Point Cloud Interaction and Manipulation in Virtual Reality

Daniel Garrido
Department of Informatics Engineering, University of Porto
Portugalup201403060@fe.up.pt

Rui Rodrigues
Department of Informatics Engineering, University of Porto
Portugalruirodrig@fe.up.pt

Augusto Sousa
Department of Informatics Engineering, University of Porto
Portugalaas@fe.up.pt

João Jacob
Department of Informatics Engineering, University of Porto
Portugaljoajac@fe.up.pt

Daniel Castro Silva
Department of Informatics Engineering, University of Porto
Portugaldcs@fe.up.pt

ABSTRACT

The use of virtual reality technologies for data visualization and analysis has been an emerging topic of research in the past years. However, one type of data has been left neglected, the point cloud. While some strides have been made in the visualization and analysis of point clouds in immersive environments, these have yet to be used for direct manipulation interactions. It is hypothesized that as with other types of data, bringing direct interactions and 3D visualization to point clouds may increase the ease of performing basic handling tasks. An immersive application for virtual reality HMDs was developed in Unity to help research this hypothesis. It is capable of parsing classified point cloud files with extracted objects and representing them in a virtual environment. Several editing tools were also developed, designed with the HMD controllers in mind. The end result allows the user to perform basic transformative tasks to the point cloud with an ease of use and intuitive feeling unmatched by the traditional desktop-based tools.

CCS CONCEPTS

• **Computing methodologies**; • **Computer graphics**; • **Shape modeling**; • **Human-centered computing**; • **Human computer interaction (HCI)**; • **Interaction paradigms**; • **Virtual reality**;

KEYWORDS

Virtual Reality, Interaction, Point Cloud, Point Cloud Classification, Object Extraction, Unity, Geometry Shader

ACM Reference Format:
Daniel Garrido, Rui Rodrigues, Augusto Sousa, João Jacob, and Daniel Castro Silva. 2021. Point Cloud Interaction and Manipulation in Virtual Reality. In *2021 5th International Conference on Artificial Intelligence and Virtual Reality (AIVR) (AIVR 2021), July 23–25, 2021, Kumamoto, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3480433.3480437

AIVR 2021, July 23–25, 2021, Kumamoto, Japan

ACM ISBN 978-1-4503-8414-8/21/07.
https://doi.org/10.1145/3480433.3480437

1 INTRODUCTION

The use of Virtual Reality (VR) technologies for data visualization and analysis is a research topic that dates to the beginning of the millennium, when Cave Automatic Virtual Environment (CAVE) systems started being used by many research laboratories [1, 10]. More recently, with the emergence of the new generation Head Mounted Display (HMD) VR technology, the research in this area has shifted to this type of systems [3, 8].

Alongside various other types of data, point clouds have been represented in immersive environments in this HMD focused era of virtual reality. At their most basic level. point clouds are "data structures used to represent a collection of multidimensional points" [14]. Most often, data points are represented with the usual 3 dimensions (XYZ) and additional information such as color, vectors (normal, velocity), scalar fields (temperature, height) and class [9, 14].

The state of the art on immersive point clouds shows that some strides have been accomplished in the interactive visualization department, while direct manipulation of the point dataset is curiously missing, aside from classification tasks. This mimics the current developments on Immersive Analytics in general, where most work being done is concerning interactive visualization and not manipulation.

This is the direction in which the present work attempts to aim. Immersive modeling of 3D objects in VR has been documented to have advantages over traditional modelling tools such as improve user creativity [4], input intuitiveness [4, 15] and visualization clarity [15].

The goal of this work is to research the usefulness of adapting typical point cloud manipulation tasks in immersive environments. This was achieved by implementing a framework for point cloud manipulation in VR using modern 6DoF HMDs and input controllers which serve as a base to implement various interaction tools. Three interaction tools were developed: selection/deselection of points; translation/rotation of selected points; and classification of selected points.

In the end, the developed solution proved to be functional for modifying the point cloud structure and further classify it. The use of geometry shaders and 3D stereoscopic vision was beneficial to distinguish points and their closeness to the user, while using one's hands to directly "touch" the points was very intuitive and increased ease of use, in relation to common, desktop-based, point cloud editing tools.

