Robotica (2024), **42**, pp. 4223–4248
doi:10.1017/S0263574724001784

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Immersive remote telerobotics: foveated unicasting and remote visualization for intuitive interaction

Yonas T. Tefera[1], Yaesol Kim[1], Sara Anastasi[3], Paolo Fiorini[2], Darwin G. Caldwell[1] and Nikhil Deshpande[1]

[1]Advanced Robotics, Istituto Italiano di Tecnologia (IIT), Genova, Italy
[2]Department of Computer Science, University of Verona, Verona, Italy
[3]Istituto Nazionale per l'Assicurazione contro gli Infortuni sul Lavoro (INAIL), Rome, Italy
Corresponding author: Yonas Teodros Tefera; Email: yonas.tefera@iit.it

Received: 14 December 2023; **Revised:** 27 August 2024; **Accepted:** 19 September 2024;
First published online: 30 October 2024

Keywords: telerobotics; telepresence; gaze tracking; mixed reality; 3D reconstruction; point clouds

Abstract

Precise and efficient performance in remote robotic teleoperation relies on intuitive interaction. This requires both accurate control actions and complete perception (vision, haptic, and other sensory feedback) of the remote environment. Especially in immersive remote teleoperation, the complete perception of remote environments in 3D allows operators to gain improved situational awareness. Color and Depth (RGB-D) cameras capture remote environments as dense 3D point clouds for real-time visualization. However, providing enough situational awareness needs fast, high-quality data transmission from acquisition to virtual reality rendering. Unfortunately, dense point-cloud data can suffer from network delays and limits, impacting the teleoperator's situational awareness. Understanding how the human eye works can help mitigate these challenges. This paper introduces a solution by implementing foveation, mimicking the human eye's focus by smartly sampling and rendering dense point clouds for an intuitive remote teleoperation interface. This provides high resolution in the user's central field, which gradually reduces toward the edges. However, this systematic visualization approach in the peripheral vision may benefit or risk losing information and burdening the user's cognitive load. This work investigates these advantages and drawbacks through an experimental study and describes the overall system, with its software, hardware, and communication framework. This will show significant enhancements in both latency and throughput, surpassing 60% and 40% improvements in both aspects when compared with state-of-the-art research works. A user study reveals that the framework has minimal impact on the user's visual quality of experience while helping to reduce the error rate significantly. Further, a 50% reduction in task execution time highlights the benefits of the proposed framework in immersive remote telerobotics applications.

1. Introduction

Remote teleoperation has emerged as a potentially critical solution to challenges in bridging geographical distances and bypassing physical limitations, thereby enabling seamless control, monitoring, and intervention in distant or inaccessible environments where direct human presence is unfeasible, unsafe, or impractical [1–3]. Teleoperation extends human reach beyond immediate physical boundaries, amplifies human capability, and provides accessibility to distant terrains and complex possibly hazardous scenarios. It has received increased interest in recent times due to the COVID-19 pandemic and the growth in immersive virtual reality (VR) interfaces that form a channel to present the remote environments to operators as dynamic 3D representations through real-time visualization [4, 5].

Despite the extensive research in controlling telerobotic systems through VR [4, 6–9], more research is needed regarding visual feedback of the remote environment, as this can significantly enhance and facilitate effective teleoperation. Immersive remote telerobotics, that is, the combination of VR and

© The Author(s), 2024. Published by Cambridge University Press.

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press

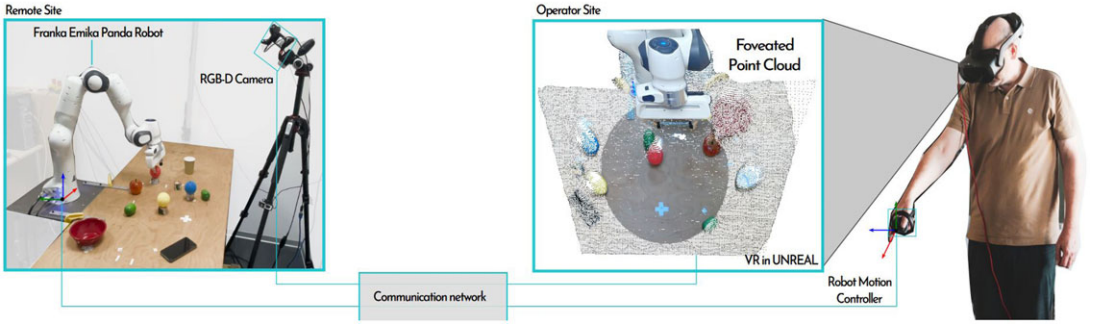


Figure 1. *Three-dimensional point partitioning – The surfel point* $\boldsymbol{P}(px, py, pz)$ *is classified using the ray* $\boldsymbol{L}$ *cast from the point of origin* $\boldsymbol{H}(hx, hy, hz)$.

actual 3D visual data from distant RGB-D cameras can allow real-time immersive visualization by the operator, simultaneously perceiving the color and the 3D profile of the remote scene [7, 10]. This can give operators enhanced situational awareness while maintaining their presence illusion [9]. This combination is the key distinguishing factor from traditional teleoperation interfaces, which rely on mono- or stereo-video feedback and suffer from limitations in terms of fixed or non-adaptable camera viewpoints, occluded views of the remote space, etc. [11]. Immersive remote teleoperation interfaces monitor user's gestures and movements in all six degrees of freedom (DOF), so that users can see the desired views, regardless of where they are located and where they are looking [12]. Nevertheless, real-time immersive remote teleportation has hard constraints regarding resolution, latency, throughput, compression methods, image acquisition, and the visual quality of the rendering [9, 13]. For instance, latency and low resolution have been shown to reduce the sense of presence and provoke cybersickness [14]. For many applications, including remote inspection and disaster response, these constraints are not trivial to overcome. They are further exacerbated by the fact that the environment may be a priori unknown and should be reconstructed in real-time from the 3D input data (e.g., depth maps, point clouds). Immersive remote teleoperation therefore presents the challenge of appropriately managing the data flow from the data acquisition end to the visualization at the operator, while allowing optimal visual quality. Typically, this data flow involves image data acquisition, some form of processing for point-cloud generation or 3D reconstruction, data compression (encoding) and streaming, decoding at the operator side, and visual rendering [13, 15]. These components must respect the limitations of the network communication, the computation, and user's display hardware.

In this paper, the human visual system provides the inspiration to address the coupling between data acquisition and its rendering for remote teleoperation using VR head-mounted display (HMD). Specifically, the concept of the human visual foveation [16], where the human eye has high visual acuity (sharpness) at the center of the field-of-view, with this acuity reducing falls off toward the periphery. This visual effect has been harnessed by gaze-contingent (foveated) graphics rendering [17] and imaging techniques. These foveation methods are designed for displays with fixed focal distances, such as desktop monitors, or stereo displays that offer binocular depth cues. Here, however, we will use this acuity fall-off to facilitate the processing, streaming, and rendering of 3D data to a remote user, thereby reducing the amount of data to be transmitted. The user's gaze or viewpoint is exploited to divide the acquired 3D data into concentric conical regions of progressively reducing resolution, with the highest resolution in the central cone. The radii of the cone are determined based on the projection of the eccentricity values of the human-eye foveal regions into the acquired 3D data. Figure 1 visually shows the concept of projecting the foveated regions into the 3D visual data.

Validation and experimental evaluations show that the system runs at reduced throughput requirements with low latency while maintaining the immersive experience and task performance. As a result, this approach presents a re-thinking of the immersive remote teleoperation processes – it includes the

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press

operators/user's visual perspective in optimizing the data flow between the remote scene and the operator, without affecting the quality of experience and task performance for the operator. The following sections outline the contribution of this research in detail.

2. Related work and contribution

Our work combines diverse fields, including gaze tracking, real-time 3D reconstruction, compression, streaming, rendering for immersive telepresence, and telerobotics. Substantial research exists in each of these areas and a full review would be out-of-scope here; however, the most relevant approaches in the related fields are briefly discussed.

A. Immersive remote telerobotics/telepresence: Researchers have long seen the advantages of using 3D VR environments in telepresence. Maimone et al. [18, 19] were among the first to investigate a telepresence system offering fully dynamic, real-time 3D scene capture and viewpoint flexibility using a head-tracked stereo 3D display. Orts–Escolano et al. [15] presented "holoportation" giving high-quality 3D reconstruction for small fixed-sized regions of interest. Authors in refs. [7, 20] present remote exploration telepresence systems for large- and small-scale regions of interest with reconstruction and real-time streaming of 3D data. In refs. [9, 21], the authors extended this idea with simultaneous immersive live telepresence for multiple users for remote robotic teleoperation and collaboration. Furthermore, VR-based immersive interfaces for robotic teleoperation have gained a lot of traction, where models of the remote robots are combined with real-time point-cloud renderings, real-time stereo video, and gesture tracking inside VR [4, 13]. Su et al. [22] present a comprehensive state-of-the-art framework developed by the robotics community to integrate physical robotic platforms with mixed reality interfaces.

B. Real-time 3D reconstruction: Recent years have seen an increasing research on dense 3D reconstruction problems, following two directions: volumetric (voxel-based) and pointwise (surfel-based). Due to the popularity of KinectFusion [23], volumetric reconstruction has become dominant owing to its relatively straightforward CPU-GPU implementation and parallelizability. The surfel-based approach, which represents the scene with a set of points [24], is inherently adaptive for higher resolution requirements as it combines frequent local model-to-model surface loop closure optimizations with intermittent global loop closure to recover from arbitrary drifts. Nevertheless, both methods have their advantages and disadvantages [25].

C. 3D Data compression and streaming: Efficient compression and representation techniques, such as 3D polygon mesh, point clouds, and signed distance fields, are being investigated intensively [21, 26]. RGB-D cameras, for example, Intel Realsense, provide direct access to fast point clouds [27], but dense and realistic remote reconstruction and streaming have large memory and bandwidth requirements. To address this, a number of point-cloud compression techniques have been proposed [26, 28–30]. An interesting work on remote teleoperation was proposed by De Pace et al. [31], where the authors utilized the libjpeg-turbo library to compress color and depth frames before transmitting them over user datagram protocol (UDP). Given that UDP does not inherently guarantee reliable data transmission, the authors implemented additional mechanisms, including compression, frame validation, and decompression, to enhance the effectiveness of data transmission while maintaining the maximum possible frame rate.

D. Gaze tracking: Dating as far back as the 18^{th} century, eye tracking has fascinated researchers studying human emotions and mental state. One of the first eye trackers was built by Edmund Huey [32], to understand the reading process of humans, using contact lenses having a hole for pupil tracking. A similar approach was used by Fitts et al. [33] when studying pilot eye movements during landings. Another significant contribution in eye tracking by Yarbus [34], showed that gaze trajectories depend on the task. The past three decades have seen a major revolution in eye-tracking research and commercial

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press

Table I. *Comparison of different state-of-the-art research works in foveated rendering.*

Method	Data type	Rendering paradigm
[39]	Image and video	Neural rendering
[40]	Image and video	Transmission
[37]	Volume data	Ray casting
[41]	Volume data	Ray casting
[36]	Geometric meshes	Rasterization
[17]	Geometric meshes	Rasterization
[42]	Point cloud	LoD simplification
Our	Point cloud	Transmission

applications due to the ubiquity of artificial intelligence algorithms and portable consumer-grade eye-trackers. Commercial HMDs that include eye trackers are the Fove-0, Varjo VR-1, PupilLabs Core, and the HTC Vive Pro Eye [35].

E. Foveated rendering: Increasingly, research is investigating how the human visual system, especially the concept of foveation, can facilitate graphics rendering, both in 2D and 3D. Guenter et al. [17] presented an early foveated rendering technique that accelerates graphics computation. This rendered three eccentricity layers around the user's fixation point with the parameters of each layer being set by calculating the visual acuity. Stengel et al. [36] proposed gaze-contingent rendering that only shades visible features of the image while cost-effectively interpolating the remaining features, leading to a reduction of fragments needed to be shaded by up to 80%. Bruder et al. [37] used a sampling mask computed based on visual acuity fall-off using the Linde-Buzo-Gray algorithm. Commercially, VR headsets are exploiting foveated rendering for increased realism and reduced graphical demands [38]. Table I summarizes and compares different state-of-the-art research works in foveated rendering.

