Immersive Captioning: Developing a framework for evaluating user needs

Chris J. Hughes
School of Computing,
Science and Engineering
Salford University
Salford, UK
c.j.hughes@salford.ac.uk
0000-0002-4468-6660

Marta B. Zapata
Department of Translation,
Interpreting & Eastern Asia Studies
Universitat Autònoma de Barcelona
Bellaterra, Spain
marta.brescia@uab.cat
0000-0002-2465-0126

Matthew Johnston
Professional Services
ThoughtWorks
London, UK
mjohn@thoughtworks.com

Pilar Orero
Department of Translation,
Interpreting & Eastern Asia Studies
Universitat Autònoma de Barcelona
Bellaterra, Spain
pilar.orero@uab.cat
0000-0003-0269-1936

***Abstract*— This article focuses on captioning for immersive environments and the research aims to identify how to display them for an optimal viewing experience. This work began four years ago with some partial findings. This second stage of research, built from the lessons learnt, focuses on the design requirements cornerstone: prototyping. A tool has been developed towards quick and realistic prototyping and testing. The framework integrates methods used in existing solutions. Given how easy it is to contrast and compare, the need to further the first framework was obvious. A second improved solution was developed, almost as a showcase on how ideas can quickly be implemented for user testing. After an overview on captions in immersive environments, the article describes its implementation, based on web technologies opening for any device with a web browser. This includes desktop computers, mobile devices and head mounted displays. The article finishes with a description of the new caption modes leading to improved methods, hoping to be a useful tool towards testing and standardisation.**

Keywords—Accessibility, VR, Immersive video, Subtitle, Captions

I. Introduction

Immersive media technologies, like Virtual Reality (VR) and 360º video have rapidly grown in popularity due to the availability of consumer level head mounted displays (HMD). This influences not only in the entertainment sector, but also in other key sectors of society, like education, arts and culture [1] especially during the times of the COVID-19 pandemic [2] when people are less able to travel. In this context, 360º videos have become a simple and cheap, yet effective and hyper-realistic, medium to provide VR experiences.

Due to this potential, the scientific community and industry have devoted significant resources to developing new solutions in terms of many relevant aspects, like authoring and playback hardware and media players. This has led to increased demand for the production and consumption of 360º videos and major platforms, like YouTube, Facebook, and news platforms such as The New York Times, currently provide 360º videos in their service offerings [1].

This has also driven the development of 360º video players for many different platforms such as desktop computers, smartphones and Head Mounted Displays (HMD's) [3,4]. As for every service, 360º media consumption experiences need to be accessible. Typically, accessibility has been considered in the media sector as an afterthought, and mainly for mainstreamed services.

Within traditional media (such as Television and Movies) there are clear regulations as to how accessibility must be delivered. However, it seems that accessibility for immersive media services is still in its infancy and although some projects, such as the EU funded Immersive Accessibility (ImAc) project [5] have begun to address the need for general accessibility in immersive environments. Their solutions have mainly been to adapt existing accessibility methods, rather than identifying the potential that the new environment can offer. In the case of captioning (often referred to as subtitling) early user trials have shown that the users want what they are used to, rather than what they could have. This means rendering the traditional caption (2 lines, ~30 characters wide) into the users view.

This paper discusses a software framework, designed to allow rapid prototyping of different captioning methods, in order to allow new ideas to be tested quickly and easily across different platforms (including desktop, mobile and HMD's). The tool allows for methods used in existing solutions to be easily contrasted and compared, as well as new ideas quickly implemented for user testing.

II. Background

Currently, there exist no standard guidelines or implementation for captions in immersive videos and although many immersive video players now offer the ability to play 360º media the support for any accessible services is extremely limited. At best the players generally support the implementation of traditional captions fixed within the users view [6].

The British Broadcasting Corporation (BBC) was one of the first research organisations to perform user testing with immersive captions [7]. All of their work was based upon projecting traditional captions into the immersive environment and they evaluated how successful this could be done in scenarios where the captions were:

1. *Evenly Spaced* - captions repeated at 120º intervals
2. *Head-locked* - captions fixed within the users view
3. *Head-locked with lag* - captions follow users view, but only for larger head movements

4. *Appear in front and then fixed* - captions are placed in the position that the user is looking and remain there until they are removed

They found that although it was easy to locate the evenly spaced captions the users much preferred the head-locked options.

A user study, conducted by the ImAc project also identified head-locked captions as the strong preference also identified the need to guide users to the sound source for the caption. To facilitate this requirement location information was added to each caption. This allowed for different guiding modes to be developed, such as a directional arrow which could guide the user to where the person speaking was located. However, this did have the drawback that the location was only specified once per caption, and if a person was moving dynamically during this period, the guide could have been wrong [8].

