Optimized Camera Handover Scheme in Free Viewpoint Video Streaming

Balázs Háló, Árpád Huszák

Department of Networked Systems and Services, Multimedia Networks and Services Laboratory
Budapest University of Technology and Economics, Magyar Tudósok krt.2., H-1117 Budapest, Hungary

{halo, huszak}@hit.bme.hu

Abstract

Free-viewpoint video (FVV) is a promising approach that allows users to control their viewpoint and generate virtual views from any desired perspective. The individual user viewpoints are synthetized from two or more camera streams and correspondent depth sequences. In case of continuous viewpoint changes, the camera inputs of the view synthesis process must be changed in a seamless way, in order to avoid the starvation of the viewpoint synthesizer algorithm. Starvation occurs when the desired user viewpoint cannot be synthetized with the currently streamed camera views, thus the FVV playout interrupts. In this paper we proposed three camera handover schemes (TCC, MA, SA) based on viewpoint prediction in order to minimize the probability of playout stalls and find the tradeoff between the image quality and the camera handover frequency. Our simulation results show that the introduced camera switching methods can reduce the handover frequency with more than 40%, hence the viewpoint synthesis starvation and the playout interruption can be minimized. By providing seamless viewpoint changes, the quality of experience can be significantly improved, making the new FVV service more attractive in the future.

Keywords: free viewpoint video; streaming; viewpoint prediction

1. Introduction

Free-viewpoint video (FVV) is a promising approach to offer freedom to users' perspective selection while watching multiview video streams. The new type of interactive FVV multimedia service allows users to control their viewpoint and generate new views of a dynamic scene from any desired perspective. The interactive free navigation within a visual scene is similar to the experiment known in 3D computer graphics applications. The main difference is that FVV targets real world scenes, captured by real cameras, without using 3D graphical models. Different views can be synthetized depending on the requested user specific perspective that can be controlled e.g. by moving or turning their head or changing position in a room. Free-viewpoint streaming with its advanced features is foreseen as the next big step in 3D video technology. These functionalities can be used for various services, such as visual communication, media broadcast and education. However, a commercial free-viewpoint television (FTV) service will be similar to the IPTV solutions, the difference is that not only one stream belongs to a TV channel, but several video streams [1]. The other difference is that the displayed media content is also dissimilar due the individual user viewpoints.

The uniquely generated and displayed user views are composed from two or more high bitrate color and corresponding depth camera streams that must be delivered to the users depending on their continuously changing perspective. By increasing the number of the deployed cameras and the density of the camera setup, the free-viewpoint video experience becomes more realistic. But on the other hand, more camera streams requires higher network capacity. Without advanced camera handover schemes the increased network traffic load and latency can disturb the user experience. Instead of forwarding all camera streams, it is a reasonable solution to deliver only those camera views that are required for the viewpoint synthesis. The intelligent camera selection is an efficient solution to reduce the network load, however other difficulties appears regarding to camera switches. Due to continuous viewpoint changes the camera inputs of the view synthesis process must be also changed with short time constrains in order to avoid the starvation of the viewpoint synthesizer algorithm. Starvation occurs when the desired user viewpoint cannot be synthetized with the currently streamed camera views, but the newly requested camera views are still missing. In case of an intensively changing user viewpoint, the seamless camera handover can be very challenging.

The research activity on FVV topic is very intensive and focusing mainly on coding and viewpoint rendering issues, while network delivery was investigated with lower intensity. However, it is true that the key techniques of FVV are still not efficient enough to provide services with acceptable quality. Viewpoint synthesis is a very computational hungry process and the existing algorithms are still trying to find the tradeoff between the video quality of the synthetized view and the rendering time of the FVV algorithms. Basically, two image-based viewpoint synthesis methods can be used to generate an individual viewpoint from the camera sequences.

The first method is the Light Field Rendering (LFR) [2] algorithm that interpolates a virtual view from multi-

camera images, therefore sufficiently large number of cameras have to be set up to achieve high performance rendering, and a tremendous amount of image data needs to be processed. The camera streams can be only transmitted in broadband IP network even the streams are compressed. Contrariwise, if the number of used cameras is too low, interpolation and occlusion artifacts will appear in the synthesized images, possibly affecting the quality.

The other method is the Depth Image Based Rendering (DIBR) [3] that uses fewer images and corresponding depth maps to render new views. The basic idea of the DIBR methods is to perform 3D warping to the virtual viewpoint using texture and depth information of the reference cameras. Based on the depth information artifacts are removed by post-processing the projected images. These images are then blended together and the remaining disocclusions are filled in by inpainting techniques [4]. In case of DIBR the amount of data that must be delivered through the network is significantly lower compared to the LFR solution, so it can be used even in lower bandwidth networks. If the depth images are generated offline, DIBR based FVV cannot support live streaming. Fortunately, real-time depth cameras, such as Microsoft Kinect or Creative Senz3D, appeared on the market, thus implementing live DIBR FVV streaming service became possible with the new generation of depth cameras [5].

From the network delivery point of view, FVV is different from traditional video streaming. An FVV service requires several video streams and depth images captured by cameras and depth sensors that are deployed in different locations. The rendering process requires two or three camera streams to synthetize the individual viewpoint, but if the viewpoint is changing, camera handovers may be requested, too. The required camera streams may change continuously due to the free navigation of viewpoint, hence effective camera switching schemes are required to avoid starvation of the viewpoint synthesizer algorithm.

In this work we were focusing on camera selection and camera handover process in case of DIBR view synthesis method. During the user viewpoint changes, new camera and depth video streams may be required, however it is not obvious which camera set will lead to highest user quality. The simplest solution is to choose those neighboring cameras that are closest to the desired viewpoint. In this case the image quality will be the highest, but on the other hand frequent camera handovers are necessary to serve the user with continuously changing viewpoint. During a camera handover the user disconnects from the unrequired camera source and attaches to a new one that will be used for the new viewpoint synthesis. Depending on the network latency conditions, the duration of the handover process can be too long and lead to playout interruption. Our aim was to find the tradeoff between the image quality and the camera handover frequency by proposing a novel FVV camera set selection methods. If the cameras are selected optimally based on the user viewpoint movement behavior, the handover frequency can be significantly reduced, hence the viewpoint synthesis starvation and so the playout interruption can be minimized. In order to analyze the performance of the proposed camera handover scheme a Java simulation environment was implemented.

The rest of this paper is organized as follows. The background of free viewpoint video viewpoint synthesis and streaming methods are presented in Section II. In Section III, the proposed camera set selection scheme based on user behavior is introduced. The evaluation of the optimized camera handover scheme and the overview of the obtained performance results are presented in Section IV. Finally, the summary of the paper and the conclusions can be found in the last section.

2. Related works on Free Viewpoint Video streaming

Delivery of multimedia content generally requires high link capacity and low latency in order to provide acceptable quality of media streams. The transmission of traditional high resolution single-view video is still challenging, but in case of multi-view videos this challenge becomes more complex. A DIBR-based free viewpoint video service model is built from five main components: scene capturing, video coding, streaming, viewpoint synthesis and display.

