

# Browsing Sport Content Through an Interactive H.264 Streaming Session

Iván Alén Fernández\*, Fan Chen\*, Fabien Lavigne<sup>†</sup>, Xavier Desurmont<sup>‡</sup> and Christophe De Vleeschouwer\*

\* *TELE, Université Catholique de Louvain-la-Neuve, Belgium*

*Email: {ivan.alen,fan.chen,christophe.devleeschouwer}@uclouvain.be*

<sup>†</sup>*Département TCTS, Université de Mons, Belgium*

*Email: Fabien.Lavigne@umons.ac.be*

<sup>‡</sup>*Image department, Multitel, Mons, Belgium*

*Email: desurmont@multitel.be*

**Abstract**—This paper builds on an interactive streaming architecture that supports both user feedback interpretation, and temporal juxtaposition of multiple video bitstreams in a single streaming session. As an original contribution, these functionalities can be exploited to offer improved viewing experience, when accessing football content through individual and potentially bandwidth constrained connections. Starting from a conventional broadcasted content, our system automatically splits the native content into non-overlapping and semantically consistent segments. Each segment is then divided into shots, based on conventional view boundary detection. Shots are finally splitted in small clips. These clips support our browsing capabilities during the whole playback in a temporally consistent way. Multiple versions are automatically created to render each clip. Versioning depends on the view type of the initial shot, and typically corresponds to the generation of zoomed in and spatially or temporally subsampled video streams. Clips are encoded independently so that the server can decide on the fly the version to send as a function of the semantic relevance of the segments (in a user-transparent basis, as inferred from video analysis or metadata) and the interactive user requests. Replaying certain game actions is also offered upon request. The streaming is automatically switched to the requested event. Later, the playback is resumed without any offset. The capabilities of our system rely on the H.264/AVC standard. We use soccer videos to validate our framework in subjective experiments showing the feasibility and relevance of our system.

**Keywords**—interactive streaming, browsing capabilities, clip segmentation, view type detection, H.264/AVC.

## I. INTRODUCTION

Mobile streaming service of *on demand* video content through cell phone is becoming one of the highlights of new value-added mobile services and it is commonly related to sports content. Lately, the number of applications for this purpose developed on smart phones increased dramatically and more and more multimedia content is proposed on these devices. In addition to this, video streaming is a technology that has gained large public attention over the last few years. The latest MPEG-4 standard for video compression, H.264, has been the subject of many studies in the field of streaming [1][2][3]. In our framework, we focus on interactive functionalities and propose a streaming architecture that offers browsing capabilities to any client using an H.264/AVC compliant player. Our system builds on an application, which targets adapting TV sport broadcasting content for mobile terminals.

Video analysis and processing have been largely investigated to identify regions or periods of interest in sport events broadcasting context, as reviewed by [4]. This knowledge is typically used to create zoomed-in content, more suited to low bandwidth (and thus low resolution) accesses. Several proposals to create summaries of sport

events have also been developed based on semantic knowledge about the scene content, as proposed by [5]. Together, those Region of Interest (RoI) detection and video summarization mechanisms can a priori support the generation of valuable content for mobile accesses. This is because mobile users generally want to access summarized versions of the content at a small resolution. However, such automatic generation of zoomed-in summaries strongly relies on the accuracy and reliability of the knowledge available about the content of the scene. To deal with the imperfections of practical and real-life fully automatic video analysis systems, our interactive streaming architecture allows the end-user to switch from the native and spatially sub-sampled content to a zoomed-in and/or temporally compacted mode, depending on his/her viewing preferences and on the content of the scene. Knowledge about video view-type and hot spots (goals, etc) of the game is also exploited to reduce the interaction load on the user. Typically, zoomed-in is only considered for far views, and fast-forward mode is systematically disabled when entering a semantically important period of the game.

Our framework proposes to allow the end-user to decide about the version of the content (s)he would like to visualize. Hence, the automatic system produces one or several versions of the content, and the user gets the opportunity to switch interactively between the versions. Typically, the (automatically) produced low-resolution versions include a sub-sampled version of the native video, plus one zoomed-in (and cropped) version for far camera views, focusing on the region-of-interest detected in the original high-resolution video. An alternative sub-sampled version in fast forward speed mode is also provided during the whole sequence. In our framework, replays of hot spots actions are also proposed to the user just by switching the playback to a highlight/prominent action displayed formerly. In practice, client-transparent switching between versions is enabled by splitting each produced video in short clips that are temporally pipelined by the server, based on user's requests. From the client's perspective, everything happens as if a single pre-encoded video was streamed by the server. Therefore, the global viewing experience is drastically improved by using the featured capabilities. In the following, the set of clips resulting from this versioning process and the associated organizational information are referred to as enriched or enhanced content.

