate and enhance their understanding of the design.

FIGURE 3: LEAP MOTION SWIPE GESTURE[48]

The third gesture, the zoom in/out feature was assigned to the hands moving toward/away gesture. Moving the hands away from each other, in a manner similar from point 1 to 2 shown in Figure 4, causes the scene on screen to become larger. The opposite, moving the hands towards each other causes the scene to become smaller. This method was implemented using the motions feature of the Leap API. However, the initial scaling was too sensitive and created many false positives. In order to combat this issue, the authors implemented a certainty function to only register a scale event over a number of frames and above a certain certainty threshold. After implementing these safeguards, the feature became more reliable.

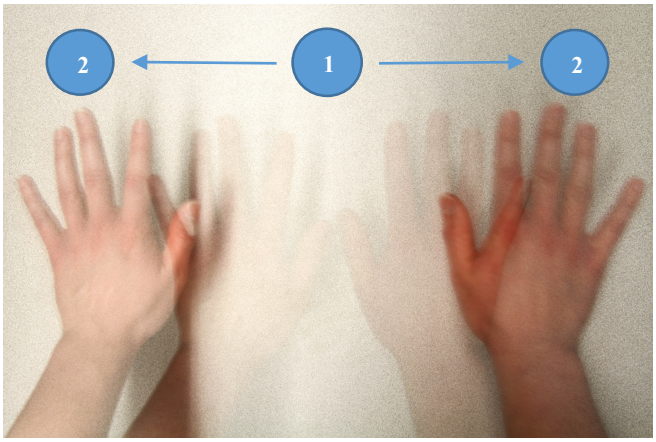

FIGURE 4: HAND SCALING ZOOM IN

Program Implementation

Implementing integration of the Oculus Rift and Leap Motion into Mockup can be separated into three main parts; information flow, distortion shader and hand tracking. The two main challenges encountered when developing the program were integrating the display device into the existing codebase and integrating the handling all the information from the Leap Motion and Oculus for display in Mockup. Figure 5 shows a use case for the system. The

DOI: 10.2352/ISSN.2470-1173.2016.4.ERVR-417

user is viewing a virtual cockpit from the Oculus and can interact with the physical model while seeing the high detail VR representation. This use case is representative of an engineering design review. In such a review engineers would bring in members of the management team to look at a design and make comments. Previous research presented in the Background section points to the ability of devices such as the Oculus to increase the user's ability to understand a design. This could help the design review team more quickly make informed decisions all without having to make expensive detailed physical models that may change after a review. However, while these devices show promise little research exists on integrating them into a commercial engineering visualization and analysis package like the example shown in the figure below.

FIGURE 5: DEMO SYSTEM SETUP

Information Handling

Data was captured by the Leap and Oculus though an extensible data structure within the native Mockup code. This extensible format served to isolate the device code from the more Mockup specific code, insulating the core functionality from any lower level changes. The Leap data structure contains the types of gestures and motions that can be registered by the program. In addition, information specific to a gesture type is also contained, like the swipe direction for the rotate gesture.

Figure 6 shows an overall diagram of code structure and the components involved in the proof of concept system. In the program architecture the Leap Motion and the Oculus hardware are

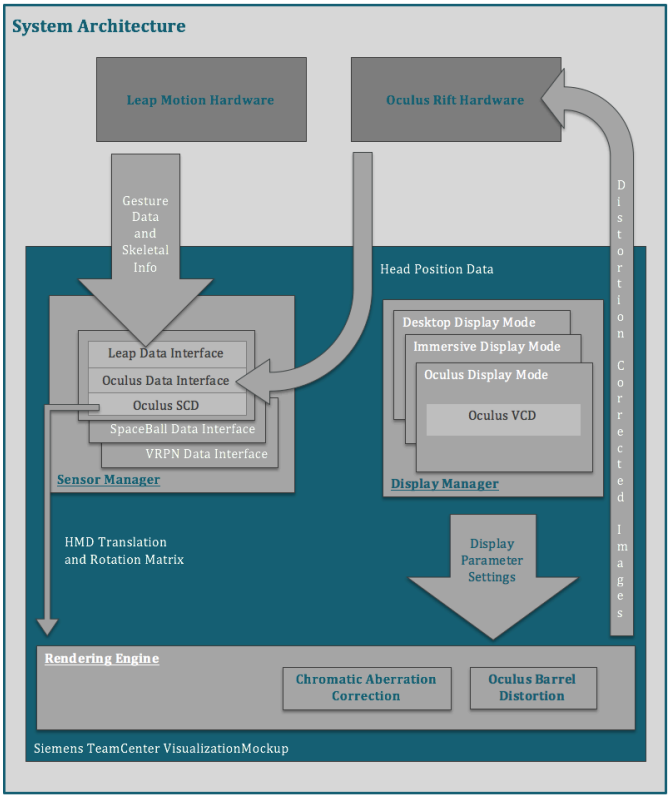

FIGURE 6: SYSTEM ARCHITECTURE

integrated natively into Mockup. This integration is facilitated by the extensive architecture in place to deal with various input devices.