Scene capturing

FVV service relies on special acquisition systems that use multiple cameras to capture real world scenery. In order to capture scene geometry, the general color cameras are combined with active depth sensors. Different camera array layouts can be used e.g., linear (1D), plane (2D) or dome type (3D) that impose practical limitations on navigation (Fig. 1). The camera density is the other important feature of a FVV system that has significant impact on the achievable quality of the synthetized view. Unfortunately, the more cameras are deployed, the more processing is required to take advantage of all the cameras and the system becomes more sensitive to camera calibration inaccuracies. Therefore, the classical trade-off must be consider between costs, complexity and quality (navigation range, quality of virtual views, etc.).

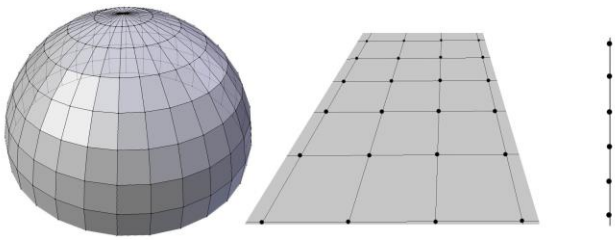

Fig. 1. Dome, plane and line camera array layouts

Free navigation has already been demonstrated in sport applications [6], where the user experience approaches the feeling of being present in the field. A DIBR-based solution was introduced by C. Kuster et al. [7]. Their FreeCam systems was built from few static color video cameras and Kinect depth sensors. The implemented FreeCam solution provided live free-viewpoint video at interactive rates using

a small number of off-the-shelf sensor components and quite standard computing power.

Multi-view video coding

Multi-view video storage and network delivery is not possible without efficient data compression. To synthetize a virtual viewpoint from existing camera views, the camera streams must be forwarded to the renderer that can be deployed in the user equipment, in a media server, or distributed in the network. Depending on the image-based 3D representation format, three different categories can be distinguished: two-view stereo video, multi-view video and multi-view video plus depth (MVD).

The two-view stereo (stereoscopic) video is the simplest scenario that consists of two videos representing the left and right views from two slightly different viewpoints. Besides the temporal correlation of the frames the spatial redundancy is also utilized in the coding process.

The same approach can be followed in the case of multi-view video that uses set of synchronized cameras, which are capturing the same scene from different viewpoints. An efficient way to encode two or more videos showing the same scenery from different viewpoints is known as multi-view video coding (MVC) [8][9]. MVC is an extension of H.264/AVC that exploits both inter-view and temporal redundancies for efficient compression and keeps full resolution of all views (Fig. 2.).

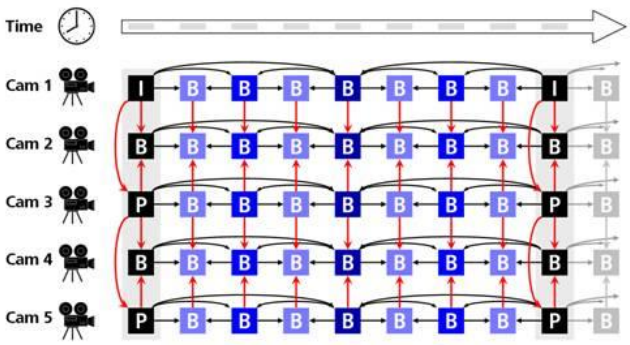

Fig. 2. MVC spatial and temporal frame dependencies

Multi-view video plus depth (MVD) [10] representation uses per-pixel depth map sequences associated with multi-view texture video. Similarly to MVC, each stream can be encoded by considering the inter-view and intra-view coherences among all frames in the depth and color information from different views to remove the temporal and spatial redundancy. Depth maps are captured originally via depth or infrared cameras simultaneously with ordinary camera array. Continuous depth data is very important in 3D warping algorithms for high quality virtual image interpolation. The depth information is generally transformed to a monochromatic, luminance-only image taking values between 0 and 255 as shown in Fig. 3. In general, the depth channel requires an extra 10–20% of bitrates to encode the depth information [11]. The coding standard that supports video plus depth is known as MPEG-C Part 3 [12].

Fig. 3. Video-plus-depth representation

Viewpoint synthesis

Image based view synthesis in real time is still an open research problem that gains a lot of attention. The intermediate virtual views are generated from available natural camera views by interpolation without 3D geometry models. However, dense sampling of the real world with a sufficiently large number of natural cameras is necessary. Hence, tremendous amount of image data needs to be processed. If the number of used cameras is too low, interpolation and occlusion artifacts will appear in the synthesized views causing reduced quality. Several image based solutions have been proposed [15][16][17] that often have problems in terms of both computation time and perceptual quality of synthesized views.

In case of Depth Image Based Rendering (DIBR) approach [3][18] at least two camera streams and the corresponding depth map sequences must be available at the renderer to generate an individual viewpoint. The color image and the associated depth map along with camera calibration information, any pixel of the image can be projected into the 3D space and then projected back onto an arbitrary virtual camera plane, creating a virtual image. Conceptually, this method can be understood as a two-step process [19]: (1) 3D image warping: it uses depth data and associated camera parameters to back-project pixel samples from reference images to the proper 3D locations and re-project them onto the new synthesized image space; (2) reconstruction and re-sampling: determination of pixel sample values in the synthesized image.

The accuracy of the depth data significantly impacts the quality of the generated virtual view. The amount of distortion increases with the difference of the virtual view and the original perspective, drastically limiting the potential navigation range using single video plus depth. The synthesis ability of image based representation has limitations on the range of view change.

Streaming

Three FVV service models can be distinguished based on the viewpoint synthesis process location in the network: server-based, client-based and distributed model. In a server-based model all the camera views and corresponding depth map sequences are handled by a media server that receives the desired viewpoint coordinates from the

customers and syntheses unique virtual viewpoint stream for each user. The drawback of the server-based solution is that the computational capacity of the media server may limit the scalability of this approach. In the client-based approach camera streams and depth sequences are delivered to the clients to generate their own virtual views independently, so the limited resource capacity problem of the centralized media server can be avoided, but huge network traffic must be delivered in the network. Fortunately, multicast delivery can reduce the overall network traffic, however the requested camera streams by a user is changing continuously that must be also handled using advanced multicast group management methods. In our previous work we proposed such solutions [20]. Finally, the third model is a distributed approach, where the viewpoint rendering is done in distributed locations in the network that was studied in [21].

In most of the FVV related works client-based streaming model was assumed. Authors of [22] proposed a QoS aware FVV streaming solution for light field rendering (LFR) algorithm. The paper focuses on I-frame retransmission and jump frame techniques in the application layer based on RTP/RTCP streaming protocols to support different level of QoS. A streaming system for DIBR based FVV over IP networks was introduced in [23]. The proposed solution divides video streams into depth video, texture video and common video, and transmits them in individual RTP/RTSP streams, making the service more robust against transfer errors, however it did not solve view switching and synchronization problems.

Camera stream selection is an efficient method to reduce the bandwidth requirements of multi-view video, where only a subset of views is streamed depending on the user's current viewing angle. To select which views should be delivered, the viewer's current viewpoint is tracked and a prediction of future perspective is calculated [20][24][25]. Kurutepe et al. [26] presented a multi-view streaming framework using separate RTSP sessions to deliver camera views allowing the client to choose only the required number of sessions. To the best of our knowledge, previous research on camera stream selection was focusing on linear camera array, while plane or dome camera layout was not investigated before.

3. Camera selection and handover scheme

The viewpoint synthesis is based on two or three real cameras depending on the deployed FVV system layout. If the cameras are deployed in line, the user can move his/her perspective only on straight trajectory. If the FVV service provider wants to offer more freedom in viewpoint selection, plane camera layout is preferred. However, in this case three camera streams are required for the viewpoint synthesis. In this work, we are focusing on plane camera layout.