In the context of real-life and large scale deployment of the system, one could imagine to monitor how the end-users actually visualize the content, so as to cancel the non-accessed versions from the available list of clips. This mechanism has not been

considered in our experiments, but would be easy to implement on top of our architecture.

The remaining of the paper is organized as follows: Section II presents the proposed architecture for interactive video streaming, through client-transparent temporal concatenation of pre-encoded video clips. In Section III, we describe how the soccer game is divided into clips based on the monitoring of production actions, and how several versions are generated to offer multiple rendering opportunities for each clip. In Section IV, we introduce the interactive commands that are offered to the client to improve his/her viewing experience and the strategy that is followed to provide the browsing capabilities based on the semantic segmentation of the game. Finally, Section V presents some qualitative results, while Section VI concludes.

## II. INTERACTIVE BROWSING ARCHITECTURE

The main objectives of our architecture is to offer interactivity to any client of a video streaming session when using an H.264/AVC compliant player, based on the content pipelining feature. As depicted in Figure 1 the communication is established with the client through the Real Time Streaming Protocol (RTSP).

### A. Architecture of the Streaming Server

The architecture proposed in a previous work by the authors [6], is now extended to a real scenario by developing the session control to offer the new browsing capabilities using its main feature: the temporal content pipelining. For this purpose the RTSP control messages are also extended and the segmentation (and metadata associated to it) is managed in a temporally consistent way.

The architecture on the server side is composed of 3 main components: the enhanced content creation unit, the streaming server and the session control module.

1) *The Enhanced Content Creation Unit* fills the Video Content Database, without actively taking part afterwards in the streaming system. Its purpose is threefold:

- It analyzes the TV like video content to identify regions-of-interest and produce several multiview replay or zoomed-in versions of the content, as described in Section III-B2.
- It divides the video sequences in small pieces that are encoded based on H.264 according to the requirements explained in sections II-B and III-A.
- It generates the metadata (shown in Section II-C) that are required to model and define the interactive playing options associated to the different clips. Therefore, the metadata information is used by the session control to monitor the streaming session in response to the interactive user requests.

2) *The Streaming Server Module* is the core of the system, which supports client-transparent interactive streaming through on-the-fly content pipelining. Client-transparent temporal content pipelining allows the server to stream multiple successive video streams in a single session, without negotiating with the client the establishment of a new streaming session. Hence, with this feature the server is able to change the streamed content while maintaining a unique output stream and keeping the existing session uninterrupted. As a result, both a temporal and computational gain are achieved as the client does not need to establish more than one streaming session. The streaming server delivers all the data content through the Real-time Transport Protocol (RTP).

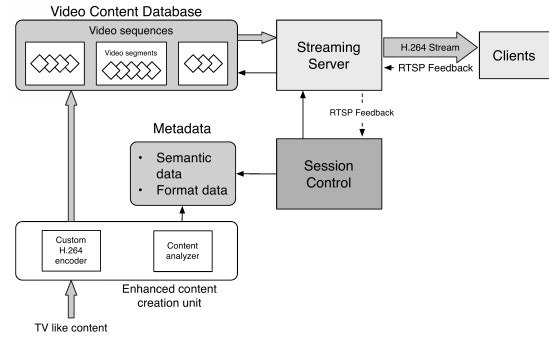


Figure 1. Diagram of the architecture's workflow

3) *The Session Control Module* determines, at every moment, which video clip has to be transmitted next. This unit consequently decides the video clips that are concatenated based on requests from the client, and on versions offered by the enhanced content creation unit. Therefore, the session control is an essential part of the system, as it monitors almost any information flowing through the system.

### B. Temporal pipelining

Temporal content pipelining is the technique that allows a streaming server to juxtapose multiple streams in a single continuous sequence, so that multiple streams can be forwarded to the client through a unique streaming session. The key for implementing this functionality is the session control module using the advanced features of the H.264 codec [7], regarding the encoding parameters transmission.

The H.264 standard defines two kinds of parameter sets: sequence parameter sets (*SPS*) and picture parameter sets (*PPS*). The first applies to a wide range of images, while the latter only to particular pictures. Every Network Adaptation Layer (*NAL*) unit containing data information (*VLC NAL* unit with *VLC* standing for Video Coding Layer), includes in its header a parameter linking it to a *PPS*, which in turn links to a specific *SPS*. In our proposed framework, the *SPS* updates are always necessarily sent between two pipelined segments. In fact, all clips are encoded independently from each others. Since the first *NAL* unit of an H.264 segment always contains the *SPS* and the *PPS*, multiple sequences can be transmitted consecutively without any interruption, and the output is still compliant to the H.264 standard. On the client's side, a unique sequence is received, which however, is built step by step by the server.

