

Low-Latency Immersive 6D Televisualization with Spherical Rendering

Max Schwarz* and Sven Behnke

Abstract—We present a method for real-time stereo scene capture and remote VR visualization that allows a human operator to freely move their head and thus intuitively control their perspective during teleoperation. The stereo camera is mounted on a 6D robotic arm, which follows the operator’s head pose. Existing VR teleoperation systems either induce high latencies on head movements, leading to motion sickness, or use scene reconstruction methods to allow re-rendering of the scene from different perspectives, which cannot handle dynamic scenes effectively. Instead, we present a decoupled approach which renders captured camera images as spheres, assuming constant distance. This allows very fast re-rendering on head pose changes while keeping the resulting temporary distortions during head translations small. We present qualitative examples, quantitative results in the form of lab experiments and a small user study, showing that our method outperforms other visualization methods.

I. INTRODUCTION

There are many ways to control robotic systems, from remote control over teleoperation to full autonomy. Teleoperation is highly relevant and valuable to the robotic research community, first of all because it allows to address tasks that are impossible to solve using state-of-the-art autonomous control methods—the human intellect and creativity in solving problems and reacting to unforeseen events is unmatched. Furthermore, teleoperation can generate training examples for improving autonomy in the future. There are also immediate benefits for the economy and daily life: Teleoperation can allow humans to work remotely in potentially dangerous environments, which is highly desirable in situations like the current COVID-19 pandemic. This interest is embodied in new robotics competitions like the ANA Avatar XPRIZE Challenge¹ and a multitude of other past and ongoing competitions [1], [2].

One important, if not the most important feature of a teleoperation system is its capability to display the remote environment to the human operator in a convincing, immersive way. Maintaining full situational awareness is necessary to both induce security and confidence for the operator as well as to solve tasks efficiently and robustly.

We present a real-time scene capture and visualization system for teleoperated robots, which gives the human operator wearing a head-mounted display full 6D control of the first-person camera perspective. In summary, our contributions include:

- 1) A robotic head assembly equipped with a 180° 4K stereo camera pair, capable of 6D movement with low latency;

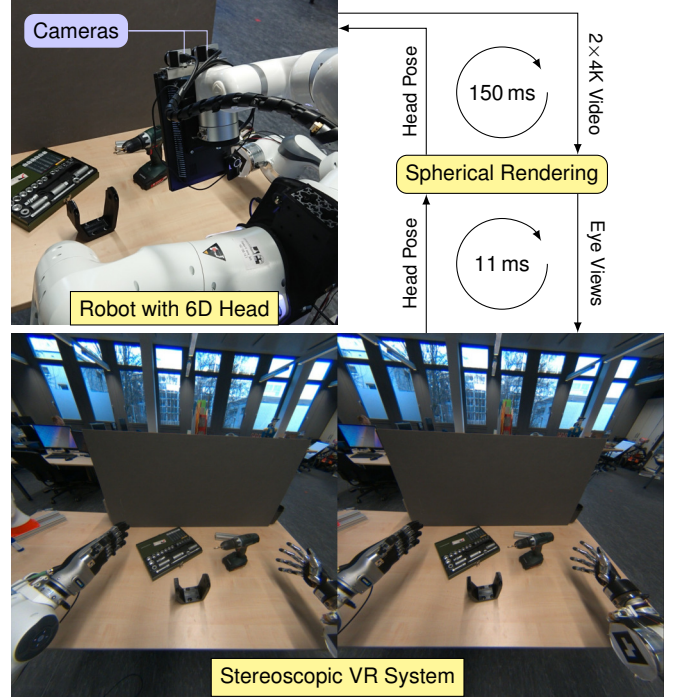


Fig. 1. Our low-latency televisualization approach. The 2×4K video stream from a stereo pair of wide-angle cameras is rendered using a sphere model to the operator’s VR headset. On a fast timescale, operator movements are handled without much latency by re-rendering from the new pose. The operator’s pose is also streamed to the robotic system, which moves the cameras accordingly in 6D, although on a slower timescale.

- 2) a combination of calibration steps for camera intrinsics and extrinsics, hand-eye, and VR calibration; and
- 3) an efficient rendering method that decouples the VR view from the latency-afflicted robot head pose, providing a smooth low-latency VR experience at essentially unlimited frame rate.

