Real-Time Remote Rendering of 3D Video for Mobile Devices

Shu Shi, Won J. Jeon, Klara Nahrstedt, and Roy H. Campbell
Department of Computer Science
University of Illinois at Urbana-Champaign
201 N Goodwin Ave., Urbana, IL 61801, USA
{shushi2, wonjeon, klara, rhc}@illinois.edu

ABSTRACT

At the convergence of computer vision, graphics, and multimedia, the emerging 3D video technology promises immersive experiences in a truly seamless environment. However, the requirements of huge network bandwidth and computing resources make it still a big challenge to render 3D video on mobile devices at real-time. In this paper, we present how remote rendering framework can be used to solve the problem. The differences between dynamic 3D video and static graphic models are analyzed. A general proxy-based framework is presented to render 3D video streams on the proxy and transmit the rendered scene to mobile devices over a wireless network. An image-based approach is proposed to enhance 3D interactivity and reduce the interaction delay. Experiments prove that the remote rendering framework can be effectively used for quality 3D video rendering on mobile devices in real time.

Categories and Subject Descriptors

H.5.1 [**Multimedia Information System**]: Video

General Terms

Design, Measurement

Keywords

3D Video, Mobile Devices, Remote Rendering

1. INTRODUCTION

3D video has recently attracted many research interests. Different from traditional 3D graphics, 3D video is captured and reconstructed from the real world in real time while the 3D attribute allows the video objects to be rendered at arbitrary viewpoint or merged with virtual 3D background. Many research results indicate a bright future of 3D video in tele-immersion and virtual reality. TEEVE (Tele-immersive Environments for EVErybody) project [24] is a good example. 3D video is captured and reconstructed by multiple 3D cameras. Different 3D video streams of different parties are exchanged over a high speed network and mixed together into a shared virtual space. Users who are physically apart can be rendered on the same display screen as if they are standing next to each other.

As computing environments become more ubiquitous, mobile devices are capable of providing similar computing experiences to desktops and laptops. However, the major bottleneck of network bandwidth makes it a big challenge to implement TEEVE on mobile platforms and render 3D video streams on mobile devices. A Gbps level bandwidth is required for real-time immersive communication [25], while the wireless network environment for general mobile devices can only provide a few Mbps of effective bandwidth. Besides, although 3D rendering has been enhanced tremendously in many smart phones and PDAs, rendering 3D video streams at an interactive frame rate is still beyond the computation capacity of most devices.

Remote rendering provides a simple but effective solution. Both bandwidth and computation problems are solved by rendering 3D video remotely on a powerful server and sending the 2D image of rendered scene back to mobile devices to display [20]. The idea of remote rendering is not new. In the early days when PC did not have enough computing power to render complex graphic models, the framework of remote rendering has been proposed.

However, a new problem of interaction delay is brought into the scope. When the user tries to change the rendering viewpoint of a 3D video, the request goes back to the rendering server and it takes at least a round trip network delay for the user to see the response. As the network traffic becomes congested, the interaction delay can finally increase to intolerable for interactive applications. In addition, compared with the rendering of static graphic models in previous work, 3D video rendering has several different features we need to consider in the design of a new rendering framework.

- *Dynamic*: 3D video is dynamically updated in every frame
- *Real-time*: processing on each frame has to be finished before the next frame arrives
- *Temporary*: every frame expires after the next frame arrives

In this paper, we present an integrated system design of a general proxy-based framework for real-time remote 3D video rendering for mobile devices with constrained interaction delay. On the mobile device side, the approach based on

MM'09, October 19–24, 2009, Beijing, China.

3D warping is proposed to reduce the interaction delay while maintaining the computation workload at a low level. On the proxy side, new algorithms of reference frame selection are derived to alleviate exposure defects. The algorithm can be easily implemented in a parallel execution environment provided by modern GPU or cloud clusters. Experiments indicate a real-time 3D video rendering performance up to 15 fps can be achieved with satisfying image quality.

For the rest of the paper, we first summarize the related work in several areas. An overview of the system architecture, data flow, proxy component, and user interaction is given in Section 3. Image-based approach is introduced in detail in Section 4. Performance is evaluated in Section 5 and Section 6 concludes the paper.

2. RELATED WORK

2.1 3D Video

"3D video" has been used in the literatures for both stereo video and free viewpoint video. Smolic *et al.* summarizes the differences in [22]. The stereo video concentrates on providing different views from different eyes to offer a 3D depth impression. It is tightly related with the stereo vision and 3D display technologies. The free viewpoint video allows video viewers to interactively select different viewpoints just as dealing with computer graphics. However, the boundary of these two categories of media is not clear in research nowadays - the free viewpoint video can be presented as stereo video by rendering two views simultaneously for both eyes, and the stereo video also needs to provide more than two views as the emergence of multi-view 3D display technology. In this paper, we focus on the 3D video with the attribute of free viewpoint navigation.

Various approaches of 3D video have been developed in the past. Matusik and Pfister [14] implemented a 3D TV prototype system for real-time acquisition, transmission, and display of dynamic scene. Multiple video streams are individually captured, encoded, and transmitted over broadband network to the multi-projector 3D display. A dense lightfield representation is used to generate multiple views. ATTEST [18] demonstrated a full 3D TV chain including content generation, stream compression, broadcasting, and 3D display. The depth image is used as video representation and different views are synthesized by image based rendering techniques. Wurmlin *et al.* [23] presented the 3D video recorder to record, process, and play 3D video. A shape-from-silhouette technique is used to reconstruct 3D models from captured 2D video streams and a point-based 3D video representation is used for transmission and storage.

Although different approaches have been used to generate 3D video in different projects, all these approaches make use of multiple views of the same scene. Multiple cameras are placed at different positions to capture all facets of the object so that the user can visually navigate the scene freely. As a result, huge bandwidth and computation required to process multiple 3D video streams become the barrier which prevents researchers from porting the system to differentmobile platforms.

2.2 Remote Rendering

Numerous remote rendering systems appeared in last two decades. Here we summarize several mobile device projects related to our work.