Mockup is designed to handle different sensor and display devices, like the 3D SpaceMouse™. Sensors devices like the Leap are handled by the Sensor Manager, which directs traffic from hardware specific API code into Mockup specific commands. In order to add the integration of the Leap Motion, the authors had to write code to translate Leap specific data from the API into information that could be interpreted by the Sensor Manager and sent to the rendering engine. While translating data between Leap and Sensor Manager formats was not exceedingly difficult, this translation is something that needs to be paid attention to in the future. If device makers like Leap, try to lock down APIs and give developers less freedom to access low level data this could create a major hurdle for device adoption. Programs like Mockup require their own data formats and do not want to have to adopt a new formats every time they integrate a new device. Being able to translate from a device specific data format into a standard Mockup format allows the handling of more devices with less overhead cost.

In addition to input devices, Mockup also currently handles multiple display configurations though the display manager. Supported display modes consist of immersive display environments like CAVEs™ and standard desktop environments. Configuration of these environments is performed using VCD and SCD configuration files. The VCD file sets up the immersive viewing window size and stereo viewing properties. For the Oculus it was imperative to set the correct parameters in order to make sure rendered images were represented correctly due to the highly immersive nature of the device. With users being able to look around the environment in a full 360 degrees', images displayed need to be portrayed in the correct perspective to 1) reduce the possibility of motion sickness and 2) to ensue the model would be as close to the

DOI: 10.2352/ISSN.2470-1173.2016.4.ERVR-417

actual product as possible. The SCD file sets up the transformation matrix adjustment to align the device's coordinate system with the coordinate system in Mockup. For the work presented all transformation matrices in the SCD file were set to identity, and the coordinate system alignment was handled inside the Sensor Manger.

Distortion Shader

The optics in the Oculus Rift create distortion in the images seen by the user [49]. The lenses in the Oculus, specifically, create what is known as pincushion distortion in the image. This occurs due to the use of the lenses to magnify the images shown on the screen to increase the field of view of the device. While using the lenses to increase the devices field of view creates a more immersive experience for the user, the distortion created in the image is undesired. This pincushion distortion, shown in Figure 7, is rectified in the Mockup rendering engine by applying what is know as a barrel distortion, shown in Figure 8 [50]. This distortion correction is applied to the rendered scene using OpenGL shaders within the standard Mockup graphics pipeline. Figure 2 shows a desktop screen capture of the distortion corrected image in Mockup. Notice how the image has black space around the edges, this is due to the morphing of the image with the distortion correction. While on a 2D screen the image looks misshapen, when viewing though the Oculus optics the image will look accurate to the user.

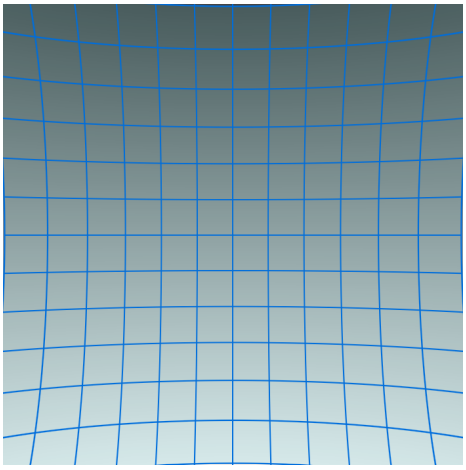

FIGURE 7: PINCUSHION DISTORTION

Since Mockup is optimized to handle large engineering geometry using specialized custom rendering routines, using the standard Oculus API for image rendering purposes was not an option. For the prototype system in the paper, the HMD needed to be fully integrated into the existing rending engine Mockup employs. This implementation of the custom distortion correction for the Oculus in Mockup was aided by the low level data access provided by the Oculus API. Due to Mockup's custom render, complete access to the data displayed by Oculus was required to make the corrections. If Oculus decided to disallow access to the raw image data this would create a barrier not only for integration into Mockup, but also to other users who employ custom rendering routines.

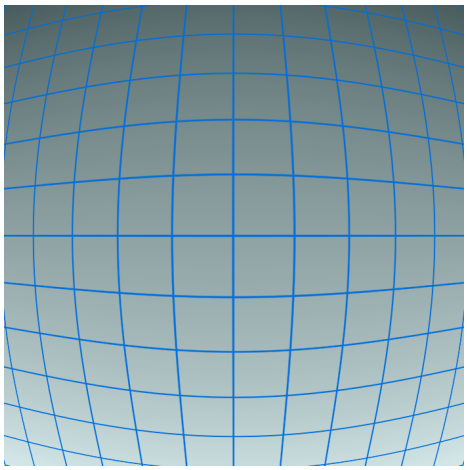

FIGURE 8: BARREL DISTORTION

In addition to the distortion issue, the lens on Oculus Rift also causes a visual effect known as chromatic aberration. This results in a colored fringe visible around the edges of the objects. The effect is especially visible around the transition point between different colors and around the edges of the view area. The "chromatic aberration" seen by the user when looking at the image though the Oculus lenses, is due to the difference in light reflection through the lenses. This is a result of all wavelengths of light passing through the curved lens having varying focal distances [51]. Viewing the image uncorrected looks similar to Figure 9. Notice the blur around the edges of objects and the rings of color around object edges such as the seat. When an image is correct for the chromatic aberration effect is looks more like Figure 10, with sharper object edges. This is due to the distortion shader pre-distorting the red, green, and blue colors of the image based on the offset from the center of the lens. Thus when the color-distorted image is projected through the curved lens, the red, green and blue colors of the image align again.

