uvgRTP 3.0: Towards V3C Volumetric Video Communication

Heikki Tampio, Joni Räsänen, Marko Viitanen,
Alexandre Mercat, Guillaume Gautier, and Jarno Vanne
Ultra Video Group, Tampere University, Tampere, Finland
{heikki.tampio, joni.rasanen, marko.viitanen, alexandre.mercat, guillaume.gautier, jarno.vanne}@tuni.fi

ABSTRACT

Low-latency volumetric video transport is a key enabling technology for more immersive communication applications. This paper presents the latest release of our open-source Real-time Transport Protocol (RTP) library called uvgRTP 3.0 that has been upgraded to support Visual Volumetric Video-based Coding (V3C) transmission in Video-based Point Cloud Compression (V-PCC) and MPEG Immersive Video (MIV) formats. uvgRTP 3.0 introduces 1) the V3C atlas RTP payload format; 2) two multiplexing methods to reduce port reservations during V3C transmission; and 3) improved packet reception with multithreading. Our performance results show that uvgRTP 3.0 can encrypt and transmit V3C bitstreams at 106 Mbit/s, with CPU core utilization of 26%, and achieving a round-trip latency of 2 ms in a local area network. The support for high-speed, encrypted V3C communication with the permissive BSD-license make uvgRTP 3.0 a potential transmission library for any industrial or academic volumetric communication system.

CCS CONCEPTS

• Networks ~ Network protocols ~ Transport protocols

KEYWORDS

Real-time Transport Protocol (RTP), Point cloud, Visual Volumetric Video-based Coding (V3C), Video-based Point Cloud Compression (V-PCC), MPEG Immersive Video (MIV), Extended Reality (XR)

ACM Reference format:

Heikki Tampio, Joni Rasanen, Marko Viitanen, Alexandre Mercat, Guillaume Gautier, and Jarno Vanne. 2024. uvgRTP 3.0: Towards V3C Volumetric Video Communication. In *15th ACM Multimedia Systems Conference (MMSys '24), April 15–18, 2024, Bari, Italy. ACM, New York, NY, USA, 7 pages.* https://doi.org/10.1145/3625468.3652185

MMSys '24, April 15–18, 2024, Bari, Italy

ACM ISBN 979-8-4007-0412-3/24/04.
https://doi.org/10.1145/3625468.3652185

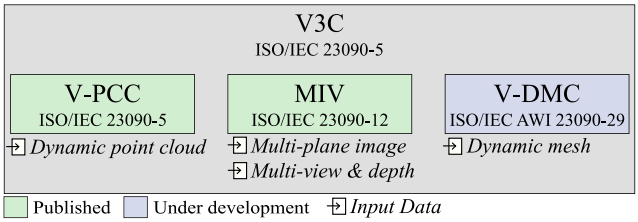

Figure 1: Overview of the V3C standard suite.

1 Introduction

Visual volumetric media technologies lay the foundation for deploying *extended reality* (*XR*) in applications like live communication, remote supervision, and remote piloting. However, this implies an order of magnitude higher data rates which challenges communication applications to meet the required transmission speed with low latency.

The *Moving Picture Experts Group* (*MPEG*) has addressed the transmission and storage needs of volumetric visual data by introducing the *Visual Volumetric Video-based Coding* (*V3C*) standard suite, specified as ISO/IEC 23090-5 [1]. As shown in Figure 1, V3C contains three standard specifications called *Video-based Point Cloud Compression* (*V-PCC*) [1], *MPEG Immersive Video* (*MIV*) [2], and *Video-based Dynamic Mesh Compression* (*V-DMC*) [3], of which *V-PCC* and *MIV* have already been published. V-PCC is designed for dynamic point clouds, MIV for multi-plane images as well as for multi-view and depth videos, and V-DMC for dynamic meshes. All of them are built using established 2D video compression techniques while conforming to the common V3C bitstream structure.

