CWIPC-SXR: Point Cloud dynamic human dataset for Social XR

Ignacio Reimat, Evangelos Alexiou, Jack Jansen, Irene Viola, Shishir Subramanyam*
Centrum Wiskunde & Informatica (CWI)
Amsterdam, The Netherlands

Pablo Cesar
Centrum Wiskunde & Informatica (CWI)
Amsterdam, The Netherlands
TU Delft
Delft, The Netherlands

Figure 1: Sample frames from the point cloud sequences released with this dataset.

ABSTRACT

Real-time, immersive telecommunication systems are quickly becoming a reality, thanks to the advances in acquisition, transmission, and rendering technologies. Point clouds in particular serve as a promising representation in these type of systems, offering photorealistic rendering capabilities with low complexity. Further development of transmission, coding, and quality evaluation algorithms, though, is currently hindered by the lack of publicly available datasets that represent realistic scenarios of remote communication between people in real-time. In this paper, we release a dynamic point cloud dataset that depicts humans interacting in social XR settings. Using commodity hardware, we capture a total of 45 unique sequences, according to several use cases for social XR. As part of our release, we provide annotated raw material, resulting point cloud sequences, and an auxiliary software toolbox to acquire, process, encode, and visualize data, suitable for real-time applications. The dataset can be accessed via the following link: `https://www.dis.cwi.nl/cwipc-sxr-dataset/`.

CCS CONCEPTS

• **Computing methodologies** → **3D imaging**; *Point-based models*; *Virtual reality*; • **Information systems** → *Multimedia streaming*.

KEYWORDS

point cloud, volumetric videos, dynamic sequences, social XR

ACM Reference Format:
Ignacio Reimat, Evangelos Alexiou, Jack Jansen, Irene Viola, Shishir Subramanyam and Pablo Cesar. 2021. CWIPC-SXR: Point Cloud dynamic human dataset for Social XR. In *12th ACM Multimedia Systems Conference (MMSys '21) (MMSys 21), September 28-October 1, 2021, Istanbul, Turkey.* ACM, New York, NY, USA, 7 pages. `https://doi.org/10.1145/3458305.3478452`

*Please contact: nacho.reimat@cwi.nl

MMSys 21, September 28-October 1, 2021, Istanbul, Turkey

ACM ISBN 978-1-4503-8434-6/21/09.
`https://doi.org/10.1145/3458305.3478452`

1 INTRODUCTION

Photorealistic 3D (or volumetric) representations have recently gained a lot of momentum, due to their ability of reproducing real-life objects from multiple points of view in a lifelike manner. Point clouds in particular have attracted a lot of interest, due to the relative ease of acquisition and lack of connectivity information. These characteristics make them particularly suitable for real-time communication between remote users, which have been made possible by the latest advances of immersive technologies.

Despite the significant steps forward, the maturity of current tele-presence systems is lagging of traditional video conferencing tools. The vast amount of data that is required for volumetric representations poses several challenges, and demands efficient solutions in acquisition, compression, delivery and rendering, with academic teams, industrial markets and standardization bodies putting efforts to further develop relevant technologies. To this aim, the accessibility to real-life acquired datasets capturing social interactions among humans is of crucial importance, as they allow immediate evaluation of novel solutions in this context and establish a common ground for comparison purposes.

In recent years, volumetric human datasets have been acquired using sophisticated technology and complex setups. Commonly used technology for 3D or 4D registration includes multi-view camera configurations, such as the 8i Voxelized Surface Light Field (8iVSLF) dataset [8], the HUMBI dataset [16], the DFAUST [1], V-SENSE volumetric video quality database [17], HHI point cloud dataset of a boxing trainer [3], and the TotalCapture dataset [15], among others. The Panoptic dataset [7] is acquired by a massively multi-view system, composed of 10 Kinect cameras, along with VGA and HD cameras, for a total of more than 500 views. Moreover, 3D body pose is given alongside the raw data. The Berkeley Multimodal Human Action dataset [11] is captured using several systems simultaneously: an optical motion capture system, 4 multi-view stereo vision camera arrays, 2 Microsoft Kinect cameras, and 6 wireless accelerometers.

