dVRK-XR: Mixed Reality Extension for da Vinci Research Kit

Long Qian, Anton Deguet, Peter Kazanzides

Laboratory for Computational Sensing and Robotics, Johns Hopkins University
{long.qian, anton.deguet, pkaz}@jhu.edu (DOI10.31256/HSMR2019.47)

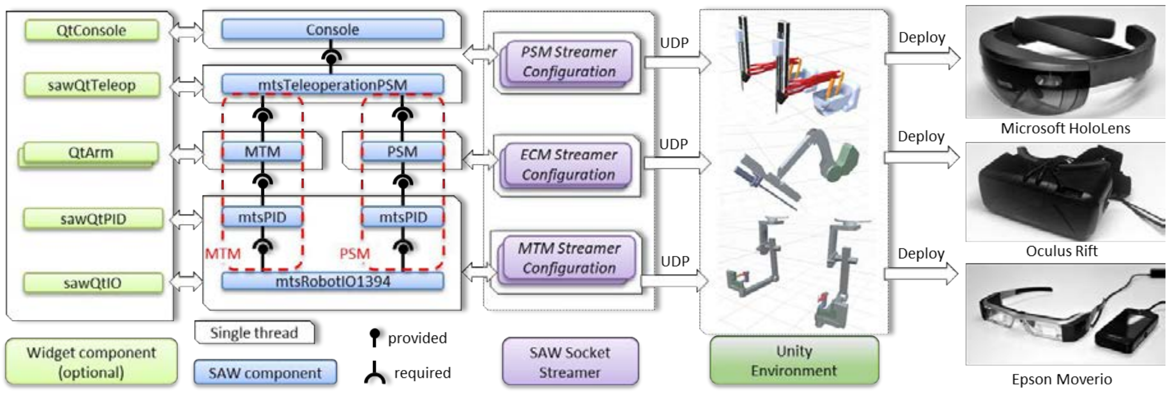

Fig.1 Overview of dVRK-XR open source package and integration with existing dVRK software stack

INTRODUCTION

The da Vinci Research Kit (dVRK) has become a widely adopted research platform for surgical robotics research since its initial public release in 2012. To date, it has been installed at 35 institutes worldwide. In the past decade, research interest in mixed reality (including augmented reality and virtual reality) has increased among both the robotics and medical communities [1], due to the improved usability of mixed reality headsets and the much reduced hardware cost. In this paper, we describe dVRK-XR (https://github.com/jhu-dvrk/dvrk-xr), an extension to the dVRK open source package that facilitates the integration of mixed reality in surgical robotics research.

MATERIALS AND METHODS

Software Architecture

The system architecture of dVRK-XR is shown in Fig. 1. The dVRK employs a component-based software architecture [3], in which different modules can be dynamically loaded with a JSON-based configuration. In fact, this feature enables us to extend the dVRK software stack in a clean and reliable way. We implement a new component: *sawSocketStreamer*, which can be connected to the existing dVRK program and send JSON-serialized messages at a fixed framerate over UDP. The message to send, the destination IP address, port number, and framerate can be dynamically configured using a JSON file. For example, with a Patient-Side Manipulator (PSM), we can inform *sawSocketStreamer* to retrieve the results of a function named "*GetStateJoint*" at 100Hz from the dVRK, then serialize the returned results and send to a specified destination.

Robot Representation in Unity

The application is built with Unity (https://unity.com), a popular development tool for mixed reality applications.

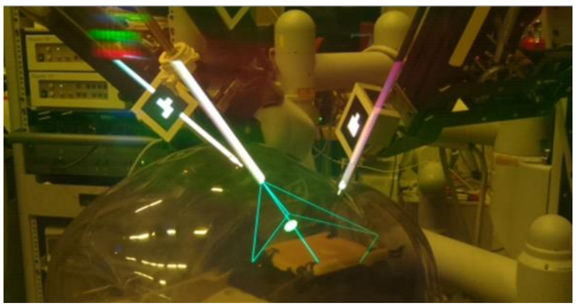

Fig. 2 An example application using dVRK-XR [2]

The Unity program can receive and deserialize the incoming messages. Therefore, the result of "*GetStateJoint*" of the PSM is available to the Unity environment. We use the DH parameters of the manipulator to build a tree of GameObjects in Unity that corresponds to its kinematics tree. The relative transformation of a parent node and child node can either be fixed, controlled by a specific field of an incoming message, or associated with another joint to reflect a passive mechanical linkage. The convention is similar to Unified Robot Description Format (URDF). We construct the PSM, ECM and MTM for Unity. With real-time messages containing joint state, the visualization of the manipulators can be synchronized with the real robot.

Target Mixed Reality Platforms

A major advantage of using Unity as a framework is its compatibility with multiple mixed reality platforms, e.g. VR platforms like Oculus (Desktop) and Google Cardboard (Android), AR platforms like Microsoft HoloLens (Windows UWP). Although Unity takes care of the graphics API of multiple platforms, the socket API and threading API does not cover Windows UWP, which is not a POSIX-compatible platform. Therefore, we explicitly implement an asynchronous UDP client using Windows UWP APIs. Currently, dVRK-XR supports Windows, Linux, Windows UWP and Android.

Multi-Threading and Rendering Performance

For a mixed reality application, it is critical to keep the rendering framerate as high as possible in order to provide a smooth user experience. We implement dVRK-XR with such concern in mind. On the server side, *sawSocketStreamer* is asynchronous with respect to the dVRK control and on the client side, the socket handling and rendering are done in different threads. When the rendering of a new frame is triggered, the latest message is pulled from the socket client, then deserialized and visualized.

