Proceedings of the IEEE-RAS 16th International Conference on Humanoid Robots
pp. 242-248, November 15-17, 2016

Panoramic View Reconstruction for Stereoscopic Teleoperation of a Humanoid Robot

Konstantinos Theofilis[1], Jason Orlosky[2], Yukie Nagai[1] and Kiyoshi Kiyokawa[2]

***Abstract*— In recent years, robots have become more commonplace as a method for humans to remotely conduct observations or interactions, with commercial applications ranging from customer service to assembly line work. Virtual and augmented reality have shown much promise as methods to visualize and assist with teleoperation of these robots. However, a number of problems still exist with such bidirectional teleoperator relationships, including the intersensory conflict induced by delay of images received on the teleoperator side of a remote session. This can result in nausea, simulation sickness, and unnatural interaction, especially when stereoscopic cameras are present.**

As a step towards addressing this problem, we introduce the use of stereo panoramic reconstruction in conjunction with an augmented reality interface to improve both the interaction and the sense of embodiment of a teleoperator working through a robot's perspective. Unlike current methods that send latent streams of the robot's eye-cameras to the user, we view reconstruction that also incorporates direct control of the robot's head. This algorithm uses queued time and angle stamped images to reconstruct the robot's stereoscopic view for the teleoperator, which allows for low-latency viewing and field of view expansion. We then conduct experiments comparing our method against a direct camera feed from the robot's eyes. In addition to measuring latency at every point of the data pipeline, we show that this panoramic representation reduces perceived visual delay and elicits positive user feedback.[3]

I. Introduction

Achieving real time remote robotic control has been a challenge for researchers for many reasons. For instance, teleoperators have traditionally viewed controlled robots through static monitors or displays, but these displays neither give the operator a stereoscopic view nor provide a spatially accurate representation of the environment. To provide a better view of the environment, Augmented (*AR*) and Virtual (*VR*) Realities have recently been implemented with some success [1] [2] [3].

However, many of the same problems with traditional remote operation still exist for AR/VR. Various differences in human anatomy and robot physics and mechanics, latency, and incorrect assumptions about the machine's capabilities such as head rotation limitations can result in uncomfortable sensations for the teleoperator, nausea, frustration, and even damage to the robot. Overcoming these challenges is a significant step towards building applications for remote manipulation and increasing sense of presence for tele-interaction. One particularly troublesome problem that has yet to be overcome is that of delay. This can be due to both bidirectional network latency and the mechanical latency of a remote robot, and results in unwanted side effects. If a user were to view this environment through the robot's eyes, latency can cause simulation sickness, fatigue, and discomfort [4]. This latency problem is compounded in the case of remote control since a remote user must first send a movement command to the robot via network, the robot must then execute the movement, the resulting stereoscopic camera images must then be captured and sent back to the user, and those images must then be rendered on the immersive display.

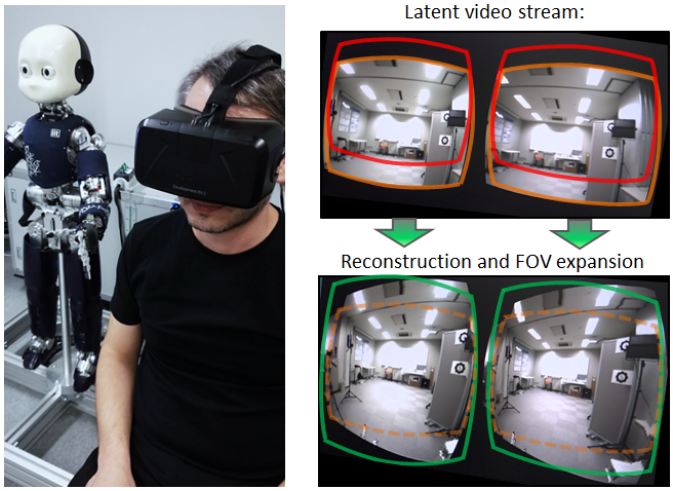

Fig. 1. Teleoperator system with bidirectional control (left), image of a typical direct see-through mode (latent video images are streamed directly to the user's eyes) where the user's viewpoint has changed, but the latent images are still rendered (top right), and our panoramic reconstruction that provides for low latency, wider field of view (FOV) rendering of a reconstructed scene image (upper right).

In current implementations, this delay can be well over 500 milliseconds (ms), which is unacceptable for real time viewing. Even in an optimized network where all machines are on the same local network and undistortion and rendering algorithms are processed by optimized graphics pipelines, delay can still amount to several hundred ms, causing intersensory conflict.

