REAL-TIME VIRTUAL NAVIGATION PROVISION BY SIMPLE MEANS

Marek Domański, Adrian Dziembowski, Tomasz Grajek, Adam Grzelka,
Dawid Mieloch, Robert Ratajczak, Olgierd Stankiewicz, Jakub Stankowski, Krzysztof Wegner

Poznań University of Technology, Chair of Multimedia Telecommunications and Microelectronics, Poznań, Poland

ABSTRACT

The paper reports the results on virtual navigation in a sports hall. The video data is gathered and processed off-line thus producing the multiview plus depth representation of a scene. This representation is an input to the rendering server. Even a typical PC computer is able to serve a number of users that freely and independently navigate in real - time and watch virtual video. The users are able to navigate in the scene changing the virtual viewpoint and virtual view direction. The service is provided through a webpage, so a viewer does not need to install any software on smartphone, tablet or notebook. The paper discusses the technical solutions together with the rationale for their choice.

Index Terms— Free-viewpoint television, free navigation, virtual reality, view synthesis.

1. INTRODUCTION

Here, we deal with the virtual navigation, i.e. a functionality of future interactive video services that provides a viewer an ability to move freely around a scene and watch it in an arbitrary view direction and from virtual viewpoints on an arbitrary navigation trajectory. The respective video communication systems are called free-viewpoint television (FTV) [1-5], and are a subject of significant research efforts in the recent years. Most of the work is aimed at high-end thus extremely expensive systems, like the recently announced system of KDDI [6]. In contrary to them, we aim at cheap and simple FTV systems [2,7,8] with such applications like sports broadcasts, performances, interactive courses and manuals, and school teaching materials. We aim at smaller events that may be reported with some delay, maybe on the next day. We also leave the audio issues out of the scope of this paper.

In this paper, we deal with free navigation provision to the viewer. Except of view synthesis algorithms [9-14] and video streaming in complex systems [1, 13, 15, 16], the issue is nearly absent in the references.

2. SYSTEM ARCHITECTURE

Although the considerations of the paper are valid for various applications, we will demonstrate the system in the context of free navigation in a sports hall (cf. Fig. 1).

Fig. 1. A sports hall with a multi-camera rig.

The system provides horizontal virtual navigation around a scene with the limited ability to go into a scene, i.e. the virtual viewpoint may move towards the players but is unable to go in between the players (cf. Fig. 2). Moreover, a viewer may also dynamically change the view direction as shown in Fig. 2. Nevertheless, the angle of viewer rotation around a vertical axis is limited, i.e. the rotation angle practically should not exceed 40 degrees. Slight tilt of the virtual view direction is possible that improves subjective quality of the virtual navigation. The abovementioned limitations result from the limitations of the visual information available from the one-dimensional camera rig shown in Fig. 1.

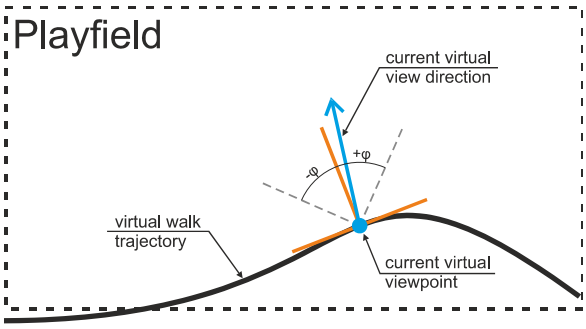

Fig. 2. An example of virtual navigation trajectory and virtual view directions.

We use the centralized model [2, 8, 15] of view synthesis, where the views requested by viewers are synthesized in the servers of the service provider, i.e. in the

rendering servers (edge servers). The number of rendering servers depends on the number of user terminals, as each server may serve a limited number of user terminals. Nevertheless, this approach does not need synthesis in user terminals thus increasing the availability of the service. Moreover, the bandwidth between the rendering server and user terminal is just the bandwidth for one video stream, and there is no need to transmit even a part of the 3D model of a scene.

In the centralized model, the user terminal sends requests for current virtual positions, and the rendering server responds with video frames synthesized for the requested position. The free navigation service is available as a video-on-demand service on the Internet. A user terminal may be as simple as a smartphone or a tablet equipped with any standard video decoder.

The general system architecture is depicted in Fig. 3. Only the lower part of the system in Fig. 3 is within the scope of the paper, as it was recently developed. Here, we consider the rendering server, the user terminal, and the communication between them. As a simple example of the rendering server, we are going to consider its implementation on a PC computer with Intel i7 processor.