In ISO/IEC 23090-10 [4], MPEG specifies a container format for V3C that can be used by *Dynamic Adaptive Streaming over HTTP* (*DASH*) for carriage. However, the transmission delay of DASH is not ideal for applications with stringent latency requirements. Instead, *Real-time Transport Protocol* (*RTP*), specified in *Request for Comments* (*RFC*) 3550 [5], is commonly used for low-latency streaming, such as communication. An RFC draft [6] describes how V3C content should be streamed using RTP.

uvgRTP [7] is our low-latency open-source RTP streaming library that supports a wide range of codec-specific payload formats, such as *Advanced Video Coding* (*AVC*) [8], *High*

Table 1: V3C unit types, their sub-bitstream types and use in standardized V3C codecs.

V3C unit type	Sub-bitstream type	V-PCC	MIV
V3C Parameter Set (VPS)	-	Yes	Yes
Atlas Data (AD)	Atlas	Yes	Yes
Occupancy Video Data (OVD)	Video	Yes	Yes
Geometry Video Data (GVD)	Video	Yes	Yes
Attribute Video Data (AVD)	Video	Yes	Yes
Common Atlas Data (CAD)	Atlas	No	Yes
Packed Video Data (PVD)	Video	No	Yes

Efficiency Video Coding (*HEVC*) [9], and *Versatile Video Coding* (*VVC*) [10]. In its third major release, uvgRTP 3.0 becomes the first open-source RTP library to support low latency transmission of V3C bitstreams over the network, making it a promising component for future volumetric video communication applications. For practical and efficient V3C streaming, the following features have been implemented:

- support for the V3C atlas RTP payload format;
- new multiplexing methods to reduce port reservations during V3C transmission;
- introduction of multithreading to enhance the packet reception process.

Furthermore, IPv6 compatibility was introduced to meet modern networking requirements. The uvgRTP library is available at:

https://github.com/ultravideo/uvgRTP

The remainder of the paper is structured as follows. Section 2 provides an overview of the V3C bitstream structure, describes V3C transmission over RTP, and goes over existing open-source libraries for RTP streaming. Section 3 presents our uvgRTP 3.0 library by focusing on its new features designed for low-latency V3C streaming, Section 4 reports the performance results, and Section 5 addresses the usage and applications of the uvgRTP library. Finally, Section 6 discusses the limitations of V3C bitstream formats in real-time applications and Section 7 concludes the paper.

2 Background and Related Works

2.1 V3C Bitstream Formats

The V3C standard [1] defines two V3C bitstream formats: *V3C unit stream* and *V3C sample stream*. Both of them can use up to seven V3C unit types. Table 1 tabulates these unit types along with their sub-bitstream types and use in standardized V3C codecs. All V3C units contain a V3C unit header that specifies the unit type. All sub-bitstream types are composed of *Network Abstraction Layer* (*NAL*) units. Video sub-bitstreams contain video data encoded with an AVC [11], HEVC [12], or VVC [13] video codec, whereas atlas sub-bitstreams contain atlas data.

The *V3C unit stream* [1] overall structure is illustrated in Figure 2. It contains V3C units segmented into groups called

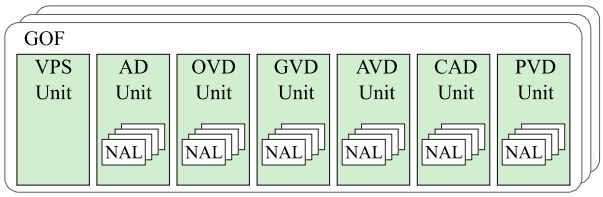

Figure 2: V3C unit stream structure.

group of frames (*GOF*). A single GOF contains one V3C unit of each type required by the codec. The V3C units can contain data to reconstruct one or more frames, depending on the *GOF size*. The *V3C unit stream* cannot be used in applications without any additional parsing information.

A *V3C sample stream* [1] is a *V3C unit stream*, where each V3C unit is prepended by its size. A *V3C sample stream* header is added at the beginning of the bitstream, specifying the number of bytes used to represent the V3C unit sizes. The *V3C sample stream* is the format used by V3C codec test models.