To mitigate this problem, we have come up with a strategy that utilizes a stereo panoramic reconstruction of the robot's eye-camera video streams to allow for low-latency viewing of a remote environment. Though updates to subsections of the reconstruction still experience delay, a remote viewer can navigate the entire reconstruction with less than 40 ms (the inherent rendering latency of the VR display plus the time

[1]Graduate School of Engineering, Osaka University, Osaka, Japan {kostas,yukie}@ams.eng.osaka-u.ac.jp
[2]Cybermedia Center, Osaka University, Osaka, Japan {orlosky@lab.,kiyo@}ime.cmc.osaka-u.ac.jp
[3]The first two authors contributed equally to this work.

required for reconstruction) of end-to-end *perceived* latency. Frame positions of remote rendering using a direct video (referred to as direct stream in this paper) are shown in the bottom right of Fig. 1, which can be in a different location than the actual current position of the robot eye cameras.

In comparison with direct stream, our stereo panoramic reconstruction (referred to as *panoramic view*), shown on the right of *Fig*. 1, decouples the robot's and user's viewspaces, and allows for navigation of sections of the environment unavailable with direct stream. This reconstruction (separate for each eye) is viewed as a stereo texture in a 3D game world, and regions of the texture for each eye corresponding to latent frames passed from the robot are updated in real time. This provides a way to significantly reduce intersensory conflict from latency since the views are decoupled. Thus, the panoramic view is both latency and field of view (FoV) independent, meaning it can be used regardless of distance and for operations with binocular and monocular robot eye/cameras where relative camera position is known.

While the technique is applicable to many robotic forms, it is particularly useful for a stationary interaction-oriented humanoid such as the iCub, which we used for our implementation. In contrast with 360 degree video reconstruction and viewing, we incorporate and test bidirectional head control of the robot via the HMD. Much like stereoscopic 3D videos, saved reconstructions can also provide training data for future teleoperators and robot/machine learning algorithms, and can be navigated independently of time delay.

II. Related Work

Since the advent of modern robotics, operators have struggled to control visuomotor systems in an effective and intuitive way. Most initial work on robotic control focuses on static screen-based interaction, which affords the operator several ways to see through the robot's view and relay commands.

One such framework was outlined by Brooks in 1986 [5], which provided a general layered format with which to robustly control a remote, mobile robot. Researchers then started to focus on more specific problems in teleoperation such as delay and control mechanisms. For example, in 1991, Bejczy et al [6] proposed using a predictive simulator to compensate for delays when visualizing the future actions of a robot. Other strategies such as collaborative control have been employed to improve driving actions in multi-robot scenarios such as that of Fong et al. [7]. For more fine-grained control, Marin et al. have proposed multimodal interaction using voice, text, and direct control to improve remote operations [8]. A number of mechanisms to reduce or simplify network latency have also been utilized, such as back-channelling [9]. Though delay was still a significant factor, remote teleneurosurgery has been performed in more recent years [10].

After the general concept of remote robot control was introduced, several researchers also proposed the use of AR and VR to improve control mechanisms by improving visualization of the physical state of the remote robot. One such system was developed by Milgram et al. in 1993 [11]. They proposed using graphic overlays as a method to improve depth judgements during remote operation. Though their system was still largely susceptible to network delay, this marked a significant step forward in the use of augmentative environments for improving robot visualization. Several years later, a cockpit type system was developed that allowed for full control of a remote humanoid robot with stereoscopic image relay. The system, developed by Tachi et al., employed a full body cockpit that even allowed for fine grained control of hand and finger movements [12]. This type of system in particular would benefit from the low latency provided by our panoramic view since good balance and minimal disorientation are key requirements for the teleoperator.

Such interaction methodologies have also been proposed for human-human tele-assistance, such as the system proposed by Adachi et al. in 2005 [13]. Another system, which is probably the most similar to our own, was that of Fiala et al. [14]. They generate a panoramic image of a single 360 degree catadioptric camera to view a remote environment through a small mobile robot. Though this helps deal with latency, it was only implemented in for a monoscopic camera with a relatively small FoV HMD, and 360 degree cameras are not always present on humanoid robots. Building on the research by Milgram et al., Hashimoto et al. [15] further developed the concept of using AR overlays for improved manipulation of a remote robot in 2011. Though the interaction with the robot was conducted via a 2D static monitor and evaluation did not compare against a non-AR assisted task, this study further motivates the use of AR interfaces for robotic control.

With the introduction of affordable VR headsets, VR interfaces to robots have become more common, and have been implemented as prototype systems. One such prototype is the driving embodiment system proposed bye Almeida et al.[3]. They tested the interface for interaction using an RGB-D sensor, but also found that visual feedback delays and limited field of view required mental compensation on the part of the user. Okura et al. developed a free-point system that allows the user to navigate a 3D space reconstructed by depth-based images [16]. Still, this method requires the presence of a depth camera, is subject to artifacts and image quality limitations, and final robot movement does not follow head movemnt.