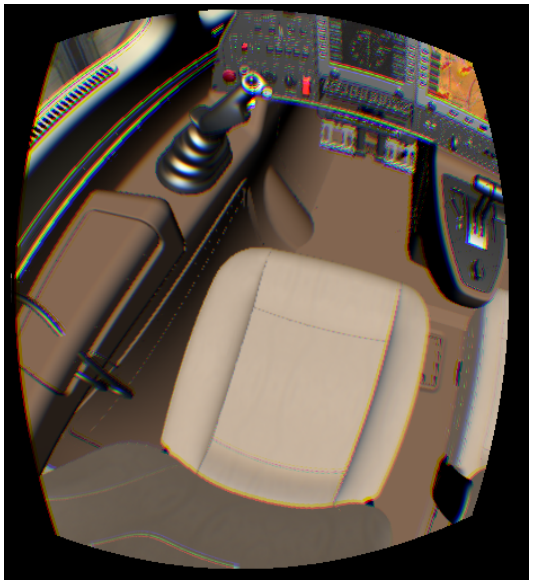

FIGURE 9: IMAGE UNCORRECTED FOR CHROMATIC ABERRATION

©2016 Society for Imaging Science and Technology
DOI: 10.2352/ISSN.2470-1173.2016.4.ERVR-417

FIGURE 10: IMAGE CORRECTED FOR CHROMATIC ABERRATION

However, even with these corrections the chromatic aberration reduction code in the distortion shader cannot completely eliminate the chromatic aberration effect. This is since the curved lens offset the color differently for all range of wavelengths, where the shader code can only pre-distort the R, G, and B colors, thus the color fringe around the edges of the objects could still be noticeable. This fringed effect can be reduced by higher samples of anti-aliasing, like 8 or 16, depending on the graphics hardware capability.

Both barrel distortion and chromatic aberration reduction are implemented inside the fragment shader. A new rendering routine was created inside the Mockup rending pipeline specifically for images displayed using the Oculus. This new rendering routine takes into account the corrections for both chromatic aberration and the lens distortion. Mockup automatically switches to the Oculus rendering routine when the device is set up using the previously mentioned configuration files. The custom rendering process for the Oculus in Mockup follows the following procedure:

1. Create a new Frame Buffer Object (FBO)
2. Bind the new FBO as the new default FBO
3. Execute the normal rendering operation. It renders the current scene into FBO
4. Unbind the FBO
5. Render a quad that covers the current view window
6. Use the Oculus Rift fragment shader to perform barrel distortion and chromatic aberration reduction by using the FBO as the pixel color lookup

Creating a custom rendering routine inside Mockup for the specific display needs of the Oculus was a fairly straight forward task once all the necessary actions were determined. Once the Oculus was hooked up to Mockup the researchers had to determine what corrections were necessary to produce an accurate high quality image. While some of the optics related issues could be predicted, it is hard to tell how a device will integrate until it is actually hooked up to a program. The data access provided by the Oculus SDK made the task less prohibitive than if the APIs tried to handle the data in a more locked down manner. However, it is a balance between usability and access. For consumer type devices, device makers want casual developers to be able to use their products, thus they offer easy to use API calls that abstract some of the complexity. However, for more commercial applications, developers require access to low level data to use in house development practices independent of the Oculus API.

Hand Tracking

In addition to the gesture data, the authors also had to correctly manipulate the Leap skeletal data for the HMD based tracking. This required transforming a number of coordinate systems. Switching from Leap to Oculus to Mockup coordinates. In addition, the authors had to make sure that the Mockup view updated with any movements from the user wearing the Oculus headset. This required number coordinate transformations and unit conversions for the model geometry and the camera view matrix.

The largest goal for the hand tracking work was to transform the Leap coordinates into Mockup world coordinates. The Leap HMD tracking SDK at the time of use was still in beta, meaning its features were still experimental and lacked some functionality. During the beta stage Leap did not change its coordinate system to match with the Oculus. The Leap controller coordinate system is set with respect to the controller and is constant no matter what orientation the controller is in. The Leap Motion coordinate system is shown below in Figure 11.

