M-TEEVE: Real-Time 3D Video Interaction and Broadcasting Framework for Mobile Devices

Shu Shi
Dept. of Computer Science
University of Illinois at Urbana-Champaign
shushi2@illinois.edu

Won J. Jeon
Dept. of Computer Science
University of Illinois at Urbana-Champaign
wonjeon@illinois.edu

Klara Nahrstedt
Dept. of Computer Science
University of Illinois at Urbana-Champaign
klara@illinois.edu

Roy H. Campbell
Dept. of Computer Science
University of Illinois at Urbana-Champaign
rhc@illinois.edu

ABSTRACT
3D video is an emerging technology that promises immersive experiences in a truly seamless environment. In this paper, we present a new proxy-based framework to extend the 3D video experience to mobile devices. The framework has two major features. First, it allows audience to use mobile devices to change rendering viewpoints and interact with 3D video, and it supports different interaction devices to collaborate with each other. Second, the proxy compresses the 3D video streams before broadcasting to mobile devices for display. A novel view-dependent real-time 3D video compression scheme is implemented to reduce the requirements of both transmission bandwidth and rendering computation on mobile devices. The system is implemented on different mobile platforms and our experiments indicate a promising future of 3D video on mobile devices.

1. INTRODUCTION
Different from traditional multimedia applications, immersive applications have multiple representations of data objects to users, based on their physical contexts and interactions with devices, such as video, audio, graphics and so on. Therefore, they provide more immersive experiences in artificially created virtual space. Our previous TEEVE (Tele-immersive Environments for EVErybody) project [15] is a good example. Multiple 3D cameras are used to capture and reconstruct 3D video streams which are transmitted over high speed network and displayed on the rendering side. By mixing different 3D video streams together, remote users can be immersed into a shared virtual space. Furthermore, different viewpoints of the video can be selected as in computer graphics to obtain the best immersive experience.

IMMERSCOM 2009, May 27-29, Berkeley, USA
Copyright © 2009 978-963-9799-39-4
DOI 10.4108/ICST.IMMERSCOM2009.6299

Inspired by the success of using Wii remote to interact with 3D video in our tele-immersive dance experiments [14], we are paying more attention to using mobile devices in immersive applications. As computing environments become more ubiquitous, the recent development enables more physical sensors equipped on mobile devices, such as touch-screen, 3-axis accelerometer, GPS , compass, camera, and so on. These sensors not only enhance user interactions with the related input methods, but also create more user-centric applications. Moreover, as mobile devices become much more powerful, they have already been capable of providing similar level of computing experience as desktops and laptops. For example, many desktop 3D applications, especially games like Quake [2], have been already ported to different mobile platforms.

This paper introduces how we implement TEEVE system on mobile platforms. As a first step, we focus on the rendering side only. The goal is to use mobile devices to receive, render, display and interact with 3D video streams, so that the physical sensors embedded in mobile devices can be utilized to provide better interactive experience. However, simply porting desktop implementation to mobile platforms has a serious bottleneck in network bandwidth. A bandwidth at the Gbps level is required in TEEVE to maintain quality, real-time and immersive communication [16], while the wireless network environment for general mobile devices can only provide a few Mbps of bandwidth. Computational capability is another issue to be considered. Computation resources of mobile devices are still limited if there is no dedicated hardware acceleration for graphics and multimedia processing.

We propose a prototype of M-TEEVE (Mobile TEEVE), a proxy-based framework to solve these constraints. The framework has two major features. First, it allows audience to use mobile devices to change rendering viewpoints and interact with 3D video. Two interaction methods, eye tracking and orientation tracking, are studied. Moreover, the collaboration mode for different mobile devices to interact with the 3D video simultaneously is supported by proposing a general control protocol. Second, the proxy framework compresses the 3D video streams before broadcasting to mobile devices for display. A novel view-dependent real-time 3D video compression scheme is implemented to reduce the requirements of both transmission bandwidth and rendering computation on mobile devices.