2.1. Contributions

Drawing on the advances in the above fields, this paper presents our research work on immersive teleoperation. A key innovation is utilizing foveation for sampling and rendering of real-time dense 3D point-cloud data for immersive remote teleoperation. This has the following contributions:

1. *Main:* A novel approach for foveated rendering of real-time, remote 3D data, for immersive remote teleoperation in VR, i.e., differentially sampling, unicasting, and rendering of real-time dense point clouds in VR, exploiting the human visual system.
2. *Additional:*

   (a) A method for streaming a partitioned point cloud and combining it at the operator site using GPU and CPU parallelization.
   (b) A new volumetric point cloud density-based peak signal-to-noise ratio (PSNR) metric to evaluate the proposed approach.
   (c) A user study with 24 subjects to evaluate the impact of the proposed approach on perceived visual quality.

3. System overview

This section briefly overviews the proposed interface and theoretical foundation behind the proposed framework. As shown in Figure 2, it follows a general teleoperation setup, which includes an operator with a visualization interface and a remote environment. As illustrated in Figure 3, the proposed

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press

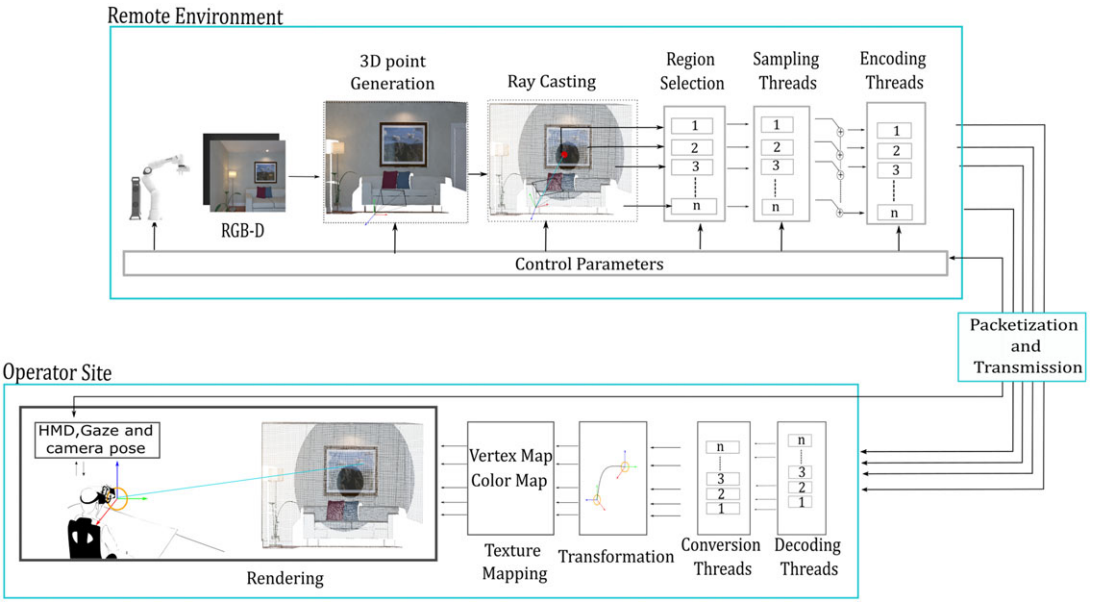


Figure 2. *Proposed telerobotics interface: A user teleoperating a remote robotic arm using the HTC VIVE focus virtual reality (VR) system. The Unreal VR graphics engine provides immersive visualization. Remote RGB-D camera provides foveated point cloud in real-time.*

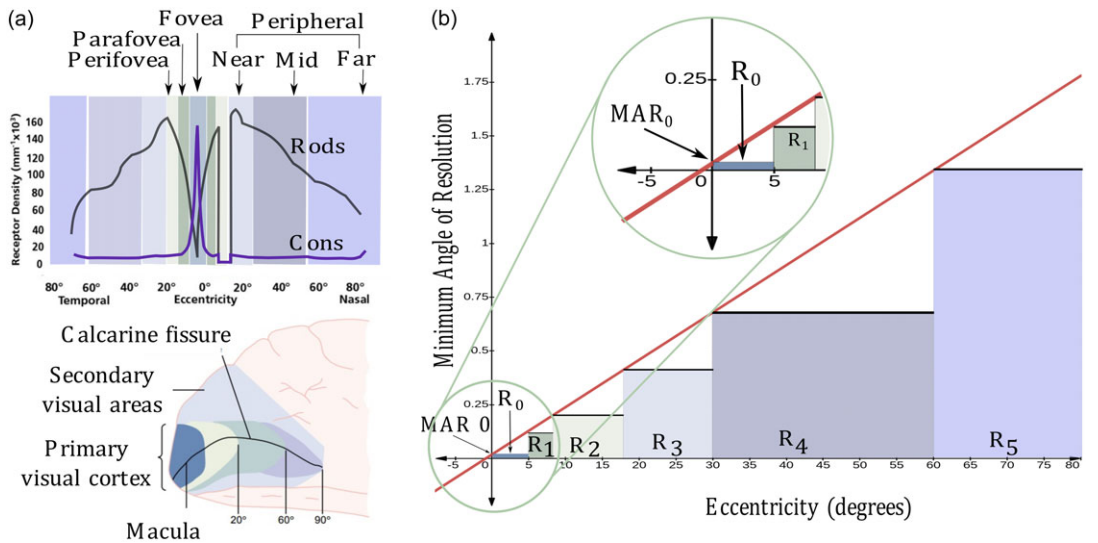


Figure 3. *Proposed interface: A user teleoperating a remote robotic arm using the HTC VIVE focus virtual reality (VR) system. Unreal VR graphics engine provides immersive visualization. Remote RGB-D camera provides foveated point cloud in real-time.*

framework comprises three primary components: the operator site, the remote environment, and a communication network. The gesture/motion controllers located at the operator site transmit instructions to the remote environment. Additionally, the gaze direction and pose of the head-mounted display at the operator site are transmitted to the remote environment. This allows for the capturing, processing, streaming, and visualization of remote environments in 3D for the operator. The communication network

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press

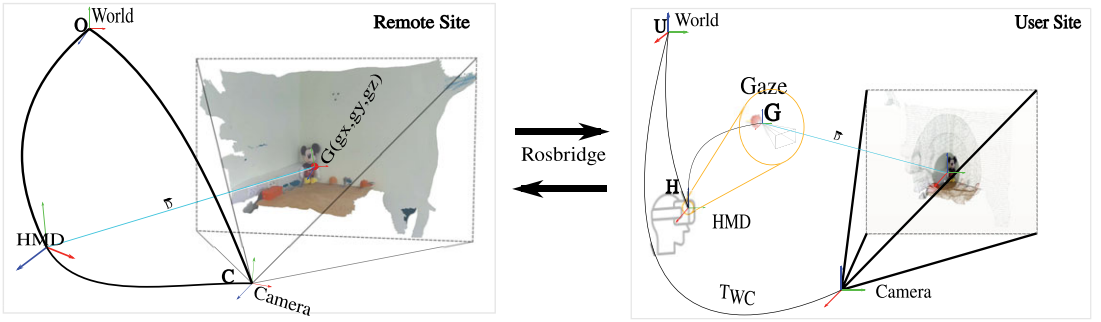


Figure 4. *A) Photoreceptor distribution in the retina and retinotopic organization, image adapted from ref. [43]. B) minimum angle of resolution against eccentricity for the retinal regions* ($R0 - R5$)*, based on Eq. (1).*

allows real-time data exchange between the operator site and the remote environment, that is, sending commands, receiving remote robot status, and receiving real-time dense point cloud. The proposed framework was implemented according to the figure outlined in Figure 3 and the complete components are further described in the subsequent sections for a detailed description.

3.1. Operator site

The operator site manages the following functionalities: (1) Gaze tracking and calculating fixation points in 3D, (2) decoding and rendering of the streamed point-cloud data, (3) facilitating functionalities for operator to command the remote robot using HTC motion controllers (HMC), and (4) real-time transfer of this information. A VR-based interface is created using the Unreal Engine (UE) to provide an immersive remote environment. Figure 3 illustrates the implementation of a parallel streamer, a point-cloud decoder, and a conversion system for transferring textures to the UE GPU shaders. It has the HTC Vive Pro Eye VR headset as the main interaction interface, which comes with gesture controllers and a tracking system, as well as a built-in Tobii Eye Tracking system. The Tobii system gives an accuracy of $0.5° - 1.1°$ with a trackable FOV of 110°. Computing resources include Windows 10, Nvidia GeForce GTX 1080 graphics card, and UE4 for the virtual environment. It also transfers the gaze and headset pose information to the remote site in real-time. The following section will outline the theoretical and practical implementation of the operator site functionalities.

3.1.1. The human visual system and gaze tracking

Humans perceive visual information through sensory receptors in the eyes. The process begins when light passes through the cornea, enters the pupil, and then gets focused by the lens onto the retina. This is then processed in the brain where an image is formed. The retina has two kinds of photoreceptors: cones and rods. Cones are capable of color vision and are responsible for high spatial acuity. Rods are responsible for vision at low light levels. As shown in Figure 4-A, the cone density is highest in the central region of the retina and reduces monotonically to a fairly even density in the peripheral retina region. Retinal eccentricity (or simply, eccentricity) implies the angle at which the light from the image gets focused on the retina. This distribution of the photoreceptors gives rise to the concept of Foveation and helps define the idea of visual acuity.

The density of photoreceptors (cones and rods) declines monotonically and in a continuous manner from the center of the retina to its periphery. Nevertheless, approximating the retina as being formed of discrete concentric regions, where the density of the photoreceptors corresponds to eccentricity angles, helps simplify the concept – this is the concept of *Foveation*. A summary of photoreceptors distribution is presented in Table II, followed by further details below:

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press

Table II. *Human retinal regions and their sizes in diameter and angles (derived from [44]).*

	Region	Size in diameter (mm)	Size in angle
R_0	Fovea	1.5	5°
R_1	ParaFovea	2.5	8°
R_2	PeriFovea	5.5	18°
R_3	Near peripheral	8.5	30°
R_4	Mid peripheral	14.5	60°
R_5	Far peripheral	26	> 60°

- The region from the center of the retina up to 5° of eccentricity is the *Fovea* region. The Fovea is only about 1% of the retina but has the highest density of cone photoreceptor cells, and the brain's visual cortex dedicates about 50% of its area to information coming from the Fovea [45]. Therefore, the Fovea has the highest sensitivity to fine details.
- The region that surrounds the Fovea is commonly known as the *Parafovea*, which goes up to 8° of the visual field [16]. The parafoveal region provides visual information as to where the eyes should move next (saccade) and supports the Fovea to process the region of interest in detail. Previous research has investigated if meaningful linguistic information can be obtained from parafoveal visual input while reading [46].
- The next region that surrounds the Parafovea is called *Perifovea*, which extends approximately up to 18° of eccentricity. In this region, the density of rods is higher than that of cones, about 2:1. Consequently, unlike the Fovea and Parafovea, only rough changes in shapes are perceived in this region [47].
- The region beyond 18°, and up to about 30° of the visual field, is known as the *Near-Peripheral Region*. It has the distribution of 2–3 rods between cones [44]. This region is responsible for the segmentation of visual scenes into texture-defined boundaries ("texture segregation") and the extraction of contours for pre-processing in pattern and object recognition [48].
- The region between 30° and 60° of eccentricity is called the *Mid-Peripheral Region* [49]. Although acuity and color perception degrade rapidly in this region, researchers have shown that color perception is still possible even at large eccentricities, up to ∼ 60° [50].
- The region at the edge of the visual field (from 60° up to nearly 180° horizontal diameter) is called the *Far Peripheral Region*. This region has widely separated ganglion cells, and visual functions such as stimulus detection, flicker sensitivity, and motion detection are still possible here [48].