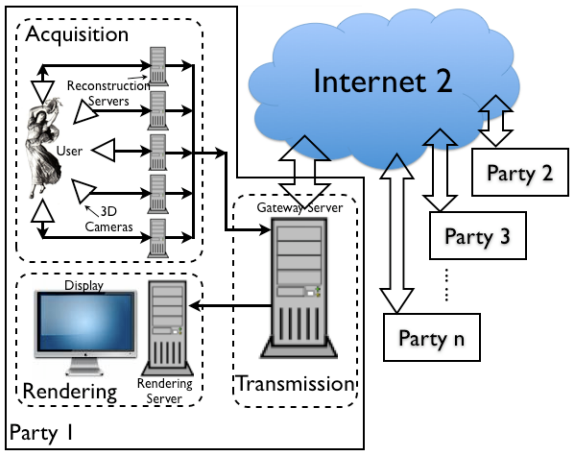

Figure 1: TEEVE Architecture

Chang and Ger [9] used image based rendering with a client-server framework to render complex graphic models on Pocket PC. The server renders the model and sends the result image to the client, which runs a 3D warping algorithm to update the display at interactive rates. Layered Depth Image [19] is implemented to avoid exposure artifacts.

Bao and Gourlay [7] built a mobile walkthrough environment, which is also based on the client-server framework and the 3D warping algorithm. The difference from the previous work is that superview warping and view compensation are proposed to overcome exposure problems. The server renders the scene at a resolution larger than display screen size to cover exposure gaps at the edge. The view compensation is similar to the motion compensation in 2D video compression - the difference of the rendered image and warped image at the new viewpoint is calculated on the server and transmitted to the client to compensate the warping error.

Boukerche and Pazzi [8] proposed a hybrid technique combining remote geometry rendering and image warping. The server renders a partial panoramic view and warps the image into cylindrical coordinate. The client extracts the required scene from a panoramic representation. Interaction latency and issues of streaming in lossy network were also studied.

Lamberti and Sanna [11] presented their cluster-based remote rendering framework. The server can run rendering and video encoding concurrently in distributed nodes to improve the performance effectively. They also discussed in depth about different timing factors of video streaming in remote rendering.

3. SYSTEM DESIGN

In this section, first we briefly introduce how 3D video is generated, transmitted, and rendered in TEEVE system, followed by the proxy-based framework design. At the end of the section, we introduce the design of user interaction methods and its importance.

3.1 TEEVE

TEEVE [24] aims to provide a seamless 3D video tele-immersive experience for broader audience by integrating existing computing and networking components. Figure 1 elaborates the system architecture of TEEVE. The whole

system comprises acquisition, transmission, and rendering three parts.

For the acquisition part, 3D cameras are used to generate multiple depth image streams. The depth image is based on traditional 2D image. Each image pixel has not only the color data, but also the depth data indicating the distance between objects and the camera. Thus, each pixel has both 2D position information in a image frame and the depth information, which is used to reconstruct the 3D coordinate in a 3D space. The depth image can be generated by many methods. In our project, the 3D camera we use is a group of four calibrated 2D cameras (Figure 2(a)) connected with a workstation for depth calculation. Three cameras below are used to calculate the pixel depth by stereo triangulation and the one above records the pixel color. The frame rate of depth map acquisition is around 10 fps. We are also testing BumbleBee 2 stereo camera from Point Grey (Figure 2(b)) [1], which improves the depth acquisition performance up to 20 frame per second. Multiple 3D cameras are placed in different positions to capture all facets of the visual scene. Figure 2(c) shows our experimental environment and how cameras are placed. It is important to calibrate all cameras before capturing the scene, so that all pixels from multiple depth image streams can be reconstructed in a global coordination system.

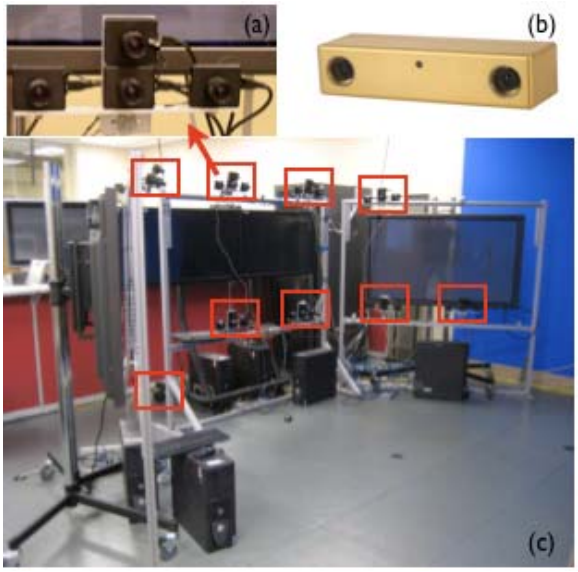

Figure 2: (a) 3D camera, (b) BumbleBee 2 Stereo Camera, (c) 3D Video Capturing Environment

For the transmission part, a gateway server aggregates and exchanges all 3D video streams generated from multiple cameras with other gateway servers over high speed Internet 2. The gateway server hides the network transmission details from other parts and plays important roles in video streams synchronization and QoS management. Techniques such as viewcast [26] are implemented to improve transmission quality efficiently.

For the rendering part, a rendering server receives streams from the gateway server. Depth images of the same 3D video stream are reconstructed and processed as a 3D point cloud. Point clouds of different 3D video streams can be merged together and rendered with other graphic models. The user can interact with 3D video by changing the rendering viewpoint or moving the 3D video objects to different positions, etc. This paper concentrates only on the user interaction of rendering viewpoint change.

3.2 Proxy-Based Rendering

In order to render 3D video streams on mobile devices, we replace the rendering part in TEEVE system with mobile devices and a powerful server as the proxy (Figure 3). The proxy comprises four components. Input interface is used to communicate with the gateway server and receive 3D video streams. Rendering module processes the rendering of 3D video at the resolution of mobile screen size. Output interface sends the generated image frames to mobile devices through output interface. Motion module manages the rendering viewpoint according to the user interaction information collected from mobile devices. The mobile device only displays the image frames received from the proxy and updates the user interaction information to the proxy if the user tries to change the rendering viewpoint.

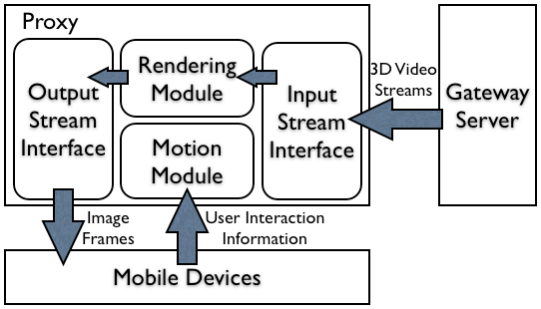

Figure 3: Proxy-Based Framework

This proxy-based framework provides two major benefits. First, the proxy offers enough network bandwidth to receive 3D video streams from the gateway server, and abundant computing resources for 3D rendering. Second, the proxy hides the details of 3D video representations from mobile devices. Although the current proxy is designed for the 3D video represented by depth images, it can be easily modified to be compatible with other 3D video formats while the implementation on each mobile platform remains unchanged.