The typical example of the early work in immersive experience is CAVE [6], which uses three, four, five or six walls covered by projected screens to form a room-sized cube. The movement of the user inside CAVE is detected by electromagnetic, acoustic, and optical tracking technologies. The scene projected on walls are changed dynamically according to the physical movements detected. For more personalized scenarios, dome-type of displays [3] are typically used in various sizes. Infrared emitter of Wii remote [8] has been popular for desktop virtual reality displays. However, these approaches are based on desktop computing which has less limitation in bandwidth and computation than mobile computing.

The rest of the paper is organized as follows. Section 2 introduces the overview of M-TEEVE. Details are discussed in Sections 3 and performance is evaluated in Section 4. Section 5 summaries related work and the final section concludes the whole paper.

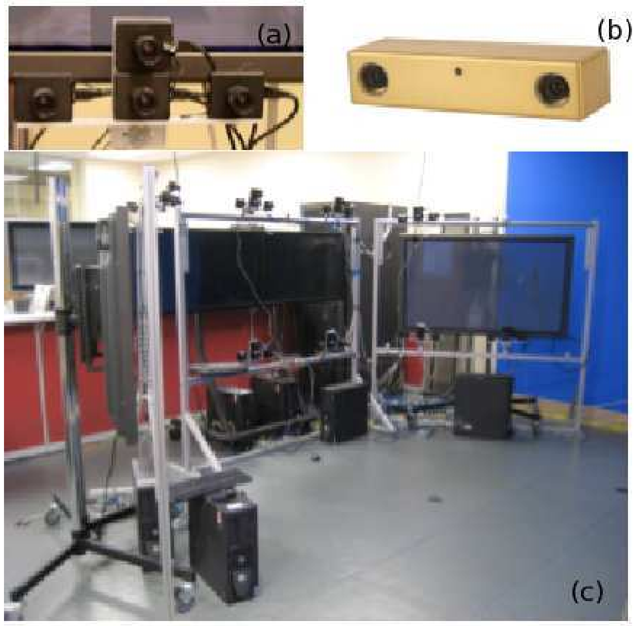

Figure 1: (a) 3D camera (c) BumbleBee 2 Stereo Camera (a) 3D Video Capturing Environment

2. M-TEEVE FRAMEWORK

2.1 3D Video

In our system, 3D video is represented by multiple depth image streams. The depth image is based on traditional 2D image, but each image pixel does not only have the color data, but also the depth data indicating the distance of the object from the camera. As a result, each pixel has the 2D position information in a image frame as well as the depth information and can be used to reconstruct the 3D coordinate in a 3D space. The depth image can be generated by many methods. In our project, we use a 3D camera, which is a group of four calibrated 2D camera (Figure 1(a)), to calculate the depth by stereo triangulation. We also test the commercial stereo camera, named BumbleBee 2 from Point Grey (Figure 1(b)) [1], to achieve a better frame rate. Multiple 3D cameras are located in different positions to capture all faces of the scene. Figure 1(c) shows how cameras are placed in our project. It is important to calibrate all cameras before capturing the scene. The calibration enables all pixels from multiple depth image streams can be reconstructed in a global coordination system.

On the rendering side, depth image streams are reconstructed and processed as point clouds. Therefore, 3D points can be merged with the other 3D video streams or graphic environments and manipulated as other graphic objects easily, such as moving the video object (point cloud) to different position, or change the viewpoint of viewing the video (we call it rendering viewpoint in the rest of this paper). This type of manipulation operation is considered as the interaction between the video viewer and 3D video. In this paper, we mainly focus on the interaction using mobile devices to change the rendering viewpoint.

