An Immersive Surgery Training System With Live Streaming Capability

Yang YANG [a], Xinqing GUO [a], Zhan YU [a], Karl V. STEINER [a,c], Kenneth E. BARNER [a], Thomas L. BAUER [b] and Jingyi YU [a]

[a] *University of Delaware*
[b] *Christiana Care Health Services*
[c] *University of Maryland, Baltimore County*

Abstract. Providing real-time, interactive immersive surgical training has been a key research area in telemedicine. Earlier approaches have mainly adopted videotaped training that can only show imagery from a fixed viewpoint. Recent advances on commodity 3D imaging have enabled a new paradigm for immersive surgical training by acquiring nearly complete 3D reconstructions of actual surgical procedures. However, unlike 2D videotaping that can easily stream data in real-time, by far 3D imaging based solutions require pre-capturing and processing the data; surgical trainings using the data have to be conducted offline after the acquisition. In this paper, we present a new real-time immersive 3D surgical training system. Our solution builds upon the recent multi-Kinect based surgical training system [1] that can acquire and display high fidelity 3D surgical procedures using only a small number of Microsoft Kinect sensors. We build on top of the system a client-server model for real-time streaming. On the server front, we efficiently fuse multiple Kinect data acquired from different viewpoints and compress and then stream the data to the client. On the client front, we build an interactive space-time navigator to allow remote users (e.g., trainees) to witness the surgical procedure in real-time as if they were present in the room.

Keywords. RGB-D Sensor, Microsoft Kinect, Real Time Immersive Surgery Training, 3D Reconstruction, Stereoscopic Display

Introduction

Providing effective immersive surgical training has become one of the key research topics in telemedicine. Traditionally, surgeons require longer education and training than other specialists. However, there are only a limited number of experts and both instructors and trainees are over-constrained by time. IN addition, new surgical procedures are emerging at a high rate with increasing complexity in protocols and devices. In traditional surgical training, videotaped instruction has long served as a workhorse. However, they are marginally effective: videotapes only provide 2D imagery from a fixed viewpoint and the trainee cannot freely navigate the procedure due to occlusions and lack of depth cues.

For the past decade, the success of 3D telepresence [2,3,4,5] has enabled state-of-the-art remote medical procedures. At the core of these solutions are techniques to acquire, reconstruct, and display the complete 3D geometry of room-sized surgical environments using 3D imaging technologies. Most existing approaches [3,6,7,8,9], e.g.,

from Fuchs's group at UNC, Bajcsy's group at UPenn, Kanade's group at CMU, and Gross's group at ETH, have adopted a sea of cameras solution, where a large number of cameras are positioned at different locations in the room. Computer vision techniques, in particular stereo matching and multi-view reconstruction, then are used to recover the 3D geometry from the imagery data. Despite their success on 3D acquisition, these solutions are still not readily applicable to real operating rooms: it is literally impractical to mount a sea of cameras within an Operation Room(OR) and robust 3D scene reconstruction is still very challenging.

Advances in commodity 3D imaging have provided a promising alternative to the "sea of cameras" solution. For example, the Microsoft Kinect sensor is able to acquire reasonable quality 3D data along with video streams. Each Kinect sensor essentially uses a RGB camera and an infrared projector and camera pair to capture color images and its corresponding depth maps. Based on the Kinect technology, Guo et al. [1] have recently presented a new immersive surgical training system. Instead of using a large number of cameras, their solution used a small number ($2 \sim 4$) of Microsoft Kinect sensors that are uniformly controlled by a single workstation. Their range and imagery data are fused via a companion computer vision algorithm for robustly recovering the 3D surgical scene. Despite its effectiveness on acquiring the data, their solution can still only replay the data offline. In contrast, an ideal remote surgical training should allow real-time navigation of the surgery.

