2026 IEEE/SICE International Symposium on System Integration (SII)
January 11-14, 2026. Cancún, México

RGB-D Fusion for Wide Field of View User Feedback in Teleoperation Context

Raphaël d'Orfani[1], Antoine N. André[1], Mehdi Benallegue[1], Rafael Cisneros-Limón[1], Guillaume Caron[1,2]

Abstract— **Effective teleoperation involves immersive and responsive visual feedback to support depth perception and spatial understanding to achieve precise control. Standard camera views naturally constrain the operator's Field of View (FoV) of the remote scene, especially in cluttered or dynamic scenarios. We present a real-time RGB-D fusion system that expands the operator's FoV by employing immersive 3D reconstruction. Our system incorporates the Azure Kinect sensor into Unreal Engine using the Robot Operating System (ROS) communication, rendering live sensor information onto a spherical mesh. This allows for smooth, wide-FoV rendering of the scene with greater peripheral context and depth continuity. In contrast to planar or depth-free systems, the proposed method is enhanced by live depth retranscription for more interactive teleoperation, leading to better scene understanding. This architecture lays the basis for flexible, high-fidelity remote interaction for robotics applications. All our developments and implementations are publicly available at `https://github.com/isri-aist/RGB-D_Fusion`.**

Index Terms— **Wide field of view, RGB-D camera, sensor fusion, teleoperation.**

I. Introduction

Teleoperated robots play a key role in hazardous or remote tasks, but their effectiveness hinges on the operator's visual immersion [1]. The success of such systems relies heavily on the quality of the visual feedback presented to the operator. Standard monocular or planar camera screens tend to limit depth perception and situational awareness—both essential for secure and precise teleoperation and for environment understanding. To overcome these limitations, immersive teleoperation interfaces using color cameras coupled with depth sensors (RGB-D) and Virtual or Augmented reality (VR/AR) have been proposed [2], [3]. Such approaches therefore allow for precise real robot teleoperation in various challenging contexts [4].

RGB-D sensors and their ability to provide both color (RGB) and depth, enable richer environmental reconstructions. Vuong et al. [5] used this approach to facilitate imitation-based manipulation, while Rolley-Parnell et al. [6] demonstrated the potential of RGB-D for fine-grained bimanual teleoperation. These techniques significantly improve perception through the usage of a dynamically deformed 3D object provided to the user for better visual feedback while demonstrating the importance of multimodal sensory integration in teleoperation context.

[1]CNRS-AIST JRL (Joint Robotics Laboratory), IRL3218, National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Japan. raphael.dorfani@gmail.com, {antoine.andre, mehdi.benallegue, rafael.cisneros}@aist.go.jp

[2]MIS laboratory, University of Picardie Jules Verne, Amiens, France. guillaume.caron@u-picardie.fr

At the same time, VR engines such as Unreal Engine or Unity find new applications with robotic systems, thanks to the ROS middlewares that allow to bring simulation and control to a realistic 3D environment. This integration enables the creation of immersive environments where the robot and its surroundings can be visualized interactively. Zaman et al. [7] presented a VR architecture for teleoperation based on the ROS-Unreal coupling. Softwares like rclUE[1] have emerged to facilitate such communications between the two systems. Naceri et al. [8] have explored this coupling to reconstruct static environments in Unreal for robot control, with a focus on scene fidelity and direct manipulation. Other recent studies such as Stotko et al. [9] and Zhao et al. [10] also explore immersive VR/AR interfaces for remote robotic control, highlighting the critical role of high-quality feedback.

A related immersive visualization approach was proposed by NimBro for the ANA Avatar XPRIZE competition[2], relying on low-latency spherical rendering for 6D immersive televisualization [11]. Earlier SLAM-based meshing systems also aimed to improve operator awareness [12], although without leveraging the same immersive rendering pipeline. While most of these approaches could effectively fuse multiple modalities to provide more immersion to teleoperators, none can yet offer single-shot real-time large FoV immersive view of the perceived environment.

Besides, sparsity and non-uniformity in depth data can result in incomplete reconstructions that compromise their operational usability. This work endeavors to complete that deficit with a combined and robust solution.

This work builds upon these immersive visualization approaches by introducing real-time RGB-D streaming and enhanced interactivity with the real environment. To validate our approach, we compare it against a planar RGB-only baseline setup adapted from [13], where visual feedback is displayed on a virtual screen within the headset. We thus propose the following contributions:

- A teleoperation targeted visual system based on wide-FoV RGB-D sensor fusion within an immersive VR interface.
- A depth densification and processing pipeline, providing rich and consistent 3D information to the user.
- A performance evaluation of the proposed method based on both quantitative task metrics and qualitative feedback collected from 20 volunteer participants.

