









# Multi-device Content Based on Video. A Practical Toolset for Creation and Delivery

Joan Llobera<sup>1</sup> , Isaac Fraile<sup>1</sup> , Juan A. Núñez<sup>1</sup> , Szymon Malewski<sup>2</sup> ,  
Xavier Artigas<sup>1</sup> , and Sergi Fernandez<sup>1</sup> 

<sup>1</sup> i2cat Foundation, C/Gran Capità 2-4, 08035 Barcelona, Spain  
joan.llobera@i2cat.net

<sup>2</sup> PSNC, Jana Pawła II 10, 61-139 Poznań, Poland

**Abstract.** The arrival of head mounted displays (HMDs) to the contemporary living room extends the need for producing content that works for the television and companion screen devices. In this work we introduce a set of three tools that can be used for the production and delivery of synchronous multi-device content, across the TV, companion screens and HMDs. The production tool is implemented as a custom Adobe Premiere Pro plugin. The publication and delivery process is implemented as an online service controlled through a web application. The content payout is implemented with a multi-device video player that combines video decoding and payout. In this article we introduce the design choices guiding our software development and the different tools we developed to realize it. We also detail some basic measures of system performance on different devices, and propose further steps towards the easy production and delivery of multi-device content.

**Keywords:** Multi-device synchronization · Multi-device content · Virtual reality Omnidirectional video

## 1 Introduction

The majority of TV consumers now watch TV programs in a multi-display environment [9]. Companion screens - most often smartphones - are generally used to check information not directly related to the events in the TV content being watched, or to interact in social media on topics related to the broadcast [4], sometimes at the expense of local social interaction [5]. Broadcasters have tried to orchestrate these different platforms, and there is reason to believe this contributes to user engagement [11]. Traditionally, companion screen applications have been developed ad hoc, and only very recently the industry has considered the need for specific production tools and production processes adapted to the reality of multi-device consumption [8]. In this context, it remains challenging to create and deliver synchronized video across the TV and companion screens, particularly when the goal is to match user expectations and preferences [1].

The arrival of virtual reality devices to the living room introduces more possibilities, but also further challenges. Since traditional video was not conceived to support



**Fig. 1.** Traditional and omnidirectional video formats. Top: an image typical of a traditional TV showing a football match. An insert informs the consumer that content is also available for tablets and HMDs. Bottom: a capture of an omnidirectional video with inserts of traditional cameras. This content is delivered synchronized with the main TV stream. Image courtesy of Lightbox ([www.lightbox.pt](http://www.lightbox.pt)).

interactive rendering techniques, the solution generally adopted in the audiovisual industry is the delivery of an omnidirectional video stream. Omnidirectional video is quite different from traditional video: the rendering process requires the image to be projected, typically, on a sphere or on a cube, and only a small portion of the image is rendered, depending on the head orientation of the user, as detected by the HMD (see Fig. 1). As a consequence, an image of much higher quality has to be transmitted to render the same effective quality on the end-user screen.

In this article we introduce our efforts to enable the easy creation and delivery a new form of broadcast multi-device video within the European H2020 ICT project ImmersiaTV [19]. The main challenge addressed is to streamline the creation and delivery of a video-based synchronous experience across displays, including TV, companion screens such as tablets and smartphones, and HMDs. An additional challenge addressed is that we want the content delivered to be device-specific both in terms of video format (omnidirectional or traditional) and on how it supports interactive input (or lacks interaction support, for the case of the TV). For TV, this means that content can be consumed simply by sitting on the couch and watching, without further input, and that the audiovisual language used follows established conventions. For tablets and smartphones, it means that user input works seamlessly with the specificities of each device (head movements for HMDs, or finger-based input for smartphones or tablets).

To address these requirements, we have designed and implemented an end-to-end production, delivery and rendering pipeline for offline content production which specifically addresses these needs. In the following sections we further outline the design principles adopted (Sect. 2), the modules developed and the performance of the critical ones (Sect. 3), and summarize our conclusions, and the next steps we want to pursue (Sect. 4).