2.2 V3C Transmission over RTP

In V3C bitstream transmission, the first step is to parse it into V3C units. Then, the sub-bitstreams inside the V3C units are parsed into NAL units, which are then transmitted via RTP. *V3C Parameter Set* (*VPS*) and the information contained in V3C unit headers are conveyed outside of RTP transmission via other means, such as *Session Description Protocol* (*SDP*) [14].

For video sub-bitstreams, existing RTP payload formats, AVC [8], HEVC [9], or VVC [10], are used. For atlas sub-bitstreams, a V3C atlas RTP payload format has been defined in an RFC draft [6]. The draft specifies rules for packetization of NAL units in atlas sub-bitstreams. To avoid *Internet Protocol* (*IP*) layer fragmentation, the fragmentation process is specified for NAL units larger than the *maximum transmission unit* (*MTU*) of the network. In addition, the draft includes instructions for mapping the information from VPS and V3C unit headers into SDP messages.

2.3 Existing RTP Streaming Libraries

Table 2 characterizes the existing open-source RTP libraries based on their support for video and atlas RTP payload formats. None of them natively support the V3C atlas RTP payload format, which is a prerequisite for V3C transmission over RTP, together with at least one of the tabulated video RTP payload formats.

Nokia Technologies has created a plugin [19] for GStreamer, which takes V3C bitstreams packaged into *ISO base media file format* (*ISOBMFF*) [20] as input and outputs RTP packets. However, there is currently no open-source tools available for packaging V3C bitstreams into the ISOBMFF format. Moreover, ISOBMFF is a storage format and is inherently not designed for low-latency communication. As described next, our uvgRTP 3.0 library is developed for real-time V3C transmission.

Table 2: Existing open-source RTP libraries and their natively supported RTP payload formats.

[Ref.] Library	V3C Atlas	Video			License
		VVC	HEVC	AVC	
[15] PJSIP	No	No	No	Yes	GPL-2.0
[16] LIVE555*	No	No	Yes	Yes	LGPLv3.0
[17] FFmpeg	No	No	Yes	Yes	LGPLv2.1
[18] GStreamer	No	No	Yes	Yes	LGPLv2.1
[7] uvgRTP 2.0	No	Yes	Yes	Yes	BSD-2
uvgRTP 3.0	**Yes**	**Yes**	**Yes**	**Yes**	**BSD-2**

**LIVE555 implements the older RTP standard RFC 1889*

3 V3C Communication with uvgRTP 3.0

3.1 Updated Software Architecture

Figure 3 illustrates the software architecture of uvgRTP 3.0 that is upgraded from uvgRTP 2.0 by adding a new module (in red) and updating three modules (in green). *Context* is a singleton module created by a *V3C application*. It contains one or more *sessions*, whereas a *session* represents connections between two IP addresses. *Session* may contain multiple *media streamers* through which media is transmitted. An *RTP Control Protocol* (*RTCP*) module is associated with each *media streamer*. It monitors the data delivery and provides feedback on the *quality of service* (*QoS*) of the media stream.

A new *Socket factory* module manages *network sockets* with support for both IPv4 and the newly added IPv6. The *packet reception* module reads incoming packets from *network socket* and distributes them for processing in other modules. The *Secure RTP* (*SRTP*) and *RTP* modules process encrypted and unencrypted packets, respectively. Meanwhile, the *Zimmermann RTP* (*ZRTP*) module handles automatic key exchange for encryption. Finally, the *RTP payload formats* module handles RTP payload format specific operations.

When streaming between two *V3C applications*, both of them create a *session* module for connections, as depicted in Figure 4. A separate *media streamer* is applied for each type of V3C unit with RTP payload. In addition, another *media streamer* is created for audio. Outside of the uvgRTP library, *V3C applications* can use SDP for transmission of VPS and V3C unit header information.

3.2 Extended RTP Payload Format Support

The *RTP payload formats* module takes care of operations specific to the chosen RTP payload format. It can support various video RTP payload formats such as AVC, HEVC, or VVC. uvgRTP 3.0 extends it with support for the V3C atlas RTP payload format. In addition, the generic RTP payload format can be used for other types of RTP payload that do not require any format-specific features.