[1]https://github.com/rapyuta-robotics/rclUE
[2]https://www.xprize.org/prizes/avatar

II. Methodology

Most RGB-D sensors, such as the Azure Kinect or Orbbec Femto Bolt, have different fields of view for color and depth streams. This is undesirable for 3D immersive rendering. The proposed teleoperation feedback setup integrates an Azure Kinect RGB-D sensor within Unreal Engine 5.1 using ROS 2 Humble to supply immersive wide FoV visualization in real-time. This section describes the data acquisition pipeline, stream fusion, 3D mesh construction, displacement mapping, and final rendering within a VR scene. The outcome is a real-time mesh featuring parallax occlusion and volume displacement from the user's perspective, greatly adding to immersion and spatial understanding in the VR environment.

A. Data Acquisition

Using the Azure Kinect SDK[3] to capture synchronized RGB, depth, and infrared (IR) streams, we provide a multi-modal view of the remote scene. The ROS 2 node provided by Microsoft streams all modalities[4], along with full intrinsic and extrinsic calibration data. This setup ensures both visual richness and temporal responsiveness—critical for immersive, real-time teleoperation.

The RGB camera offers full-color high-definition images with a resolution of 2048×1536 pixels, and a $90° \times 74.3°$ FoV. The depth sensor, based on Time-of-Flight technology, offers a much wider FoV of $120° \times 120°$ with a 1024×1024 pixels resolution. The IR stream further provides complementary intensity data useful in low-visibility conditions.

This multimodal acquisition has a couple of benefits over human eye acquisition. While most VR headsets offer a FoV of approximately $120°$ horizontally and $60°$ vertically, this is still narrower than the full extent of human peripheral vision—particularly in the horizontal plane, which can reach up to $200°$. However, VR still provides a significantly wider and more immersive view compared to conventional screens, supporting better situational awareness during teleoperation.

B. Depth Densification

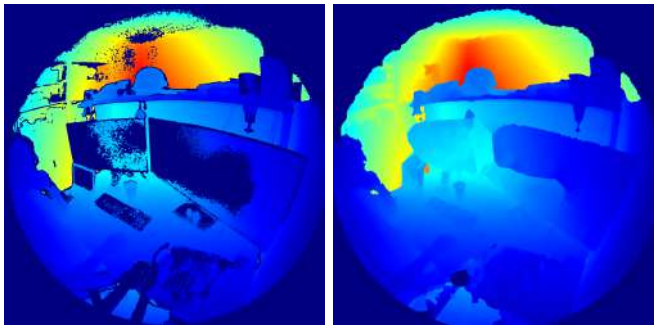
Depth maps obtained from the Azure Kinect sensor are relatively sparse, especially near object boundaries, reflective surfaces, and occluded regions (see Fig. 1a). These discontinuities—in the form of noisy or missing depth values—can significantly deteriorate downstream mesh reconstruction and displacement applied to mesh vertices. To alleviate these limitations, the proposed method includes a dedicated real-time depth map densification step immediately following acquisition via a hybrid multi-scale interpolation approach [14].

The depth data is treated as a 2D projection for pixel-wise filtering and interpolation, but all operations preserve the geometric consistency of the 3D scene through the sensor's intrinsic parameters. We first apply a multistage filtering and inpainting[5] process to the depth image acquired through the Azure Kinect ROS node, using OpenCV functions:

[3]https://github.com/microsoft/Azure-Kinect-Sensor-SDK

[4]https://github.com/microsoft/Azure_Kinect_ROS_Driver

[5]Inpainting is a restoration technique used to reconstruct an image in damaged, missing, or deteriorated areas with the aim of producing a visually complete image.



(a) Initial Depth (b) Filled Depth

Fig. 1: Azure Kinect Depth map.

1) Region-Based Filtering: The scene is split into three depth ranges: close (0.1 m - 15 m), medium (15 m - 30 m), and distant ($\geq$ 30 m). A different structuring element[6] for morphological operations is assigned to each range—small for nearby objects, large for distant objects—to adapt the smoothing scale to spatial resolution and depth noise levels.

2) Multi-Scale Dilation & Merge: The valid pixels within each segment are dilated to propagate available values into sparsely filled regions. The results are then combined back into one map by compositing the near, medium, and far channels in order of availability.

3) Morphological Closing & Temporal Smoothing [15]: In order to enhance surface continuity, a morphology operation is applied, allowing to close small gaps, leading to the obtention of a dense map. It is followed by a median filter that keeps suppressing outliers without impacting structural edges. An optional bilateral filter enhances local smoothness without impacting discontinuities, especially at object contours.

4) Top Mask Completion: A custom mask is built to identify top areas of the image with no valid measurements, which typically arise from occlusion or low IR reflectivity. These are filled in iteratively through flood-filling and conditional dilation. The purpose is to avoid empty areas that would disrupt visual immersion or distort geometry.