A very recent design by Martinez-Hernandez et al. incorporated the use of a fully immersive wide field of view display both for control and streaming of a remote humanoid robot viewpoint [1]. Even more recently, Fritsche et al. added the ability to manipulate objects using this same technique by incorporating a haptic *SensorGlove* into a VR interface [2] but their implementation used a direct streaming mode but ignores the image latency issue.

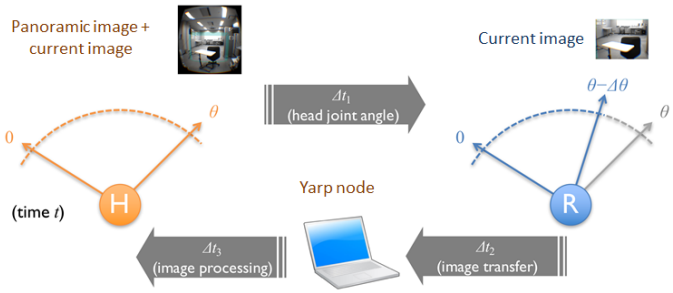

Fig. 2. The three main categories of delay: Δt_1 is the delay due to network but primarily mechanical translation of the movement of the user's head to the robot's head. Δt_2 is the delay that is caused mostly by the network transfer. This is by far the largest delay, especially as the image sizes increase. Δt_3 is caused by the final processing of the image before rendering on the HMD.

Though these designs represent good initial steps towards embodiment, intersensory conflict induced by latent images is still present, affecting user actions and perceptions. In order to reduce rendering latency, present the image at the teleoperator's remote environment with correct perspective, and still ensure the teleoperator has a contiguous experience, our panoramic reconstruction design can be used effectively.

III. Research Goals

Our primary goal is to mitigate the negative effects of this compounded latency on immersive remote teleoperation for a stereoscopic humanoid robot. Picking up where Martinez et al. left off [1], we develop a panoramic reconstruction that allows for low latency viewing of the iCub's current and previous viewpoints.

There are three main categories of delay that contribute to the end-to-end latency (Δt_1, Δt_2, and Δt_3 shown in Fig. 2). For most systems, end-to-end latency (from the time motion occurs to the time a new image is presented to the user's eyes) is at least 300 ms, which includes sending a movement command from the immersive display to the remote robot (Δt_1), executing the physical movement, acquiring both camera images from the new camera position (Δt_2), sending both camera images back to the display, and rendering. As a result, the uses at time t observes the image corresponding to his head position at $\Delta t_1 + \Delta t_2 + \Delta t_3$ before.This assumes 640×480 pixels images and that both server and client are on the same local network (and ideally the same subnet). The numerous bottlenecks along these networked, mechanical, and rendering pathways simply result in too much delay to be usable, so a different strategy to compensate for latency is necessary.

As is shown in Fig. 2, if the orientation of the human's head at time t is $H(t) = \theta$ and the orientation of the robot's head at the same time is $R(t) = \theta - \Delta\theta = H - (t - \Delta t_1)$. The image that the robot captures at the same time is $I(R(t)) = I(H(t - \Delta t_1 - \Delta t_2 - \Delta t_3)$.

We then conduct a comparative evaluation of the latency of the method with regards to the traditional direct stream technique using the system described below.

IV. System

A. iCub and Oculus Rift Framework

The hardware used in our system consists of a number of different parts, including the iCub robot, the Oculus Rift DK2 VR headset, the server to run the Oculus (*http://oculus.com*), and an intermediary laptop for the teleoperator.

The iCub humanoid robot [17] is used primary to study embodied cognition in artificial systems and for Human-Robot Interaction (HRI). The robot's software is based on the YARP [18] robotic middleware, that facilitates, among others, the communication between the nodes to the distributed system.

For our system, the movement range for the three axes of the head of the robot (pitch, roll, yaw) was, respectively, (-30,22), (-20,20) and (-45,45) degrees, and the joints were set in direct position control mode, i.e., there was not minimum-jerk trajectory generator. While the possible movement range of the joints of the robot can exceed the above values, safety guards were in place to ensure that the human teleoperator did not exceed the angular limitations of the neck joints.

The second part of our framework is the Oculus Rift DK2. In order to display the iCub camera video streams and render virtual objects, we have merged the iCub camera view with the lightweight java gaming library (*lwjgl - http://www.lwjgl.org*). This allows us to render video streams from the cameras as textures, move them in the 3D virtual environment as necessary, and also render virtual objects on top of the see-through components for AR/MR assistance. Stereo calibration of the two camera planes was accomplished initially by the iCub's calibration module, which is optimized via Cuda, and then by making fine, manual adjustments to the size and position of the rendered planes. The rendering process for these textures has been hand optimized so that the camera streams, gaming library (including the barrel undistortion and aberration correction for the Oculus Rift), and reconstruction framework can all run in real time at a steady 30 frames per second.