The viewpoint must be always within the area determined by the three selected cameras that are used for the vie synthesis. When a FVV user freely changes his/her desired viewpoint, the requested real camera streams may also change, triggering camera selection process. Each change interrupts the streaming of the previous camera and initializes the delivery of the new one. Due to network bandwidth limitations, sending all images to every client is not possible, thus the number of camera streams must be minimized. In case of client based or distributed viewpoint synthesis approach, the late arrival of the new camera stream can interrupt the viewpoint rendering and the video playout. If the user changes the viewpoint too fast, the required camera flows will not arrive in time due to network delivery delay. Therefore, our aim was to minimize the number of camera changes avoiding interruptions (camera starving), but still offer high video quality.

If we choose the closest cameras to the desired viewpoint, the synthetized image quality will be the best, but on the other hand frequent camera changes will occur. Hence, the camera change minimization approach has the opposite effect on the rendered virtual view: the less camera streams are used, the more inaccurate the produced images will be. One of our main goals was to find the optimum number of cameras, which can be streamed without disruption to the playout and also provides an adequate quality of rendered images.

In order to find the tradeoff between the synthetized video quality and the playout interruption, we proposed new algorithms that minimizes the following values:

- number of camera changes (handover)
- average distances between cameras in a group of three (this is a kind of qualitative parameter, because better images can be produced using cameras located close to the selected viewpoint)
- starvation of the viewpoint rendering process (by using viewpoint prediction, the probability of a required camera image being late can be reduced) (Fig. 4.)

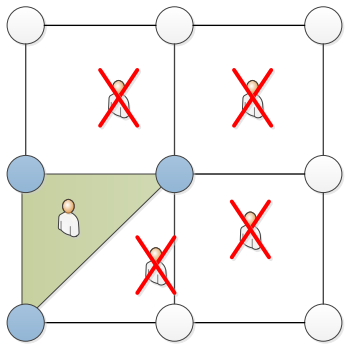

Fig. 4. Starvation of the viewpoint rendering process

Different approaches can be considered in order to build a FVV streaming service offering seamless viewpoint changes. The seamlessness relies on the camera switching performance, so in this paper three competing camera selection algorithms are proposed that can be also extended with viewpoint prediction. Of course each one has benefits and drawback that can be less or more important depending on the user behavior.

Three Closest Cameras approach (TCC)

The simplest approach and also the principle for all further algorithms is to find the closest cameras (Fig. 5.). This procedure includes three minimum search algorithms that pick out the three cameras with the shortest distance to the viewpoint. At lower viewpoint movement speed this algorithm will cause only one camera stream to be replaced, but with the speed increasing this value can reach a maximum of three camera handovers. On the other hand, this algorithm always finds the closest cameras, so in terms of quality (estimated from the sum of the distances between cameras) it provides the best possible image quality in every case. Nevertheless, due to the small distance of the cameras causing frequent camera handovers, starvation can occur more frequently interrupting the production of the FVV stream.

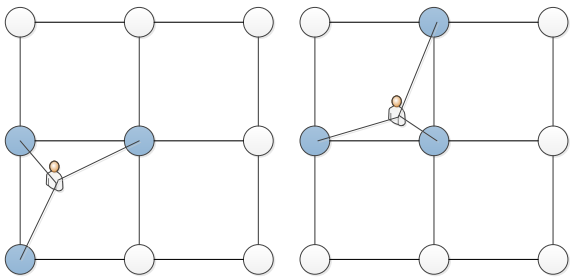

Fig. 5. Three Closest Cameras (TCC) algorithm

Scaling algorithm (SA)

The main purpose of the Scaling algorithm is to improve the TCC approach by estimating future viewpoint coordinates and the potential viewpoint route. Instead of selecting the closet cameras (TCC), the SA approach will choose further cameras if the user viewpoint is moving fast. The SA-based camera selection depends on the viewpoint change velocity and its direction, so we have differentiated three basic instances of viewpoint change behavior:

1. slow: the algorithm is not scaling (same as TCC)
2. normal: adds +1 camera distance to the nearest group of three
3. fast: adds +2 camera distances to the nearest group of three

The algorithmic structure behind scaling is the following. The potential layouts how the camera triangle can be scaled are determined by both the velocity vector and the three closest camera triangle. The SA algorithm examines the closest camera coordinates and decides, whether another camera should be selected, located further from the viewpoint. In this decision procedure each three cameras must be examined and find which one will be the reference point of the scaling. The camera in the reference point will remain unchanged, while the other two closest camera can be replaced by other ones, depending on the velocity vector (Fig. 6.).

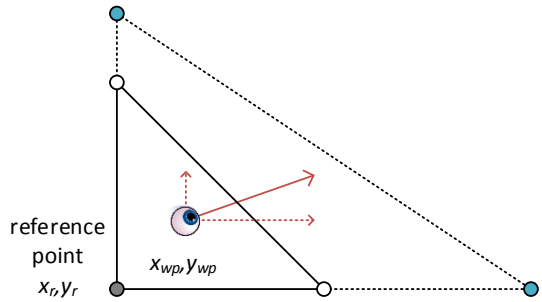

Fig. 6. Reference camera position in SA

This reference point (black point on Fig. 7.) can be calculated by subtracting the coordinates of the camera from the viewpoint's coordinates. The sign (sgn(x_{wp}-x_r) and sgn(y_{wp}-y_r)) of the values are then compared with the signs of the velocity vector coordinates. If the signs are the same, scaling is not used on the examined camera, but performed on every other point at the same time. It is highly important to scale only two of the points, as scaling all vertices can cause the viewpoint to slip out from the camera triangle causing starvation in the viewpoint synthesis process.

Once the fixed reference point is chosen, scaling can be applied to other vertices. As shown in Fig. 7, the potential replacements of the points are given by the area marked by the rectangles. It can be a one-dimensional camera row or a two-dimensional camera block depending on the scaling factor determined by *x* and *y* axes values of the viewpoint velocity vector.

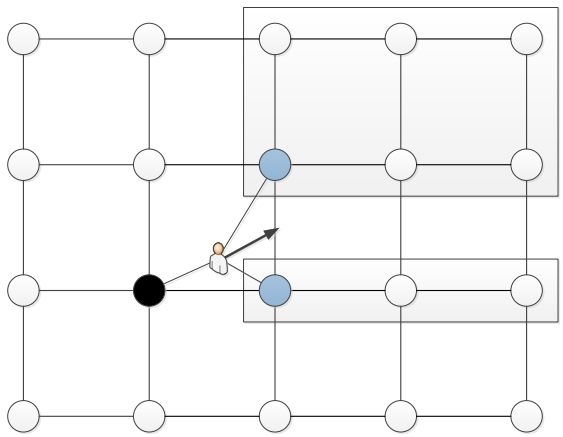

Fig. 7. The Scaling algorithm (SA)

The signs of the viewpoint velocity vector coordinates (sgn(x_{wp}), sgn(y_{wp})) shows in which direction it is possible to scale. After finding the fixed reference point, the following action is to perform on the other cameras: a vector (v_x,v_y) is calculated by subtracting the coordinates of the camera (x_{cam},y_{cam}) from the coordinates of the viewpoint.

$$\left(v_x, v_y\right) = \left(x_{wp} - x_{cam}, y_{wp} - y_{cam}\right) \tag{1}$$

Thereafter a comparison is performed between the signs of the vector given above and the sign of the velocity vector. If the signs do not match, the scale value with the sign of the velocity vector is to be added to the selected camera's coordinate.

$$\text{if } \left\{\left(\text{sgn}(x_{vel})>0\right) \&\& \left(\text{sgn}(x_{wp}-x_{cam})<0\right)\right\} \text{ then } x_{cam}=x_{cam}+v_x \quad (2)$$
$$\text{if } \left\{\left(\text{sgn}(x_{vel})<0\right) \&\& \text{s}\left(\text{gn}(x_{wp}-x_{cam})>0\right)\right\} \text{ then } x_{cam}=x_{cam}-v_x$$

Likewise process must be performed in all other cases (Table 1) depending on the velocity vector, fixed point and the scalable cameras.