5) Inpainting Pass: The rest of the holes are filled using Navier-Stokes-based inpainting [16], where the mask specifying the outer edges of sparse regions is used as a constraint. This technique propagates local structure into missing areas while preserving shape gradients.

6) Final Clamping and Export: The output is post-processed to remove out-of-bound values by applying a threshold to clamp the depth range to physical limits (≥ 0 in our case). All zero-depth values are filled with a scene-consistent maximum depth. The final map is converted back to a raw buffer and transferred to two output queues: one for real-time displacement mapping and one for RGB-D alignment.

This processing pipeline significantly improves the quality and spatial coherence of the depth map (see Fig. 1b). A brief ablation study was conducted by comparing the full

[6]A structuring element is a small, pre-specified shape (e.g., matrix of pixels) used as a probe to analyze or transform an image.

densification pipeline against raw and single-pass filtered depth maps. The complete multi-stage approach reduced the proportion of missing pixels and yielded smoother geometry continuity, confirming its added value for immersive visualization.

Combined altogether, the RGB, depth, and IR streams form the foundation for a more immersive and high-data-rate remote visualization. As part of the VR visual feedback, the wide FoV and multimodal sensing allow the user to naturally observe the scene, with peripheral and depth information present alongside what is normally visible with the unaided eye.

C. Merging of Streams

To achieve a spatially coherent alignment between color and depth modalities, the RGB stream is first projected onto the depth frame reference using the Azure Kinect SDK, which allows to account for the intrinsic and extrinsic parameters of each camera. This information is then extracted and saved for future use as a calibration file.

This calibration file is then loaded at runtime to recover the calibration and transformation context, allowing one to perform color-to-depth reprojection both accurately and efficiently, using only the precomputed parameters. As a result, each RGB pixel is accurately registered to its corresponding depth value, enabling pixel-wise fusion of the two modalities and staging the data for subsequent 3D rendering and mesh building processes.

This registration process is necessary to provide geometric consistency across the streams, especially as the depth map is later used to displace or shape 3D geometry based on color information. It also enables the generation of multimodal textures (e.g., RGB + IR) with spatial coherence.

In addition to depth and RGB, the IR stream—captured using the same optical path as the depth—provides complementary radiometric data to assist with geometry augmentation in low-texture or low-light regions. The IR intensity values are directly mapped as texture onto the reconstructed geometry, improving visual continuity where RGB information is unreliable.

D. Construction of the Sphere

In order to accurately model the FoV and the projection geometry of the Azure Kinect RGB-D camera, a portion of spherical mesh is built using the camera specifications as the recipient for all considered modalities' streams. The mesh is designed to spatially match the angular range of the camera so that immersive rendering can be achieved whereby all image pixels are accurately assigned to their respective 3D locations on the surface of the sphere.

The mesh is generated using a recursively subdivided icosahedron, i.e. a polyhedron with 20 equilateral triangular faces. Each level of subdivision introduces increased vertex density and allows for a more accurate resolution of the resulting mesh, thus yielding higher precision in depth projection and texture mapping (see Fig. 2), while keeping a regular span of vertices on the mesh.

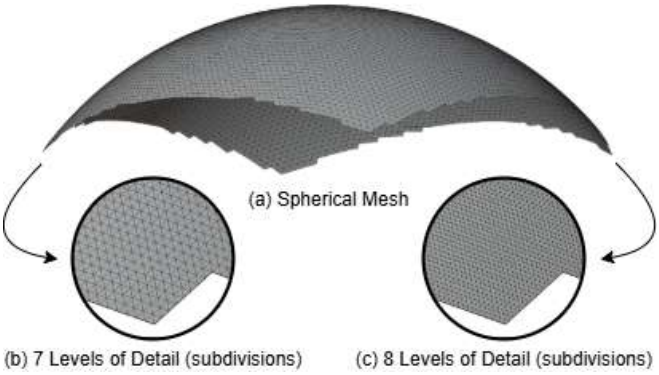


Fig. 2: Spherical Mesh.

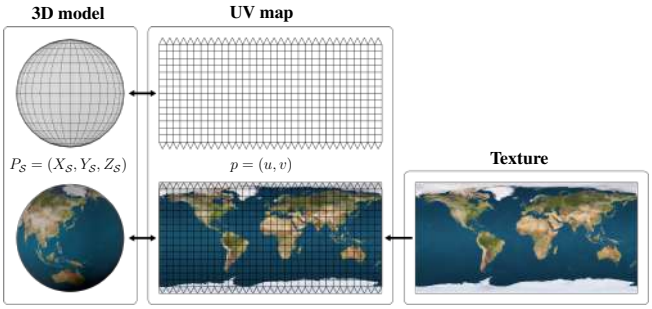


Fig. 3: Projection of pixels from a 2D Image to a 3D Sphere.