The remainder of this document echoes the following structure. In Section 2 (State of the Art) a literature review search and analysis of related works is conducted. Section 3 (Methodology) documents the approach of the solution, including the data and tools used, and its architecture. In Section 4 (Implementation), the details for the implementation following the proposed solution can be found. Section 5 (Results and Discussion) presents the advantages and disadvantages of the implemented solution and how the user tests should be conducted. Finally, Section 6 (Conclusion and Future Work) ends the article with a conclusion of the developed work and ideas for future improvements.

2 STATE OF THE ART

This section presents a literature review of the current related work in this field. It starts with a quick overview on the search methodology, followed by a description of the relevant publications and finishes with a comparisons and conclusions segment.

2.1 Methodology

To adequately acquire a picture of the current state of the art in point cloud manipulation in virtual reality environments, a systematic literature review was performed with the intent of finding most of the performed scientific work in this area. First, a set of searchable terms were gathered that specifically pointed to the topic being researched, which were then plugged into the most popular databases and search engines (Scopus, Web of Science and Google Scholar). The chosen terms were:

- "virtual reality" OR vr
- point* NEAR cloud*

This produced hundreds of results, most of which were not directly relevant to the research topic. To cut out the unwanted results, two criteria were composed:

1. Point cloud visualization must be immersive.
2. Some type of interaction or manipulation must happen in the immersive environment.

After applying these restrictions, the number of pieces of literature was reduced to only five, but in a niche area of research such as this one, that was expected.

2.2 Reviewed Literature

Despite being a short selection, the gathered literature review provides very interesting insights into the world of interactive point clouds in VR. The first two examples share a lot of similarities, as they focus on the visualization of LIDAR data of buildings.

The first one presents an open-source platform for visualizing classified point clouds built on top of Unity. The visualization system uses several optimization techniques, as simply rendering all points of a point cloud (which can be composed of billions of points) is not feasible. Their approach utilizes a modified octree to partition the point cloud based on the points position and class. This approach later allows the rendering engine to selectively and dynamically load and display only the points that are currently visible to the user. To help with the perception of depth a custom shader was developed that simulates shadows. In terms of interaction features, the user can select and deselect which classes should be visible through a custom GUI interactable via a ray casted pointer [5].

The other one also incorporates rendering optimizations and increases the interactivity of the user with the point cloud. The rendering engine was developed with openGL, GLSL and OpenVR. It incorporates several optimizations to reduce the number of rendered points such as view frustum and detail culling, and early fragment test for the cases not covered by the previous two. On top of this custom rendering engine, the authors present useful interaction features.: aside the basic manipulations of translating, scaling and rotating the point cloud, the user has a virtual measuring tape and surface area calculators at his disposal [13].

In a more practical application case, the next example uses point clouds generated from photogrammetry reconstruction of aerial pictures taken from a UAV of patches of forests. Their goal was to create an explorable immersive environment where the user could investigate and analyze the structure of the forest and individual trees. To aid in these tasks, the user has access to a virtual tape measure, similar to the one from the previous work, and each tree can be individually selected, displaying relevant information [6].

The last two examples are the only literature works found that incorporate direct interaction with the point cloud in an immersive environment. They both focus on the selection and annotation of points (which can also be referred as labeling or classification), usually to later train classification models. The first one uses the then novel Leap Motion to enable the user to use his hands and fingers as input. The tip of the index finger selects and deselects points to then label. Additionally, two hand gestures enabled translation, rotation and scaling of the point cloud itself. Since this solution was meant to be used while sitting in a desk, large point clouds can become cumbersome [7].

The other one introduces the labeling of animated point cloud data to immersive environments. The annotation is made through a virtual painting brush, similar to the previously mentioned approach. The innovation is the tracking of already labeled objects through time, meaning that after labeling the first frame of the animation, only touch-ups are required in the following frames, drastically reducing classification times [12].

2.3 Comparison and Conclusions

In terms of direct comparisons, only the first two have performance test data for direct comparison. The solution from Thiel et al. was able to maintain a steady 90 frames per second while displaying a point cloud of 11.6 million points [13], while the one by Kharroubi et al. averaged about 50 frames per second for 10 million points [5]. This could be due to the different performance enhancing features both systems present or, on a more basic level, the difference between the platforms in which they were built upon (Unity3D vs OpenGL + GLSL).