Visual acuity can be quantitatively represented in terms of *minimum angle of resolution* (MAR_0, measured in arcminutes) [17, 48, 51]. MAR_0 can be understood as the smallest angle at which two objects in the visual scene are perceived as separate [51]. MAR_0 accounts for the number of neurons allocated to process the information from the visual field, as a function of the eccentricity. This relation between MAR_0 and eccentricity can be approximated as a linear model, which has been shown to closely match the anatomical features of the eye [17, 37, 48, 51].

$$\text{MAR} = mE + MAR_0 \tag{1}$$

Here, MAR_0 is the intercept, which signifies the smallest resolvable eccentricity angle for humans, and m is the slope of the linear model. MAR_0 for a healthy human varies between 1 and 2 arcminutes, that is, $1/60°$ to $1/30°$ ($1° = 60$ arcminutes). Authors in ref. [17] experimentally determined the values of m based on observed image quality, ranging between 0.022 and 0.034. Figure 4-B captures this linear relationship of Eq. (1), showing how visual acuity degrades as a function of eccentricity, represented

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press

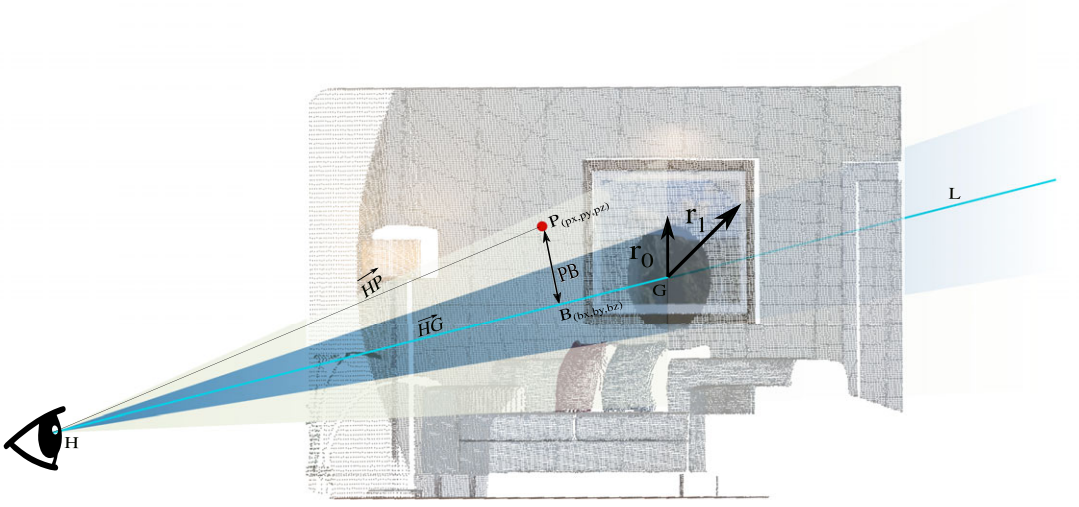


Figure 5. *A schematic showing the coordinate system on the remote and operator sites.*

with a piece-wise constant approximation, that is, each retinal region has a distinct constant MAR value [17].

The information from the operator's site is subsequently transmitted to foveate the 3D point cloud. *Foveating* a 3D point cloud implies introducing concentric regions in it that correspond to the retinal fovea regions. The regions are centered on the human eye-gaze direction, each of them having a specific radius and its associated visual acuity, that is, rendering quality.

3.1.2. Visualization and coordinate transformations

To visualize and explore the incoming point cloud, as well as the remote robotic platforms, a VR-based interface is designed using the Unreal graphics engine on Windows 10. This creates the immersive remote teleoperation environment for the operator. As noted earlier, there is an interdependence between the operator site and the remote site, in that, the eye-gaze data from the operator site is required for the foveation model at the remote site. The foveated point cloud is then streamed back to the operator site to be rendered for visualization. Furthermore, the operator site and the remote site are independent environments with their respective reference frames. It is therefore necessary to implement appropriate transformations among all the entities to ensure correct data exchange and conversion. As shown in Figure 5, the reference frames are as follows: UE world coordinate frame **U**, HMD coordinate frame **E**, and the gaze direction vector ${}^{E}\vec{D} \in R^3$ on **E**.

To calculate the correct gaze pose in **U**, transforming ${}^{E}\vec{D}$ from $\mathbf{E} \rightarrow \mathbf{U}$ is required, through the head pose ${}^{U}\mathbf{H}$ on **U**, as follows:

$$ {}^{U}\vec{D} = {}^{U}\mathbf{H} \cdot {}^{E}\vec{D} \tag{2} $$

This gaze direction ${}^{U}\vec{D}$, along with the head pose **H** are communicated to the remote site for additional processing. Further, the received point cloud from the remote site is visualized in Unreal and needs to be positioned based on the pose of the camera positioned at the remote site. At the remote site, the coordinate system of the camera pose, ${}^{O}\mathbf{P}$ is in OpenGL, **O**. The pose has to be transformed, using a change-of-basis matrix, into the UE coordinate system. UE uses a left-handed, z-up coordinate system, while the camera coordinates of OpenGL use a right-handed coordinate system, y-up. Eq. (3) provides the coordinate transformation formula, where **B** is the change-of-basis transformation matrix.

$$ {}^{U}\mathbf{P} = \mathbf{B} \cdot {}^{O}\mathbf{P} \cdot \mathbf{B}^{-1} \tag{3} $$

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press

At the operator site, rendering the received real-time dynamic point-cloud data from the remote site requires a high-speed large data transfer, as well as efficient and high-quality visualization. To meet these requirements, the following modules were developed, as seen in Figure 3:

A real-time point-cloud decoder: that decompresses the data received at the operator site. The decoding module includes the state-of-the-art point-cloud codec algorithm from ref. [26] and implemented the Boost ASIO over a TCP socket for data transfer. As the point cloud codec needs to compress and stream point clouds using gaze information from master station, TCP was chosen to ensure data integrity and packet reception.

Conversion system: Each decoded point-cloud region $\mathcal{P}_n(\forall n \in 1, \cdots, N)$ has to be converted into a texture for visualization, where the reference frame of the received data has to be transformed into that of the user site, that is, the unreal graphics engine coordinate system.

A rendering system: The data should be transferred to the GPU and made accessible to the graphics engine shader for real-time rendering. We've implemented a splatting-based technique to render point clouds using UE's Niagara particle system. After decoding the point cloud data, positions, and colors, we transferred this data to the GPU via the Niagara module. Once on the GPU, each particle's UV coordinates are calculated based on its unique identifier, texture size, and virtual camera positions. This process ensures precise mapping of the point cloud data onto the rendered surfaces. By integrating Niagara's rendering capabilities, we achieve faster rendering performance, which is around eight milliseconds.

3.1.3. Robot control

The motion controllers of the HTC Vive Pro Eye allow remote teleoperators to send control commands in real-time using a wireless motion controller interface. These HTC Motion Controllers (HMC) are tracked in space, allowing the teleoperator to freely explore the VR environment and interact naturally with the remote robots. The HMC has two buttons: To engage/disengage motion commands between the HMC and remote robot, overcoming range limitations, and to open/close the remote robot's end-effector for precise object manipulation and control. Since the operator directly commands the motion of the remote robot using a HMC controller, our design follows a human-in-the-loop design paradigm without assuming any autonomy or semi-autonomy of the robots [4]. The remote environment point cloud is sampled, streamed, and rendered for the human operator in real-time. Based on the remote scene, the operator plans the next step and the action commands are sent to the remote robot. The remote robots are considered passive and do not act autonomously. Velocity controller was used due to its ability to provide smooth and continuous control over the movement of a remote robotic system. It calculates joint angles, velocities, and accelerations, which are then transmitted from the HMC interface to the remote robots. The joint states of the remote robots are relayed back to the interface to update the poses of the virtual robot's models. This integration allows the operator to see and command the motion of the virtual robot, with the motion of the real robot being mapped 1:1 to the virtual robot motion. Due to the non-homothetic nature of the kinematics between the remote robot and the operator controllers, the velocity-control commands the 7-DOF pose of the robot. The inverse Jacobian method is employed to calculate the velocity of the HMC, which is then mapped to the robot. To command the robot pose, the position error $\mathbf{e} \in SE(3)$ is determined by comparing the current pose with the desired pose. The damped least-squares solution, as shown in Eq. (4), is iteratively used to find the change in joint angles $\Delta\mathbf{q}$ that minimizes the error $\mathbf{e}$.

$$\Delta\mathbf{q} = (\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})^{-1}\mathbf{J}^T\mathbf{e}, \tag{4}$$

where $\mathbf{J}$ is the manipulator Jacobian, $\lambda \in R$ is a non-zero damping constant, and $\mathbf{I}$ is the identity matrix. Finally, a proportional controller $\dot{\mathbf{q}} = K_p \cdot \Delta\mathbf{q}$ sets the joint velocities to achieve the desired pose. The values for λ and K_p were determined empirically as 0.001 and 0.6, respectively.

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press

3.2. Remote site

The remote site consists of modules implemented in OpenGL for coordinate frame transformation, RGB-D data acquisition, real-time 3D reconstruction, partitioning, foveated sampling, encoding, and streaming of each foveated region in separate parallel streams, as shown in Figure 3. It has the computing resources of Nvidia GeForce GTX 1080 graphics card with Ubuntu 20 operating system.

3.2.1. Coordinate transformations

As shown in Figure 3, a parallel module will receive information about head pose ${}^{U}\mathbf{H}$ and the gaze direction ${}^{U}\vec{D}$ from the Operator site. The coordinate system of the head pose ${}^{U}\mathbf{H}$ and the gaze direction ${}^{U}\vec{D}$ has to be transformed from the Unreal frame, **U**, to the remote site OpenGL coordinate system, **O**, using a similar change-of-basis matrix, Eq. (5). Here: **Q** is the change-of-basis transformation matrix from UE to OpenGL. Figure 5 left shows the reference frames of the remote site, and they are described as follows: OpenGL world coordinate frame **O** and the camera frame **C**.

$$\begin{aligned} {}^{O}\mathbf{H} &= \mathbf{Q} \cdot {}^{U}\mathbf{H} \cdot \mathbf{Q}^{-1} \\ {}^{O}\vec{D} &= \mathbf{Q} \cdot {}^{U}\vec{D} \cdot \mathbf{Q}^{-1} \end{aligned} \tag{5}$$

Gaze direction vector ${}^{O}\vec{D}$ and the head pose ${}^{O}\mathbf{H}$ have to be transformed into the camera coordinate frame, in order to perform 3D reconstruction, map partitioning, and sampling. For every frame, the color image **C** and depth map **D** are registered into the map model $\mathcal{M}$ by estimating the global pose of the camera **P**. Section 3.2.2 will provide a concise description of this.

$$\begin{aligned} {}^{C}\mathbf{H} &= \mathbf{P}^{-1} \cdot {}^{O}\mathbf{H} \\ {}^{C}\vec{D} &= \mathbf{P}^{-1} \cdot {}^{O}\vec{D} \end{aligned} \tag{6}$$

The head pose in the camera frame is used as a point of gaze origin **H** (hx,hy,hz) and using the gaze direction vector, ${}^{C}\vec{D}$, a ray, that is, the *gaze vector* **L** is projected into the 3D map.

3.2.2. 3D data acquisition, mapping, partitioning, and sampling

The acquisition and point generation module acquires RGB-D images from the RGB-D cameras, for example, Intel RealSense and ZED stereo camera. The generated points (maps) pipeline leverages the state-of-the-art real-time dense visual SLAM system, ElasticFusion [24] for initial camera tracking. The map $\mathcal{M}$ is represented using an unordered list of surfels, where each surfel $\mathcal{M}^{s}$ has a position $\mathbf{p} \in \mathbb{R}^3$, a normal $\mathbf{n} \in \mathbb{R}^3$, a color $\mathbf{c} \in \mathbb{R}^3$, a weight $w \in \mathbb{R}$, a radius $r \in \mathbb{R}$, an initialization timestamp t_0, and a current timestamp t. The camera intrinsic matrix **K** is defined by: (i) the focal lengths f_x and f_y in the direction of the camera's $x-$ and $y-$ axes, (ii) a principal point in the image (c_x, c_y), and (iii) the radial and tangential distortion coefficients k_1, k_2 and p_1, p_2 respectively. The domain of the image space in the incoming RGB-D frame is defined as $\Omega \subset \mathbb{N}^2$, with the color image **C** having pixel color $\mathbf{c} : \Omega \rightarrow \mathbb{N}^3$, and the depth map **D** having pixel depth $d : \Omega \rightarrow \mathbb{R}$.

A. 3D point partitioning: For brevity, the symbol $\mathcal{M}$ is used for real-time point-cloud. The density of $\mathcal{M}$, especially at high resolutions, implies increased computational complexity and more graphical and time resources for streaming it in immersive remote teleoperation. The foveation model can be utilized here to reduce the data. By projecting the retinal fovea regions into it, $\mathcal{M}$ is partitioned into regions. It is then resampled to approximate the monotonically decreasing visual acuity in the foveation model, termed *foveated sampling*.