For the direct streaming implementations, eye camera images carry no information about their location at the time of capture, and the images are directly in front of the user's eyes, even if they do not correspond to the robot's real, current view. At the same time, the game world is rendered using only the current camera objects. For the panoramic reconstruction view, the image from each of the robot's eyes that is sent to the network is coupled with the orientation of the robot's head at the time of capture in Tait-Bryan angles (yaw,pitch,roll).

B. Software and Data Flow

The first step in setting up our software framework was initializing bidirectional communication between the iCub and Oculus rift through YARP. Across the whole system, data is sent and received through the framework as follows:

- **iCub side input:**
  - Oculus rift pose data: to control iCub head
- **iCub side output:**

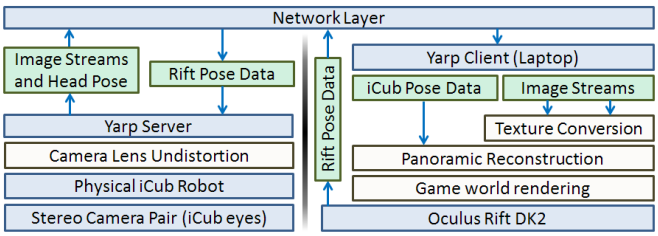

Fig. 3. System flow showing the YARP server side (left) and Oculus node side (right) processing, including hardware components (blue), software components (white), and streaming data (green).

  - Left and right eye RGB image
  - Current head pose (synchronous with images)
- **Teleoperator side input:**
  - Latent iCub left and right eye image
  - Head pose at time of image retrieval
- **Teleoperator side output:**
  - Oculus Rift user's head pose

A visual representation of this process can be seen in Fig. 3. Additionally, latencies are measured at each step of the input/output process, which allows us to find and evaluate bottlenecks in the network and mechanics of the robot.

C. Panoramic Reconstruction

The key to the panoramic reconstruction is decoupling the user's head movement from the current robot eye window. In traditional direct streaming methods, the image presented to the teleoperator will be the same, despite user head movement, until the latent images catch up to the teleoperator's current head position, as denoted on the left hand side of Fig. 4. This means that the same image is presented while the user's head is moving, which generates intersensory conflict. A simple way to understand how the panoramic view works is to imagine a large pre-generated panorama plane that sits in front of a user in a static position in immersive VR. By static position, we mean that the position of the user's head is not coupled to the panorama, letting the Oculus Rift compensate for head movement. In this case, the time between a user head movement and the updated image is only limited by rendering speed and reconstruction, which amounts to an average of 55ms. When designing this strategy, we drew from different reconstructive methods such as that of Gauglitz et al. [19].

The panoramic image is generated by our reconstruction functions in the same way, but portions of the panorama corresponding to incoming frames from the iCub are updated as they are received, as shown in the right hand images of Fig. 4. This way, the scene available to the user is much wider than the FoV of the robot's cameras, though it still suffers from minor perspective artifacts.

During startup, the panoramic image shows a center frame (corresponding to (0, 0, 0) pitch, roll and yaw, respectively, of the iCub head), as outlined by the orange box in the top left image of Fig. 3, prior to generating the reconstruction. Since the borders of the panorama are initialized as a black frame, we start a "warmup" phase that moves the iCub's head to the 4 corner points near the limits of its joints to complete an initial reconstruction. After this period, control is relinquished to the user, and he or she is then be able to control robot head movements using his or her own head motions. Creation of the panorama (rightmost images in Fig. 4) as new data is received can be described as follows.

The sub-region to be updated for the current frame is denoted by the orange dotted line in the bottom right image of Fig. 4. The relative position of this frame to the panorama matrix P_M is calculated with

$$P_{M(i-P_{Eye(i)},j-P_{Eye(j)})} = E_{M(i,j)}(t_L),$$

where P_{Eye} represents the eye pose of the latent frame at time t_L, and E_M represents the eye camera image matrix, as shown in Fig. 4. Any time a new frame is received, the panorama is updated on the next rendering frame of the oculus.