Regarding the interactions employed by each, they have their similarities and differences. All except one [7] allows the user to move freely in the environment, either physically or by teleporting.

Interactions with the point cloud seem to be related to the intent of the explored solutions. For the ones focusing on the exploration of the point cloud, informative tools like virtual measuring tapes

[6, 13] and GUIs to select which classes to visualize [5] or with information about the point cloud objects [6] were used.

The one thing that wasn't observed in the examples mentioned is the direct manipulation of points in the point cloud. Most desktop-based point cloud visualization and editing tools grant the user the ability to modify and make adjustments to the cloud structure and even the scalar variables associated with each point.

3 METHODOLOGY

This chapter details the approach taken in the development of the solution. It starts with a description of the proposed solution, followed by a description of the point cloud and software used, and ending with an explanation of the solution architecture.

3.1 Proposed Solution

To achieve the proposed goal of recreating common point cloud editing tools in an immersive environment, a new VR application will be created, specifically designed for interacting with point clouds. Unity will be used as a development base, as it integrates easily with most VR SDKs, and several open-source utilities for VR and point clouds are readily available.

Not all point clouds are alike with some being more complex than others. The point cloud data used in the work of Kharroubi et al. [5] includes the classification of the point, and to which object it belongs. With this in mind, it was decided that the solution would work with the PLY file type, which can be used for all kinds of 3D objects and allows the definition of additional scalar variables in its header. By designing our application to support classes and objects from the start, it can cover more use cases.

The development of the solution will split in two main parts: loading the PLY file in Unity and render the point cloud; and implement the interaction features. For part one, each object will be represented by an individual Unity *GameObject*, which has a limit of 65,000 mesh vertices. This means special care needs to be given to cases where an object exceeds that number of points.

For part two, the interactions to be implemented are as follows: select / deselect points, translate selected points, rotate selected points and classify selected points. Translating and rotating selected points of a point cloud is currently a novelty in immersive environments, while selecting / deselecting and classifying will follow a similar approach to the work of Stets et al. [12].

3.2 Point Cloud Data

For testing purposes, it was necessary to source a point cloud dataset. Since the presented solution was designed to work with classified point clouds with extracted objects, a suiting point cloud was needed. Currently, a popular use for classified point clouds is to train models to be used in self driving cars with LiDAR scanners.

A quick search returned several point clouds already classified of street scenes for this purpose. One of which was the Paris-Lille-3D dataset [11] which was encoded in the desired PLY file format, and already segmented in 6 classes: undefined, ground, building, car, post and vegetation. The "Mini Lille" point cloud, containing approximately 2 million points, was used. Object identification will have to be done separately, but it can be easily done in common point cloud editing software.

3.3 Used Tools and Software

For preparing the point cloud to be used in the proposed application, two free software were used for different tasks. CloudCompare is a "3D point cloud (and triangular mesh) processing software" (https://www.danielgm.net/cc/) that includes many point cloud processing, analysis, segmentation and reconstruction tools and algorithms. It was used to extract individual objects from the chosen dataset.

Meshlab is "the open-source system for processing and editing 3D triangular meshes." (https://www.meshlab.net/). It has similar functionalities to CloudCompare, but focuses more on algorithms for cleaning, reconstructing and texturing meshes. It is also possible to run Meshlab through the command line, feature that was used to create meshes from the point cloud objects extracted in CloudCompare for better visualization when in an immersive environment.

As mentioned before, Unity was used as the basis for the solution implementation. On top of that, the SteamVR Unity plugin (https://github.com/ValveSoftware/steamvr_unity_plugin) was used to increase the compatibility with most HMD system on the market. The free-to-use VRTK (VR ToolKit) (https://vrtoolkit.readme.io/) was also used to handle the basics of the interactions.

3.4 Solution Architecture

The proposed solution includes several steps and additional external modules, which work together to create the final immersive application. Its structure can be visualized in Fig. 1

In summary, and in order, a classified point cloud file (PLY format) is transformed in CloudCompare to extract the individual objects of each class. This new PLY file is read in Unity by a custom PLY file Parser capable of reading the class and object attributes. Each object point cloud is then processed externally in Meshlab to retrieve a mesh that represents that object. After all objects are ready, the immersive application loop starts. The point cloud is rendered and shown to the immersed user through SteamVR. The user inputs are then captured with help from VRTK and the point cloud is modified according to the user actions. The loop is then closed by rendering and showing the altered point cloud to the user.