To partition $\mathcal{M}$ into regions, the discussion in Section 3.1.1 is taken forward. The center of the eye gaze is used as a point of origin $\mathbf{H}(hx, hy, hz)$. To partition $\mathcal{M}$ into $\mathcal{M}_n$ regions $\forall n \in \{0 \cdots N\}$, for each of the N retinal regions, a ray is cast from $\mathbf{H}(hx, hy, hz)$. This ray, i.e., the *gaze vector* $\mathbf{L} \in \mathbb{R}^3$ is extended up to last point of intersection $\mathbf{G}(gx, gy, gz)$ with the surfel map. The foveated regions are now structured around **L** and (Figure 1) shows the foveated regions sampled, streamed and rendered. With 3D data, the concentric regions are conical volumes, with their apex at $\mathbf{H}(hx, hy, hz)$, with increasing radii away

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press

Algorithm 1. 3d Point Partitioning Algorithm.

Input: $\mathcal{M}$ /* Map to be partitioned */
$\mathbf{L}$ /* Gaze direction vector */
$e_{0\cdots n}$ /* Eccentricity angles */
foreach *surfel* P_i *in the map* $\mathcal{M}$ **do**
  $\mathbf{B} \leftarrow \mathbf{proj}_{\mathbf{L}}^{P_i}$ /* projection of point P_i on L */
  $d^{vi} \leftarrow \|\vec{HB}\|$ /* distance between H and B */
  $\boldsymbol{d} \leftarrow \mathbf{PB} \perp \mathbf{L}$ /* shortest distance */
  for *j=1* **to** *max(e)* **do**
    $r_j \leftarrow \tan(e_j) * d^{vi}$ /* compute the radii r_j */
  end
  /* put P_i into the maps $\mathcal{M}_0 \ldots \mathcal{M}_n$ */
  if $\boldsymbol{d} < r_0$ **then**
    $\mathcal{M}_0 \leftarrow \mathbf{P}_i$;
  else if $\boldsymbol{d} > r_0$ ***AND*** $\boldsymbol{d} \leq r_1$ **then**
    $\mathcal{M}_1 \leftarrow \mathbf{P}_i$;
  ⋮
  else
    $\mathcal{M}_n \leftarrow \mathbf{P}_i$
  end
end

from **H**. Algorithm 1, which is implemented in Compute Unified Device Architecture (CUDA) in the GPU for faster processing, details how the radii are calculated based on d^{vi} for each surfel. To assign each surfel in $\mathcal{M}$ to a particular region $\mathcal{M}_n$, the shortest distance, i.e., the perpendicular distance between the surfel and **L** is used. As shown in Figure 1, the shortest distance from the surfel $\mathbf{P}(px, py, pz)$ to the ray **L** is the perpendicular $\mathbf{PB} \perp \mathbf{L}$, where $\mathbf{B}(bx, by, bz)$ is a point on **L**. $\|PB\|$ can be obtained using the the projection of $\vec{HP}$ on **L**, that is, the cross product of $\vec{HP}$ and $\vec{HG}$, normalized to the length of $\vec{HG}$, refer Eq. (7). Algorithm 1 assigns surfel **P** to the region $\mathcal{M}_n$.

$$\|PB\|_{\mathbf{P}} = \frac{\|\vec{HP} \times \vec{HG}\|}{\|\vec{HG}\|} \tag{7}$$

B. Foveated Point-Cloud (PCL) Sampling: The partitioned global map $\mathcal{M}$, with the region-assigned surfels, is converted into a PCL point-cloud data structure, $\mathcal{P}_n$ for each $\mathcal{M}_n$ region $\forall n \in \{0 \cdots N\}$. This conversion is sped up using a CUDA implementation in the GPU. To implement the foveated sampling, the $\mathbb{R}^3$ space of each $\mathcal{P}_n$ region needs to be further partitioned into an axis-aligned regular grid of cubes. This process of re-partitioning the regions is called *voxelization* and the discrete grid elements are called *voxels*. After voxelization, the down-sampling of the PCL follows the foveation model – the voxels in the fovea region of the PCL are the densest, and this density progressively reduces toward the peripheral regions.

This voxelization and down-sampling is a three-step process: (1) calculating the volume of the voxel grid in each region, (2) calculating the voxel size, that is, dimension, $\mathfrak{v}_n$, for the voxelization in each region, and (3) down-sampling the point cloud inside each voxel for the region by approximating it with the 3D centroid point of the point cloud.

Calculating the volume of the voxel grid for each region is done by simply calculating the point cloud distribution for that region, $[(x_{n,min}, x_{n,max}), (y_{n,min}, y_{n,max}), (z_{n,min}, z_{n,max})]$. Calculating the voxel size, $\mathfrak{v}$, is a more involved process. Here, the visual acuity discussion from Section 3.1.1 is utilized. Consider the

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press

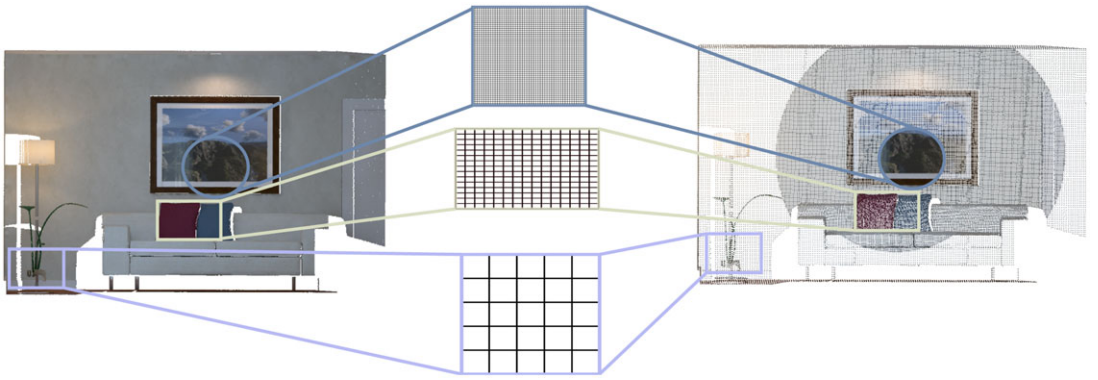


Figure 6. *Foveated point-cloud sampling example showing the different voxel grid sizes for the different regions.*

voxelization of the central fovea region, $\mathcal{P}_0$. As noted earlier, the smallest angle a healthy human with a normal visual acuity of 20/20 can discern is 1 arcminute, that is, 0.016667°. Following Eq. (1) therefore, MAR_0, which is the smallest resolvable angle, is 0.016667°. The smallest resolvable object length on a virtual image can be calculated as:

$$\mathfrak{l} = d^{vi} * \tan(\text{MAR}_0) \tag{8}$$

Equation (8) itself could provide the optimum voxel size, $\mathfrak{v}$. The important consideration here is the value of d^{vi}. In Algorithm 1, a d^{vi} value for each point is calculated. In contrast, here in order to down-sample the region based on the voxelization, we calculate one d^{vi} value for the entire $\mathcal{P}_0$ region, approximated as the distance from the eye (point of gaze origin) to the 3D centroid of the point-cloud in the region, Eq. (9).

$$\mathfrak{pc}_0 = \frac{1}{N_{\mathcal{P}_0}} \left(\sum_{i=1}^{N_{\mathcal{P}_0}} x_i, \sum_{i=1}^{N_{\mathcal{P}_0}} y_i, \sum_{i=1}^{N_{\mathcal{P}_0}} z_i \right) \tag{9}$$

$$d_0^{vi} = \mathbf{d}(\mathbf{H}, \mathfrak{pc}_0) \tag{10}$$

where $N_{\mathcal{P}_0}$ is the number of PCL points in $\mathcal{P}_0$, and $\mathbf{H}$ is the eye-gaze origin. Then, Eq. (8) is re-written as Eq. (11) to give the voxel size $\mathfrak{v}_0$ for the region.

$$\mathfrak{v}_0 = d_0^{vi} * \tan(\text{MAR}_0) \tag{11}$$

Once the voxelization of region $\mathcal{P}_0$ is finalized, for the subsequent concentric regions from $\mathcal{P}_1$ to $\mathcal{P}_n$, the voxel sizes are correlated with the linear MAR relationship in Figure 4. Eq. (12) shows that as the eccentricity angle of the regions increases, so do the voxel sizes.

$$\text{MAR}_n = m \cdot E_n + \text{MAR}_0$$

$$\mathfrak{v}_n = \frac{\text{MAR}_n}{MAR_{n-1}} * \mathfrak{v}_{n-1} \tag{12}$$

The increasing voxel size away from the fovea region implies more and more points of the point cloud of the corresponding regions are now accommodated within each voxel of that region. Therefore, when the down-sampling step is applied, the approximation of the point cloud within a voxel is done over progressively dense voxels. For the down-sampling part, the region $\mathcal{P}_0$ being the fovea region is left untouched so its density is the same as the incoming global map density. The down-sampling in the subsequent regions is done by approximating the point cloud within each voxel with its 3D centroid,

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press

using Eq. (13).

$$\mathfrak{pc}_n^v(x,y,z) = \frac{1}{N_{\mathcal{P}_n}^v}\left(\sum_{i=1}^{N_{\mathcal{P}_n}^v} x_i, \sum_{i=1}^{N_{\mathcal{P}_n}^v} y_i, \sum_{i=1}^{N_{\mathcal{P}_n}^v} z_i\right) \tag{13}$$

Here, $N_{\mathcal{P}_n}^v$ is the number of points in voxel v of the region $\mathcal{P}_n$ ($\forall n \in \{1 \cdots N\}$). Figure 6 shows the sample voxel grids for the different regions. The foveated sampling and compression is the most computationally expensive system component, and the OpenMP multi-platform, shared-memory, parallel programing method is used to achieve real-time performance.

3.3. Communication network

Point cloud can be transmitted and received as a single stream, combining all the foveated regions of the point-cloud, or as parallel separate streams of the regions. Single streams have the advantage of synchronized data but can be very heavy in terms of bandwidth requirements. Parallel streams can help the network optimize the data transmission, reducing simultaneous bandwidth requirement. However, parallel streams may also suffer from varying data transmission rates due to network delays and size differences. To address this issue, all streamed point-cloud regions are timestamped and a buffer resource module is created at the user site for software synchronization using the local clock synchronized with a central NTP server. Between the remote and user sites, a new point-cloud streaming pipeline was implemented using the Boost ASIO cross-platform C++ library for network and low-level I/O programing. This pipeline accounted for the throughput-intensive point-cloud data. The user site communicates with the remote site using TCP sockets. Further, the Robot Operating System (ROS) served as an additional connection to exchange lightweight data between the user VR interface and the remote site. This included the head pose ${}^{U}\mathbf{H}$, the gaze direction vector ${}^{U}\vec{D}$, and the pose of the remote camera ${}^{O}\mathbf{P}$, among other things.

4. Experiment design and evaluation metrics

The experiment design focuses on a thorough evaluation of the proposed framework using; datasets, online and acquired, using defined experimental conditions and benchmarking against defined metrics.

4.1. Experiments

To thoroughly assess the effectiveness of the proposed system, we carefully conducted a series of three subjective experiments. These experiments were designed to investigate the system's functionality and performance deeply.

1. ***Quality of Experience* (QoE):** To assess the impact of the proposed framework on the quality of the experience, we pose the following two research questions to guide the study, **RQ1**: Can subjects differentiate between scenes with varying graphical contexts, streamed with and without the proposed system? and **RQ2**: How do different combinations of the foveated regions impact subjective quality of experience?
2. ***Visual search:*** The visual world is overwhelmingly rich – it contains far too much information to perceive simultaneously. Given these limits, the human visual system needs mechanisms to allocate processing resources optimally according to task demands. Several studies have shown that targets presented near the fixation point (fovea) are found more efficiently than targets presented at more peripheral locations. However, when targets are presented away from the fovea, accuracy reduces and increases search times and number of eye movements [52]. In applications such as search and rescue using teleoperated robots, rapid visual responses to a potentially dynamically

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press

changing can be critical. Target locations should be assessed with time and bandwidth constraints in mind. Inspired by ref. [53], a user study is conducted to assess the effect of peripheral quality loss on search performance.

3. ***Verification through remote telemanipulation:*** Following an in-depth analysis of the proposed system's performance in terms of its Quality of Experience and suitability for visual search tasks, we pursued a comprehensive evaluation. This evaluation aimed to investigate the system's impact on performance and execution times in real remote robotic telemanipulation scenarios. To achieve this, we propose a pick-and-place task experiment, which is detailed in Section 6.