Target pose of the reconstruction texture P_{MO} in the Oculus relative to head movement (the solid blue outlines in Fig. 4) is represented by

$$P_{MO} = P_M \times P_{Pcalib} \times P_O^{-1},$$

where P_{Pcalib} is the initial position of the image plane after stereo calibration and P_O is the current pose of the Oculus. An objective measure of the sum of angular errors β_{off} for the number of latent frames N_L in spherical coordinates θ, ϕ (later shown in Fig. 5), can be measured with

$$\beta_{off} = \sum_{i=1}^{N_L} \Delta(\theta, \phi). \ (later\ in\ Fig.\ 5)$$

The portion of the panorama that is updated corresponds to angular values which we have embedded into each of the images sent from the iCub. These image and head position data pairs are received on the teleoperator client side, and the correct portion of the panoramas for both left and right eye data are buffered and written as shown in the system flow diagram in Fig. 3. The sub-image of the panorama that is updated corresponds to the position of the iCub at the time the images were pulled from the eye cameras.

Note that if the teleoperator continues to move his or her head at this point, he or she will be able to view a different part of the panoramic view despite not having received new image data. This is the main difference from other implementations that have to wait for a new frame. Similar methodology has been used to compensate for rendering lag in older head mounted display systems, such as the work by Kijima et al. [20], though this has not been applied to stereoscopic humanoid systems.

This strategy also accounts for misalignments between the teleoperator's head movement and the actual movement of the robot head. Misalignments can often occur in human-robot systems since acceleration of the mechanical parts of the robot may not always exhibit one-to-one correspondence to human head movements. Though recent improvements in the iCub's control software can compensate for most of these changes to some extent, other robots may not have the same

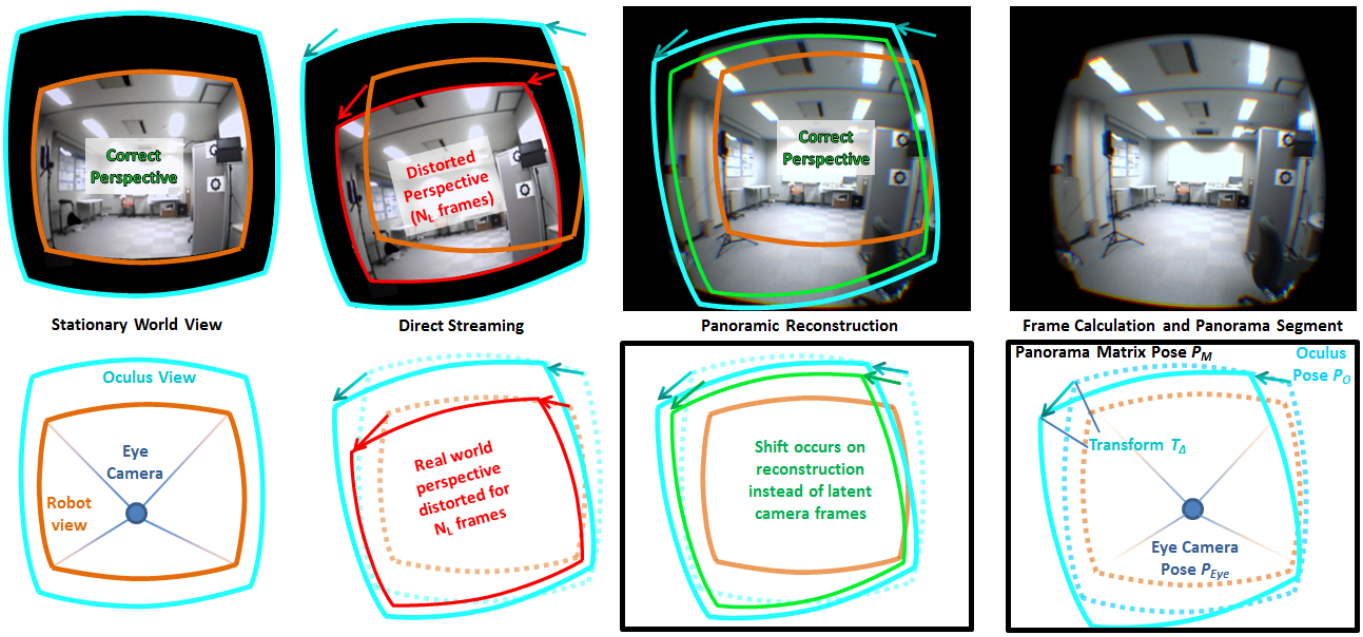

Fig. 4. From left to right: **Stationary World View** showing the correct perspective of the iCub robot camera at rest (top) and a respective pose diagram (bottom). **Direct Streaming** showing the incorrect rotation of a frame after a head rotation in a direct streaming implementation (top) and a respective pose diagram (bottom). **Panoramic reconstruction** showing the reconstruction window in black with current reconstructed data visible in the rift window (top) and respective layout diagram showing the correctly rotated perspective relative to the latent camera frame (orange frame, bottom). **Frame Calculation and Panorama Segment** showing a completed reconstruction through the Oculus viewport (top) and the pose label diagram showing names of each matrix and transform as described in the reconstruction calculations (bottom).

fine-grained control mechanisms in place. Unwanted motion due to jerks or overcompensation from prediction is also alleviated.