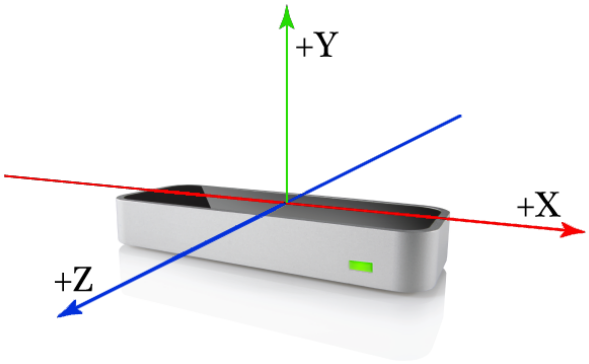

FIGURE 11: LEAP MOTION COORDINATE SYSTEM [48]

As a result of the set coordinate system, when mounted to the Oculus the Leap controller sends data that is not in the same frame of reference as the Oculus data. This creates an issue because, when in HMD mode the goal is to get the hands to be displayed relative to the head position. This allows the hand position data captured by the Leap rendered in the virtual environment to parallel the real world, adding realism and an environment that can be intuitively interacted with. In order to draw the hands relative to the head, coordinates need to be transferred from Leap coordinates to Oculus coordinates, shown in Figure 6, then finally to Mockup world coordinates.

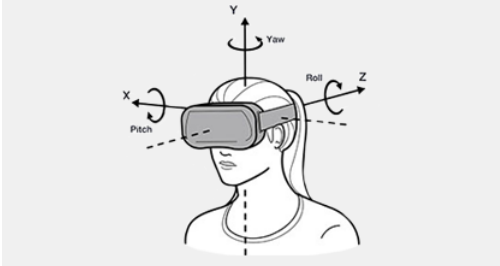

FIGURE 12: OCULUS RIFT COORDINATE SYSTEM [48]

DOI: 10.2352/ISSN.2470-1173.2016.4.ERVR-417

A collection of matrix transforms is conducted to change Leap to Oculus coordinates then to Mockup world coordinates. As with the distortion shader portion of the work, the implementation of the hand tracking was aided by the low level access provided by the Leap Motion API. Being able to access the raw coordinate data was helpful. Allowing the researchers to manipulate the data as needed. As Leap is continuing to adjust and make the API more user friendly it is imperative that they still allow for access to the raw data for groups that seek to use it for more commercial purposes. The first step in this process is to grab the world coordinates from the Oculus head tracking API. This is preformed though a function call to the API that returns the translation and rotation matrices. Multiplied together, as shown in Equation 1, returns the Oculus position and rotation in the physical world.

$$M_{Oculus/World} = M_{OculusTrans} M_{OculusRot} \quad \text{EQUATION 1}$$

The second step is to figure out where in relation to the Oculus the Leap is mounted, so the correction can be calculated. Equation 2 shows the matrix that describes this correction. The t in the matrix represents the offset from the user's eye position to the Leap controller. This depends on the mounting position of the Leap on the Oculus. For this project, the researchers mounted the Leap centered on the Oculus. Ensuring that there was no offset in the x or y position. The Leap coordinate system is also rotated 90 degrees to match the Oculus world coordinate system using the equation below.

$$M_{Leap/Oculus} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix} \quad \text{EQUATION 2}$$

The only translation needed for the Leap coordinates was an offset of -0.08 meters in the z direction, seen in Equation 3. This is due to the Leap being 0.08 meters in front of the user's eye position.

$$t_z = -0.08\,[m] \quad \text{EQUATION 3}$$

Multiplying Equation 1 and Equation 2 as seen in Equation 4 results in a matrix that will adjust the Leap motion coordinates into in the same coordinate system as the Oculus. This conversion matrix will be updated continuously while running the program. As the Oculus headset moves about the origin, the origin is the Oculus tracking camera, the view shown in Mockup adjusts to render the correct view perspective in the Oculus headset. This conversion matrix also allows for mapping the points captured by the Leap to be adjusted into the same coordinate system as the Oculus and then mapped into the virtual Mockup environment.

$$M_{Leap/World} = M_{Oculus/world} M_{Leap/Oculus} \quad \text{EQUATION 4}$$

Converting the points captured by the Leap controller into points displayed in the virtual Mockup environment is accomplished using Equation 5. In this equation the local Leap coordinates are multiplied by a matrix containing the conversion from millimeters to meters, then the Leap to Oculus world coordinates and finally then by a scaling factor specific to the Mockup scene. The Mockup scaling factor will depend on both the size of the scene and the document units.

$$\begin{bmatrix} P_{Mockup} \\ 1 \end{bmatrix} = M_{MockupScaling} M_{Leap/world} \begin{bmatrix} 0.001 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_{Leap} \\ 1 \end{bmatrix} \quad \text{EQUATION 5}$$

Implementing the hand tracking piece of the project was not exceedingly difficult. The most difficult part of the task was finding the conversion factors for Mockup. Scaling the hands to the proper size depending on the model viewed in the virtual environment was important to create a realistic immersive scene. Aspects had to be considered like hand size compared to the model, reach of the hands and adjusting the hand size based on the model units proved challenging. For the prototype the hand size was determined based on the scaling of the model and the reach of the hands was a one to one scale with the reach of the user in the physical space.

In the future a type of gain based hand placement could be explored to provide the user with more interaction space for larger models. Increasing their ability to interact outside of their immediate physical reach. The hand scaling for the model based on document units was difficult. The way the code was structured it was hard to get the document unit information into the section of code controlling the Oculus. The problem of dealing with legacy code structures to accomplish device integration is most likely going to be a real problem moving forward. Integrating these new devices with the existing codebase that were not designed to support these interactions and display mediums, will take some dedication and work on the part of developers.