In this paper, we present a new real-time immersive 3D surgical training system. Our solution builds upon Guo's multi-Kinect surgical training system [1] and provides real-time streaming capability. Specifically, we develop a client-server model. On the server front, we efficiently fuse multiple Kinect data acquired from different viewpoints and compress and then stream the data to the client side. On the client front, we build an interactive space-time navigator to allow remote users (e.g., trainees) to witness live surgical procedure as if they are personally on the scene. We address multiple technical challenges including color and range data compression, multi-tasking, data streaming, and online data visualization. Preliminary experiments show that our new real-time immersive training system is portable, effective, and reliable.

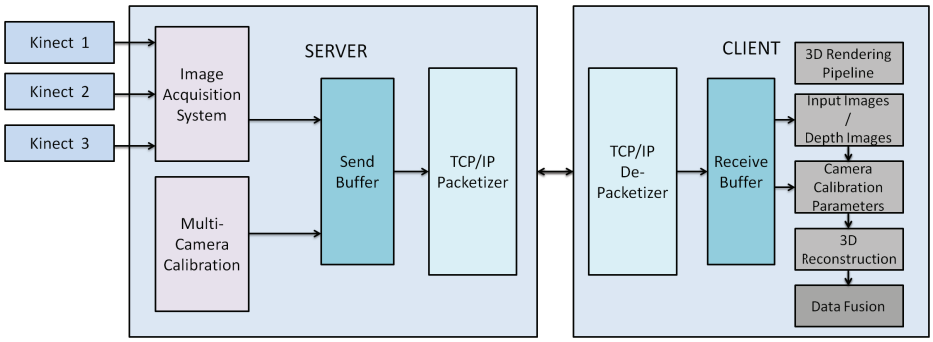

Figure 1. Our Online Surgical Training System.

Figure 2. (a) Microsoft Kinect has a microphone array, an infrared projector, an infrared sensor and a VGA camera. (b) Acquisition system consists of a set of three Microsoft Kinect cameras.

1. 3D Video Steaming System

Figure 1 shows the architect of our system. Our processing pipeline is composed of three components: 1) the server calibrates a set of three Microsoft Kinect sensors and uses them to acquire RGB-D images. 2) The communication framework efficiently packs the data into the send buffer and streams it via communication protocols based on Windows Socket API. 3) The client first unpacks the calibration information and RGB-D images from the receive buffer and uses them to render and fuse the three views to a 3D point cloud. Our system allows the user at the client side to witness the captured scene in real-time.

1.1. Server and Client

Figure 2 shows our image acquisition system. It consists of the server and three Microsoft Kinect sensors to capture the depth and color information of the scene. We build our system by using the open source Nestk library [10] and synchronize the three cameras. We first save the color information as 24 bit RGB image and the depth information as 16 bit image, and then compress both color and depth raw data for fast data transmission.

Similar to previous approaches [3,4,11,12,13], we first separately pre-calibrate the intrinsic parameter for each Kinect. The Nestk library [10] supports stereo matching and image warping between the color and the depth images so that each pixel on the depth image corresponds to the pixel at the same location on the color image. As a result, our calibration process only needs to find the intrinsic/extrinsic parameters for the color camera. Specifically, we use each Kinects color camera to capture 30 images of a rotating 10×7 checkerboard. Next, we use Zhang's approach [14] to accurately recover the intrinsic parameters. To estimate the extrinsic parameters of the color cameras during capture, we position the checkerboard approximately at the center of the scene and estimate the relative positions of the cameras with respect to the top-left corner of the checkerboard. The centroid of the scene is then used as the origin of the 3D world.

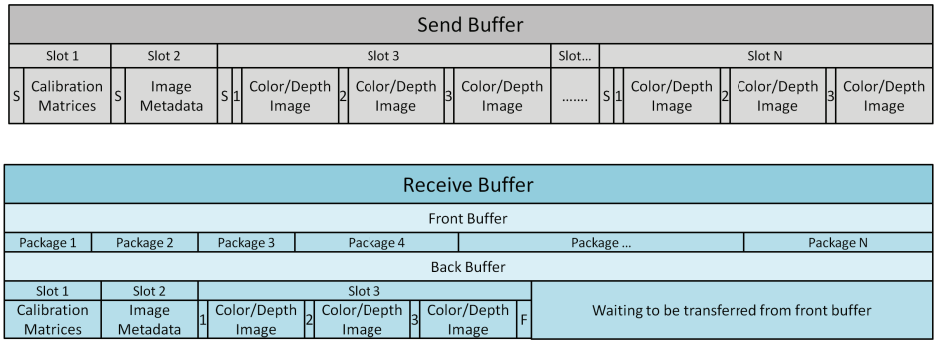

Figure 3. Our buffer design.

