

Simulating the Experience of Home Environments

Kevin Ponto

University of Wisconsin – Madison
Wisconsin Institute for Discovery
Madison, Wisconsin 53715
Email: kbponto@wisc.edu

Ross Tredinnick

University of Wisconsin – Madison
Wisconsin Institute for Discovery
Madison, Wisconsin 53715
Email: rdtredinnick@wisc.edu

Gail Casper

University of Wisconsin – Madison
Wisconsin Institute for Discovery
Madison, Wisconsin 53715
Email: gcasper@wisc.edu

Abstract—Growing evidence indicates that transitioning patients are often unprepared for the self-management role they must assume when they return home. Over the past twenty five years, LiDAR scanning has emerged as a fascinating technology that allows for the rapid acquisition of three dimensional data of real world environments while new virtual reality (VR) technology allows users to experience simulated environments. However, combining these two technologies can be difficult as previous approaches to interactively rendering large point clouds have generally created a trade-off between interactivity and quality. For instance, many techniques used in commercially available software have utilized methods to sub-sample data during interaction, only showing a high-quality render when the viewpoint is kept static. Unfortunately, for displays in which viewpoints are rarely static, such as virtual reality systems, these methods are not useful. This paper presents a novel approach to the problem of quality-interactivity trade-off through a progressive feedback-driven rendering algorithm. This technique uses reprojections of past views to accelerate the reconstruction of the current view and can be used to extend existing point cloud viewing algorithms. The presented method is tested against previous methods, demonstrating marked improvements in both rendering quality and interactivity. This algorithm and rendering application could serve as a tool to enable virtual rehabilitation within 3D models of one’s own home from a remote location.

I. INTRODUCTION

Growing evidence indicates that patients in transition from institutional care settings to home are often unprepared for the self-management role they must assume when they return home [1]. Key among the challenges to transition is the lack of representation and consideration of the context of the home environment in discharge planning. Co-occurring challenges include uncertainty about self management due to conflicting or unclear discharge instructions [2]. The patient’s experience of returning home may be complicated further by anxieties, confusion, and distorted sensations. Discharge plans that are comprehensive and support self-management in the home context can promote safe transition to home and reduce readmissions. Creating the simulated experience of home environments poses new possibilities for the purposes of discharge planning and tele-rehabilitation. However, creating this simulated experience requires both the capture and display of actual physical environments.

Over the past twenty five years, LiDAR scanning has emerged as a fascinating technology that allows for the rapid acquisition of three dimensional data of real world environments [3]. LiDAR scanning results in data known as a “point

cloud”, which is normally represented as an unstructured list of 3D positions with optional color and normal data per point. While LiDAR scanning has been utilized in such diverse applications as archaeology, geology, and cultural heritage [4]–[6], the capture of the home environment has seen less research interest [7]–[9].

While there have been approaches to convert point cloud data into standard 3D models, these approaches have shown rather limited results [10]. While features such as walls and ceilings can be easily extracted, other features, such as clutter on a floor, do not map well into standard model formats. In this regard, in order to fully visualize the entropy of the home environment it is important to keep the data as points.

In recent years, virtual reality (VR) technology has been experiencing a resurgence in popularity due to the availability of lower cost, consumer friendly, VR display hardware. Despite this, little work exists which combines VR display hardware with rendering of LiDAR point clouds. Kreylos et al., developed a LiDAR viewing application that allows for immersive viewing and the editing of large point clouds in a CAVE VR display environment [11], but their system presents results on a relatively low (1280x1024) resolution display system [12]. Extending their work to higher resolutions has resulted in relatively low frame rates with [7] showing fifteen frames per second when displaying on a tiled display wall of 41 megapixels. Two significant problems exist when attempting to render large point clouds using VR display technology that are likely culprits for the lack of LiDAR rendering applications for VR display technology:

- 1) **Physical and Graphics Memory Limits:** As many scanners are capable of rapid acquisition, it is possible to create data sets that are billions of points in size within just a few hours [13]. The resulting point cloud is often larger than a machine’s physical RAM capacity and much larger than the amount of GPU graphic memory.
- 2) **Primitive Count:** Due to the architecture of the GPU, the number of points drawn per frame are inversely proportional to the frame rate [14]. This problem is further complicated when multiple views need to be rendered, such as for stereo Virtual Reality (VR) systems or with high-resolution displays.

While newer hardware may help to alleviate the issues of memory limits, if done naively, this only comes at the

expense of an increased primitive count. Therefore, it is critical to prioritize the rendering of points which contribute to the generated image. One solution for non-immersive applications relies on creating two different rendering techniques – one for interaction and one for progressively creating a correctly sampled image.

During interaction and viewpoint manipulation only a subset of the points are rendered which may result in an under-sampled display but high frame rate. Once the user stops changing the viewpoint the rendering algorithm switches to a progressive renderer that fills in the missing information. Since the user is not interacting with the scene, this data can be displayed on top of existing information (i.e., the buffers do not need to be cleared) and over-sampling is not problematic. Unfortunately, this approach is inaccessible for rendering techniques that require continuous activity, such as for head-tracked VR displays where the viewpoint changes every frame. High frame rates and low system latency are required in immersive VR environments to help prevent the negative effects of simulator sickness on users [15]. This paper describes the development of a VR software application as shown in Figure 1 for simulating home environments using LiDAR point clouds and VR display technology at highly interactive frame rates.