2.2 Architecture

The proxy based framework of M-TEEVE is shown in Figure 2. Based on the original TEEVE framework, a proxy is added on the rendering side. It receives 3D video streams from the gateway server, collects the interaction control information from *control units*, adjusts the rendering viewpoint of 3D video and other properties such as ”pause” or ”stop” according to the control information, compresses 3D video streams, and broadcasts the compressed stream to all *display units*. The *control unit* is the device for users to interact with 3D videos. It generates the interaction control signals and sends these control information to the proxy. In M-TEEVE framework, the control unit can be any interesting user interaction method, for example, a computer program running on another workstation which sends control information corresponding to the rhythm of a music. In our research, we focus on using mobile devices with physical sensors as control units. The *display unit* receives the compressed 3D video stream and displays the 3D video on its own screen. It can be either the mobile device or the workstation with a big screen. The control unit can be used as the display unit as well although they are separated in the framework architecture.

The proposed framework features in its generality and scalability. It supports mobile devices as interaction and display devices while keeping compatible with previous desktop TEEVE implementation. Since the framework is not designed for any specific interaction device, the system is easily scaled to add support for new mobile devices.

2.3 Mobile Interaction

We are targeting two interaction methods using mobile devices, eye tracking and orientation tracking.

Eye tracking changes the rendering viewpoint by detecting the movement of viewer’s eyes. It has been exploited in different systems of virtual and augmented reality. Two com-

Digital Object Identifier: 10.4108/ICST.IMMERSCOM2009.6299
http://dx.doi.org/10.4108/ICST.IMMERSCOM2009.6299

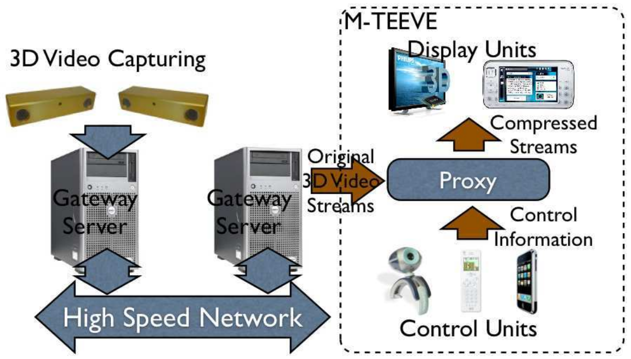

Figure 2: M-TEEVE Framework

mon approaches are to track either the physical location of eyes or the point of gaze (i.e., where we are looking). Different mechanisms for eye tracking result in different tracking capabilities and require different software/hardware specifications. Compared with tracking the point of gaze which requires additional hardware typically mounted to the head, tracking the physical location of eyes can be achieved relatively easily with a normal web camera which is equipped in many mobile devices. By comparing the physical location of eyes in different image frames, we can track the eye movement in 2D plane (left-right, up-down) and these movement will be translated into different control signals and adjust the rendering viewpoint accordingly. For example, when I move my eyes up, it usually means that I want to look at the upper side of the video objects. Consequently, the rendering viewpoint should also be lifted up and cover more upper side area of the video object.

Orientation tracking changes the rendering viewpoint by detecting the orientation of the device, which is achieved by accelerometers. Accelerometers are usually used to detect the hand movement and generate the motion vectors. Our previous Wii remote interaction approach [14] is based on these motion vectors. However, this approach is appropriate for wii remote but not for the mobile device which has the display screen and also acts as a display unit. For example, the motion of the hand, which is intended to change the rendering viewpoint, also brings the display screen away from the user. Therefore, we propose orientation tracking: the rendering viewpoint is changed according to the device's orientation, rather than the instantaneous motion. Figure 3 explains the idea. It correlates the device's orientation with the rendering viewpoint. For example, if the device is facing up, the viewer has to look downward above the device to get the best view. Then the facing up orientation can be linked with the rendering viewpoint of looking downward above the video objects.