Results

The final prototype allows users to interact with models via the Leap Motion in Mockup. The models are viewed using the Oculus Rift hooked into the Mockup graphics pipeline. The Oculus display device works well. Especially, after the distortion and chromatic aberration corrections were applied to the images, viewing the model with the Oculus is a smooth and seamless process. The user is able to move their head and receive a full 360 degree view the the environment thanks to head tracking. The manipulation using the Leap, however, proved to be more difficult. The results of both user interface testing and the system performance measurements are discussed below.

User Interface Testing

Overall, tracking with the Leap was inaccurate and jittery. The quality of tracking was very sensitive to lighting conditions. Often the device would lose tracking when operating under standard office fluorescent lighting. Also the HMD tracking mode on the Leap, which is meant to improve tracking of the device when mounted to a head mounted display, was lackluster. This first beta released by Leap was not accurate and often resulted in jittery tracking. When demoing the prototype, the regular desktop tracking mode for the Leap was preferable to the the HMD mode. In addition, the tracking range of the Leap was limited and unreliable. This intermittent tracking often would cause the hands in the virtual world to jump and twist. Degrading the user experience.

For this application the Leap controller interface was challenging to design since during normal operation the hands are generally pointing way from the Leap. This results in the controller not having line of sight to the fingers, creating problems for tracking and gesture recognition. When the camera cannot see the fingers it

DOI: 10.2352/ISSN.2470-1173.2016.4.ERVR-417

causes the hand model in the virtual world to not match up with user movements. While this mismatch was a problem in the prototype, the authors believe this issue can be mitigated in the future via code to keep the fingers as is, in relation to the palm when tracking is lost. However, this still leaves the problem with gestures. In the HMD tracking mode, the gestures were unreliable. The Leap was not able to recognize gestures with any degree of regularity. This made the gesture interface as designed unusable. The Leap couldn't even recognize the the scaling gesture when the hands were pointed so all the fingers were visible to the camera, the best case scenario. This failure to work even under prefect conditions does not endear itself well to the use of the Leap controller for any type of robust fluid gesture interface when mounted on the Oculus.

After initial testing with the prototype, it is evident that the interface needs do not match with the current capabilities of the Leap when mounted on the Oculus. With the unreliable tracking and line of sight problems, the Leap does not produce a quality user experience. Moving forward, Leap needs to continue working on tracking and gesture recognition in HMD mode to make it robust enough for production code.

System Performance

To understand how the addition of these commodity devices impacted the performance of Mockup, the authors measured the frame rates of the system under different loading conditions. Looking at the frame rates can provide valuable information about how well the low-cost commodity devices integrate together with Mockup, irrespective of usability. The Oculus display device with Leap attached was tested with varying model sizes and rendering pipelines to see if the prototype experience in Mockup meets a threshold of 30 frames per second for real time rendering quality [52]. The authors selected the rendering threshold because, rendering below 30 fps can start to degrade the user experience. In a commercial product such as Mockup, the quality of the experience needs to be high to maintain customer satisfaction. The frame rate testing is performed on a system with 3.6 GHZ Intel Xeon CPU, 16G of RAM, and nVidia Quadro K4200 graphics card. The frame rate is measured at the end of the rendering routine by recording the difference between two sequence frames, and taking the average time in seconds over 50 frames. Table 1 below contains the results of this testing.

The first testing mode, standard desktop interface, was tested to provide a baseline performance. The desktop rendering pipeline in Mockup is designed to provide real time model rendering for large models at high frame rates, in order to produce a quality user experience. This testing used a standard desktop monitor to display images rather than the Oculus. Where as, the second testing mode used the standard rendering pipeline found in the desktop test, but added the Oculus and the Leap as display and input devices instead of a desktop monitor. The third mode used both the Oculus and the Leap, but the rendering pipeline incorporating the chromatic aberration and distortion shader correction for the Oculus. Based on the architecture of Mockup, the authors hypothesized that hooking up devices like the Oculus and Leap would somewhat decrease the frame rate. However, the authors were unsure how much extra overhead these devices would add. The results of the testing and their implications are discussed below.

TABLE 1: AVERAGE FRAMES PER SECOND TESTING RESULTS

Testing / Model Size	Number of Polygons	Desktop [fps]	Standard Rendering Pipeline [fps]	Modified Rendering Pipeline [fps]
Small	1,179,883	60.24	31.55	30.03
Medium	8,337,490	59.88	30.03	28.99
Large	18,137,354	59.17	29.94	28.57

The results show that for the standard desktop interface the frame rates are well above the 30 fps benchmark. These results for desktop demonstrate Mockup's custom rendering routines ability to handle large numbers of polygons while maintaining real time interactive frame rates. Within the desktop condition, the frame rate drops only 1.07 fps for around an 18 times increase in polygons. The custom rendering pipeline unique to Mockup, developed to support large scale model visualization, was one of the reasons why Mockup needs to be able to natively integrate these low-cost interaction devices. Interacting directly with the devices, at a low level, allows Mockup to maintain its real time rendering capabilities by employing code optimizations that incorporate these new devices. These optimizations are integral to providing a degree of control over the end product. For commercial software, specifically, this ability to refine the end product and create a high quality user experience is a must. As these new devices are released, companies will want to integrate them into existing products to maintain their specific branding, meaning low level API access to data is an important feature to aid device adoption by commercial companies.

