OBJECT-ORIENTED APPROACH TO VIDEO EDITING AND BROADCASTING TO THE INTERNET

KOROLEV, D.

Abstract: *Using video on the Internet has become a common practice, but the television-like 'passive viewer' approach misses the benefits of the interactive nature of the Internet. The technological limitations of television can be overridden by the Internet. Having multiple sources of input does not mean they should be merged into one editor-controlled flat output. Treating streams as objects, it is possible to make viewers editors for their screens whenever they want, or let them watch a pre-edited version. Active streams are distributed to viewers to gain control over the scene layout. Recorded scenes can be remastered whenever needed and represented in different views simultaneously. For lectures and conference recordings, inline slide browsing is also possible. This approach was successfully tested in the Viditory.net project for the broadcasting and recording of conferences with multi-camera shots and remote speakers. Despite the Adobe Flash platform becoming obsolete, it is possible to implement similar capabilities on modern platforms and by using modern technologies.*

Key words: *video editing, broadcasting, synchronization, streaming, interactivity*

Authors´ data: Associate Prof. **Korolev**, D[enis]. National Research University Higher School of Economics (HSE), Moscow, Russian Federation. dkorolev@hse.ru.

This Publication has to be referred as: Korolev, D[enis] (2014). Object-Oriented Approach to Video Editing and Broadcasting to the Internet, Chapter 49 in DAAAM International Scientific Book 2014, pp.605-614, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-901509-98-8, ISSN 1726-9687, Vienna, Austria
DOI: 10.2507/daaam.scibook.2014.49

1. Introduction

The broadcasting of conferences often demands making camera work cheap and easy, but at the same time adding specific features like teleconference and re-mastering of recordings. It requires using a completely different ‘object-oriented’ approach to the contribution, editing, storing and representation of the video streams.

The editing of motion pictures has come a long way since the invention of cinema in 1895. The first attempts to formalize film editing in 1919 are known as Kuleshov's "Principles of montage" (Levako, 1974). These principles define a set of recommendations for ‘viewer-comfortable’ film editing. Later, new technologies and techniques were invented and applied (Medynsky, 1992), but the result was a sequence of frames, only consumed on the viewer's side: it was always dictated by an editor and seen as the ‘author's view’ and the author's right. The only action the viewer could take was to switch the channel or control the tape. Late analogue television technologies allowed the viewer to partly control the picture on screen [e.g. teletext overlay (ETS 300 706), electronic program guides (ETS 300 707), picture-in-picture], but generally the viewer remained passive until videogame came along. The digital era brought interactive technologies to the screen, but television programs and even most multiple part events remain passive-view programs, no matter whether seen on TV or on a website.

The Internet currently uses unicast network technology, demanding a personal connection from each window to a broadcasting server. It is wasteful in terms of network traffic, but allows personalization and tracking of every action taken by the user. Generally, there is no other way to watch a video on a website at the moment.

This paper will focus on making the most of a medium while videoconferencing and broadcasting multiple camera shots. ‘Objectively’ recorded content includes all the source streams and thus can be easily remastered and represented in multiple layouts and decor simultaneously. The streams, unlike traditional video, are normally lightweight and cheap to store on a server. However, traffic issues, multiple stream synchronization and latency problems may arise in a heterogeneous network environment and will be covered below.

2. Current technology

Normally, video sources (cameras, video players, computer graphics and title overlays) come to the input of a video switcher where they are mixed according to an editor's decision. Output is represented by a composite analog or digital video signal, which possibly contains a transparency channel.

A typical conference video recording (broadcasting) schema is shown in Figure 1: two cameras provide switchable plans and one source provides a screen capture. This is enough to produce a video program if no shots of the hall are needed.

The editor takes decisions at appropriate moments and switches between the three inputs. Modern video mixers often have useful functions, to extend the visual effect of the resulting program, such as "Picture in Picture" (PiP).