4 IMPLEMENTATION

In the wake of the above outlined methodology, this section describes the implementation process, including how the point cloud data was prepared, imported into Unity, and describing the interaction tools.

4.1 Preparing Point Cloud Data

As mentioned in the previous section, the first step is preparing the data for use in the application. This was achieved in CloudCompare, using its *Label Connected Components* tool. It forms groups of points given a certain minimum distance. This worked perfectly on the buildings, cars and posts, while on vegetation it is unclear, as it is difficult to differentiate between distinct bushes, for instance. This segmentation was then saved in a new PLY file, now including an object variable for each point, in the form of an integer. It is saved in ASCII format, which will be substantially easier to parse further ahead, when compared to the original binary format.

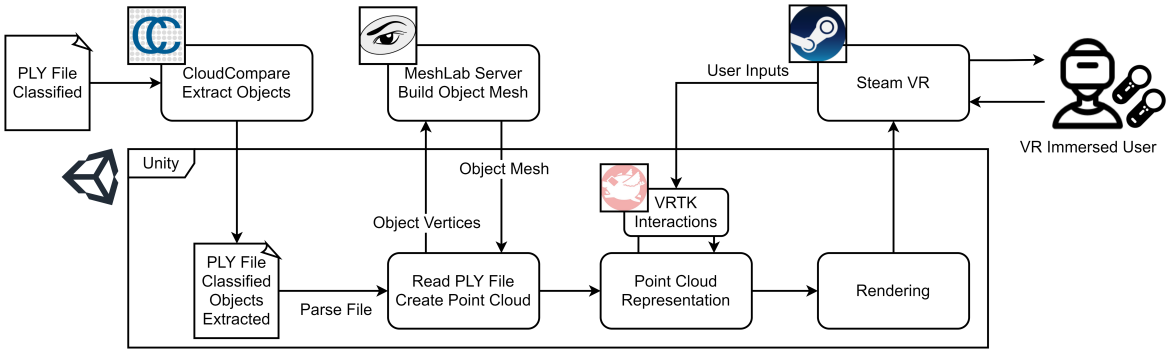

Figure 1: Proposed solution architecture, with the different components and their integration.

4.2 Importing the Point Cloud to Unity

The next step is importing the point cloud to Unity and achieving a visible and interactable point cloud. This step is divided in three phases: parsing the PLY file; creating the object polygon mesh; and rendering the point cloud.

4.2.1 Parsing the PLY File. Before implementing a parser from scratch, a search was conducted on the Unity asset store for potential implementations of PLY file readers. The closest tool to the requirements was the "*Point Cloud Free Viewer*" by Gerard Llorach (https://github.com/gerardllorach/Unity-Point-Cloud-Free-Viewer), that uses the .OFF file type as input, which shares some similarities with the PLY format.

A PLY file parser was then created, following the structure of Llorach's implementation. This included dividing the point cloud while making sure that each vertex mesh did not surpass 65.000 points, as Unity would then ignore the excess. Each object's points were stored in one or more *Mesh Filters*, under the same *GameObject*.

4.2.2 Creating Object Polygon Mesh. One difficulty of working with point clouds in general, but specially in immersive environments, is the difficult perception of what the points represent. To help combat this, a surface mesh of the objects captured in the point cloud was created. To avoid implementing this directly in Unity, the specialized tool MeshLab was used.

The MeshLab command line tool, MeshLab Server, was used for ease of integration. The instructions to be followed by MeshLab are detailed in a custom MLX script. The commands in the script are as follows: first the normal vector for each point is calculated; followed by the use of the *Ball Pivoting Algorithm* [2] to create the mesh, which despite being 20 years old, still produces good results and is fast.

To run the MLX script, the point cloud is first saved in a temporary file and an external process is executed. Upon finishing the script, a new PLY file is created with the object's polygon mesh. This is also parsed in Unity, and the new mesh is added to the object's *GameObject.* As this process is very time consuming, it only needs to be executed once per point cloud, provided that no modifications were made.

