DISTORTION OVER LATENCY: NOVEL METRIC FOR MEASURING INTERACTIVE PERFORMANCE IN REMOTE RENDERING SYSTEMS

Shu Shi, Klara Nahrstedt, Roy Campbell

Department of Computer Science
University of Illinois at Urbana-Champaign
201 N Goodwin Ave., Urbana, IL 61801, USA
{shushi2, klara, rhc}@illinois.edu

ABSTRACT

A new metric *distortion over latency* (DOL) is proposed in this paper to overcome the deficiency of the traditional metric *interaction latency* in measuring the interactive performance of the modern remote rendering systems, which are enhanced with different latency reduction techniques. The proposed metric is novel in combining both latency and rendering quality into one score for measurement. Our experiments validate that in many scenarios, our new metric can effectively distinguish the performance difference between systems while *interaction latency* can not. The paper also introduces how DOL can be efficiently calculated at runtime.

Index Terms— Remote Rendering, Interaction Latency, Interactive Performance

1. INTRODUCTION

Conventionally, the interactive performance of a rendering system is evaluated by *interaction latency* [4][10][13][6], which is defined as the time from the generation of user interaction request till the appearance of response image. However, the metric is not sufficient to evaluate the real interactive performance of recent network based remote rendering systems.

A good example is the server-to-mobile 3D remote rendering system that uses 3D image warping to reduce latency [7][11][12] (Figure 1). A workstation with enough computation and network bandwidth resources (e.g., cloud server) is served as the rendering server. It renders the source 3D contents and sends the rendering results to one or multiple mobile clients. The mobile client receives and displays the result images. Users can interactively change the rendering viewpoint on their mobile devices. The mobile client sends the interaction requests back to the rendering server, and the server will send back the scene at the updated rendering viewpoint. If the rendering server only sends 2D image to the mobile client, the system suffers from an *interaction latency* no less than a network roundtrip time (highlighted with the dash line). In order to reduce the latency, a post rendering module is added on the

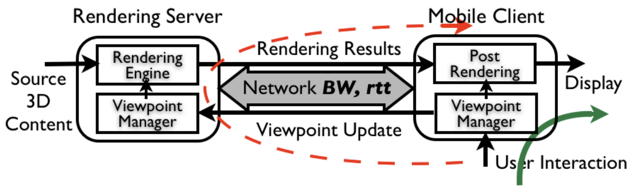

Fig. 1. Framework of the remote rendering system with post rendering latency reduction support

mobile client. The module runs 3D image warping algorithm [8] to synthesize the image for the updated viewpoint based on the received images that are rendered at old viewpoints, so that it allows the mobile client to respond immediately to any user interaction (highlighted with the solid line in the figure). The post rendering module significantly reduces the *interaction latency* at the cost of increasing network bandwidth usage (image warping requires the rendering server to send a depth map together with the color image to mobile clients) and degrading rendering quality (image warping introduces hole artifacts), so that *interaction latency* is no longer an effective performance metric. Instead, metrics such as *warping error* or *viewpoint range* have been used for performance evaluation in [7][12][11].

Another example is the remote rendering system that implements prefetch policies [5][1][2]. The client predicts the future viewpoint changes and uses the idle bandwidth to prefetch the scene images rendered at the predicted viewpoints even before any user interaction happens. The prefetch scheme can eliminate the *interaction latency* when the prediction hits. Therefore, the interactive performance of such prefetch based systems is more related to the prediction hit rate.

In this paper, we study how to measure and compare the interactive performance of remote rendering systems by considering not only *interaction latency* but also rendering quality. We propose a new interactive performance metric: *distortion over latency* (DOL) that integrates both latency and quality. We also introduce how remote rendering systems can be modified to calculate DOL at runtime. Real device experiments are designed to prove that DOL can be used for interactive performance evaluation even when *interaction latency* fails to. Our work makes two major contributions. First, we present that our proposed metric DOL is an effective method to measure and compare the interactive performance of both traditional rendering systems and the remote rendering systems with different latency reduction improvements. Second, we demonstrate that our runtime DOL calculation can be used for online interactive performance monitoring, and this real-time performance feedback scheme can contribute to better remote rendering system designs in the future.