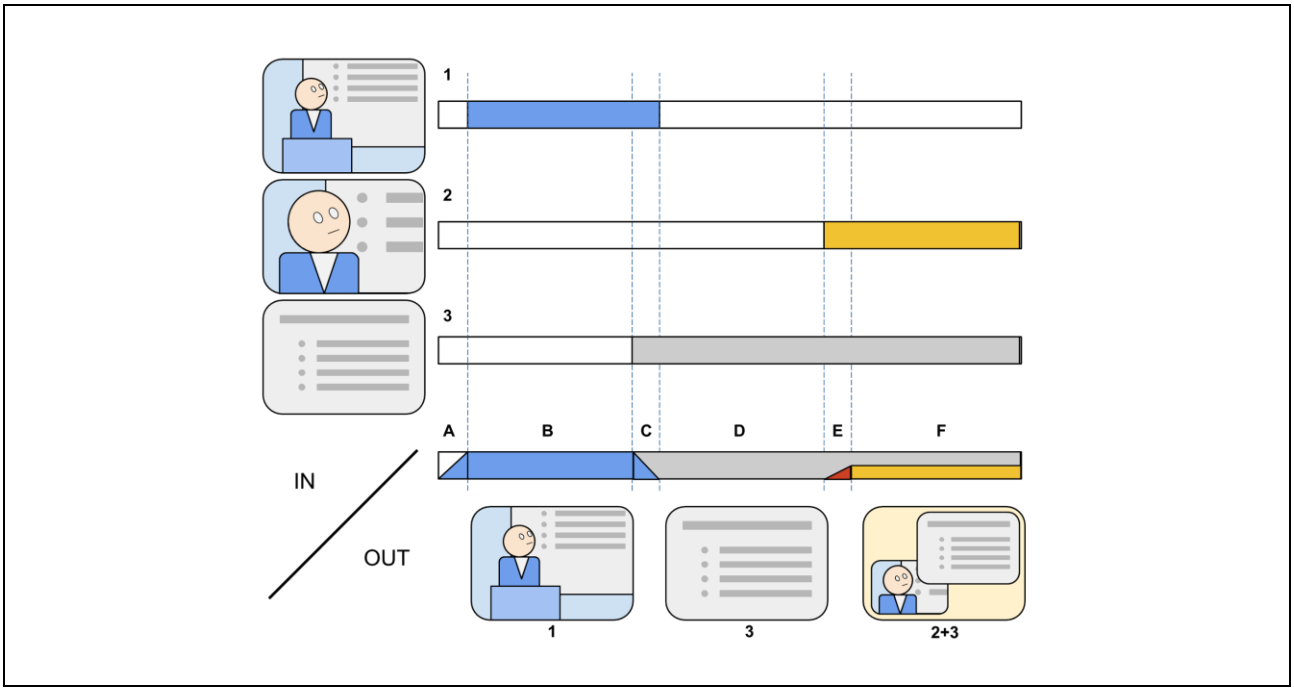

Fig. 1. Ordinary linear video editing process

In this case different sources come to a video mixer or non-linear editing (NLE) system and come out in a ready-to-watch format. This is similar to program code being compiled in binary code to be optimized for execution on a computer.

When working with remote sources such as videoconferencing, we need to compose an individual program for every participant of the videoconference. For two participants it is easy – just watch each other's video – but for multiple participants either a centrally managed or manual mode is needed to switch between the speakers. Currently, there are two different solutions:

- server-based, with all streams coming to a server and composed streams broadcasted to participants and viewers,
- serverless, also known as peer-to-peer, when all streams come to all participants.

3. An object-oriented approach to multiple source video representation

When working with all streams separately, the 'object-oriented' approach becomes possible. It can be compared with program interpretation rather than compilation: a computer takes more processor time each time it executes the code, but script can be changed (personalized) whenever needed.

To explain the difference from the classic approach, we will produce the same sequence from the same sources as in the previous example. There are templates with predesigned locations of regions, where the sources can be mapped to use these streams in a resulting scene. As seen on Figure 2, presets can have different layouts; they also can be used multiple times with a different set of sources. Every source can be mapped to any preset.