However, although these methods provide natural and non-invasive interaction, they both have deficiencies. Eye tracking is only effective to detect the eye movement in 2D plane. The distance of two eyes can be measured to provide functions of zoom in/zoom out, but no other complicated 3D motions can be effectively recognized. Orientation tracking provides flexible navigation to any viewpoint direction in 3D space, but lacks the capability of zoom in/zoom out

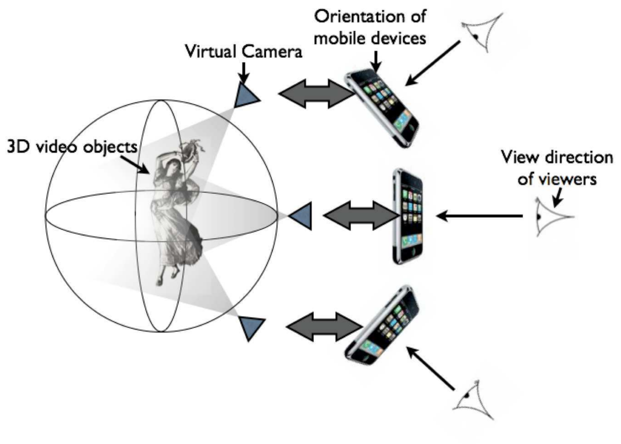

Figure 3: Orientation Tracking

and moving in the 2D plane while keeping the orientation fixed. As a result, it is important to combine two methods together for collaborative interaction.

Collaborative interaction means multiple control units are collaborating with each other to obtain a better interaction experience. For example, the mobile device which is equipped with both web camera and accelerometers can be used to run eye tracking and orientation tracking at the same time. The orientation of the device is used to decide the rendering viewpoint direction in 3D space. While keeping the orientaion fixed, eye tracking can be used for zoom in/zoom out and 2D navigation.

2.4 3D Video Compression

Compression techniques on depth image have been studied from many aspects. ATTEST [12] suggests using the standard video codec H.264 to compress both color and depth frames. The proxy-based compression proposed by Penta and Narayanan [11] and the skeleton-based compression introduced by Lien [9] both use pre-defined models to reduce data redundancy. Yang [16] proposed two general schemes for intra-stream compression and Kum [7] studied the inter-stream compression using reference stream. However, these techniques either need off-line processing or fail to meet the compression ratio required by wireless bandwidth.

Therefore, in M-TEEVE, a new view-dependent 3D video compression scheme is used in the proxy to compress original depth image streams before broadcasting to display units. The main idea of the compression scheme is to compress and transmit only useful data to mobile display units. As the proxy receives the control signals from control units, it knows exactly what the rendering viewpoint at the display units. Given the rendering viewpoint of the 3D video, the proxy pre-renders the 3D video scene for display units and finds all 3D points which are visible at current viewpoint. Other 3D points in the cloud are occluded or invisible at current viewpoint and can be highly compressed or even discarded if the bandwidth is limited. Moreover, for the standalone display unit which can't also act as the control unit (e.g., a monitor), the proxy can only send the 2D image of the pre-rendered scene for the display unit to display directly. For more details of the view-dependent compression scheme, please refer to our previous work [13]. This view-dependent compression scheme works only for 3D video compression but not 2D video or 3D graphics, because 2D video does not have the viewpoint property and 3D graphics does not update the scene every frame.

3. IMPLEMENTATION

3.1 Eye Tracking

Eye tracking is implemented using OpenCV computer vision library [4] (Figure 4). Internally, viewer's face is captured with the 2D web camera equipped with the mobile device and eyes are manually selected as the feature points at the beginning. Lucas-Kanade algorithm [10] is used to calculate the optical flow of consecutive frames in order to track the movement of feature points. This is a well-known method as the best combination of accuracy and speed for determining optical flows. By calculating optical flows of eyes, this software sensor provides not only directions of the movement (such as left, right, up, or down) of eyes, but the distance of the movement in 2-dimensional Cartesian coordinates. In addition, by measuring the change in distance between two eyes, the sensor tracks a user's zooming motion to the display.