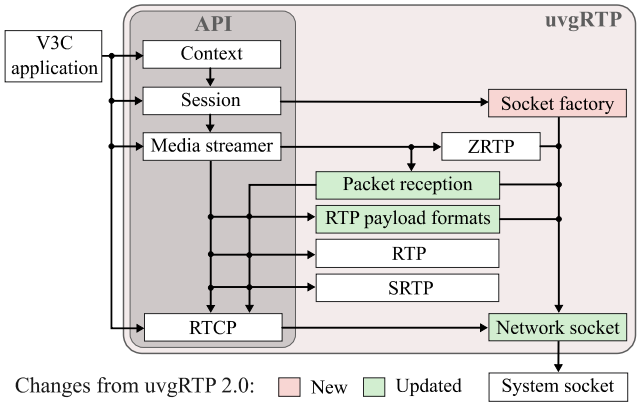

Figure 3: Software architecture of uvgRTP 3.0.

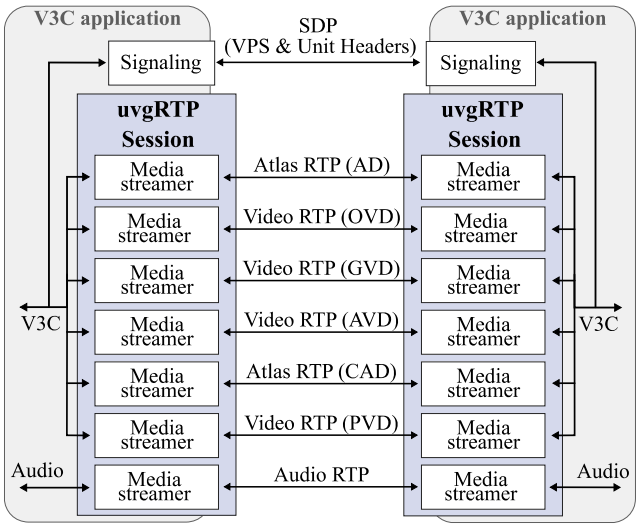

Figure 4: uvgRTP 3.0 *session* structure for streaming V3C.

3.3 Port Multiplexing and Demultiplexing

Transporting a V3C bitstream with the uvgRTP library requires multiple *media streamers*. uvgRTP 2.0 required one network port for each *media streamer* and another for each associated *RTCP module*, so *V3C application* would have needed to reserve up to 14 ports for each *session*. This increases the probability of port collisions with other applications. Furthermore, it raises the complexity of required firewall and router configurations. To that end, uvgRTP 3.0 introduces two multiplexing techniques to minimize the number of ports required: 1) *synchronization source* (*SSRC*) multiplexing [21] and 2) RTCP to RTP port multiplexing [22].

Using SSRC multiplexing, as described in RFC 8872 [21], all *media streamers* within one *session* share a single *network socket* module for all transmissions. This *network socket* module binds to a specific port number. With multiple peers, the *socket factory* module distributes the *network sockets* for different *sessions*. Packets are forwarded to correct *media*

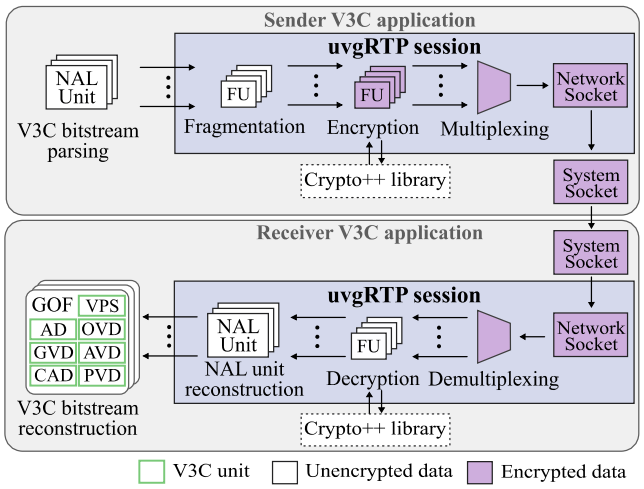

Figure 5: uvgRTP 3.0 end-to-end workflow for encrypted transmission of V3C bitstreams.