4.2.3 Rendering the Point Cloud. Rendering a point cloud requires a different process to rendering a typical mesh with faces. Two different approaches were found from open-source projects. The first one, from the aforementioned Gerard Llorach, used a simple shader to paint the closest camera pixel the color of the vertex. This is a fast solution, but with the points being one pixel in width with no shading or defined edge, the clarity was subpar.

An alternative was to use geometry shaders, as implemented by Keijiro Takahashi in his point cloud renderer (https://github.com/keijiro/Pcx) for Unity titled "*Pcx*". By using geometry shaders, it is possible to create points of any size and any regular polygonal shape. For greater point differentiation, a black outline was given to each point, making it easier to discern which points are closer and which ones are further away. The difference between the original single pixel point shader and geometry shader with outline can be seen in Fig. 2

4.3 Immersive Point Cloud Interactions

With this being a VR application, frames per second (FPS) performance must be given special attention to prevent the user from breaking immersion or suffering simulator sickness. To this end, the point cloud is rendered using the pixel point shader, which is significantly less computational heavy than the geometry shader. To allow the user to benefit from the latter, he can select one object at a time to be rendered using this shader, while the rest use the former. This selection is made with a raycasted pointer.

With an object selected, the user can use his left controller to access a radial menu that lets him turn visible or invisible the polygon mesh for that object and increase or decrease the size of the geometry shader points. The remainder of the interaction tools will be available to the user through a virtual, right wrist mounted, toolbox, in the form of another radial menu. In terms of user movement through space, two option are provided. The first and simpler one is a simple teleport with a Bézier curve pointer to select the destination. The other is smooth locomotion, which also allows the user to move vertically, since some point cloud objects, like buildings, are very tall when using 1:1 scaling. The implementation of the tools was divided in 3 phases: selecting / deselecting points; transform the selected points; and classification of selected points.

4.3.1 Selecting/Deselecting Points. As mentioned in the methodology section, the selecting and deselecting tools work similarly to a brush (as done by Stets et al. [12]), but with some differences, the first being that instead of having to "brush" each individual

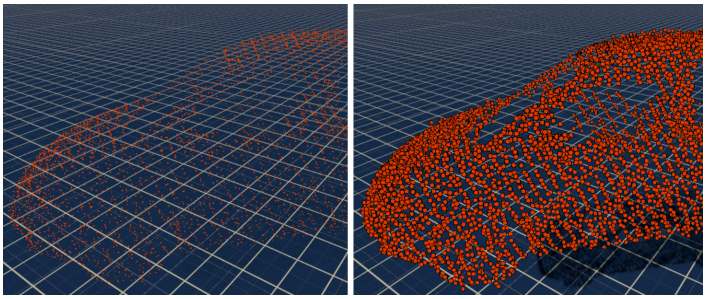

Figure 2: Comparison between single pixel point shader (left) and outlined geometry shader (right) for a point cloud of a car.

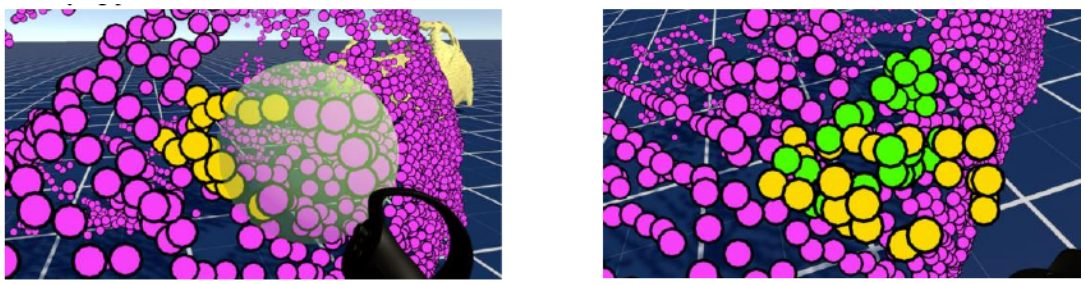

Figure 3: Left: user selecting points from the left side mirror of a car. Selected points turn yellow. Right: User translating and rotating the same selection. Notice how the yellow points indicate the original position, while the green ones show the user where he is moving them to.

point one by one with a "small and sharp tip" our solution makes use of a size-adjustable sphere as the brush. The second difference is that instead of the brush selecting what comes in contact with the tip immediately, it only does so when the right trigger button is pressed. The objective of this change is to prevent accidental selection of points. The selection sphere itself is tinted either green or red, representing the selection and deselection tools, respectively. It is also translucent to allow the user to see which points are correctly being encircled by the selection tool. Once a point is selected, it changes color to yellow, strongly contrasting with the magenta of the unselected points.