To project the pixel coordinates of the camera onto the same locations on the mesh, the intrinsic calibration parameters of the depth camera are used. These include the focal lengths f_x, f_y and the principal point coordinates c_x, c_y. Using a standard pinhole camera model, any image plane pixel x, y is projected onto the 3D unit sphere using the following mapping, with u, v the indexes of vertices on the UV Map[7]:

$$\begin{cases} u = f_x \times x + c_x \\ v = f_y \times y + c_y \end{cases} \quad (1)$$

While (1) illustrates a simplified pinhole model for conceptual clarity, the actual Azure Kinect cameras (RGB and Depth) rely on a rational polynomial distortion model. Consequently, forward and inverse projections are handled using the full factory calibration model, via SDK calls, rather than analytical expressions. This projection ensures that each 2D pixel of the image maps onto a correct physical location on the surface of the mesh, as presented in Fig. 3. Once the projection is complete, the mesh that contains vertex positions, normals, and UV mappings is saved in PLY (Polygon File Format), allowing it to store user-defined attributes and seamlessly interoperate with off-the-shelf 3D processing software. Before being converted to a format readable by Unreal Engine (FBX), the quality of the mesh can be assessed using other 3D rendering software such as Blender to improve details by adding more subdivisions. The completed mesh, along with its registration to the Azure Kinect's optical geometry, forms a structural base for the following displacement mapping and VR visualization.

[7]UV mapping is a technique of 3D modeling where the surface of a model is unwrapped and laid out on a 3D plane with the intent of applying texture.

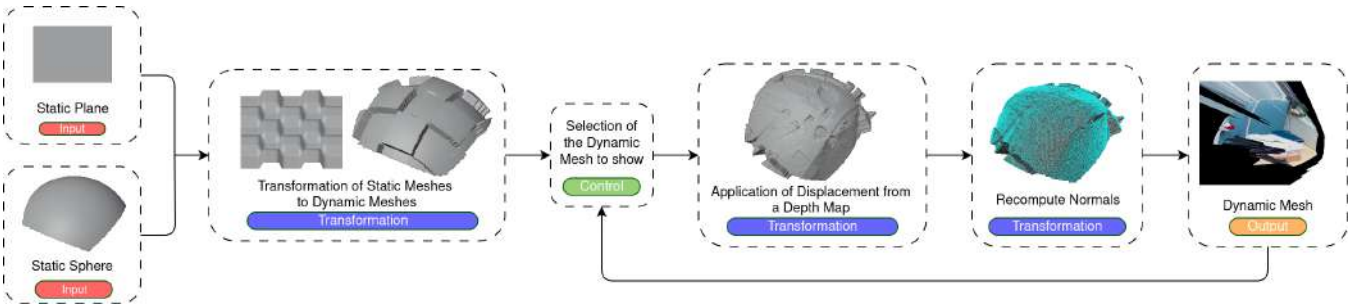


Fig. 4: Dynamic Mesh Blueprint.

E. Displacement Mapping

In order to achieve a credible illusion of spatial depth and structure in the virtual world, we employ the calculated depth map from the Azure Kinect camera to perform a displacement mapping operation on the previously mentioned spherical mesh. This process enables the transformation of a spherical textured surface into a pseudo-3D object, where surface relief and geometry vary according to the distance viewed of scene elements from the camera. The result is an increase in the sense of immersion, as it allows the user to view realistic spatial distortions and proximity of objects in relation to the recorded environment while keeping the wide FoV provided by the camera.

The displacement itself is done through a dedicated node which takes as an input the preprocessed depth map—already converted and completed from the ROS 2 acquisition pipeline (see Sect. II-B)—and utilizes it as a displacement texture (see Fig. 4). The node queries the grayscale value of the texture at each UV coordinate based on a vertex of the dynamic mesh, where each sample represents a relative depth value, which is then used to shift the corresponding vertex outward (or inward) along its normal vector.

By integrating this approach, the mesh subsequently expands from a simple sphere mapping into a volume that holds authentic geometric variation. VR users are able to observe parallax, occlusion, and depth coherence across the surface, contributing significantly to realism. This approach, based on the single point of view, simulates stereoscopic indicators and spatial depth with sufficient realism for the majority of telepresence applications.

Additionally, since the displacement is calculated dynamically inside Unreal Engine, this workflow is still highly flexible and efficient. Changes to the depth map—e.g., new captures or filtered augmentations—can be applied directly to the geometry with no delay, and therefore is appropriate for both static visualizations and real-time remote interaction applications.

F. Integration into Unreal Engine

Thanks to its native support of VR, Unreal Engine was chosen as the foundation of the proposed telepresence system, leveraging a rich pipeline for material customization and real-time capabilities. Everything from visual feedback, spatial processing of data, and user interaction is brought under one roof within this engine, which acts as both renderer and control center. Its `C++` integration at the low level and its blueprint system result in it being particularly well-suited to be interfaced with external middlewares like ROS 2 and custom data streams processing on-the-fly[8].