## 2 Design Principles and Related Work

### 2.1 Synchronous Multi-platform Layout

Typically, people watch TV in their living room. Often, they do so while doing other activities. These can range from engaging in conversation, playing with kids, social interaction on mobile phones, but can also span a myriad different activities. It is therefore unlikely that end-users will be actively engaged in trying different devices for media consumption, and checking what is possible at every time. On the contrary, it seems more likely that they will switch their attention to a particular content or device alternatively with other activities. We must therefore create content that provides experiences which allow for such limited attention span, and which allow the end-user to switch freely between devices.

In addition, if we want end-users to switch to particular devices in particular moments of the experience, we must indicate so across media. This is easy to do by adding overlays within the videos being delivered. For example, at a certain moment in the TV content, a small icon appears to indicate there is additional content available on HMDs. However, to enable such novel applications we must guarantee certain coherence across the overall experience, which seems only possible if we can deliver synchronous playout across devices.

In other terms: to create content for all devices, we need to create content that is adapted to each of them, and play it synchronously [9, 11, 18]. To play synchronized content, we have adapted emerging standards [20] and Gstreamer's version of the Precision Time Protocol (IEEE 1588) [21], as done, for example, in [17]. We have also embraced the use of omnidirectional video for HMDs and smartphones, in order to allow the user to visualize the scene in different directions. To further facilitate user adoption,

we have also extended synchronized playout to a web-based player. This allows delivering synchronized experiences also with web browsers.

Through the different devices the audience is still able to watch TV sitting on their couch, or tweet comments about it. However, the audience can also use immersive displays to feel like being inside the audiovisual stream, or use tablets and smartphones to explore these omnidirectional videos, or even, in the future, to zoom in, or share portions of it through social media.

## 2.2 Portals

In the context of streaming omnidirectional video, we introduce the idea of portals as video inserts that can be rendered in the HMD. The idea of portals is inspired from the homonymous and famous videogame Portal [22]. In the context of video streaming, these portals can be portions of other omnidirectional videos, which allows introducing basic interactive storytelling techniques such as scene selection or forking paths. Portals can also be inserts of traditional or directive videos. Traditional video inserts also allow reintroducing classical audiovisual language that is not possible to render solely with omnidirectional videos, such as close-ups, slow motion, shot-countershot, etc. (see also Fig. 2).

These strategies will not avoid the necessary precautions needed for shooting omnidirectional video [12]. Omnidirectional video requires thinking very carefully about how the end-user's attention is guided within the scene, and has a relatively narrow range of distances where the action is actually perceived clearly. If the action is too close, it will feel very flat, or deformed, and it can rapidly feel appalling for the end-user. If the action is too far, it will be difficult to see by the content consumer. Since omnidirectional video does not allow typical video techniques such as zooming in, or shooting close-ups, we believe it is likely that video inserts can facilitate the portraying of relevant details within the omnidirectional scene.

Actually, video inserts allow reintroducing the entire set of conventions of classic audiovisual language. In addition to close-ups, we believe that classical shooting strategies such as shot-countershots can be adapted to deliver a richer experience in omnidirectional video and help the user transition from traditional media to such emerging formats.

Portals also open the door for richer interaction, either inside the HMD or, since a portal can render the content available on another device, interact between devices. For example, it would be possible to present a video insert that is actually a transition towards another omnidirectional video scene. This enables the easy integration of branching narratives within audiovisual scenes based on omnidirectional video.

Last but not least, an additional reason to consider video inserts as portals is that such metaphor can also work for more immersive media. To understand why this is relevant we must step back, and consider the fact that virtual reality experiences work better when they support sensorimotor correlations [14]. For the case of rendering visual stimuli, this means that when the end-user changes his head position or orientation, the rendered content updates the rendered perspective accordingly. Despite omnidirectional



**Fig. 2.** The recording setup. Top: a camera setup to record traditional and omnidirectional video simultaneously. Bottom: a schematic diagram of possible directive inserts located within the omnidirectional video. Image courtesy of Lightbox ([www.lightbox.pt](http://www.lightbox.pt)).

video supports head rotations, it still falls very short at delivering sensorimotor correlations.