Within VR captions are now becoming essential to video games. The Last of Us: Part II was released in 2020 [9] with a significant focus given to accessibility. Throughout the game the user has the opportunity to enable and customise captions (such as size, font, whether character name is displayed). It also includes a guide arrow to direct the user to the location of the character speaking.

Rothe et al. [10] conducted tests with fixed captions and compared this presentation mode to head-locked captions. Their result didn't find that one option was significantly preferred over the other. However, in terms of comfort, fixed captions led to a better result even though fixed captions in general mean that the user may not always be able to see the caption as it may be outside of their view.

A W3C Community group [11] focused on developing new standards for immersive captioning recently conducted a community survey to gather opinions. A small group of users with different hearing levels (Deaf, Hard of Hearing, and Hearing) were asked to evaluate each of the identified approaches for captions within immersive environments.

Head-locked was clearly identified as the preferred choice, however it was noted that this was most likely as it replicated the experience that users were familiar with. It was also acknowledged that it was difficult for users to properly evaluate new methods theoretically without the opportunity and content to enable them to be experienced properly. Although all agreed that head-locked should be set as default, other choices should be made available. Other suggestions were made which included changing the font size and colour and number of lines (two lines being the default number). Multiple captions should also be in different positions, each being near to the speaker.

Therefore, the focus of this research is to produce a framework enabling delivery of the full experience of each captioning mode, in an environment where an extensive user study can be conducted.

III. Methods

A. Implementation

Part of the ambition for a framework which is generic enough to enable testing in different environments with a variety of devices is portability. Our implementation is based on web technologies allowing it to be used on any device with a web browser. This includes desktop computers, mobile devices and head mounted displays (HMD's).

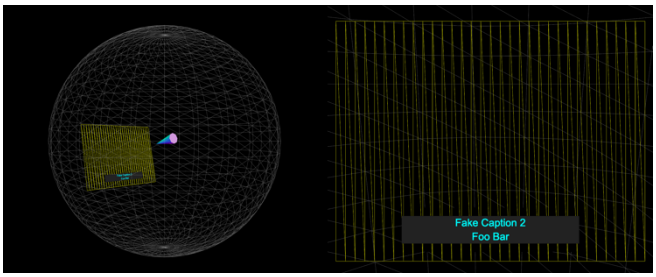

Fig. 1. The initial split view of the player allows the user to see both the caption and view window relative to the 360o world and from the user's perspective

Three.js [12] is a cross-browser JavaScript library and application programming interface (API) used to create graphical processing unit (GPU) accelerated 3D animations using the JavaScript language on the web without the need for proprietary web browser plugins. In our implementation it provides high level functionality to WebGL [13] allowing us to define our scene as objects and manipulate them within the space.

In addition, we use a WebVR Polyfill [14] - which provides a JavaScript implementation of the WebVR specification [15]. This enables three.js content to work on any platform, regardless whether or not the browser or device has native WebVR support, or where there are inconsistencies in implementation. The polyfill's goal is to provide a library so that developers can create content targeting the WebVR API without worrying about what browsers and devices their users are using. This gives our framework maximum compatibility across a large range of devices. Also, as many devices have limited interfaces we add an option to automatically play or pause the video when the user enters or leaves VR modes to avoid the need for a play button if controls are available.

Our framework allows for the user to switch between several different views or enter VR mode. The default view is a split screen as shown in figure 1. This clearly demonstrates how the captions are being rendered and positioned by showing both the user's viewpoint and a representation of the caption and view window within the space.

In order to consume 360º video, the scene contains a *sphere* centred around the origin and it is assumed that the user's viewpoint remains at the origin. When the framework is not connected to a video, the sphere is rendered as a wireframe, however once a video has been loaded the equirectangular video is texture mapped onto the inside of the sphere. As the sphere primitives are generally designed to have a texture mapped to the outside, it is necessary to invert the faces (also known as 'flipping the normals') in order to make this work.

Three.js provides a *videoTexture* object, which can connect to any HTML5 video object. Therefore, an HTML5 video is embedded in the webpage, with its display set to 'none'. The video playback is then managed through JavaScript manipulating the HTML5 video object.