For the rest of this paper, Section 2 summarizes the related works. Section 3 introduces our approach, including the design details of our new metric and runtime DOL calculation. Experiment setups and results are presented in Section 4. Related issues are discussed in Section 5. We conclude the paper and future work in Section 6.

2. RELATED WORK

Endo *et al.* suggested *interaction latency* as a better metric to evaluate interactive system performance than *throughput* [4]. Thereafter, researchers started to care more about latency, especially when designing remote rendering systems [10][13]. Salzmann studied combining multiple parameters as a metric for QoS in real-time interaction over internet [9]. Our previous work [11] has surveyed different remote rendering and remote visualization systems in the history. Most early systems did not pay much attention to *interaction latency*. Lamberti and Sanna [6] discussed in depth about different factors that affect *interaction latency* in remote rendering. Some projects [7][5][12] studied the issue of rendering quality. Our work is different from previous works in integrating quality together with latency as one metric to measure the interactive performance of remote rendering systems.

3. APPROACH

The new metric should provide two abilities:

- Measure both the latency of interaction events and the rendering quality on mobile clients.
- Integrate two measurements into one result, the value of which can be used to evaluate interactive performance.

In this paper, we define the new metric *distortion over latency* (DOL) as the product of time and MSE (mean square error). We will discuss how this metric can be improved in Section 5. In this section, we first introduce the definition of DOL and the scheme for runtime DOL calculation.

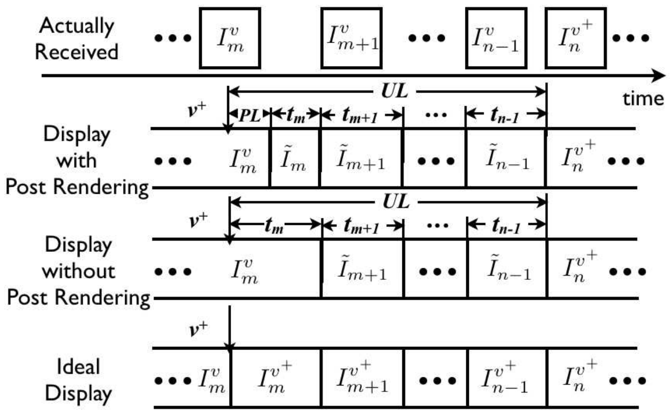

Fig. 2. Images frames displayed on the mobile client of a remote rendering system. Distortion over latency is the sum of difference between the actually displayed frames and the ideal frames during the interaction latency

3.1. Distortion Over Latency

We first define several notations and concepts. Figure 2 shows an illustration of image frames received and displayed on the mobile client in time sequence. I_x^v $(m \le x \le n-1)$ denotes the frame x which is rendered at viewpoint v on the rendering server. The user interaction that changes the viewpoint from v to v^+ happens right after the display of frame m. Frame n is the first frame that is rendered at v^+ on the rendering server. $\tilde{I}_x$ demotes the frames that are actually displayed on the mobile client. If the mobile client has a post rendering module, $\tilde{I}_x$ is the result frame of re-rendering I_x^v to the new viewpoint v^+. Otherwise, $\tilde{I}_x$ is identical to I_x^v. We define *frame interval* t_x $(m+1 \le x \le n-1)$ as the time interval that frame x stays on the screen. Note that the *frame interval* t_m is defined in a different manner. For a post-rendering system, t_m is the time interval that $\tilde{I}_m$ appears on the screen. For a non-post-rendering system, t_m is the time from the moment of user interaction till the appearance of $\tilde{I}_{m+1}$. *Post-rendering latency* PL is defined as the time from the moment of user interaction till the appearance of $\tilde{I}_m$. If the mobile client does not have any post rendering module, then $PL = 0$. *Update latency* UL is defined as the time from the moment of user interaction till the appearance $I_n^{v^+}$. It can also be expressed as:

$$UL = PL + \sum_{x=m}^{n-1} t_x \tag{1}$$

According to the traditional definition of *interaction latency*, PL is the *interaction latency* for post-rendering systems and UL is the *interaction latency* for non-post-rendering systems.