We envision a potential application of the VR technology to improve rehabilitation efforts with patients scheduled for joint arthroplasty procedures. Leveraging LiDAR scans of key areas of the home, and using the presented algorithm for displaying LiDAR scans within a VR CAVE would enable 1) pre-operative multi-disciplinary home assessment and anticipatory discharge planning with the patient; 2) learning and rehearsal of key health behaviors related to mobility, strength-building and self-care activities in a familiar engagement; and 3) requisite modification of the home environment to accommodate any equipment. Inclusion of informal caregivers and multiple disciplines is facilitated by the use of this simulated environment. We have explored partnerships with clinicians interested in demonstrating the capacity and benefits of simulating the home environment.

The presented application focuses on an improved solution for the *Primitive* Count problem explained above by introducing a new algorithm that takes advantage of the temporal coherency of point clouds between frames.

- 1) **Interactive Progressive Rendering:** While the view in a VR system changes continuously between frames, the change is small in nature, and thus large parts of a previously rendered frame can be re-used to provide an initial best guess for the current frame’s render output. As additional points are added to the feedback loop, view quality is progressively improved over time.
- 2) **User Definable Interactivity Independent of Quality:** In traditional point cloud rendering techniques, higher quality rendering output comes at the cost of interactivity. In the presented method, interactivity remains constant with the trade-off simply being a slightly longer time to converge to a high quality rendering output.



Fig. 1: An immersed individual experiencing a home environment at 75 frames per second of a 1.9 billion-point data set.

It is also worth noting that the presented algorithm can be adapted to prior point cloud rendering approaches. As shown in section VI, this algorithm is able to improve quality without degrading performance.

The rest of this paper is structured as follows: Section II introduces work in out-of-core point-based rendering and related graphics algorithms, Section III introduces the algorithm in detail, Section IV describes details of the algorithm’s implementation, and Sections V and VII provides an evaluation of the algorithm and discuss performance and quality metrics.

II. RELATED WORK

This section considers some of the relevant point-based techniques in relation to the presented application, namely in the area of out-of-core rendering and VR displays. There are more general point-based rendering techniques, and for an in-depth discussion of point-based graphics, the reader can refer to [16].

QSplat stands as the earliest out-of-core point cloud renderer, and it introduced several concepts, such as ideal splat kernels and out-of-core file layout for successfully rendering high-quality point cloud models [17]. This work compares point rendering results to traditional polygon results, showing higher quality images in a shorter amount of rendering time for point-based rendering.

Sequential point trees extended the hierarchical data structure of QSplat for sequential processing on the GPU, but their implementation does not support out-of-core processing [18]. Layered point clouds [19] extends sequential point trees by incorporating out-of-core loading and network streaming whereas Botsch et al. [20] seek to render high quality point cloud results of non out-of-core scenes. Instant points supports out-of-core processing and extended sequential point trees by improving preprocessing times and reducing the amount of data needed to be sent to the GPU in exchange for reduced rendering quality [21]. [22] introduced a multi-way kd-tree data structure to merge and cluster data for optimal vertex buffer object size creation. The presented work simplifies the concept of optimizing vertex buffer object sizes further by only needing a single vertex buffer object.

[23] visualizes massive point clouds of airborne LiDAR scans using an octree. [24] extends this work to support point clouds up to one billion points in size using an octree. [25], [26] developed the ability to edit massive out-of-core point clouds. Their applications support deleting points, selecting points, and inserting new points into existing clouds. All work ran on desktop machines and did not support stereo viewing or VR display systems. More recently, [27] extended techniques for visualization of cultural-heritage point clouds to mobile devices. The presented technique is concerned with improving frame rates for high-resolution and virtual reality displays, rather than mobile devices.

The presented work resembles approaches introduced by [28] and [29]. The former introduced the idea of ping-ponging point data back and forth between two textures bound to frame buffer objects. The latter extended this to calculating k-nearest neighbors on the GPU, thus leading to normal vector calculation on the GPU for points. Unlike [28], the presented application does not use an image pyramid technique and can achieve high-quality results at interactive rates without it. The work in [29] is concerned with achieving high-quality rendering results, but not at interactive rates. Due largely to the k-nearest neighbor calculation, their application gradually converges to high quality rendering. The proposed application is less concerned with normal vector estimation and instead focuses on high-speed frame rates for interactive display on VR hardware; however, extending this work to compute normal vectors at run time would serve as a future direction of this application with the challenge of maintaining interactive frame rates.

The presented application also builds upon work for image reprojection originally introduced in [30]. Their work maintains interactivity during camera movement by progressively refining the resultant image. More recently [31] extended these early techniques to maintain high interactivity for VR display systems when rendering meshes or point clouds and shows results for both multi-GPU and single-GPU frameworks. Unlike their work, and instead of reprojecting additionally rendered frames from adjacent locations to the main viewport, this presented work uses a single GPU to reproject immediately previous frames, thus reducing the overall amount of points rendered in a single frame.

III. METHOD

The maximum number of points that can be rendered per frame without overflow (which is the desired case) is equal to the number of pixels on screen with each point sampled by a unique pixel; however, such an ideal case rarely exists. The three-dimensional projection of point clouds causes multiple points to be projected onto the same pixel thus resulting in overflow. Because it is impossible to know a-priori which points will be visible and which will not, z-buffer depth testing, occlusion culling and view frustum culling are usually employed at different stages of the rendering pipeline; however, this still requires the submission of many points to the graphics card which will not contribute to the final rendered image.