### C. Session Control and Metadata

The session control processes the user's feedback and uses the metadata associated to the clips, to decide the next clip to be delivered. The metadata information is generated by the enhanced content creation and is stored within a Extensible Markup Language (XML) file.

Two different kinds of temporal relationships between clips are introduced, as depicted in Figure 2. Case A typically corresponds to an optional inclusion of a replay within the stream. The sequence playback is resumed after the additional content without any offset. The same relationship can be considered if *target advertising* is inserted in the stream according to the user preferences. In contrast, case B considers contending versions, and only one version is

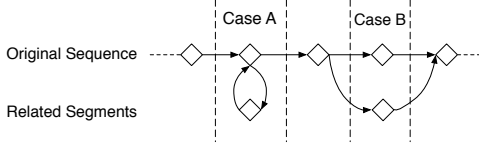


Figure 2. Metadata considered structures

actually included in the output stream. As an example, several video resolutions (zooming) and fast-forward/regular speed mode are taken into account.

When temporal continuity is required, switching can only occur at the intersection between two consecutive clips. Those instants are depicted with vertical dashed lines in Figure 2. For this reason, the sequences have to be divided in very small clips as each clip has to be completely delivered before switching. Otherwise the browsing capabilities would only be offered in a coarse granularity basis.

In the case that temporal continuity is not required, as happens when the user wants to skip some non-relevant content, any buffered data in the server is discarded, so as to start reading the new clip file as soon as possible, thereby reducing to the minimum the overall latency associated to the switching mechanism. Like in the previous cases, the playback proceeds without causing any decoding error and the streaming behavior is not damaged, performing the switching flawlessly.

#### D. Interaction with the Client's Video Player

The system's interactivity relies on the RTSP commands that are exchanged between the server and the client. The user must be able to send a switching command, which induces a system response according to its content. The browsing features are then triggered by sending the appropriate request to the server.

A standard RTSP message is used by the client player to communicate its feedback. The considered RTSP command in our architecture is *OPTIONS*, as described in [8]. Combined with the header *Require*, it provides an efficient and simple way to signal user's feedback during the transmission. A specific value in the field of this header such as "*Switching-feature*", directly associates the RTSP command with the browsing capabilities of our server. A new line in the header, starting like "*Switching-Parameter:* " signalizes and conveys the different possible requests of the user (zooming, replay or fast forward mode). The mentioned interactive requests are associated one-by-one to new-functional buttons of the player's interface. These buttons consequently trigger a RTSP command from the user side when they are pressed.

### III. CLIPS DEFINITION AND VERSIONING

This section explains how a broadcasted video content is split into non-overlapping clips. It then associates a discrete set of versions to each clip, depending on their view type. In Section III-A, the clip definition relies on video shot extraction. A shot is defined as a portion of the broadcasted video that has been produced with constant or smoothly evolving camera parameters.

The reason why we define clips based on video shots is because, as explained in Section III-B1, the versions that are available for a portion of video directly depend on the characteristics of the shot (replay, far or close view, etc). Hence, switching between versions should be allowed between shots, meaning that a boundary between

shots should also define a boundary between clips. By first dividing the video into a sequence of shots and dealing each one with different strategies according to the view type provides a reasonable and computationally efficient base for further processing.

#### A. Clips Definition based on Shot Boundary Detection

On the one hand, as explained above, clips should be at least delimited by shot boundaries. On the other hand, by definition (see Figure 2), the switching operations are limited to the border of the clips. Hence, reducing the size of the clips below the one of the shots is required to offer the switching capabilities that are available into each view type shot. In our implementation, the segmentation of the video in clips is thus based on the monitoring of the production actions using shot boundary detection and then, a finer segmentation in clips. The last segmentation, in principle as fine as possible, is based on a trade-off between the switching capabilities and the streaming performance, as explained in Section V.

Compared to general videos, sport team videos usually have well-organized structures of shots, based on several elemental view types of cameraworks. For each shot, the cameraman can give either a far view for describing the complexity of the team sports, show more details of the action in a local area with a medium view, or zoom into a close-up view for enhancing the emotional involvement of the audience. Furthermore, sudden view switching during the evolving of a tight game action is suppressed in order to avoid the distraction of audience attention from the current game.