Table 1

Finding fixed point from velocity vectors and scale values

velocity vector	camera	x_{cam} scale	y_{cam} scale
x: +, y: +	x: +, y: +	x: - → x+v	y: - → y+v
x: +, y: -	x: +, y: -	x: - → x+v	y: + → y-v
x: -, y: +	x: -, y: +	x: + → x-v	y: - → y+v
x: -, y: -	x: -, y: -	x: + → x-v	y: + → y-v

With the SA extension, the cameras covers larger area depending on the viewpoint velocity. Larger area means that the number of camera handovers is lower, but on the other hand the cameras used for the viewpoint synthesis are further. Therefore, the composed image quality will be lower. The operation of this method requires viewpoint velocity values calculated from previous viewpoint coordinates, thus the free viewpoint video service must begin with the TCC camera selection approach and switch to SA later when historical viewpoint coordinates can be used for the velocity vector determination.

Mirroring algorithm (MA)

We have investigated another camera selection approach based on our observation during the TCC performance evaluation. In cases when the predicted viewpoint slipped out from the camera triangle during the simulation of TCC algorithm, typically only one change of camera streams happens. In SA, only the fixed reference camera remains unchanged, while the two other cameras are replaced. However, it is true that camera handover process happens rarer compared to TCC. Our purpose was combine the benefits of TCC and SA by having only one camera handover in the same time, but also decreasing the frequency of camera handovers. Therefore, in cases of starvation the three cameras are not entirely replaced using the Mirroring Algorithm (MA), but only one of them as far as possible. The benefit of this method is that the streamed camera needs to be interrupted and replaced by only one other camera, therefore the probability of starvation is lower.

In case of the MA approach, the camera is reflected symmetrically in the center point of one of the edges defined by the camera's vertices (Fig. 8.). To do this, the camera that will replaced needs to be identified.

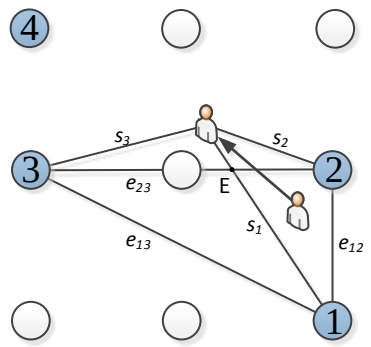

Fig. 8. Mirroring algorithm (MA)

Finding the camera to be mirrored goes as follows. First, all cameras needs to be connected to each other with edges (e_{12}, e_{13}, e_{23}), while the predicted viewpoint is connected with a segment (s_1, s_2, s_3) as shown on Fig. 8. The intersection of these edges and segment (E) is now to be examined in order to find the camera to reflect. Three cases can be differentiated according to this examination:

1. no intersection point found: the camera which is currently linked to the viewpoint will not be reflected in this case. For example: no intersection of the edge e_{13} and the segment (s_1)
2. the intersection point (E) is on the segment: the camera is to be mirrored on the center point of the edge linking the other cameras. For example: the intersection of the edge e_{23} and the segment s_1.
3. parallel: same as in the first case.

After finding the right camera to be mirrored, the position of the new camera needs to be calculated. Adding the difference between the coordinates of the two other cameras and the initial camera to the coordinates of the initial one gives the new camera triangle.

Viewpoint prediction methods

One main requirement of FVV systems is the fast cameras switchover, in order to avoid starvation of the viewpoint synthesis process. To do that, potential viewpoints have to be predicted. The efficiency of camera selection algorithms is highly influenced by the accuracy of this prediction. We used two basic methods of prediction and tested via simulations.

The first method is based on averaging the viewpoint motion parameters. We differentiated two variants, depending on the amount of previous viewpoint coordinates used for the estimation:

1. using all former data
2. using only the latest n viewpoint coordinates

In the first case the next viewpoint coordinates ($x_{wp,n}$,$y_{wp,n}$) are estimated by calculating the average of all displacements so far. The second case shows high similarity to the first one. However, it includes a window (w), which limits the inputs used for the average calculations:

$$x_{wp,n} = x_{wp,n-1} + \frac{\sum_{i=1}^{w}\left(x_{wp,n-i} - y_{wp,n-i-1}\right)}{w} \quad (3)$$

$$y_{wp,n} = y_{wp,n-1} + \frac{\sum_{i=1}^{w}\left(y_{wp,n-i} - y_{wp,n-i-1}\right)}{w} \quad (4)$$

This method can be more efficient when patterns are alternating on a high degree as it ignores old input data.

The second method is based on Kalman-filter. The Kalman-filter provides an optimal estimate for a state of a variable system through series of measurements with the confounding factors being taken into account. The advantage of the model is that it finds the optimal averaging factor for each state based on past events. The algorithm works in two steps: Time Update – Predict, Measurement Update – Correct. In the first step (Predict) the state and error covariance ahead are projected. In the second step (Correct) the Kalman gain is computed first. The Kalman gain is then used to update the estimation along with the next measurement input. Thereafter the error covariance is updated similarly. The output of Correct is always the input of Predict. In the beginning it is an estimated initial value.

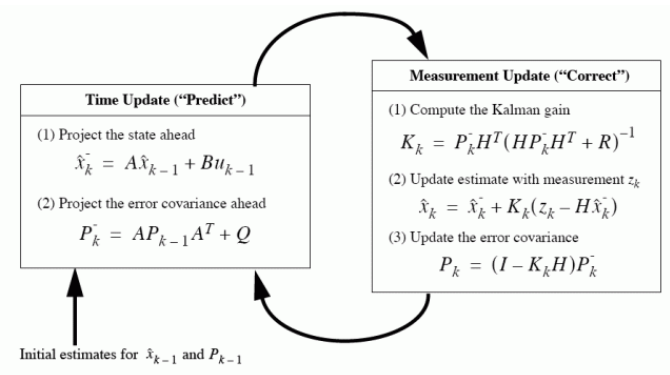

Fig. 9. The Kalman-filter

We used an open-source JAVA implementation of Kalman-filter (JKalman) for our simulation. This way, we were able to specify the number of dynamic parameters and the number of measured data. Four dynamic parameters were added, so the distances in the system could be tracked as well (by each coordinate). We selected the number of measured data to be two as our aim was to predict the viewpoint *x* and *y* coordinate.

4. Evaluation

In order to test the algorithms introduced previously, a simulation environment was implemented, where the proposed algorithms can be evaluated. The simulation framework and the algorithms were implemented in JAVA language. The implemented framework captures the path of the viewpoint movement and determines which cameras must be used for the viewpoint synthesis in order to minimize the camera handover frequency, but also maximize the quality of the rendered stream. In an advanced solution head tracking can be used to determinate the current viewpoint coordinates, however we emulated the viewpoint changes by fixed viewpoint trajectories.

