# A Virtual Reality Interface for Interactions with Spatiotemporal 3D Data

Hunter Quant, Sean Banerjee, and Natasha Kholgade Banerjee(✉)

Clarkson University, Potsdam, NY 13699, USA
{quanthd,sbanerje,nbanerje}@clarkson.edu

**Abstract.** Traditional interfaces for interacting with 3D models in virtual environments lack support for spatiotemporal 3D models such as point clouds and meshes generated by markerless capture systems. We present a virtual reality (VR) interface that enables the user to perform spatial and temporal interactions with spatiotemporal 3D models. To accommodate the high volume of spatiotemporal data, we provide a data format for spatiotemporal 3D models which has an average speedup of 3.84 and a space reduction of 43.9% over traditional model file formats. We enable the user to manipulate spatiotemporal 3D data using gestures intuitive from real-world experience or by using a VR user interface similar to traditional 2D visual interactions.

**Keywords:** Virtual reality · Spatiotemporal 3D models
User interface

## 1 Introduction

With the ubiquity of motion capture systems and the rise of multi-camera systems for markerless motion capture, a large quantity of spatiotemporal 3D data is being generated in the form of point clouds [7] and meshes [2,3]. While software such as AutoDesk MotionBuilder enables the user to visualize motion capture points, there exist limited approaches [1,2] to visualize spatiotemporal point cloud and mesh data. The interactions provided by these approaches are restricted to spatial manipulations and temporal playback using traditional user interface devices such as mouse, keyboard, and screen. Approaches in augmented reality [4] allow for temporal interactions on animated 3D models, however this does not accommodate non-rigged spatiotemporal 3D data obtained by markerless motion capture, such as sequenced point clouds and meshes.

In this demo, we present a virtual reality (VR) interface for interacting with spatiotemporal 3D models. In addition to spatial transformations such as rotation, scaling, and translation provided by traditional approaches of VR interaction on objects [5,6,9], our approach provides novel temporal transformations

**Electronic supplementary material** The online version of this chapter (https://doi.org/10.1007/978-3-319-73600-6_29) contains supplementary material, which is available to authorized users.

in VR such as rewind, fastforward, and playback on spatiotemporal 3D models such as point cloud and mesh collections, where the object has a separate model specification at each time instant. Our work is distinct from current temporal approaches for VR interaction that focus on editing of VR footage [8]. Our interface allows users to interact with spatiotemporal 3D models either using gestures intuitive through real-world experience or using interactions with a user interface (UI) intuitive through traditional interactions with 2D screen content. Our VR interface is implemented in the Unity 3D Game Engine [10] and runs on the HTC Vive and the Oculus Rift CV1 with Oculus Touch.

To load high volumes of spatiotemporal 3D data into memory for VR interactions, we create a novel data file format that provides an average speedup of 3.84 and a space reduction of 43.9% over traditional model file formats such as OBJ and PLY. We report load time and space usage results for temporal 3D point cloud and 3D mesh collections on a Windows 10 desktop with an NVIDIA GTX 1080 Ti graphics processing unit (GPU) and an Intel i7 7700K processor.

## 2 VR Interface to Interact with Spatiotemporal 3D Data

**Gestural Interactions.** As shown in Fig. 1, the user can perform spatial and temporal interactions on spatiotemporal 3D data using swipe gestures on the main controller. The user switches between transformation modes by swiping the auxiliary controller. Our approach allows the user to perform translation along the axis of the pointer using vertical swipes. We also provide a freeform translation mode, where the user grabs the object by pulling the trigger on the main controller while positioning its pointer on the object, moves the controller, and releases the trigger to set the object at a desired destination.

