Remote Rendering for Mobile Devices Literature Overview

Chanchan Xu, Guangzheng Fei(✉), and Honglei Han

School of Animation and Digital Arts,
Communication University of China, Beijing, China
gzfei@cuc.edu.cn

Abstract. Mobile device such as mobile phone, PDA (Personal Digital Assistants), HPC (Handheld Personal Computer) and so on has become a prevalent commodity and also a significant influence power that dominates people's daily life. Remote rendering over these platforms is a continuous research subject that still attracts many people's attention. It is also a promising topic for its extensive usage in applications for mobile devices. It is still a challenging issue for the limitation of the wireless network and the process ability of the client side. Here we introduce the state of the art remote rendering techniques over mobile devices and analyze them in order to get a clear perception and a better understanding of this topic.

Keywords: Remote rendering · Mobile and personal devices · Compression

1 Introduction

Due to the light-weight, handheld-size, portability and availability to the internet, mobile device such as mobile phone, PDA (Personal Digital Assistants), HPC (Handheld Personal Computer) and so on has become a prevalent commodity and also a significant influence power that dominates people's daily life. More applications are ported to it and more powerful hardware devices are developed for these applications. This brings not only entertainment, excitement and convenience to people, but also requirement for more complex applications such as 3D games which perform the same quality as the video games on the computer.

Despite the growing process ability of the CPU, it is still unfeasible for 3D graphic applications to run directly on those lightweight platforms. First, their CPU is not so powerful enough to handle large sums of data. Second, they lack the specific graphics hardware for GPU-based solutions to accelerate the computation process. Third, their limited memory and lower resolution restrict the running programs and the storable data size. Moreover, their battery usually can't last for long. The exponential growth in data storage capacity and collection of applications such as 3D digital museum display, large terrain navigation and so on pushes the exploiting of the hardware capabilities to the edge.

Remote rendering is a common and also effective solution to tackle this issue. The idea is early introduced by Schmalstieg [1]. The application runs on a dedicated workstation (called the server-side or server) with sufficient computation ability and

A. El Rhalibi et al. (Eds.): Edutainment 2016, LNCS 9654, pp. 173–181, 2016.
DOI: 10.1007/978-3-319-40259-8_15

network resources. The workstation bears most of the computation burden while only a little computation work needs to be done on a not so powerful device (called the client-side or client). Communication between the server and the client goes through the internet, transferring commands of the user or results of images, video, geometry data or other media data. In this way, the client can ease lots of workloads and can also display relatively high quality results, which makes 3D graphics application rendering viable for lower-end computers or mobile platforms. This stimulated the research on remote walkthrough, remote visualization, 3D cloud games and so on lower-end devices such as mobile devices.

However, the remote rendering also confronts with many challenging issues, such as response latency during an interactive operation, compression for decreasing the transmitted data amount, rendering quality at multiple resolutions of end devices, and finally, workload distribution between the server and the client.

We survey the state of the art research techniques related to remote rendering, especially on mobile devices, summarize and analyze them in order to get a clear perception and a better understanding for further study. We arrange the rest of our paper as follows: first we list state of the art techniques, classify them according to the rendering side and analyze the quality of different categories (Sect. 2). Then the main issues involved in remote rendering are discussed in Sect. 3. In the end, we discuss and conclude the whole article in Sect. 4.

2 Categories of the Remote Rendering

Considering the rendering side that happens, the remote rendering can be divided into client-side rendering, server-side rendering and hybrid rendering.

2.1 Client-Side Rendering

Rendering on the client side demands an adequate computation capability for the hardware. Models of the scene need to be downloaded from the server and rendered on the client. The aim is to keep the graphics which the client should render as simple as possible while the client still can achieve satisfactory performance. Usually the transmitted data are geometry information, including the meshes, triangles, indices, textures and so on. That causes a high occupation of the bandwidth and a heavy render burden for the client.