In order to compare the different camera switch strategies, the viewpoint coordinates were captured and stored in *xml* format, so the same viewpoint trajectory was used for each camera selection algorithm. Every step of the viewpoint movement is required to be given in *xml* format, containing the time elapsed since the start of the movement (*timestamp*) and the coordinate of *x* and *y* axis. The description of the camera network was also done in *xml* format similarly to the definition of movement series. The location of each camera was given with its *x* and *y* coordinates.

To implemented simulation tool records the particular viewpoint and its estimated value. Thus occurrences of starvation can be detected by comparing the particular viewpoint coordinates and the estimated ones. If the real and the estimated viewpoint cannot be served with the same three cameras, starvation occurs. Both the number of camera switches in the last step and the overall number of switches in the movement series are also counted. The quality parameters of the three cameras are given calculated as the average distance from the actual viewpoint.

In the analyzed FVV service 100 cameras were deployed in a 10×10 grid. Different predefined viewpoint trajectories were used in order to simulate different user behaviors.

Predefined viewpoint trajectories

We captured three viewpoint movement series in order to simulate user perspective changes. Each one contains hundred viewpoint positions, describing a slow, a normal and a high-intensity movement. In our measurements the reference distance was the distance between two adjacent cameras (hereinafter referred as camera unit, *CU*). Time, similarly to distance was not given in exact units. The time unit (hereinafter referred to as *TU*) was the duration of one viewpoint movement step. In our measurements we simulated 100 TU long viewpoint movements that represents a viewpoint trajectory with 100 viewpoint coordinates as shown in Fig. 10.

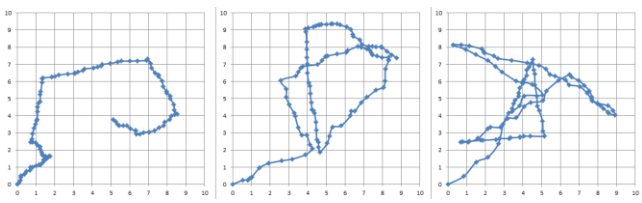

Fig. 10. Defined viewpoint trajectories

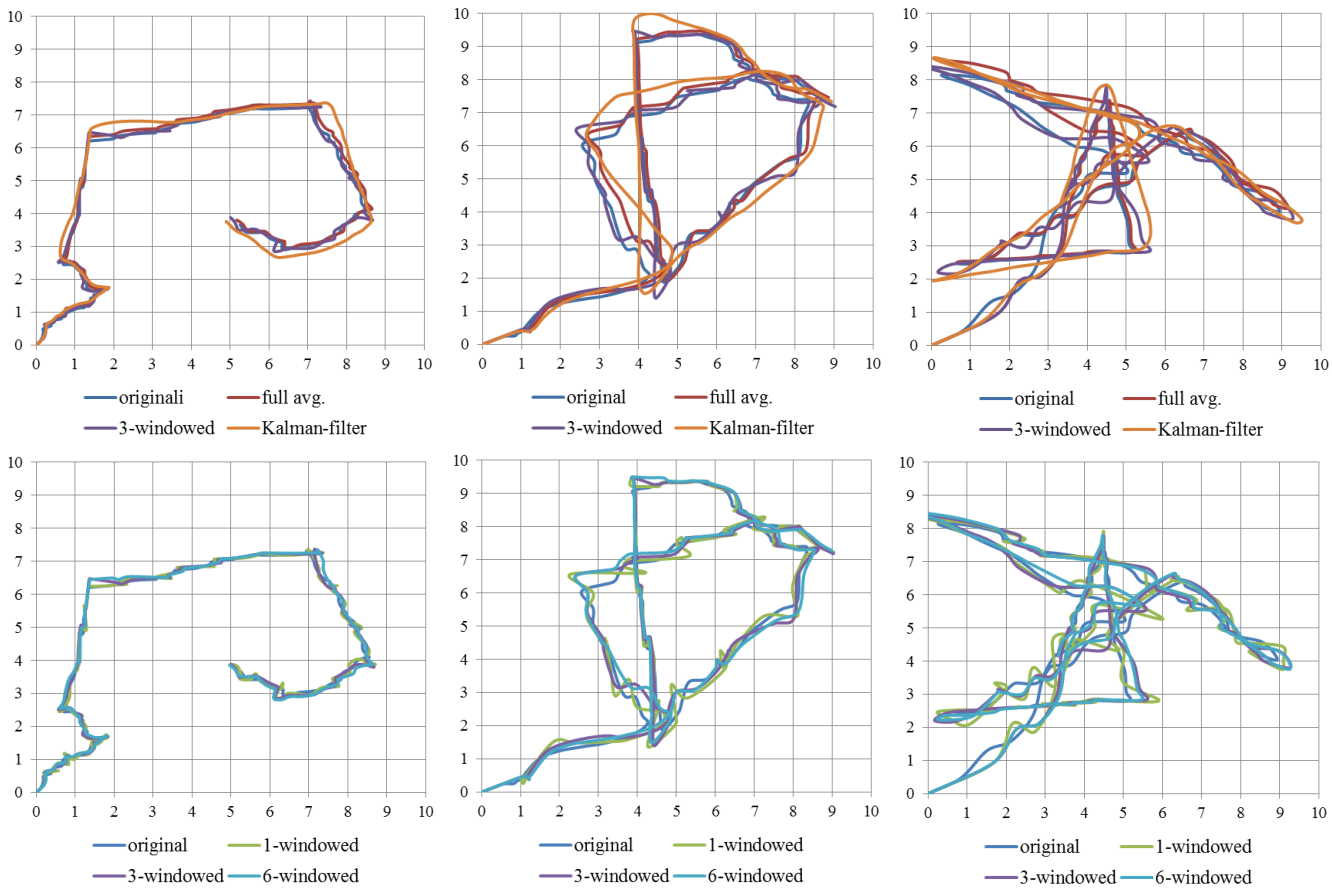

Fig. 11. Viewpoint prediction of *move01*, *move02* and *move03* trajectories

The average and maximum speed values were determined using the absolute values of the speed parameter, keeping it independent from its direction. Thus the minimum speed is 0 *CU/TU* in all three instances. The characteristics of the three different viewpoint movement series (*move01*, *move02* and *move03*) are presented in Table 2, where *move01* has the lowest and *move03* has the highest displacement behavior.

Table 2

Parameters of predefined viewpoint movements

	move01		***move02***		***move03***	
number of viewpoints	100 steps		100 steps		100 steps	
components	*x*	*y*	*x*	*y*	*x*	*y*
average speed (CU/TU)	0.028	0.25	0.042	0.047	0.066	0.052
maximum speed (CU/TU)	0.1	0.1	0.16	0.136	0.194	0.226

In a realistic scenario only the current viewpoint and the viewpoint history is known, however the knowledge of future perspectives are required to make optimal camera selection decision. In order to estimate future viewpoint positions different prediction methods can be used.