Rendering quality is determined by calculating the distortion between the actually displayed $\tilde{I}_x$ and the corresponding ideal frame $I_x^{v^+}$, which is rendered at the correct viewpoint. The ideal frame is introduced for distortion calculation only

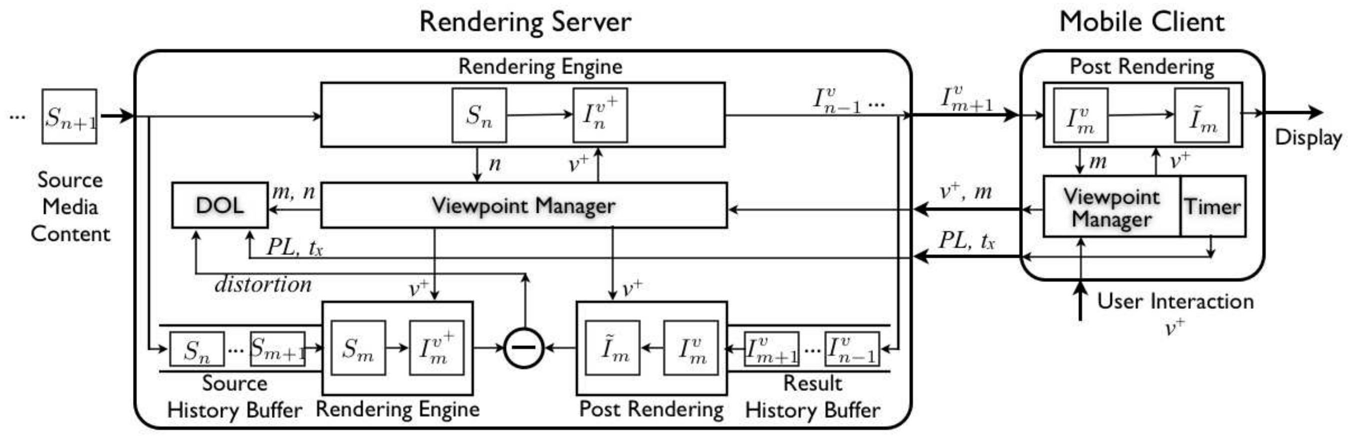

Fig. 3. Framework of the remote rendering system with online interactive performance monitor

and it is not actually displayed or generated. The distortion between two image frames is calculated by *mean square error* as follows:

$$D(\tilde{I}_x, I_x^{v^+}) = \frac{\sum_{x=1}^{h}\sum_{y=1}^{w}[\tilde{I}_x(x,y) - I_x^{v^+}(x,y)]^2}{w \cdot h} \tag{2}$$

where w and h are the width and height of the image, and $I(x, y)$ is the intensity value of pixel (x, y) in the image frame I.

We now define *distortion over latency* (DOL) as the total distortion of all image frames displayed during the *update latency* UL.

$$DOL = 10\log_{10}\frac{MAX_I^2}{PL \cdot D(I_m^v, I_m^{v^+}) + \sum_{x=m}^{n-1} t_x \cdot D(\tilde{I}_x, I_x^{v^+})} \tag{3}$$

where MAX_I is the maximum possible pixel value of the image. DOL is expressed in terms of the logarithmic decibel scale so that the unit is dB.

According to Eq 3, the DOL score is determined by both latency and the quality of displayed images. Either reducing the latency or increasing the post rendering quality leads to high DOL scores. Therefore, we believe that the higher DOL score represents the better interactive performance. Since DOL is normalized with image resolution, pixel value, and frame rate, the score can also be used to compare the interactive performance of rendering systems with different configurations.