To integrate Unreal Engine with the perception pipeline, we used a native node capable of subscribing to depth, RGB, and infrared topics published by the ROS 2 Azure Kinect pipeline. Upon reception, raw data is translated into Unreal textures and passed on to dedicated rendering components for display and further processing.

1) Custom Material: To spatially align and visually integrate the acquired perception data, a custom Unreal Engine material that accepts multiple streams of input data was developed (see Fig. 5). The material accepts RGB, Depth, and Infrared textures as input parameters and utilizes a parallax node to generate visually appropriate 3D cues from depth.

Furthermore, the view mode parameter supports runtime switching between:

- Pure RGB stream (for standard camera visualization).
- Pure IR stream (useful in low-light environments).
- Blended RGB+IR view, where infrared data is softly composited on RGB for hybrid perception.

This multi-stream adaptability enables the operator to customize visualization according to varying real-world conditions; for instance, low light conditions, reflective surfaces, or occlusions.

2) Dynamic Mesh: To go beyond surface-level effects, real geometry deformation driven by the incoming depth data is also accounted in the proposed method. This is achieved by initializing a **dynamic mesh component** from a subdivided spherical geometry that gets reloaded each frame, thus ensuring a constant rest state, and avoiding the application of displacement multiple times on the same mesh.

All instances of mesh are PN-Tessellated[9] to gain increased vertex density, allowing for more detailed displacement mapping. Dynamic meshes in Unreal, unlike standard static meshes, are dynamic at runtime. This allows us to displace each vertex individually based on depth information, enabling volumetric relief effects which correspond very accurately to the real-world geometry that the sensor detects.

[8]https://roscon.ros.org/jp/2021/presentations/10.pdf

[9]PN Tessellation, or Point-Normal tessellation, is a method used to add details on a 3D models by subdividing their surface and creating smoother, curved geometry by creating smaller faces.

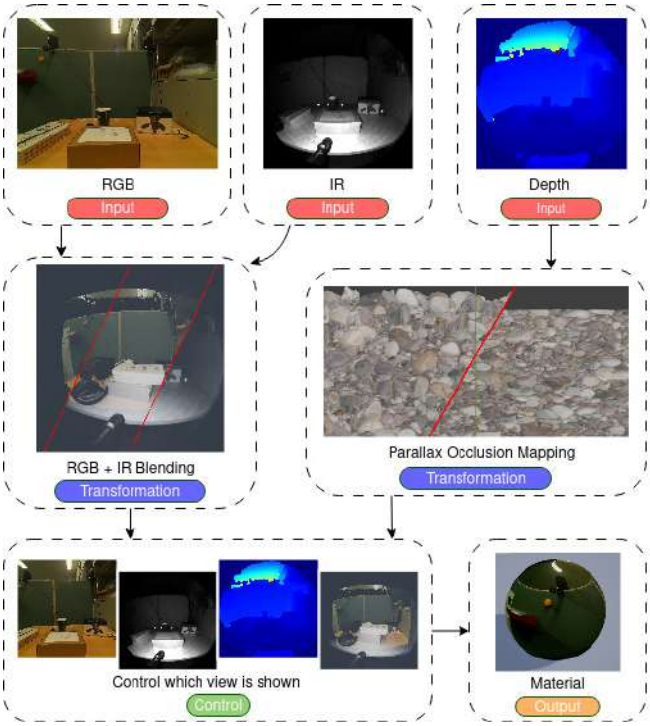


Fig. 5: Custom Unreal Engine material used for modalities fusion.

As detailed in Sect. II-B, the depth map is preprocessed in real time to fill in gaps, producing a smooth, dense, and artifact-free displacement texture. This filtered depth map serves as input for the displacement node in Unreal, which computes each vertex's offset along its normal vector according to the corresponding grayscale value at the vertex UV position.

By combining this approach with ongoing reprojection and temporal filtering, we obtain a temporally coherent and visually stable 3D surface. The outcome is a real-time mesh featuring real parallax, occlusion, and volume from the user's perspective, greatly adding to immersion and spatial understanding in the VR environment.

3) User Interface: To facilitate immersion and offer runtime control over visualization modes along with material parameters, we add a diegetic user interface directly within the virtual world. Rather than employing external menus or keyboard shortcuts, the proposed interface is physically built into the operator avatar, further reducing the mental load and possible disruption from immersive VR environment.

Especially, a wrist-mounted Heads-Up Display (HUD) is placed on the left arm of the user. The display only appears when the user raises the wrist to look at it—mimicking reading a watch—or when pressing a button on the left controller. As soon as the user looks away or releases the button, the interface will fade out naturally, preserving the immersion of the scene.