D. Improvements to mechanical performance

Due to the nature of the iCub's mechanics and stock software, head movements, especially in the vertical axis, had a delay of up to $500ms$. A customized version of the Direct Position Control module, from the iCub's software library, was used that eliminates that delay. In this version, the slew rate limiter was disabled, leading to much better reaction time of the head and reducing the Δt_1 delay. Both direct and panoramic modes benefited equally from that change.

V. System Evaluation

Our evaluation was conducted primarily to test end-to-end latency of the system for both direct streaming and panoramic modes of operation and net rotation error in each mode. This end-to-end latency refers to the *perceived* latency by the human, i.e. how much delay the image suffers in order to be projected onto the correct position at capture time. This delay causes the image to be projected at an incorrect angle, also defined by β_{off} above. To measure this, video frames of both the display and content were recorded (a common method for determining rendering latency) to calculate an initial approximation. Net rotation error was then calculated more accurately for a number of different head movements using frames stamped with position data and synchronized logging on both the YARP and Oculus side. We also ran an informal demo session to gather user feedback.

A. Frame update measurement

The first evaluation made use of an HD camera to simultaneously take video of an initial rotation of the Oculus Rift and the following frames captured from the robot's cameras and projected on a monitor, which eventually showed an updated image with the movement generated by the Oculus. By counting the number of frames between the initial movement of the Oculus and the updated image, we can compute the latency as the

$$number\ of\ frames \times \frac{1000}{framerate} ms$$

for an approximate latency calculation, with the framerate being $60 fps$. This is a common method used for verifying motion-to-photon latency in HMDs and VR systems. This process was repeated ten times, which generated the average and min/max latencies shown in Table 1. The delay for panoramic mode was taken to be that of the Oculus Rift plus the time to render for the panoramic view.

B. Net rotation error comparison

The net rotation error is a more suitable metric, due to the different nature of direct vs panoramic mode. The method used for the measurement of the rotation error, employed

TABLE I
Average end-to-end latency for Direct and Panoramic modes

Mode	Average	SD	Minimum	Maximum
Direct	702 ms	155.0	510 ms	1080 ms
Panoramic	55.4ms	27.4	5ms	165ms

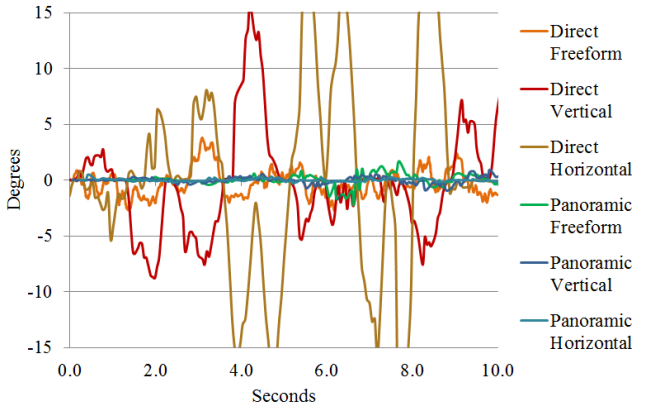

Fig. 5. This graph shows net rotation error as defined in section IV.C in degrees per second for direct (orange/red/brown) and panoramic (green/dark blue/light blue) modes. This delay is tested for three different categories of head movement, including vertical (up/down), horizontal (side-to-side), and freeform (including diagonal movement). This gives a visual representation of the reduction in perceived rotation error for panoramic vs. direct.

the logging capabilities of the YARP middleware, in order to have synchronized timestamps across all the computers in the cluster. The following information was logged: the iCub's head position (with timestamps), the human's head position (with timestamps), and the timestamps of each image for each eye when it was transmitted from the robot, received from the Oculus-connected computer, processed by our system and then displayed on the headset.

We recorded data in two conditions, direct vs panoramic, and three categories: horizontal movement, vertical movement and freeform movement, i.e., including diagonal movements. In each category, similar patterns of movement and velocity were used for both conditions. Fig. 5 shows the sum of the angular rotation errors as defined by the following equation

$$\beta_{off} = \sum_{i=1}^{N_L} \Delta(\theta, \phi)$$

for both modes. Here it is obvious that in direct mode, the sum of the error is much larger than in panoramic mode. In direct mode the error ranges from approximately -15 to +15 degrees while in the panoramic mode the error ranges from approximately -2.5 to +2.5 degrees. It is important to note that these errors (for both modes) increase the further the human's head moves from the center axis and the faster the head moves.

