Freedom: Fast Recovery Enhanced VR Delivery Over Mobile Networks

Shu Shi, Varun Gupta, Rittwik Jana
AT&T Labs Research
Bedminster, NJ, USA
{shushi,vgupta,rjana}@research.att.com

DESCRIPTION

Although the mobile devices are easier to use and more portable today, the image quality is lower because the mobile devices have calculation and energy consumption limits. If we use mobile devices for watching virtual reality content, the images rendering will be performed on a chip which is placed on the headset.

For displaying the images usually the mobile device and the server which performs the rendering are placed on the same WLAN or will be used the viewpoint prediction or the frame prerendering. In the commonly used systems the server sends data to the client and these are shown directly to the user. These systems depend on the network and on the preprocessing of data because we need to have a small latency and a good performance.

Freedom gives us the possibility to keep the information on a remote server and we use 4G/LTE cellular networks in order to access the information (one issue here is the 4G network's latency, but Freedom can handle those variations). Freedom saves 80% of bandwidth compared to a classical 360 streaming because Freedom does not send the full panorama and does not perform a prefetching for the unused data.

Freedom can be used also for gaming (the user can enter data in real time using a connected controller) and in live streaming. As opposed to the other methods the server does not send data directly to the client, but creates Visible Area and Margin frames (VAM) at runtime. A Visible Area and Margin frame contains the pixels around the field of view.

Using this subset of pixels the device can render images for changing the view in a specified interval. This way, the local rendering can be applied and the frames can be intercalated by applying a fast refresh before the server performs the update. The client receives a new Visible Area and Margin frame (VAM) before the viewpoint is not placed in the old VAM area anymore (the margin size is large enough).

When the user is moving his head the client sends a request to the server for establish a new viewpoint, but until then the client can render locally usind the previous VAM frame. When the user is moving his head again, the client cannot render the new viewpoint anymore, but the previous viewpoint is rendered and the new viewpoint will be used when the server will send the new VAM frame.

At every step the system will need enough pixels for rendering the viewpoint locally. We must anticipate the next changes of user's view until the server will send a new VAM frame with pixels to the headset (to the client). For creating a VAM frame the image coresponding to the current viewpoint is rendered, this image must be scaled, a central area of the image must be cut and this area will be send to the client.

For the smaller devices the resolution is not so important, because the video quality remains the same (instead of 8K resolution can be used the 1080 resolution, which means that the bandwidth is not wasted anymore). Each VAM frame has an 32-bit integer (which is composed of 4 8-bit integers (scale, vertical movement parameter, horizontal movement parameter, roll).

People usually move their heads on the horizontal axis or on the vertical axis, almost never the movement will be more complex than this. The margin size and the scale are inversely proportional. For an easier rendering will be used an auxiliary frame, but it will be an issue (if the user moves his head fast enough the current viewpoint should not be rendered using the auxiliary frame which coresponds to the first viewpoint (because there are not enough remaining pixels) and a new VAM frame will be needed.

We must anticipate the way that user will move his head because the next viewpoints must be rendered using the auxiliary frame. For example, if the user is changing his view continuously to the left, we will not consider the current viewpoint in order to create a new auxiliary frame, we will take a viewpoint which is placed further left than the current viewpoint because we expect that the user will move his head further to the left.

The speed of the head's movement is also important for calculating the new viewpoint based on the current viewpoint. The VAM frames generation affects the performance because scaling and reprojecting operations must be applied for each pixel and is done in GPU (the GPU optimization plays an important role for the performance analysis).

The network latency is also important and is affected by the frame size. The LTE latency can be affected by transmission initiation, but LTE can handle big size frames because LTE has a large bandwidth. Wifi can be affected by the frame size because Wifi has not a large bandwidth. The latency and the bandwidth savings are inversely proportional.

Related Work

In the scenario when Freedom is used, the server can send VAM frames and using these frames the client can render locally until the viewpoint is not placed in the current VAM frame area anymore. Unlike Freedom Furion [1] creates a 360 degree view for each potential viewpoint (this is why Freedom is far better than Furion in terms of bandwidth savings).

Furion does not handle dynamic games or live content because the server must render the scene every time in advance (for example, if there is a game with static objects only the scene must be rendered, but Furion does not support the movements of a character which is controlled by the user).

If the scene uses only static objects and characters, Furion will not use any bandwidth, but if the scene uses dynamic objects the movement must be 5 times slower in order to keep the streaming without pauses. For a static scene Furion is more suitable than Freedom because Furion does not need refresh, but Freedom is still doing refresh.

The [2]nd article describes a classical system: the client receives the virtual reality content from the server and displays it. Unlike this system, Freedom creates VAM frames and those pixels are used by the client to render locally until the server will send the next VAM frame. For the classical system the refresh frequency is 90Hz and for Freedom the refresh frequency is 60Hz.