Despite the advent of affordable commercial devices for 3D acquisition, however, there is still a shortage of public datasets of dynamic humans acquired with commonly attainable hardware in easily replicable setups. The Microsoft Voxelized Upper Bodies [9] was acquired using 4 frontal RGB-D cameras. However, as the name suggests, it only depicts the upper body of the user, thus not offering a full body representation. The Human3.6M dataset [5] was acquired using video cameras, motion capture cameras, and time-of-flight sensors. The 11 actors were scanned using a 3-sensor 3D scanner and then animated using the acquired poses. The Human4D dataset [2] contains human activities simultaneously captured using motion capture and volumetric sensors. However, the strict requirements for accurate motion capture resulted in actors wearing uniform black clothing with colored markers, thus reducing the range of textures that are present. Moreover, the datasets presented above use older 3D sensing hardware; hence, the quality of the volumetric acquisition does not accurately match the possibilities offered by more recent depth-sensing equipment.

In this paper, we release the first dynamic point cloud dataset captured by several synchronized Azure Kinect DK devices[1], depicting humans performing common social activities in real-time communication scenarios. The primary goal of this dataset is to make available multi-modal recordings of humans interacting in social contexts through tele-presence systems, using off-the-shelf hardware that can be assembled at a reasonable cost. A secondary goal is to explore how a capturing system of this type operates in the real world, where occlusion, shiny surfaces or small and thin objects may confuse the sensors. To this aim, we focus on four key use cases of volumetric videos for social XR systems [4, 13], namely, "Education and Training", "Healthcare", "Communication and Social interactions", and "Performance and Sports". For each use case, a number of representative screenplays involving either 1 or 2 roles is devised, with every screenplay thoroughly scripted and executed once or more by different actors.

[1] https://azure.microsoft.com/en-us/services/kinect-dk/

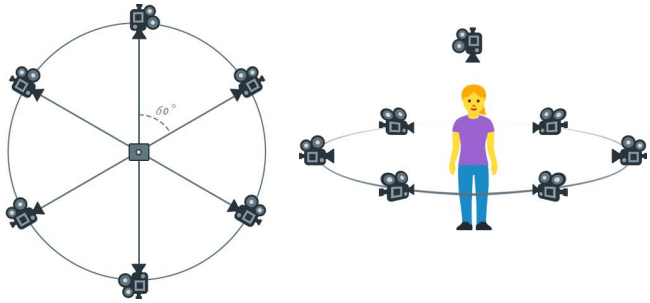

Figure 2: Geometrical camera arrangement, seen from the top (left) and from the front (right).

To facilitate research on adjacent computer vision communities, as part of our release, we additionally provide the annotated raw material that was recorded by our capturing system, which is suitable for testing point cloud registration and reconstruction techniques, as well as visual enhancement proposals for outlier removal, or filling missing regions. We conclude our contributions by making publicly available an auxiliary toolbox of software utilities built on top of the Point Cloud Library [14], that allows for alignment and processing of point clouds, as well as real-time capturing, encoding, transmission, and rendering.

2 CAPTURING SETUP

For this dataset filming, a custom room setting with a recording area of size 280x280cm was employed. The recording scene was illuminated by 2 lamps placed diametrically opposed, and each lamp was equipped with 2 light bulbs of model Bresser BR-JDD-6 45W and color temperature 5500K.

To capture color and depth data, we used 7 Azure Kinect DK sensors that were distributed around the recording area. In particular, a total of 6 devices were uniformly placed over a circle of radius 140cm, such that each successive pair differs by an angle of 60°, and were mounted on tripods at a height of 140cm. The 7th device was placed at the center of this circle and at a height of 242cm using an articulated arm, oriented perpendicular to the floor in order to capture top views of the recording scene. A snapshot illustrating the camera layout is depicted in Figure 2, while the recording area with the physical arrangement of cameras can be seen in Figure 3.

The Azure Kinect DK is equipped with an RGB camera, a depth sensor, and an IR emitter in addition to other hardware, such as a microphone array, a gyroscope and an accelerometer that are not utilized in our setup. The depth camera supports a number of modes, which determine the resolution, frame rate, exposure time, field of interest, and operating range of the device. Similarly, the modes of the color camera determine the resolution, aspect ratio, format, frame rate, and nominal field of view (FOV). Each device also includes dedicated ports in order to enable synchronous acquisition of a scene from multiple viewpoints.

Figure 3: Physical camera arrangement.

Table 1: Usage percentage per module for the host computer, while recording from a different number of Kinect Azure DK devices.