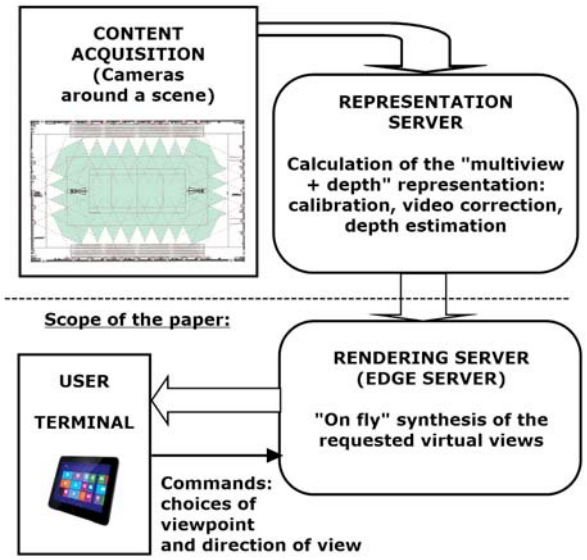

Fig. 3. The free-viewpoint television system.

3. REAL-TIME VIEW SYNTHESIS

More than 50% of the computational load of the rendering server corresponds to virtual view synthesis. Its algorithm is an improvement of VSRS [20] as it has to be simple because of the real-time requirement. Therefore, the virtual view is synthesized in a simplified way using only two real views (Fig. 4) as in [20]. Therefore, in the first step the proper real views have to be chosen – the chosen cameras should capture possibly highest percentage of the scene visible in the virtual view. Performed experiments show, that in the most cases the highest quality of the virtual view can be obtained using nearest left and nearest right real view – both for the linear (e.g. cameras located along a sports hall's sideline) and arc (sports hall's corner) camera setups.

When two real views are chosen, the depth map of the virtual view is being calculated – position of each pixel of two real views is multiplied by inverted projection matrix of the real view and the projection matrix of the virtual view. Then, these depth maps are being filtered in order to remove errors and artifacts caused by finite resolution of the depth and image and to smoothen surface of the objects thus improve the depth map quality. After filtering, the texture of the real views is being projected to the virtual view using virtual view's depth information. As a result, we have two separate virtual views – one for each real camera. These two images should be blended. Unfortunately, due to different cameras' characteristics and lighting conditions, virtual view synthesized from the left camera may have different color characteristics than the. view obtained for the right real view. Therefore, before blending, a simple color correction method should be applied. In proposed approach the color characteristics of the further view is adjusted to the characteristics of the closer one by adding an offset for all the pixels within the virtual view.

After view blending the virtual view contains information from both the real views, but it also contains areas where no pixel was projected. These areas have to be filled in using inpainting. In proposed approach we used realtime bilinear inpainting, where estimated color of the non-synthesized pixel is the weighted average of the closest synthesized pixel to the left, right, top and bottom (and the weight is the distance to each).

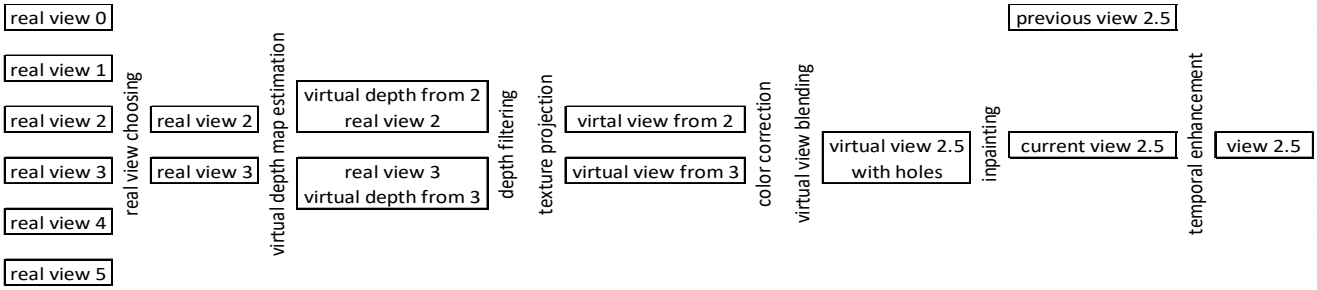

Fig. 4. Real-time view synthesis.

The last step of the view synthesis algorithm is the temporal enhancement operation. It is simple and thus very fast – for every pixel the color difference between two consecutive frames is checked. If the difference is less than a small threshold, both values are being averaged. This operation reduces flickering of static objects, i.e. in the background.

