Volumetric Capture and Replay in Virtual Reality – Entering the age of immersive and volumetric analysis in CSCL

Jacob Davidsen, Paul McIlvenny, Artúr Barnabás Kovács
jdavidsen@ikp.aau.dk, paul@ikl.aau.dk, kovacs.artur.barnabas@gmail.com
Aalborg University - BigSoftVideo

Abstract: Throughout the history of CSCL, video has been used to capture, analyze and present collaborative learning activities. Recently, the use of eye-tracking and sensor tracking of movement has been used along with video data. These new computer-supported analytics can tackle big data sets, but we suggest that volumetric data is also important to explore qualitatively in the context of CSCL. While it is impractical to capture collaborative activities volumetrically in everyday physical space, it is feasible in Immersive Virtual Reality (IVR). In IVR, researchers are offered new opportunities for keeping track of and sharing an analysis using Volumetric Capture and Replay (VolCap). In this paper, we present a VolCap solution for immersive analysis of video data in CSCL. With this solution, researchers can inhabit earlier analyses using *AVA360VR* and follow all the virtual actions performed by the VolCapper volumetrically.

Introduction

Researchers in CSCL study collaborative activities using different types of data and methodological frameworks (Jeong et al., 2014). In recent years, video data has played a prominent role in both Multimodal Learning Analytics (MMLA) research (Martinez-Maldonado et al., 2013) and studies focusing on the interactional work students do when collaborating (e.g. Steier & Davidsen, 2021). In addition, CSCL researchers have also used bio-sensing data for studying collaborative learning, e.g. eye-tracking data (Schneider et al., 2018). A relatively unexplored CSCL setting is Immersive Virtual Reality (IVR), which is difficult to capture using traditional 2D video recordings because the immersive experience of the 3D environment will be reduced to a monoplanar version of the event. In addition, although IVR actions can be tracked using bio-sensing data and input controller behavior, that is still not close to the volumetric experience of collaborating in an immersive environment. That is why we introduce Volumetric Capture and Replay (VolCap), which has a crucial role in the development of an innovative research infrastructure to support the analysis of immersive collaborative activities.

Over the last four years, the BigSoftVideo team at Aalborg University (Denmark) have developed a free software package called *AVA360VR* (Annotate, Visualise and Analyse 360° video in Virtual Reality) for working with 360° video using IVR (https://bigsoftvideo.github.io/AVA360VR/). The basic 3D virtual toolset of *AVA360VR* includes a drawing pen, a key frame sequencer, a video re-camming tool, a synced transcript viewer and a comic creator (see McIlvenny 2018 for a full review of the software), which the researcher can use in the analysis of collaborative activities recorded with 360° video. While *AVA360VR* is an innovative tool for working with video data because it allows researchers to be immersed in the data, it also offers a novel new way of capturing and replaying an analytic performance using VolCap (McIlvenny, 2020). The purpose of this paper is to present our VolCap solution and to discuss how it can be used in the context of CSCL research. First, we outline the differences between video and volumetric data, and then we present VolCap as implemented in *AVA360VR*.

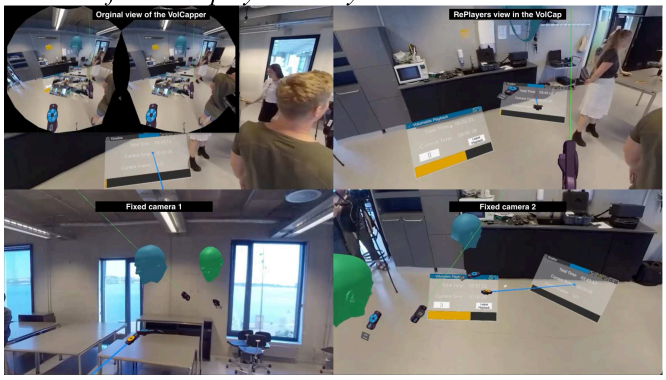
From video and transcripts to volumetric capture and replay