3.2. Runtime DOL Calculation

DOL can be easily calculated offline by saving all system execution profiles. The difficulty of online calculation is that the mobile client does not have $I_x^{v^+}$ at runtime to calculate the distortion. In this subsection, we propose a light-weight scheme that simulates all behaviors of the mobile client on the rendering server and then calculate DOL at runtime. Figure 3 explains how components work and how data flows.

Client

Two modifications are made on the client side. The viewpoint manager needs to check for the current image frame number (m in Eq. 3) when user interaction happens. The frame number together with the new viewpoint v^+ are sent back to the rendering server. In addition, a timer component should be added to record the *post-rendering latency* PL and all *frame interval* t_x and send back to the rendering server. The timer starts on the arrival of the user interaction that changes v to v^+, records the elapsed ticks every time when the mobile screen is updated, and stops when the first frame $I_n^{v^+}$ is displayed. However, in our implementation we notice that the v^+ (a set of float vectors to indicate camera position, camera direction and camera up direction) on the mobile client can be slightly different from the v^+ processed on the rendering server due to the difference of floating point precision, and the difference can keep the timer running forever. In order to resolve this problem, we introduce the concept of *viewpoint age*. A 32-bit unsigned integer is assigned as the "age" to every viewpoint. The *viewpoint age* increases upon user interaction. Every image frame inherits the *viewpoint age* from its rendering viewpoint. Therefore, the timer module compares only the integer *viewpoint age* of the current mobile viewpoint and the received frame rather than the float vectors.

Server

Two history buffers are added on the rendering server to store both source content frames and rendering results frames. The buffer size is determined by the maximum possible *update latency* and the rendering frame rate. For example, if rendering frame rate is 10 fps and the maximum possible *update latency* is 2 seconds, the buffer should store no less than 20 frames. Two buffers are organized as FIFO queues. The source history buffer is connected to a rendering engine and the result history buffer is connected to a post rendering module identical to the one on the mobile client (if the mobile client does not have post rendering support, the result history buffer outputs the saved frames directly). Under the control of viewpoint manager, the source history buffer can generate the history frame rendered at the correct viewpoint ($I_x^{v^+}$) and the result history buffer can generate the history frame exactly displayed on the mobile client ($\tilde{I}_x$). The distortion of these two frames are calculated and sent to the DOL module. The viewpoint manager initiates the DOL calculation when the new viewpoint information and the frame number (m in Eq. 3) arrive. Meanwhile, it checks the current frame number (n in Eq. 3) in the rendering engine because it is the first frame rendered at the updated new viewpoint. Both frame numbers (m, n) together with PL and t_x are passed to the DOL module to calculate the DOL score at runtime.

Our runtime DOL calculation scheme is light-weight for the mobile client and network usage. Neither timer recording nor the network transmission of extra data can cause noticeable performance degradation. The added components and functions on the rendering server may double the computation workload. However, given the assumption that the rendering server has abundant computing resources, our scheme should not impair the rendering performance of the server. In the worst case, all DOL calculation related components can be deployed in an individual workstation so that the server for rendering is not affected.

4. EVALUATION

In this section, we demonstrate that DOL, although looks simple, actually works well. Our experiments try to prove that given two rendering systems, the one with higher DOL score has better interactive performance. We set up a remote rendering system with different configurations which result in different interactive performance, run experiments for all configurations, and compare the collected DOL scores with the actual performance.

Experiments are running on our self-implemented remote rendering system prototype for the previous work [11]. A 3D image warping module has been implemented to provide post rendering support. The prototype can work in three different rendering modes:

- *Image Streaming*: the rendering server sends only the 2D image of the rendering result. The client simply displays the received image on the screen. The interactive performance of this rendering mode degrades as the network delay increases.