4. RENDERING SERVER

When a new user is connected to the rendering server, a new rendering pipeline is created. This enables independent navigation for each user. At the start, each user watches the scene from a default viewpoint. By swiping across the displayed image a user can request a change of his/her viewpoint. The information about requested change in viewing position is transmitted by client application through WebSocket connection to the server. The server updates virtual view position in the user's rendering pipeline (virtual view and input view selector block) and selects appropriate input views for the best rendering of the quality from the requested viewpoint. This new information is used immediately in the next time slot for frame rendering.

The necessary data (video plus depth) for input views related to the virtual view are pointed out by the input view selector block, and are fed into the user's rendering pipeline at the beginning of each time slot for frame rendering. The input views and the corresponding depth data are passed to synthesizer module and the new frame of a requested virtual view is rendered. In the case of fast virtual walk, each consecutive frame may be synthesized from a completely different viewpoint and view direction.

Therefore, the rendered image is compressed by the all-intra video AVC [17] encoder (or JPEG encoder [18]) and sent via WebSocket to the client application.

Obviously, the rendering server comprises also the respective audio processing blocks that are not considered here as mentioned in Introduction.

The rendering server is composed of: HTTP server, WebSocket server, a source pipeline, and the rendering pipelines (see Fig. 5).

The implementation is featured by the parallel processing using a pipeline and a thread pool, highly optimized CPU synthesis implemented using vector instructions (256bit AVX) and multithreading as well as by hardware compression.

5. USER TERMINAL

The service can be obtained from the rendering server by simply typing a rendering server address. Once the web page is loaded, connection with a rendering server is established automatically via WebSocket and video starts playing. Therefore, no special software needs to be downloaded before the start of the interactive viewing.

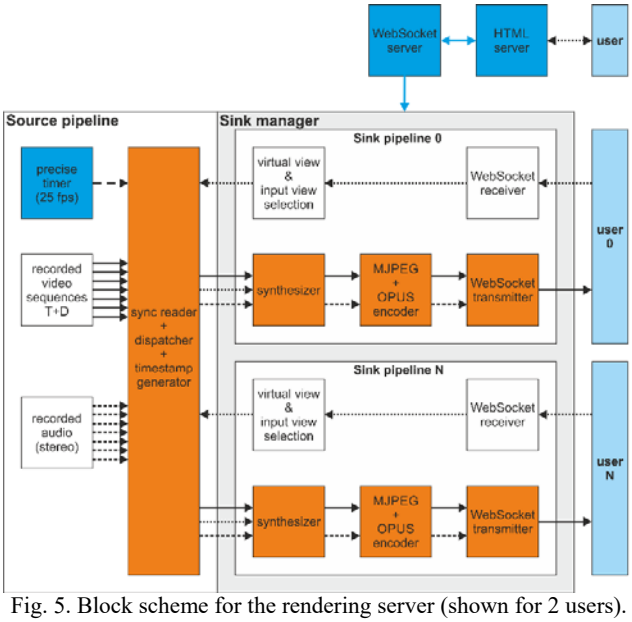

Fig. 5. Block scheme for the rendering server (shown for 2 users).

Using the devices with touchscreens, a user can shift the virtual viewpoint around the playfield by swiping the displayed image in two directions (left and right) as demonstrated in Fig. 6. It is also possible to step into the playfield and back by swiping the screen vertically. The rotation of the view direction may be achieved by diagonal swipe. The allocation of the finger motions to the changes of the viewpoint and the view direction is currently tested in order to make the viewer-terminal communication be the most intuitive and comfortable. The satisfaction of a viewer depends also on sensitivity of the swipe, so the sensitivity tune should be done experimentally for various virtual navigation scenarios.

Similar means of control is used on the touchpad of a laptop.

Fig. 6. Virtual navigation around the volleyball playfield controlled by motion of finger on a touchscreen of a smartphone.

From the finger motion, the application estimates the changes of the viewpoint and the view direction, and signals them to the rendering server. Rendering server calculates new

virtual view position, and immediately transmits a rendered video frame that corresponds to the chosen viewpoint and view direction.

Client application is implemented as web application in JavaScript with help of HTML 5 video component and WebSocket API. It has been successfully tested with modern web browsers on PC as Mozilla Firefox version 57.0.2 and Microsoft Edge version 41.16299.15.0, and with Google Chrome and Mozilla Firefox on the Android-powered mobile devices.

6. EXPERIMENTS AND THEIR RESULTS

The system exploits the centralized model of view synthesis where the client requests a frame with a defined viewpoint and view direction, and promptly receives the video frame ready to display. Therefore, the system latency is the issue of paramount importance. This latency consists of the latency of the rendering server, the latency of the client terminal, and latency of the network.