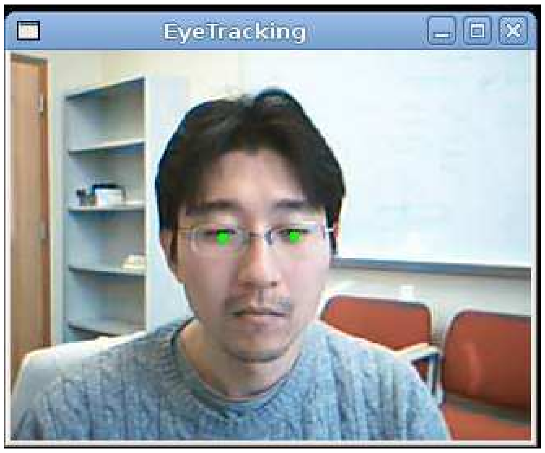

Figure 4: Eye Tracking

3.2 Orientation Tracking

Orientation tracking is achieved by using accelerometers, which are currently widely equipped in many game devices and mobile phones. Taking iPhone/iPod Touch as an example, three accelerometers (Figure 5) are embedded, one along each of the primary axes of the device. An accelerometer measures changes in velocity over time along a given linear path. So the combination of three accelerometers can detect movement of the device in any direction. Besides, it can also be used to track both sudden movements in the device and the device's current orientation relative to gravity.

For iPhone/iPod touch [5], the orientation of the device can be simply read from **UIDeviceOrientation** in **UIDevice** class. However, the obtained value does not give the exact orientation vector, but only indicates whether the device is in landscape or portrait mode or whether the device is face

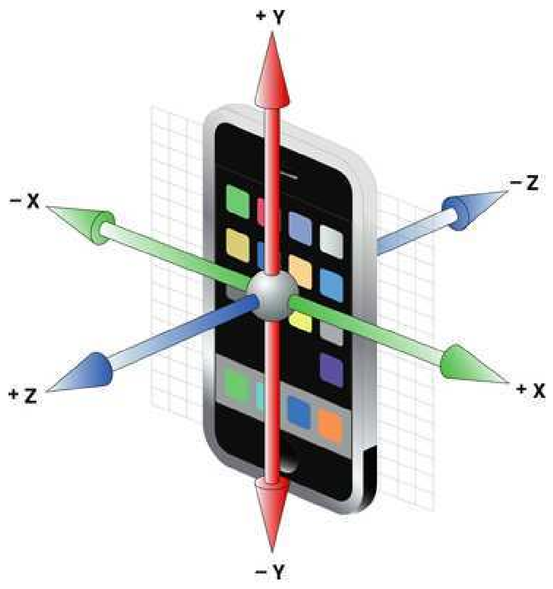

Figure 5: Accelerometer Axes in iPhone [6]

up or face down. The accurate orientation vector has to be extracted from raw data of accelerometers manually. In our implementation, we use a high data sampling frequency and a low-pass filter to reduce the influence of the sudden motion on the accelerometer data and extract the vector of gravity, which can be simply translated into the accurate orientation vector.

3.3 Collaborative Interaction

In our implementation, we propose an interaction protocol for different control units to exchange control information with proxy in order to achieve collaborative interaction. A general protocol is crucial because the control information from different interaction devices can be interpreted in the same representation and the proxy does not have to care about the detail features of various control units. Although we are focusing on only two interaction methods, we expect to include as much content in this protocol prototype as necessary for future extension. The protocol is divided into four main categories as listed in Table 1.