In this context, we must consider emerging formats such as free viewpoint video (FVV). Despite free viewpoint video was introduced several years ago [2, 15, 16], and improvements on the ease of production and delivery of such content appear regularly [3, 7], to the best of our knowledge currently there is no easy and cheap commercial solution that enables the intuitive creation of FVV. However, since such formats would radically improve the quality of the virtual reality experience, it is not impossible that the rise of virtual reality displays also comes together with a novel generation of FVV production techniques. In this perspective, the introduction of portals as a means to reintroduce classical audiovisual conventions or branching narratives is particularly promising, due to the fact that portals allow preserving at all times the capability of the content to adapt the perspective to the end-users actions. This is true *even for the content rendered in the portal*. In this way, were FVV made available for production, we could introduce conventions of classical audiovisual language, as well as branching narratives, while preserving place illusion, i.e., the feeling of *being there*. This might seem far-fetched, but given the current pace of media evolution in relation to VR, and the benefits FVV can bring to it, in relation to supporting sensorimotor correlations, it is not completely impossible that such format will have wider industrial adoption.

All in all, from our current perspective, which is focused on trying to identify good design principles to build meaningful multi-device experiences, these different arguments suggest that the consideration of video inserts as portals to be rendered within an omnidirectional scene is a good design choice.

### 3 An End-to-End Pipeline

Designing and implementing a broadcast audiovisual production chain is challenging due to the diversity of processes, technologies and production practices that it requires. In this section we outline the main solutions, either adopted or implemented, for our purpose, with content examples.

#### 3.1 Capture

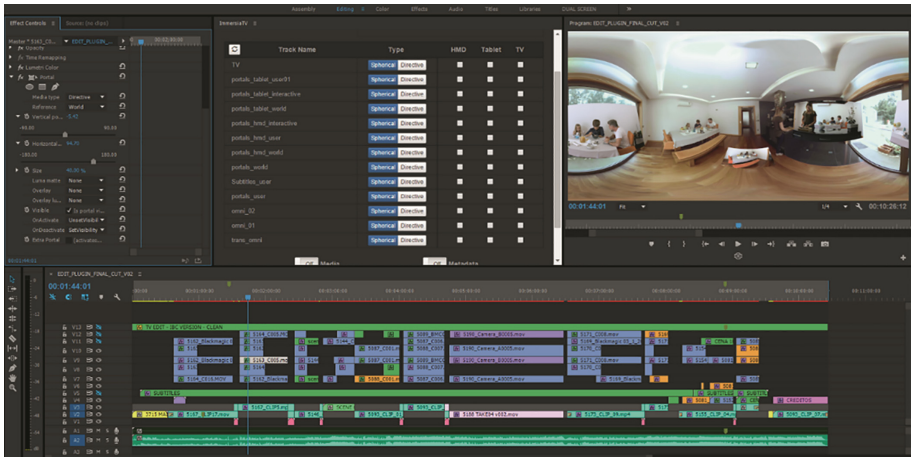
The creation of content that is both omnidirectional and traditional requires shooting simultaneously in both content formats. Preliminary tests with separate shootings for omnidirectional and traditional cameras revealed it was unfeasible to synchronize two different shootings, even when the actors in the scene were repeating the same actions.

The solution found by the production team was to use two *BlackMagic Micro Studio Camera 4k* micro-cameras for the traditional shooting, which could be hidden or, if visible, removed in post-production with a reasonably small amount of effort. This combined with an omnidirectional capture rig, which was either composed of 6 GoPro 3 Black Rig cameras, or 3 of them with fish-eye lenses (see Fig. 2) allowed capturing simultaneously traditional and omnidirectional footage. However, for a joint shooting, we must address the fact that omnidirectional cameras capture the whole visual field,

and therefore would show the traditional camera and the film crew behind it. This is not problematic for sports or music events, but it goes strongly against the conventions of fiction or documentary.