The total latency of the rendering server was measured using two multiview plus depth test video sequences: Poznan_Volleyball2 [19] and Poznan_Blocks2 [2]. The latency was measured for 10-second virtual-navigation video clips (i.e. 250 frames in each sequence). For the rendering server, the system CPU was Intel® Core™ i7-6700K @ 4GHz (4 cores, 8 threads). The processing times are summarized in Table I. Note that the pipeline processing is used, therefore, the maximum frame rate is not simply determined by the total processing time of a frame.

Table I. Processing times in the rendering server.

Sequence	Resolution		Average processing time per frame [ms]		Maximum achievable frame rate [Hz]
	Input	Output	Total frame processing	Virtual view synthesis	
Poznan_ Volleyball 2	Full HD	FullHD	44.28	25.65	39
	Full HD	qHD	21.97	11.64	86
	Full HD	SD	19.98	10.47	96
	SD	SD	6.64	3.75	266
Poznan_ Blocks2	Full HD	FullHD	40.75	23.04	43
	Full HD	qHD	20.62	10.30	97
	Full HD	SD	18.79	9.20	109
	SD	SD	6.43	3.64	275
qHD (quarter HD) = 960 × 540, Full HD = 1920 × 1080, SD = 720 × 576, all video is 25 frames per second progressive					

The fluctuation of the processing times was mostly below ±20% for individual frames. The ratio of the time for virtual view synthesis to the total processing time was almost always within the interval 0.50 ÷ 0.57.

The results from Table I demonstrate that a rendering server may be implemented on a PC computer than can serve even a number of viewers who navigate independently in the same scene.

The latency of the network is very changeable, even assuming lack of network traffic congestions. The total bi-directional latency between the rendering server and the user terminal is shown in Fig. 7. The values for individual packets are very different, therefore the distributions are shown. For the considered scenarios (Fig. 7), for more than 98% of packets the bi-directional latency was below 60 milliseconds.

From the experiments, it is known that users accept the delay between finger motion and a scene motion up to 250 – 350 milliseconds. Therefore, the total system latency is well below that limit.

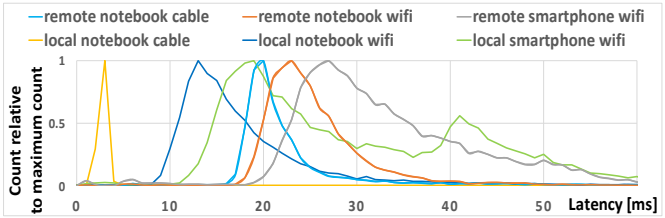

Fig. 7. Distributions of the network packet latency due to transmission:

- between *remote* location, ca. 10 kilometers within a city, or within *local* network in a single building,
- using a *notebook* or a *smartphone* as the user terminal,
- using the user terminal connected via *cable* or *wifi*.

The AVC bitrates are about 15-20 Mbps for HD and 4-6 Mbps for SD for fast virtual walking.

The extended report on subjective quality cannot to be included here for the sake of brevity. Nevertheless, the quality in the central parts of the playfield is classified as "good" while artifacts occurs at the edges of the scene.

7. CONCLUSIONS

In the paper we consider an original architecture of the rendering server and its communication with the user terminal in a simple low-cost FTV system. To our best knowledge, such a solution was not presented elsewhere. A preliminary limited version of the system was demonstrated during ICIP 2017 conference with no description or the experimental result. Now, the paper provides an original implementation of the rendering server as well as communication with the user, in particular with untethered one. The advantage of the solution is that no dedicated software need to be installed in the user terminal. From the user side, the solution is "log in to a webpage and navigate". Therefore, the solution is easy to be implemented commercially. All these results encourage us to believe that the development of usable FTV systems will be possible very soon.

ACKNOWLEDGEMENT

The research project was supported by The National Centre for Research and Development, Poland. Project no. TANGO1/266710/NCBR/2015.

REFERENCES

[1] M. Tanimoto, M. Panahpour, T. Fujii, T. Yendo, "FTV for 3-D spatial communication," *Proceedings of the IEEE*, vol. 100, Issue 4, pp. 905-917, Feb. 2012.

[2] O. Stankiewicz, M. Domański, A. Dziembowski, A. Grzelka, D. Mieloch, J. Samelak, "A free-viewpoint television system for horizontal virtual navigation," *IEEE Trans. Multimedia*, Early access, 2018.