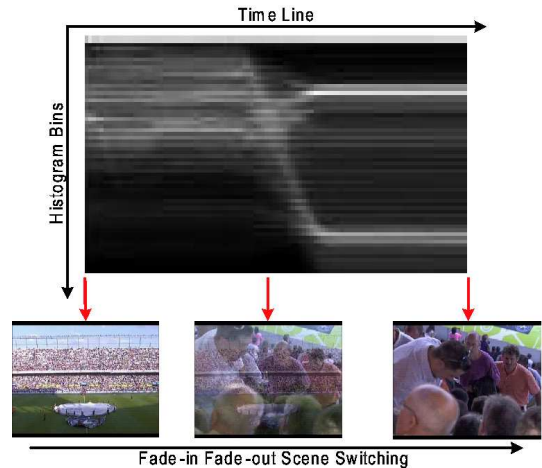


Figure 3. Histogram transition around a typical fade-in fade-out shot boundary.

A difficult problem in shot boundary detection is to deal with special effects supporting smooth transition between two scenes, e.g., fade-in fade-out. Using histogram-features as in [9], we notice that the histogram is gradually varying along with this smooth scene switching, as shown in Figure 3. Hence, shot-boundary detectors based on difference of histograms between two successive frames are not efficient in this case. Therefore, an improved shot-boundary detector is proposed based on the difference between the average histogram of its left and right neighborhoods as displayed in Figure 4.

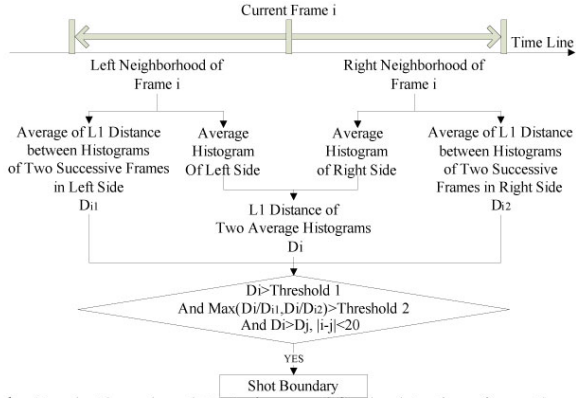


Figure 4. Shot boundary detector improved for the detection of smooth varying shot boundaries.

### B. Clips Versioning

A similar analysis for view type, as the one proposed next, was used in [10] to detect exciting events (i.e., game parts with both close-up scenes and replays). In our framework, the view types are classified in different groups: replays, non-grass close-up views and close, medium or far grass view camera. At the end, far views are computed in order to obtain an alternative zoomed-in version that is stored in the enhanced content creation unit.

1) *View Type Definition*: The two major methods for detecting replays are detection of replay-logos, and detection of slow-motions [11]. Although replay-logos are producer-specific, this approach is the one followed because it is easier and more accurate to detect replay logos than to detect slow-motions, due to the fact that the view angle in the replay might change a lot from the normal play.

A simple but efficient method to detect view types in soccer context has been proposed by [12]. For scenes having a large portion of grass area, the non-grass blobs within the grass area reflect objects in the soccer field. The basic idea is to evaluate the ratio between grass area and non-grass area in each subdivision of the scene to identify the view type. Scenes with few or even without grass region could be either a public view or a game view. A game view without grass area usually gives a quite close view of the scene, even though it is a medium view, e.g., a scene focusing on the foot actions of the players. Therefore, it is safe to treat all these scenes without grass area as close-up views. Based on the method in [12], we further preclassify the scene type according to the percentage of overall grass ratio, and use support vector machine to replace the linear classifier to achieve a better performance, as shown in Figure 5. Extra robustness is achieved by running the view-type classification over all frames within the shot and making the final decision by taking a majority vote.

2) *Zoom-in for Far View*: The zoom-in algorithm is applied only if the camera view type is far and there is no replay in the concerned shot. In a soccer scene with far view, RoI is usually unique and well defined. The ball is the central element of the scene. Indeed, players react according to its position. Consequently, detecting the ball in the video generally provides focusing on the RoI of the scene. Based on the work published already by the authors in [13], a ball detector localizes the interesting area of the scene. Then, the current frame is resized and centred on the ball, taking into account some parameters, such as the ball position and