The HUD allows users the following interactives:

- Switch between view modes (RGB, IR, blended RGB+IR).
- Turn depth displacement on/off.
- Teleport to the center of the mesh.

(a) Without camera (b) With camera

Fig. 6: Contraption to attach the Azure Kinect on top of the Valve Index VR Headset.

Moreover, two other interaction modes provided by the system are available:

- A **free mode**, in which the interface panel can be dynamically teleported to the user's head center—useful when particular positioning or comfort is required.
- A **constrained mode**, in which the interface is fixed to the user's head and follows its movement, simulating a stationary "helmet-mounted" display.

This flexibility allows the operator to customize interaction ergonomics based on use case. Combined with the other features described above, this interface design facilitates intuitive and seamless interaction with the VR environment.

III. Experiments

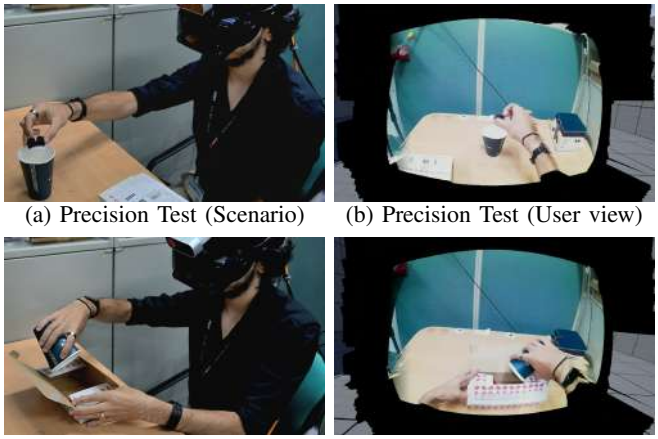
To evaluate the proposed RGB-D fusion system's performance, we conducted a user study involving 20 participants aged from 21 to 42 years with varying levels of VR and video game experience. All participants were volunteers from our laboratory and received no compensation for their participation. We measured task accuracy and efficiency, as well as participants' subjective perception of the system while performing standard tasks with and without the proposed immersive visual feedback system.

To better approximate the user's native viewpoint, the RGB-D camera was firmly mounted on the front of a Valve Index VR headset using a custom 3D-printed device (see Fig. 6). While this head-mounted configuration allows a direct egocentric viewpoint to be evaluated, it does not fully represent a deployable teleoperation setup where the viewpoint of the robot and the operator may diverge. In future work, the transformation between the operator's virtual camera and the robot-mounted sensor will be explicitly modeled to account for viewpoint discrepancies during remote manipulation.

All participants were asked to complete two simple tasks under both visual representation (deformed sphere, see Sect. II-F) and planar perspective with direct camera stream displayed on it, which will be used as a baseline [13]). Condition order was counterbalanced among participants in order to equate for learning effects, thus reducing possible bias sources. The two tasks were:

- **Task 1: Precision Test (see Fig. 7a and 7b)**

The subject was asked to place a small object into a cup without touching its borders and without looking from above. Each time the border was touched, it was considered as a failure. This task was intended to evaluate fine motor control and spatial information understanding.



(a) Precision Test (Scenario) (b) Precision Test (User view)

(c) Speed Test (Scenario) (d) Speed Test (User view)

Fig. 7: Experiment realized.

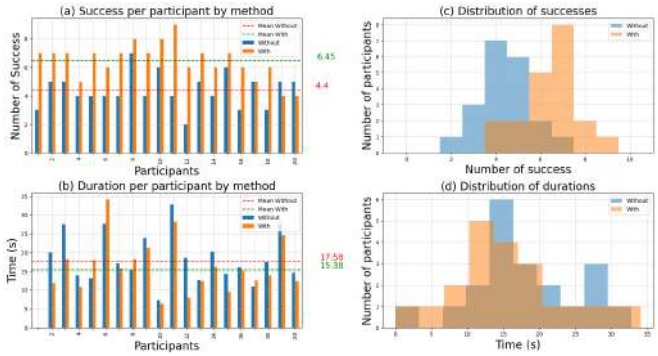


Fig. 8: Performance of participants during the tasks with the depth deformation (orange) and without using a planar view (blue). (a) Success rate per participant; (b) Histogram of success rates; (c) Task duration per participant; (d) Histogram of durations.

- **Task 2: Speed Test (see Fig. 7c and 7d)**

The subject then performed a sequential task that was a simple "assembly" operation: empty the cup into a box, close the box, place the box on a container, and flip the now-empty cup down on top of the box. This task was used to assess execution time and overall confidence towards achieving teleoperation tasks.