The net rotation error in Fig.5 correspond to the position of the incoming image. While in direct mode, the incoming images are the sole view of the human, while, in panoramic mode, the incoming images only update *part* of the panorama. The reconstructed panorama itself is always aligned with the center of the view of the human.

Smaller differences in vertical and horizontal head rotation are due to the mechanical characteristics and safety mechanisms built into the iCub robot. Realistically, other robots will experience somewhat different mechanical and network latencies, which should be addressed on an individual basis. Though it is known that local rendering reduces perceived latency for a user, this analysis provides us with information about exactly how much latency a user would experience over time for which movements.

C. Initial User Feedback

In addition to the latency measurements, we also used a demo session with 15 users to gather initial, informal, feedback on the two modes. Users tried both modes for a total of approximately 5 minutes, always starting with direct streaming. Some of the more relevant comments included:

- The panoramic view gives the perception of a circular space, whereas direct mode appears flat.
- Direct mode appears to be more consistent that panoramic mode, but delay is noticeable.
- When trying to gaze at certain targets, most participants also observed some overshooting of the robot's head in direct mode.
- Perception of depth was enhanced in panoramic mode.
- Four participants noted that they felt nauseous in direct mode, but this feeling was reduced after switching to panoramic mode. A majority of participants mentioned they felt more comfortable with panoramic mode compared to direct.

These comments give us good subjective evidence that the system can alleviate intersensory conflict, and a controlled user study is being planned.

VI. Discussion

Remote operation of a robot or telepresence device can become difficult and unpleasant if latency causes a discontinuity between head movement and virtual space. To add to this, the field of view of robotic eye cameras are typically narrow (60 degrees) when compared to the human eye, which can hamper interaction, reduce sense of embodiment, and decrease the feeling of presence for the teleoperator.

Our strategy provides a good practical way to reduce the effects of mechanical and network latency on human perception. One point to note is that the panoramic reconstruction technique will take time to update if the robot's body moves, though this wouldn't be the case for standing or stationary robots. To use the torso or a mobile robot where the camera position would undergo various translations, simultaneous localization and mapping techniques [21] or omnidirectional cameras could alleviate the problem, but would not necessarily provide a view of occluded points not yet visible to the reconstruction and would require additional cameras. Another optimal solution may be to have a 3D reconstruction of the workspace and only update new information in the panorama in addition to displaying 3D point information. Considering the number of remote interfaces that do not yet use stereoscopic reconstructions for teleoperation, this data should serve as motivation for instituting some form of reconstruction as a requirement for remote viewing in general.

One other obstacle that must be overcome is jitter in the reconstruction, which was observed in some of the image

frames, particularly at the edges of the screen. This is mostly due to the jitter and misalignments of the iCubs head at certain points in its trajectory, but this can potentially be overcome with filtering or alignment correction algorithms.

One more interesting discovery from initial evaluation is the feeling of presence when looking at a reflection of the iCub. Much like looking at one's reflection in the mirror gives the immediate sensation of embodiment, similar reflection techniques can potentially increase embodiment for teleoperators.

A. Future Work

Future work includes adding graphical overlays that will provide feedback on the robot's internal states and current position for a more accurate representation of the 3D environment. In essence, adding an Augmented Reality interface on top of the AR interface, for the teleoperator. Additionally, a user study, using both qualitative and quantitative metrics, is being planned to measure both comfort and task performance of the teleoperator in direct vs panoramic mode.

VII. Conclusion

In this paper, we present a method for reducing the perceived visual latency during remote robot teleoperation. Unlike direct visualization strategies that feed the camera streams directly to the user's VR headset, we propose a stereoscopic panoramic reconstruction that compensates for head movement by reducing perceived rendering delay. Evaluation and comparison of the system against the conventional direct streaming technique for different types of head movement shows that our reconstruction method significantly reduces end-to-end latency. A short session with 15 participants showed that this strategy has the potential to reduce simulation sickness and improve sense of presence and perception of depth. We hope that this research will promote more effective stereoscopic humanoid robot teleoperation.

Acknowledgments

This work is partially supported by the MEXT/JSPS Grants (Research Project Numbers: JP24119003, JP24000012, JP24300048 and A15J030230).

References

[1] U. Martinez-Hernandez, L. W. Boorman, and T. J. Prescott, "Telepresence: Immersion with the icub humanoid robot and the oculus rift," in *Conference on Biomimetic and Biohybrid Systems*. Springer, 2015, pp. 461–464.

[2] L. Fritsche, F. Unverzag, J. Peters, and R. Calandra, "First-person tele-operation of a humanoid robot," in *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*. IEEE, 2015, pp. 997–1002.