A number of approaches can be identified to tackle these problems. The classical method of culling, including view-frustum culling, occlusion culling and back-face culling can skip the rendering of unsighted geometry. A LOD (Level of Detail) method can greatly simplify the complex structure of the graphics. Lluch et al. [2] designs a multi-resolution method for diverse resolution of handheld devices by using a view-dependent LOD strategy. For further decreasing the data that should be transmitted, sending line primitives which represent the outline of the graphics [3] or point clouds [4] for non-photorealistic rendering (NPR) will do help.

2.2 Server-Side Rendering

For server-side rendering methods, an ad hoc computer, i.e., the server, which possesses sufficient resources, both on computation ability and network, takes charge of the graphics rendering and interaction with the clients. The rendering results which formed in 2D images, videos and so on are sent to the client through the internet. Usually, the server side uses cluster PCs or graphics accelerators to enhance the performance. The aim of this approach is to make the 2D raster image get their best performance in a 3D interactive virtual scene. This can be achieved by decreasing the frequency of transmitting images [5, 6] and decreasing the size of the images or videos which have to be transmitted [7, 8].

Demand driven is a common used strategy to reduce the transmission times. Only when a demand is sent to the server for updates, will new frames be rendered for the client. The use of image warping can generate new frames from some reference images, which can also decrease the frequency of sending. Shi et al. [5] employs a reference prediction algorithm to find the viable reference images which also covers the largest area according to some predefined motion patterns. Paper [9] compares the quality of three main image warping methods: point splat image warping, quad splat image warping and mesh-based image warping.

As for cutting down the images' or videos' size, compression and video coding method need to be used before the transmission [10]. The server-side rendering does a great job in remote walkthrough applications [6] and medical uses [11].

2.3 Hybrid Rendering

The hybrid rendering approaches combine both the computation resources of client-side and server-side, aiming to limit the transferred data amount and enhancing the rendering quality of the client. It is very suitable for digital cultural heritage models display [12, 13] and large terrain navigation [14].

In order to protect the privacy of the original cultural heritage models from misuse yet still provide the viewers a proper interaction for navigation, the server sends sparse 3D mesh patches [13], simplified models [15] or even point models [12] to the client for interaction. When the interaction is over, the high-resolution map can be transmitted to the client for display. For large terrain navigation, paper [14] divides the scene into two parts: the one closes to the users' position and the one far away from the user's position. The closer part is rendered on the client with chunks which delivered from the server, while the far-off part is rendered on the server as impostor for the background of the client. The efficiency relies heavily on both sides and workload distribution algorithms need to be used to improve it. For more details of hybrid rendering, search paper [16].

2.4 Analysis of the Three Methods

Below we analyze the three kinds of remote rendering in Table 1.

Table 1. Analysis of client-side/server-side/hybrid rendering

	Client-side rendering	Server-side rendering	Hybrid rendering
Render side	C	S	C & S
Data transmission	**Geometry data** Partial model Simplified model Line primitives Point cloud	**Images or video** Image for mapping Image for impostor Video for interaction	**Images and geometry data** Images for reference Geometry for interaction
Aim	Keep graphics simple enough Satisfactory performance	Make 2D image fit 3D scene Satisfactory performance	Limit transferred data amount Enhance the rendering quality
OBW[a]	High	Low	Medium
Limitations	High occupation of BW Hard-/software compatibility Limited resource on the client Risk of model data leaking	Unnatural interaction Artifacts on image mapping Artifacts on image warping Threshold of updating	Hard-/software compatibility Limited resource on the client
Improve approach	LOD method Line extraction Prefetching and prediction	Image warping Video compression Video coding	Image warping Workload distribution Video compression and coding
Application	**Small graphics and interaction** NPR Multi-resolution rendering	**Static or almost static scene** Large virtual city navigation Remote walkthrough Medical display	**Scene with simple interaction** Remote walkthrough Large terrain visualization Heritage model display