Viewpoint prediction comparison

We have implemented five viewpoint prediction algorithms and tested for all three movement series: averaging based on the entire history, 1-windowed averaging, 3-windowed averaging, 6-windowed averaging and Kalman-filter based prediction (Fig. 11.). We were looking for the most accurate viewpoint movement prediction method for further simulations by comparing the performance of the prediction methods. Comparison was done based on the following parameters:

1. *avg.*: average difference between measured and predicted data (lower the better)
2. *avg.(abs)*: average of absolute values of differences between measured and predicted data (lower the better)
3. *max.(abs)*: maximum value of absolute values of differences between measured and predicted data (lower the better)
4. *min.(abs)*: minimum value of absolute values of differences between measured and predicted data (lower the better)

Due to low viewpoint movement values, the 6-windowed averaging scheme achieved the best results in case of the first movement series (*move01*). Although

averaging based on the entire history gave more accurate values in the parameters of minimum and maximum difference, but it has the worst averaging absolute difference values. Our results show that the averaging window length is poorly correlating with the average values of estimation errors, e.g. the minimum value of 1-windowed prediction is the most accurate among all. Fig. 11. shows that Kalman-filter predicts smooth viewpoint path similarly to a long term trendline. In our FVV viewpoint prediction application Kalman-filter did not achieved an outstanding result, although the efficiency of prediction (average, minimum /maximum of prediction errors) is not extremely wrong. The prediction errors of different schemes for *move01* viewpoint trajectory are introduced in Table 3.

Table 3

Comparing viewpoint prediction algorithms used on *move01* trajectory

	full averaging	**1-windowed**	**3-windowed**	**6-windowed**	**Kalman-filter**
avg.	-0.07/-0.088	-0.05/-0.037	-0.05/-0.037	-0.05/-0.037	0.07/-0.038
avg. (abs)	0.070/0.088	0.139/0.124	0.134/0.122	0.128/0.121	0.159/0.147
max. (abs)	0.107/0.140	0.5/0.5	0.443/0.363	0.395/0.281	0.51/0.479
min. (abs)	0.023/0.032	0/0	0.003/0.006	0.001/0	0.001/0.004

The higher speed of viewpoints in case of *move02* trajectory amplified the measured differences between prediction methods (Fig. 11.). The 3-windowed and 6-windowed averaging methods achieved the best results in both minimum and maximum values of prediction errors. The prediction errors of full averaging and Kalman-filter based estimation had almost doubled in many cases, while window based averaging methods show a much slighter increase in the prediction error.

The prediction error statistics related to *move02* viewpoint trajectory are presented in Table 4.

Table 4

Comparing viewpoint prediction algorithms used on *move02* trajectory

	full averaging	1-windowed	3-windowed	6-windowed	Kalman-filter
avg.	-0.14/-0.148	-0.08/-0.073	-0.08/-0.769	-0.09/-0.081	-0.09/-0.103
avg. (abs)	0.149/0.148	0.208/0.234	0.203/0.227	0.193/0.220	0.263/0.284
max. (abs)	0.55/0.303	0.8/0.68	0.65/0.52	0.55/0.5	1.001/0.915
min. (abs)	0.051/0.033	0/0	0.01/0.01	0.006/0.003	0.022/0.001

In case of *move03* (Fig. 11.) viewpoint movement the differences of prediction method's efficiencies are even more apparent. The best results were achieved by the 3-windowed averaging method. We had observed that these results stay in midrange without showing extremely high or low prediction error values. Also, Kalman-filter showed the same „smoothing" effect as it did in cases of *move01* and *move02* viewpoint trajectories. The measured prediction error results of *move03* are shown in Table 5.

Table 5

Comparing viewpoint prediction algorithms used on *move03* trajectory

	full averaging	1-windowed	3-windowed	6-windowed	Kalman-filter
avg.	-0.15/-0.188	-0.03/-0.043	-0.04/-0.051	-0.05/-0.063	-0.107/-0.13
avg. (abs)	0.151/0.188	0.326/0.259	0.308/0.252	0.289/0.251	0.453/0.372
max. (abs)	0.84/0.7	0.97/1.13	0.84/0.853	0.84/0.7	1.296/1.017
min. (abs)	0.01/0.035	0.01/0	0.016/0.003	0.04/0.001	0.021/0.013

Based on obtained prediction error results of different schemes, we decided to keep only three of the methods to work further with: full average, 3-windowed average and Kalman-filter. The 1-windowed and 6-windowed averaging methods were eliminated due to the dichotomy of having a good average value, but a bad maximum value and vice versa.

Testing camera handovers

The main goal of the proposed camera switching schemes is to reduce the number of camera handovers, but on the other hand choose cameras that are close to the current FVV perspective in order to gain higher synthetized video quality. We have analyzed numerous scenarios and measured the performance of the introduced camera switching schemes (TCC, SA, MA) in case of different viewpoint trajectory behaviors (*move01*, *move02*, *move03*) and prediction methods. According to our expectations, the more accurate the predictive algorithm is, the less camera handover occurs during the viewpoint movement. The results of camera change frequency measurements are presented in the following figures (Fig. 12-14.).

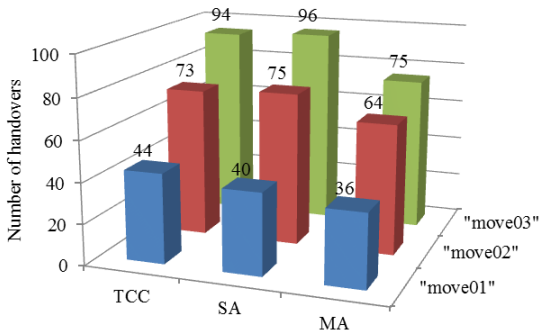

Fig. 12. The number of handovers using full averaging

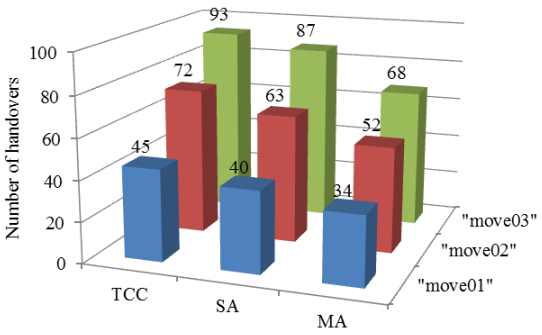

Fig. 13. The number of handovers using 3-windowed averaging

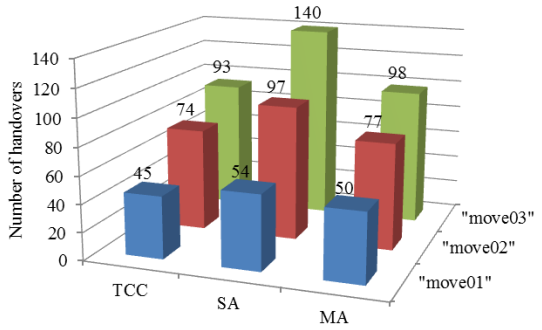

Fig. 14. The number of handovers using Kalman-filter

The obtained results show that in case of Kalman-filter and full averaging based viewpoint predictions the SA and TCC algorithms resulted higher number of camera changes. The reason is that in case of viewpoint synthesis process starvation (when the desired user viewpoint cannot be synthetized with the current camera set), the SA will probably choose the three closest cameras as new ones, which basically means three camera handovers. Although in case of TCC scheme two cameras can remain the same and only one needs to be replaced. The number of handovers can be kept low with the MA, but using Kalman-filter prediction it is not able to keep it lower than TCC.