- *Single Warping*: the rendering server sends one depth image (including color map and depth map) per frame. The client displays the color map if the received frame is rendered at the correct viewpoint. Otherwise, the client warps the depth image to the correct viewpoint using 3D warping algorithm [8]. This rendering mode uses the synthesized images which have warping holes (Figure 4) to compensate the interaction latency. It is expected to have better interactive performance than *image streaming* when using the same network.

- *Double Warping*: the rendering server generates two carefully selected depth images per frame based on the reference selection algorithm proposed in [11]. The client warps both depth images to the current rendering viewpoint if neither can be displayed directly. This rendering mode can generate the warping results with less holes (Figure 4) and should have the best interactive performance of all three modes when using the same network.

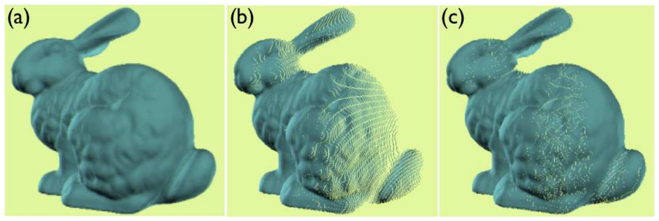

Fig. 4. (a) Result image of 3D rendering; (b) Result image of warping from one depth image; (c) Result image of warping from two carefully selected depth images.

Given the same network condition, the interactive performance the three rendering modes can be ranked as: *Double warping* $>$ *Single warping* $>$ *Image streaming*[1].

The rendering server runs on a workstation that has an AMD Phenom II 3.2GHz quad-core CPU, 4GB memory, an Nvidia GeForce 9800GT GPU, and connects to the Gbps ethernet university network. The mobile client has been implemented to run on Apple iOS platforms. Three different setups are selected to run all experiments:

- *Simulator*: the mobile client runs on an iPhone simulator hosted by an Apple MacBook laptop. The host machine has a dual-core CPU, 2GB memory, and connects to the university network through 802.11n Wi-Fi.

- *Wi-Fi*: the mobile client runs on an Apple iPhone 4 smartphone. The phone connects to the university network through 802.11g Wi-Fi.

- *3G*: the mobile client runs on an Apple iPhone 4 smartphone. The phone connects to Internet through 3G (HSDPA/UMTS) network operated by AT&T.

[1] $A > B$ means A leads to better interactive performance than B

Table 1. SpeedTest Results of Three Network Setups

	Simulator	*Wi-Fi*	*3G*
Ping (ms)	8	43	248
Download (Mbps)	61.75	9.91	2.02
Upload (Mbps)	7.85	7.73	0.71

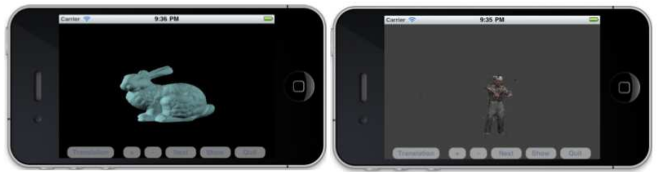

Fig. 5. Remote rendering *bunny* (left, static 3D model) and *taichi* (right, dynamic 3D video) on iPhone.

Table 1 shows the SpeedTest[2] results of the network used in three setups above. *Ping* denotes the network roundtrip time between server and client. *Download* and *upload* characterize the practical network bandwidth available for communication. The network with short roundtrip time and high bandwidth is expected to deliver better interactive performance than the network with long roundtrip time and low bandwidth. Thus, we can rank these setups as: *Simulator* >*Wi-Fi* > *3G*.

We also select two different rendering sources: *bunny* and *taichi* to run experiments (Figure 5). *Bunny* is a static 3D model constructed by 69,450 triangles. *Taichi* is a 3D video reconstructed from depth images captured from 12 different 3D cameras. Both source contents are rendered on the rendering server at a fixed resolution of 480×320. In order to save transmission bandwidth, color images are compressed with JPEG and depth images are compressed with ZLIB on the rendering server. TCP protocol is used to guarantee the reliable transmission. The static model *bunny* is either re-rendered every second or updated when any user interaction changes the rendering viewpoint. The 3D video *taichi* is rendered at a constant frame rate of 8 fps with one exception that when the rendering server is on *double warping* mode and the mobile client connects through *3G*, only 5 fps can be achieved due to the limited bandwidth. Table 2 lists the actual network bandwidth used in our experiments for different configurations.