[a] OBW is short for Occupied Bandwidth

The client-side rendering methods are mainly limited by the capability of the client hardware. The high occupation of the bandwidth further slows down the download and interaction. These approaches are suitable for those in which the generation of 3D meshes takes a far more computational amount than the rendering process. Improvement methods focus on minimizing the size of the transferred data which is rendered on the client side: LOD methods, Culling methods, Line rendering methods, Point rendering methods and so on. The server-side rendering approaches ease the rendering burden of the client side and lower down the bandwidth occupation so that the system can run more complicated applications without considering the hardware capacity and the hardware/software compatibility on the client-side. Image mapping and image warping are common methods in server-side rendering techniques. However, this comes at the expense of trade-offs, which may cause unnatural user experience in the interaction and artifacts in the display. Compensation is a crucial method in solving the

artifacts. For more fluent interaction, a threshold of updating frames and compression methods should also be used. The hybrid rendering methods inherit both the limits of the former two types: the limitation of the client resources; the high occupation of the internet; the artifacts of the display during interaction. Model simplification methods, image warping with depth images, video compression can greatly improve the performance of the system.

3 Main Issues Involved in Remote Rendering

In this section, we summarize some of the main issues involved in remote rendering approaches, and list them below for detailed illustrations.

3.1 Compression for Transmission

The size of data which transmitted through the internet, and further processed by the client is crucial for the performance of the whole remote rendering pipeline. Limited by the capacity of the client devices and the bandwidth of the network, a good compression method is needed to decimate the volume of the data. According to the data type which should be compressed, we divide the compression method into geometry compression and image compression.

For cutting down the size of the geometry, surfaces decimation algorithms and view-dependent strategies can help a lot. LOD methods are effective ways in getting simplified models. Partial models lying in an interesting area/region can be extracted and lower down the bandwidth occupation [14] according to the users' position. What's more, the reconstruction of the whole scene with artistic representation like feature lines or point clouds can reduce the volume of the data while still maintaining the virtual environment's nature.

As for image compression, main stream codecs lead the way. The JPEG compression method can greatly decrease the size of the image (see 0.8 % ~ 2.1 % of the original BMP image [12]). With interaction, the server will send a short video to the client for display. Paper [10] proposes an adaptation algorithm to optimize the video encoding quality with an ROI based partitioning according to the depth map of the original image. Paper [17] also applies the ROI method in video encoding. In paper [18], a wavelet based PTC (Progressive Transform Coder) codec is used to compress the residue of the tiles. Paper [19] uses a warping-based motion estimation method in augmenting compression of the video on the server side with the depth information of the external and internal camera parameters.

3.2 Quality in Display

The quality of performance on the client mainly relies on the capacity of the hardware and the resolution of the models or images. The representation of geometry models is independent of the client's resolution. The qualities mainly depend on the resolution of the model itself, i.e., the LOD. Multi-resolution of the geometry which represents

different densities of geometry cells and also different approximations of the shape can be achieved by the LOD method and used for multiple client devices [2].

However, the quality of image-based rendering method depends heavily on the resolution of the client [7]. Extra information of images with high-resolution will be ignored by the client with relatively low resolution on the screen. Paper [20] uses two types of streaming according to the target resolution of the end devices: graphics commands for high resolution devices and graphical output for relatively low resolution devices.

3.3 Latency in Interaction

Dealing with latency in interaction is an inevitable problem with the graphics applications with which the user interact frequently. It is defined as "the time from the generation of user interaction request till the appearance of the first updated frame on the mobile client" [5]. The interaction latency has a great influence on the user's experience, especially in games. Due to [21], the tolerable latency limit in a first person shooting game is 100 ms, but most of the remote rendering systems possess more than that: 700+, 300+, and 200+ ms latencies in GPRS, EDGE, and UMTS cellular networks [22]. Ways to reduce the interaction latency can be image warping, compression of the data, protocols for transmission and prefetching strategies.

Image warping methods can generate new view frames from some reference images and depth images. An essential issue of image warping is minimizing the frequency and amount of the reference images. Paper [5] proposes a novel algorithm to find the most proper references with which the view positions of warped images cover the largest area. Image warping method are at the price of trade-offs. Holes or sampling artifacts can have great impacts on the image. Approaches have come out to solve the occlusion exposure and insufficient sampling. Paper [2] reduces the latency by only sending geometry updates to the client with a synchronization process to keep the geometry coherence of the server and the client. Video compression codecs used for real-time display can also be helpful in decrease the sending time of the data.