According the simulation results, using 3-windowed averaging the number of handovers was decreased due to its more accurate prediction efficiency. We measured the best performance in case of MA, resulting an average decrease of 26% compared to TCC.

In the tested scenarios we have analyzed perspective trajectories containing 100 viewpoint coordinates. The number of camera handovers in each state of the *move02* viewpoint trajectory and using the 3-windowed prediction model are shown in Fig. 15. By using other viewpoint prediction methods, the number of overall camera switches was higher in all of the cases.

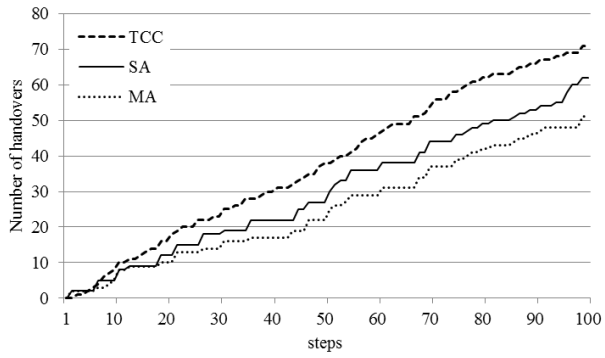

Fig. 15. Movement series *move02* with 3-windowed averaging

Examination of FVV synthesis process starvations

Due to viewpoint changes, the camera view synthesis process must continuously change the input camera streams without interrupting the playout. Starvation occurs when the desired user viewpoint cannot be synthetized with the currently streamed camera views and the required camera streams are still not available at FVV processor. We had tested starvation frequency for different camera switching schemes (TCC, SA, MA), viewpoint trajectory behavior (*move01*, *move02*, *move03*) and prediction methods (Fig. 16-18.). The results show that the best performance for all movement series was achieved using the 3-windowed averaging prediction scheme and the worst with Kalman-filter.

The efficiency of starvation reduction is effected by the combination of the prediction method and the speed of the viewpoint movement. The obtained results show that compared to TCC method, the MA camera selection scheme produced 46% less starvations, while SA reduced the starvation occurrence with 42% in average. The following figures show the number of FVV process starvations that causes playout interruptions while the missing camera stream is not delivered to the FVV synthesizer entity (Fig. 16-18.).

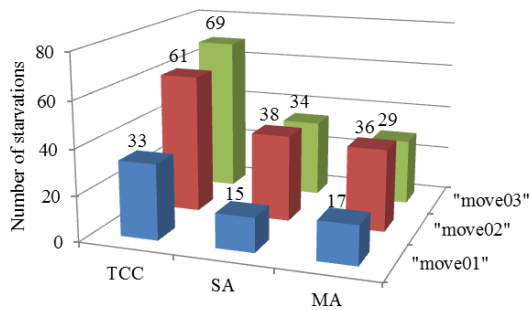

Fig. 16. The number of starvations using full averaging prediction

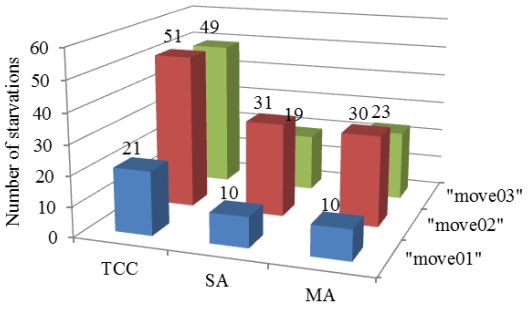

Fig. 17. The number of starvations using of 3-windowed averaging prediction

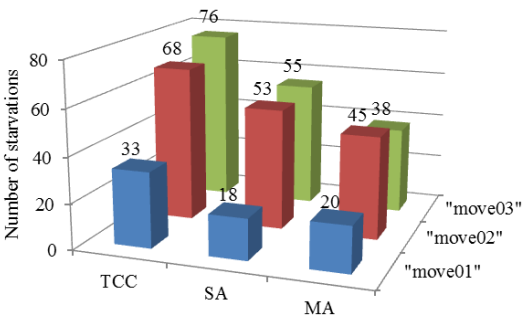

Fig. 18. The number of starvations using 6-windowed averaging prediction

Examination of video quality parameters

Generally, different video quality measurement metrics (e.g PSNR, VQM, SSIM) are used to evaluate the video quality by comparing the analyzed video to a reference video stream. In case of FVV, the virtual view is synthetized from two or three camera images, therefore no reference image can be used for video quality measurements. In our previous work we have shown that the distance between the desired viewpoint and the cameras used for the synthesis are strongly correlating to the rendered video quality. The best synthetized video quality can be achieved if the virtual view is generated using the closes camera images. In order to evaluate the performance of the proposed camera selection schemes, we used the average camera distance as a quality metric, which is a lower-the-better value.

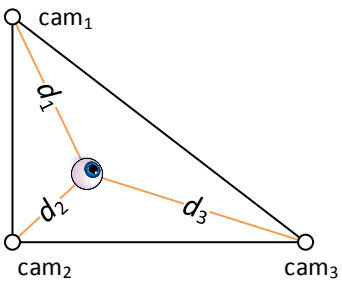

Fig. 19. Distance based quality metric

$$Q = \frac{d_1 + d_2 + d_3}{3} \tag{5}$$

According the nature of the viewpoint prediction algorithms, we expected TCC to perform better than the MA and the SA, because as further the selected cameras are, the lower the synthetized virtual view quality will be. As the method of 3-windowed averaging seemed to be the best viewpoint prediction method so far, we decided to test the qualitative parameter with it, using a variety of movement series and algorithms. The obtained results are presented in Fig.20.

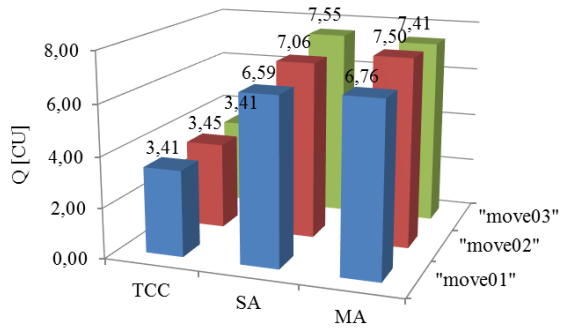

Fig. 20. The average of qualitative parameter with different combinations of algorithms and movements

The results of the SA camera handover algorithm showed an average increase of 106% in terms of the *Q* metric (average camera distance), while the Q quality metric of MA was 110% compared to TCC. Although, MA and SA schemes were selecting camera from more than two times higher distance, the frequency of FVV synthesis process starvations was the half compared to TCC method. In our opinion the starvation rate has higher impact on the experienced user quality than the view synthesis distortion due to higher camera distances.

The measured *Q* parameter is continuously changing depending on the current viewpoint position in the trajectory. The graph below shows the measured values of the qualitative parameter for each viewpoint position in case of *move02* trajectory (Fig. 21).