Devices	CPU (%)	RAM (%)	GPU (%)	SSD (%)
1	8	24	6	4
2	16	25	8	8
3	22	28	11	12
4	32	31	13	16
5	42	33	16	20
6	49	34	19	23
7	58	35	22	28

In our capturing setup, the devices were interconnected using 3.5-mm audio cables and synchronized following a "master" and "subordinates" architecture. Each Kinect was connected to a separate USB 3.0 port of a host computer. For some cameras the connection was direct, while for the more distant cameras an active USB 3.0 extender cable of 300cm length was additionally employed. The host computer was running Windows 10 Pro x64 on an Intel Core i7-7700K CPU, with an NVIDIA GeForce GTX 1080 Ti GPU, 32 GB of RAM, and 3 Samsung 960 EVO 500GB SSDs installed. The resource usage of each component while recording with a different number of cameras is indicated in Table 1.

To capture audio data, a lavalier microphone was attached to the actor's clothing and connected to a mobile phone, which was carried with him/her during acting.

3 SOFTWARE

Our group has developed an open source suite of libraries and tools, named `cwipc` (abbreviation of CWI Point Clouds), in order to facilitate working with point clouds as opaque objects, similarly to how most software works with images, or audio samples. The implementation builds on the Point Cloud Library and various vendor-specific capturing libraries, but this is transparent to software using the `cwipc` suite.

The idea behind a `cwipc` object is that it represents a point cloud as a collection of points with coordinates (x, y, z) and color values (r, g, b), with the possibility of carrying additional data such as the camera angle each point is captured from, global information such as the timestamp of the capture and the voxel size, as well as the original RGB and D images, or skeleton data. A `cwipc` object can be passed between modules without knowing what is inside it, and this can be done across implementation language boundaries while minimizing unnecessary memory copies. The library makes it possible to create end-to-end pipelines in order to capture, send, receive, and render dynamic point clouds [6]. It is suitable for real-time applications, and due to the vast amount of data that might be carried, special care is given to memory management in order to minimize the amount of copying needed.

The core of our suite is `cwipc_util`, which handles the `cwipc` object implementation, its memory management and the multiple language bindings (C, C++, Python and C#). It also contains utility functions to read and write `cwipc` objects in PLY format, and apply different filters and transformations to the `cwipc` objects. In addition, it contains a set of tools to align point clouds obtained from multiple cameras and optionally enable auxiliary tunable filters (`cwipc_calibrate`), a customized viewer to playback dynamic point clouds (`cwipc_view`), and a grabber tool that allows to grab point cloud frames from multiple devices or from offline prerecorded files (`cwipc_grab`). The suite also contains modules to capture point clouds from one or multiple Kinect and Realsense cameras (`cwipc_kinect` and `cwipc_realsense2`, respectively). Finally, the well-established codec described in [10] is provided as a module of our suite, in order to enable real-time compression and decompression (`cwipc_codec`) of dynamic content.

The library core is written mainly in C++, with most of the utilities written in Python. The suite is platform-independent and runs on Windows, Linux and MacOS (with the caveat that the Kinect module is not available for MacOS currently, because the underlying Kinect SDK is not available to this date). As of this writing, the `cwipc` suite is provided as source code that is released on GitHub[2]. This is in part due to the fact that there is no sufficiently popular package manager that can handle multi-platform as well as multi-language packages.

4 DATASET

To design the dataset, 4 key use cases for social XR were selected, namely, "Education and Training", "Healthcare", "Communication and Social interactions", and "Performance and Sports". The use cases form our categories, as listed in Table 2. For each of them, screenplays were carefully devised to be representative of the corresponding category, involving one or multiple roles for actors that interact with each other according to necessity. In all cases, a single actor was placed to the scene and captured, in order to ensure the highest possible quality for each individual 3D scan. To achieve synchronization across different actors in multi-person screenplays, the timing of each actor's activity was determined based on external audio signals.