In the case of the classical system any kind of latency must be avoided in the communication between server and client, but in the case of Freedom this latency can be handled because of the locally rendering.

The image quality is pretty good for Freedom, but if the user moves his head very fast the distance between the current viewpoint and the next viewpoint will be large and the image quality will suffer. In this case we need to wait for the VAM frame which is provided by the server, which means that Freedom has a similar dependency on the latency of communication between server and client as the classical system [2] has.

When using Freedom the density of pixels and margin sizes are always the same. Outatime [3] (a cloud gaming system) uses a clipped cube map, which means that the density of pixels is different. This is an advantage of Freedom compared to Outatime because the frames have the same properties.

The human eye cannot notice a 100 ms delay, which means that Outatime [3], which is a gaming system, must hide the latency of communication between server and client by anticipating the character's movement. Freedom does not anticipate the character's movement, it only anticipates the user's head movement.

Another approach is based on hiding the network latency using the image decomposition and creating a new image [4]. The view is divided into 2 separate views: a primary view and a secondary view. The focus will be on the primary view and the details such as pixels shadows will be ignored. The performance of this system is better than the performance of Freedom and the system requirements of [4] are lower than the system requirements of Freedom, but Freedom is more accurate than this system.

In a simple arhitecture the actual image relies on the previous image, but the user cannot see the objects which were occluded in the previous scene. [4] shows that there are 2 cases of occluded objects: 1. An object is between the camera and the other object or 2. The objects are outside the field of view. The second problem can be solved by extending the field of view to 180 degrees and the first problem can be solved by showing the occulted objects only for the primary view.

Flashback [5] avoids the real-time image rendering by keeping in a cache the potential images which can be sent to the client. Flashback and Freedom can both handle static and dynamic scenes. If the mobile has a weak GPU, Flashback is better than Freedom because the Flashback's cache can keep an entire virtual reality scene and Freedom relies on a high performance GPU.

Flashback can support dynamic scenes because each object has its own cache. For a dynamic object we keep in the cache its relative distance to the player position and its orientation. The latency and the image quality of Flashback are similar to the latency and the image quality of a good gaming desktop.

QuickTime VR [6] is an image file format which uses 360-degree cylindrical panoramic images (Furion does the same thing). These images can be created using computer rendering or using normal photos. When the virtual camera is moving there are 6 degrees of freedom which are divided into 3 categories: 1. The movement on the 3 axis Ox, Oy and Oz, 2. Rotating the camera near the object while keeping the view to the center of the object (the viewpoint has been changed) and 3. Changing the viewpoint and the view (the view direction is not oriented to the center of the object anymore).

The panoramas are captured using multiple angles and the objects are captured by moving the camera to different positions, but keeping the viewpoint direction oriented to the center of the object. In 1995, the researchers found out how to explore VR objects from various angles using QuickTime and in 2019 they developed a system called Freedom which saves bandwidth by anticipating the user's head movements.

References

[1] Zeqi Lai, Y Charlie Hu, Yong Cui, Linhui Sun, and Ningwei Dai. 2017. Furion: Engineering High-Quality Immersive Virtual Reality on Todays Mobile Devices. In Proc. ACM MobiCom17.

[2] Luyang Liu, Ruigang Zhong,Wuyang Zhang, Yunxin Liu, Jiansong Zhang, Lintao Zhang, and Marco Gruteser. 2018. Cutting the Cord: Designing a High-quality Untethered VR System with Low Latency Remote Rendering. In Proc. ACM MobiSys18.

[3] Kyungmin Lee, David Chu, Eduardo Cuervo, Johannes Kopf, Yury Degtyarev, Sergey Grizan, AlecWolman, and Jason Flinn. 2015. Outatime: Using speculation to enable low-latency continuous interaction for mobile cloud gaming. In Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services. ACM, 151165.

[4] B. Reinert, J. Kopf, T. Ritschel, E. Cuervo, D. Chu, and H.-P. Seidel. Proxy-guided image-based rendering for mobile devices. Computer Graphics Forum, 35(7):353362, 2016.

[5] Kevin Boos, David Chu, and Eduardo Cuervo. 2016. Flashback: Immersive virtual reality on mobile devices via rendering memoization. In Proc. ACM MobiSys16.

[6] Chen, S.E., QuickTime VR An Image-Based Approach to Virtual Environment Navigation, Proc. SIGGRAPH 1995 (Los Angeles, CA, August 6-11, 1995). In Computer Graphics Proceedings, Annual Conference Series, 1995, ACM SIGGRAPH, pp. 29-38.