1.2. Communication Framework

Our communication framework consists of two parts: the sending and receiving processes. They are designed as separate threads on Server and Client so that they do not interfere with the capturing and rendering threads.

As shown in Figure 3, the sending process first packs the calibration information and the captured RGB-D images into the send buffer. Specifically, we divide our send buffer into different slots with a special symbol indicating the start of each slot. At the beginning of each slot, we use 64 bits to record the size of this slot for later unpacking. We place the calibration information for three Kinect sensors at the first slot because the location and orientation of Kinects are fixed during capturing. Next, we extract the image's metadata from the first frame of images and pack them into the second slots. Note that we only send the 1^{st} and 2^{nd} slots once for compression. After the calibration and image information have been sent, we consecutively pack each frame of three views into the following slots with each slot containing three color and depth images in different blocks. We use the first 2 bits to record the view number to distinguish images from different views. We also use the next bit to indicate if the image is color information or depth information. We assign the rest of the block to save the image data.

The receiving process receives and unpacks the data using the receive buffer. To efficiently unpack the received data, we design our receive buffer with a two-layer structure: the front buffer is used to rearrange the received packages and the back buffer is used to unpack the rearranged data for further rendering.

Note that the Socket API does not guarantee that each received data is a complete slot. Therefore we first put the received data in the front buffer and check if the size of the current data in the front buffer is larger than the size indicated at the beginning of the buffer. If so, we move the data into the back buffer. Otherwise, we wait for the new data to come in.

The back buffer contains complete slots transferred from front buffer. Each time the back buffer receives a new slot, the client sends 1 bit acknowledgement signal informing server the arrival of new slot with given size. The server does not send next slot until it receive the acknowledgement signal. This mechanism maintains the clients buffer from being flushed by the server's messages.

2. 3D Scene Reconstruction

We reconstruct the 3D scene using the approach proposed by Guo et al. [1]. Here we briefly reiterate their method.

Data Fusion. The two-pass rendering approach first recovers a point cloud for each viewpoint, and then fuses the results to generate a global 3D representation of the scene. Given a depth-RGB image pair, we first trace a ray for each pixel in the image and utilize the depth data to find the corresponding 3D coordinates in world space. Next, we uses the camera calibration parameters to transform the point cloud of each Kinect camera from local coordinates into a global coordinate system. Then we fuse multiple point clouds into one global representation. Notice that Kinect is designed as a stand-alone solution that delivers quite robust depth maps, while simultaneously running multiple sensors may lead to deteriorated results.

3D Rendering. With the generated point cloud, we begin to render a 3D stereoscopic view of the scene. Traditional 3D rendering generates a single perspective view by synthesizing a pinhole camera image in the scene. We extend this approach by simultaneously setting two cameras in the scene with a user specified baseline so that two cameras capture two views of the point cloud and render them as a red-cyan anaglyph. We also utilized the NVIDIA 3D API to render two regular color images and synchronize with the NVIDIA 3D glasses to deliver a better user experience.

3. Results and Discussions

We use a PC with a 3.2GHz i7-3930k processor to communicate with the Kinect sensors through USB 2.0 interfaces. Our 3D reconstruction client runs on a 3.5GHz i7-2700K machine equipped with an NVIDIA GeForce GTX 480 graphic card. Our server and client are interconnected in a Fast Ethernet local area network at 100 MB/s.

The basic transmission rate is determined by the clients rendering speed and desired frame per second (fps). For our system, the rendering speed is around 15 fps while our transmission rate is above 20 fps. Specifically, the color image size is less than 100 KB and depth image size is less than 120 KB after compression. During the experiment, our network transmission rate is 14 MB/s, which indicates that the server transmits 21 frames in one second to the client (14MB / ((100KB + 120KB)$\times$3) $\approx$ 21fps). The results demonstrate that our system can effectively capture and reconstruct 3D video in real-time.