The presented method works on the assumption that while the viewpoint changes between frames, it does not do so rapidly or erratically. The overall majority of points will be therefore visible in two consecutive frames. As such, the previous frame can be reprojected using the current frame’s camera information using a feedback loop. In order to fill the image in, additional points are added into the feedback loop on a per-frame basis.

In this regard, the number of points rendered per frame can be described as: $\mathbf{P} = \mathbf{V}(\mathbf{R} + \mathbf{S})$ with V being the number of viewpoints (for stereo rendering, this would be two), R being the number of points retained in the display system, and S being the number of points being streamed in a single frame. While V and R are usually predefined and static, S can be modified dynamically providing a means of providing a stabilized frame rate.

We note that with little movement between frames, this algorithm achieves a high reuse rate of points, and therefore, the number of points that need to be reprojected is low. Furthermore, the number of reprojected points is known and constant in each frame and allows a good render time estimate due to their fixed rendering cost. The number of additional points to be drawn depends on the total frame budget and this reprojection cost. The more additional points to be drawn, the quicker the image converges and the higher the image quality is at a lower refresh rate. The sections below detail how the algorithm is constructed and implemented.

A. Algorithm

The developed algorithm enables high resolution point cloud rendering at interactive frame rates using VR display hardware. Two render target frame buffer objects are employed in a ping-pong configuration. The render loop consists of four distinct parts: 1) visibility determination, 2) reprojection, 3) drawing of additional points, and 4) display of the result. The first part traverses the octree and determines the visible nodes using view frustum culling technique. The visible nodes are then sorted front-to-back based on euclidian distance to the current camera. The point density estimate function determines how many points are potentially visible to the user for each visible octant and determines how many points are loaded from file. Note that unlike previous object space approaches, this estimation is only for the number of points to read, rather than the number of points to draw for the octant.

During the reprojection step, a screen-filling point grid with one point per pixel is drawn. The previous frame’s render target buffer is bound as a read texture and the result is drawn to the second render target. For each point, the input vertex coordinates are used as texture coordinates into the color texture. If the value read is different from the cleared depth value of the depth attachment of the frame buffer object, the point is considered valid and the RGB colors are treated as world coordinates. Its clip space position is calculated from these coordinates and the fragment shader writes the point’s world position at the appropriate texel in the target render buffer.

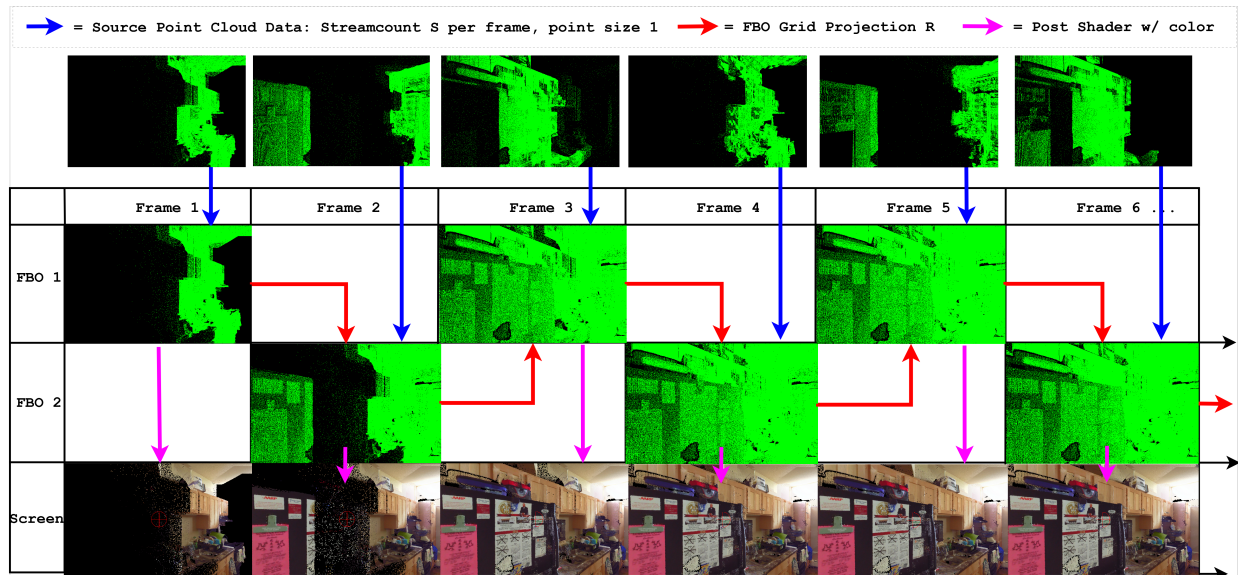


Fig. 2: Point feedback rendering for a single viewpoint over five consecutive frames. S in this case is one million points. The diagram shows that the scene converges in roughly five to six frames. FBO 1 and FBO 2 contain the world space position of the points as color, with actual point color applied during post processing to the screen.