However, if the proxy only sends the rendered 2D image frames to mobile devices, interaction delay becomes an issue. Figure 4 explains the timing sequence in the proxy-based rendering framework. Besides the network transmission time, other factors (e.g. 3D rendering, buffering, image encoding/decoding, etc.) can also contribute hundreds of milliseconds or even seconds to the interaction delay. The large interaction delay becomes a barrier for many applications that require frequent user interaction, such as games and real-time immersive communication. The best way to reduce the interaction delay is to allow the result frame to be rendered to the new rendering viewpoint on the local mobile device. The image-based approaches is proposed to achieve the local rendering and reduce the interaction delay. We introduce this approach in detail in Section 4.

3.3 User Interaction

Before introducing the details of rendering 3D video in the proxy server, we take a look at user interaction first. The

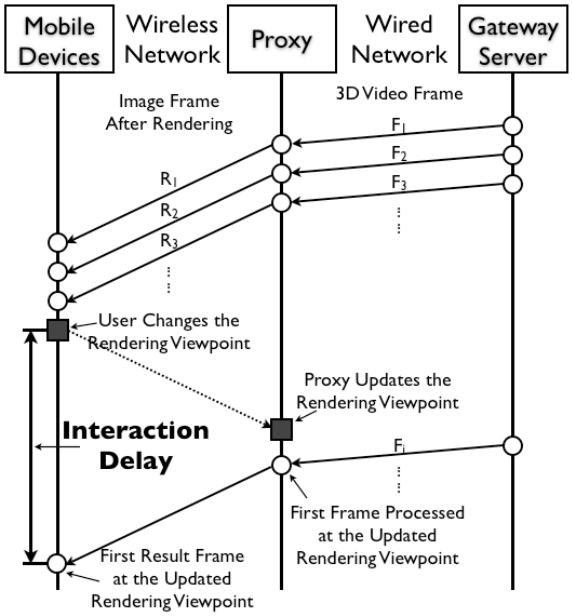

Figure 4: Interaction Delay

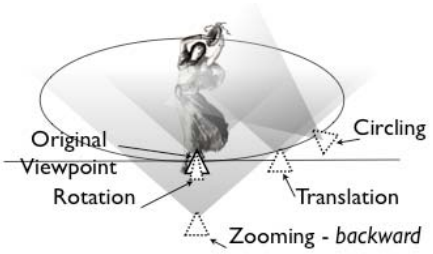

Figure 5: User Interaction Modes

design of user interaction methods has significant influence in user experience. 3D video allows user to interactively change the rendering viewpoint. The rendering viewpoint is decided by three camera parameters: *Pos* (camera position), *Dir* (look-at direction), and *Up* (up direction), each of which is a three-dimensional vector. However, it is not necessary to allow rendering viewpoint to be freely changed in the 9-dimensional space. Since 3D video is captured by fixed cameras, 3D video scene has the convex attribute.Therefore, the 3D video can be viewed effectively only from a limited range of viewpoints, which corresponds to the placement of capturing cameras. Besides, physical limitations of interaction devices also need to be considered for user interaction design. Most widely used interaction devices, such as button/keyboard and touch screen can only detect 1D or 2D motion effectively.

Based on these factors, we design four different user interaction modes for navigating 3D video - *translation*, *circling*, *zooming*, and *rotation*. *Translation* and *circling* allow viewpoint to change in four directions while *zooming* and *rotation* only allow two moving directions. The movement of each interaction is also discretized. The actions of each mode are summarized in the Table 1 and elaborated in Figure 5. The design maintains the accessibility of all necessary 3D video rendering viewpoints[1]. Moreover, it greatly simplifies the search space of motion prediction which we will discuss in Section 4.4.

Table 1: Four User Interaction Modes

Mode	Description	Direction
Translation	*Pos*: change a unit distance toward $\pm Dir \times Up$ or $\pm Up$; *Dir*: unchanged; *Up*: unchanged.	*Left, Right, Up, Down*
Circling	*Pos*: change a unit angle in the circle with the tangent toward $\pm Dir \times Up$ or $\pm Up$; *Dir*: remains pointing to the circle center; *Up*: unchanged.	*Left, Right, Up, Down*
Zooming	*Pos*: change a unit distance toward $\pm Dir$; *Dir*: unchanged; *Up*: unchanged.	*Forward, Backward*
Rotation	*Pos*: unchanged; *Dir*: unchanged; *Up*: change a unit angle in the circle with the tangent toward $\pm Dir \times Up$.	*Clockwise Anticlockwise*

4. IMAGE-BASED APPROACH

The image-based approach allows mobile devices to warp the received 2D images to the new rendering viewpoint immediately when the user interaction happens. Instead of waiting for the response from the proxy, the image warping on mobile devices reduces the interaction delay significantly.

As a well-known technique in image based rendering [21], 3D warping was first proposed by McMillan in 1995 [16]. Our implementation is different from previous work in dealing with exposure problems. In this section, we first briefly introduce how 3D warping works. Then we will propose our techniques to overcome exposure problems in the context of 3D video rendering.

4.1 3D Warping

McMillan fully explains the 3D warping algorithm in [15]. Camera parameters (i.e., camera position, view direction, focal length, and image resolution) and pixel depth values are required for input to the algorithm. When OpenGL is used for 3D rendering, the camera parameters can be obtained from *modelview* and *projection* matrix, and the pixel depth values are stored in *z-buffer*. The warping is computed as the following equation.

$$(u_2, v_2) = (\frac{u_1 a + v_1 b + c + \delta_1 d}{u_1 + v_1 j + k + \delta_1 l}, \frac{u_1 e + v_1 f + g + \delta_1 h}{u_1 i + v_1 j + k + \delta_1 l})$$

In the equation, the coordinates of (u_2, v_2) are calculated according to each input pixel at (u_1, v_1). δ_1, also named generalized disparity, is inversely related to the depth value of the pixel at (u_1, v_1) by a scale factor that is also determined by (u_1, v_1). Other variables a through l are decided

[1]We consider the rendering viewpoints where the 3D video scene can be seen as necessary rendering viewpoints.