Figure 4 shows our 3D view tests which capture a subject playing chess. Figure 5 shows the 3D stereoscopic view using red-cyan anaglyph. The results show that our system is capable of delivering live 3D data to a remote site in real time. We are currently coordinating with the VEST Center at the Christiana Care Health Service in Delaware to record and stream live surgical training to the University of Delaware.

4. Conclusions and Future work

We have developed a novel real time immersive surgery training system by coupling the emerging 3D imaging with advanced computer vision and graphics techniques. Specifi-

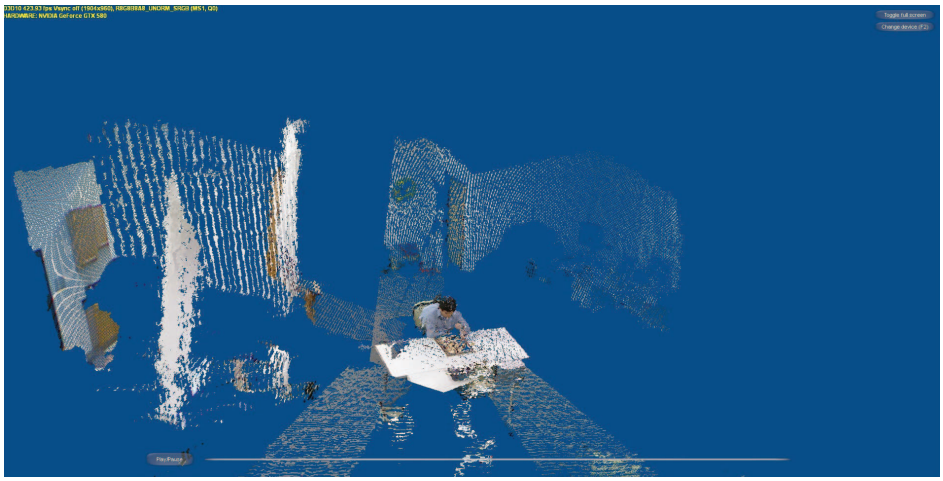

Figure 4. We use our system to acquire a subject playing chess. The figure shows the 2D rendering of the streamed data.

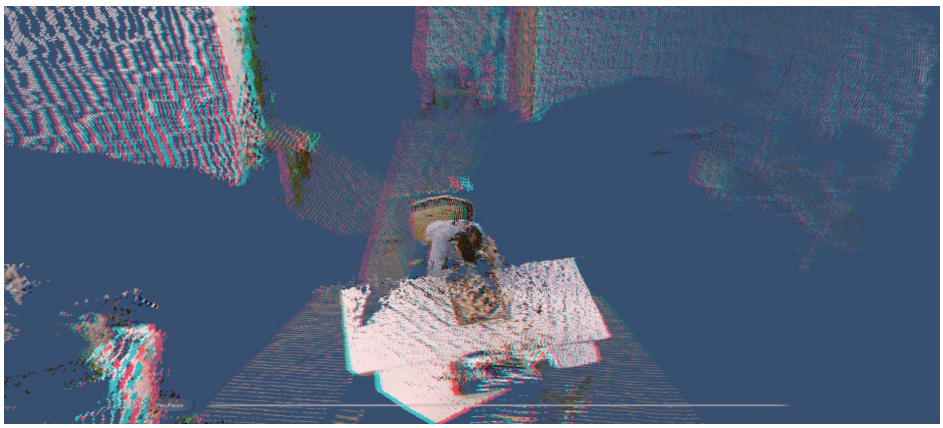

Figure 5. The same scene as Fig. 4 but we render the streamed data using 3D rendering.

cally, our system is based on a client-server model for real-time streaming. On the server front, we efficiently fuse multiple Kinect data acquired from different viewpoints and compress the data to stream to the client. On the client front, we build an interactive space-time navigator to allow remote users to witness the surgical procedure in real-time. We have validated our system by several preliminary tests of surgical training. In the future, we will explore possible integration with advanced image processing techniques filters in our system to achieve better 3D reconstruction results. We will also investigate how to support a large of number of clients at the same time as well as to provide 2D alternatives of our 3D data when the client does not have a 3D display. Finally, we plan to work closely with surgeons and residents to evaluate and improve the design of our system.