II. RELATED WORK

Viewing a stereo camera video feed is one of the established methods for remote immersion, providing live imagery with intuitive depth perception.

Martins *et al.* [3] mount a stereo camera on a field robot developed for rescue applications. While the camera is fixed to the robot, its track-driven body can bend to adjust the camera pitch angle. The operator wears a head-mounted display, whose tracking pose is used to control the pitch angle and the yaw angle of the entire robot. Roll motions are simulated by rotating the camera images in 2D. The authors report that the high latency on yaw changes was confusing for operators. We also note that this method

*Both authors are with the Autonomous Intelligent Systems group of University of Bonn, Germany; schwarz@ais.uni-bonn.de

¹<https://www.xprize.org/prizes/avatar>

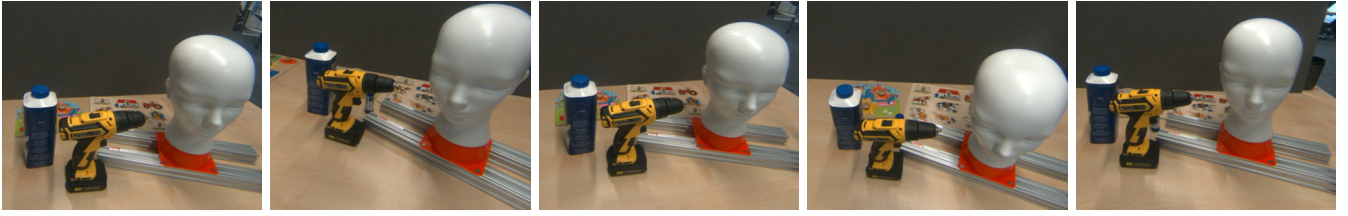


Fig. 2. Range of motion supported by our setup. We show the left eye view of the HMD, cropped to the central part for clarity. Note that the human operator can easily change perspective and observe parts of the scene that were previously hidden.

will not accurately represent head orientations due to the decomposition into Euler angles, which are then applied at different locations. In contrast, our method supports full 6D head movement (see Fig. 2) and reacts to pose changes with almost zero latency.

Zhu *et al.* [4] study whether head or gaze tracking is more suitable for camera orientation control. However, they do not use a HMD but a 2D monitor for displaying camera images. Both head and gaze control modes operate in a relative fashion, moving the camera in the indicated direction until the head/eye returns to its neutral position. The authors conclude that the gaze-based control method feels more natural. In contrast, our system uses the absolute head pose to directly control the camera pose, leading to very natural and intuitive control of the camera.

This technique can also be seen in the work of Agarwal *et al.* [5], who control a robot’s head pose directly from a human head pose measured with a Microsoft Kinect 2. Their robot head is only capable of 2D movement, pitch and yaw, so the human pose is mapped with a neural network to the lower-dimensional space. The robot head is not equipped with cameras, however.

Very similarly, in one of the earliest works we could find, Heuring and Murray [6] control a stereo camera pair mounted on a pan/tilt unit with sufficiently low latency to follow human head motions precisely.

Conversely, Lipton *et al.* [7] combine stereo capture with VR to enable teleoperation, but mount their camera in a fixed location on the robot, providing only limited field of view.

When the robot or its head cannot be moved fast enough, latencies are introduced into a real-time teleoperation system. One particularly effective approach for removing latencies is to display not the live camera feed, but a reconstructed 3D scene, in which the operator can move freely without much latency. We explored this technique in our own work, where we rendered reconstructed 3D laser maps of the environment to the user [8] and displayed textured meshes generated from a live RGB-D feed [9]. In both cases, we used a VR headset which allowed the user free movement inside the reconstructed data with low latency. Later on, we improved on these techniques in Stotko *et al.* [10] by displaying a high-fidelity 3D mesh of the environment generated from an automatically moving RGB-D camera mounted on the end-effector of a mobile manipulator.

Visualizing reconstructed scenes has the disadvantage that although the system reacts to observer pose changes with

almost no latency (requiring just a rendering step), changes in the scene itself typically only show up after longer delays—if the scene representation supports dynamic content at all. This makes these approaches completely unsuited for many telepresence situations, e.g. when interacting with a person or performing manipulation. Furthermore, reconstruction methods typically rely on depth, either measured by depth cameras or inferred (SfM). Many materials and surfaces encountered in everyday scenes result in unreliable depth and may violate assumptions made by the reconstruction method (e.g. transparent objects, reflections), resulting in wrong transformations or missing scene parts.