[3] E. Bondarev, R. Miquel, M. Imbert, S. Zinger and P. H. N. de With, "On the technology roadmap of Free-Viewpoint 3DTV receivers," in *2011 IEEE International Conference on Consumer Electronics (ICCE),* Las Vegas, NV, USA, 2011, pp. 687-688.

[4] G. Lafruit, M. Domański, K. Wegner, T. Grajek, T. Senoh, J. Jung, P. Kovács, P. Goorts, L. Jorissen, A. Munteanu, B. Ceulemans, P. Carballeira, S. García, M. Tanimoto, "New visual coding exploration in MPEG: Super-MultiView and Free Navigation in Free viewpoint TV," in *IST Electronic Imaging, Stereoscopic Displays and Applications XXVII*, San Francisco 2016, pp. 1-9.

[5] C.-C. Lee, A. Tabatabai, K. Tashiro, "Free viewpoint video (FVV) survey and future research direction," APSIPA Transactions on Signal and Information Processing, vol. 4, Oct. 2015.

[6] "Introduction to KDDI's 5G network service", presentation at H20 seminar during *121th MPEG Meeting*, Gwangju, January 2018.

[7] M. Domański, M. Bartkowiak, A. Dziembowski, T. Grajek, A. Grzelka, A. Łuczak, D. Mieloch, J. Samelak, O. Stankiewicz, J. Stankowski, Krzysztof Wegner, "New results in free-viewpoint television systems for horizontal virtual navigation," in *2016 IEEE International Conference on Multimedia and Expo (ICME),* Seattle, WA, 2016.

[8] M. Domański, A. Dziembowski, D. Mieloch, A. Łuczak, O. Stankiewicz and K. Wegner, "A practical approach to acquisition and processing of free viewpoint video," in *2015 Picture Coding Symposium (PCS)*, Cairns, QLD, 2015, pp. 10-14.

[9] C. Zhu, S. Li, "Depth image based view synthesis: New insights and perspectives on hole generation and filling", IEEE Trans. Broadcasting, Vol. 62, pp. 82-93, 2015.

[10] T. Tezuka, M. Tehrani, K. Tkahashi, T. Fuji, "View synthesis using superpixel based inpainting capable of occlusion handling aand hole filling", *Picture Coding Symposium*, pp. 124-128, 2015.

[11] M. Köppel, K. Müller, T. Wiegand, "Filling disocclusions in extrapolated virtual views using hybrid texture synthesis", *IEEE Trans. Broadcasting*, vol. 62, 2016, pp. 457-469.

[12] A. Dziembowski, A. Grzelka, D. Mieloch, O. Stankiewicz, K. Wegner, M. Domański, "Multiview synthesis – improved view synthesis for virtual navigation", in *32nd Picture Coding Symposium PCS 2016*, Nuremberg, Germany, Dec. 2016.

[13] L. Toni, G. Cheung, P. Frossard, "In-network view synthesis for interactive multiview video systems", *IEEE Trans. Multimedia*, vol. 18, 2016, pp. 852-864.

[14] B. Ceulemans; Shao-Ping Lu; G. Lafruit; A. Munteanu, "Robust multiview synthesis for wide-baseline camera arrays," *IEEE Trans. Multimedia*, Early access, 2018.

[15] J. Kim, J. Jang and D. H. Kim, "Design of platform and packet structure for the free-viewpoint television," in *The 18th IEEE International Symposium on Consumer Electronics (ISCE 2014),* JeJu Island, 2014.

[16] T. Fujihashi, Z. Pan and T. Watanabe, "UMSM: A traffic reduction method on multi-view video streaming for multiple users," *IEEE Transactions on Multimedia*, vol. 16, no. 1, pp. 228-241, Jan. 2014.

[17] "Coding of audio-visual objects, Part 10: Advanced Video Coding," ISO/IEC IS 14496-10, 2014.

[18] "JPEG 2000 image coding system, Part 3: Motion JPEG2000," ISO/IEC IS 15444-3, ITU-T Rec. T.802, 2007.

[19] M. Domański, A. Dziembowski, T. Grajek, A. Grzelka, K. Klimaszewski, D. Mieloch, R. Ratajczak, O. Stankiewicz, J. Siast, J. Stankowski, K. Wegner, „Free-viewpoint television demonstration for sports events," *ISO/IEC JTC1/SC29/WG11* Doc. MPEG/ M41992, January 2018, Gwangju, Korea.

[20] O. Stankiewicz, K. Wegner, M. Tanimoto, M. Domański, "Enhanced Depth Estimation Reference Software (DERS) for Free-viewpoint Television," ISO/IEC JTC1/SC29/WG11, Doc. MPEG M31518, Geneva, 2013.