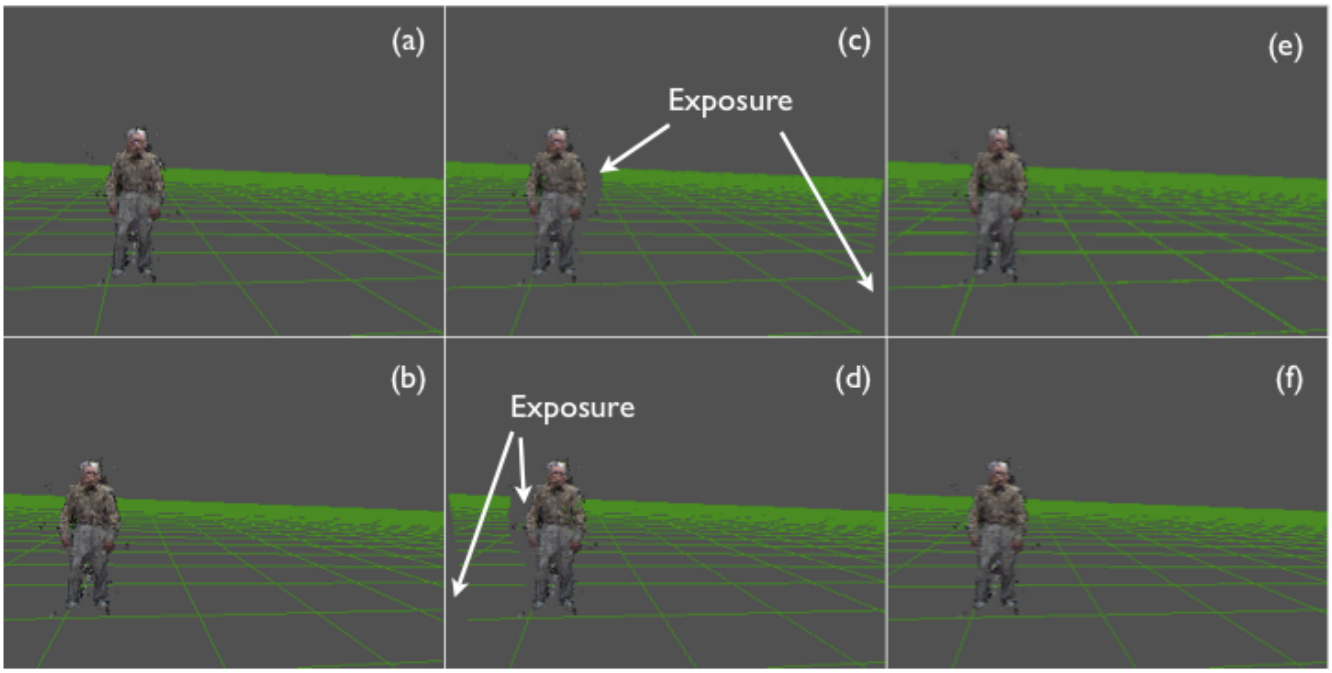

Figure 6: 3D Warping. (a)(b)Reference image frames from two different rendering viewpoints. (c)(d) Output images of running 3D Warping algorithm to (a) and (b). Warping errors can be clearly seen in output images due to exposure problems. (e) The composition of (c) and (d). Exposure gaps are eliminated. (f) Image directly rendered at the target rendering viewpoint

by camera parameters of the input and output image. They are computed only once for each image frame. As (u_2, v_2) is calculated, we just copy the color data of pixel (u_1, v_1) in the input image to the pixel (u_2, v_2) in the output image.

The 3D warping algorithm is very efficient. The warping equation requires only 20 arithmetic operations and each pixel of the image is computed only once in this algorithm. Therefore the complexity of the whole algorithm is proportional to the image resolution. The algorithm also handles the situation when multiple pixels in the input image correspond to the same pixel in the output image. McMillan also proved that the specific pixel processing order, which is also determined by camera parameters, can guarantee that any pixel that is hidden by other pixel in the output image will always be processed prior to its occluder. The 3D warping algorithm requires much less computation resources compared with the traditional graphics rendering pipeline.

Drawbacks always come with benefits. 3D warping requires the depth map of each image scene as well as the color map. Therefore, more network bandwidth is required to send image frames to mobile devices. Compression can be applied on depth images to reduce bandwidth consumption. Depth image compression have been studied from many aspects [18, 17, 12, 25, 10]. Of all the techniques, the general scheme proposed by [24] for real-time intra-stream compression can be directly used in our system. We will compare the bandwidth usage in Section 6.

Another critical deficiency of 3D warping is that warping errors are generated when the occluded scene objects become visible in the new rendering viewpoint. There is no pixel in the input image to reference when drawing the new image. It is also called exposure problem.

In our approach, two reference image frames are selected to overcome the exposure problem. 3D warping is processed for both frames. If a region of the scene is visible in any reference image, it can be correctly placed it in the output image (Figure 6). The idea of using multiple reference frames was first proposed by Mark *et al.* in [13]. Compared with other solutions including LDI [19], view compensation [7], depth filter [18], and so on, selecting two reference images is more suitable for our real-time 3D video rendering framework, although both transmission bandwidth and warping computation on mobile devices are doubled. The same depth compression technique can be applied for both reference images.

The proxy renders 3D video frames at two rendering viewpoints, generates two depth image frames, and sends both to mobile devices. On the mobile device, if the rendering viewpoint equals to either input viewpoints, the corresponding frame can be directly displayed. Otherwise, both depth images are warped to the actual rendering viewpoint and two output images are composed together. The composition algorithm described in [13] is used in our system. Two reference frames are sufficient to resolve most occlusion problems if appropriately selected. Thus, we discuss how to select appropriate reference frames in next subsection.