RGB-D-based visualization solutions as the one by Whitney *et al.* [11], or Sun *et al.* [12], who display live RGB-D data as point clouds for teleoperation of a Baxter robot, can cope with dynamic scenes but still suffer from measurement problems and the inherent sparsity—the operator can often look through objects.

Displaying a live stereo camera feed as done in our method neatly sidesteps these issues, since the 3D reconstruction happens inside the operator’s visual cortex. We address the resulting higher observer pose latency using a smart rendering method discussed in the next section.

III. METHOD

A. Robotic Platform & VR Equipment

Our robot’s head is mounted on a UFACTORY xArm 6, providing full 6D control of the head (see Figs. 1 and 4). The robotic arm is capable of moving a 5 kg payload, which is more than enough to position a pair of cameras and a small color display for telepresence (not used in this work). Furthermore, the arm is very slim, which results in a large workspace while being unobtrusive. Finally, it is capable of fairly high speeds (180°/s per joint, roughly 1 m/s at the end-effector), thus being able to match dynamic human head movements.

We chose two Basler a2A3840-45ucBAS cameras for the stereo pair, which offer 4K video streaming at 45 Hz. The cameras are paired with C-Mount wide-angle lenses, which provide more than 180° field of view in horizontal direction. We also experimented with Logitech BRIO webcams with wide-angle converters, which offer auto-focus but can only provide 30 Hz at 4K, resulting in visible stutters with moving objects. The Basler cameras are configured with a fixed exposure time (8 ms) to reduce motion blur to a minimum.



Fig. 3. Raw image from one of the wide-angle cameras. The horizontal field of view is higher than 180°.

The robot head is mounted on a torso equipped with two Franka Emika Panda arms and anthropomorphic hands.

The operator wears an HTC Vive Pro Eye head-mounted display, which offers 1440×1600 pixels per eye with an update rate of 90 Hz and 110° diagonal field of view. While other HMDs with higher resolution and/or FoV exist, this headset offers eye tracking which will be beneficial for building telepresence solutions in future work. As can be seen, the camera FoV is much larger than the HMD FoV, which ensures that the visible FoV stays inside the camera FoV even if the robot head should be lagging behind the operator’s movement.

The VR operator station is driven by a standard PC (Intel i9-9900K 3.6 GHz, NVidia RTX 2080), connected via Gigabit Ethernet to the robot computer (same specifications).

B. Calibration

Before the system can be used, multiple transforms and parameters need to be calibrated. We devised a principled approach starting at camera intrinsic calibration over hand-eye calibration on the robot side, to VR calibration on the operator side.

Since the cameras have very high FoV with significant fish-eye distortion (see Fig. 3), we use the Double-Sphere camera model [13] to describe the intrinsics. Its parameter sets μ and γ for the left and right cameras, respectively, can be calibrated easily using the kalibr software package [14], just requiring an Aprilgrid calibration target. The kalibr software also estimates the extrinsic transformation between the two cameras T_R^L .

As shown in Fig. 4, there are further extrinsic transformations to be calibrated, especially if one wants to accurately perform manipulation with a robotic arm. We follow a classic hand-eye calibration approach with an ArUco marker on the robot’s wrist, with the goal of estimating T_{cam} , but also the transformation T_{mount} between the mounting of the head arm and the robot arm, and T_{mark} , the pose of the marker on the wrist.

We can compute the 2D pixel positions p^L and p^R in the

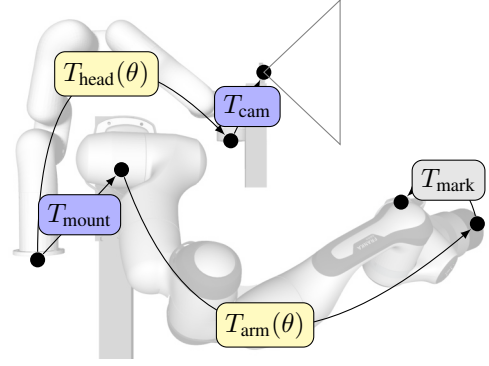


Fig. 4. Frames involved in the calibration procedure. Transformations colored yellow are computed using forward kinematics from measured joint angles while blue transformations are optimized. The auxiliary transform T_{marker} is optimized, but not used in the calibrated system. Note: The camera FoV is shown for illustration purposes and is not to scale.

