Leveraging Enhanced Virtual Reality Methods and Environments for Efficient, Intuitive, and Immersive Teleoperation of Robots: Supplementary Materials

***Abstract*— This document describes supplementary materials and methods employed in the development of an Enhanced Virtual Reality (EVR) interface for robotic teleoperation. It details the framework architecture, focusing on the point cloud streaming and compression. It also comments on the visual artifacts appearing in the interface, discussing the possible implications.**

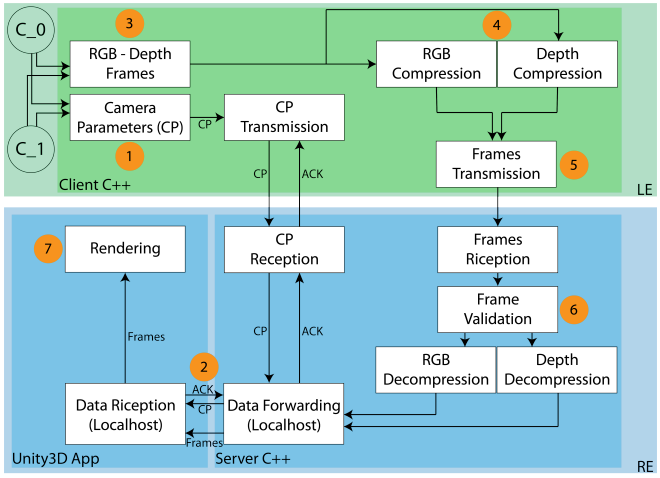

Fig. 1. The point cloud streaming. The frames captured by cameras C_0 and C_1 are streamed over the network to RE. After a validation process, they are decompressed and rendered in the Unity3D application.

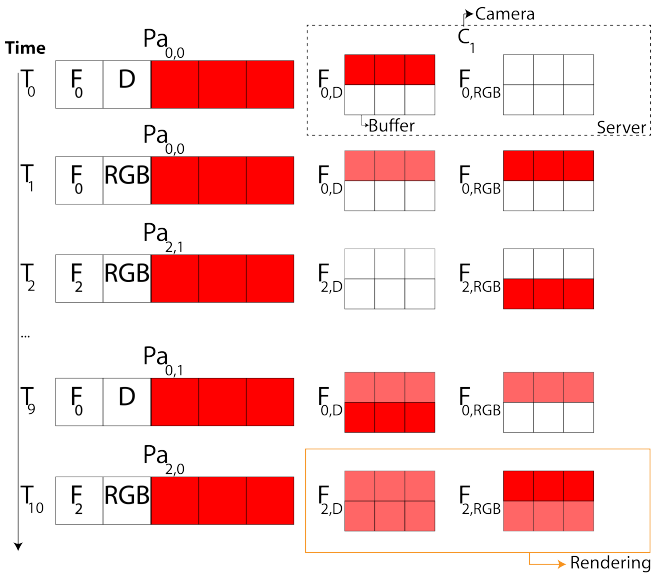

Fig. 2. The frames' validation procedure. As time passes, the frame buffers are randomly filled and only the frames that are full received (color and depth) are rendered, discarding the previous ones.

I. The Point Cloud Validation Procedure

This section adds some useful details regarding the point cloud streaming. Referring to Figure 3, LE sends the compressed cameras' frames over the network to RE. The frames are sent using the User Datagram Protocol (UDP) that provides fast but unreliable transmission. In fact, UDP does not ensure data reception nor data ordering, making the transmission fast but arduous to be handled. Hence, a frame validation procedure is applied to verify whether the frames have been fully received or not (Fig. 1 (6)).

Each frame is divided in packets of 1500 byte (maximum size), carrying an header containing: i) the packet type (RGB or depth), ii) the camera index, iii) the frame index, iv) the buffer index, and v) the compressed frame size that is used to decompress the frame at the receiving side. The remaining bytes are used to carry the payload, that is, the compressed frame data. Figure 2 depicts the frames validation check procedure (for the sake of clarity, only one camera is considered in this example but the proposed system supports multi-camera configuration). Let $Pa_{k,i}$ represent an i-th packet of frame k ($1 <= i <= L$, with L representing the number of frame packets and $0 <= k <= N$) with N representing the number of frames. Let C_n be the n-th camera, with $n >= 1$ and $F_{k,type}$ be the k-ith frame of type RGB or D (depth). At time T_0 a packet $Pa_{0,0}$ of frame $F_{0,D}$, camera C_1 is received by the server and it starts filling the corresponding buffer. Since UDP does not guarantee packet ordering, it is possible that at time T_1, the server receives another packet $Pa_{0,0}$ of the other frame $F_{0,RGB}$. As time goes by, the packets may arrive not in order and the frame buffers are filled in a non linear way (it is unlikely that the frames arrive ordered as the transmission side). Hence, it is not possible to determine in advance which frame will be fully collected, nor whether both color and depth frames can be entirely received. To overcome this limitation, only the frames that are fully received (color and depth) are considered for rendering and the previous frames are ignored and deleted. Referring to Figure 2, at time T_{10} all the buffers of frame F_2 are full. Hence, they will be rendered and all previous frames are ignored. This procedure has been generalized for a multi-camera configuration and only the frames that are fully received from all the cameras