Table 2. Network Bandwidth (Kbps)

	Image Stream.	*Single Warp.*	*Double Warp.*
Bunny	56	496	992
Taichi	320	1024	2048

For each run of our experiment, we send the same group of user interaction requests to the mobile client with a fixed interval of 10 seconds and last for 10 minutes. The user interaction either translates or rotates the rendering viewpoint. The interaction is controlled by script so that the same request is always sent at the same time for every experiment run.

[2]http://www.speedtest.net

We present the data collected in our experiments in Table 3. There are three numbers in each table cell: average *interaction latency* (before slash), the average *update latency* (after slash), and average DOL score (second line). We list *interaction latency* in the table to compare with our proposed metric DOL. As we have discussed in Section 3.1, *interaction latency* equals to UL for image streaming mode and equals to RL for other two warping modes.

From the data in the table, we can find the *interaction latency* works well for the image streaming mode. According to the *interaction latency* numbers in the first row, *Simulator* >*Wi-Fi* > *3G* (the shorter latency means better performance). However, it fails to work for two warping modes. The *interaction latency* numbers in last two rows lead to incorrect ranks: *3G* > *Wi-Fi*, *Single warping* > *Double warping*.

Our proposed DOL does a much better job in ranking all cases. Comparing the DOL scores in every row and column, we can easily conclude that *Simulator* >*Wi-Fi* > *3G*, and *Double warping* > *Single warping* > *Image streaming*. These conclusions perfectly match our experiment expectations.

5. DISCUSSION

Although we only used the post-rendering system for explanations and experiments in the previous sections, DOL also works for the rendering system with prefetch support. When the motion prediction fails, the system acts exactly the same as "Display without Post Rendering" shown in Figure 2, and we reuse Eq 3 to calculate DOL. When the motion prediction hits, the DOL calculation can be simplified as

$$DOL = 10 \log_{10} \frac{MAX_I^2}{DL \cdot D(I_m^v, I_m^{v^+})}$$

where DL is the time interval for the mobile client to display the prefetched frame $I_m^{v^+}$. Since DL is much smaller than UL in Figure 2, the DOL score when prediction hits is much higher than the score when prediction fails. Therefore, the average DOL score is determined by prefetch prediction rate and can be used to evaluate the performance of prefetch enhanced rendering systems.

Although we have demonstrated that DOL outperforms *interaction latency*, there are many places we can improve the metric in the future. First, MSE is not the best metric for image quality evaluation. We can test more image assessment tools, like SSIM, for the metric. Second, the human reaction to the *interaction latency* is not linear. For example, the system user is not sensitive to the latency if it is less than a threshold [3]. The next version of DOL should take this fact into consideration. Third, the current study of DOL is still limited in using the metric to measure which system has the better interactive performance. However, a complete

Table 3. *Interaction Latency*, *Update Latency*, and *Distortion Over Latency*

	Bunny			*Taichi*		
	Simulator	*Wi-Fi*	*3G*	*Simulator*	*Wi-Fi*	*3G*
Image Streaming	40ms / 40ms 37.91dB	255ms / 255ms 29.84dB	449ms / 449ms 27.39dB	285ms / 285ms 35.65dB	474ms / 474ms 33.44dB	610ms / 610ms 32.34dB
Single Warping	17ms / 70ms 38.47dB	161ms / 367ms 30.25dB	144ms / 847ms 28.04dB	16ms / 256ms 42.13dB	156ms / 521ms 36.60dB	143ms / 746ms 35.98dB
Double Warping	18ms / 122ms 40.00dB	177ms / 377ms 31.12dB	161ms / 1038ms 30.52dB	16ms / 274ms 43.37dB	154ms / 536ms 37.15dB	146ms / 931ms 36.33dB

evaluation also requires some quantitative studies to correlate the numbers in DOL score with the difference of interactive performance that human can perceive. Thus, more subjective tests should be carried out to better understand the connections between DOL scores and user satisfaction.