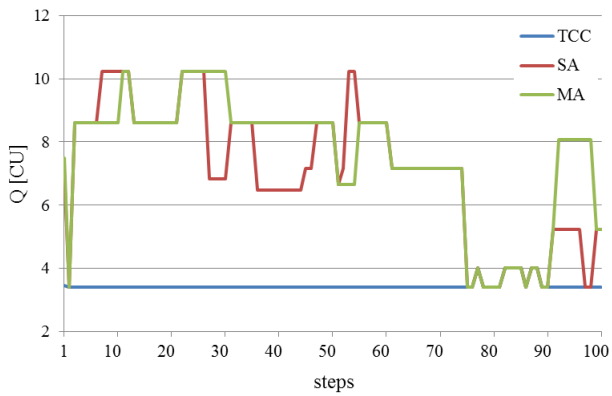

Fig. 21. Movement series *move02*

5. Conclusions

Free-viewpoint video is a promising approach that allows users to control their viewpoint and generate virtual views from any desired perspective. In order to change the viewpoint in a plane, each viewpoint must be synthetized from at least three high bitrate color cameras and corresponding depth sequences that are used to capture the scene from different locations. Continuous change of the viewpoint can cause frequent camera handovers that can lead to the starvation of the view synthesis process. Starvation occurs when the desired user viewpoint cannot be synthetized with the currently streamed camera views causing interruption in the playout. In order to minimize the number of camera handover occurrences, we proposed novel camera switching schemes based on viewpoint prediction. Our aim was to find the tradeoff between the image quality and the camera handover frequency, therefore three different camera selection algorithms were presented and analyzed by simulations. According to the obtained results, SA and MA were performing similarly showing 42-46% decrease in the number of camera switches. The price of the good performance was that each algorithm selected cameras that are located further from the current viewpoint. Thus, the quality of the synthetized images were lower, however it can be proved that reduced playout interruption frequency has higher impact on the experienced user quality than the view synthesis distortion due to higher camera distances.

Acknowledgement

The authors are grateful for the support of the Hungarian Academy of Sciences through the Bolyai János Research Fellowship.

References

[1] L Chiariglione, Cs. A. Szabó, “Multimedia Communications: Technologies, Services, Perspectives, Part II: Applications, Services and Future Directions”, Infocommunications Journal VI:(3) pp. 51-59., 2014

[2] M. Levoy and P. Hanrahan., "Light field rendering", Computer Graphics, Proceedings. SIGGRAPH96, August 1996

[3] Christoph Fehn, “Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV”, Proc. of SPIE, Vol. 5291, Stereoscopic Displays and Virtual Reality Systems, May 2004, pp. 93-104

[4] Aljoscha Smolic, “3D video and free viewpoint video-From capture to display”, Pattern Recognition Vol. 44 (9), pp. 1958-1968., September 2011

[5] Claudia Kuster, Tiberiu Popa, Christopher Zach, Craig Gotsman, Markus Gross, Peter Eisert, Joachim Hornegger, and Konrad Polthier, “Freecam: A hybrid camera system for interactive free-viewpoint video,” in Proceedings of vision, modeling, and visualization (VMV), 2011.

[6] A. Ishikawa, M. Panahpour Tehrani, S. Naito, S. Sakazawa A. Koike, “Free Viewpoint Video Generation for Walk-through Experience Using Image-based Rendering”, 16th ACM international conference on Multimedia, pp. 1007-1008., 2008

[7] C. Kuster, T. Popa, C. Zach, C. Gotsman, and M. Gross, "FreeCam: A Hybrid Camera System for Interactive Free-Viewpoint Video," Proc. Int'l Workshop Vision, Modeling, and Visualization, 2011.

[8] K. Mueller, P. Merkle, A. Smolic, and T.Wiegand, “Multiview coding using AVC,” MPEG2006/m12945, 75th MPEG meeting, Bangkok, Thailand, Jan. 2006

[9] P. Merkle, A. Smolic, K. Mueller, T. Wiegand, “Efficient prediction structures for multiview video coding”, IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on Multiview Video Coding and 3DTV, 2007

[10] Merkle P, Morvan Y, Smolic A, Farin D, Muller K, Wiegand T, “The effects of multiview depth video compression on multiview rendering”, Signal Processing: Image Communication 2009, 24(1):73–88.

[11] Guan-Ming Su, Yu-Chi Lai, Andres Kwasinski, and Haohong Wang, “3D video communications: Challenges and opportunities”, nternational Journal of Communication Systems, Vol. 24/10, pp. 1261-1281., October 2011

[12] ISO/IEC JTC 1/SC 29/WG 11. Committee Draft of ISO/IEC 23002-3 Auxiliary Video Data Representations. WG 11 Doc. N8038. Montreux, Switzerland, April 2006.

[13] Christoph Fehn, “Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV”, Proc. of SPIE, Vol. 5291, Stereoscopic Displays and Virtual Reality Systems, May 2004, pp. 93-104

[14] Fehn C. “3D-TV using depth-image-based rendering (DIBR)”, Proceedings of Picture Coding Symposium, San Francisco, CA, U.S.A., December 2004.

[15] M. Domański, M. Gotfryd, and K. Wegner, "View synthesis for multiview video transmission," in The 2009 International Conference on Image Processing, Computer Vision, and Pattern Recognition, Las Vegas, USA, 2009, pp. 1-4.

[16] S. Jo, D. Lee, Y. Kim, Ch. Yoo, “Development of a simple viewpoint video system”, IEEE Int. Conf. Multimedia and Expo, Hannover, June 2008, pp. 1577-1580.

[17] H. Kimata, S. Shimizu, Y. Kunita, M. Isogai, K. Kamikura, Y. Yashima, “Real-time MVC viewer for free viewpoint navigation”, IEEE Int. Conf. Multimedia

[18] J. Starck, J. Kilner, and A. Hilton, “A Free-Viewpoint Video Renderer”, Journal of Graphics, GPU and Game Tools, 14(3):57-72, Jan. 2009.

[19] Zefeng Ni; Dong Tian; Bhagavathy, S.; Llach, J.; Manjunath, B.S., "Improving the quality of depth image based rendering for 3D Video systems," Conf. on Image Processing (ICIP), 2009, 7-10 Nov. 2009

[20] Árpád Huszák, "Predictive Multicast Group Management for Free Viewpoint Video Streaming", International Conference on Telecommunications and Multimedia (TEMU 2014), ISBN 978-1-4799-3199-6, Heraklion, Greece, 28-30 July 2014

[21] Árpád Huszák, "Optimization of Distributed Free-viewpoint Video Synthesis", 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), Budapest, Hungary, July 2-4 2014

[22] Zhun Han; Qionghai Dai, "A New Scalable Free Viewpoint Video Streaming System Over IP Network," Acoustics, Speech and Signal Processing, ICASSP 2007, pp.II-773,II-776, 15-20 April 2007

[23] Goran Petrovicand Peter H. N. de With, “Near-future Streaming Framework for 3D-TV Applications”, ICME2006

[24] Gürler, C.G.; Görkemli, B.; Saygili, G.; Tekalp, A.M., "Flexible Transport of 3-D Video Over Networks," Proceedings of the IEEE , vol.99, no.4, pp.694-707, April 2011

[25] C. De Raffaele, C.J. Debono, "A Comparison of the Performance of Prediction Techniques in Curtailing Uplink Transmission and Energy Requirements in Mobile Free-Viewpoint Video Applications," International Journal on Advances in Telecommunications, vol. 4, no. 1 & 2, September 2011, pp. 1 - 11.

[26] E. Kurutepe, A. Aksay, C. Bilen, C. G. Gurler, T. Sikora, G. B. Akar, and A. M. Tekalp, “A standards-based, flexible, end-to-end multi-view video streaming architecture”, in Proc. Int. Packet Video Workshop, Lausanne, Switzerland, Nov. 2007, pp. 302–307