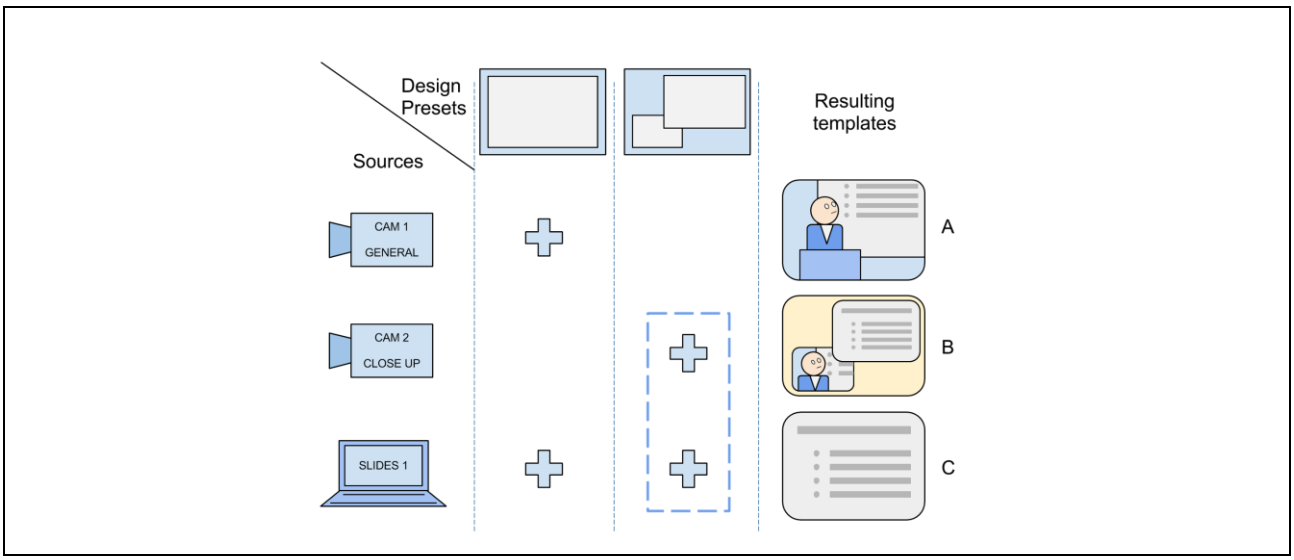

Fig. 2. Mapping the sources to templates

Once the resulting templates are named and stored, the editor can switch between them. Figure 3 shows how these three templates switch. Transitions and fades are replaced by template swapping animation.