streamers based on a unique 32-bit SSRC identifier. The SSRC values of *media streamers* are communicated outside of the uvgRTP library, for example via SDP. For sent packets, the SSRC field in the RTP packet header is set to the SSRC identifier of the *media streamer* module. Packets from multiple *media streamer* modules are then multiplexed together and transmitted through a single port. On the receiving end, the *packet reception* module uses the SSRC identifier in the headers of received packets to distribute them to corresponding *media streamers*.

In addition, the usage of RTCP to RTP port multiplexing, as specified in RFC 5761 [22], enables the use of the same port for both RTCP and RTP transmissions, with RTCP and RTP packets being demultiplexed based on packet header structure. This technique is compatible with SSRC multiplexing, as RTCP packet headers also include the SSRC field.

3.4 Multithreaded Packet Reception

In uvgRTP 2.0, a single thread handles reading and processing of received packets, causing reliability issues and increased latency. To address this, uvgRTP 3.0 improves the reception process with multithreading. One thread is dedicated to reading packets from a *network socket* module and another one processes them. These threads communicate through a ring buffer, the size of which can be configured by the user. Multithreading in packet reception improves the processing capacity and reduces latency with large bitstreams.

3.5 End-to-end Workflow

Figure 5 illustrates the end-to-end workflow of uvgRTP 3.0 for encrypted transmission of a V3C bitstream. The *sender V3C application* gives NAL units as input to the *media streamer* modules inside a *session*. Fragmentation is performed for large NAL units by the *RTP payload formats* module. This produces smaller *fragmentation units* (*FU*) for transportation. Encryption is carried out using the Crypto++ library [23], and

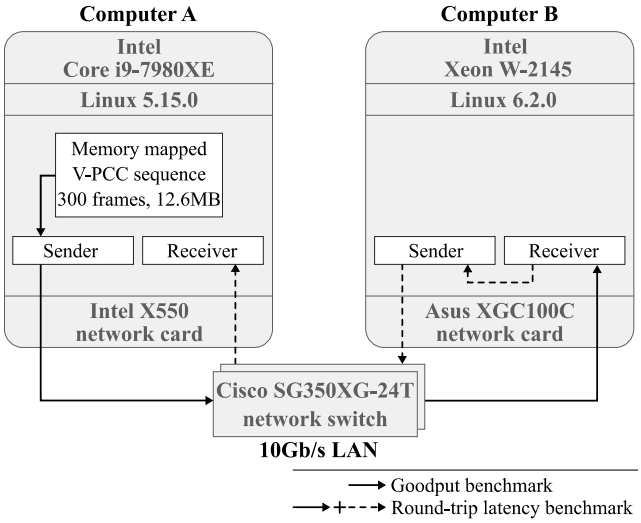

Figure 6: Experimental Setup.

Table 4: Test sequence characteristics.

Snapshot	Characteristics	
	Sequence name	*longdress*
	Number of frames	300
	Frame rate	30 fps
	Average number of points per frame	~ 800 000 pts
	Bitrate	10.1 Mbit/s
	V3C bitstream size	13 204 201 bytes
	RTP payload size	13 191 700 bytes

resulting units are multiplexed and sent through the *network socket*. At the receiving end, these operations are reversed and uvgRTP 3.0 outputs NAL units to *receiver V3C application* for V3C bitstream reconstruction.

4 Experimental Setup and Results

4.1 Experimental Setup

uvgRTP 3.0 was benchmarked by transmitting an encoded V-PCC bitstream between two desktop computers, as illustrated in Figure 6. The computers were equipped with Intel Core i9-7980XE and Intel Xeon W-2145 processors along with Linux kernel versions 5.15.0 and 6.2.0, respectively. The network consisted of a 10-Gbit Cisco SG350XG switch between the 10-Gbit network cards in both computers.