The second testing condition, no distortion shaders, tests the performance of the prototype with both the Leap and the Oculus devices hooked up to Mockup. This testing condition looks at the sensor integration into Mockup, independent of the rendering pipeline corrections for chromatic aberration and distortion described above. Results from this test show that the frame per second performance drops to around half of the desktop, to 31.55 fps, for a small model. When the distortion shader and chromatic aberration correction are applied the frame rate again drops, but only by 1.52 fps. This small drop off in frame rate between the standard rendering pipeline and the modified rendering pipeline indicate that the bulk of the performance degradation is coming from the sensor manager portion that handles the Leap and the Oculus positioning.

Based on the results of the testing, in order to improve the frame rate measurements of the prototype system, the next step is to identify where in the sensor manager the bottle neck occurs. Based on the amount of coordinate data from the Oculus head positioning and the Leap hand tracking this portion would be a likely starting point for code optimization. Finding a way to more efficiently handle and pass this data may improve frame rate speeds in future work.

Conclusion

The end result of the project was a functioning prototype demonstrating the integration of two, low-cost immersion devices into a commercial engineering visualization and analysis platform. The two devices, Leap and Oculus, were used to test the feasibility of using low-cost commodity hardware in production code. From testing the interface, the authors found that the Leap controller was not accurate or reliable enough for use in the prototype system. The occlusion of the fingers by the hand created jittering and loss of tracking. Also, during testing the gestures recognition was found to be unreliable. Moving forward the Leap could work as an interface

DOI: 10.2352/ISSN.2470-1173.2016.4.ERVR-417

option if it was mounted on the desktop, however, the usability would be impacted since interaction would be limited to the desktop tracking volume. Another possibly for an interface, would be using a more expensive physical controller such as a data glove.

From the performance testing, the authors found that the sensor manager is the main bottle neck for frame rate performance. While frame rates did not change for different model sizes they did change between the desktop and Oculus/Leap tests. For a small model, the frame rate drops in around half when hooking up the Oculus/Leap prototype to Mockup. This indicates that for future work, to ensure a quality customer experience, the code dealing with integrating the sensors needs to be optimized or parallelized, where as the rendering corrections for chromatic aberration and the distortion shader do not significantly impact performance. One likely way to improve the performance of the prototype, would be to add another thread dedicated to handling the positioning information coming from the Leap and the Oculus. Splitting this information processing off into a separate thread would likely eliminate some of the bottlenecks contributing to the drop in frame rates.

Overall, the work demonstrates the feasibility of using these low-cost commodity devices in a commercial engineering visualization and analysis package. While the Leap controller may not be well suited for an interface while mounted on an HMD, the Oculus itself shows promise as a new medium for visualizing models in a commercial engineering visualization and analysis package.

References

[1] Berta, J., 1999, "Integrating VR and CAD," Computer Graphics and Applications, IEEE , 9(5), pp.14-19.

[2] Schuemie, M. J., V an Der Straaten, P ., Krijn, M., V an Der Mast, C. A., 2001, “Research on presence in virtual reality: A survey,” CyberPsychology & Behavior, 4(2), 183-201.

[3] Gruchalla, K., 2004, "Immersive W ell-path Editing: Investigating the Added Value of Immersion," Virtual Reality - Proceedings, pp.157-164.

[4] Swink, M.L., Sandvig, J.C., Mabert, V .A., 1996, "Customizing concurrent engineering processes: Five case studies," Journal of Product Innovation Management, 13(3), pp. 229-244.

[5] Shen, W., Hao, Q., Li, W., 2008, "Computer Supported Collaborative Design: Retrospective and Perspective," Computers in Industry, 59(9), pp. 855- 862.

[6] Belkadi, F., Bonjour, E., Camargo, M., Troussier, N., & Eynard, B., 2013, “A situation model to support awareness in collaborative design,” International Journal of Human-Computer Studies, 71(1), 110-129.

[7] Elrod, S., Bruce, R., Gold, R., Goldberg, D., Halasz, F., Janssen, W., Lee, D., McCall, K., Pedersen, E., Pier, K., Tang, J., Welch, B., 1992, "Liveboard: a large interactive display supporting group meetings, presentations, and remote collaboration," Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '92), pp. 599-607.

[8] Bouchlaghem, D., Shang, H., Whyte, J., Ganah, A., 2005, "Visualisation in Architecture, Engineering and Construction (AEC), Automation in Construction,” 14(3), pp. 287-295.

[9] Zhong, H., W achs, J. P ., & Nof, S. Y ., 2014, “Telerobot-enabled HUB-CI model for collaborative lifecycle management of design and prototyping,” Computers in Industry.