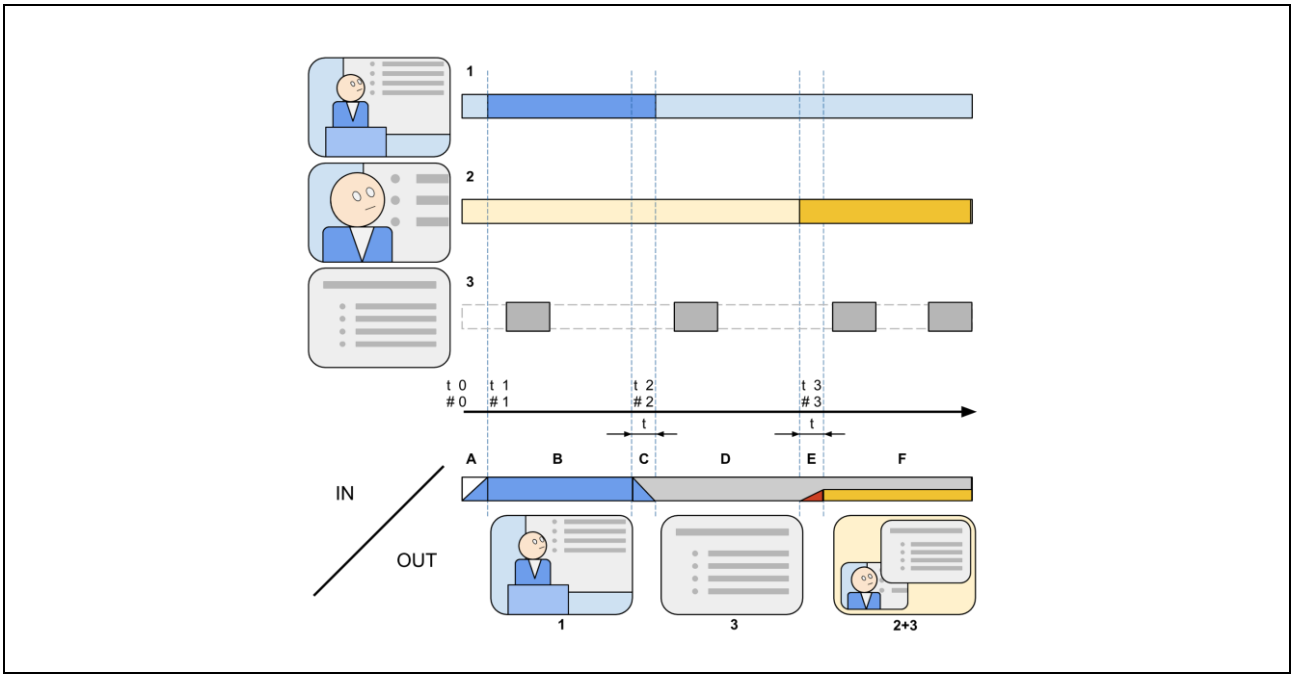

Fig. 3. Object-oriented video editing

Both cameras upload video streams to the server continuously. In order to reduce traffic, a screen capture client only uploads still images if a change on the screen is detected. To capture motion from the screen, screen casting should be used and it will be treated as a video source.

3.1 Traffic issues

We will take two services as an example. A few years ago the object-oriented approach was implemented on an experimental platform, "Viditory.net". It was supposed to become a multiple-sided video interaction server-based recording engine,

but development was canceled because it was based on Adobe Flash, which did not work on mobile devices. Another reason to stop further development was the emergence of Google Hangouts, which was the nearest functional analog and with potential to take the whole market. Here we will compare two examples.

The bitrate on the participant's side for uncomposed video is counted as

$$B = \sum_{i=1}^{n} R_i + r \quad (1)$$

where:
B is the overall bitrate;
n is the number of participants;
R is the bitrate, uploaded by *n*-th user;
r is the concomitant bitrate from titles, commands, text and static graphics.

As seen from the formula, participants can have different upload bitrates. Sources may be of different types: video, text, graphics or any kind of content.

Compared to a single stream normally received from the server, it looks like a waste of traffic. The formula above describes only end-user traffic, but normally home or office internet connections are fast and stable enough to pay more attention to the server side first. If there was a single stream for every participant and viewer, as with ordinary live broadcasts, it would depend only on the number of connections:

$$B = R \cdot N \quad (2)$$

where:
B is overall upload bitrate from server;
R is stream bitrate;
N is number of connections (viewers and participants).

This means that the number of connections is limited by channel bandwidth on the server hosting. To avoid overloads and the reservation of unused channels, commercial content delivery networks (CDN) can be used (Rajkumar et al., 2008).

There is a trick to lessen traffic on the user side: the stream that is not shown at the moment can be received and saved on server, but not delivered to all participants. If a video meeting has *N* participants with equal video stream upload bitrate and only *n* of them are active, traffic will reduce almost in *N*-*n* times, because other traffic *r* is disproportionately small.