4.3.2 Move/Rotate Selection. With a selected subsection of the object's point cloud, the user can translate and rotate that specific selection, thus modifying the point cloud structure. To do so, he must first deselect any other active tool. To translate the subset, the user simply has to press the right hand grab button and move the right controller. By doing so, an identical subset of points colored in green appears, to give a visual indication of where the points will be shifted to. Once the grab button is release, the green subset disappears, and in its place are now the selected points.

To rotate the selection, the user must press the left hand grab button while already pressing the right hand grab button (translation mode). Then, by rotating the left hand, the green subset point cloud rotates too. Figure 3 shows the user selecting and moving/rotating a subset of points. It is expected that putting the transformation of the points right in the hands of the user, the experience can be more natural, immersive, and straightforward.

4.3.3 Classify Selection. To use the classification tool, the user must first make a selection. After adjusting his selection as desired, the user can select the classification tool in his toolbox, which spawns the tool's panel. This panel is composed of three main parts: the keyboard, the color selector and the class selector.

In order for the user to create a new class, he must input its name with the keyboard and use the color sliders to create the color with which to identify that class. In case the intended class was previously segmented in that object, the user can simply select it from the dropdown menu on the right. After either creating a new class or choosing an existing one, when the user clicks the save button, those points change color (to the selected color) and can no longer be selected. The performed classifications are stored in memory, to later be saved once the user is finished. Figure 4 shows the user classifying points with the tool.

5 RESULTS AND DISCUSSION

The development of the immersive point cloud manipulation application was successful and works as intended. After getting used to the placement of the tools in the controller buttons, it becomes very intuitive to select and adjust the position and rotation of the selected points. When compared to the traditional point cloud editing tools like CloudCompare and MeshLab, the presented immersive solution is less cumbersome to use, due to the simplicity of the GUI, which in the mentioned 2D tools, can be cluttered and difficult to use.

The use of immersive 3D technologies also helps with point depth perception, especially when combined with the outlined

Figure 4: User classifying the A-Pillar of the car. Notice how the previously classified left mirror is now blue/grey and that the class *MIRROR LEFT* is present in the dropdown.

geometry shader. In addition, being immersed in a virtual world with the point cloud and using one's own hands to interact directly with the points feels more natural and straightforward.

In terms of performance, no hang-ups or declines in frame rate were detected while testing with the solution.

While no technical user tests were performed, they will be conducted in the future to validate the usefulness and ease of use of the solution. For this, test subjects will be given similar point cloud modification and segmentation tasks to be performed in a traditional desktop environment and using the developed solution. To evaluate their performance, the time to completion and result accuracy will be compared. Questionnaires about the ease of use, immersion and engagement will also be given to the subjects.

6 CONCLUSION AND FUTURE WORK

An immersive point cloud editing and classification tool was developed, bringing for the first time per point manipulation to an immersive environment. In terms of contributions, the developed PLY file parser and renderer for Unity is a plus for future users and developers. With the groundwork done, it is also now easier for other researchers to develop additional point cloud interaction features. In the end, the test performed on the solution indicates that working with point clouds in an immersive environment brings significant advantages such as effortless point depth perception and intuitive hand gesture-based point cloud manipulation. It still falls short of the versatility that regular point cloud manipulation software provides, which include several tools and features. However, as demonstrated with MeshLab Server, these can be brought into an immersive environment too.

As for future work, as mentioned in the previous section, it is important to conduct the described evaluation methodology with test subjects to ascertain the value of this solution in terms of usability and usefulness. As for improvements to the solution, to guarantee the scalability in relation to the number of points of the cloud, performance enhancing features like the ones used by Kharroubi et al. and Thiel et al. must be implemented, as some point clouds can reach billions of points. Additional manipulation features and integrations with MeshLab could also benefit usability, as well as improvements such as an undo button or a cooperative mode.