[2] `https://github.com/cwi-dis/cwipc`

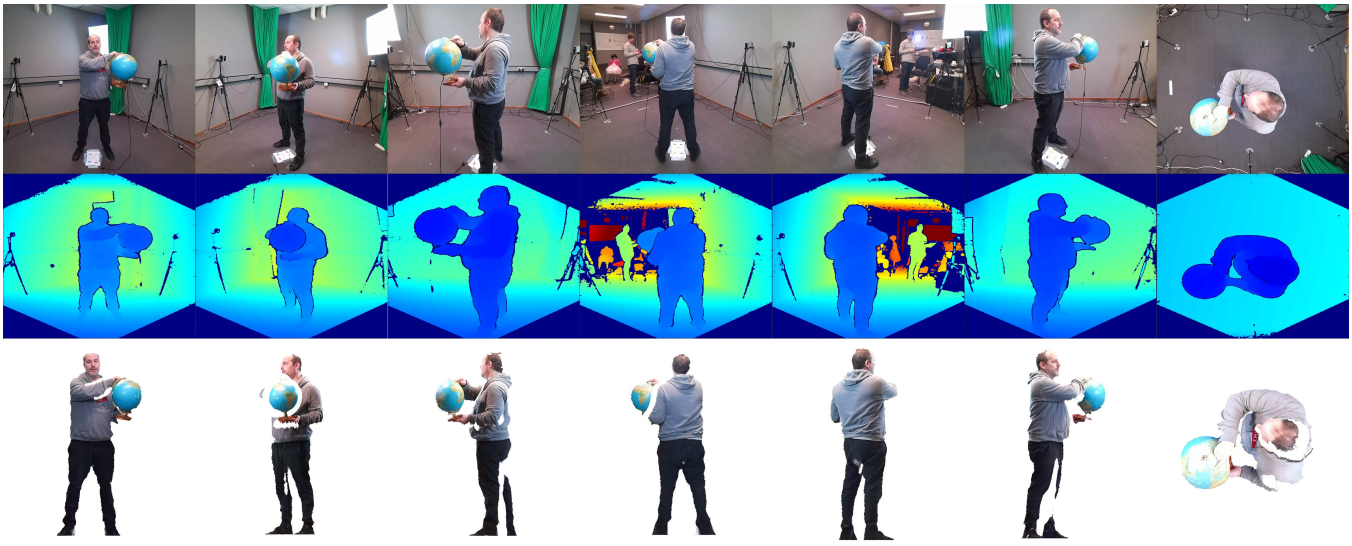

Figure 4: Illustration of RGB and Depth raw data captured by 7 Kinect Azure DK devices that comprise our capturing system, and corresponding point cloud frames that are generated offline.

Table 2: Dataset description: Every screenplay falls under a separate use case, involves one or more roles, and can be repeated in one or more takes. For every screenplay, the total number of point cloud frames is provided, and we annotate whether the audio was recorded in addition to the RGB-D data.

Use case	ID	Description	Roles	Takes	Total frames	Audio
Education and Training	01	Scientist explaining formula	1	1	1,177	✓
	02	Teacher with globe	1	1	1,117	✓
	03	Flight attendant	1	1	1,101	✗
	04	Padel coaching	2	1	1,744	✓
	05	Woman showing bottle	1	1	870	✓
	06	Book presentation	1	1	1,257	✓
	07	Woman wearing scarf	1	1	1,157	✗
Healthcare	08	How to wear face mask	1	1	985	✓
	09	Doctor with spine model	1	1	1,087	✓
	10	Arm injury	2	1	2,030	✓
Communication and Social interactions	11	Bollywood dialogue	2	1	2,768	✓
	12	Watching football	2	1	1,612	✓
	13	Card trick	2	1	2,536	✓
	14	Rock-paper-scissors in VR	2	1	1,926	✗
Performance and Sports	15	Basketball player	1	1	1,118	✗
	16	Acoustic guitar player	1	1	914	✓
	17	Electric guitar player	1	1	596	✓
	18	Boxer	1	1	1,087	✗
	19	Boxer in VR	1	1	1,118	✗
	20	Karateka	1	1	829	✓
	21	Dancing YMCA	19	1	various	✗

Every screenplay was thoroughly scripted to contain relevant motions and narrations or dialogues, while particular care was devoted in order to capture a reasonable range of variability between the actors, diverse materials, as well as kinematic complexity. That involved capturing multiple takes to cover all possible interactions between actors; for example, in screenplay 14 ("Rock-paper-scissors in VR"), sequences depicting all hand shapes are recorded for both

actors. Indicatively, challenging props, such as thin paper or cardboard, are used in screenplays 01 ("Scientist explaining formula") and 06 ("Book presentation"), while in screenplays 09 ("Doctor with spine model") and 16 ("Acoustic guitar player") objects with high degree of complexity for the sensor were part of the scene; that is, the composite geometry of the spine, and the reflective material of the guitar's pick guard. The final screenplay 21 ("Dancing YMCA") is provided in 19 takes, each involving a different actor, all synchronized to the audio cue to ease simultaneous playback.