left and right camera image frame as follows:

$$p^L = f_{\mu}(T_{\text{cam}}^{-1} \cdot T_{\text{head}}^{-1}(\theta) \cdot T_{\text{mount}} \cdot T_{\text{arm}}(\theta) \cdot T_{\text{mark}}), \quad (1)$$

$$p^R = f_{\gamma}(T_R^L \cdot T_{\text{cam}}^{-1} \cdot T_{\text{head}}^{-1}(\theta) \cdot T_{\text{mount}} \cdot T_{\text{arm}}(\theta) \cdot T_{\text{mark}}), \quad (2)$$

where $T_{\text{arm}}(\theta)$ and $T_{\text{head}}(\theta)$ are the poses of the arm and head flanges relative to their bases for the current joint configuration θ , and f is the double-sphere projection function with parameters μ and γ found above.

We collect samples with known 2D pixel coordinates \hat{p}^L and \hat{p}^R extracted using the ArUco marker detector of the OpenCV library. During sample collection, the head continuously moves in a predefined sinusoidal pattern while the robot’s arm is moved manually using teach mode. Sample collection takes about 5 min for an engineer.

Finally, we minimize the squared error function

$$E = \sum_{i=1}^N \|p^L - \hat{p}^L\|_2^2 + \|p^R - \hat{p}^R\|_2^2 \quad (3)$$

over all N samples using the Ceres solver, yielding optimal transforms T_{cam} and T_{mount} .

C. Head Control

The remaining question is how to transfer the head pose, measured in the VR space, to the robot side. To initialize the mapping, we keep the robot head in predefined nominal pose $T_{\text{nom}}^{\text{robot}}$, looking straight ahead. We ask the human operator to do the same and record the resulting head pose in VR space $T_{\text{nom}}^{\text{VR}}$, taking care to remove any remaining undesirable pitch and roll component.

During operation, the head pose $T_{\text{head}}^{\text{robot}}$ is computed from the tracking pose $T_{\text{head}}^{\text{VR}}$ in straightforward fashion:

$$T_{\text{head}}^{\text{robot}} = T_{\text{nom}}^{\text{robot}} \cdot (T_{\text{nom}}^{\text{VR}})^{-1} \cdot T_{\text{head}}^{\text{VR}}, \quad (4)$$

directly mapping the operator’s head movement to the robot’s head.

The head pose is smoothed by a recursive low-pass filter with a cut-off frequency of 100 Hz to remove very high frequency components. It is also monitored for larger jumps,

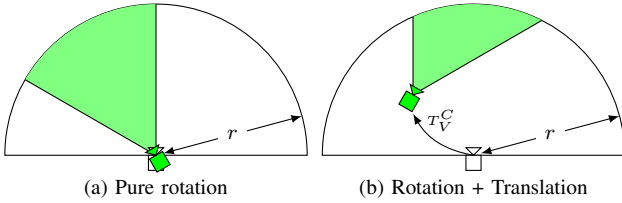


Fig. 5. Spherical rendering example in 2D. We show only one camera C of the stereo pair, the other is processed analogously. The robot camera is shown in white with its very wide FoV. The corresponding VR camera V , which renders the view shown in the HMD, is shown in green. The camera image is projected onto the sphere with radius r , and then back into the VR camera. Pure rotations (a) result in no distortion, while translations (b) will distort the image if the objects are not exactly at the assumed depth.

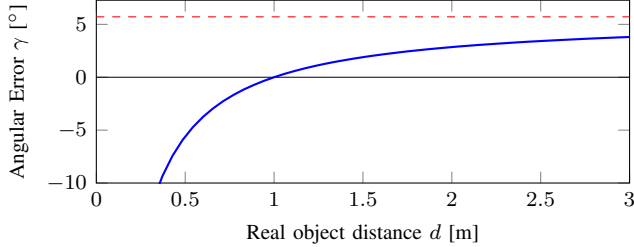


Fig. 6. Angular error introduced by translation of the operator head by $\Delta x = 10$ cm depending on the distance to the object. Since the spherical reprojection assumes a distance of 1 m, we see no error at that distance. The red dashed line shows the error upper bound for large distances d .

in which case we switch to a slower initialization mode that moves to the target pose with limited velocity.