Inside the main scene container we first add a world group. This allows us to reposition the entire contents of the scene to ensure that the user's viewpoint is kept at the origin. For example, when using an HMD the user is automatically given a height which cannot be overridden. Translating the world

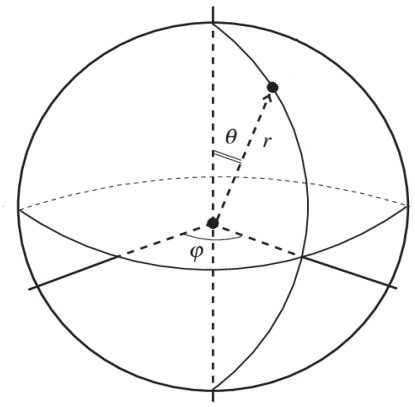

Fig. 2. Spherical Coordinate system user to position the caption target

back to the users eye position allows us to keep their view centred. Within the *world* there are three main components: 1) A video container, 2) a *userView* container and 3) a fixed caption container. The video container is a three.js group which contains the video texture mapped sphere. The *userView* container is a group designed to replicate the behaviour of a camera but which is updated each time the scene is rendered to align with the users' viewpoint. This allows us to always keep the components within the group locked into the users view and it contains a caption container, for placing captions which are fixed into the users view window. Finally, within the *world* there is a fixed caption container which is not updated when the user moves. This allows us to place a caption object into either the *userView* group or the fixed-caption group depending on whether the caption is locked in the scene or the users view.

A wireframe plane is displayed by default in each view and attached to the *userView* to show the user's viewpoint and help provide a coordinated understanding of how the views fit together. The *userView* and the fixed-caption container both contain a pivot point ensuring that as they are rotated around the origin the caption aligns with the video sphere. This allows us to simply position the caption anywhere in the video using a spherical coordinate system and by applying a radial distance (r), polar angle (θ) and azimuthal angle (φ) values which are stored in the caption file, as illustrated in figure 2.

B. Contrast and Compare

Fundamentally, from our review there are two primary mechanisms for caption rendering. 1) *Head-locked* where the caption is rendered relative to the user's viewpoint and 2) *Fixed*, where the caption is rendered relative to a fixed location in the world, generally at the position of the character speaking.

Three.js allows for the textures to be generated from any HTML5 canvas object. In addition to the hidden video our HTML page contains a hidden canvas element which allows us to render any caption using any HTML or CSS styles. This canvas texture is then mapped to a *plane* and positioned into the scene.

An update is triggered every time a video frame changes, and the player checks to see if the caption has changed. If there is a new caption then 1) The canvas is updated to the text and style of the new caption, 2)The texture is updated and 3) the position of the caption is updated. For a *fixed* caption this position is attached to its relative position in the scene and placed within the fixed-caption container, however *head-locked* captions are *userView* object which gets repositioned each time the users' viewpoint is changed.

For each generated caption it is assigned a target location. In the first instance this is the position that is specified in the caption file. This concept was first used in the ImAc project where a single location was stored for each caption in an extended Timed Text Markup Language (TTML) file [16] and the location is defined in spherical coordinates. Within our player the user can enable the target position to be displayed in order to help with debugging, and understanding, however the captions do not necessarily get rendered at this location as the user may have chosen to offset the position, or it may be overridden by the display mode, for example *head-locked* will always render the caption into the users view.

On opening our framework uses a random caption generator to show what is happening in the current display mode. A text string is generated and given a polar position (θ) between -π rad and π rad (-180º to 180º) and azimuthal position (φ) between -0.4 rad and 0.4 rad (~-23º to ~23º) as captions are rarely positioned towards the top or bottom vertical pole.

The user has the opportunity to select from the following default modes:

- *Fixed in Scene, Locked Vertical* - The caption is positioned at the target, but the azimuthal position (φ) is restricted to 0 so that it remains locked to the horizon.
- *Fixed in scene, repeated evenly spaced* - The caption is positioned at the target location then duplicated at 2π/3 rad (120º) intervals around the horizon.
- *Appear in front, then fixed in scene* - The caption is rendered in the centre of the user's current view and remains there until the caption is updated.
- *Fixed, position in scene* - The caption is rendered at the target location.
- *Head-locked* - The caption is rendered in the user's view point and is moved in sync with the user to ensure the caption remains statically attached to the view point.
- *Head-locked on horizontal axis only* - The caption is rendered as *head-locked*, however the azimuthal angle (φ) is restricted to 0, ensuring that the caption is always rendered on the horizon.
- *Head-locked on vertical axis only* - The caption is rendered as head-locked, however the polar position (θ) is locked to the target.
- *Head-locked with lag, animate into view* - The caption is rendered in the *head-locked* position, however as the users' viewpoint changes the caption is pulled back towards the *head-locked* position. An animation loop moves the caption incrementally causing it to smoothly animate into view.
- *Head-locked with lag, jump into view* - This is the same as above, except the animation time is reduced to 0, forcing the caption to jump into the users view.