### 3.2 Edition

Dedicated stitching tools such as Video-stitch studio by Video-stitch [23], or Autopano by Kolor, allow stitching video footage captured with camera rigs in order to create omnidirectional video. Tools for omnidirectional video edition, such as CaraVR [24] and Mettle’s suite [25] allow further post-production. However, we are not aware of an editing tool targeting synchronous rendering across devices. To address this fact, we have designed and implemented a plugin for Adobe’s Premiere Pro. The ImmersiaTV Premiere Pro plugin (Fig. 3) allows defining the inserts that are placed within an omnidirectional scene, and how they should behave relative to the movements of the user. For example, they can either be static on the screen, or static on the omnidirectional scene. They can also trigger transitions between different omnidirectional videos.



**Fig. 3.** Video editing tool for multi-platform content. The Adobe Premiere ImmersiaTV panel, shown at the center, allows defining omnidirectional and directive (i.e., traditional) tracks, as well as which devices does each track target. The inserts added to the omnidirectional view, shown at right, can be edited with the ImmersiaTV Portal Effect, whose widgets are shown at the left. Image courtesy of Lightbox ([www.lightbox.pt](http://www.lightbox.pt)).

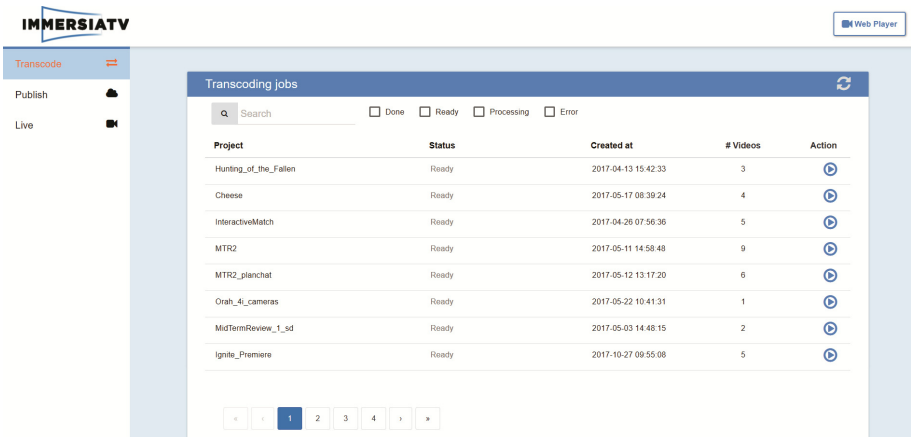
The ImmersiaTV Premiere Pro plugin also allows selecting which tracks should be rendered in each of 3 possible devices (TV, tablet or HMD). It works both with Mac and Windows, and has been tested with a variety of video editors.

Since this is the main element allowing creative minds to make design choices in the experience created by multi-device content, we have invested a considerable amount of effort to make sure we integrated with existing features of video edition software, and in particular with Premiere Pro. An example of such features is the use of interactive

transitions which are triggered to introduce or conclude a particular video insert. These transitions mimic exactly the behavior and user interface of traditional Premiere Pro transitions, but they are only triggered by the end-users input. Another example is the possibility of using nested sequences to combine different edits, something which has shown extremely useful to combine different tracks in a coherent edit, and then use it in different ways in different devices.

### 3.3 Encoding

The media encoding uses readily available tools for encoding in H.264 and AAC encoding formats. Adaptive bitrate streaming is based on MPEG-DASH (ISO/IEC 23009-1:2014). Encoding is implemented as a cloud service, running on a Linux server using the Dockers virtualization tool as well as MP4Box from Gpac’s MP4Box for MPEG-DASH multiresolution encoding [26]. Video decoding uses the Gstreamer library [27]. The additional metadata required for playout, which relates audiovisual streams with devices (i.e., allows selecting different streams for TVs and tablets), as well as to define interaction and media orchestration requirements, follows closely the format of MPEG-DASH manifests, and its XML specification is publicly available [28]. Content publication is performed through a custom built website (Fig. 4) which allows triggering media conversion, as well as monitoring progress on media encoding and publishing content generating a list of content parsed at the delivery stage.



**Fig. 4.** The web app allowing to control to trigger the transcoding of the different assets, as well as their publication for consumption.

### 3.4 Delivery

To combine universality, ease of use and flexibility, we combine an app-based solution together with a player based on web technologies. In both cases, metadata parsing is



done with a custom parser, which also generates the appropriate geometry and provides the DASH player with the appropriate DASH manifests.