Table 1: Interaction Protocol

Category	Description	Example
ADMIN	System initialization, authentication and registration	*connect, login*
CONTROL	Adjust system parameters and video properties	*pause, stop*
NAVIGATE	Adjust the 3D video rendering viewpoint	*zoom, rotate*
ADVANCED	Query protocols for testing and debugging	*dump*

3.4 Compression and Broadcasting

The flow of compression and decompression is elaborated in Figure 6 [13]. However, in M-TEEVE framework, the compression scheme could be further simplified due to the separation of control units and display units. Since the display

Digital Object Identifier: 10.4108/ICST.IMMERSCOM2009.6299
http://dx.doi.org/10.4108/ICST.IMMERSCOM2009.6299

unit does not change the rendering viewpoint, the viewpoint module in the decompression flow can be removed. As a result, the enhanced layer in the stream frame is not necessary and the corresponding module in compression flow to generate enhance layer can also be removed.

For the network transmission, both push and pull modes are supported. For the push mode, the proxy broadcasts the compressed video stream to all display units. Since the current network does not provide enough support for broadcast and multicast, each display unit needs to register with the proxy to receive the compressed video streams. The proxy sends the packets to each registered display unit through UDP unicast. For the pull mode, the display unit may request the specific video frame from proxy through TCP connection. This mode is provided for some display units which try to prevent the quality loss brought by the unstable UDP transmission.

4. EXPERIMENTS

We have implemented M-TEEVE for Nokia N800 tablet and Apple iPod Touch. Nokia N800 is a Linux based PDA and has the front-facing camera for eye tracking. The ongoing implementation will be on NVidia Tegra, which is greatly enhanced in graphics rendering and video decoding. (Figure 7).

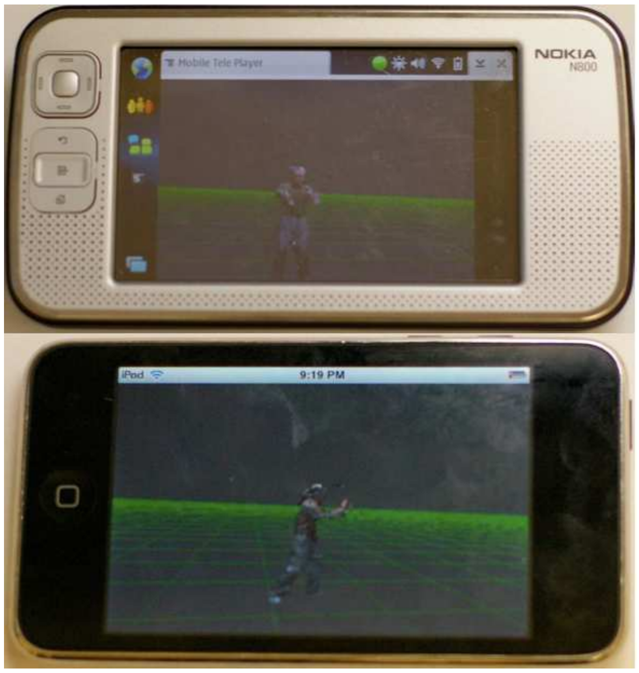

Figure 7: Experimenting Mobile Devices: Nokia N800, Apple iPod Touch

Experiments indicate the impressive performance of our proxy-based system in terms of transmission bandwidth and frame rate of 3D video playback for mobile devices. Table 2 shows the improvement achieved by Nokia N800 (TI OMAP2420 330 MHz CPU and 128MB memory, 802.11g Wi-Fi). In the table, the display frame rate and the transmission bandwidth of proxy-based M-TEEVE framework are compared with original TEEVE implementation which is simply ported to mobile platform. As we can see obviously, the original implementation has a poor performance in mobile devices. The frame rate is still less than 1 fps even when only 3 out of total 12 camera streams are transmitted and processed. However in M-TEEVE framework, all 12 video streams can be transmitted to mobile devices. Even when the 3D video is displayed at the full screen resolution (no device in our experiment has the display resolution larger than 1024*768), the bandwidth requirement can still be easily achieved in the current Wi-Fi network environment. The display frame rate for the small display resolution can be further improved by using a more powerful workstation as the proxy. However, since the overall 3D video frame rate of our tele-immersive applications remains around 10 fps [15], the frame rate of M-TEEVE shown in the table is enough for existing applications.