D. Spherical Rendering

If operator and robot head poses matched exactly, the camera images could be rendered immediately to the HMD by projecting each pixel in HMD image space to the camera images and looking up the corresponding color value. However, when the poses are not identical, as is usually the case, we have a 6D transform T_C^V between the virtual VR camera V (representing the operator's eye) and actual camera position C . Traversing this transform requires 3D positions, but we do not have depth information.

Hence, we must make assumptions which will not impact the result quality much, since the transform T_C^V will be quite small. The simplest assumption is one of constant depth for the entire scene. Since we are dealing with wide-angle cameras, it is natural to assume constant *distance*, leading to spherical geometry around the camera. With this assumption, we can raycast from the VR camera V , find the intersection with the sphere, and find the corresponding pixel in the camera image (see Fig. 5).

In practice, the rendering part is implemented in OpenGL using the Magnum 3D Engine². We use the standard rasterization pipeline by rendering a sphere of radius $r = 1$ m at the camera location. A fragment shader then computes the pixel location (in C) of each visible point of the sphere and performs a texture lookup with bilinear interpolation to retrieve the color value.

²<https://magnum.graphics>

Using this design, head movements are immediately reacted to. Pure rotations of the head will have no perceptible latency, since the wide-angle cameras will already have captured the area the operator will look at. Translational movements work as if everything is at a constant distance, yielding distortions wherever this assumption does not hold (see Fig. 5). The amount of distortion can be quantified by the angular projection error γ and depends on the distance d to the object and the head translation Δx (here perpendicular to the view ray):

$$\gamma = \arctan\left(\frac{d}{\Delta x}\right) - \arctan\left(\frac{r}{\Delta x}\right). \quad (5)$$

Note that γ quickly saturates for large d against the value $\frac{\pi}{2} - \arctan\left(\frac{r}{\Delta x}\right)$ (see Fig. 6). For $d < r$, the distortion can become significant. In conclusion, it is important to choose r close to a typical minimum object distance. Of course, as soon as the robot head catches up and moves into the correct pose, any distortion is corrected.

E. Latency Improvements

We found that we needed to take special care at nearly all parts of the pipeline to ensure low-latency operation with the required time precision.

Starting at the cameras, we wrote a custom driver which uses the built-in timestamping feature of the cameras to obtain high-precision timing information in order to correlate images with head-arm transforms. Furthermore, it is necessary to compress the image stream for transmission over Ethernet, since the raw stream bandwidth exceeds 350 MB/s. For easy interoperability with standard ROS packages, we chose an MJPEG compression, which is done efficiently on the GPU [15]. The resulting stream has a bandwidth of 30 MB/s, which is acceptable in our setup. If even less bandwidth is required, hardware-accelerated H.264 encoding could be a solution.

Furthermore, we developed a custom driver for the xArm, which provides joint position measurements at a high rate. This required modifications to the arm firmware. Our modified version can control and report joint positions at 100 Hz. Its measurements are interpolated and synchronized with the timestamped camera images using the ROS `tf` stack.

Finally, the VR does not use the standard `image_transport` ROS package, which cannot decompress the two MJPEG streams with 4K resolution at 45 Hz. Instead, we use the GPUJPEG decoder [15].

All in all, these examples highlight that many off-the-shelf products are not designed for the required low-latency operation and manual adaptation requires considerable effort.

IV. EXPERIMENTS

A. Quantitative Experiments

In a first experiment, we measured the total image latency from acquisition start inside the camera to display in the HMD. For this, we used the timestamping feature of the camera, with the camera time synchronized to the robot computer time. Both computers are synchronized using the

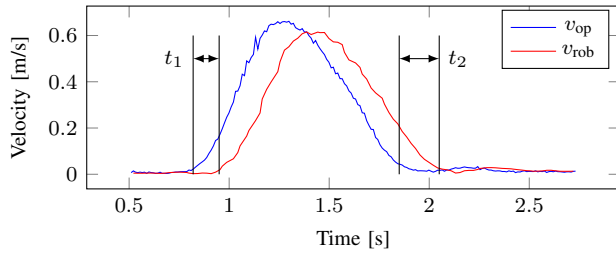


Fig. 7. Head movement latency experiment. Shown are the Cartesian velocities of the operator's head v_{op} and the robot's head v_{rob} during a fast (0.5 m/s) head movement. Vertical lines have been added to ease analyzing latencies (start latency $t_1 \approx 130$ ms, finish latency $t_2 \approx 100$ ms).