Next, a predetermined number of new points (Streamcount S) are drawn by drawing the points of visible octants. These points are drawn in a sequential manner over multiple frames and transformed similarly as above. While some of these points will fail the depth test due to the already drawn points, those that pass will force the render target converging to the correct solution. Finally, the result is rendered to the screen and the render targets are swapped for the next pass.

IV. IMPLEMENTATION

Whereas section III gave an overview of the algorithm, this section provides specific implementation details.

Out-of-Core Foundation: Similar to prior approaches [23], [24], to efficiently determine which S points should be added to the scene, a multi-resolution octree data structure is created. Construction recursion continues until a minimum octant bounding size is reached. Each point is composed of four 32-bit floating point numbers: x, y, z , which are the point's world-space coordinates, and w , which stores a packed 32-bit RGBA color. Unlike prior approaches, and due to the fact that the presented algorithm progressively draws all points within each visible octant, a simple sampling approach such as randomly shuffling points within the octant suffices for achieving fast convergence within a scene. The resulting octree is written into a single binary file together with an index file to the start of each octant's points. Optionally, more intricate approaches for the subdivision structure such as that presented in [22] could be employed, or LOD point estimation methods presented in [21] could be used for determine how many and which points within a node are drawn per frame.

The octree on file is accessed using a two-tiered approach with two concurrently running threads. One thread continually calculates octant visibility and point estimation for all visible

nodes for the active viewpoint. In addition, it handles copying points from a user defined CPU memory cache (MC) to the GPU read buffer (RB) of user defined size (RBS). The GPU read buffer constitutes the total amount of GPU memory dedicated for point streaming. Once RBS worth of points have been copied to the GPU buffer, the visibility thread briefly waits to make sure that the main thread has drawn RBS amount of points. The main thread is continually uploading and drawing the user defined S amount of points and therefore will achieve drawing RBS worth of points in at worst RBS / S frames. This method ensures that no points are overwritten from the physical cache without streaming to the GPU. A second thread reads octant points from disk into the CPU memory cache. For each node, the application keeps track of the number of points that have already been copied to the GPU buffer for the node. For each octant, if the algorithm has reached the total number of original points within the octant, it resets the amount of read points to 0 and continues re-reading the octant's points as requested. This wrap-around sequential point read, combined with the point shuffle during octree creation results in a uniform sampling of a node's points and is a different approach from previous point cloud out-of-core rendering applications as *all* points within an octant are gradually displayed, not just the 'best' points for a given view which depend on a level of detail metric chosen.

Render Loop: Figure 2 shows an overview of the data flow during rendering. The render loop makes use of two 4-channel, 32-bit floating point frame buffer objects for rendering plus a source data set (the out-of-core point cloud). Each frame buffer stores the resulting 3D world position of a point in the color components of the render target as well as the point's color packed in the alpha channel. These frame buffer objects are used together to pass points that have been drawn back and

forth, thereby allowing the application to retain points that have been drawn in a previous frame and are still visible. In turn, the application is able to spread the number of additional points to be rendered across several frames.

A fixed number of sequential points (S) from the entire set of visible octants are uploaded each frame to the GPU and rendered to the bound frame buffer object. These points are always rendered at a point size of one. Next, points drawn into the previous frame’s frame buffer object are drawn into the currently bound frame buffer object. In each of these draws, the point’s original vertex position in world coordinates is written to the RGB channels and the point’s RGB color is packed into the alpha channel.

Filtering and Display: As a final step, the currently bound frame buffer object is drawn to the application window using a screen-filling quad. The color value is unpacked from the color buffer’s alpha to be used for a filtering pass. In order to account for small data shadows from the use of point sprites, a filtering approach is used to fill in empty regions. The approach fills shadowed regions by checking the surrounding region for valid depths and weighting color values based on a Gaussian fall-off function. These filtered values are simply used for the final display and are not put into the feedback algorithm.

V. EVALUATION

We compare the presented technique against rendering approaches used by a commercially available *Desktop* viewer¹ and a *Previous* immersive point cloud viewer [7]. The Desktop system subsamples the point cloud while interacting with the system before filling the data when motion is stopped. The previous immersive viewer only uses a level of detail scheme to determine which points to display, with the overall number of points drawn configurable via a command line argument.

These three systems were evaluated both for performance and for quality. Performance was measured by evaluating the number of frames drawn on a per-second basis. Quality metrics are common in the field of image and video compression and can be done either subjectively or objectively. As subjective measures may contain biases, we chose to use a common quality metric in the field of image compression: peak signal to noise ratio (PSNR) [32]. PSNR is generally used by comparing a compressed image to a ground-truth uncompressed image with typical values falling between 20-40 dB [33]; however, it is not our intention to quantify a PSNR value for our render as ‘good’ but simply to use the metric as a comparison between quality levels.

The difficulty of using this method for the purposes of evaluation is in the generation of the ground-truth image. As each application has its own nuances in rendering (e.g., point spread, background color), it was decided that each application would create its own ground-truth image individually. To create this, all points were rendered from a single viewpoint without sub-sampling or LOD methods. The renders were left

to converge for 15 minutes to ensure all visible points were contributing to the final image.