A prefetching method is very useful in the interaction of walkthrough [18, 23]. With the Grid-Lumigraph method, the arbitrary views can be reconstructed [13], still with the help of sampling images.

3.4 Workload Distribution and Acceleration

The efficiency of remote rendering relays heavily on the both side devices' capacities. An efficacious splitting of workload will do a lot help in improving the performance of the remote rendering system. This includes a splitting of the rendering burden and an arrangement of the rendering tasks on the multiple rendering nodes or clusters.

Paper [14] splits the workload between the client and the server with scenes in close up views rendered on the client and scenes in medium or far range in sight rendered on the server. For serving multi-client synchronously, Neven [11] employs an algorithm

based on the current frames' noise and choose the proper render node according to a probability density function.

As for the acceleration method, GPU based acceleration methods are normally used for hardware acceleration, such as the Doellner's Render Worker [6]. Paper [24] uses a parallel rendering strategy for real time rendering.

3.5 Level of Detail (LOD)

The aim of LOD technique is to decrease the complexity of the geometry, and then further reduce the size of data which should be computed. The kernel of LOD is how to create the LOD, how to select the proper LOD and how to transit between two LODs. A LOD model can be generated through a structure of Octree. Different levels of primitives represent different levels of details. Edge collapse operation can create a continuous change in geometry's resolution. A simply way to select the proper LOD is according to the distance of the viewer's position or the projected area on the screen. Lluch et al. [2] proposes a method to efficiently compute the difference between two given level of details.

Strictly speaking, the LOD method is not the main issues in remote rendering, but it is an important strategy for data compression [7, 18], progressive rendering [25], adaptively display in resolution for the client [2]. In paper [18], the LOD method is used with a tile-based division and stitching in the terrain rendering, and successfully achieved a real-time frame rate (about 20–40 fps for low end client with low bandwidth). Quillet et al. [7] uses a line-based rendering with LOD and reduces the data size as about 2 Mb smaller than the method with textures in storage. In paper [25], a smooth level of detail selection is provided for progressive rendering.

4 Conclusion

For a better understanding and a clear conception of remote rendering, we illustrate the state of the art techniques, especially those over mobile devices. We then divide these techniques into several groups according to the rendering side that happens, and then introduce them with further analysis. Moreover, we list some main issues which involved in remote rendering, including data compression for quicker transmission, adaptable display quality for multiple end devices, lower interaction latency of instant response, and workload distribution for more efficient performance of the remote rendering system.

With the prevalent usage of mobile devices, demands to display more complex graphics on these platforms will become more urgent. However, the insufficiency of these platforms' hardware and the wireless network resources will still remain unchanged for quite a long time. Regardless of the hardware updates, future research works will focus on the reducing latency method of interaction, the simplification method of graphical models, compression method for more effective compression rates, workload balance between the client side and server side for more sufficient use of the hardware in both sides.

Acknowledgments. This work is supported by the National Key Technology R&D Program under Grant No. 2012BAH62F02.