[3] L. Almeida, B. Patrao, P. Menezes, and J. Dias, "Be the robot: Human embodiment in tele-operation driving tasks," in *Robot and Human Interactive Communication, 2014 RO-MAN: The 23rd IEEE International Symposium on*. IEEE, 2014, pp. 477–482.

[4] M. H. Draper, E. S. Viirre, T. A. Furness, and V. J. Gawron, "Effects of image scale and system time delay on simulator sickness within head-coupled virtual environments," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 43, no. 1, pp. 129–146, 2001. [Online]. Available: http://hfs.sagepub.com/content/43/1/129.abstract

[5] R. Brooks, "A robust layered control system for a mobile robot," *Robotics and Automation, IEEE Journal of*, vol. 2, no. 1, pp. 14–23, Mar 1986.

[6] A. K. Bejczy, S. Venema, and W. S. Kim, "Role of computer graphics in space telerobotics: preview and predictive displays," in *Proc. SPIE*, vol. 1387, 1991, pp. 365–377.

[7] T. Fong, C. Thorpe, and C. Baur, "Multi-robot remote driving with collaborative control," *Industrial Electronics, IEEE Transactions on*, vol. 50, no. 4, pp. 699–704, Aug 2003.

[8] R. Marin, P. Sanz, P. Nebot, and R. Wirz, "A multimodal interface to control a robot arm via the web: a case study on remote programming," *Industrial Electronics, IEEE Transactions on*, vol. 52, no. 6, pp. 1506–1520, Dec 2005.

[9] M. F. Jung, J. J. Lee, N. DePalma, S. O. Adalgeirsson, P. J. Hinds, and C. Breazeal, "Engaging robots: Easing complex human-robot teamwork using backchanneling," in *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, ser. CSCW '13. New York, NY, USA: ACM, 2013, pp. 1555–1566. [Online]. Available: http://doi.acm.org/10.1145/2441776.2441954

[10] C. Meng, T. Wang, W. Chou, S. Luan, Y. Zhang, and Z. Tian, "Remote surgery case: robot-assisted teleneurosurgery," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 1, April 2004, pp. 819–823 Vol.1.

[11] P. Milgram, S. Zhai, D. Drascic, and J. Grodski, "Applications of augmented reality for human-robot communication," in *Intelligent Robots and Systems '93, IROS '93. Proceedings of the 1993 IEEE/RSJ International Conference on*, vol. 3, Jul 1993, pp. 1467–1472 vol.3.

[12] S. Tachi, K. Komoriya, K. Sawada, T. Nishiyama, T. Itoko, M. Kobayashi, and K. Inoue, "Telexistence cockpit for humanoid robot control," *Advanced Robotics*, vol. 17, no. 3, pp. 199–217, 2003.

[13] T. Adachi, T. Ogawa, K. Kiyokawa, and H. Takemura, "A telepresence system by using live video projection of wearable camera onto a 3d scene model," in *at International Conference on Computational Intelligence and Multimedia Applications, Las Vegas*. Citeseer, 2005.

[14] M. Fiala, "Pano-presence for teleoperation," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 3798–3802.

[15] S. Hashimoto, A. Ishida, and M. Inami, "Touchme: An augmented reality based remote robot manipulation," in *The 21st International Conference on Artificial Reality and Telexistence, Proceedings of*, Nov 2011.

[16] F. Okura, Y. Ueda, T. Sato, and N. Yokoya, "[paper] free-viewpoint mobile robot teleoperation interface using view-dependent geometry and texture," *ITE Transactions on Media Technology and Applications*, vol. 2, no. 1, pp. 82–93, 2014.

[17] G. Metta, G. Sandini, D. Vernon, L. Natale, and F. Nori, "The icub humanoid robot: an open platform for research in embodied cognition," in *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, ser. PerMIS '08. New York, NY, USA: ACM, 2008, pp. 50–56.

[18] G. Metta, P. Fitzpatrick, and L. Natale, "Yarp: yet another robot platform," *International Journal on Advanced Robotics Systems*, vol. 3, no. 1, pp. 43–48, 2006.

[19] S. Gauglitz, C. Sweeney, J. Ventura, M. Turk, and T. Hollerer, "Live tracking and mapping from both general and rotation-only camera motion," in *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, Nov 2012, pp. 13–22.

[20] R. Kijima and T. Ojika, "Reflex hmd to compensate lag and correction of derivative deformation," in *Proceedings of the IEEE Virtual Reality Conference 2002*, ser. VR '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 172–. [Online]. Available: http://dl.acm.org/citation.cfm?id=580130.835887

[21] M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *IEEE Transactions on robotics and automation*, vol. 17, no. 3, pp. 229–241, 2001.