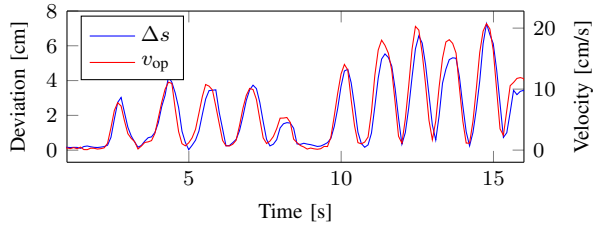


Fig. 8. Head tracking performance. We show the distance Δs of the VR eye camera V from the robot camera C for the left eye stream. The operator was performing highly dynamic head movements. As expected, the position deviation is highly correlated with operator head velocity v_{op} .

chrony NTP system, so that timestamps are comparable. Finally, the VR renderer compares the system time against the camera timestamp at rendering each frame. On average, the images are shown with 40 ms latency. A considerable fraction of this time is consumed by the camera exposure time (8 ms) and USB transfer time (around 10 ms).

Since the VR renderer achieves the maximum HMD update rate of 90 Hz without difficulty, rotational head movements have the same latency as in local VR applications, which is not noticeable—according to publicly available information, the Lighthouse VR tracking system operates with 250 Hz update rate³.

To analyze the total latency of head control, we plotted Cartesian velocities of the raw target head pose (as recorded by the VR tracking system) and the measured head pose during a fast head motion (see Fig. 7). The latency varies along the trajectory, but the head comes to a stop roughly 200 ms after the operator's head stopped moving. We note that this is a maximum bound of the latency; slower head motions result in lower latencies.

Furthermore, we analyzed the absolute deviation between the HMD eye pose and the camera pose, i.e. the error that is introduced during the spherical rendering step (see Fig. 8). We note that even during fast motions, only minor distortions were noticeable to the operator.

B. User Study

We conducted a small user study in order to measure the immersiveness and efficacy of our pipeline. Due to the ongoing COVID-19 pandemic, we were limited to immediate colleagues as subjects, which severely constrained the scope of our study to seven participants.

³https://en.wikipedia.org/wiki/HTC_Vive

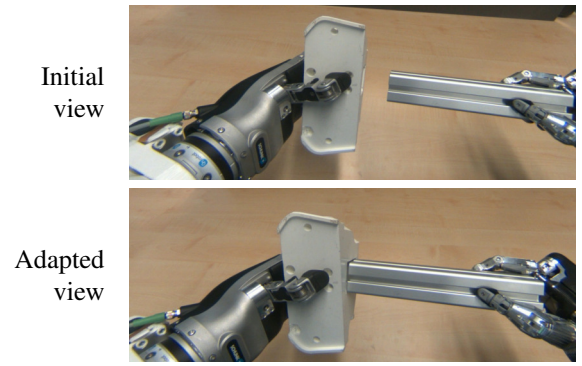


Fig. 9. Operator view (cropped) during the user study task. The peg has square cross section of 4×4 cm. The hole in the white plastic object is 0.3 mm larger. In the bottom situation, the user repositioned their head to better view the interfacing parts.

TABLE I
USER STUDY SUCCESS RATES AND TIMINGS

| Visual mode | Success | Completion time [s] | |
|----------------------|---------|---------------------|--------|
| | | Mean | StdDev |
| a) Full 6D | 7/7 | 71.0 | 50.6 |
| b) 3D orientation | 7/7 | 111.7 | 87.5 |
| c) Fixed perspective | 6/7 | 158.3 | 39.6 |