Video has been used to capture, analyze and present collaborative learning activities in CSCL for more than three decades. In CSCL, video is not exclusively used in quantitative studies or qualitative studies. In qualitative studies, video is often transcribed and transformed into a text or a comic, which serve as the basis for the analysis (Steier & Davidsen, 2021). In quantitative studies, coding and counting of specific actions or types of utterances captured on video are transformed into graphs and statistics, e.g. MMLA. In both cases the video data of the event is left behind in the shadows – and it is difficult for readers of the papers to follow how the analysis actually was made and how the transformed representations were derived from the video recording. Furthermore, video, transcripts and statistics are a ‘flat’ version of the three-dimensional world (McIlvenny, 2020; McIlvenny & Davidsen, 2017). In most cases, interaction in IVR environments is still transformed into 2D video using screen capture tools. This is problematic as the experience of interacting in an immersive environment is markedly different when compared to the monoplanar 2D representation, which might lead to false assumptions about what is going on in IVR. But with the introduction of 360° video, researchers are now exploring new ways of working with video data (McIlvenny, 2020; McIlvenny & Davidsen, 2017, McIlvenny & Davidsen, accepted) that is not restricted to media players or transcription tools on a flat screen. That is why a further step is required so that immersive activities in the context of CSCL can be captured and replayed using volumetric data. Volumetric performance data "relies on a procedural camera/spectator-independent representation of a dynamic real or virtual volumetric space over time." (McIlvenny, 2020, p. 801). As noted by McIlvenny (2020), the volumetric real time capture of a dynamic, everyday physical space is still far off, but with faster lidar scanning combined with smaller consumer devices we start to see the affordances of capturing data volumetrically. It is, however, possible to capture data volumetrically in IVR. With volumetric data, we are regenerate a three-dimensional space, in which one is not a passive spectator of the recording, but an active participant in re-activating the volumetric capture (McIlvenny & Davidsen, accepted).

Introducing VolCap and RePlay in *AVA360VR*

In the past three years, the BigSoftVideo team have developed a tool for analyzing, visualizing and annotating 360° video in IVR. In the most recent version of the software, a VolCap tool has been developed by which users can capture what they do in IVR and re-activate (RePlay) that volumetric capture afterwards. When capturing a VolCap in *AVA360VR*, all the actions of the user in IVR are captured volumetrically in IVR, including the HMD and controllers, and a microphone in the Head Mounted Display (HMD) is recording what the user is saying. The VolCap is saved and can be imported and reviewed by the same or another user at a later point in time. Figure 1 shows a composite image from a re-activated VolCap, in which a researcher reviews an analysis by another researcher of a distributed collaborative learning exercise recorded with multiple 360° cameras. At top left is the viewport from the original VolCapper overlaid by the RePlayer's viewport in the HMD, while at top right is the 2D viewport of the RePlayer. The two images below are showing outputs from two different virtual cameras that can be positioned where the RePlayer wants to – the green avatar head is the RePlayer and the blue is the Volcapper. When the VolCap is re-activated, the RePlayer will see a blue avatar head in the scene (and their controllers), which represents the user who performed the original VolCap. Now the RePlayer will be able to follow each and every action of the original VolCapper – and can even teleport around the VolCapper to obtain the best possible position for following the analysis by the VolCapper in 3D. Moreover, many of the same 3D tools and views available to the VolCapper are also available (emic) to the RePlayer.

Figure 1
A re-activation of a VolCap by a RePlayer

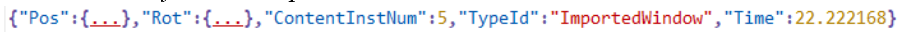


Technical implementation of VolCap and RePlay

A volumetric capture in *AVA360VR* can be separated into two main components. One component is the recorded voice of the user and the other is everything else in the virtual world. We record the voice using the microphone API provided by the game engine coupled with an Ogg encoder to produce an ogg file for each volumetric capture. In the following, we describe how the rest of the virtual world is captured and replayed in the context of the object-oriented game engine (Unity).

At any given time, all objects (e.g. avatars, cameras, videos, UI, controllers) in the virtual world have a state in the game engine. This state consists of properties like position, orientation, or in case of multimedia: current volume, current playhead position, etc.. When the user starts the capture, a "capture file" gets created and the state of all existing objects is written to the file. Afterwards, whenever the state of an object changes, the new state is written to the JSON file. More precisely, the capture system checks about 15 times a second if the state of an object has changed since the last check and writes the new state to the file when a change is detected. Each state contains an object type (*TypeId*), an instance number (*ContentInstNum*) and the timestamp (*Time*) that is just the elapsed time from the beginning of the capture (i.e. all initial states have a timestamp of zero). The instance number is used to identify objects. A simplified state looks like this:

Figure 2
Simplified JSON state for a VolCap in AVA360VR

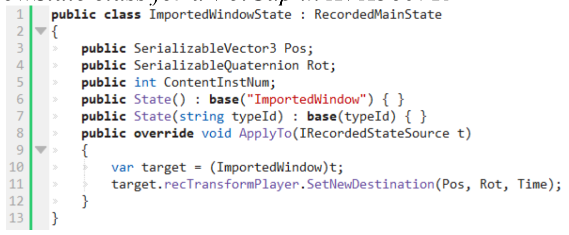


```
{"Pos":{...},"Rot":{...},"ContentInstNum":5,"TypeId":"ImportedWindow","Time":22.222168}
```

Each state is serialized into a single line of JSON and written to disk. Whenever the state needs to be captured it is converted to a single line of JSON first and then that line of JSON is appended to the end of the capture file followed by a line feed character (\n). It may seem inefficient to capture 3D, real-time events in a text-based format, but for our use-case it is fast and compact enough. On average, one minute of capture takes up less than a megabyte (about 1.5 megabytes if we add the recorded voice). In addition, the text format allows easier debugging.

During replay, *AVA360VR* reads this file from the beginning line-by-line. Each line is parsed into JSON. Next the program reads the *TypeId* field and determines what function to call to convert the JSON to the C# class of the state. Each state has a corresponding C# class. For example, here is the simplified state class for the *ImportedWindow:*

Figure 3
Simplified ImportedWindowState class for a VolCap in AVA360VR



```
public class ImportedWindowState : RecordedMainState
{
    public SerializableVector3 Pos;
    public SerializableQuaternion Rot;
    public int ContentInstNum;
    public State() : base("ImportedWindow") { }
    public State(string typeId) : base(typeId) { }
    public override void ApplyTo(IRecordedStateSource t)
    {
        var target = (ImportedWindow)t;
        target.recTransformPlayer.SetNewDestination(Pos, Rot, Time);
    }
}
```

As shown in the example, each C# state class has an *ApplyTo* method that takes the target object as a parameter and performs the required steps to reset the target object to the appropriate state. Note that *RecordedMainState* defines the *TypeId* and the *Time* fields. From here, there are two scenarios: (1) the object already exists (the program previously encountered a state with the same *ContentInstNum*), or (2) the object does not yet exist (this is the first state with this specific *ContentInstNum*). If the object already exists, the program calls the *ApplyTo* function of the state passing the existing target object as its parameter. In scenario 2, when the object does not yet exist, the program uses the *TypeId* field of the state to determine what function to call for instantiating the object. For example, if the *TypeId* is ‘ImportedWindow’ the program would call *CreateImportedWindow*. Then the new object gets registered with the *ContentInstNum* of the state, so that all following states belonging to this instance can be applied to it.

As the replay is running in linear time, the program reads all states from the capture file until it finds a state that has a timestamp higher than the time elapsed since the beginning of the replay. This way the software only reads as much data from the capture file as it needs to display the objects at the current moment, but it does not read any further. This means that the length of the capture has no effect on memory consumption during replay; the capture file can be arbitrarily long.

As previously mentioned, the state of an object is captured when it changes. This means that if a single attribute changes in a state of 100 attributes, then all 100 attributes would get captured again. This is wasteful. A concrete example where something similar is the case is a ‘drawing’. In *AVA360VR*, a drawing is a list of points in 3D space, each connected to the next point with a straight line. In other words, the drawing is a curve. (This description is not exactly the truth, but it is a useful simplification in this context.) When the drawing is long, it can contain more than a hundred points. Each point has 3 coordinates, so the state of the drawing can be relatively large. However, the list of points never changes after the drawing is created. Only the position and orientation may change but those may change several times a second. It is unnecessary to capture the list of points, every time the position or orientation changes. This is why we introduced ‘sub-states’. A sub-state is itself a state (for example its C# class has an *ApplyTo* method just like the main state). All captured objects have exactly one main state and zero or more sub-states. When splitting the attributes of an object into sub-states, the guiding principle is to group together attributes that usually change at the same time. For example, the list of points of the drawing is an attribute that never changes; we put it into the main state. But the position and the orientation usually change together so those are captured in one sub-state.

While capturing, we not only capture the state of objects around the user, but we also capture the controllers and the position and orientation of the IVR headset. We do this so that the user who made the VolCap is visible during RePlay (Figure 1 – blue avatar head). To be more precise, only the controllers and an avatar head is visible, but this is often enough for the RePlayer to feel that the VolCapper is immersively present in that virtual space. The controller and head movement (just like body language) gives a lot of information that cannot be deduced from the recorded voice alone.