3.2 Cashing

To avoid dropped frames and play a stable soundtrack, both the server and the players usually cache the video streams (Safar, 2014). When an inactive video stream is activated (shown on the screen), users see a black screen at first until the stream is cached on the player side. For real-time video meetings the cache is normally small (0,2 – 1 sec.), but playing stored records normally has a larger delay to provide better user experience with less requirements for communication infrastructure. Google Hangouts caches up to a minute for broadcasting, while conferencing has an imperceptible delay. Thus, real-time object-oriented editing is not as transparent as

traditional linear video editing. To avoid showing an uncached stream to viewers there should be a delay in the editor's command execution, but replaying the same scene from record can be done in accordance with the editor's command – the stream will be cached in advance.

There is a simple way to automate prior caching of an inactive video stream: using queue (preset) bus as in ordinary video mixers. When directing multicamera live video recordings or broadcasts, an editor normally uses preview, program and preset buses. Choosing a source in a preset bus means it is the next on air when the "Action" button is pressed.

3.3 Videoconferencing

Normally every participant should see a unique stream representing all other participants, but himself (Maldow, 2012). This means that the server should encode as many streams as there are users participating in a meeting. In this case, peak load refers to CPU time while overall upload bitrate remains almost constant. Google Hangouts allows up to 15 simultaneous participants: unlike viewers, participants control the screen and switch the main screen themselves (The Verge, 2013). If we handle separate streams, no extra CPU job is needed to recode the video.

Figure 2 represents videoconferencing screens on the right side and corresponding video sources on the left. Let us see how multi-screen is composed in Google Hangouts and then come to the object-oriented approach.

For N participants, there are N managed (operated by the participant) screens generated on the server. The preview line at the bottom is equal for all, but every user can change the main screen for himself. In this multipart videoconferencing system, a global editor is not supported and every user chooses the screen to be shown in the main window. There can be maximum N different screens generated by the server. With Google Hangouts the number of participants is the same as the number of previews ($N=M$).

To compare this classic approach to the object-oriented approach, we assume that every stream is sent separately and collected on the user side in the same composition, as in Google Hangouts. This enables personalization to all viewers, not only participants. At the same time an editor’s decisions can be applied and optionally overridden or supplemented with viewer's commands.

The Viditory project experimented with the object-oriented approach. It used templates to operate objects and the editor could change any of the previously prepared templates, containing a set of objects from a given number of sources.

3.4 Synchronization issue

The main concern with independent upload is synchronization: while television equipment uses precise frame synchronization, here a signal is delivered with absolutely no interconnection using TCP/IP. It does not guarantee exact delay. Most unsynchronization appears while encoding video on the camera side if the computers significantly differ in power or have different load at the moment. Wireless connection also increases the possibility of unsynchronization (Rakun et al., 2011),

but a stable connection to a dedicated Wi-Fi access point gives deviations within acceptable 0,5 second latency interval.

A simple way to keep the resulting scene synchronized is to take sound from a close-up camera stream and to avoid other close plans in one scene (Kuzmanic et al, 2007). However, flashes and fast movements can clearly show the delay between different cameras. Precise synchronization can be implemented using timestamp, included in video stream metadata or tone signal, but it increases caching delay and does not guarantee synchronized delivery from the server to viewers. Synchronizing streams on the user side will require an application rather than browser.

Another case, that excuses most technical issues, is using mobile phones or tablets as video sources. Public broadband wireless networks are not reliable for media streaming and cause high and variable latency. However, the resulting effect, when any event can be presented on the internet with as many viewpoints as needed, or when a 'moving camera' can be added to several static ones, gives a major advantage in user experience and reporter's efficiency. Mass events produce masses of amateur video footage and even broadcasts, usually run on free public services like U-Stream. Collecting them in a package of streams gives the editor angles to select for the broadcasting. When sources are distributed so that field of view differs (another room, building, city), synchronization is not so important and the benefits of the object-oriented technique become more evident.

4. Objectified benefits

What is the benefit for the viewer, participant or editor in choosing such an unusual and consumptive approach to this stable and conservative field? Nearly 100 years of linear video editing has produced composite video stream for television and video records, why should we now set it upside down and waste more traffic than ever before? There are at least two answers.