The benchmarks were performed using the *longdress* sequence from *common test conditions* (*CTC*) for V3C and V-PCC [24]. Table 4 tabulates the sequence characteristics. It was encoded using the reference V-PCC encoder *test model category 2* (*TMC2*) version 21 [25] with low-delay configuration at the rate *R3* as defined in CTC [24]. The V3C bitstream includes parts that are not transmitted with RTP, such as V3C VPS units or fields denoting V3C or NAL unit sizes.

These parts were subtracted from the total bitstream to obtain the total size of transmitted RTP payloads.

As our V-PCC test bitstream contained V3C units of types AD, OVD, GVD, and AVD, four uvgRTP *media streamer* modules were created in the *session* module: one for the atlas payload and one for each video payload. The traffic of all *media streamers* was multiplexed through a single *network socket*. Although the tests were performed only with V-PCC bitstreams, similar results can be expected for other V3C compliant bitstreams like MIV. Indeed, RTP streaming performance is only affected by the bitrate, regardless the type of payload.

Figure 6 also introduces the two separate benchmarks used to evaluate our solution. In the goodput benchmark, the CPU core utilization values of the sender and receiver were measured at different bitrates. The bitrate was virtually increased by sending *n* times the same V-PCC bitstream with 30 fps increments. The structure of the *session* remained the same. In a peer-to-peer conferencing use case, *n* represents the number of participants in a call at 30 fps. The round-trip latency benchmark measured round-trip latency of V3C streaming at a constant bitrate of 10.1 Mbit/s. Both encrypted SRTP and unencrypted RTP streaming modes for V3C were tested. All tests were performed 100 times, and the results were averaged.

4.2 Performance Evaluation

Figure 7 illustrates the results of the goodput benchmark. In addition, results of the average round-trip latency benchmark are the following:

- RTP streaming mode: 1.7 ms
- SRTP streaming mode: 2.2 ms

In both RTP and SRTP modes, the goodput benchmark shows that uvgRTP 3.0 is able to stream the V-PCC sequence at a bitrate of 105.9 Mbit/s. At this bitrate, the CPU core utilization values of the sender and receiver averaged at 18.9% and 5.0% for RTP streaming and 26.2% and 5.4% for SRTP streaming, respectively. These results amount to peer-to-peer V-PCC two-way communication at 30 fps with 10 other peers simultaneously.

To conclude, our results show that the efficient CPU core utilization of uvgRTP 3.0 leaves computing power for other processes, such as V3C encoding and decoding. Moreover, the low round-trip latency addresses the needs of low-latency communication in various volumetric video applications.

5 Practical Aspects of uvgRTP 3.0

5.1 Usage and Examples

The uvgRTP repository [26] on GitHub has several resources to facilitate the adoption of the library for streaming applications. Instructions for building uvgRTP 3.0 with CMake [27] can be found in `/BUILDING.md`. A step-by-step tutorial and more in-depth documentation can be found in `/USAGE.md` and `/docs/README.md`, respectively. The uvgRTP library also provides a documentation for its public API, which has been generated using Doxygen [28] and is hosted on GitHub [29]. Additionally, the `/examples` folder contains programs that demonstrate different usage scenarios of the uvgRTP library. Instructions for running these programs are given in `/examples/README.md`.

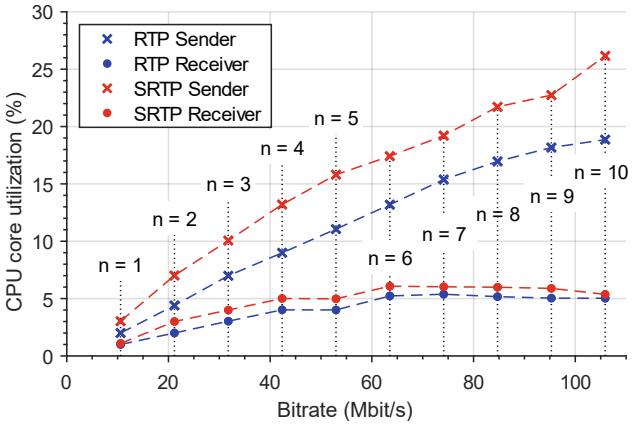

Figure 7: CPU core utilization of uvgRTP, *n* indicate the corresponding number of V-PCC streams at 30fps.