These ground-truth images were then used to test the quality of each application. Viewpoints matching the ground-truth were loaded in a “cold-start” fashion, with each rendered frame recorded to video. Additional metrics, such as frame rate were also monitored. The test was run to mirror two systems: a CAVE-like environment, using a mono rendering of a 1920x1920 resolution display at 120 Hz using an nVidia Quadro 5000 GPU with 2.5 GB of RAM with vertical sync off and a head mounted display (HMD-like) environment, in this case on a laptop using a mono rendering of a 1920x1080 resolution display at 60 Hz using an nVidia GeForce GTX 960M with vertical sync off. Results of these tests are discussed below. For the CAVE-like system, four different conditions were used consisting of the desktop viewer, the previous point cloud viewer [7], and two configurations of the presented method optimized to run at 20 and 45 frames per second, respectively. For the HMD-like system, three configurations of the presented system were used on top of the existing approaches, optimized to run at 30, 60, and 90 frames per second, respectively. The tests continued until the desktop system was run for 60 seconds for the HMD-like system and 120 seconds for the CAVE-like system due to the difference in resolutions of the two systems and consequent time to convergence.

A LiDAR-scanned data set of Frank Lloyd-Wright’s ‘Taliesin’ home and office was used as a test case for evaluation. This data set contains both exterior and interior spaces, intricate details and complex shading features. More than 1.9×10^9 points make up this data set; it requires about 30 Gigabyte of storage which is beyond the capabilities of current-generation graphics cards. The data set contains varying point density as 80 LiDAR scans make up the full model and to preserve high detail of certain objects in the space (a desire common for interior spaces), preprocessing of the data set does not include homogenization of point density.

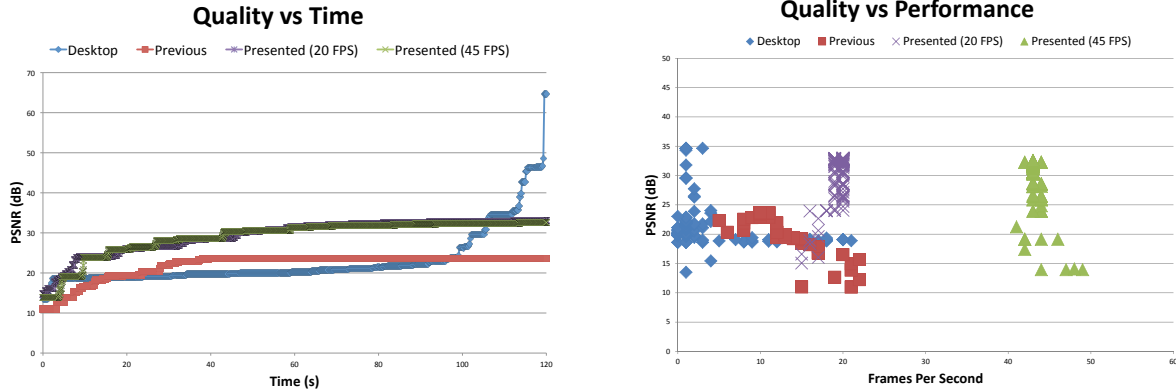
VI. RESULTS

Figure 3 demonstrates the results of these tests for the CAVE-like and HMD-like systems. The left images map the achieved quality (PSNR) over elapsed time of a non-moving view into the scene. The images show that the presented method converges to a higher quality much more quickly than existing approaches. While previous techniques plateau once they reach their memory limit, the presented method is able to continually improve in the quality of the rendered image throughout the trial period. The Desktop approach is able to eventually produce a higher quality image compared to the reference applications.

The right column shows achieved quality (PSNR) over a sustained frame rate as scatter plots. Not only is the presented method able to achieve high quality, the vertical columns highlight that a fixed refresh rate can be guaranteed with the presented method.

¹www.faro.com/en-us/products/faro-software/scene/

CAVE System



HMD System

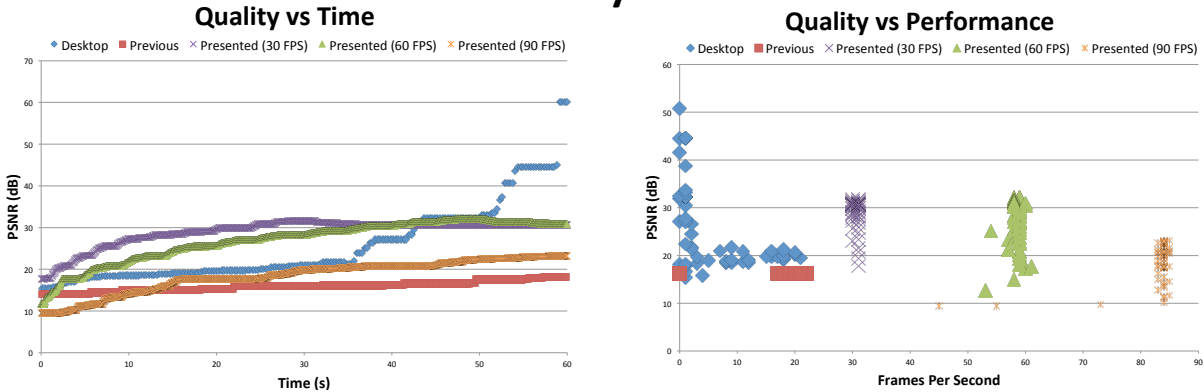


Fig. 3: The results of our evaluation method analyzing quality (larger PSNR values result in higher quality images). The left images show the convergence in image quality over time in a line graph with one sample per rendered frame, while the right images show scatter plots comparing the image quality to the frame rate of the render on a per second basis. Note that existing approaches create a trade-off between image quality (y-axis) and interactivity (x-axis) while the presented method is able to achieve both simultaneously (top-right).