4.2. Experimental conditions

Three test foveation conditions were created, each having a different combination of the six regions mentioned in Section 3.1.1, going from high-performance gain to high visual quality.

- **F1:** The point cloud has four partitions – Fovea, Parafovea, Perifovea, and the remainder. The progressive foveated sampling in the regions follows Eq. (12). For the *remainder* of the point-cloud region, it is sampled using the voxel sizes for the *Far Peripheral* region.
- **F2:** has five partitions – Fovea, Parafovea, and Perifovea, *Near Peripheral*, and then the remainder, with a similar sampling strategy to F1.
- **F3:** includes all six partitions as seen in Table II – Fovea, Parafovea, Perifovea, Near-, Mid-, and the Far Peripheral regions, with the corresponding sampling strategy.

In addition to the three conditions above, two further conditions are created to represent the two ends of the sampling scale, i.e., full sampling (F0) and no sampling (FREF).

- **F0:** To simulate the approach of fully down-sampling a point cloud to allow the least streaming costs, the whole point cloud is down-sampled using the voxel size of the *Near Peripheral* region in Eq. (12).
- **FREF:** We used the state-of-the-art point-cloud compression method from ref. [26] as the baseline reference and named FREF, for our comparisons across conditions F0 to F3. This approach was selected because comparing our framework against uncompressed raw RGB-D data would not provide a fair or practical evaluation due to the significantly larger data sizes associated with raw RGB-D formats. Additionally, in this baseline condition, the entire visual field remains unaltered, and our proposed framework is not applied.

4.3. Datasets

For the evaluation, four datasets were used. Sample images are shown in Figure 7. One of the datasets is a dynamic scene where a balloon (**BAL** a) moves within a lab environment. Additionally, two static datasets, shown in Figure 7, are synthetic online datasets representing static environments. These include a living room (**LIV**) and an office scene (**OFF**), both of which come with ground truth data [54]. Figure 8 presents a dataset specifically gathered for conducting visual search experiments.

4.4. Evaluation metrics

The following objective and subjective metrics were used to evaluate the proposed framework:

1. Data transfer rate: measured as an overall value between the user and remote sites, using the network data packet analysis tool, Wireshark [55].

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press

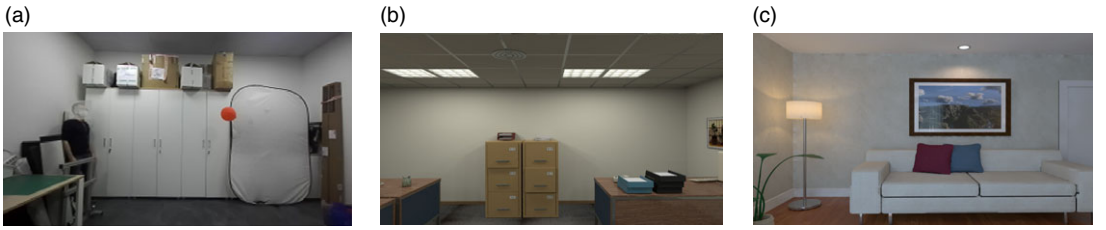


Figure 7. *Sample frames from three of the four evaluation datasets.*

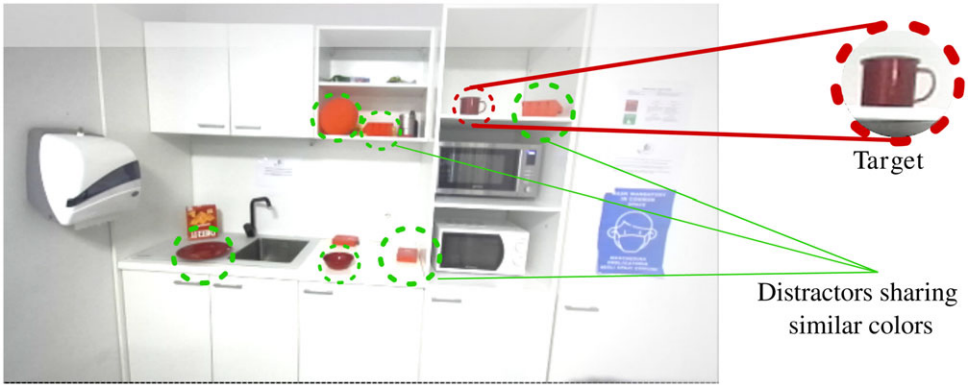


Figure 8. *Visual search: Target and distractors sharing similar color.*

2. Latency: The end-to-end latency is composed of the sub-components in the framework: (1) at the remote site - data acquisition (log-read RGB-D images), ray-casting, conversion (surfels into PCL data structure), sampling, and encoding; and (2) at the user site – decoding, conversion, and rendering. In addition, the pre-specified latency includes: (1) the eye tracker – around 8ms (120 Hz); (2) the ROSbridge network to communicate the gaze pose to the remote site – 10ms (100 Hz).
3. PSNR metric: The reduction of points degrades the visual quality of the peripheral region when rendered to the user. To quantify this, the *Point-to-Point* peak signal-to-noise ratio (PSNR) based geometry quality metric is a frequently used measure of distortion [56]. It is deemed insufficient though, as it does not consider the underlying surfaces represented by the point clouds when estimating the distortion. Further, it can be sensitive to size differences and noise when calculating the peak signal estimation. A new *volumetric density-based* PSNR metric is proposed, which utilizes two volumetric densities for the data under consideration: (1) the general volumetric density (proposed in CloudCompare [57]), computed using the number of neighbors $N_{\mathcal{P}}^{v}$ for each point $\mathfrak{p}$ in the point-cloud $\mathcal{P}$ that lie inside a spherical volume v, as seen in Eq. (14). Figure 9 visualizes the concept, where the foveated point-cloud shows higher density around the fovea region and lower density in the peripheral regions; and (2) its maximum volumetric density as the peak signal. For the peak signal, the volumetric density is calculated with the k-nearest neighbor approach, as seen in Eq. (15), to account for the distribution of the density across the point cloud and avoid any skew in the values due to sensor noise.

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press

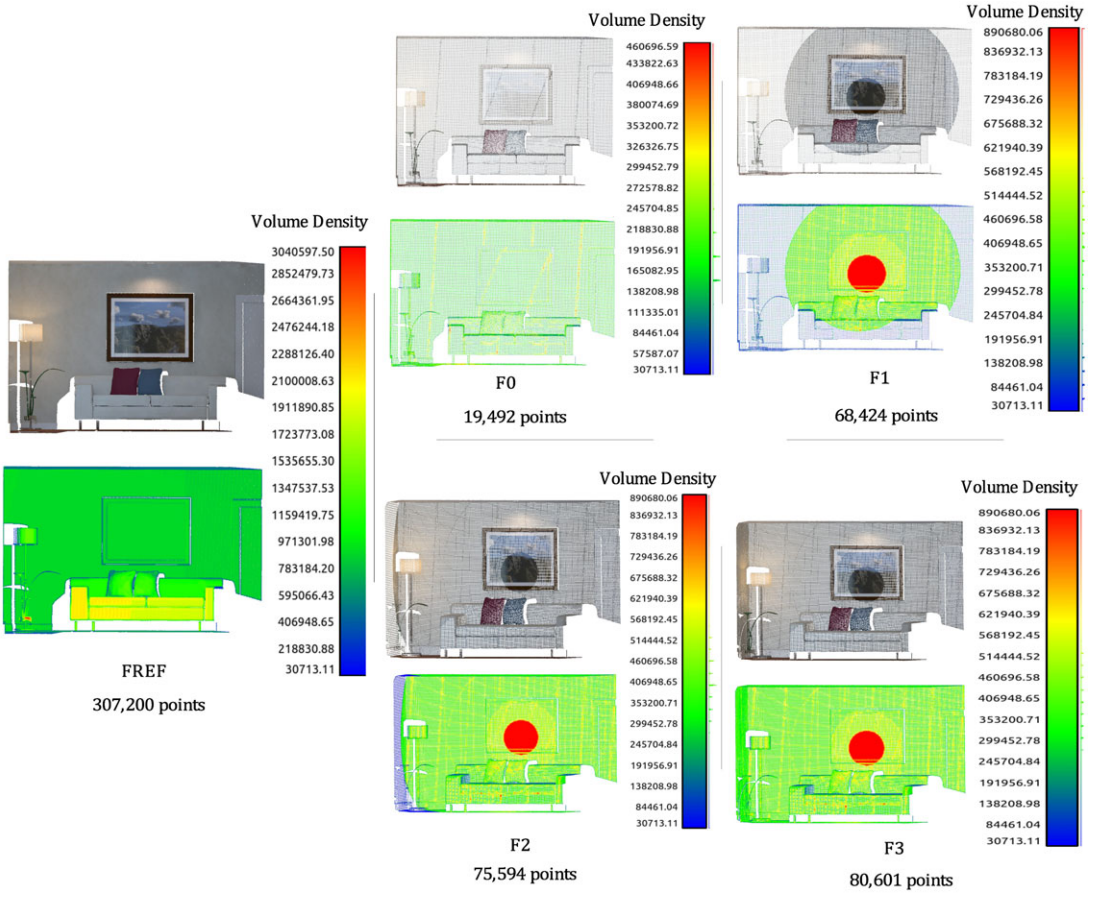


Figure 9. *The foveated conditions and the color-coded map for the volumetric density estimation for the* ***LIV*** *dataset using Eq. (14) and plotted by CloudCompare [57] using radius of* $R = 0.019809$.

$$\mathbf{vd} = \frac{1}{N_{\mathcal{P}}} \sum_{\forall \mathfrak{p} \in \mathcal{P}} \frac{N^v_{\mathcal{P}}}{\frac{4}{3} \cdot \pi \cdot R^3} \tag{14}$$

$$\mathbf{vd}^k_{\mathfrak{p} \in \mathcal{P}_1} = \frac{1}{k} \sum_{i=1}^{k} \frac{N^v_{\mathcal{P}_1}}{\frac{4}{3} \cdot \pi \cdot R^3},$$

$$\mathbf{vd}^{max}_{\mathcal{P}_1} = \underset{\forall \mathfrak{p} \in \mathcal{P}_1}{max} \left(\mathbf{vd}^k_{\mathfrak{p}} \right) \tag{15}$$

The value of $k = 10$ was found experimentally and the choice depends on the input dataset and the resolution, as data with more depth measurement errors will likely perform better when the value of k is higher. The value of the radius R, for consistency, is estimated by averaging the voxel sizes across all the foveated regions in the F3 condition (Figure 9). For the symmetric density difference calculation, for every point $\mathfrak{p}$ in the reference (original) point-cloud $\mathcal{P}_1$ (FREF), the closest corresponding point in the degraded cloud $\mathfrak{p}_{nn} \in \mathcal{P}_2$ (F0-F3) is found. The density $\mathbf{vd}_{\mathcal{P}_1}$ and $\mathbf{vd}_{\mathcal{P}_2}$ are then estimated using Eq. (14). The density-based PSNR is calculated as a ratio of the maximum density of a reference point-cloud using the experinmental condition (FREF) to the symmetric root-mean-square (rms) difference in the general densities. Eq. (16) provides the equations; $N_{\mathcal{P}_1}$ is the number of points in region $\mathcal{P}_1$.