In our recordings, a total of 23 individual actors were captured, with 3 of them participating in two takes, while 19 out of 23 subjects additionally performed the screenplay 21 ("Dancing YMCA"). The actors were dressed according to their character, with more daily clothing and relative diversity in colors and materials for more generic roles, to better simulate real life, as can be seen in Figure 1. The 15 out of 21 screenplays were written for a single role, whereas the remaining involve two-person interactions. From the total of actors, 9 are females, while the rest 14 are males. The duration of the videos is spanning between 20-50 seconds, while the corresponding number of point cloud frames that were extracted per sequence (i.e., a specific combination of screenplay, role, and take) ranges between 596 and 1, 384.

In the rest of this section, we provide a description of the system configuration for the recordings and the processing applied to the raw material to extract point clouds and synchronized audio files, as well as an outline of the structure of the released material.

4.1 Hardware and software settings

To record visual information, all 7 cameras of our camera arrangement were enabled. Following the selected synchronization architecture, a script that uses 7 instances of the `k4arecorder` tool provided by the Azure Kinect SDK[3] was created in order to launch the 6 subordinate cameras before initiating the master. A script example can be found among the released material.

The capturing system is suitable for real-time operation, but for the purposes of this dataset, we opted for recording RGB-D movies, which were subsequently post-processed to point clouds. In particular, we selected the NFOV_UNBINNED mode for the depth camera, and a resolution of 2048×1536 with aspect ratio of 4:3 for the color camera, which is preferred over 16:9 in NFOV mode for better pixel overlap[4]. Both cameras were set to a frame rate of 30 frames per second (fps), delivering approximately 480 Mbps of recorded depth and color data; that is, using all 7 devices, a total of $\sim$ 3.36 Gb is recorded per second. Note that the selected color resolution is the highest in 4:3 aspect ratio that supports 30 fps. The result of a recording is 7 video files, with each file containing 3 tracks for RGB, Depth, and Infrared data. An example of the recorded color and depth frames, along with the corresponding point clouds obtained can be seen in Figure 4.

The audio was recorded at 48 kHz. Audio and video streams were synchronized manually in post-production using the cues provided by a clapperboard. In particular, at the start of each take, the operator was entering in the recording scene holding a clapperboard in view of the cameras and was clapping the filmsticks shut, which was recorded by both the RGB-D and audio files.

Before generating the point clouds from the recorded RGB-D data we followed a two-step offline process for the cross-calibration of the devices. In the first step, a coarse calibration was performed using the `cwipc_calibrate` module of the software and the `a4floor` as target. This target was placed at the floor in the center of the recording area, as depicted in Figure 3, and is composed of 4 distinct color markers with known positions in an A4 paper. A manual selection of the markers in the frames captured by each camera was required. In the second step, an automatic refinement was executed using a cumulative multi-scale ICP algorithm based on the point-to-plane distance [12]. The result of this cross-calibration was a configuration file (`cameraconfig.xml`), which contains the extrinsic matrices for conversion of each individual's camera domain to a common coordinate system.

During point cloud generation, a cylindrical filter was enabled in order to discard regions of the scene that are irrelevant to the recorded actor, such as the background. Moreover, erosion was applied on the depth maps, in order to compensate for the mismatch between color and depth data that are captured from the Kinect sensors. The exact configurations were adjusted per content for higher reconstruction quality, and are reported in the `cameraconfig.xml` that is coming with every sequence of this dataset.

Using the computed transformations for camera alignment, the point cloud sequences were extracted from the prerecorded videos using the `cwipc_grab` module. In order to extract one, or more point cloud frames that correspond to a desirable part of the videos, starting and ending timestamps should be given as arguments. The exact inpoints and outpoints that were used per sequence, can be also found in the released material.

Note that the recorded video and audio files start earlier and end later than the provided point cloud sequences, in order to contain the clapperboard for audiovisual synchronization and the calibration target.