Table 2: Performance of 3D Video Display on Nokia N800

Display	Frame Rate(fps)		Bandwidth(bps)	
Resolution	M-TEEVE	Orig	M-TEEVE	Orig
176*144	12.9	0.9	414K	
320*240	11.6	0.9	742K	280K
640*480	7.6	0.8	1.2M	(3 camera
1024*768	4.7	0.8	1.6M	streams)

Another important performance issue is the control delay. The control delay refers to the delay time from the behavior of interaction is generated by user till the expected effects appear on the display screen. The control delay can be divided into two parts. The first part is the control detection delay and the second part is the control transmission delay. In our experiments, eye tracking incurs a large control detection delay since we notice that running eye racking algorithm in mobile devices is very expensive. Figure 8 compares the eye tracking performance on Nokia N800 and a Dell laptop (Pentium M 1.7GHz, 1GB Memory). It takes on average 220 ms for the PDA to detect the eye movement on the captured frames. Orientation tracking does not have this problem, since it replies on the data provided by the hardware sensors and no complicate algorithm is needed to analyze the data. For the control transmission delay, since all control signals are transmitted through network, the delay can be variant from time to time. In our experiments, the average network transmission delay time is no more than 150 ms. However, the network delay could be increased under heavy network load, which impairs the user interactive experience. We will focus more on reducing the network delay in our future research.

5. CONCLUSIONS

We proposed M-TEEVE framework in this paper to support mobile devices in tele-immersive applications. The physical sensors provide new 3D video interactive experience by moving eyes or changing the orientation of mobile devices. The proxy-based framework does not only allow different mobile devices to collaborate during interaction, but also reduce the transmission bandwidth and rendering computation for display devices. Experiments shows the great progress of mobile performance in 3D video applications as well as the deficiency of control delay, which we will try to improve in our future work.