References

1. Schmalstieg D.: The remote rendering pipeline: managing geometry and bandwidth in distributed virtual environments. Ph.D. Vienna University of Technology, Vienna (1997)
2. Lluch, J., Gaitán, R., Escrivá, M., Camahort, E.: Multiresolution 3D rendering on mobile devices. In: Alexandrov, V.N., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2006. LNCS, vol. 3992, pp. 287–294. Springer, Heidelberg (2006)
3. Diepstraten, J., Gorke, M., Ertl, T.: Remote line rendering for mobile devices. In: Computer Graphics International, pp. 454–461 (2004)
4. Ji, G., Shen, H.W., Gao J.: Interactive exploration of remote isosurfaces with point-based non-photorealistic rendering. In: Visualization Symposium, pp. 25–32 (2008)
5. Shi, S., Nahrstedt, K., Campbell, R.: A real-time remote rendering system for interactive mobile graphics. ACM Trans. Multimedia Comput. Commun. Appl. (TOMM) **8** (2012)
6. Doellner, J., Hagedorn, B., Klimke, J.: Server-based rendering of large 3D scenes for mobile devices using G-buffer cube maps. In: Proceedings of the 17th International Conference on 3D Web Technology pp. 97–100 (2012)
7. Quillet, J.C., Thomas, G., Granier, X., Guitton, P., Marvie, J.E.: Using expressive rendering for remote visualization of large city models. In: The 11th International Conference on 3D Web Technology, Columbia, Maryland, USA, pp. 27–35 (2006)
8. Tizon, N., Moreno, C., Preda, M.: ROI based video streaming for 3D remote rendering. In: 2011 IEEE 13th International Workshop on Multimedia Signal Processing (MMSP), pp. 1–6 (2011)
9. Smit, F., van Liere, R., Beck, S., Froehlich, B.: An image-warping architecture for VR: low latency versus image quality. In: Virtual Reality Conference, pp. 27–34. IEEE (2009)
10. Tizon, N., Moreno, C., Cernea, M., Preda, M.: MPEG4-based adaptive remote rendering for video games. In: The 16th International Conference on 3D Web Technology, pp. 45–50 (2011)
11. Neven, D.M.: Interactive remote rendering. Master of Science, Computer Science, Delft University of Technology (2014)
12. Su, C., Ping, J., Yue, Q., Xukun, S.: Protected-3DMPS: remote-rendering based 3D model publishing system in digital museum. J. Comput. Inf. Syst. **2**, 277–283 (2006)
13. Okamoto, Y., Oishi, T., Ikeuchi, K.: Image-based network rendering of large meshes for cloud computing. Int. J. Comput. Vis. **94**, 12–22 (2011)
14. Noguera, J.M., Segura, R.J., Ogáyar, C.J., Joan-Arinyo, R.: A scalable architecture for 3D map navigation on mobile devices. Pers. Ubiquit. Comput. **17**, 1487–1502 (2013)
15. Koller, D., Turitzin, M., Levoy, M., Tarini, M., Croccia, G., Cignoni, P.: Protected interactive 3D graphics via remote rendering. ACM Trans. Graph. (TOG) **23**, 695–703 (2004)
16. Schoor, W., Hofmann, M., Adler, S., Benger, W., Preim, B., Mecke, R.: Remote rendering strategies for large biological datasets. In: Proceedings of the 5th High-End Visualization Workshop, Baton Rouge, Louisiana (2009)
17. Makhinya, M.: Performance challenges in distributed rendering systems. Department of Informatics, Computer Science, Knowledge and Systems, University of Zürich, Zürich (2012)

18. Deb, S., Bhattacharjee, S., Patidar, S., Narayanan, P.J.: Real-time streaming and rendering of terrains. In: Kalra, P.K., Peleg, S. (eds.) ICVGIP 2006. LNCS, vol. 4338, pp. 276–288. Springer, Heidelberg (2006)
19. Giesen, F., Schnabel, R., Klein, R.: Augmented compression for server-side rendering. In: Vision Modeling and Visualization, pp. 207–216 (2008)
20. Eisert, P., Fechteler, P.: Remote rendering of computer games. In: SIGMAP, pp. 438–443 (2007)
21. Claypool, M., Claypool, K.: Latency and player actions in online games. Commun. ACM **49**, 40–45 (2006)
22. Marquez, J., Domenech, J., Gil, J., Pont A.: Exploring the benefits of caching and prefetching in the mobile web. In: Proceedings of WCITD p. 8 (2008)
23. Lazem, S., Elteir, M., Abdel-Hamid, A., Gracanin, D.: Prediction-based prefetching for remote rendering streaming in mobile virtual environments. In: IEEE International Symposium on Signal Processing and Information Technology, pp. 760–765 (2007)
24. Yoo, W., Shi, S., Jeon, W.J., Nahrstedt, K., Campbell, R.H.: Real-time parallel remote rendering for mobile devices using graphics processing units. In: IEEE International Conference on Multimedia and Expo (ICME), pp. 902–907 (2010)
25. Callahan, S.P., Bavoil, L., Pascucci, V., Silva, C.T.: Progressive volume rendering of large unstructured grids. IEEE Trans. Visual. Comput. Graph. **12**, 1307–1314 (2006)