The web player is based on WebGL and MPEG-DASH, implemented in a module for the generation and reproduction of the contents, and based on three.js and dash.js standard libraries. A second module synchronizes the contents following the DVB-CSS standard. Our web-based solution allows scene rendering without third party players or native applications. It can be served from a Content Delivery Network (CDN), allowing automatic updates of both the contents and the player. In addition, since it is based on web standards, it can be easily adapted to HbbTV solutions. In practice, this solution can reproduce up to  $4096 \times 2048$ , with 15 Mbps of bitrate and 30 frames per second. However, web-based solutions are intrinsically limited by the web browser stack to support communication and streaming technologies. For our use case, this has limiting implications for performance, codec constraints and hardware integration.

As an alternative, to facilitate the integration of video rendering with user input on native apps, the simplest option seemed to combine GStreamer, the reference library for multimedia pipelines, and Unity3D, the most accessible game engine for videogame and virtual reality developers. We designed and implemented the GStreamer Unity Bridge (GUB) to realize precisely this. The GUB has three parts. The GStreamer part receives the video and decodes it. This process is based on the GStreamer component playbin, and allows playing at least .mov, mp4 and MPEG-DASH. The texture passing is a key technical element: each frame decoded in GStreamer is passed to Unity3D as a hardware texture, and is suitable for rendering in the standard game engine environment. Specific code for windows (Direct3D 9) and Android (OpenGL-ES 2.0) has been developed for texture passing.

In addition, since copying full textures between system memory and the graphics processing unit (GPU) can have prohibitive performance costs at certain resolutions, particularly in low-end devices such as Android mobile phones, in the Android pipeline we have implemented a *Rendering to Texture* solution based on a Frame Buffer Object. This allows rendering a frame decoded in GStreamer without leaving the GPU, which brings significant boost in performance. Despite the overhead of handling a 3D Engine like Unity3D, the GUB can play resolutions that are competitive with commercial players (see Table 1). However, we also need to consider that rendering for mobile-based HMD, either cardboard or Samsung GearVR, imposes a double rendering process (one for each eye), which further decreases performance. Therefore, despite we can currently reproduce synchronously video up to  $4096 \times 2048$ , bitrate of 50 Megabits per second (Mbps) and 30 frames per second on a Samsung Galaxy S6, this resolution drops to  $1024 \times 512$ , bitrate of 2.3 Mbps and 25 frames per seconds when VR rendering is required.

To facilitate user adoption, we have made it publicly available under a LGPL license [29], raising considerable interest (In the first 10 months since it was published, we have had an average of over 250 downloads per month).

**Table 1.** Performance measurements for different test vectors. We show Frames per Second and a subjective estimate of a Mean Opinion Score by one user. Test vectors are: Hd (1980 × 1080), Bitrate: 3 Mb/s, Framerate: 25, Codec: H264, 2K (2560 × 1440), Bitrate: 3,5 Mb/s, Framerate: 25, Codec: H264; 4K (3840 × 2160), Bitrate: 5 Mb/s, Framerate: 25, Codec: H264 PC is Processor: Intel Core i7-6500U CPU @ 2.50 Ghz, Ram: 16 Gb, Graphics card: Intel Graphics 520, SO: Windows 10 Home edition 64 bits

Device	Test vector	FPS	MOS
Samsung S6	4K	25	5
	2K	25	5
	HD	25	5
Samsung S7	4K	20	4
	2K	25	5
	HD	25	5
Galaxy Tab S	4K	–	0
	2K	20	4
	HD	25 fps	5
PC	4K	25	5
	2K	25	5
	HD	25	5

## 4 Conclusions and Future Work

We have introduced two simple design principles which, when combined, seem appropriate to address the challenge of creating experiences that integrate TVs, companion screens and HMDs in a coherent experience.