Every task was performed one time with the RGB-D system and one time without it using a planar view, with the same physical setup. To broaden the baseline, a planar RGB view was compared against a version of the scene rendered on a non-displaced spherical mesh, isolating the contribution of depth deformation to the perceived immersion. User interactions were monitored and timed. A short subjective questionnaire (available in the Supplementary Material video) on perceived control, understanding of depth, and immersion was also filled out by the participants after performing the tasks in both conditions.

The graph presented in Fig. 8 shows a summary of the 20 participants' performance, listing the success on the precision task on 10 repetitions and the time (in seconds) on the speed task. Fig. 9 on the other hand, presents some opinions and impressions about the proposed system.

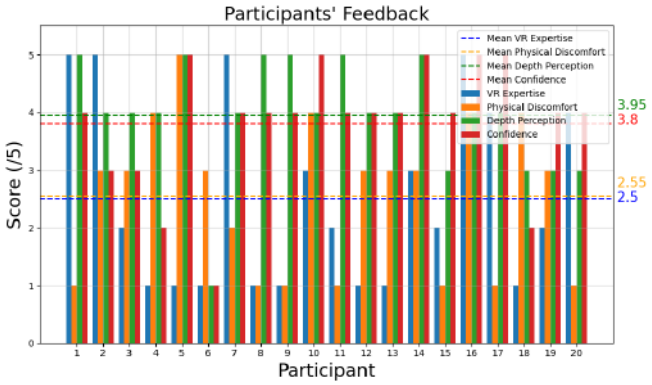


Fig. 9: Participants feedback on the experiment over multiple criterion along with their mean values.

These results allow us to compare user performance and feelings in both conditions, giving precious insight on the impact of immersive RGB-D feedback on teleoperation accuracy and efficiency.

IV. Discussion

The experimental results show that the performance improvement is notable when the participants utilized the RGB-D fusion system compared to the baseline. This section presents quantitative and qualitative findings from precision and coordination tasks, as well as subjective responses collected after the experiment. We used the Kruskal-Wallis test [17] to evaluate the statistical significance of the differences between the planar baseline and the immersive RGB-D condition.

1) Precision Performance: For the precision task, participants performed 10 trials with and without the RGB-D system. Results show a 47% increase in task accuracy when using the latter. On average, participants succeeded in $4.4/10$ trials without the system, but $6.45/10$ with it ($p < 0.0001$, where a p-value close to 0 indicates strong statistical significance, whereas this significance decreases as p increases), highlighting the benefit of the wide FoV RGB-D fusion.

This enhancement supports the hypothesis that enriched, depth-enhanced visual feedback improves spatial awareness and comprehension. By providing real-time 3D context, the system enables users to make fine manipulations with more confidence, even under indirect viewing conditions.

2) Task Completion Time: The average amount of time taken by the participants to complete the task without the proposed approach was around 17.58 s, while the average time using our system was around 15.38 s. Whereas this represents a 13% performance improvement, the statistics are less significant ($p \approx 0.25$). Some subjects were faster when using the immersive view, while others were slower, likely due to other variables, biases of visual information or cautious motion allowed by enhanced depth perception during the experiment.

These findings imply that while depth feedback does not accelerate task execution, it does not incur significant latency overhead either—the system maintained real-time responsiveness ($\approx$ 30–45 FPS) during all tests.

3) Subjective Feedback: Participant feedback post-experiment revealed overwhelmingly favorable reactions toward the system (details can be found in the Supplementary Material video).

- **Comfort and Visual Immersion**

On a 5-point scale, the average reported visual comfort was $\sim 3.55/5$, spatial disorientation was low at $1.65/5$, while immersion in the 3D environment was at $\sim 3.95/5$, implying that participants perceived the rendering as natural and easy to follow with few reporting visual discomfort or confusion. However, physical discomfort may be high, at around $\sim 2.55/5$, mainly due to the head-mounted contraption used to simulate a remote teleoperation setup without an actual robot. This discomfort likely stems from the weight and rigidity of the camera mount, which would not be present in a real teleoperation system where the camera is robot-mounted.

- **Interface Intuitiveness**

Almost all the subjects expressed that the interface was intuitive and easy to understand. The responsiveness helped users to interact with the proposed method, allowing multiple view mode to explore the system.

- **Confidence and Control**

More than half of the subjects indicated that they felt more self-control and confident when interacting with the system, compared to using the planar perspective. The feedback frequently consisted of enhanced object distance perception and scene structure, which assisted planning and execution of the tasks.

These subjective findings complement the quantitative improvement in performance, highlighting that the RGB-D fusion system not only improves accuracy, but also raises the operator's situational awareness and comfort—key factors for teleoperation in real-world applications.

V. Conclusion and Future Work

This paper presented a RGB-D fusion system for enhancing visual feedback in teleoperation by taking advantage of real-time depth sensing and immersive rendering capabilities through wide FoV cameras. The system combined the Azure Kinect camera and Unreal Engine 5.1 with ROS 2 communication, enabling the projection of sensor information on a dynamically displaced spherical mesh. This layout increases the operator's FoV and allows for more consistent spatial perception by combining color and depth modalities into one, real-time VR environment.