6. CONCLUSION

In this paper, we have proposed a new metric DOL, a weighted sum of all frame distortions during the whole *update latency*, to measure the interactive performance of rendering systems. The new metric can better evaluate the interactive performance of rendering systems, especially the remote rendering systems with different latency reduction improvements, because it successfully combines both latency and rendering quality in one score. We have designed several real-device experiments to prove the argument. In addition, we also introduced a light-weight scheme to calculate DOL at runtime, which can potentially play an important role in developing future remote rendering systems.

7. ACKNOWLEDGEMENTS

This research has been supported by the National Science Foundation under Grant CNS 05-20182 and CNS 07-20702.

8. REFERENCES

[1] Addison Chan, Rynson W. H. Lau, and Beatrice Ng, "A hybrid motion prediction method for caching and prefetching in distributed virtual environments," in *VRST*, 2001, pp. 135–142.

[2] Jerry Chen, Ilmi Yoon, and Wes Bethel, "Interactive, internet delivery of visualization via structured prerendered multiresolution imagery," *IEEE Trans. Vis. Comput. Graph.*, vol. 14, no. 2, pp. 302–312, 2008.

[3] Mark Claypool and Kajal Claypool, "Latency can kill: precision and deadline in online games," in *MMSys '10: Proceedings of the first annual ACM SIGMM conference on Multimedia systems*, New York, NY, USA, 2010, pp. 215–222, ACM.

[4] Yasuhiro Endo, Zheng Wang, J. Bradley Chen, and Margo I. Seltzer, "Using latency to evaluate interactive system performance," in *OSDI*, 1996, pp. 185–199.

[5] Gerd Hesina and Dieter Schmalstieg, "A network architecture for remote rendering," in *DIS-RT*. 1998, pp. 88–91, IEEE Computer Society.

[6] Fabrizio Lamberti and Andrea Sanna, "A streaming-based solution for remote visualization of 3D graphics on mobile devices," *IEEE Trans. Vis. Comput. Graph.*, vol. 13, no. 2, pp. 247–260, 2007.

[7] William R. Mark, "Post-rendering 3d image warping: Visibility, reconstruction, and performance for depth-image warping," in *Ph.D. Dissertation*. 1999, University of North Carolina at Chapel Hill, Department of Computer Science.

[8] Leonard McMillan and Gary Bishop, "Plenoptic modeling: an image-based rendering system," in *SIGGRAPH '95*, 1995, pp. 39–46.

[9] Christophe Salzmann, "Real-time interaction over the internet," in *Ph.D. Dissertation*. 2005, Ecole Polytechnique Federale de Lausanne.

[10] Brian K. Schmidt, Monica S. Lam, and J. Duane Northcutt, "The interactive performance of slim: a stateless, thin-client architecture," in *SOSP*, 1999, pp. 32–47.

[11] Shu Shi, Mahsa Kamali, Klara Nahrstedt, John C. Hart, and Roy H. Campbell, "A high-quality low-delay remote rendering system for 3d video," in *ACM Multimedia*, Alberto Del Bimbo, Shih-Fu Chang, and Arnold W. M. Smeulders, Eds. 2010, pp. 601–610, ACM.

[12] Ferdi A. Smit, Robert van Liere, Stephan Beck, and Bernd Fröhlich, "An image-warping architecture for vr: Low latency versus image quality," in *VR*. 2009, pp. 27–34, IEEE.

[13] S. Jae Yang, Jason Nieh, Matt Selsky, and Nikhil Tiwari, "The performance of remote display mechanisms for thin-client computing," in *USENIX Annual Technical Conference, General Track*, Carla Schlatter Ellis, Ed. 2002, pp. 131–146, USENIX.