To demonstrate the feasibility of this approach we have developed an end-to-end solution to enable the production and delivery of video-based multi-device synchronous and, at some extent, interactive experiences, with a performance that is comparatively equivalent to standard commercial video players. Performance tests show that the limit in delivered quality is determined by hardware processing load, rather than bandwidth limitations. Further work will be needed to optimize the media quality delivered, particularly for VR-based content, which requires separate renderings for each eye. For this purpose, tiling strategies [10, 13] seem a good direction to explore. For mobile devices, we are also considering a more heterodox DASH client which considers additional factors, beyond bandwidth, to select the most appropriate encoded quality [6].

On the content creation side, further development of content examples exploring more exhaustively the interaction possibilities enabled by inter-device synchronization is a different but complementary work that we would also like to pursue. In this direction, further work to refine and expand the possibilities given by the Premiere Pro plugin here introduced is desirable, particularly regarding the definition of interactive functionality, such as how the consumer’s input affects the media being rendered and the overall experience. Its usage with video editors, has showed that, although intuitive, these tools

present many limitations, particularly for interactive content. Adopting the fixed timeline characteristic of video, and central to the interaction metaphors on which video-editing software is based, rapidly feels quite limiting when we want to explore richer interactivity. It is therefore possible that, in order to expand the interactive possibilities we must stop using the fixed timeline metaphor, and switch to a node-based software, as typically found in sophisticated post-production solutions.

Globally, the integration of HMDs within the living room consumption habits is still a matter open to speculation. In this context, the evolution of innovative video formats such as FVV, together with the increasing ease with which we can produce mesh-based three dimensional content, as typically found in videogames, raises questions on content format which are difficult to answer beyond trying different options, and studying what works best. In this work we have demonstrated an end-to-end toolset based on simple design choices, and showed it can work in practice.

All in all, we believe the two simple design principles - synchronous playout and portals, and particularly their combination, provide a good starting point from which to design, produce and deliver multi-device experiences. However, the question of whether these principles should be implemented on a video-based pipeline, or an entirely different media format, is one which is still difficult to answer, given the speed of evolution and the variety of formats that are currently being used for VR production.

**Acknowledgements.** This work has been developed with the support of the European Union's Horizon 2020 programme under grant agreement No. 688619 ([www.immersiatv.eu](http://www.immersiatv.eu)). Developing an end-to-end pipeline requires competent and motivated programmers. In the work here introduced we acknowledge the contributions of Wojciech Kapsa and Daniel Piesik, from PSNC, as well as Ibai Jurado, David Gómez, Einar Meyerson, David Cassany and Juan Gordo from i2cat Foundation.

## References

1. Boronat, F., Montagud, M., Marfil, D., Luzón, C.: Hybrid broadcast/broadband TV services and media synchronization: demands, preferences and expectations of spanish consumers. *IEEE Trans. Broadcast.* (2017)
2. Carranza, J., Theobalt, C., Magnor, M.A., Seidel, H.-P.: Free-viewpoint video of human actors. In: *ACM Transactions on Graphics (TOG)*, pp. 569–577 (2003)
3. Collet, A., Chuang, M., Sweeney, P., Gillett, D., Evseev, D., Calabrese, D., Hoppe, H., Kirk, A., Sullivan, S.: High-quality streamable free-viewpoint video. *ACM Trans. Graph. (TOG)* **34**(4), 69 (2015)
4. Courtois, C., D'heer, E.: Second screen applications and tablet users: constellation, awareness, experience, and interest. In: *Proceedings of the 10th European Conference on Interactive TV and Video*, pp. 153–156 (2012)
5. D'heer, E., Courtois, C.: The changing dynamics of television consumption in the multimedia living room. *Convergence* **22**(1), 3–17 (2016)
6. Gómez, D., Boronat, F., Montagud, M., Luzón, C.: End-to-end DASH platform including a network-based and client-based adaptive quality switching module. In: *Proceedings of the 7th International Conference on Multimedia Systems*, p. 38 (2016)