1. *Amateurs*. 'Television of professionals' still remains in studios, but 'television of amateurs', casual broadcasts, run by unskilled staff is becoming more commonplace and often brings the most interesting events that would be unshown by 'big TV'. Single camera broadcasts from the rear of a hall are boring. Viewers are used to seeing quickly changing plans on a TV-screen with no long shots (Simicevic & Racic, 2008). There is nothing difficult about making a video broadcast from any event with a mobile phone, tablet or a special device like Teradec Cube, but once at least two cameras are used, it requires a video mixer, editor, cables, intercom, tally-lights and other specific things. The object-oriented approach gives amateurs a chance to make a mistake and gives them a tool to easily fix it. It becomes easy to take any source to live air, mix online, personalize output and narrate remotely.

2. *Multiple participants.* Video and television are not interactive. Once the internet gives us interactivity, video and television will become interactive to survive. Currently Google Hangouts is pushing this trend in public services: Coursera teachers often use it [McGuire, 2013], though it is limited to maximum 15 participants with

lots of viewers – it is a step toward converting a ‘talking head’ to a person talking to viewer.

5. Fields of application

General applications of the object-oriented approach are:

- Most multiple camera broadcasts and video meetings;
- Any live event or its recording where interactive elements are required;
- Cases of multiple representation of the same content (PC, mobile, ‘view-only’ big screen).

For low-cost amateur use, the following schema can be suggested: every camera angle is equipped with a laptop or even a smartphone. Each angle is broadcasted to the server independently and streams are shown together or sequentially in templates. Thus, if the upload channel is wide enough, a large number of streams can be contributed to the server, combined for live broadcast and replayed later. If the event is distributed at least between neighboring rooms (or continents – it does not matter), synchronization becomes less critical and more benefits can be gained from the object-oriented approach.

The object-oriented approach in video broadcasting was initially intended for recording and Internet broadcasting of conferences, lectures and other spoken genre events. The idea was to make the video service user experience better than being in the hall, and even to bring this service to the audiences’ devices. First of all, it is possible to give a viewer control over template switching and slide flipping. Viewers in the hall cannot do this, but they at least can move their eyes from the speaker to the screen whenever they want. To give a remote viewer an advantage over those present, it is possible to make past slides browsable.

When a distributed event needs to be broadcasted to a big screen, to an internet-site and to remote participants simultaneously, it requires multiple output programs. In this case multiple-view templates can be used and no extra hardware will be needed. Such templates have different sources to be shown to different participants. It is also possible to provide an information support service, giving a set of source streams and tools of narration and packaging to end-user video programs. Thus, different media companies and individuals could produce their own live reports using this platform as it requires no extra hardware or software.

The reconstruction of a service like Viditory still makes sense, because the initial impression from Google Hangouts as a ‘killer-application’ has faded after its real-life experience. This service can be used with strong disadvantages: sound quality is poor and speakers should be close to the microphone; disconnections are fatal for broadcasting; and no screen re-composing or template management is available.

In Viditory all streams were recorded and the scene could be replayed right after the broadcast was finished. The recording could then be remastered. Having all

streams untouched and the editor's decision represented as a set of simple recordings (timestamp, template_id, captions_id, animation), it would be easy to edit this 'program'. Also, the technology could allow every user to override the editor's choice and switch templates himself. The editor could block this feature, but in some cases it would be useful to have both options.

6. Future work

Viditory was developed on the Adobe Flash platform, which is now obsolete. To revive it needs to be re-constructed with modern technologies. While HTML5, CSS3 and other established standards allow rich multimedia functions, video streaming protocols still remain an issue: the only applicable protocol is RTMP [Koeing et al., 2009, Dragan & Ivetic, 2009], which comes from the Adobe Flash Media Server and has similar problems of Adobe Flash. However, emerging technologies, such as MPEG DASH promise more elegant solution, but still remain in the testing phase of development.