Participants were asked to teleoperate the robot, performing a rather difficult peg-in-hole task involving grasping two objects, and inserting one into the other with tight tolerances (see Fig. 9). The task places high demands on the visual system since a) even small misalignments would prevent success and b) the peg object was quite hard to hold and slippage could only be detected visually. For controlling the two robot arms and hands, the participants used a custom force-feedback exoskeleton system which is not the focus of this work. The participants performed the task three times, with different visual modes:

- a) *Full 6D*: Entire pipeline with full 6D movement.
- b) *3D orientation*: The operator only controls the orientation of the head (similar to [3], [6]). The spherical rendering system is used, but the camera spheres are translated to the HMD eye position. Notably, this variant can still react to orientation changes with low latency by re-rendering.

c) *Fixed perspective*: The robot head orientation is fixed to a 45° downward pitch, focusing on the table in front of the robot. The camera images are rendered directly to the HMD, i.e. the HMD pose has no effect. This is similar to devices sold as First-Person-View (FPV) glasses, which are commonly used in drone racing.

The order of the three trials was randomized to mitigate effects from learning during the trials. Participants could attempt the task as often as they liked (with an assisting human resetting the objects on the table if moved out of reach). When a maximum time of 5 min was reached without success, the trial was marked as failed.

Although the task itself was not the primary goal of the user study, effects of the visual mode on success rates and average completion time can be observed (Table I). The full 6D mode never led to a failure and yielded the lowest average

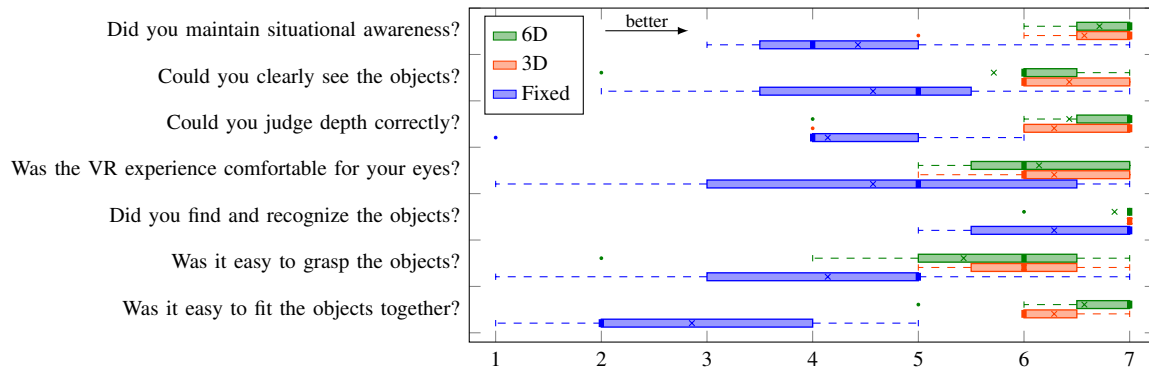


Fig. 10. Statistical results of our user questionnaire. We show the median, lower and upper quartile (includes interquartile range), lower and upper fence, outliers (marked with •) as well as the average value (marked with ×), for each aspect as recorded in our questionnaire.

completion time, although the standard deviation was quite high—mostly due to external effects such as the robot arms requiring a reset after too much force was applied, but the skill level also varied highly from participant to participant.

After the three trials, we asked the participants questions for each of the visualization modes, with answer possibilities from the 1-7 Likert scale. From the results reported in Fig. 10, it is immediately visible that the fixed perspective variant leads to significantly lower user acceptance. The users reported that this was mainly due to inability to pan the camera and instead having to move the workpieces to a better observable pose. Furthermore, the head translation seems to have no impact on grasping the objects, but impacts insertion—where it is beneficial to be able to look slightly from the side to see the involved edges (compare Fig. 9).

V. DISCUSSION & CONCLUSION

We have demonstrated a method for 3D remote visualization that does not rely on geometry reconstruction and is thus suited for highly dynamic scenes, but is still able to deliver low-latency response to 6D head movements. The user study shows that allowing full 6D head movement leads to faster task completion and better user acceptance when compared with fixed-mount and pan-tilt systems, which are still common in teleoperation.

There are also some points of the system that can be improved. The work space of the arm used to move the head is—in its present mounting position—limited and cannot match the full motion range of sitting humans moving their torso and neck. For this, it would be interesting to investigate a more biologically inspired joint configuration.