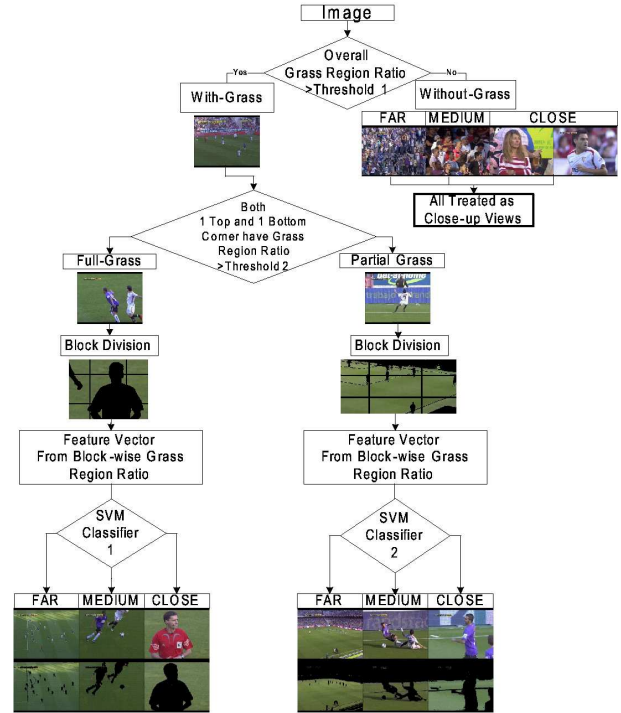


Figure 5. View Type Classification based on Grass Region Ratio.

its speed. Moreover, if a shot-boundary is detected, the parameters of the zooming frame are reinitialized. In the Figure 6 we show an example of our framework. The purpose of this algorithm is to adapt the size of a football video match extracted from TV broadcasting to a small device screen. The zoomed-in sequences are offered to the user as an alternative to the native sub-sampled sequence replacing the original segments upon request.



Figure 6. Original and processed zoom versions of the same frame.

## IV. SWITCHING CONTROL BASED ON SEMANTIC SEGMENTATION

This section presents the browsing capabilities provided by our system based on version switching. Zoom-in versions of far view shots, replays of highlight moments of the match, and fast forward mode are proposed as alternatives to the regular mode.

### A. Definition of Segments and Interaction Strategy

Section III has described how multiple versions can be generated automatically for each shot of the broadcasted video. Furthermore, we have explained how a shot is divided into clips to support switching between versions within a shot. In this section, we

explain how those switching opportunities are exploited in practice. The purpose is twofold. On the one hand, the switching control strategy should offer personalization capabilities to the user. On the other hand, it should be defined in a way that limits the load on the end-user, i.e. we do not want that the user ends up in controlling the switching instead of enjoying the content.

To achieve a reasonable trade-off between load and flexibility of control, we introduce the notion of semantically meaningful segment. It is defined by the set of consecutive clips that describe a single action of the game. All the video segments are thus independent and self-contained. The interaction strategy is based on the definition and semantic relevance of these segments. Specifically, the segments are divided in two main groups: highlighted segments and non-highlighted segments. The first ones contain the crucial actions of the soccer game, meanwhile the latest include portions of the match that have less relevance. The level of relevance of a segment directly affects the switching mechanisms. In particular, the fast-forward mode is automatically interrupted when starting an important action of the game.

Figure 7 presents the interaction strategy supported by our framework. In the figure three segments of the match are represented, and the one in the middle contains a highlight action. The browsing features are described next:

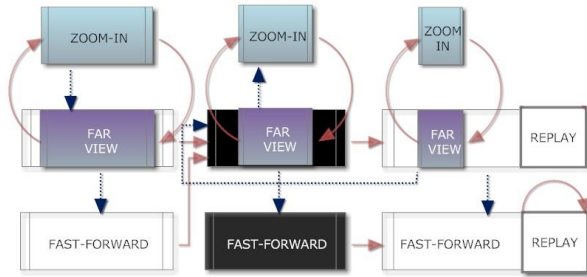


Figure 7. Switching control strategy. Dashed arrows represent potential requests from the user, while continuous arrows depict automatic connections between versions based on the interaction strategy. The central segment corresponds to an important action of the match.

1) *The fast forward version* is available for all the segments of the match, and the user can request to switch from the regular mode to the fast-forward mode at whatever point by pressing a dedicated button on the player interface. By pressing again the same button the regular mode can be recovered when desired. In both cases temporal consistency is preserved during the streaming. It is remarkable that every time a new highlight segment starts, the playback is automatically switched to the regular mode independently of the ongoing speed of the playback. The objective is to allow the user to watch very fast the parts of the game that are not relevant under his/her request, but also catching back his/her attention when an important action is close to happen. Again, the user has the possibility to switch to fast-forward mode when he/she is not interested in a certain gameplay even if this has been categorized as important. Between two non-relevant segments the regular mode is not launched if fast-forwarding is activated. To provide a better user experience, our strategy has also considered to skip the replay clips of the segments that are played by the user in fast forward mode, as shown in the Figure 7 in the last segment.