7. Conclusion

- Object-oriented video editing and broadcasting brings new possibilities to the video experience: managed user-side control of the scene, editability, ease of amateur use and scalability.
- The suggested approach integrates interactivity and editability at all stages of the lifecycle.
- The current state of consumer mobile platforms and communications does not allow the use of the full range of capabilities given by the object-oriented approach, but individual advantages can be used to extend current services or bring new ones to former offline events, such as mobile application for real-time slide browsing or a bookmarking and sharing tool for conference audiences.
- Amateur or casual use of Internet broadcasts often puts efficiency over quality of video. Here syncronization issues can be shelved, making possible the cheap and easy multiple camera broadcasting of almost any event from almost any video-capable device (tablets, mobile phones, web-cameras or traditional equipment).

8. Acknowledgements

The experimental work would not be done without the enthusiasm and hard work of my former students and later colleagues: Andrey Bulatov, Dmitry Andreev, Konstantin Aksenov and Ivan Pashintsev. I appriciate the support of the ICT department of MIEM HSE and its head professor Vladimir Azarov.

9. References

ETS 300 706 (1997) *Enchanced Teletext Specification*, European Telecommunication Standards Institute. ISBN 2-7437-1488-3

ETS 300 707 *(1997) Electronic Programme Guide (EPG); Protocol for a TV Guide using electronic data transmission.* European Telecommunication Standards Institute. ISBN 2-7437-1486-7

Giuliano L. A. (2011) *Next generation TV over the Internet: This Revolution will be televised*. Network World

Levako R. (1974) *Kuleshov on Film. Writings by Lev Kuleshov.* The Regents of the University of California. ISBN 0-520-02659-4

Maldow, 2012 David Videoconferencing Infrasrtucture: A Primer. Webtorials.com

Medynsky S. E. (1992) *Komponuem kinokadr* (*Composing Film Frame),* Iskusstvo. ISBN 5-210-00236-5

Rajkumar Buyya, Mukaddim Pathan, Athena Vakali (2008) *Content Delivery Networks.* Springer. ISBN 978-3-540-77886-8

Safar Seema et al. (2014) *Capacity Managed Adaptive Videostreaming Based On Peer Cache Adaptation Mechanism.* Int. Journal of Engeneering Research and Applications. ISSN 2248-9622 Vol. 4 Issue 4

The Verge (2013) *Exclusive: Inside Hangouts, Google's big fix for its messaging mess*

Rakun, J; Berk, P; Ocepek, M & Lakota, M (2011). *Wireless Performance of an Embedded Video Acquisition System*, Chapter 34 in DAAAM International Scientific Book 2011, pp. 425-432, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3- 901509-84-1, ISSN 1726-9687, Vienna, Austria

Kuzmanic, I.; Beros, S.M. & Vujovic, I. (2007). *Collaborative Experimental Results on Visibility Threshold and Calculation of JND Based on Human Visual Perception*, Chapter 21 in DAAAM International Scientific Book 2007, B. Katalinic (Ed.), Published by DAAAM International, ISBN 3-901509-60-7, ISSN 1726-9687, Vienna, Austria

Simicevic, V & Racic, I (2008). *The Content Analysis of TV Commercials: Statistical Approach*, Chapter 62 in DAAAM International Scientific Book 2008, pp. 773-780, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-901509-66-7, ISSN 1726-9687, Vienna, Austria

Koenig, S & Seminsky, J (2009). *Real-Time Data Transmission Protocols for Industrial Networks*, Chapter 79 in DAAAM International Scientific Book 2009, pp. 823-830, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-901509-69-8, ISSN 1726-9687, Vienna, Austria

Dragan, D & Ivetic, D (2009). An Approach to DICOM Extension for Medical Image Streaming, Chapter 04 in DAAAM International Scientific Book 2009, pp. 025-034, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-901509-69-8, ISSN 1726-9687, Vienna, Austria