7. Huang, J., Chen, Z., Ceylan, D., Jin, H.: 6-DOF VR videos with a single 360-camera. In: *Virtual Reality (VR)*, 2017 IEEE, pp. 37–44 (2017)
8. Meixner, B., Glancy, M., Rogers, M., Ward, C., Röggl, T., Cesar, P.: Multi-screen director: a new role in the TV production workflow? In: *Adjunct Publication of the 2017 ACM International Conference on Interactive Experiences for TV and Online Video*, pp. 57–62 (2017)
9. Montagud, M., Boronat, F., Stokking, H., van Brandenburg, R.: Inter-destination multimedia synchronization: schemes, use cases and standardization. *Multimed. Syst.* **18**(6), 459–482 (2012)
10. Niamut, O.A., Thomas, E., D’Acunto, L., Concolato, C., Denoual, F., Lim, S.Y.: MPEG DASH SRD: spatial relationship description. In: *Proceedings of the 7th International Conference on Multimedia Systems*, p. 5 (2016)
11. Rainer, B., Timmerer, C.: Self-organized inter-destination multimedia synchronization for adaptive media streaming. In: *Proceedings of the 22nd ACM International Conference on Multimedia*, pp. 327–336 (2014)
12. Sheikh, A., Brown, A., Watson, Z., Evans, M.: Directing attention in 360-degree video, pp. 43–47 (2016). <http://dx.doi.org/10.1049/ibc.2016.0029>
13. Skupin, R., Sanchez, Y., Hellge, C., Schierl, T.: Tile based HEVC video for head mounted displays (2016)
14. Slater, M.: Place illusion and plausibility in virtual environments. *Philos. Trans. R. Soc. B: Biol. Sci.* **364**(1535), 3549 (2009)
15. Starck, J., Hilton, A.: Spherical matching for temporal correspondence of non-rigid surfaces. In: *Tenth IEEE International Conference on Computer Vision (ICCV 2005)*, Volume 1, vol. 2, pp. 1387–1394 (2005). <https://doi.org/10.1109/iccv.2005.229>
16. Starck, J., Miller, G., Hilton, A.: Video-based character animation. In: *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation - SCA 2005*, p. 49 (2005). <https://doi.org/10.1145/1073368.1073375>
17. Veenhuizen, A., van Brandenburg, R.: Frame accurate media synchronization of heterogeneous media sources in an HBB context. In: *Media Synchronization Workshop* (2012)
18. Vinayagamoorthy, V., Ramdhany, R., Hammond, M.: Enabling frame-accurate synchronised companion screen experiences. In: *Proceedings of the ACM International Conference on Interactive Experiences for TV and Online Video*, pp. 83–92 (2016)
19. The ImmersiaTV Project. [www.immersiatv.eu](http://www.immersiatv.eu). Accessed 08 Jan 2018
20. The DVB-CSS Protocol. [http://www.etsi.org/deliver/etsi\\_ts/103200\\_103299/10328601/01.01.01\\_60/ts\\_10328601v010101p.pdf](http://www.etsi.org/deliver/etsi_ts/103200_103299/10328601/01.01.01_60/ts_10328601v010101p.pdf). Accessed 08 Jan 2018
21. Description of the Precision Time Protocol, as implemented in the Gstreamer framework. <https://gstreamer.freedesktop.org/data/doc/gstreamer/head/gstreamer-lib/html/GstPtpClock.html>. Accessed 08 Jan 2018
22. The Portal Game. <http://store.steampowered.com/app/400/>. Accessed 08 Jan 2018
23. VideoStitch Studio. <https://www.orah.co/software/videostitch-studio/>. Accessed 21 Feb 2018
24. CaraVR. <https://www.thefoundry.co.uk/products/cara-vr/>. Accessed 21 Feb 2018
25. Mettle. <https://www.mettle.com>. Accessed 21 Feb 2018
26. Gpac. <https://gpac.wp.mines-telecom.fr/mp4box/>. Accessed 21 Feb 2018
27. Gstreamer. <https://gstreamer.freedesktop.org/>
28. ImmersiaTV Server. [http://server.immersiatv.eu/public\\_http/metadata/ImmersiaTV.html](http://server.immersiatv.eu/public_http/metadata/ImmersiaTV.html). Accessed 21 Feb 2018
29. Gstreamer Movie Texture. <https://www.assetstore.unity3d.com/en/#!/content/59897>. Accessed 21 Feb 2018