Our approach provides horizontal swipes to rewind and fast-forward temporal changes and to scale the object size. To perform rotation the user swipes in the desired rotation direction, and rotation occurs along the axis $\mathbf{v}_{\text{rot}}$ calculated as $\mathbf{v}_{\text{rot}} = x\mathbf{v}_r - y\mathbf{v}_{\text{up}}$. Here, $\mathbf{v}_{\text{up}}$ is the up vector given as $\left[0\ 1\ 0\right]^T$, $x \in [-1, 1]$ and $y \in [-1, 1]$ are coordinates of a 2D vector $\mathbf{v}_{\text{input}}$ along the swipe direction, and $\mathbf{v}_{\text{r}}$ is a unit vector perpendicular to the up vector $\mathbf{v}_{\text{up}}$ and a unit vector $\mathbf{v}_p$ in the direction of the pointer through the controller into VR space. The value of $\mathbf{v}_{\text{r}}$ is calculated as $\mathbf{v}_{\text{up}} \times \mathbf{v}_{\text{p}} / \left\|\mathbf{v}_{\text{up}} \times \mathbf{v}_{\text{p}}\right\|$. The rotation occurs around the origin of the model coordinate system.

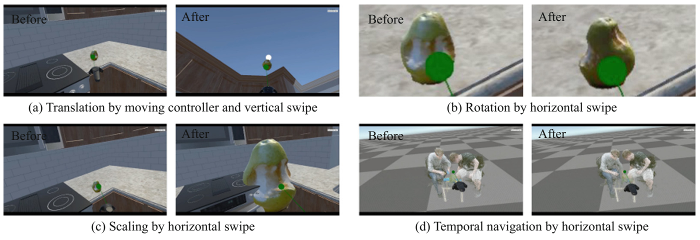
Before
After
(a) Translation by moving controller and vertical swipe
Before
After
(b) Rotation by horizontal swipe
Before
After
(c) Scaling by horizontal swipe
Before
After
(d) Temporal navigation by horizontal swipe

**Fig. 1.** Gestural interactions for (a) translation, (b) rotation, (c) scaling, and (d) temporal navigation.

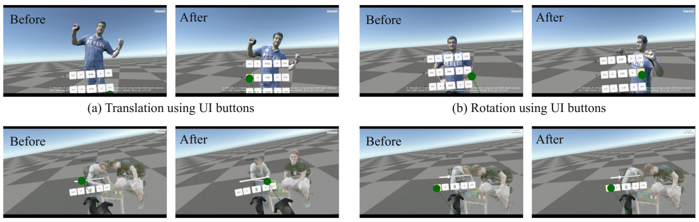
Before
After
(a) Translation using UI buttons
Before
After
(b) Rotation using UI buttons
Before
After
(c) Fast forward using UI slider
Before
After
(d) Rewind using UI buttons

**Fig. 2.** UI interactions for (a) translation, (b) rotation, (c) temporal navigation slider, and (d) temporal navigation buttons. Translation and rotation is performed on data associated with [2].

**Interactions with user interface (UI).** As shown in Fig. 2, the user can also use the pointer and trigger on the main controller to interact with a UI displayed to the user in VR whose motion in VR space is attached to the auxiliary controller. The user can point to and trigger the single and double arrow buttons respectively to perform small and large increments of spatial transformation along the three axes or temporal transformation forward and backward in time. We provide a UI slider that enables the user to perform navigation back and forth in the temporal domain. The user triggers the pause/play button on the UI to pause or playback the spatiotemporal 3D data in the virtual space.

## 3 3D Spatiotemporal Data Format

To load spatiotemporal 3D models with high vertex and face counts of between 10,000 to 500,000 vertices that are typical outputs of modern markerless motion-capture systems, we provide a file structure that consists of a header file to specify the number of frames in the object, a list of properties present in the model such as vertices, faces, and color, and a flag that specifies whether a property is shared by all frames or not. Properties are stored as binary files. In the supplementary video, we show meshes of fruit aging that contain vertices and a face structure shared by all frames, meshes of single person captures from data associated with [2] that contain individual vertices and faces for each frame, and point clouds of single and multi-person captures that contain individual vertices with no faces.