4.2 Reference Frame Selection

The selection of reference frames has significant influence on the output image quality. We take the *translation* mode as an example to explain our selection algorithms. Table 2 defines all symbols we are going to use. For example, V_i and V_j $(i < j)$ are the selected rendering viewpoints of two reference frames, and V_{act} is the actual rendering viewpoint

Table 2: Variables and Symbols

Name	Description
V_{curr}	Current rendering viewpoint. A set of three vectors $\{Pos_{curr}, Dir_{curr}, Up_{curr}\}$
T	The unit translation to the right. Let $n > 0$, $V_{curr} + n \cdot T$ means the viewpoint moves right n-unit distance and $V_{curr} - n \cdot T$ means the viewpoint moves left n-unit distance
V_i	$V_i = V_{curr} + i \cdot T$, $i \in Z$. V_j, V_{est}, and V_{act} all have the same definition.
V_{est}	Predicted rendering viewpoint when the frame is displayed on mobile devices.
V_{act}	Actual rendering viewpoint when the frame is displayed on mobile devices.
$<, \leq$	Define $V_i < V_j$ when $i < j$, $V_i \leq V_j$ when $i \leq j$
I_i	2D image exported from frame buffer after rendering 3D video at V_i
D_i	Depth map exported from *z-buffer* after rendering 3D video at V_i
F_i	Depth image frame for V_i. A set of $\{I_i, D_i\}$
W_j^i	Output image of warping F_i to a V_j
$W_k^{i,j}$	Output image of warping F_i and F_j to V_k
$\sum$	Sum of the color value of all pixels in a image frame.
err_j^i	Warping error caused by warping F_i to V_j. $err_j^i = \sum(I_j - W_j^i)$.
$err_k^{i,j}$	Warping error caused by warping F_i and F_j to V_k. $err_k^{i,j} = \sum(I_k - W_k^{i,j})$.
H	Threshold value decided by experiments.

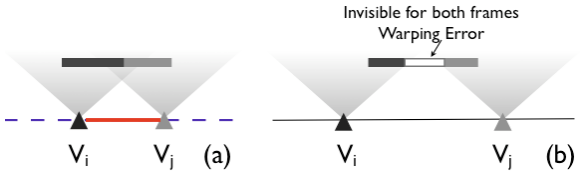

Figure 7: Reference Frame Selection

when the frame is displayed on the mobile device. If V_{act} drops into the solid line region in Figure 7(a), or formally expressed as:

$$V_i \leq V_{act} \leq V_j \tag{1}$$

where the exposure problem can be best recovered. Otherwise, if V_{act} drops into the dashed line region, the result is not better than using only one reference frame. Although the actual V_{act} is not known, we make an assumption that the proxy can make a good prediction of the actual rendering viewpoint V_{est}, which is close to V_{act} and has the following relationship.

$$V_{est} \leq V_{act} \tag{2}$$

We try to set two reference viewpoints far apart around V_{est} so that there is a high probability that V_{act} will fall in between. However, if reference viewpoints are too far away from each other, more warping errors are introduced and the warping quality of intermediate viewpoints is also degraded. Figure 7(b) elaborates one obvious scenario - some regions in the middle are missed by both reference frames. Therefore, the goal of reference frame selection is to find the two viewpoints far away from each other while the warping error is maintained at an acceptable level. It can be formally expressed as below.

$$\max_{i,j}\{j - i \mid V_i \leq V_{est} < V_j, \forall k \in (i,j), err_k^{i,j} < H\} \tag{3}$$

Based on Equation (1) and (2), V_{est} can be selected as the left reference viewpoint and the problem is simplified to find the best selection of the right reference. A *Full Search Algorithm* can be easily derived from Equation (3). The algorithm is described in Table 3. The time complexity of this *Full Search Algorithm* is $O(n)$ for frame rendering and $O(n^2)$ for 3D warping.

Obviously, the *Full Search Algorithm* is not efficient. For every pair of V_{est} and V_j, it is not necessary to calculate the warping error of all intermediate viewpoints, but find V_k which has the maximum warping error and check whether $err_k^{est,j}$ is larger than H. Intuitively, we can guess that the median viewpoint $V_{\lfloor \frac{est+j}{2} \rfloor}$ seems to be the best candidate for the maximum warping error viewpoint (Figure 7(b)). The following experiment is made to verify the guess.

Suppose that N different 3D video frames are selected. Given V_{est} for each frame, the *Full Search Algorithm* decides the reference viewpoints V_i and V_j. Then we find the maximum warping error $err_{max}^{i,j}$ of all intermediate viewpoints.

$$err_{max}^{i,j} = \max_k\{err_k^{i,j} \mid k \in (i,j)\}$$

A score P_k is given for each intermediate viewpoint where

$$P_k = \begin{cases} 1 & err_k^{i,j} = err_{max}^{i,j} \\ 0 & otherwise \end{cases}$$

Of all N different 3D video frames, the probability of each intermediate viewpoint V_k to have the maximum warping error can be described with a random variable $p(k)$ where

$$p(k) = \frac{\sum P_k}{N}$$

We normalize intermediate viewpoints of all frames by setting the median viewpoint $V_{median} = V_{\lfloor \frac{i+j}{2} \rfloor}$ to the origin point and draw the PDF function (Figure 8). We can verify our guess that V_{median} has the highest probability to generate maximum warping error.

The *Median Search Algorithm* is described in Table 4. For each pair of viewpoints, we only calculate the warping error of median viewpoint. Although there exist exceptions that median search does not give the best result, the algorithm successfully reduces the time complexity of 3D warping to $O(n)$. Moreover, the algorithm can be further optimized by increasing the searching step exponentially and setting

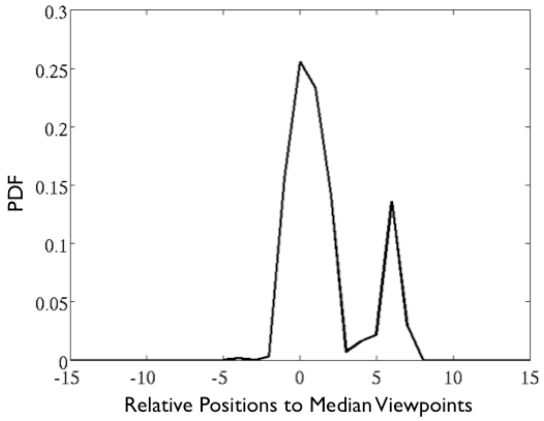

Figure 8: Probability of Maximum Warping Error

the initial search stride to a larger value. The *Fast Median Search Algorithm* (Table 5) implements these optimization methods and requires only $O(\log(n))$ time complexity for both frame rendering and 3D warping.