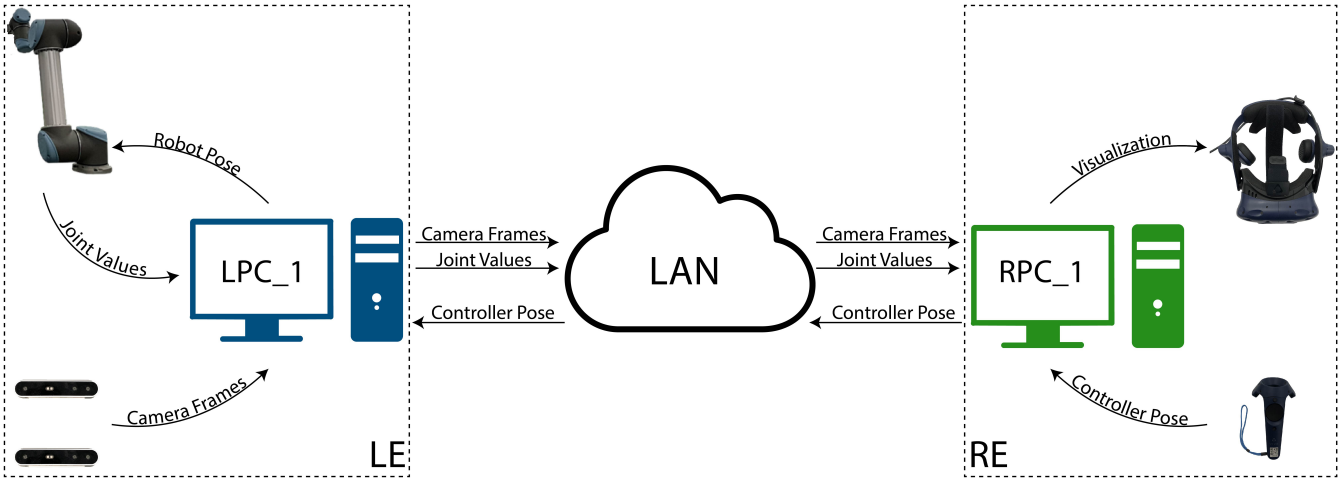

Fig. 3. Left-side: the local environment with the robot and the depth cameras. Right-side: the remote environment with the VR device.

TABLE I

THE FRAMES' SIZES IN BYTE FOR AN INTEL D415 CAMERA WITH A RESOLUTION OF 1280X720.

Raw Color (byte)	Raw Depth (byte)	Raw Frame (byte)	Raw Frame
2764800	1843200	4608000	4608000

TABLE II

THE COMPRESSED FRAMES. EACH LINE REPRESENTS A DIFFERENT COMPRESSED FRAME.

Compr. Color (byte)	Compr. Depth (byte)	Tot Compr. (byte)	Ratio Raw/Compr.
44967	466668	511635	9.006
44786	468980	513766	8.969
44761	466548	511309	9.012

are rendered. Therefore, it is ensured that distinct frames capturing the same scene from different viewpoints at the same time are rendered and visualized at the same moment.

A. The Point Cloud Compression Ratio

Hereby the point cloud compression ratio is reported to support the adopted compression strategy. Each raw frame (the union of color and depth frames) generated by an Intel RealSense D415 camera[1] carries a considerable amount of data, more than 4.6 MB (Table I). Considering a frame rate of 30fps, it means that each second more than 130MB of data should be transmitted over the network, exceeding the capacity of a standard 1 Gigabit Ethernet cable (125 MB). Hence, the color and depth frames have been compressed using the libjpeg-turbo library [1] and the approach proposed in [2], respectively. This strategy ensures high compression ration, allowing to send multi-camera frames at maximum resolution (1280x720) and frame rate (30 fps, see Table II).

[1]https://www.intelrealsense.com/depth-camera-d415/

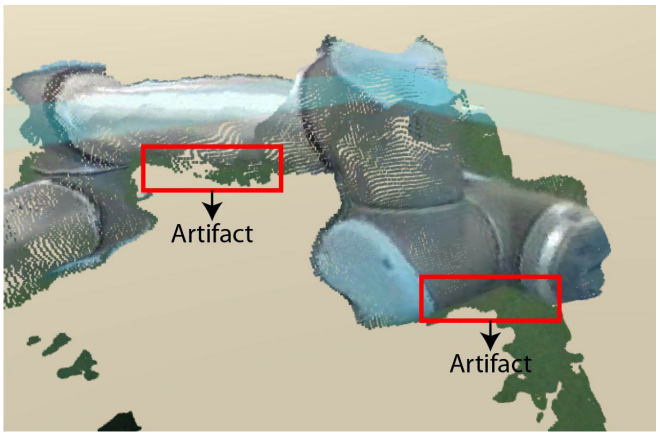

Fig. 4. **The red rectangles highlight the point cloud artifacts. The depth camera cannot clearly distinguish between a point lying on an edge (the robot edge) an one of the background (in green) and thus they are displayed together, negatively affecting the visualization on the environment.**

II. THE POINT CLOUD VISUAL ARTIFACTS

One of the main issues of I_{EVR} was related to the resolution of the point cloud. That is, it seems that a pure point cloud is not enough detailed to clearly visualised small/medium size objects. Specifically, it generates artifacts close to the objects' edges (Figure 4) that do not allow to clearly distinguish the object from the background. The 3D points placed very close to the edge are correctly colored using the color information of the background but mistakenly positioned using the depth information of the edge points, generating artifacts all around the objects. This limitation greatly affects the visualization of the robot end-effector, making really hard the robot control. Strategies should be pursued to improve the quality of the point cloud, making it less noisy. As an example, the objects' of interest (e.g., the robot arm) should be detected in the real environment and substituted in real-time with the corresponding CAD model (including textures).

REFERENCES

[1] "libjpeg-turbo." [Online]. Available: https://www.libjpeg-turbo.org/About/TurboJPEG

[2] A. D. Wilson, "Fast lossless depth image compression," in *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*, 2017, pp. 100–105.