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press

$$\mathbf{vd}^{rms}(\mathcal{P}_1, \mathcal{P}_2) = \sqrt{\frac{1}{N_{\mathcal{P}_1}} \sum_{i=1}^{N_{\mathcal{P}_1}} \left[\mathbf{vd}^{i}_{\mathcal{P}_1} - \mathbf{vd}^{i}_{\mathcal{P}_2}\right]^2}$$

$$\mathbf{vd}^{sym}(\mathcal{P}_1, \mathcal{P}_2) = max\left(\mathbf{vd}^{rms}(\mathcal{P}_1, \mathcal{P}_2)\right), \mathbf{vd}^{rms}(\mathcal{P}_2, \mathcal{P}_1))$$

$$PSNR_{\text{vd}} = 10 \cdot log_{10} \frac{(\mathbf{vd}^{max}_{\mathcal{P}_1})^2}{(\mathbf{vd}^{sym}(\mathcal{P}_1, \mathcal{P}_2))^2} \tag{16}$$

4. *Metrics for Quality of Experience* (QoE): We used the experimental conditions outlined earlier to assess research questions **RQ1** and **RQ2** comprehensively. **RQ2** specifically addresses the question: "How do various combinations of foveated regions influence the subjective quality of experience?" In this context, the independent variable is the real-time point-cloud stimulus (foveated vs. non-foveated), while the dependent variable is the capacity to detect quality degradation. Using the Double Stimulus Impairment Scale (DSIS) study approach [58] with the **LIV** dataset, subjects were first presented with the **FREF** condition, followed by a 3-second pause, and one of the altered conditions (F0-F3) following immediately after, the presentation of the altered conditions was carefully changed using the Latin Squares design approach [59]. Both FREF and the altered conditions had 450 frames and were shown for 35 s before and after the 3 s pause. The subjects were then asked to rate the second presented stimulus on a 5-point scale [58], on whether the alteration was: (5) imperceptible; (4) perceptible, but not annoying; (3) slightly annoying; (2) annoying; and (1) very annoying. The arithmetic mean opinion score (MOS) was calculated for each condition.
5. *Metrics for Visual Search:* Eye movements are influenced by various target features, including color, size, orientation, and shape, as illustrated in Figure 8. Introducing distractors that share these features with the target significantly affects performance, leading to longer response times during search tasks. Therefore, reaction time was chosen as an evaluation metric for the visual search experiment and it was evaluated using the dataset in Figure 8. Subjects were asked to search for the target within the scene. Upon locating the target, they press the motion controller trigger button, recording their reaction time. The sequence for introducing experimental conditions between **F0, F1, F2, F3, or FREF** and the target-distractor position was carefully changed using the Latin Squares design approach cited in ref. [59].

For the user study, there were 24 subjects (9 females and 15 males), aged 21–35 years and the mean age ($\mu = 28.9$ years) of the sample population had a standard deviation ($\sigma = 4.3$ years). All subjects had a 20/20 or corrected vision, and the eye tracker was calibrated for all subjects. Based on the ITU-T [58] recommendation, subjects were made familiar with the experimental setup using the **OFF** dataset, with the VR headset and the gesture controller devices. Each subject performed two trials for each test condition, within the QoE and visual search experiments. In addition, the experiment was designed as a within-subjects study, with the experimental conditions presented in a pseudo-random order to minimize carryover effects between them.

5. Results and discussion

Following a recommendation by ref. [60], five randomized HMD positions with varying distances to the center of the datasets were used for the objective metrics evaluation. Four hundred frames were tested for each HMD position from each dataset. The Shapiro–Wilk test was performed to assess the normality of the evaluation metrics across all experiments. Results indicated normal distributions for all metrics except in the case of the visual search experiment. For this experiment, The Shapiro–Wilk

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press

Table III. *Mean number of points per frame.*

	BAL	OFF	LIV	Reduction(%) 70 75 80 85 90 95
F0	39K	18K	20K	
F1	45K	51K	49K	
F2	65K	54K	69K	
F3	66K	56K	73K	
FREF	252K	307K	307K	

Table IV. *Compressed bandwidth (MBytes/sec; top row) and latency (ms; bottom row).*

	BAL	OFF	LIV	Reduction(%) 20 40 60 80 100
F0	0.78	0.49	0.25	
	198	196	190	
F1	0.80	1.02	0.50	
	226	224	223	
F2	0.97	1.03	0.55	
	235	240	242	
F3	1.03	1.22	0.74	
	256	256	257	
FREF	1.82	1.78	1.32	
	562	622	618	

test p-values were below 0.05 for all conditions (F0, F1, F2, F3, and FREF), suggesting a potential deviation from normal distribution. However, after applying a log transformation to the data, the variables exhibited a log-normal distribution. Therefore, a two-way Student's t-test was employed to analyze the log-transformed data. Likewise, the same test was applied to the other evaluation metrics.

5.1. Data transfer rate

Table III shows the relative reduction in the number of points per frame and Table IV reports the average bandwidth required for streaming (MBytes/sec), and the relative percentage reduction in bandwidth as compared to the FREF condition. From these results, it can be seen that the mean bandwidth required for the F1 condition gives an average 61% reduction as compared with FREF. The numbers are similar for the F2 condition, with an average 61% reduction, while F3 offers a lower 55% reduction. Statistical t-test analysis showed that these reductions are significant at 95% CI (p-values < 0.05). Within the three conditions, although F1 is the most advantageous, the difference between the three reductions is not statistically significant (p-value $= 0.3$). On the other hand, as expected, the foveation conditions perform worse than the F0 condition, which offers the highest bandwidth reduction, at up to 81%.

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press

Table V. *Sample comparison of averaged system components execution times per frame for the* ***OFF*** *and* ***LIV*** *datasets.*

		OFF					LIV				
	Component	**F0**	**F1**	**F2**	**F3**	**FREF**	**F0**	**F1**	**F2**	**F3**	**FREF**
Remote	RGB-D reader	4	4	4	4	4	4	4	4	4	4
	Partitioning	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
	Conversion	46	59	43	51	56	54	53	47	46	56
	Sampling	21	22	23	25	0	30	21	22	22	0
	Encoding	60	88	114	111	319	104	73	109	119	396
	Decoding	25	36	46	47	121	47	38	59	58	158
User	Conversion	0.7	1.3	1.8	2	7	1	1.2	1.6	1.6	6
	Rendering	18	18	18	18	18	14	14	14	14	14
Total(ms)		175	228	250	258	525	254	204	257	265	634

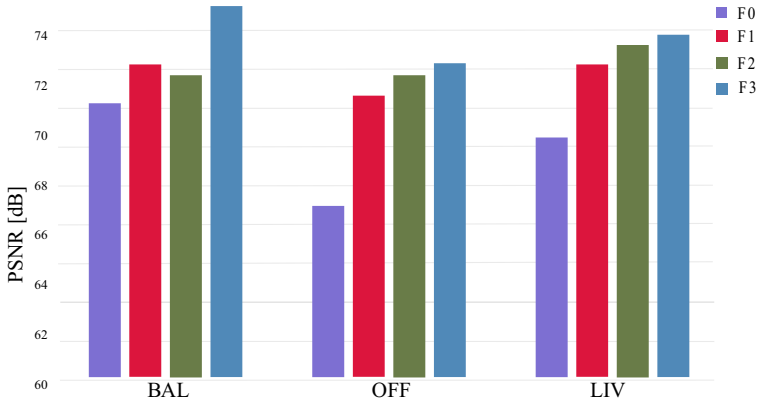


Figure 10. *Volumetric density-based peak signal-to-noise ratio for all experimental conditions and data-sets.*

5.2. Latency reduction

The mean latency values for our framework are listed in Table IV. Again, the foveation conditions offer between 60% (F3) and 67% (F1) speedup over the FREF condition. However, they also perform worse than the F0 condition, being between 20% and 35% slower. These speedups (and slowdowns) are statistically significant, p-values < 0.05. A more detailed system component level evaluation is seen in Table V. The most time-consuming elements are related to data conversion and compression. As expected, the numbers show an upward trend from F0 to FREF. It is noted that this trend is not linear - latency increases at a greater rate with increasing point-cloud density.

5.3. PSNR metric

Figure 10 illustrates the volumetric density-based PSNR metric that helps objectively discriminate between the test conditions in terms of the costs they impose on the visual quality. In all cases, the F0 PSNR is significantly worse (p-value < 0.05), which negates the bandwidth and latency advantages it offers. The foveation conditions offer progressively better PSNR values, averaging 69.5 dB (F1), 70 dB (F2), and 71.6 dB (F3), over the four datasets. The F3 PSNR is significantly better than F1 (p-value < 0.05), but not F2 (p-value $= 0.64$).

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press

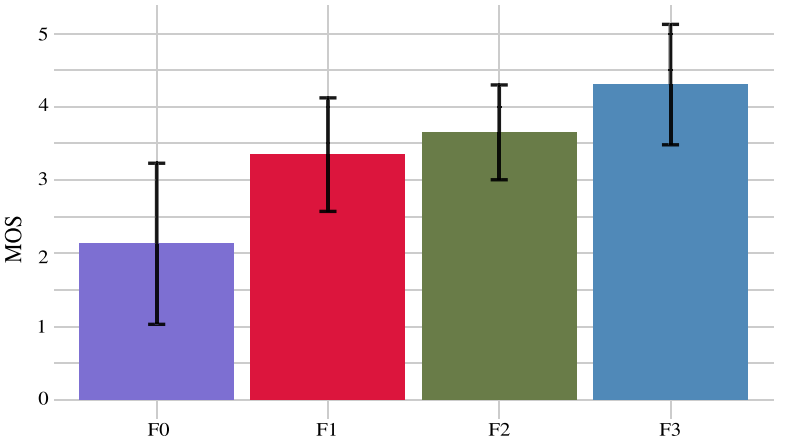


Figure 11. *Mean opinion score (MOS) and Standard deviation (SD) with error bars for the Quality of Experience (QoE) metric.*

5.4. QoE metric

Figure 11 shows the mean opinion score (MOS), averaged over the 24 subjects. It is seen that all three foveation conditions have a MOS > 3. For the F1 and F2 conditions, the foveation is certainly perceptible, but it may not hinder the users' experience, since the perceived degradation is only "slightly annoying" (F1) or "perceptible, but not annoying." With a MOS > 4, the F3 condition shows that subjects are not able to easily perceive the degradation, and even if they do, it is "not annoying." The F0 condition has a MOS < 3, implying the degradation can be annoying for subjects, which further negates the benefits it offers on the other metrics.

The Shapiro–Wilk normality tests for each condition revealed significant deviations from normality ($F0 : p = 0.00118, F1 : p = 7.179e-05, F2 : p = 0.0001979, F3 : p = 0.0001523$). Given these results, the Friedman test, which indicated significant differences between conditions ($Friedmanchi - squared = 42.347, p = 3.386e-09$). Pairwise comparisons using the Wilcoxon signed-rank test showed significant differences between the following condition pairs: F0 versus F1 ($p = 0.000237$), F0 versus F2 ($p = 0.000419$), F0 versus F3 ($p = 0.000055$), and F1 versus F3 ($p = 0.003$), while F1 versus F2 ($p = 0.137$) and F2 versus F3 ($p = 0.010$) were not significant after Bonferroni correction, with adjusted p-values of 0.001422, 0.002514, 0.000330, 0.822, 0.018, and 0.060, respectively.

5.5. Visual search metric

As shown in Figure 12, the mean and standard deviation of participants in condition **F0** gives a mean of ($\mu = 1219.6, \sigma = 883$), showing that participants were very slow and the large standard deviation indicates that the reaction time in this condition is farther away from the mean. The mean and standard deviation for the F1 condition is ($\mu = 512, \sigma = 198$), for F2 ($\mu = 442.4, \sigma = 283$), F3 ($\mu = 288.9$, $\sigma = 138.2$), and FREF ($\mu = 248.8, \sigma = 120.8$).

The Shapiro–Wilk normality test results revealed that F0 ($W = 0.86143, p - value = 0.0001218$) and F2 ($W = 0.78733, p - value = 2.405e-06$) are not normally distributed, while F1 ($W = 0.96261$, $p - value = 0.183$) is approximately normally distributed and F3 ($W = 0.92441, p - value = 0.008433$) is not normally distributed. Consequently, the Friedman test was employed, yielding a chi-squared value of 77.699 with 4 DOF and a p-value of $5.349e-16$, indicating significant differences among conditions. Post-hoc pairwise comparisons using the Wilcoxon signed-rank test, adjusted for multiple comparisons with Bonferroni correction, showed significant differences between F0 and all other conditions ($F1 : p - value = 0.00036, F2 : p - value = 0.00016, F3 : p - value = 1.3e-08, FREF : p - value = 1.7e-06$), between F1 and F3 ($p - value = 8.7e-08$) and FREF ($p - value =$

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press

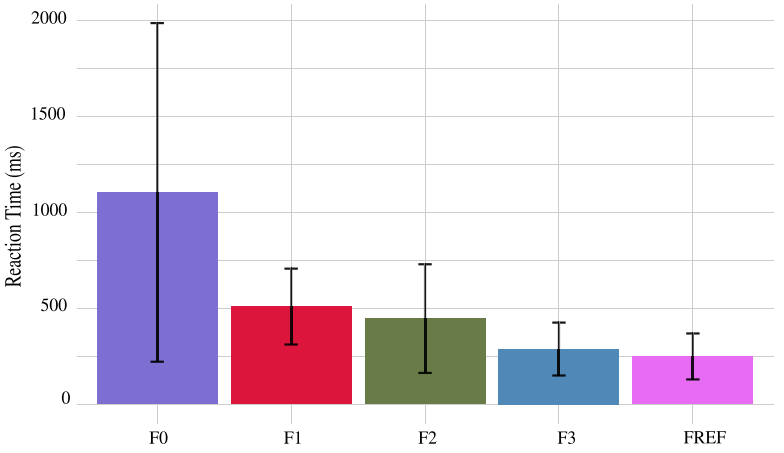


Figure 12. *Reaction time as a function of experimental conditions.*

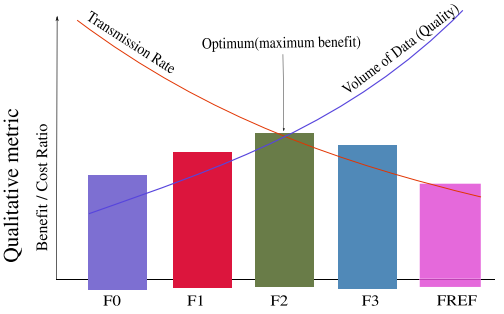


Figure 13. *Benefit-cost ratio against different conditions.*

$2.3e-06$), and between F2 and F3 ($p-value=0.00921$) and FREF ($p-value=0.00012$), with no significant differences between F1 and F2 and between F3 and FREF.