[10] Stelzer, R., Steindecker, E., Arndt, S., Steger, W., 2014, "Expanding VRPN To Tasks in Virtual Engineering” Proceedings of the ASME 2014 Design Engineering Technical Conferences and Computers and Information in Engineering Conference, pp. 1-8.

[11] Shiratuddin, M.F.; Wong, K.W., 2011, "Non-contact Multi-hand Gestures Interaction Techniques for Architectural Design in a Virtual Environment," Information Technology and Multimedia (ICIM), pp.1-6.

[12] Witmer, B. G., Singer, M. J., 1998, “Measuring presence in virtual environments: A presence questionnaire,” Presence: Teleoperators and virtual environments, 7(3), 225-240.

[13] Mujber, T.S., Szecsi, T., Hashmi, M.S.J., 2004, "Virtual Reality Applications in Manufacturing Process Simulation," Journal of Materials Processing Technology, Vol. 155–156, pp. 1834-1838.

[14] Bowman, D.A., and McMahan, R.P., 2007, "Virtual Reality: How Much Immersion Is Enough?," Computer , 40(7), pp.36-43.

[15] McMahan, R.P., Gorton, D., Gresock, J., McConnell, W., Bowman, D.A., 2006, "Separating the Effects of Level of Immersion and 3D Interaction Techniques," Proceedings of the ACM symposium on Virtual reality software and technology (VRST ’06), pp.108-111.

[16] Kalivarapu, V., MacAllister, A., Hoover, M., Sridhar, S., Schlueter, J., Civitate, A., Thompkins, P., Smith, J., Hoyle, J., Oliver, J., Winer, E., Chernoff, G., “Game-day Football Visualization Experience on Dissimilar Virtual Reality Platforms”, Accepted for publication at the IS&T/SPIE Electronic Imaging, San Francisco, CA, February 2015.

[17] Barfield, W., Hendrix, C., Bystrom, K., 1997, “Visualizing the Structure of Virtual Objects Using Head Tracked Stereoscopic Displays,” Virtual Reality Annual International Symposium, pp. 114-120.

[18] Ware, C., and Mitchell, P., “Reevaluating stereo and motion cues for visualizing graphs in three dimensions,” Proc. of the 2nd Symposium on Applied Perception in Graphics and Visualization, 1-8 (2005).

[19] Kosmadoudi, Z., Lim, T., Ritchie. J., Louchart, S., Liu, Y., Sung, R., 2013, "Engineering design using game-enhanced CAD: The potential to augment the user experience with game elements," Computer- Aided Design, 45(3), pp. 777-795.

[20] Ju, W., Madsen, S., Fiene, J., Bolas, M., McDowall, I., Faste, R., 2003, “Interaction Devices for Hands-On Desktop Design,” SPIE, pp. 585-595.

[21] Francese, R., Passero, I., Tortora, G., 2012, "Wiimote and Kinect: Gestural User Interfaces Add a Natural Third Dimension to HCI," Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI '12), pp.116-123.

[22] Shiratuddin, M.F.; Wong, K.W., 2011, "Non-contact Multi-hand Gestures Interaction Techniques for Architectural Design in a Virtual Environment," Information Technology and Multimedia (ICIM), pp.1-6.

[23] Purschke, F., Schulze, M., Zimmermann, P., 1998, "Virtual Reality-new Methods for Improving and Accelerating the Development Process in Vehicle Styling and Design," Computer Graphics International, pp.789-797.

[24] Fiorentino, M., Uva, A.E., Monno, G., Radkowski, R., 2012, “Augmented Technical Drawings A Novel Technique for Natural Interactive Visualization of Computer-Aided Design Models,” Journal of Computing and Information Science in Engineering, Vol. 12, pp. 1-8.

DOI: 10.2352/ISSN.2470-1173.2016.4.ERVR-417

[25] Fiorentino, M., Radkowski, R., Stritzke, C., Uva, A.E., Monno, G., 2013, “Design Review of CAD Assemblies Using Bimanual Natural Interface,” International Journal on Interactive Design and Manufacturing, 7(4), pp. 249-260.

[26] Nanjundaswamy, V.G., Kulkarni, A., Chen, Z., Jaiswal, P., Shankar, S.S., Verma, A., Rai, R., 2013, “Intutive 3D computer-Aided Design (CAD) System With Multimodal Interfaces,” Proceedings of the ASME 2013 International Design Engineering Technical Conferences and Computers in Engineering Conference, pp. 1-11.

[27] Park, H., Park, J., Kim, M., 2012, “3D Gesture-Based View Manipulator for Large Scale Entity Model Review,” Asia Simulation Conference, pp. 524-533.

[28] Sabir, K., Stolte, C., Tabor, B., O’Donoghue, S., 2013, “The Molecular Control Toolkit: Controlling 3D Molecular Graphics via Gesture and Voice,” Biological Data Visualization, pp. 49-56.

[29] Gallo, L., Placitelli, A.P., Ciampi, M., 2011, "Controller-Free Exploration of Medical Image Data: Experiencing the Kinect," Computer-Based Medical Systems (CBMS), 2011 24th International Symposium on, pp. 27-30.