4.2 Dataset structure

The organization of the dataset is illustrated in Figure 5. For every screenplay, the relevant material is enclosed in a single folder, with sub-folders specifying the data collected for a different role and a different take. In every sub-folder, the `raw` folder contains the recorded RGB-D videos in MKV format and the audio files in WAV format, while the `ply` folder includes the generated point cloud sequences that are stored in PLY format in binary little-endian. For the latter, the timestamp of each individual frame, expressed as the elapsed time in microseconds since the start of the recording, is used as naming convention (e.g., `pointcloud-12345678.ply` with timestamp `12345678`, captured 12.345678 seconds from when the recording started). Sequence-specific configurations are provided in the `cameraconfig.xml`, such as the camera transformation matrices and the filtering parameters that were employed for point cloud frames extraction. Finally, generic information related to a particular sequence is given in the `sequenceinfo.txt`, including the duration and the fps of the video recordings, the timestamps in which the clapperboard shut is captured in the video and audio files, the input and output video timestamps that were employed

[3] https://github.com/microsoft/Azure-Kinect-Sensor-SDK
[4] https://docs.microsoft.com/en-us/azure/kinect-dk/hardware-specification

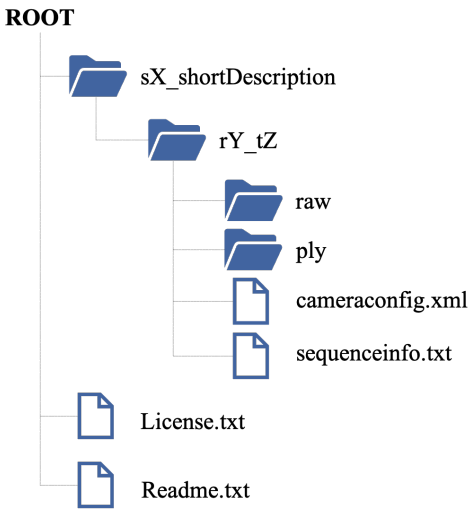

Figure 5: Organization of the dataset into folders (s: screenplay, r: role, t: take).

to extract point cloud frames that display only the actor's performance (i.e., excluding clapperboard), and the corresponding count of frames.

5 DISCUSSION

To record our dataset, we purposely selected consumer hardware, which was placed in an easily reproducible setup. Thus, our data mimics real-world acquisition mechanisms, which could be adopted by research labs and future consumers alike. State-of-the-art algorithms for compression, transmission, and quality evaluation of point clouds are currently tested on professionally-acquired data; in providing a dataset which exhibits the characteristics of user-generated content, we aim to contribute to the development of such algorithms beyond their current capabilities to embrace a larger, and consumer-ready, market. Furthermore, by providing synchronized dynamic human sequences involving more than one actor in social settings, we aim to promote development in research areas such real-time multi-party communication and user adaptation.

Using commodity depth-sensing devices comes with the limitations of the visual impairments generated in the acquisition, due to accuracy errors. Those become more visible when incorporating data from multiple cameras, due to the non-identical behaviour of the individual devices, the different scanning accuracy due to a different angle and/or distance between a common point captured by the FOV of multiple sensors, or even due to temporal discrepancies arising from the different shooting times of the cameras to avoid interference. An example is provided in Figure 6, where an actor is captured by one camera, three cameras with an offset of 120°, and all 7 cameras of our capturing setup. As can be seen, by increasing the number of cameras, the occluded regions are reduced at the expense of visual distortions. In particular, the regions captured by multiple cameras present texture impairments and outliers, due to overlapping scans. However, the benefit of using multiple sensors

Figure 6: An actor as captured by 1, 3, and 7 cameras, from left to right. In the top row, geometry and texture is presented, whereas the bottom row depicts the color-encoded geometry of the obtained point clouds from each camera.

can be seen in the geometric composition, with denser regions and reduced holes.

The above issues denote open problems under research for the broader computer vision community, which has allocated a substantial amount of efforts in proposing new methods for calibration, alignment, outlier detection, geometry and/or color smoothing. From this aspect, the raw-captured materials and the generated point clouds of the released dataset, may additionally serve as a benchmarking setup for testing novel post-processing solutions.

6 CONCLUSIONS

In this study, we record and release a volumetric video dataset that depicts humans under representative social activities over real-time tele-presence systems, with the recordings performed using consumer market acquisition devices to better resemble real-life settings. On top of the obtained point clouds, the raw captured data and a toolbox of software utilities covering the entire pipeline from real-time capturing to rendering of dynamic point clouds are additionally published. The provided material allows for custom point cloud generation, fosters development of real-time immersive media systems, and provides a basis for the integration and evaluation of new post-processing techniques.