The average frame-rate for a trial period was approximately 3 frames per second in the CAVE-like system and 5 frames per second in the HMD-like system when using the Desktop viewer. The previous approach for immersive environments is able to improve on this interactivity, with an average frame rate of approximately 10 frames per second in the CAVE-like system and 11 frames per second in the HMD-like system, but also produces a lower quality output in comparison. The presented method is able to achieve an average of 19, and 43 frames per second in the CAVE-like system and 30, 58, and 83 frames per second in the HMD-like system with significant improvements in quality compared to the previous approach, shown in the right images in Figure 3. We note that the presented method is able to achieve frame rates within 10% of the target frame rate in all cases. In all cases, the quality for the presented method is greater than that shown in the previous method.

Pilot Study of Usability: To test the improvements that this new technique might offer, we chose to examine the usability of the developed approach compared to the previous point cloud viewer [7] using the NASA-TLX [34]. NASA-

TLX is a multidimensional workload rating scale composed of six items (mental demands, temporal demands, effort, performance, physical demands, and frustration level), rated on a range from 0-100 (0-low, 100-high). Because of the small sample size and NASA's multidimensionality, interpretation of scores is based on viewing of trends as well as statistical analysis. Analysis was performed using a Welch Two Sample t-test.

We utilized a variety of strategies to recruit the convenience samples including deploying flyers, posting on community websites, word of mouth and referrals. For group A, 16 participants were recruited. The age range in group A spanned from young to middle aged adults. For group B, 20 participants were recruited with a mean age of 53 years (range 22-75). Eligibility criteria included the ability to read and speak English, able to stand for two separate 15 minute sessions, able to walk up and down 6 steps without assistance, and at least 21-years of age or older. Using a between subjects design, two groups were created, with group A being presented the previous point cloud viewer and group B being presented the developed method at 20 fps. Each participant was given

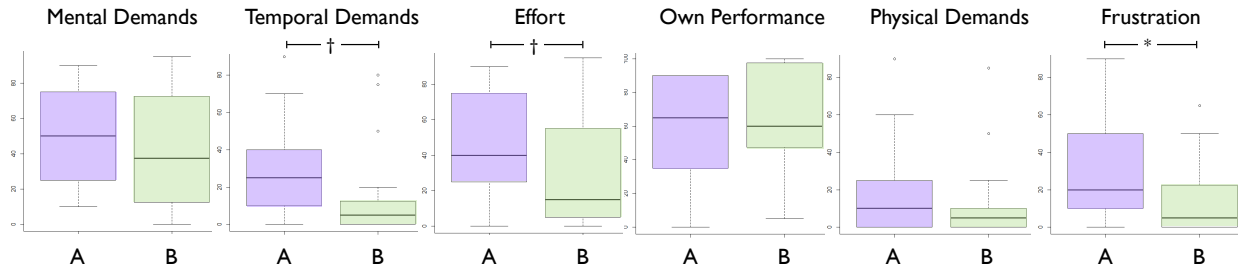


Fig. 4: Boxplots from the 36 subjects of the pilot study of usability showing the six subscales NASA-TLX between (A) the previous immersive pointcloud application and (B) the presented approach. Differences labeled (†) demonstrate $p < 0.10$ while ($*$) demonstrate $p < 0.05$.

15 minutes to traverse a virtual home inside of the CAVE environment. The virtual models are highly detailed real scale representations of home environments. The CAVE environment is a six sided rear-projected display system as shown in figure 5 that measures nine and one half feet cubed. The display system is also combined with a six degree of freedom tracking system for head tracking and tracking of an input device used for selection and navigation within the experiment. During this time, the participant was instructed to mark objects which would facilitate health information management using the tracked input device. All tasks, interfaces and environments were kept constant between groups A and B.

The results (shown in Figure 4) demonstrate that the presented method was significantly better in regards to user’s sense of frustration ($t(23) = 2.17; p = 0.040$), while the method showed significant trends towards improvements in the subscales for temporal demands ($t(32) = 1.99; p = 0.055$) and effort ($t(33) = 1.89; p = 0.070$). None of the other measures were found to be significant.

VII. DISCUSSION

As shown in Section VI, previous approaches created a trade-off between interactivity and quality, while the presented method is able to achieve both simultaneously. This section will discuss the presented approach.

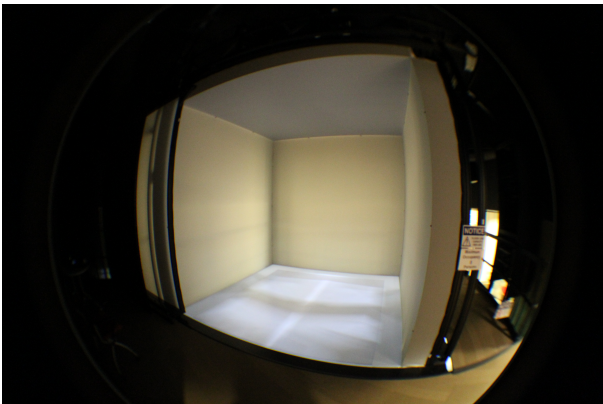


Fig. 5: The CAVE VR system for the usability study.