To help users develop applications built on V3C bitstream transmission, uvgRTP 3.0 includes a new example program, which demonstrates the implementation of an end-to-end workflow for transmitting a V3C bitstream. For straightforward testing, an encoded V-PCC test sequence is also provided alongside. This example program is composed of the following two parts:

1) *Sender V3C application* that takes as input a V-PCC bitstream and maps the locations and sizes of its NAL units into memory. One *session* is created for V3C transmission, in which *media streamers* are initialized for each type of sub-bitstream. When the program is running, *media streamers* send their corresponding data in parallel threads.
2) *Receiver V3C application* that creates one *session*, initializes the *media streamers* for sub-bitstreams, and begins reception. When a GOF is received, the respective part of the V3C bitstream is reconstructed. Finally, the GOFs are combined to form a complete V-PCC bitstream after the whole sequence is received.

5.2 Applications and Impact

After its release, uvgRTP 2.0 became a popular solution for real-time streaming thanks to features such as low latency, high performance, and ease of use. It has gathered multiple contributors and with a permissive BSD-license, the library is used by both industrial and academic applications. With the

introduction of V3C transmission in uvgRTP 3.0, solutions taking advantage of both the V-PCC and the MIV standards can be developed.

V-PCC streaming could be used as part of visual volumetric communication applications. In immersive video conferencing, users could interact as life-like virtual avatars. In addition to communication, the real-time V-PCC streaming would enable additional applications for remote object manipulation such as surgery, pottery, and art, where low latency is either an advantage or a prerequisite.

MIV streaming could be used for *virtual reality* (*VR*) applications such as virtual architectural tours. Each participant could view the scene in real-time from their own location and move around freely. Furthermore, the applications of MIV could include gaming, sports, and entertainment in visual volumetric format.

Thanks to visual volumetric content and more immersive applications, the low-latency streaming capabilities of uvgRTP 3.0 have the potential to reduce global plane travel by bringing forth new ways of accomplishing tasks remotely. As plane travel alone is responsible for 4% of total global warming [30], the impact of transporting bits instead of people can be significant.

6 Compatibility for Live Communication

As depicted in Figure 2, V3C units of the current V3C bitstream formats encapsulate the complete sub-bitstream of a GOF. *GOF size* specifies the number of frames in a single GOF, with 32 being the value used in CTC for V3C and V-PCC [24]. With *GOF size* of 32 and visual volumetric content recorded at 30 fps, the encoder introduces a latency of more than one second.

G.114 [31] defines latencies above 400 ms as generally unacceptable for audio communication, whereas user satisfaction starts to decline after 150 ms. Because the visual information needs to be synchronized with the audio conversation, visual latency should also be minimized to meet acceptable user satisfaction in live communication applications.

The encoder latency induced by the current V3C bitstream formats could be tackled by setting *GOF size* to 1, but it would lead to all intra coding with a significant coding overhead. This limitation motivates us to propose that this latency aspect be addressed in the upcoming updates to the V3C bitstream formats.

7 Conclusion

In this paper, we presented the latest version of our RTP library called uvgRTP 3.0. The introduced features make it the first open-source RTP library for low-latency transmission of V3C encoded visual volumetric content, such as V-PCC and MIV. Our measurements show that uvgRTP 3.0 can transfer an encrypted V3C bitstream at 106 Mbit/s with CPU core utilization of 26% and with a low round-trip latency of 2 ms. Given that our library is distributed under a permissive license and also supports popular AVC, HEVC and VVC formats, it is a potential candidate for a broad range of real-time communication applications.

For future research, a complete open-source end-to-end V3C pipeline still calls for practical solutions for volumetric video acquisition and coding. In addition to that, making V3C bitstream compatible with low-latency scenarios warrants further exploration. This work creates a solid foundation for these follow-up activities in unlocking the potential of low-latency V3C volumetric video communication.