REFERENCES

[1] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2017. Dynamic FAUST: Registering Human Bodies in Motion. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[2] Anargyros Chatzitofis, Leonidas Saroglou, Prodromos Boutis, Petros Drakoulis, Nikolaos Zioulis, Shishir Subramanyam, Bart Kevelham, Caecilia Charbonnier, Pablo Cesar, Dimitrios Zarpalas, et al. 2020. HUMAN4D: A Human-Centric Multimodal Dataset for Motions and Immersive Media. *IEEE Access* 8 (2020), 176241–176262.

[3] T Ebner, I Feldmann, O Schreer, P Kauff, and T v Unger. 2018. HHI point cloud dataset of a boxing trainer. *ISO/IEC JTC1/SC29 Joint WG11 (MPEG) input document m42921* (2018).

[4] Simon Gunkel, Hans Stokking, Martin Prins, Omar Niamut, Ernestasia Siahaan, and Pablo Cesar. 2018. Experiencing virtual reality together: Social VR use case study. In *Proceedings of the 2018 ACM International Conference on Interactive Experiences for TV and Online Video*. 233–238.

[5] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. 2014. Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36, 7 (jul 2014), 1325–1339.

[6] Jack Jansen, Shishir Subramanyam, Romain Bouqueau, Gianluca Cernigliaro, Marc Martos Cabré, Fernando Pérez, and Pablo Cesar. 2020. A Pipeline for Multiparty Volumetric Video Conferencing: Transmission of Point Clouds over Low Latency DASH. In *Proceedings of the 11th ACM Multimedia Systems Conference* (Istanbul, Turkey) *(MMSys '20)*. Association for Computing Machinery, New York, NY, USA, 341–344. https://doi.org/10.1145/3339825.3393578

[7] Hanbyul Joo, Tomas Simon, Xulong Li, Hao Liu, Lei Tan, Lin Gui, Sean Banerjee, Timothy Scott Godisart, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. 2017. Panoptic Studio: A Massively Multiview System for Social Interaction Capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

[8] M Krivokuća, PA Chou, and P Savill. 2018. 8i voxelized surface light field (8iVSLF) dataset. In *ISO/IEC JTC1/SC29/WG11 MPEG, input document m42914*.

[9] Charles Loop, Qin Cai, S Orts Escolano, and Philip A Chou. 2016. Microsoft voxelized upper bodies-a voxelized point cloud dataset. *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document m38673/M72012* (2016).

[10] Rufael Mekuria, Kees Blom, and Pablo Cesar. 2017. Design, Implementation, and Evaluation of a Point Cloud Codec for Tele-Immersive Video. *IEEE Transactions on Circuits and Systems for Video Technology* 27, 4 (2017), 828–842. https://doi.org/10.1109/TCSVT.2016.2543039

[11] Ferda Ofli, Rizwan Chaudhry, Gregorij Kurillo, René Vidal, and Ruzena Bajcsy. 2013. Berkeley mhad: A comprehensive multimodal human action database. In *2013 IEEE Workshop on Applications of Computer Vision (WACV)*. IEEE, 53–60.

[12] Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. 2017. Colored Point Cloud Registration Revisited. In *2017 IEEE International Conference on Computer Vision (ICCV)*. 143–152. https://doi.org/10.1109/ICCV.2017.25

[13] LLP Perkins Coie. 2020. Augmented And Virtual Reality Survey Report. 4 (2020), 2020.

[14] Radu Bogdan Rusu and Steve Cousins. 2011. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China.

[15] Matt Trumble, Andrew Gilbert, Charles Malleson, Adrian Hilton, and John Collomosse. 2017. Total Capture: 3D Human Pose Estimation Fusing Video and Inertial Sensors. In *2017 British Machine Vision Conference (BMVC)*.

[16] Zhixuan Yu, Jae Shin Yoon, In Kyu Lee, Prashanth Venkatesh, Jaesik Park, Jihun Yu, and Hyun Soo Park. 2020. HUMBI: A large multiview dataset of human body expressions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2990–3000.

[17] Emin Zerman, Pan Gao, Cagri Ozcinar, and Aljosa Smolic. 2019. Subjective and Objective Quality Assessment for Volumetric Video Compression. In *IS&T Electronic Imaging, Image Quality and System Performance XVI*.