5. Acknowledgments

This project is supported by the Delaware INBRE under a grant from NIGMS (8P20 GM103446) at NIH.

References

[1] Xinqing Guo, Luis D. Lopez, Zhan Yu, Karl V. Steiner, Kenneth E. Barner, Thomas L. Bauer, and Jingyi Yu. A portable immersive surgery training system using rgb-d sensors. In *MMVR*, pages 161–167, 2013.

[2] H. Fuchs and U. Neumann. A vision of telepresence for medical consultation and other applications. In *Proceedings of the Sixth International Symposium on Robotics Research*, pages 565–571, 1993.

[3] Ramesh Raskar, Greg Welch, Matt Cutts, Adam Lake, Lev Stesin, and Henry Fuchs. The office of the future: a unified approach to image-based modeling and spatially immersive displays. ACM SIGGRAPH, pages 179–188, 1998.

[4] L.-Q. Xu, B. Lei, and E. Hendriks. Computer vision for a 3-d visualisation and telepresence collaborative working environment. *BT Technology Journal*, 20(1):64–74, January 2002.

[5] E. Trucco, K Plakas, Nicole Brandenburg, Peter Kauff, Michael Karl, and Oliver Schreer. Real-time disparity maps for immersive 3-d teleconferencing by hybrid recursive matching and census transform. In *ICCV, Proceeding of Workshop on Video Registration*, 2001.

[6] Henry Fuchs, Gary Bishop, Kevin Arthur, Leonard McMillan, Henry Fuchs Gary Bishop, Ruzena Bajcsy, Sang Wook Lee, Hany Farid, and Takeo Kanade. Virtual space teleconferencing using a sea of cameras. In *Proc. First International Conference on Medical Robotics and Computer Assisted Surgery*, pages 161–167, 1994.

[7] Oliver G. Staadt, Markus H. Gross, Andreas Kunz, and Markus Meier. The blue-c (poster session): integrating real humans into a networked immersive environment. In *Proceedings of the third international conference on Collaborative virtual environments*, CVE '00, pages 201–202, 2000.

[8] Kok lim Low, Adrian Ilie, Greg Welch, and Anselmo Lastra. Combining head-mounted and projector-based displays for surgical training. In *in Proceedings of IEEE Virtual Reality 2003. Los*, pages 110–117, 2003.

[9] Greg Welch, Andrei State, Adrian Ilie, Kok-Lim Low, Anselmo Lastra, Bruce Cairns, Herman Towles, Henry Fuchs, Ruigang Yang, Sascha Becker, Dan Russo, Jesse Funaro, and Andries van Dam. Immersive electronic books for surgical training. *IEEE MultiMedia*, 12(3):22–35, July 2005.

[10] Nestk library. https://github.com/nburrus/nestk.

[11] Yuanjie Zheng, Chandra Kambhamettu, Jingyi Yu, Thomas Bauer, and Karl Steiner. Fuzzymatte: A computationally efficient scheme for interactive matting. In *CVPR*, 2008.

[12] Yuanjie Zheng, Jingyi Yu, Chandra Kambhamettu, Sarah Englander, Mitchell D. Schnall, and Dinggang Shen. De-enhancing the dynamic contrast-enhanced breast mri for robust registration. In *Proceedings of the 10th international conference on Medical image computing and computer-assisted intervention - Volume Part I*, MICCAI'07, pages 933–941, 2007.

[13] Yuanjie Zheng, Karl Steiner, Thomas Bauer, Jingyi Yu, Dinggang Shen, and Chandra Kambhamettu. Lung nodule growth analysis from 3d ct data with a coupled segmentation and registration framework. In *ICCV*, pages 1–8, 2007.

[14] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11):1330–1334, November 2000.