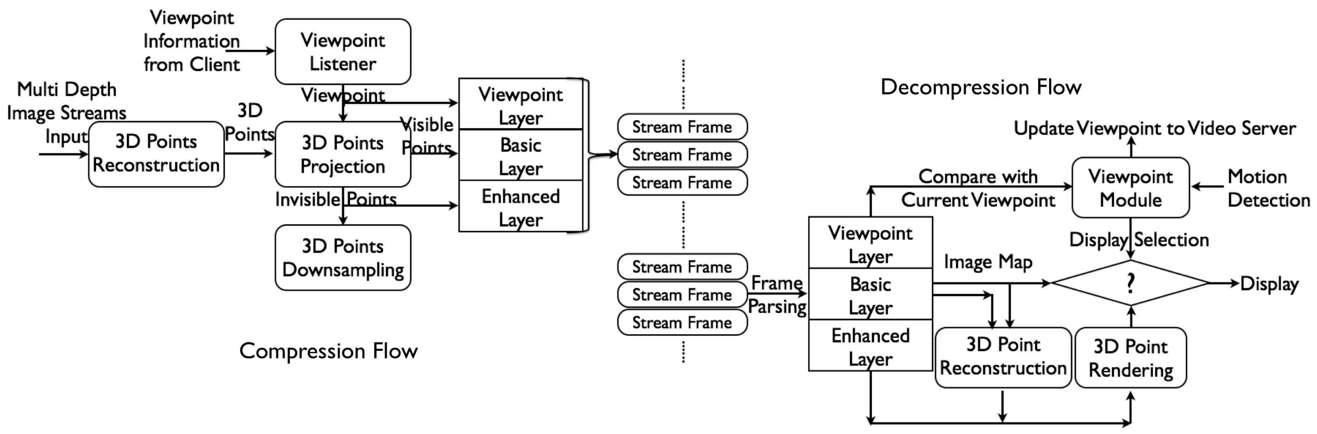

Figure 6: Compression and Decompression Flow

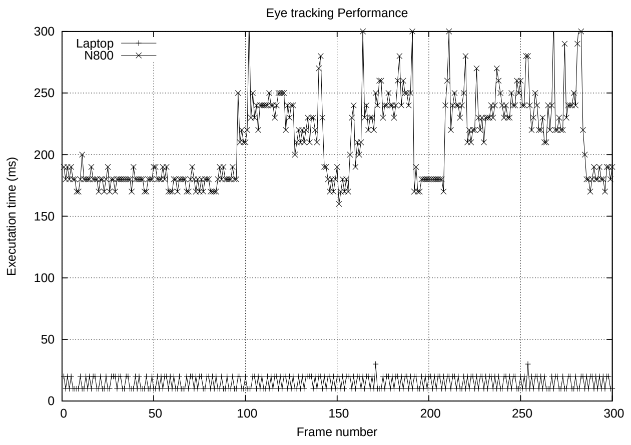

Figure 8: Eye Tracking on Laptop and Nokia N800

6. ACKNOWLEDGMENTS

This work is supported by the National Science Foundation under Grant CNS 05-20182 and CNS 07-20702. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. We also thank Samsung for providing research funding and NVIDIA for sponsoring Tegra APX development board for experiments.

7. REFERENCES

[1] Bumblebee 2 camera. `http://www.ptgrey.com/products/bumblebee2/index.asp`.

[2] id software: Quake. `http://www.idsoftware.com/games/quake/quake/`.

[3] Immersive display solutions. `http://www.immersivedisplayinc.com`.

[4] Open computer vision library. `http://sourceforge.net/projects/opencvlibrary/`.

[5] Apple. iphone application programming guide. http://developer.apple.com/iphone/.

[6] C. Cruz-Neira, D. Sandin, T. DeFanti, R. Kenyon, and J. Hart. The cave: Audio visual experience automatic virtual environment. *Communications of the ACM*, 35(6):65–72, June 1992.

[7] S.-U. Kum and K. Mayer-Patel. Real-time multidepth stream compression. *ACM Trans. on Multimedia Computing, Communications and Applications (TOMCCAP)*, 1(2):128–150, 2005.

[8] J. Lee. Head tracking for desktop vr displays using wii remote. `http://www.cs.cmu.edu/~johnny/projects/wii/`.

[9] J.-M. Lien, G. Kurillo, and R. Bajcsy. Skeleton-based data compression for multi-camera tele-immersion system. In *Proc. of Advances in Visual Computing, Third International Symposium (ISVC'07)*, pages 714–723, November 2007.

[10] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision.

[11] S. K. Penta and P. J. Narayanan. Compression of multiple depth maps for ibr. *The Visual Computer*, 21(8-10):611–618, 2005.

[12] A. Redert and et al. Attest: Advanced three-dimensional television system technologies. In *Proc. of 1st International Symposium on 3D Data Processing Visualization and Transmission (3DPVT'02)*, pages 313–319, June 2002.

[13] S. Shi, K. Nahrstedt, and R. H. Campbell. View-dependent real-time 3d video compression for mobile devices. In *Proc. of ACM International Conference on Multimedia (MM'08)*, Vancouver, BC, Canada, 2008.

[14] M. Tamai, W. Wu, K. Nahrstedt, and K. Yasumoto. A view control interface for 3d tele-immersive environments. In *Proc. of IEEE International Conference on Multimedia & Expo (ICME'08)*, Hannover, Germany, 2008.

[15] Z. Yang and et al. Teeve: The next generation architecture for tele-immersive environments. In *Proceedings of the 7th IEEE International Symposium on Multimedia (ISM'05)*, Irvine, CA, 2005.

[16] Z. Yang and et al. Real-time 3d video compression for tele-immersive environments. In *Proc. of SPIE/ACM Multimedia Computing and Networking (MMCN'06)*, 2006.