5.6. Discussion

The five metrics analyzed here offer a cost-benefit understanding of the tested conditions, that is, the benefits in bandwidth and latency versus the costs in PSNR and QoE. For instance, the F0 condition, as expected, offers the most benefit for bandwidth and latency, but the costs in PSNR, QoE, and visual search are the highest. Although the FREF condition is the best in terms of PSNR and QoE, the overall analysis demonstrates that the foveated conditions together provide the optimal cost-benefit ratio, as compared to both the F0 and FREF conditions. The perceived degradations are seen not to impact QoE significantly. A deeper analysis shows that the F3 condition performs significantly better in the benefit metrics, while its costs are not significantly worse than FREF. As expected, the F1 condition falls at the lower end within the three conditions but still offers significantly higher benefits on latency and bandwidth. The three test foveation conditions created by combining the six regions offer a key advantage: real-time usage requirements and a user-selectable approach that allows users to choose among the three conditions and switch among them as required. As shown in Figure 13, The F2 condition offers a good cost-benefit compromise between the two conditions.

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press



Figure 14. *Teleoperation experimental setup, a virtual model of the robot, and point clouds streamed in FREF and F2 experimental conditions.*

6. Verification through remote telemanipulation

To evaluate the impact of the proposed system on the performance and execution times during real remote telemanipulation tasks, we propose a pick-and-place task experiment. This experiment uses the cost-benefit analysis results from five metrics in Figure 13 and uses the F2 as optimal experimental condition, while the reference condition FREF is compared with it. This approach enables the validation of the proposed system through teleoperation experiments. The framework was then tested on a separate real-world setup, based on a dynamic pick-and-place remote telemanipulation user trial (termed as **Teleop**), as explained in the following below.

A user study was conducted in a real-world scenario and at the remote site Figure 14(a), we used a remotely teleoperated manipulator (the 7 DOF Franka Emika Panda with its gripper), along with balls at specific locations, a bowl as target drop-off point, and other objects were used for a pick-and-place task. At the user site, participants were provided with a VR interface that included a virtual model of the actual robot and the point clouds streamed in different modes as illustrated in Figure 14(b) for FREF and Figure 14(c) is the experimental conditions F2. This experimental setup allowed for a comprehensive analysis of the system's impact on user performance in the context of pick-and-place tasks. To objectively assess the effectiveness of the tasks, the execution time for each task was recorded, along with manipulation or grasping errors. These errors included inaccurate positioning, failed grasps, premature dropping of objects, or placing objects in incorrect locations.

Participants viewed the remote scene through the HTC Vive Pro Eye, and the remote area used the Intel Realsense RGB-D camera to capture the remote 3D data in real-time. The scene consisted of different objects, and included three color-coded tennis balls at specific locations, and a bowl at a target location. A sample scene is seen in Figure 14. At the "Go" signal from the experimenter, with the robot starting from a home location, the participant picked up the tennis ball, based on the requested color code (red, yellow, or blue), and placed it inside the bowl. Participants released the grasped ball, based on their judgment of the end-effector location vis-a-vis the target location. At the end of each condition (3 sequential pick-ups and drop-offs), the experimenter gave a "relax" signal to the participants. The robot end-effector was moved to the home location, and the balls and the bowl were replaced for the next condition; their locations were randomized across trials using the Latin Squares design approach. Additionally, the conditions were alternated for all subjects.

6.1. Remote telemanipulation result

Figure 15 displays the task completion time for the FREF and F2 conditions, providing valuable insights into the system's effectiveness in teleoperation tasks. The FREF condition took 84 s, while the F2 condition performed better, taking approximately 43 s each. The Student's t-test results indicated statistically significant differences in execution time (p-values < 0.05) with the FREF condition. In other words, the F2 condition was more efficient than the FREF condition in terms of task completion time. The Student's t-test results showed that there was a significant difference in execution times between the two

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press

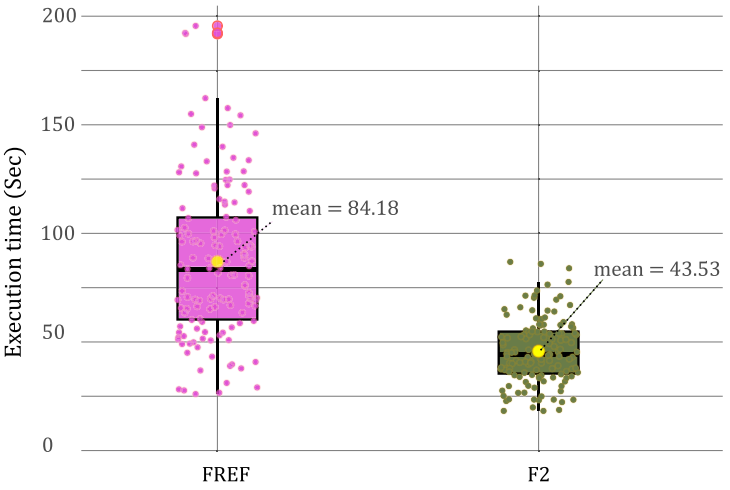


Figure 15. *Participants' execution time results for FREF and F2 conditions.*

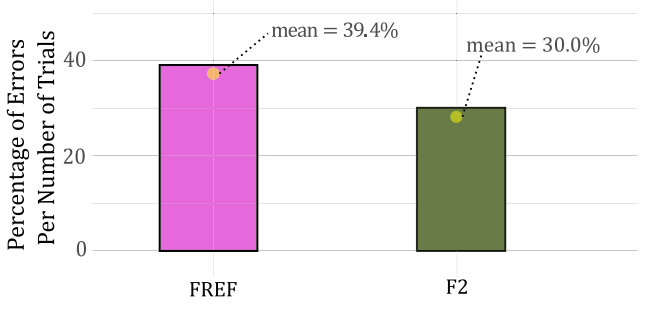


Figure 16. *Percentage of errors per number of trials in FREF and F2 conditions.*

conditions. This suggests that the F2 condition is a better choice for teleoperation tasks, as it is faster and more effective than the FREF condition.

Figure 16 presents the grasping errors observed during the study, with error rates calculated as a percentage of the total number of trials (three trials per condition). For example, if a subject made one error (such as during picking, moving, or dropping off) in one trial and had no errors in the remaining two trials, the error rate would be 33%, indicating one error out of three trials in that specific condition. Based on the collected data, the mean error rates were calculated as follows: FREF = 39.4% and F2 = 30.3%. Statistical analysis using the Wilcoxon rank-sum test, which is appropriate for non-normally distributed data, revealed a significant difference between the FREF and F2 conditions. These findings indicate that the error rates for FREF are higher.

7. Conclusion

This work presented a systematic approach that successfully leverages the vast potential of immersive remote teleoperation interfaces and the human visual system for foveated sampling, streaming, and intuitive robot control that allows the user to optimize bandwidth and latency requirements without sacrificing the quality of experience. In addition, the proposed work enables users to freely explore remote environments in VR to understand better, view, locate, and interact with the remote environment to make it adaptable for demanding telerobotic domains, for example, disaster response, nuclear decommissioning, telesurgery, etc.

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press

The experimental results reveal significant enhancements in visual search experiments. Moreover, latency and throughput were reduced by more than 60% and 40%, respectively. Furthermore, a user study focusing on remote telemanipulation showed that the framework has minimal impact on the users' visual quality of experience while significantly enhancing task performance. The users reported no compromise in the quality while experiencing improved efficiency in task execution. The proposed framework demonstrates its substantial benefits by reducing task execution time and highlighting its effectiveness and practicality in real-world scenarios. Although the QoE scores do not reflect it, future investigations will prioritize addressing distortion and aliasing issues resulting from discontinuities at region boundaries and over-sampling in the peripheral regions. Further user trials in contextual tasks with end-users will help establish the utility and suitability of the framework in real-world applications.

Author contributions. YT developed the algorithm and research, conducted the robotic experiment, analyzed outcomes, and authored the paper. KY conducted the robotic experiment and formulated the robot's control strategy. SA initiated brainstorming sessions and successfully secured funding. PF supervised the research, designed the experiment, and provided supervision. ND supervised the research, designed the experiment, reviewed ethical approval, and contributed to paper writing. DC supervised the research, contributed to paper writing, facilitated funding acquisition, and offered technical insights on the experiment. Together, all authors collaborated on manuscript revisions and content development, contributed to the article, and approved the submitted version.

Financial support. This research is supported by and in collaboration with the Italian National Institute for Insurance against Accidents at Work, under the project "Sistemi Cibernetici Collaborativi – Robot Teleoperativo 3," and supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2022R1A6A3A03069040).

Competing interests. The authors declare no conflict of interest.

Ethical approval. Not applicable.

References

[1] S. Cobos-Guzman, J. Torres and R. Lozano, "Design of an underwater robot manipulator for a telerobotic system," *Robotica* **31**(6), 945–953 (2013).
[2] S. Hokmi, S. Haghi and A. Farhadi, "Remote monitoring and control of the 2-DoF robotic manipulators over the internet," *Robotica* **40**(12), 4475–4497 (2022).
[3] G. G. Muscolo, S. Marcheschi, M. Fontana and M. Bergamasco, "Dynamics modeling of human-machine control interface for underwater teleoperation - erratum," *Robotica* **40**(4), 1255–1255 (2022).
[4] A. Naceri, D. Mazzanti, J. Bimbo, Y. T. Tefera, D. Prattichizzo, D. G. Caldwell, L. S. Mattos and N. Deshpande, "The *Vicarios* Virtual reality interface for remote robotic teleoperation," *J. Intell. Robot. Syst.* **101**(80), 1–16 (2021).
[5] Y. Tefera, D. Mazzanti, S. Anastasi, D. Caldwell, P. Fiorini and N. Deshpande, "Towards Foveated Rendering for Immersive Remote Telerobotics," **In:** *The International Workshop on Virtual Augmented, and Mixed-Reality for Human-Robot Interactions at HRI* 2022, Boulder, USA (2022) pp. 1–4.
[6] M. Mallem, F. Chavand and E. Colle, "Computer-assisted visual perception in teleoperated robotics," *Robotica* **10**(2), 93–103 (1992).
[7] A. Mossel and M. Kröter, "Streaming and Exploration of Dynamically Changing Dense 3D Reconstructions in Immersive Virtual Reality," **In:** *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, IEEE, Merida, Mexico (2016) pp. 43–48.
[8] E. Slawiñski and V. Mut, "Control scheme including prediction and augmented reality for teleoperation of mobile robots," *Robotica* **28**(1), 11–22 (2010).
[9] P. Stotko, S. Krumpen, M. Schwarz, C. Lenz, S. Behnke, R. Klein and M. Weinmann, "A VR System for Immersive Teleoperation and Live Exploration with a Mobile Robot," **In:** *IEEE/RSJ IROS*, Macau, China (2019) pp. 3630–3637.
[10] P. Stotko, S. Krumpen, M. B. Hullin, M. Weinmann and R. Klein, "SLAMCast: Large-scale, real-time 3D reconstruction and streaming for immersive multi-client live telepresence," *IEEE Trans. Vis. Comput. Graph.* **25a**(5), 2102–2112 (2019).
[11] M. Kamezaki, J. Yang, H. Iwata and S. Sugano, "A basic framework of virtual reality simulator for advancing disaster response work using teleoperated work machines," *J. Robot. Mechatron.* **26**(4), 486–495 (2014).
[12] E. Dima, K. Brunnström, M. Sjöström, M. Andersson, J. Edlund, M. Johanson and T. Qureshi, "View Position Impact on QoE in an Immersive Telepresence System for Remote Operation," **In:** *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*, IEEE, Berlin, Germany (2019) pp. 1–3.