While the number of viewpoints V and the number of points in the feedback buffer R are fixed, the number of points streamed in S can be varied in order to alter performance dynamically; however, when S is extremely small, convergence times can increase dramatically. In this regard, setting S to 1 point per frame gives a theoretical (albeit impractical) maximum level of interactivity. In practice, the proposed approach has been shown to work well when the frame rate is set to $(180/R)$ or lower.

The pilot study of usability demonstrates improvement in three of the six NASA-TLX subscales. While the lack of difference in the subscales for physical demands and performance are to be expected and the lack of difference in mental demands were likely due to the simplicity of the task, the difference shown in the temporal demands is somewhat surprising. This may be due to the increased interactivity in the system making the participants feel lowered temporal demands. Due to flaws in the sampling strategy, the number of participants were uneven between groups A and B. It was determined that for the purposes of the paper to present all data and to account for this discrepancy in the data analysis.

In the presented approach, the feedback buffer is directly mapped to screen pixels; however, feedback buffers of other sizes could be used. For instance, larger-than-screen feedback buffers could be used to help predict future states, while smaller-than-screen frame buffers could be used to improve interactivity. These changes would modify the R component of above equations and will be explored in future iterations of this work.

VIII. CONCLUSION

This paper presents a novel solution to rendering large point cloud data sets inside of virtual environments. While previous approaches have created a trade-off between quality and interactivity, the presented solution is able to overcome these limitations through a progressive feedback driven rendering scheme. This scheme enables high-quality visuals to be created at interactive rates. The results show that the generated method provides a better overall experience.

Despite the excitement about VR-based simulation in healthcare many challenges remain. First, little is known about what aspects of the simulation contribute to learning and transfer of learning, critical to rehabilitation. The primary reason

why existing VR scenarios do not scale to the clinical needs of a broad range of patients awaiting discharge is that they are very specific to a given task in a given environment, usually a laboratory or clinical setting. In traditional, task-based simulation all participants receive the same visual cues; nuances and context are lost. The range of risks for readmission is substantial; it is impossible to build a simulation to specifically address each one. Our context-focused simulation approach, in which the simulated environment is the re-creation of the actual home, will allow the maximum tailoring to the individual patient. Future work will aim to better understand the experience of the end user, develop ways to measure the quality of the rendering system during motion, and explore alternative sampling strategies for accelerated convergence.