RESULTS

End-to-End Latency

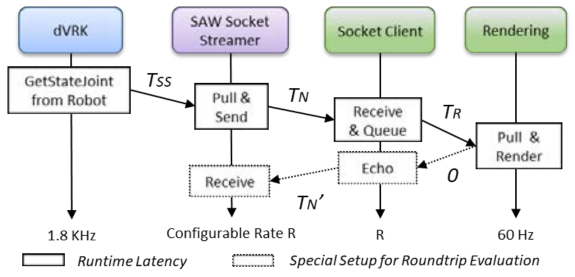

Fig. 3 The evaluation of end-to-end latency

We evaluate the end-to-end latency (T) between a real da Vinci robot and its virtual representation in dVRK-XR, as shown in Fig. 3. The runtime latency is:

$$T = T_{SS} + T_N + T_R \qquad (1)$$

where, i) T_{SS}: the latency between *sawSocketStreamer* and the joint state generated from the robot, ii) T_N: the networking time for sending the UDP packet, iii) T_R: the time between arrival and rendering. We assume that the latency introduced by the receiving mechanism of *sawSocketStreamer* and rendering is half of the average frame time, therefore: $T_{SS} = 0.000278\ ms$, $T_R = 0.5 / R$. We add a timestamp to the data packet, recording the time that a specific joint state is generated from the robot. The evaluation of T_N is not trivial because the time on the device and robot is not synchronized accurately. Therefore, we configure the socket client to echo the exact same message back, and we assume that $T_N' = T_N$. We measure the round-trip time T_{RD}, w.r.t. different frame rate of socket streamer. So,

$$T_{RD} = T_{SS} + T_N + T_R + T_N' \qquad (2)$$

Combining (1), (2), the overall latency is:

$$T = (T_{RD} + T_{SS} + T_R) / 2 \qquad (3)$$

We configured the framerate of *sawSocketStreamer* to be *50 Hz*, *100 Hz*, *150 Hz*, *200 Hz* and *250 Hz*, and used Windows Desktop (tethered device, e.g., Oculus Rift or HTC Vive) and HoloLens as mixed reality platforms.

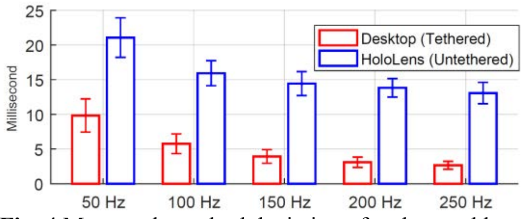

Fig. 4 Mean and standard deviation of end-to-end latency

The resulting end-to-end latency is shown in Fig. 4. The latency for a tethered device is less than *10 ms*, and for an untethered device, it is also smaller than a single frame if the *sawSocketStreamer* runs at *100 Hz* or higher. The visualization is real-time and low-latency.

DISCUSSION

dVRK-XR can enable many research opportunities that integrate surgical robotics and mixed reality. As an example, we developed an augmented reality application for the patient-side assistant in da Vinci surgery [2]. The assistant may need to insert a robotic instrument without knowing the position of the tip before it enters the field-of-view of the endoscope. Moreover, he or she may navigate a laparoscopic instrument with bad hand-eye coordination, due to the mis-orientation of the endoscopic view. With dVRK-XR and fiducial tracking, we can provide real-time visualization of the robotic instruments, endoscope and its field-of-view indicator, via the optics of HoloLens. The virtual rendering appears overlaid on top of the actual instrument (Fig. 2). A comparative study between ARssist and traditional visualization for the first assistant shows that the hand-eye coordination was much improved and the time to complete tool manipulation was significantly reduced [4]. Apart from benefiting the patient-side assistant, there are many other application scenarios with dVRK-XR:

- Teleoperation of virtual PSM or ECM
- Port/Trocar planning in robotic surgery
- Collaboration and interaction with remote experts
- Immersive visualization for training and education.

CONCLUSION

We developed dVRK-XR, an open source extension that enables mixed reality research with surgical robotics. The design considered compatibility, modularity, and user comfort. Our evaluation shows that dVRK-XR achieves real-time low-latency visualization on both tethered and untethered mixed reality platforms. We believe dVRK-XR will benefit the open source surgical robotics research community by easing the implementation of XR applications and extending the current research spectrum.

REFERENCES

[1] Barsom EZ, Graafland M, Schijven MP. Systematic review on the effectiveness of augmented reality applications in medical training. Surgical Endoscopy. 2016 Oct; 30 (10): 4174-4183.

[2] Qian L, Deguet A, Kazanzides P. ARssist: augmented reality on a head-mounted display for the first assistant in robotic surgery. Healthcare Technology Letters. 2018 Jan; 5(5): 194-200.

[3] Kazanzides P, Chen Z, Deguet A, Fischer GS, Taylor R H, DiMaio SP. An open-source research kit for the da Vinci® Surgical System. In IEEE Intl. Conf. on Robotics and Automation (ICRA) 2014 May (pp. 6434- 6439).

[4] Qian L, Deguet A, Wang Z, Liu YH, Kazanzides P. Augmented reality assisted instrument insertion and tool manipulation for the first assistant in robotic surgery. In IEEE Intl. Conf. on Robotics and Automation (ICRA) May 2019.