While our method has been evaluated only in a seated configuration, it should be applicable with a completely mobile robot and operator as well. This can be investigated in future research.

Our work also does not focus on the transmission of data to remote sites. It is clear that for transmission over the Internet, bandwidth requirements would have to be lowered, e.g. through the use of more efficient video compression codecs. The system should, however, be able to cope with typical latencies found on Internet connections (up to 200 ms for connections halfway around the world).

REFERENCES

- [1] E. Krotkov, D. Hackett, L. Jackel, M. Perschbacher, J. Pippine, J. Strauss, G. Pratt, and C. Orlowski, “The DARPA Robotics Challenge finals: Results and perspectives,” *Journal of Field Robotics*, vol. 34, no. 2, pp. 229–240, 2017.
- [2] H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou, and S. Shimada, “RoboCup Rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research,” in *Int. Conf. on Systems, Man, and Cybernetics (SMC)*, vol. 6, 1999.
- [3] H. Martins, I. Oakley, and R. Ventura, “Design and evaluation of a head-mounted display for immersive 3D teleoperation of field robots,” *Robotica*, vol. 33, no. 10, p. 2166, 2015.
- [4] D. Zhu, T. Gedeon, and K. Taylor, “Head or gaze? Controlling remote camera for hands-busy tasks in teleoperation: A comparison,” in *Conf. of the Computer-Human Interaction SIG of Australia*, 2010.
- [5] P. Agarwal, S. Al Moubayed, A. Alspach, J. Kim, E. J. Carter, J. F. Lehman, and K. Yamane, “Imitating human movement with teleoperated robotic head,” in *International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2016, pp. 630–637.
- [6] J. J. Heuring and D. W. Murray, “Visual head tracking and slaving for visual telepresence,” in *International Conference on Robotics and Automation (ICRA)*, vol. 4, 1996, pp. 2908–2914.
- [7] J. I. Lipton, A. J. Fay, and D. Rus, “Baxter’s homunculus: Virtual reality spaces for teleoperation in manufacturing,” *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 179–186, 2017.
- [8] T. Rodehutsors, M. Schwarz, and S. Behnke, “Intuitive bimanual telemanipulation under communication restrictions by immersive 3D visualization and motion tracking,” in *International Conference on Humanoid Robots (Humanoids)*, 2015, pp. 276–283.
- [9] T. Klamt, M. Schwarz, C. Lenz, L. Baccelliere, D. Buongiorno, T. Cichon, A. DiGuardo, D. Droschel, M. Gabardi, M. Kamedula, N. Kashiri, A. Laurenzi, D. Leonardi, L. Muratore, D. Pavlichenko, A. S. Periyasamy, D. Rodriguez, M. Solazzi, A. Frisoli, M. Gustmann, J. Roßmann, U. Süß, N. G. Tsagarakis, and S. Behnke, “Remote mobile manipulation with the Centauro robot: Full-body telepresence and autonomous operator assistance,” *Journal of Field Robotics*, vol. 37, no. 5, pp. 889–919, 2020.
- [10] P. Stotko, S. Krumpfen, 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 *International Conference on Intelligent Robots and Systems (IROS)*, 2019.
- [11] D. Whitney, E. Rosen, E. Phillips, G. Konidaris, and S. Tellex, “Comparing robot grasping teleoperation across desktop and virtual reality with ROS reality,” in *Robotics Research*, 2020.
- [12] D. Sun, A. Kiselev, Q. Liao, T. Stoyanov, and A. Loutfi, “A new mixed-reality-based teleoperation system for telepresence and maneuverability enhancement,” *IEEE Transactions on Human-Machine Systems*, vol. 50, no. 1, pp. 55–67, 2020.
- [13] V. Usenko, N. Demmel, and D. Cremers, “The double sphere camera model,” in *Int. Conf. on 3D Vision (3DV)*, 2018, pp. 552–560.
- [14] J. Rehder, J. Nikolic, T. Schneider, T. Hinzmann, and R. Siegwart, “Extending kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes,” in *Int. Conf. on Rob. and Aut. (ICRA)*, 2016.
- [15] P. Holub, J. Matela, M. Pulec, and M. Šrom, “UltraGrid: Low-latency high-quality video transmissions on commodity hardware,” in *20th ACM International Conference on Multimedia*, 2012, pp. 1457–1460.