REFERENCES

[1] Ayman Ammoura. 2001. DIVE-ON: From Databases to Virtual Reality. XRDS 7, 3 (March 2001), 4-ff. https://doi.org/10.1145/367884.367891
[2] F. Bernadini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. 1999. The ball-pivoting algorithm for surface reconstruction. IEEE Transactions on Visualization and Computer Graphics 5, 4 (October 1999), 349-359. https://doi.org/10.1109/2945.817351
[3] Ciro Donalek, S. G. Djorgovski, Alex Cioc, Anwell Wang, Jerry Zhang, Elizabeth Lawler, Stacy Yeh, Ashish Mahabal, Matthew Graham, Andrew Drake, Scott Davidoff, Jeffrey S. Norris, and Giuseppe Longo. 2014. Immersive and collaborative data visualization using virtual reality platforms. In 2014 IEEE International Conference on Big Data. 609-614. https://doi.org/10.1109/BigData.2014.7004282
[4] Seth M. Feeman, Landon B. Wright, John L. Salmon. 2018. Exploration and evaluation of CAD modeling in virtual reality. Computer-Aided Design and Applications 15, 6 (2018), 892-904. https://doi.org/10.1080/16864360.2018.1466807
[5] A. Kharroubi, R. Hajji, R. Billen, F. Poux. 2019. Classification and Integration of Massive 3d Points Clouds in a Virtual Reality (vr) Environment. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XLII-2/W17 (December 2019), 165-171. https://doi.org/10.5194/isprs-archives-XLII-2-W17-165-2019
[6] D. Lee, W. Muir, S. Beeston, S. Bates, S. D. Schofield, M. J. Edwards, and R. D. Green. 2018. Analysing Forests Using Dense Point Clouds. In 2018 International Conference on Image and Vision Computing New Zealand. https://doi.org/10.1109/IVCNZ.2018.8634651
[7] P. Lubos, R. Beimler, M. Lammers, and F. Steinicke. 2014. Touching the Cloud: Bimanual annotation of immersive point clouds. In 2014 IEEE Symposium on 3D User Interfaces. 191-192. https://doi.org/10.1109/3DUI.2014.6798885
[8] Stefan Marks, Javier E. Estevez, and Andy M. Connor. 2014. Towards the Holodeck: Fully Immersive Virtual Reality Visualisation of Scientific and Engineering Data. In Proceedings of the 29th International Conference on Image and Vision Computing New Zealand. Hamilton, New Zealand, 42–47. https://doi.org/10.1145/2683405.2683424
[9] Sven Oesau. 2015. Geometric modeling of indoor scenes from acquired point data. PhD Thesis, Université Nice Sophia Antipolis.
[10] Noritaka Osawa, Kikuo Asai, Yuji Y. Sugimoto. 2000. Immersive Graph Navigation Using Direct Manipulation and Gestures. In Proceedings of the ACM Symposium on Virtual Reality Software and Technology. Seoul, Korea, 147–152.
[11] Xavier Roynard, Jean-Emmanuel Deschaud, and François Goulette. 2017. Paris-Lille-3D: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification. The International Journal of Robotics Research 37, 6 (May 2018), 545-557. https://doi.org/10.1177/0278364918767506
[12] Jonathan Dyssel Stets, Yongbin Sun, Wiley Corning, and Scott W. Greenwald. 2017. Visualization and Labeling of Point Clouds in Virtual Reality. In SIGGRAPH Asia 2017 Posters. Bangkok, Thailand, Article 31, 1–2. https://doi.org/10.1145/3145690.3145729
[13] F. Thiel, S. Discher, R. Richter, and J. Döllner. 2018. Interaction and Locomotion Techniques for the Exploration of Massive 3D Point Clouds in VR Environments. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XLII-4 (October 2018), 623-630. https://doi.org/10.5194/isprs-archives-XLII-4-623-2018
[14] Martin Weinmann. 2016. Preliminaries of 3D Point Cloud Processing. In: Reconstruction and Analysis of 3D Scenes. Springer, Cham. https://doi.org/10.1007/978-3-319-29246-5_2
[15] Josef Wolfartsberger. 2019. Analyzing the potential of Virtual Reality for engineering design review. Automation in Construction 104 (August 2019), 27 – 37. https://doi.org/10.1016/j.autcon.2019.03.018