Table 3: Full Search Algorithm

$F_{est} = render\_video(V_{est})$
$F_{est+1} = render\_video(V_{est+1})$
FOR $stride = 2$ TO ∞
  $j = est + stride$
  $F_j = render\_video(V_j)$
  FOR $k = est + 1$ TO $j - 1$
    warping and calculate calculate $err_k^{est,j}$
    IF $err_k^{est,j} > H$
      RETURN V_{j-1} as the right reference viewpoint
    END
  END
END

Table 4: Median Search Algorithm

$F_{est} = render\_video(V_{est})$
$F_{est+1} = render\_video(V_{est+1})$
FOR $stride = 2$ TO ∞
  $j = est + stride$, $median = \lfloor \frac{est+j}{2} \rfloor$
  $F_j = render\_video(V_j)$
  warping and calculate $err_{median}^{est,j}$
  IF $err_{median}^{est,j} > H$
    RETURN V_{j-1} as the right reference viewpoint
  END
END

Although the proposed algorithms are derived based on the *translation* mode, they can also be used for other interaction modes with little modification. For example, for the *circling* mode, the viewpoint will be expressed as $V_i = R_{\theta_i} \cdot V_{curr}$, where R_{θ_i} is the rotation matrix for angle θ_i. Other analysis and derivation of reference frames selection are similarly achieved. The *rotation* mode is an exception because *rotation* can be directly achieved by 2D image rotation rather than 3D warping.

Table 5: Fast Median Search Algorithm

$F_{est} = render\_video(V_{est})$
$F_{\lfloor \frac{2est+init\_stride}{2} \rfloor} = render\_video(V_{\lfloor \frac{2est+init\_stride}{2} \rfloor})$
$stride = init\_stride$, $j' = est$
WHILE TRUE
  $j = j' + stride$, $median = \lfloor \frac{est+j}{2} \rfloor$
  $F_j = render\_video(V_j)$
  warping and calculate $err_{median}^{est,j}$
  IF $err_{median}^{est,j} < H$
    $j' = j$, $stride = 2 \times stride$
  ELSE
    $stride = \frac{stride}{2}$, BREAK
  END
END
WHILE $stride > 0$
  $j = est + stride$, $median = \lfloor \frac{est+j}{2} \rfloor$
  $F_{median} = render\_video(V_{median})$
  $F_j = render\_video(V_j)$
  warping and calculate $err_{median}^{est,j}$
  IF $err_{median}^{est,j} < H$
    $j' = j$, $stride = \frac{stride}{2}$
  ELSE
    $stride = \frac{stride}{2}$
  END
END
RETURN $V_{j'}$ as the right reference viewpoint

4.3 GPU Implementation

Another important feature of the proposed algorithm is that it can be easily decomposed into individual elements and executed in parallel. In this subsection, we briefly discuss using CUDA to implement our algorithms in graphical processing unit (GPU).

CUDA is the parallel computing architecture developed by NVIDIA, which manages computation on the GPU in a way that provides a simple interface for the programmer. In our implementation, the dimension of CUDA kernels (e.g., the number of thread blocks, grids, etc.) is determined based on the resolution of reference frames. Different memory models are utilized to efficiently manage data in processing. Typically, since the size of color and depth map rendered for 3D warping is larger than the shared memory (typically 16KB in a single thread block), they are stored in either shared or global memory which can be accessed by multiple threads in multiple grids. From the 3D warping equation presented in Section 4.1, each thread with partitioned pixels and shared depth information calculates a part of new coordinates (i.e., indices in matrices) of the warped image frame, and the full frame based on the new coordinate is merged together for further use.

Compared with executing reference frame selection algorithm on CPU, GPU computation has two important benefits. The first is the scalability as the frame sizes getting larger, hundreds of GPU cores provide far more computation resources than CPU to exploit the maximum computational parallelism on the given data. The second advantage is data transfer. Since we combine both frame rendering and 3D warping in our searching algorithms and the input frames for 3D warping is exported from the frame buffer and *z-buffer* of graphics pipeline. Thus the GPU implemen-

tation saves the large data transfer between CPU memory and GPU memory.

Inherently rendering is user-specific, which means that the rendering server has to render the 3D scenes from each user's perspective. Therefore, if there are n number of client devices accessing the same rendering server to watch the 3D video, the server would be loaded with n times of rendering processes. Even though there are work to extract 3D models from rendering process [4, 5] to reuse them, it is virtually impossible to process such an extraction in real-time due to complex model-view integration in 3D rendering process. Instead, utilizing more parallel rendering hardware via parallel or distributed rendering [2, 3, 6] has improved the performance of computer graphics, which would be the key to support large number of client devices and resolve the scalability issue in our remote rendering task.

Note that, although GPU provides a parallel environment that greatly accelerates the algorithm execution, it is not able to guarantee the real-time performance as the 3D video frame rate increases or the stream content becomes more complicated. Fortunately, the proposed search algorithms can stop at any time and also generate a suboptimal result, which means the selected reference frames are not far away from each other, but the warping errors of their intermediate viewpoints do not exceed the preset threshold. We define the *search range* as the maximum distance between selected reference frames that the algorithm can find without compromising any real-time requirements. For example, if there is only enough time to execute two frame rendering and one 3D warping. The *Fast Median Search Algorithm* can only verify one pair of candidates. Therefore the *search range* is *init_stride*. In Section 5, we will use this metric to evaluate three algorithms.

4.4 Motion Prediction

The assumption we have made that the proxy can predict the possible position of the actual rendering viewpoint is critical to the system performance. If V_{est} is far away from V_{act} or fails to meet Equation (2), then V_{act} may not fall between two reference viewpoints and more warping error can be introduced. However, the correct motion prediction is actually difficult to achieve. Although the user interaction design helps to restrict the motion prediction into limited modes and directions, human behaviors can never be perfectly predicted. The performance of motion prediction methods varies a lot for different interaction devices and different user groups.

In our research, we focus on the motion prediction for touch screen events. A pattern matching strategy is used for prediction. In the training process, the proxy records all user interaction information. Consecutive interaction signals are grouped into patterns with attributes such as interaction mode, duration, start point, end point, previous pattern, subsequent pattern, etc. Patterns are stored and maintained in a database for the online query. In the online execution process, the prediction is made for two different contexts. When there is a new interaction signal received by the proxy, it is matched with already known patterns stored in the database. The prediction is made based on the information of the most similar pattern. When there have been no interaction signal heard for a long time, a guess is made based on the number of pattern appearance. For example, in a preliminary user study, we have observed that

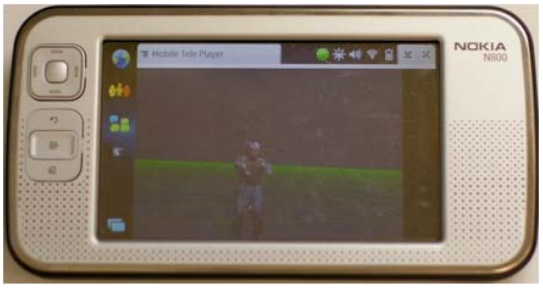

Figure 9: 3D Video on N800

of all the patterns we have collected in the *circling* mode, most patterns are *left/right circling*, and *up/down circling* seldom appears. Thus we can predict the next motion as left or right when the *circling* mode is selected.