ACKNOWLEDGMENTS

This work was supported in part by the AI-based Situational Awareness (AISA) project, the XR Simulation and Presence at the Cloud Edge (XR-SPACE) project both led by Nokia and funded by Business Finland, and the Research Council of Finland (decision no. 349216).

REFERENCES

[1] ISO/IEC 23090-5:2023, "Information technology — coded representation of immersive media — part 5: visual volumetric video-based coding (V3C) and video-based point cloud compression (V-PCC)," Nov. 2023.
[2] ISO/IEC 23090-12:2023, "Information technology — coded representation of immersive media — part 12: MPEG immersive video," Aug. 2023.
[3] ISO/IEC AWI 23090-29, "Information technology — coded representation of immersive media — part 29: video-based dynamic mesh compression (V-DMC)".
[4] ISO/IEC 23090-10:2022, "Information technology — coded representation of immersive media — part 10: Carriage of visual volumetric video-based coding data," May 2022.
[5] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "IETF RFC 3550 RTP: a transport protocol for real-time applications," IETF, Jul. 2003.
[6] L. Ilola and L. Konrad, "IETF RFC draft: RTP payload format for visual volumetric video-based coding (V3C)," Jul. 2023.
[7] J. Räsänen, A. Altonen, A. Mercat, and J. Vanne, "Open-source RTP library for end-to-end encrypted real-time video streaming applications," in Proc. *IEEE Int. Symp. Multimedia*, Naples, Italy, Nov.-Dec. 2021.
[8] IETF, "IETF RFC 6184 RTP payload format for H.264 video," May 2011.
[9] IETF, "IETF RFC 7798 RTP payload format for high efficiency video coding (HEVC)," Mar. 2016.
[10] IETF, "IETF RFC 9328 RTP payload format for versatile video coding (VVC)," Dec. 2021.
[11] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Aug. 2003.
[12] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.
[13] ITU "New 'Versatile Video Coding' standard to enable next-generation video compression," Sep. 2020.
[14] IETF, "IETF RFC 8866 SDP: session description protocol," Jan. 2021.
[15] PJSIP library. [Online]. Available: https://www.pjsip.org/
[16] LIVE555 library. [Online]. Available: http://www.live555.com/
[17] FFmpeg. [Online]. Available: https://www.ffmpeg.org/
[18] GStreamer. [Online]. Available: https://gstreamer.freedesktop.org
[19] Atlas data payloader and depayloader for RTP payload format for visual volumetric video-based coding (V3C). [Online]. Available: https://github.com/nokiatech/v3crtp
[20] ISO/IEC 14496-12:2022, "Information technology — coding of audio-visual objects — part 12: ISO base media file format," Jan. 2022.
[21] IETF, "IETF RFC 8872 guidelines for using the multiplexing features of RTP to support multiple media streams," Jan. 2021.
[22] IETF, "IETF RFC 5761 multiplexing RTP data and control packets on a single port," Apr. 2010.

[23] Crypto++. [Online]. Available: https://www.cryptopp.com/
[24] ISO/IEC JTC1/SC29/WG11, "Common test conditions for V3C and V-PCC," *document N19518*, Online, Jul. 2020.
[25] Test Model Category 2 Version 21. [Online]. Available: http://mpegx.int-evry.fr/software/MPEG/PCC/TM/mpeg-pcc-tmc2 (accessed Mar. 20, 2023).
[26] uvgRTP GitHub. [Online]. Available: https://github.com/ultravideo/uvgRTP
[27] Kitware, inc., CMake. [Online]. Available: https://cmake.org/
[28] Doxygen. [Online]. Available: https://www.doxygen.nl/
[29] uvgRTP Public API documentation. [Online]. Available: https://ultravideo.github.io/uvgRTP/html/index.html
[30] M. Klöwer, M. R. Allen, D. S. Lee, S. R. Proud, L. Gallagher, and A. Skowron, "Quantifying aviation's contribution to global warming," *Environmental Research Letters*, vol. 16, no. 10, 2021.
[31] ITU-T Recommendation G.114, "One-way transmission time," *Int. Telecommunication Union*, Online, Nov. 2003.