Note that we are saving object *states* and *sub-states*, but not *events* or *state transitions*. In theory, this allows us to jump to a random time in the replay without having to 're-enact' all events leading up to that point. However, there is difficulty with this: the state of an object is only captured when it changes. To give an example the object 'A' changes its state 5 seconds into the replay but then never again and the replay is two hours long and new states get captured almost every second. Then, imagine that during replay we want to jump to the one-hour mark. How would the program know where to look for the most recent state of object 'A'? One possible solution is to capture the objects' state every few minutes to ensure that the replay system never has to look too far to find the most recent state of all objects.

Conclusion and future work

IVR is seen as a new technology for supporting collaborative learning (Wise & Schwarz, 2017), and researchers are searching for new ways to gather different types of data from IVR, e.g. biosensing data, tracking of controllers, etc.. The simplest thing to do is to record the screen that shows a 2D rendered version of what the participant in IVR is seeing and doing, but that is perhaps the least innovative way to capture and analyze data collected from collaborative activities in IVR. Indeed, the output on the flat screen is much less immersive than the experience of being in IVR. It is important to understand that a VolCap is not a video. A video is a rendering of an event in flat pixels on a 2D screen. However, much of human life is sensed volumetrically. A rendering of a VolCap also generates pixels, but, in contrast to flat video, one has the possibility to move around the rendered spatial environment and view it from all possible angles.

Based on our practical experiences of *AVA360VR*, we propose that VolCap is an important alternative that enhances a more qualitatively oriented type of computer-supported analysis in CSCL. With the help of VolCaps, the analysis is archived and can be re-activated in the future by another researcher or be used to involve practitioners in the analysis of a CSCL activities. This is a completely new and innovative way of presenting and disseminating research in CSCL because the viewer can now follow and re-enact the analytical process in each and every detail.

In the future, we plan to enhance VolCap to allow users to rewind and fast-forward a RePlay and most importantly capture VolCaps with multiple virtual users. Our vision is to establish an immersive collaborative VolCap that allows multiple researchers and practitioners to create as well as re-activate their analyses at a later date. Furthermore, we hope to make it possible to create a VolCap of a VolCap, which would afford recursive volumetric commentary.

References

Jeong, H., Hmelo-Silver, C. E., & Yu, Y. (2014). An examination of CSCL methodological practices and the influence of theoretical frameworks 2005-2009. *International Journal of Computer-Supported Collaborative Learning*, *9*(3), 305–334.

Martinez-Maldonado, R., Dimitriadis, Y., Martinez-Monés, A., Kay, J., & Yacef, K. (2013). Capturing and analyzing verbal and physical collaborative learning interactions at an enriched interactive tabletop. *International Journal of Computer-Supported Collaborative Learning*, *8*(4), 455–485.

McIlvenny, P. (2018). Inhabiting spatial video and audio data: Towards a scenographic turn in the analysis of social interaction. *Social Interaction. Video-Based Studies of Human Sociality*, *2*(1).

McIlvenny, P. (2020). The future of 'video' in video-based qualitative research is not 'dumb' flat pixels! Exploring volumetric performance capture and immersive performative replay. *Qualitative Research*, *20*(6), 800-818

McIlvenny, P., & Davidsen, J. (2017). A Big Video manifesto: Re-sensing video and audio. *Nordicom Information*, *39*(2), 15–21.

McIlvenny, P., & Davidsen, J. (Accepted). Beyond video: Using practice-based VolCap analysis to understand analytical practices volumetrically. In P. Haddington (Ed.), *Methodological explorations in and for EMCA: Emerging directions for the study of social order*. Routledge.

Schneider, B., Sharma, K., Cuendet, S., Zufferey, G., Dillenbourg, P., & Pea, R. (2018). Leveraging mobile eye-trackers to capture joint visual attention in co-located collaborative learning groups. *International Journal of Computer-Supported Collaborative Learning*, *13*(3), 241–261.

Steier, R., & Davidsen, J. G. (2021). Adapting Interaction Analysis to CSCL: a systematic review. In C. E. Hmelo-Silver, B. De Wever, & J. Oshima (Eds.), *14th International Conference on Computer-Supported Collaborative Learning (CSCL)* (pp. 157–160). International Society of the Learning Sciences (ISLS).

Wise, A. F., & Schwarz, B. B. (2017). Visions of CSCL: Eight provocations for the future of the field. *International Journal of Computer-Supported Collaborative Learning*, *12*(4), 423–467.