4.5 Discussion

Although using three or more reference frames can cover more exposure gaps and further improve the warping quality, only two reference frames are selected on the proxy to overcome exposure problems. The bandwidth consumption is the major concern. Since the 3D video content is dynamically updated frame by frame, the reference selection and transmission is required for every video frame. Thus, the more reference frames are selected for image warping, the more bandwidth for transmission and computation for image decoding are required on mobile devices.

Another issue is the assumption that median viewpoint has the maximum warping error. In some scenarios, the maximum warping error does not necessarily appear on the median viewpoint due to the occlusion of 3D objects. Figure 8 also indicates the case - a small peak appears away from the origin point which is very probably caused by the occlusion. Since the occlusion is content dependent, we can add more checkpoints in our algorithms to solve the problem. A trade-off needs to be made between quality approximation and computation complexity.

5. EVALUATION

The system is evaluated from following aspects - rendering performance, bandwidth consumption, and rendering quality.

We select Nokia N800 (TI OMAP2420 330 MHz CPU and 128MB memory, 802.11g Wi-Fi, Linux based OS) as the experimenting mobile device and implement the 3D video viewer (Figure 9) to support 3D warping on the platform. Table 6 presents the performance of viewing 3D video on Nokia N800. The processing of each frame includes decoding two reference frames (two JPEG compressed color images and two ZIP compressed depth maps), running 3D warping algorithm twice for each reference and displaying the output image. The execution time is profiled on the real hardware. However, the number of frame rates shown in the table only indicates the worst case. When the actual rendering viewpoint equals to the predicted rendering viewpoint, no 3D warping is necessary and only one reference frame is decoded. In our experiments, when the viewpoint change is not frequent, we achieve the frame rate up to 10 fps for the resolution of 320×240 and more than 15 fps for the resolution of 176×144.

Table 6: Worst Case Performance of Nokia N800

Processing Time per frame (ms)	Resolution		
	176 × 144	320 × 240	640 × 480
Decoding	42	89	224
3D Warping	**30**	**71**	**191**
Display	5	13	31
Total	78	177	449
Frame Rate (fps)	12.8	5.6	2.2

Table 7: Input and Output Bandwidth (bps)

3D Video Streams	Input Bandwidth	Display Resolution	Output Bandwidth
1-site, 3 streams	4M	176 × 144	640K
1-site, 12 streams	16M	320 × 240	1.4M
2-site, 24 streams	32M	640 × 480	4.1M

Table 7 summarizes the required network bandwidth. The input bandwidth indicates the bandwidth of original 3D video streams which are transmitted from the gateway server to the proxy. The output bandwidth indicates the bandwidth of reference frames (two depth images per frame) transmitted from the proxy to mobile devices. Both streams are compressed with depth image compression techniques proposed in [25]. From the table, the input bandwidth is proportional to the number of 3D video streams generated by different 3D cameras, while the output bandwidth is only related with the display resolution of mobile devices.

The proxy for 3D video rendering and reference frame selection has an Intel Quad-Core Xeon processor running at 2.33GHz with 3GB memory and an NVIDIA GeForce 9800 GX2 GPU with 256 cores and 1GB of memory. 3D video streams are rendered with OpenGL and reference selection algorithms are implemented with CUDA SDK 2.1. A 3D video with 12 camera streams is transmitted to the proxy to test the performance. The frame rate of 3D video rendering is up to 100 frames per second and executing 3D warping takes less than 5 ms per frame. Table 8 compares the average *search range* of three algorithms. The *init_stride* is set to 8. Obviously, with the same computation resources, the *Fast Median Search Algorithm* generates the reference frames with the largest range .

Figure 10 evaluates the video quality displayed on mobile devices based on the average *search range*. The *x-coordinate* indicates the relative position of all possible rendering viewpoints to the median rendering viewpoint. The *y-coordinate* indicates the PSNR of the average picture quality if the video is actually displayed at the corresponding viewpoint on mobile devices. The standard image is chosen as the image directly rendered from 3D video. Images generated by warping reference frames are compared with the standard image to calculate PSNR. The red line is the threshold value set for reference frame selection algorithms. Three algorithms are represented by different lines. Two circle points on each line indicate the positions of two reference frames. The image quality of circle points is considered as standard so that circle points do not have PSNR value. If the actual rendering viewpoint is correctly predicted and falls into the line segment between two circle points, the display image is warped from two reference frames. Otherwise, only one reference frame can be helpful in 3D warping. In the figure, the lines

Table 8: Proxy Performance

Display Resolution	Frame Rate(fps)	Search Range		
		Full	Median	Fast
176 × 144	15	5	6	15
320 × 240	10	5	6	14
640 × 480	5	3	5	12

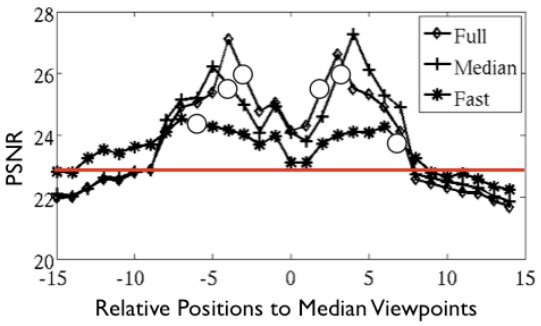

Figure 10: Rendering Quality

for *Full Search* and *Median Search* have high PSNR values in the center region. That is because these two algorithms have short *search range* and two reference frames are close to each other. However, if the actual rendering viewpoint is not correctly predicted, the rendering image quality drops quickly. On the contrary, the line for *Fast Median Search* remains relatively flat over all areas because the long distance between reference frames can guarantee the decent image quality of the largest range. Moreover, the fact that the median viewpoint ($x = 0$) has the lowest PSNR value in the line between circle points also verifies the assumption which both *Median Search Algorithm* and the *Fast Median Search Algorithm* depend on.