The proposed pipeline enhances depth quality and color-depth alignment, enabling consistent 3D visualization. Real-time integration in Unreal Engine allows adaptive rendering across sensing modes without loss of interactivity.

Experiments with users conducting teleoperation tasks demonstrated that immersive feedback improved accuracy and user confidence compared to conventional planar displays, without introducing excessive latency or complexity.

This framework offers a foundation for further advancements of immersive teleoperation interfaces. Future works involve testing our proposed system in actual robotic environments, ergonomics refinement, and determining scalability to multi-sensor configurations or dynamic environments.

Acknowledgments

This paper is based on results obtained from a project of Programs for Bridging the gap between R&D and the IDeal society (society 5.0) and Generating Economic and social value (BRIDGE)/Practical Global Research in the AI x Robotics Services, implemented by the Cabinet Office, Government of Japan.

We also thank all the subjects who volunteered their effort and time for the experiments.

References

[1] E. Triantafyllidis, C. Mcgreavy *et al.*, "Study of multimodal interfaces and the improvements on teleoperation," *IEEE Access*, vol. 8, pp. 78 213–78 227, 2020.

[2] J. Du, C. Mouser *et al.*, "Design and Evaluation of a Teleoperated Robotic 3-D Mapping System using an RGB-D Sensor," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 5, pp. 718–724, 2016.

[3] Y. Pan, C. Chen *et al.*, "Augmented reality-based robot teleoperation system using RGB-D imaging and attitude teaching device," *Robotics and Computer-Integrated Manufacturing*, vol. 71, p. 102167, 2021.

[4] M. Benallegue, G. Lorthioir *et al.*, "Humanoid robot RHP Friends: Seamless combination of autonomous and teleoperated tasks in a nursing context," *IEEE Robotics & Automation Magazine*, vol. 32, no. 1, pp. 79–90, 2025.

[5] Q. Vuong, Y. Qin *et al.*, "Single RGB-D Camera Teleoperation for General Robotic Manipulation," *arXiv:2106.14396*, 2021.

[6] D. Rolley-Parnell, D. Kanoulas *et al.*, "Bi-Manual Articulated Robot Teleoperation using an External RGB-D Range Sensor," *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 298–304, 2018.

[7] N. Zaman, A. Tavakkoli *et al.*, "Tele-robotics via An Efficient Immersive Virtual Reality Architecture," *ACM/IEEE Int. Conf. on Human-Robot Interaction Workshop on Virtual, Augmented, Mixed Reality for Human-Robot Interaction*, 2020.

[8] A. Naceri, D. Mazzanti *et al.*, "The Vicarios Virtual Reality Interface for Remote Robotic Teleoperation," *J. of Intelligent & Robotic Systems*, vol. 101, no. 80, 2021.

[9] P. Stotko, S. Krumpen *et al.*, "A VR System for Immersive Teleoperation and Live Exploration with a Mobile Robot," *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 3630–3637, 2019.

[10] F. Zhao, W. Deng *et al.*, "A Robotic Teleoperation System with Integrated Augmented Reality and Digital Twin Technologies for Disassembling End-of-Life Batteries," *Batteries*, vol. 10, no. 11, p. 382, 2024.

[11] M. Schwarz and S. Behnke, "Low-Latency Immersive 6D Televisualization with Spherical Rendering," *Proc. of IEEE-RAS Int. Conf. on Humanoid Robots*, pp. 320–325, 2021.

[12] Y. Chen, L. Sun *et al.*, "Enhanced visual feedback with decoupled viewpoint control in immersive humanoid robot teleoperation using SLAM," in *Proc. of IEEE-RAS Int. Conf. on Humanoid Robots*, 2022, pp. 306–313.

[13] L. Penco, K. Momose *et al.*, "Mixed Reality Teleoperation Assistance for Direct Control of Humanoids," *IEEE Robotics and Automation Letters*, vol. 9, no. 2, pp. 1937–1944, 2024.

[14] J. Ku, A. Harakeh *et al.*, "In Defense of Classical Image Processing: Fast Depth Completion on the CPU," in *Proc. of Conf. on Computer and Robot Vision*, 2018, pp. 16–22.

[15] P. Soille, *Morphological Image Analysis: Principles and Applications*. Springer, 2004.

[16] M. Bertalmío, A. L. Bertozzi *et al.*, "Navier-Stokes, fluid dynamics, and image and video inpainting," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 1, 2001, pp. I–I.

[17] P. E. McKight and J. Najab, *Kruskal-Wallis Test*. John Wiley & Sons, Ltd, 2010. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470479216.corpsy0491