2) *Zoomed-in versions* are available for the far view clips. Again,

the user can switch to the zoom-in mode or switch back to the regular mode by properly interacting with the player interface. The zoomed-in version is the one proposed by default to the viewer at the beginning of every far view shot of the camera. It may happen that at a certain moment the automatically generated zoom-in version is not well centred in the RoI or that the viewer wants to watch the position of certain players that are not close to the ball. This is the reason why we give the viewer the faculty to decide the mode he/she considers convenient to receive. By default, the purpose of the zoom-in version is to replace the far-view clip of the original segment adapting the view to a small-screen device. At the end of every zoomed-in clip, the playback in regular mode continues with a new clip, the one right after the far-view. Note that when the fast forward mode is active the zoomed version of the video is not available as the user is not interested to focus in the current gameplay. Therefore, zoomed-in and fast-forward features are completely decoupled.

3) *Replay of certain segments of the match*: It may happen that the user is really interested in one of the segments of the match. Even if this segment includes replay clips, the user can be interested to see it several times. This is typically the case for a beautiful goal or a penalty action. Therefore, the user can request at the beginning of every new segment the replay of the previous one by pressing a dedicated button. After the repeated segment is displayed, the playback of the current segment where the replay was requested is recovered without any offset. The user can request this replay multiple times. In Figure 7 is shown this possible request from the viewer at the beginning of the last segment, as the previous one contains a highlight moment of the game. Note that the opportunity to replay a non-relevant segment is also provided and protects the viewer from missing an action when displaying the segment in fast-forward mode and suddenly realizing that (s)he has missed a detail (s)he wants to focus back.

### B. Automatic definition of semantically meaningful segments

A video segment is defined as a period covering several successive clips closely related in terms of story semantic, e.g., clips for an attacking action in football, including both a free-kick and its consequence. Proposed by the authors in [14], a general diagram of state transition consists in one round of offense/ defense as described in Figure 8. The diagram contains the essential structure of a segment and the different view types (clips) that are included. Each segment usually starts with a close view for highlighting the player who kicks off. Then, the offensive side makes trials of score after several passing actions, rendered through far or medium views that are the major part of the segment. This trial ends up in one of three possible results: being intercepted, scoring, or being an opportunity.

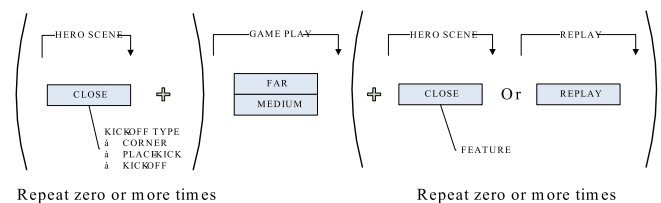


Figure 8. General structure of a gameplay and view types.

After the key event is finished, some close-up shots might be given to raise the emotional level. According to the importance of the corresponding event, replay clips might also be appended. Close view, medium view and replay are all optional. The state chain from the action-start to one of the results is regarded as a semantically complete segment, and based on this structure, the video is divided into a series of segments. Before a new round, exceptional actions might happen, which include foul, medical assistance, and player exchange. These actions are separated in the state graph from the main action as individual segments.

State transition motivates scene switching, and is thus reflected in the production actions. This observation is exploited to segment the video based on the monitoring of production actions, instead of (complex) semantic scene analysis tools. Although there are some complex cases, e.g., the offensive side tries many times of shooting, our segmentation rule is still applicable because the producer will not switch the view type during those periods due to the tightness of match. Hence, a boundary between two segments will always be associated to a boundary between two clips.

We assume that manual annotations are provided at production time to identify the most interesting actions in terms of game and emotional relevance. Such assumption is in accordance with the current practice in production rooms, as long as the annotations are only provided for the key actions of the game, in the form of a star rating system. Our approach follows this practical use case. Hence, each segment is assumed to be characterized by a set of annotations. Therefore, highlight actions, such as a goal or a red car, are related to the segment where they occur. A possible empty set of annotations for a particular segment would mean that is not essential. As a result, the segments are classified in two groups: relevant and non-relevant segments. The interaction strategy presented in the previous section is based on this division. More complex analysis based on audio-only and video-only analysis perspective, or more generally, as a multimodal problem [15], could be used to automatically detect outstanding events in the game. By using them, manual annotations would not be longer required.

## V. RESULTS AND VALIDATION

The streaming abilities are implemented using the liveMedia library [16] that has been extended to deliver H.264 encoded files, including the advanced feature for temporal content pipelining through the session control and the enhanced content creation unit. Our simulations have revealed that the fact that the video sequence is segmented in small pieces, as described in Section III-A, does not penalize the fluency of the streaming playback. On the server side, although clips have to be pipelined dynamically in the transmission buffer, the processing load is not dramatically increased, and the rhythm of delivery of RTP packets is preserved.