[30] Wright, M., Lin, C., O’Neill, E., Cosker, D., Johnson, P., 2011, “3D Gesture Recognition: An Evaluation of User and System Performance,” International Conference on Pervasive Computing, pp. 294-313.

[31] Lee, S., Chae, J., Kim, H., Lim, Y., Lee, K., 2013, “Towards a More Natural Digital Content Manipulation via User Freehand Gestural Interaction in a Living Room,” Proceedings on the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp ’13), pp. 617-626.

[32] Fikkert, W., van der Vet, P., van der Veer, G., Nijholt, A., 2010, “Gesture for Large Display Control,” 8th International Gesture Workshop, pp. 245-256.

[33] Tumkor, S., Esche, S. K., Chassapis, C., 2013, “Hand Gestures in CAD Systems,” Proceedings of the ASME 2013 International Mechanical Engineering Congress and Exposition, pp. 1-9.

[34] Song, J., Cho, S., Baek, S., Lee, K., Bang, H., 2014, “GaFinC: Gaze and Finger Control Interface for 3D Model Manipulation in CAD Application,” Computer-Aided Design, Vol. 46, pp. 239-245.

[35] Araullo, J., & Potter, L. E., 2014, “Experiences using emerging technology,” In Proceedings of the 26th Australian Computer-Human Interaction Conference on Designing Futures: the Future of Design (pp. 523-526). ACM.

[36] Donalek, C., Djorgovski, S. G., Cioc, A., Wang, A., Zhang, J., Lawler, E., Yeh, S., Mahabal, A., Graham, M., Drake, A., Davidoff, S., Norris, J. S., Longo, G., 2014, "Immersive and Collaborative Data Visualization Using Virtual Reality Platforms,” IEEE International Conference on Big Data.

[37] Lobos, P., Beimler, R., Lammers, M., Steinicke, F., 2014, "Touching the Cloud : Bimanual Annotation of Immersive Point Clouds," 2014 IEEE Symposium on pp. 191-192. IEEE.

[38] Plemmons, D., & Holz, D., 2014, "Creating next-gen 3D interactive apps with motion control and Unity3D," In ACM SIGGRAPH 2014 Studio, p. 24, ACM.

[39] Coelho, J. C., & Verbeek, F. J., 2014, "Pointing Task Evaluation of Leap Motion Controller in 3D Virtual Environment," In Creating the Difference, Proceedings of the ChiSparks 2014 Conference, The Hague, The Netherlands, pp. 78-85.

[40] Tomori, Z., Jan K., Peter K., Petr J., Mojmir S., Silvie B., Marian A., and Pavel Z., 2014, "Natural user interface as a supplement of the holographic Raman tweezers," In SPIE NanoScience+ Engineering, International Society for Optics and Photonics, pp. 91642P-91642P.

[41] Van Thanh, T., Kim, D., & Jeong, Y. S., 2015, "Real-Time Virtual Lego Brick Manipulation Based on Hand Gesture Recognition," In Advanced Multimedia and Ubiquitous Engineering, Springer Berlin Heidelberg, pp. 231-238.

[42] Karolczak, K., & Klepaczko, A. A stereoscopic viewer of the results of vessel segmentation in 3D magnetic resonance angiography images.

[43] Frederick Thompson, MS, 2014, “Evaluation of a commodity VR interaction device for gestural object manipulation in a three dimensional work environment.”

[44] Weichert, F., Bachmann, D., Rudak, B., & Fisseler, D. 2013, "Analysis of the accuracy and robustness of the leap motion controller," Sensors, 13(5), pp. 6380-6393.

[45] MacAllister, A., Winer, E., Yeh, T., Seal, D., Degenhardt, G., 2014, “A Natural User Interface for Immersive Design Review,” International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, ASME.

[46] Anastacia MacAllister, MS, 2014, “Natural user interfaces for interdisciplinary design review using the Microsoft Kinect.”

[47] Fikkert, W., van der Vet, P., van der Veer, G., Nijholt, A., 2010, “Gesture for Large Display Control,” 8 International Gesture Workshop, pp. 245-256.

[48] Leap Motion, July 2015, https://developer.leapmotion.com/documentation/cpp/devguide/Leap_Overview.html#gestures

[49] “Oculus Rift in Action”, 2013, “Understanding the Oculus Rift Distortion Shader,” http://rifty-business.blogspot.com/2013/08/understanding-oculus-rift-distortion.html, 06 July 2015.

[50] Khattak, S., Cowan, B., Chepurna, I., & Hogue, A. 2014, “A real-time reconstructed 3D environment augmented with virtual objects rendered with correct occlusion.” In Games Media Entertainment (GEM), pp. 1-8.

[51] Javidi, B., Okano, F., 2011, “Three-Dimensional Television, Video, and Display Technologies,” Springer.

[52] Meehan, M., Insko, B., Whitton, M., Brooks Jr., F.P., 2002, “Physiological Measures of Presence in Stressful Virtual Environments,” ACM Transactions on Graphics (TOG), pp. 645-652.