The framework also allows for the comparison of default guiding modes (as shown in figure 3). These guide modes always direct the user to the target location as this is the source of the identified action. When the captions are fixed they therefore direct the user to the caption, whereas when they are

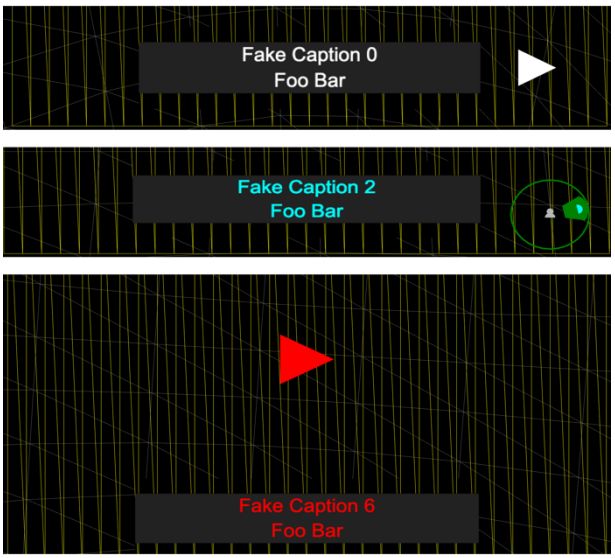

Fig. 3. Guide modes (Top: ImAc Arrow, Middle: ImAc radar, Bottom: Large Arrow)

head-locked the user can read the caption in their view whilst being directed to the target:

- *ImAc Arrow* - An arrow positioned left or right, directs the user to the target.
- *ImAc Radar* - A radar is shown in the users view. This identifies both the position of the caption, and the relative viewing angle of the user.
- *Big Arrow* - a large arrow is displayed in the centre of the users view.

The javascript implementation allows for additional display modes and guide modes to be created quickly by simply creating rules for the caption creation, update and removal.

C. New Methods

Due to the large file size of immersive videos, the player was updated to support both HTTP Live streaming (HLS) [17] using hls.js [18] and Dynamic Adaptive Streaming over HTTP (DASH) [19] streams using dash.js [20]. This massively improves the performance of video playback, where network bandwidth was limited and therefore improves the user experience.

Based on anecdotal feedback from the community additional functionality was added to the player in order to allow further customization. Firstly it was identified that it was necessary to be able to display multiple captions simultaneously. This is because 1) sometimes multiple people are speaking simultaneously and 2) there is a need for captions to remain longer in order to give users time to find and read them.

Our framework was therefore extended to support multiple captions based on a particle system [21] approach. This allows within the framework for the captions to behave as independently - they are created, their mode defined and rules defined for their update and removal. This means that it is possible to have captions of different modes concurrently within a scene. A *captionManager* is used to keep track of each of the captions in the scene and update them where necessary. This allows the user to override their set mode, and handle basic collision avoidance within the scene.

The user is given a choice of how the *captionManager* can remove the captions from the scene. This can be set to either

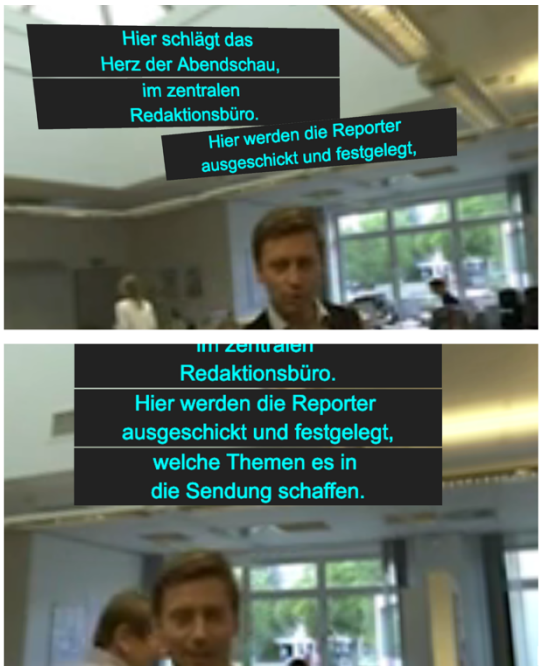

Fig. 4. Captions stacked to avoid collisions (Top: In original position Bottom: realigned to current position)

the time defined in the caption file, a delay can be added, or it can be specified the maximum number of captions to be displayed. In this case the oldest caption is removed once the maximum threshold is reached.