However, slight bitrate cost and some constraints are applied over the encoder H.264 and the subsequent NAL units, in order to enable adaptive streaming and video content segmentation:

1) The overall compression speed is clearly damaged as the encoding process of every sequence is divided in the multiple segments and several alternative versions are provided. Nevertheless, the scenarios we consider are based on *on demand* video content. Hence, all the segments are preprocessed and included in the video database *in advance*, and because of this, the performance of the global system is not damaged.

2) Every new clip has to start with a new Instantaneous Decoding Refresh (*IDR*) frame, penalizing the encoding flexibility. Therefore, the segmentation in multiple pieces of every sequence constraints the maximum size of the *GOP* (Group of Pictures) to the size of the encoded clips. Moreover, bitrate overhead is resulting from the use of *IDR refresh* frames. For this reason, a trade-off between the time of the system's response to the user's feedback, and the size of the clip has to be achieved, as every clip has to be completely delivered before starting to send the new one (due to the constraint of switching between versions in a temporally consistent way). If the clips are short, the system switches the playback very fast independently of the instant when the user's request is received by the server. However, the penalty in terms of bitrate increases when the clip size decreases (*GOP* is also small increasing the bitrate). The opposite result occurs if the clips are longer. In our simulations we used sequences encoded at 25 *fps* and clip segmentation approximately every 15 frames. On the one hand, using 1 *GOP* per clip, a *GOP* of 15 frames is good enough in order not to penalize the global bitrate. The global loss in quality in PSNR in the luminance and chroma is less than 0.5 dBs respect to encoding the same sequence without the *GOP* constraint across several bitrate configurations (as depicted in Figure 9). On the other hand, the maximum latency in the server due to the clip segmentation is less than 700 milliseconds, as in the worst of the cases, the server has just sent the first frame of a new clip when receiving a typical request to switch the version. This delay is a good approach as depending on the Round Trip Time (*RTT*) of the wireless network and the preroll buffer of the player, the minimal delay is already in the order of 2-3 seconds. Algorithms for adaptive video streaming could be used in order to minimize the latency due to the clip segmentation, just by decreasing the delivery time between frames belonging to the same clip and increasing the delivery time before starting a new one. In the fast-forward mode, by using the same granularity of clip segmentation, the maximum latency is already divided by the acceleration factor.

The cost of the restriction is also low when measuring the loss of quality with other techniques such as Structural similarity (SSIM). In this case, when handling very low bitrates (150-600 kbps) the loss of similarity can drop until 0,002 meanwhile for higher bitrates (1200-2000 kbps) this difference is lower than 0,0005.

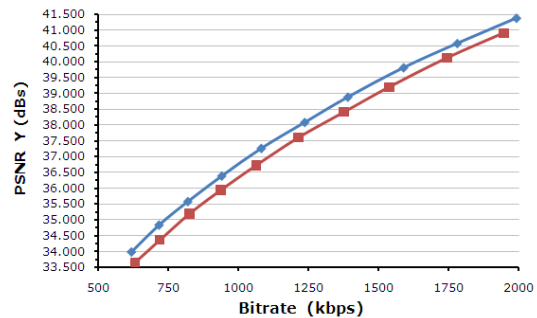


Figure 9. Video quality comparison in the luminance component when applying or not the GOP constraint. Red line represents a sequence encoded with GOP 15, while the blue line depicts the same sequence encoded without GOP restrictions. The Bitrate is computed for different QPs.

Sequence dimensions	Quantization Parameter	Bitrate increment (%)
176x144	16	0,86
176x144	32	5,95
352x288	16	0,68
352x288	32	5,73
720x576	16	0,76
720x576	32	3,84

Table I  
INCREMENT OF BITRATE USING VIDEO SEGMENTATION DUE TO THE REQUIRED SPS AND PPS HEADERS TO SYNCHRONIZE THE DECODER

3) Finally, it is also important to consider the increment of bitrate due to the *SPS* and *PPS* headers that are used in every new video segment. In the case that all the video sequence is encoded once, they have to be sent to the client just one time at the beginning. This is not the case when the sequence is splitted in several segments as in our framework. In Table I we include the increment of bitrate for different video resolutions at different levels of quality (by modifying the quantization parameter: *QP*). As we can observe, the cost of the headers is very low and almost negligible for higher quality encoding parameters ( $QP=16$ ). The size of the header is almost constant in every case, independently of the encoding parameters that are being used. Hence, when the quality of the image is increased at the cost of spending bitrate, the related cost of the headers gets lower and lower. The video segmentation occurs again approximately every 15 frames.