REFERENCES

- [1] A.-M. Hill, T. Hoffmann, C. Beer, S. McPhail, K. D. Hill, D. Oliver, S. G. Brauer, and T. P. Haines, "Falls after discharge from hospital: is there a gap between older peoples knowledge about falls prevention strategies and the research evidence?" *The Gerontologist*, vol. 51, no. 5, pp. 653–662, 2011.
- [2] E. A. Coleman, C. Parry, S. Chalmers, and S.-J. Min, "The care transitions intervention: results of a randomized controlled trial," *Archives of internal medicine*, vol. 166, no. 17, pp. 1822–1828, 2006.
- [3] E. S. Cameron, R. P. Szumski, and J. K. West, "Lidar scanning system," Apr. 9 1991, uS Patent 5,006,721.
- [4] J. L. Lerma, S. Navarro, M. Cabrelles, and V. Villaverde, "Terrestrial laser scanning and close range photogrammetry for 3d archaeological documentation: the upper palaeolithic cave of parpalló as a case study," *Journal of Archaeological Science*, vol. 37, no. 3, pp. 499–507, 2010.
- [5] N. Rosser, D. Petley, M. Lim, S. Dunning, and R. Allison, "Terrestrial laser scanning for monitoring the process of hard rock coastal cliff erosion," *Quarterly Journal of Engineering Geology and Hydrogeology*, vol. 38, no. 4, pp. 363–375, 2005.
- [6] H. Rüter, M. Chazan, R. Schroeder, R. Neeser, C. Held, S. J. Walker, A. Matmon, and L. K. Horwitz, "Laser scanning for conservation and research of african cultural heritage sites: the case study of wonderwerk cave, south africa," *Journal of Archaeological Science*, vol. 36, no. 9, pp. 1847–1856, 2009.
- [7] R. Tredinnick, M. Broecker, and K. Ponto, "Experiencing interior environments: New approaches for the immersive display of large-scale point cloud data," in *Virtual Reality (VR), 2015 IEEE*, March 2015, pp. 297–298.
- [8] P. F. Brennan, K. Ponto, G. Casper, R. Tredinnick, and M. Broecker, "Virtualizing living and working spaces: Proof of concept for a biomedical space-replication methodology," *Journal of biomedical informatics*, vol. 57, pp. 53–61, 2015.
- [9] R. Tredinnick, M. Broecker, and K. Ponto, "Progressive feedback point cloud rendering for virtual reality display," in *Virtual Reality (VR), 2016 IEEE*. IEEE, 2016, pp. 301–302.
- [10] N. A. Subramaniam and K. Ponto, "Hierarchical plane extraction (hpe): an efficient method for extraction of planes from large point cloud datasets," in *Proceedings of the 34th Annual Conference of the Association for Computer Aided Design in Architecture on, ACADIA*, 2014, pp. 627–636.
- [11] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti, "Surround-screen projection-based virtual reality: the design and implementation of the cave," in *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. ACM, 1993, pp. 135–142.
- [12] O. Kreylos, G. W. Bawden, and L. H. Kellogg, "Immersive visualization and analysis of lidar data," in *Advances in visual computing*. Springer, 2008, pp. 846–855.
- [13] J. Wang, J. Zhang, and Q. Xu, "Research on 3d laser scanning technology based on point cloud data acquisition," in *Audio, Language and Image Processing (ICALIP), 2014 International Conference on*, July 2014, pp. 631–634.
- [14] M. Pharr and R. Fernando, *Gpu gems 2: programming techniques for high-performance graphics and general-purpose computation*. Addison-Wesley Professional, 2005.
- [15] S. V. Cobb, S. Nichols, A. Ramsey, and J. R. Wilson, "Virtual reality-induced symptoms and effects (vrise)," *Presence*, vol. 8, no. 2, pp. 169–186, 1999.
- [16] M. Gross and H. Pfister, *Point-based graphics*. Morgan Kaufmann, 2011.
- [17] S. Rusinkiewicz and M. Levoy, "Qsplat: A multiresolution point rendering system for large meshes," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 343–352.
- [18] C. Dachsbacher, C. Vogelgsang, and M. Stamminger, "Sequential point trees," in *ACM Transactions on Graphics (TOG)*, vol. 22, no. 3. ACM, 2003, pp. 657–662.
- [19] E. Gobbetti and F. Marton, "Layered point clouds: a simple and efficient multiresolution structure for distributing and rendering gigantic point-sampled models," 2004.
- [20] M. Botsch, A. Hornung, M. Zwicker, and L. Kobbelt, "High-quality surface splatting on today's gpus," in *Point-Based Graphics, 2005. Eurographics/IEEE VGTC Symposium Proceedings*, June 2005, pp. 17–141.
- [21] M. Wimmer and C. Scheiblauer, "Instant points: Fast rendering of unprocessed point clouds," in *Proceedings of the 3rd Eurographics/IEEE VGTC conference on Point-Based Graphics*. Eurographics Association, 2006, pp. 129–137.
- [22] P. Goswami, F. Erol, R. Mukhi, R. Pajarola, and E. Gobbetti, "An efficient multi-resolution framework for high quality interactive rendering of massive point clouds using multi-way kd-trees," *The Visual Computer*, vol. 29, no. 1, pp. 69–83, 2012.
- [23] R. Richter and J. Döllner, "Out-of-core real-time visualization of massive 3d point clouds," in *Proceedings of the 7th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*. ACM, 2010, pp. 121–128.
- [24] J. Elseberg, D. Borrmann, and A. Nüchter, "One billion points in the cloud—an octree for efficient processing of 3d laser scans," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 76, pp. 76–88, 2013.
- [25] C. Scheiblauer and M. Wimmer, "Out-of-core selection and editing of huge point clouds," *Computers & Graphics*, vol. 35, no. 2, pp. 342 – 351, 2011, virtual Reality in Brazil Visual Computing in Biology and Medicine Semantic 3D media and content Cultural Heritage.
- [26] M. Wand, A. Berner, M. Bokeloh, P. Jenke, A. Fleck, M. Hoffmann, B. Maier, D. Staneker, A. Schilling, and H.-P. Seidel, "Processing and interactive editing of huge point clouds from 3d scanners," *Computers & Graphics*, vol. 32, no. 2, pp. 204 – 220, 2008.
- [27] M. B. Rodriguez, E. Gobbetti, F. Marton, R. Pintos, G. Pintore, and A. Tinti, "Interactive Exploration of Gigantic Point Clouds on Mobile Devices," in *VAST: International Symposium on Virtual Reality, Archaeology and Intelligent Cultural Heritage*, D. Arnold, J. Kaminski, F. Niccolucci, and A. Stork, Eds. The Eurographics Association, 2012.
- [28] R. Marroquim, M. Kraus, and P. R. Cavalcanti, "Efficient point-based rendering using image reconstruction," in *Proceedings Symposium on Point-Based Graphics*, 2007, pp. 101–108.
- [29] R. Preiner, S. Jeschke, and M. Wimmer, "Auto splats: Dynamic point cloud visualization on the gpu," in *Proceedings of Eurographics Symposium on Parallel Graphics and Visualization*, H. Childs and T. Kuhlen, Eds. Eurographics Association 2012, May 2012, pp. 139–148.
- [30] B. Walter, G. Drettakis, and S. Parker, *Rendering Techniques' 99: Proceedings of the Eurographics Workshop in Granada, Spain, June 21–23, 1999*. Vienna: Springer Vienna, 1999, ch. Interactive Rendering using the Render Cache, pp. 19–30.
- [31] F. Smit, R. van Liere, S. Beck, and B. Froehlich, "An image-warping architecture for vr: Low latency versus image quality," in *Virtual Reality Conference, 2009. VR 2009. IEEE*, March 2009, pp. 27–34.
- [32] A. Hore and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *Pattern Recognition (ICPR), 2010 20th International Conference on*, Aug 2010, pp. 2366–2369.
- [33] D. Salomon, *Data compression: the complete reference*. Springer Science & Business Media, 2004.
- [34] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," *Advances in psychology*, vol. 52, pp. 139–183, 1988.