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press

[13] E. Rosen, D. Whitney, M. Fishman, D. Ullman and S. Tellex, "Mixed Reality as a Bidirectional Communication Interface for Human-Robot Interaction," **In:** *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA (2020) pp. 11431–11438.

[14] J.-P. Stauffert, F. Niebling and M. E. Latoschik, "Latency and cybersickness: Impact, causes, and measures. A review," *Front. Virtual Real.* **1**, 31 (2020).

[15] S. Orts-Escolano, C. Rhemann, S. Fanello, W. Chang, A. Kowdle, Y. Degtyarev, D. Kim, P. L. Davidson, S. Khamis, M. Dou, V. Tankovich, C. Loop, Q. Cai, P. A. Chou, S. Mennicken, J. Valentin, V. Pradeep, S. Wang, S. B. Kang, P. Kohli, Y. Lutchyn, C. Keskin and S. Izadi. "Holoportation: Virtual 3D Teleportation in Real-Time," **In:** *29th Annual Symposium on User Interface Software and Technology (UIST)*, New York, NY, USA, Association for Computing Machinery (2016) pp. 741–754.

[16] A. Hendrickson, "Organization of the Adult Primate Fovea," **In:** *Macular Degeneration* (Penfold P. L. and Provis J. M, eds.) (Springer Berlin Heidelberg, Berlin, Heidelberg, 2005) pp. 1–23.

[17] B. Guenter, M. Finch, S. Drucker, D. Tan and J. Snyder, "Foveated 3D graphics," *ACM Trans. Graph.* **31**(6), 1–10 (2012).

[18] A. Maimone and H. Fuchs, "Encumbrance-Free Telepresence System with Real-Time 3D Capture and Display Using Commodity Depth Cameras," **In:** *10th IEEE International Symposium on Mixed and Augmented Reality*, IEEE, Basel, Switzerland (2011) pp. 137–146.

[19] D. Ni, A. Song, X. Xu, H. Li, C. Zhu and H. Zeng, "3D-point-cloud registration and real-world dynamic modelling-based virtual environment building method for teleoperation," *Robotica* **35**(10), 1958–1974 (2017).

[20] A. J. Fairchild, S. P. Campion, A. S. García, R. Wolff, T. Fernando and D. J. Roberts, "A mixed reality telepresence system for collaborative space operation," *IEEE Trans. Circ. Syst. Vid. Technol.* **27**(4), 814–827 (2017).

[21] M. Weinmann, P. Stotko, S. Krumpen and R. Klein, "Immersive VR-Based Live Telepresence for Remote Collaboration and Teleoperation," **In:** *Wissenschaftlich-Technische Jahrestagung der DGPF*, Stuttgart, Germany (2020) pp. 391–399.

[22] Y.-P. Su, X.-Q. Chen, C. Zhou, L. H. Pearson, C. G. Pretty and J. G. Chase, "Integrating virtual, mixed, and augmented reality into remote robotic applications: A brief review of extended reality-enhanced robotic systems for intuitive telemanipulation and telemanufacturing tasks in hazardous conditions," *Appl. Sci.* **13**(22), 12129 (2023).

[23] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D.Freemanand A. Davison, "Kinectfusion: Real-Time 3D Reconstruction and Interaction Using a Moving Depth Camera," **In:** *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, Association for Computing Machinery, New York, NY, USA (2011) pp. 559–568.

[24] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker and A. Davison, "ElasticFusion: Dense SLAM without a pose graph," *In Robot.: Sci. Syst.* **11**, 1697–1716 (2015).

[25] T. Schöps, T. Sattler and M. Pollefeys, "SurfelMeshing: Online surfel-based mesh reconstruction," *IEEE Trans. Pattern Anal. Mach. Intell.* **42**(10), 2494–2507 (2020).

[26] R. Mekuria, K. Blom and P. Cesar, "Design, implementation, and evaluation of a point cloud codec for tele-immersive video," *IEEE Trans. Circ. Syst. Vid. Technol.* **27**(4), 828–842 (2016).

[27] S. Schwarz, N. Sheikhipour, V. F. Sevom and M. M. Hannuksela, "Video coding of dynamic 3D point cloud data," *APSIPA Trans. Signal Info. Process.* **8**(1), 31–43 (2019).

[28] Y. Huang, J. Peng, C.-C. J. Kuo and M. Gopi, "A generic scheme for progressive point cloud coding," *IEEE Trans. Vis. Comput. Graph.* **14**(2), 440–453 (2008).

[29] Y. Shi, P. Venkatram, Y. Ding and W. T. Ooi, "Enabling Low Bit-Rate mpeg v-pcc Encoded Volumetric Video Streaming with 3D Sub-Sampling," **In:** *Proceedings of the 14th Conference on ACM Multimedia Systems*, New York, NY, USA (2023) pp. 108–118.

[30] J. Van Der Hooft, T. Wauters, F. De Turck, C. Timmerer and H. Hellwagner, "Towards 6DoF http Adaptive Streaming Through Point Cloud Compression," **In:** *Proceedings of the 27th ACM International Conference on Multimedia*, New York, NY, USA (2019) pp. 2405–2413.

[31] F. De Pace, G. Gorjup, H. Bai, A. Sanna, M. Liarokapis and M. Billinghurst, "Leveraging Enhanced Virtual Reality Methods and Environments for Efficient, Intuitive, and Immersive Teleoperation of Robots," **In:** *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, Xi'an, China (2021) pp. 12967–12973.

[32] E. Huey. *The Psychology and Pedagogy of Reading: With a Review of the History of Reading and Writing and of Methods, Texts, and Hygiene in Reading*, M.I.T. Press paperback series (M.I.T. Press, New York, NY, USA, 1968).

[33] P. M. Fitts, R. E. Jones and J. L. Milton, "Eye movements of aircraft pilots during instrument-landing approaches," *Aeronaut. Eng. Rev.* **9**(2), 24–29 (1949).

[34] A. L. Yarbus, *Eye Movements and Vision* (Springer Berlin/Heidelberg, Germany, 1967).

[35] N. Stein, D. C. Niehorster, T. Watson, F. Steinicke, K. Rifai, S. Wahl and M. Lappe, "A comparison of eye tracking latencies among several commercial head-mounted displays," *i-Perception* **12**(1), 2041669520983338 (2021).

[36] M. Stengel, S. Grogorick, M. Eisemann and M. Magnor, "Adaptive image-space sampling for gaze-contingent real-time rendering," *Comput. Graph. Forum* **35**(4), 129–139 (2016).

[37] V. Bruder, C. Schulz, R. Bauer, S. Frey, D. Weiskopf and T. Ertl, "Voronoi-Based Foveated Volume Rendering," **In:** EuroVis (Short Papers) pp. 67–71 (2019).

[38] A. Charlton, What is foveated rendering? Explaining the VR technology key to lifelike realism (2021) (Accessed: 05-Sep-2021).

[39] A. S. Kaplanyan, A. Sochenov, T. Leimkühler, M. Okunev, T. Goodall and G. Rufo, "Deepfovea: Neural reconstruction for foveated rendering and video compression using learned statistics of natural videos," *ACM Trans. Graph.* **38**(6), 1–13 (2019).

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press

[40] P. Lungaro, R. Sjöberg, A. J. F. Valero, A. Mittal and K. Tollmar, "Gaze-aware streaming solutions for the next generation of mobile VR experiences," *IEEE Trans. Vis. Comput. Graph.* **24**(4), 1535–1544 (2018).
[41] T. Ananpiriyakul, J. Anghel, K. Potter and A. Joshi, "A gaze-contingent system for foveated multiresolution visualization of vector and volumetric data," *Electron. Imaging* **32**(1), 374-1–374-11 (2020).
[42] M. Schütz, K. Krösl and M. Wimmer, "Real-Time Continuous Level of Detail Rendering of Point Clouds," **In:** *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, Osaka, Japan (2019) pp. 103–110.
[43] A. C. Guyton and J. E. Hall, *Guyton and Hall Textbook of Medical Physiology* (Elsevier, New York, NY, USA, 2011) pp. 597–608.
[44] N. Quinn, L. Csincsik, E. Flynn, C. A. Curcio, S. Kiss, S. R. Sadda, R. Hogg, T. Peto and I. Lengyel, "The clinical relevance of visualising the peripheral retina," *Prog. Retin. Eye Res.* **68**, 83–109 (2019).
[45] W. R. Sherman and A. B. Craig, "Chapter 3 - the Human in the Loop," **In:** *Understanding Virtual Reality*, The Morgan Kaufmann Series in Computer Graphics, (W. R. Sherman and A. B. Craig, eds.) (Second Edition) (Morgan Kaufmann, Boston, 2018) pp. 108–188.
[46] J. Hyönä, "Foveal and Parafoveal Processing during Reading," **In:** *The Oxford Handbook of Eye Movements* (S. P. Liversedge, I. Gilchrist and S. Everling, eds.) (Oxford University Press, New York, NY, USA, 2011) pp. 820–838.
[47] Y. Ishiguro and J. Rekimoto, "Peripheral Vision Annotation: Noninterference Information Presentation Method for Mobile Augmented Reality," *In Proceedings of the 2nd Augmented Human International Conference*, New York, NY, USA (2011) pp. 1–5.
[48] H. Strasburger, I. Rentschler and M. Jüttner, "Peripheral vision and pattern recognition: A review," *J. Vision* **11**(5), 13–13 (2011).
[49] M. J. Simpson, "Mini-review: Far peripheral vision," *Vision Res.* **140**, 96–105 (2017).
[50] J. Gordon and I. Abramov, "Color vision in the peripheral retina II Hue and saturation," *JOSA* **67**(2), 202–207 (1977).
[51] F. W. Weymouth, "Visual sensory units and the minimal angle of resolution," *Am. J. Ophthalmol.* **46**(1), 102–113 (1958).
[52] M. P. Eckstein, "Visual search: A retrospective," *J. Vision* **11**(5), 14–14 (2011).
[53] B. Olk, A. Dinu, D. J. Zielinski and R. Kopper, "Measuring visual search and distraction in immersive virtual reality," *Roy. Soc. Open. Sci.* **5**(5), 172331 (2018).
[54] A. Handa, T. Whelan, J. B. McDonald and A. J. Davison, "A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM," **In:** *IEEE International Conference on Robotics and Automation, ICRA*, Hong Kong, China (2014) pp. 1524–1531.
[55] C. Sanders, *Practical Packet Analysis: Using Wireshark to Solve Real-World Network Problems* (No Starch Press, San Francisco, CA, USA, 2017) pp. 1–155.
[56] M. 3DG and Requirements 2017. Call for Proposals for Point Cloud Compression V2. Technical report, MPEG 3DG and Requirements, Hobart, AU.
[57] D. Girardeau-Montaut, "Cloudcompare-open source project," *OpenSource Project*, Paris, France **588** (2011) pp. 2–37.
[58] International Telecommunication Union, *Recommendation ITU-T P.919: Subjective Test Methodologies for 360° Video On Head-Mounted Displays* (ITU, Geneva, Switzerland, 2020), pp. 1–38.
[59] J. T. Richardson, "The use of latin-square designs in educational and psychological research," *Educ. Res. Rev.* **24**, 84–97 (2018).
[60] V. Bruder, C. Müller, S. Frey and T. Ertl, "On evaluating runtime performance of interactive visualizations," *IEEE Trans. Vis. Comput. Graph.* **26**(9), 2848–2862 (2019).

Cite this article: Y. T. Tefera, Y. Kim, S. Anastasi, P. Fiorini, D. G. Caldwell and N. Deshpande (2024). "Immersive remote telerobotics: foveated unicasting and remote visualization for intuitive interaction", Robotica **42**, 4223–4248. https://doi.org/10.1017/S0263574724001784

https://doi.org/10.1017/S0263574724001784 Published online by Cambridge University Press