## VI. CONCLUSION AND FUTURE WORK

In this paper, we described a flexible interactive streaming system, over one underlying key mechanism: temporal content pipelining, which allows to switch the video content at whichever point of the playback in a temporally consistent way. This mechanism uses the client's feedback, requiring only one open streaming session per user and no advanced implementation mechanisms. Experimental results show that the video segmentation doesn't have any effect in the fluency of the streaming playback and in addition, the bitrate is not significantly increased. Therefore, the browsing features don't damage the global performance of the system.

We also present three different switching capabilities when streaming video soccer content: zooming over RoIs, fast forward and additional replays selection. All together, subjectively increases the perceived quality of the streaming experience. The profits of our architecture mainly rely on supporting personalized content selection according to the interaction with the viewer.

Finally, our framework is also able to include, for example, targeted advertising just by implementing the concept of client profile. In addition to the interactive selection of versioned video segments, the architecture is also designed to allow the insertion of promotional or any other kind of content in the middle of the main streaming playback. Later, the playback can be resumed directly without any kind of offset, interruption or efficiency cost. Hence, our interactive architecture can be extended to offer support to multiple streaming applications. In this paper we focus on adapting broadcasting TV soccer content for smart phone terminals.

## ACKNOWLEDGMENT

The authors would like to thank Walloon Region project Walcomo and Belgian NSF for funding part of this work

## REFERENCES

- [1] M. F. Sayit and T. Tunah, "Video streaming with h.264 over the internet," in *Signal Processing and Communications Applications*, Apr. 2006, pp. 1–4.
- [2] Z. Li and Z. Zhang, "Real-time streaming and robust streaming h.264/avc video," in *Third International Conference on Image and Graphics*, Dec. 2004, pp. 353–356.
- [3] A. Argyriou and V. Madiseti, "Streaming h.264/avc video over the internet," in *Consumer Communications and Networking Conference*, Jan. 2004, pp. 169–174.
- [4] X. Yu and D. Farin, "Current and emerging topics in sports video processing," in *IEEE International Conference on Multimedia and Expo (ICME)*, 2005.
- [5] Y. Takahashi, N. Nitta, and N. Babaguchi, "Video summarization for large sports video archives," *Multimedia and Expo, IEEE International Conference on*, vol. 0, pp. 1170–1173, 2005.
- [6] E. Bomcke and C. De Vleeschouwer, "An interactive video streaming architecture for H.264/AVC compliant players," in *IEEE International Conference on Multimedia and Expo(ICME), New-York, USA*, 2009.
- [7] ITU-T, "H.264: Advanced video coding for generic audiovisual services," *Series H : Audiovisual and multimedia systems*, 2005.
- [8] H. Schulzrinne, A. Rao, and R. Lanphier, "Real time streaming protocol (rtsp)," RFC 2326 (Proposed Standard), Apr. 1998.
- [9] D. Delannay, C. De Roover, and B. Macq, "Temporal alignment of video sequences for watermarking systems," Santa Clara, USA, 2003, pp. 481–492, SPIE.
- [10] J. Owens, "Television sports production, 4th Ed," in *Focal press*, 2007.
- [11] H. Pan, P. van Beek, and M. I. Sezan, "Detection of slow-motion replay segments in sports video for highlights generation," in *ICASSP '01: Proceedings of the Acoustics, Speech, and Signal Processing, 2001. on IEEE International Conference*, Washington, DC, USA, 2001, pp. 1649–1652, IEEE Computer Society.
- [12] A. Ekin, A. M. Tekalp, and R. Mehrotra, "Automatic soccer video analysis and summarization," *Image Processing, IEEE Transactions on*, vol. 12, no. 7, pp. 796–807, 2003.
- [13] F. Lavigne, F. Chen, and X. Desurmont, "Automatic video zooming for sport team video broadcasting on smart phones," in *International Conference on Computer Vision Theory and Applications, VISAPP, Angers, France*, 2010.
- [14] F. Chen and C. De Vleeschouwer, "A resource allocation framework for summarizing team sport videos," in *IEEE International Conference on Image processing(ICIP)*, Cairo, Egypt, 2009.
- [15] J. Li, T. Wang, W. Hu, M. Sun, and Y. Zhang, "Soccer highlight detection using two-dependence bayesian network," in *IEEE International Conference on Multimedia and Expo (ICME)*, 2006.
- [16] "Live555 media server streaming application library's webpage," <http://www.live555.com/liveMedia/faq.html>.