Table 1 shows counts of properties such as vertices and faces for the fruit meshes and the point clouds, together with space usage in MB and runtimes in milliseconds per frame (ms/frame) for our file format as compared to traditional file formats such as OBJ for the fruit, and ASCII PLY for the point clouds. As shown by the table, our file format uses lesser space with an average space reduction of 54.2% from the OBJ format, 38.7% from the PLY format, and 43.9% overall. Our format also loads faster into memory, with an average loading speedup of 2.70 over the OBJ format, 4.40 over the PLY format, and 3.84 overall. By using a global header file, we minimize per-frame header reading time, and provide faster load time per MB, with an average speedup normalized for storage of 1.24 for OBJ files, 2.7 for PLY files, and 2.21 overall.

**Table 1.** Vertex and face counts, space usage (MB), space reduction, load time (ms/frame), speedup, and speedup normalized for space usage for our file format versus traditional formats such as OBJs for pear and banana meshes and PLYs for point clouds of paper plane, guitar, rock-paper-scissors (RPS), and origami.

| Name | Verts | Faces | Space usage | | Space reduction | Load time | | Speedup | Normalized speedup |
|---|---|---|---|---|---|---|---|---|---|
| | | | Ours | Trad. | | Ours | Trad. | | |
| Pear | 22633 | 45262 | 103 | 225 | 54.2% | 76.6 | 238.6 | 3.12 | 1.43 |
| Banana | 18226 | 36448 | 83.4 | 182 | 54.2% | 82.0 | 188.0 | 2.29 | 1.03 |
| Plane | 82609 | - | 189 | 309 | 38.9% | 70.9 | 295.7 | 4.17 | 2.55 |
| Guitar | 82079 | - | 187 | 315 | 40.6% | 84.7 | 302.9 | 3.57 | 2.12 |
| RPS | 134778 | - | 308 | 482 | 36.1% | 94.8 | 427.2 | 4.51 | 2.88 |
| Origami | 146198 | - | 334 | 552 | 39.5% | 97.5 | 522.6 | 5.36 | 3.24 |

## 4 Discussion

We have provided a virtual reality interface that expands the gamut of VR interactions to handle temporal transformations to spatiotemporal 3D data such as point clouds and meshes in conjunction with traditional spatial transformations. Our current data file format provide support for vertices and mesh faces. We will expand the file format to include support for geometric properties such as normals and mesh smoothness values, and reflection models with their associated material properties. In future, we will provide visualization of multiple forms of textural information such as thermal and multispectral data, depth and shadow maps, and ambient occlusion. Future work will also include enabling creation and editing of spatiotemporal 3D data through our interface.

**Acknowledgements.** This work was partially supported by the National Science Foundation (NSF) grant #1730183.

## References

1. Akhter, I., Simon, T., Khan, S., Matthews, I., Sheikh, Y.: Bilinear spatiotemporal basis models. ACM Trans. Graph. **31**(2) (2012)
2. Alexiadis, D., Zarpalas, D., Daras, P.: Fast and smooth 3D reconstruction using multiple RGB-depth sensors. In: 2014 IEEE VCIP, pp. 173–176, December 2014
3. Collet, A., Chuang, M., Sweeney, P., Gillett, D., Evseev, D., Calabrese, D., Hoppe, H., Kirk, A., Sullivan, S.: High-quality streamable free-viewpoint video. ACM Trans. Graph. **34**(4) (2015)
4. Dong, S., Behzadan, A.H., Chen, F., Kamat, V.R.: Collaborative visualization of engineering processes using tabletop augmented reality. Adv. Eng. Softw. **55**, 45–55 (2013). https://doi.org/10.1016/j.advengsoft.2012.09.001
5. Google: Blocks. https://vr.google.com/blocks/
6. Han, S., Lee, H., Park, J., Chang, W., Kim, C.: Remote interaction for 3D manipulation. In: Proceedings CHI (Extended Abstracts) (2010)

7. Kowalski, M., Naruniec, J., Daniluk, M.: Live scan3D: a fast and inexpensive 3D data acquisition system for multiple kinect v2 sensors. In: IEEE 3DV (2015)
8. Nguyen, C., DiVerdi, S., Hertzmann, A., Liu, F.: Vremiere: in-headset virtual reality video editing. In: Proceedings of the CHI (2017)
9. Oculus: Medium. https://www.oculus.com/medium/
10. Unity: Unity 3D. https://unity3d.com/