Actually, the absolute value of PSNR in this scenario does not reflect the quality of images. The accuracy limitation of depth values can cause the shifting of all image pixels after 3D warping. Figure 11 elaborates the problem. Although the image (b) generated by appropriately selected reference frames looks identical to the standard image, the difference image (c) indicates the invisible shifting grid lines, which results in the low PSNR value. However, if reference frames are not appropriately selected, more visible errors will be generated and the PSNR value is lower.

6. CONCLUSION AND FUTURE WORK

This paper makes the following contributions to real-time remote rendering of 3D video for mobile devices.

- We design and implement a proxy-based framework for real-time remote 3D video rendering on mobile devices without obvious interaction delay. Our framework features in the compatibility with different 3D video representations, the scalability to support multiple mobile viewers and the high performance in both frame rate and rendering quality.
- Reference frame selection algorithms are studied. The algorithm is designed for 3D video rendering. It can also be used for remote rendering of static graphic models.

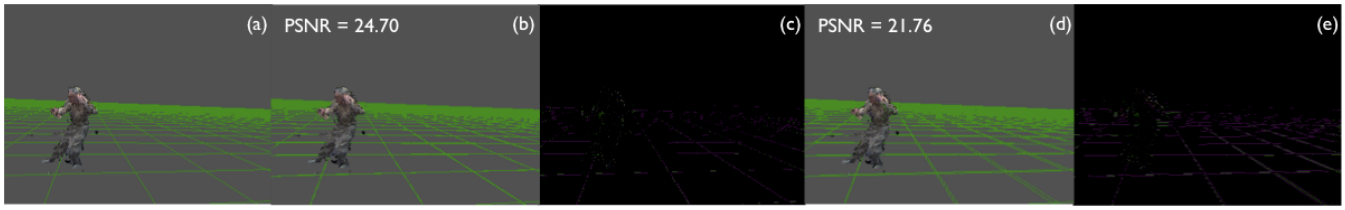

Figure 11: Quality Comparison of Result Images. (a) Standard image. (b) Result image warped from appropriately selected reference frames. (c) Difference image of (a) and (b). (d) Result image warped from inappropriately selected reference frames. (e) Difference image of (a) and (d).

- Parallel execution framework is used to implement computation intensive algorithms and greatly enhance the performance. It also indicates a future direction of combining mobile clients with powerful computing resources to provide services.

As for the future work, we will try to expand the current implementation to more mobile platforms. The graphics processing in mobile devices have been greatly enhanced. Devices such as iPhone, NVIDIA Tegra have already equipped powerful GPU chips inside. Therefore, reconstructing 3D meshes for those devices can provide a better performance than the current image-based approach. Besides, how to introduce mobile devices in 3D video acquisition is another interesting topic.

7. ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under Grant CNS 05-20182 and CNS 07-20702. Any opinions, finding 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.

8. REFERENCES

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

[2] Chromium. http://chromium.sourceforge.net/.

[3] Equalizer: Parallel rendering. http://www.equalizergraphics.com/.

[4] Glintercept. http://glintercept.nutty.org/.

[5] Gltrace. http://www.hawksoft.com/gltrace/.

[6] Sgi opengl multipipe sdk. http://www.sgi.com/products/software/multipipe/sdk/.

[7] P. Bao and D. Gourlay. Remote walkthrough over mobile networks using 3-D image warping and streaming. *Vision, Image and Signal Processing, IEE Proceedings*, 151(4):329–336, Aug. 2004.

[8] A. Boukerche and R. Pazzi. Remote rendering and streaming of progressive panoramas for mobile devices. In *Multimedia '06*, pages 691–694. ACM, 2006.

[9] C.-F. Chang and S.-H. Ger. Enhancing 3D graphics on mobile devices by image-based rendering. In *PCM '02*, pages 1105–1111, London, UK, 2002.

[10] S.-U. Kum and K. Mayer-Patel. Real-time multidepth stream compression. *TOMCCAP*, 1(2):128–150, 2005.

[11] F. Lamberti and A. Sanna. A streaming-based solution for remote visualization of 3D graphics on mobile devices. *IEEE Trans. Vis. Comput. Graph.*, 13(2):247–260, 2007.

[12] J.-M. Lien and et al. Skeleton-based data compression for multi-camera tele-immersion system. In *ISVC'07*, pages 714–723, November 2007.

[13] W. R. Mark and et al. Post-rendering 3D warping. In *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*, 1997.

[14] W. Matusik and H. Pfister. 3D TV: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. In *SIGGRAPH '04*, pages 814–824, 2004.

[15] L. McMillan. An image-based approach to three-dimensional computer graphics. In *Ph.D. Dissertation*. University of North Carolina at Chapel Hill, Department of Computer Science, 1997.

[16] L. McMillan and G. Bishop. Plenoptic modeling: an image-based rendering system. In *SIGGRAPH '95*, pages 39–46, 1995.

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

[18] A. Redert and et al. ATTEST: Advanced three-dimensional television system technologies. In *3DPVT '02*, pages 313–319, June 2002.

[19] J. Shade and et al. Layered depth images. In *SIGGRAPH '98*, pages 231–242, 1998.

[20] S. Shi and et al. View-dependent real-time 3D video compression for mobile devices. In *Multimedia '08*, pages 781–784. ACM, 2008.

[21] H.-Y. Shum, S. B. Kang, and S.-C. Chan. Survey of image-based representations and compression techniques. *IEEE Trans. Circuits Syst. Video Techn.*, 13(11):1020–1037, 2003.

[22] A. Smolic and et al. 3D video and free viewpoint video - technologies, applications and mpeg standards. In *ICME*, pages 2161–2164. IEEE, 2006.

[23] S. Würmlin and et al. 3D video recorder: a system for recording and playing free-viewpoint video. *Comput. Graph. Forum*, 22(2):181–194, 2003.

[24] Z. Yang and et al. TEEVE: The next generation architecture for tele-immersive environment. In *ISM '05*, pages 112–119, December 2005.

[25] Z. Yang and et al. Real-time 3D video compression for tele-immersive environments. In *MMCN '06*, 2006.

[26] Z. Yang and et al. Viewcast: view dissemination and management for multi-party 3D tele-immersive environments. In *Multimedia '07*. ACM, 2007.