Basic collision detection is used to avoid captions occluding each other. For example when one character is speaking, but previous captions remain within the scene, if an older caption is not moved then the new caption is likely to be drawn over the top. Therefore, the *captionManager* implements a *stacking* system as shown in figure 4 (top). When a new caption is created it is added to a *stack* - where a *stack* has been created for each character in the scene, plus an additional stack for *head-locked* captions. When a caption is added to a stack the *captionManager* iterates through each of the captions in the stack, from newest to oldest increasing their azimuthal position by the existing height of the stack. As only the vertical position is updated, if the character speaking is moving the horizontal position of captions indicate the path the character has taken. However, there is also an option in the interface to force each stack to realign when it is updated, as shown in figure 4 (bottom).

Each of the stacks is grouped, allowing the user to apply an offset on each axis, allowing the entire stack to be repositioned. For example the captions can be moved upwards to place the captions above the person speaking, rather than on top of them.

To support the location of multiple captions the guides were also extended. Each caption object contains its own guide components, so when the ImAc arrow, or ImAc radar mode are enabled in a *head-locked* mode, each caption can display its own guide. As shown in figure 5, in the case of the ImAc arrow, each caption can display its own arrow, however in the case of the radar an additional overlay is added for each caption target. The opacity of each caption in the radar is reduced as the caption gets older.

Additional tools were also added such as a *timecode* display in the view window. This can be fully customized for style and position, in order to help the user understand where

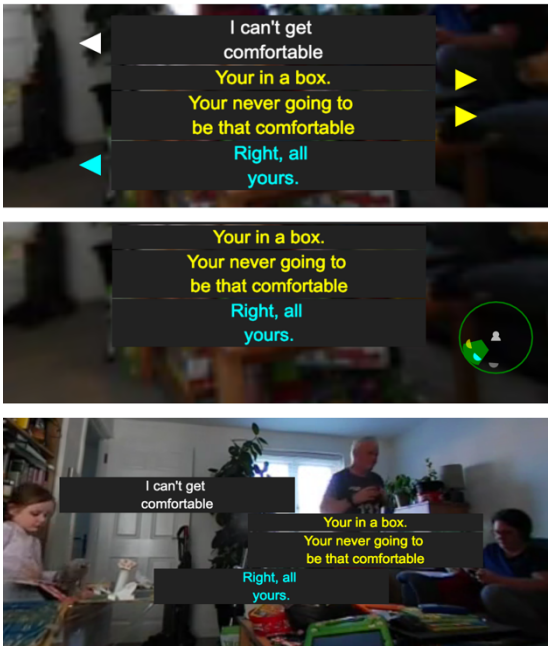

Fig. 5. Enhanced guides extend the ImAc approach to support multiple captions (Top: Arrows, Middle: Radar and Bottom: Justified)

they are temporally whilst immersed, as shown in figure 6. Also parameters such as animation speed, offset position are all exposed to the user through the graphical user interface (GUI). An additional option to lock the caption azimuthally was also added to force the captions to remain on the horizon. This may be helpful to those users who find it difficult to look up and down.

D. *Responsive Captions*

In previous work we developed a JavaScript library for managing responsive captions [22], This library allows for captions to be dynamically restructured into different lengths. This is done by following the principles of text flow and line length, informed by the semantic mark-up along with styles to control the final rendering.

Captions are re-blocked by adhering to the number of characters that can fit into the display container at the chosen font size. Firstly each paragraph is recombined, based on a unique speaker. A best-fit algorithm then breaks each paragraph up to individual captions in order to fit the container. Due to the nature of the changing font size this may provide more or less captions than the originally authored, however as the number of words remains the same the reading speed never changes. As words are evenly distributed it also avoids leaving orphaned words, as shown in figure 7.

This approach is particularly effective when adapting content from traditional television displays into an immersive environment, such as rendering the caption in a speech bubble attached to a character, or for instance if you wish to reduce the width of the caption in order to make room for other captions or graphics, as shown in figure 8.

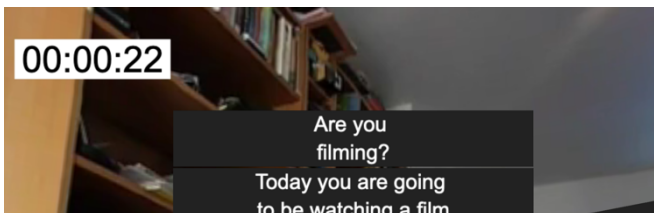

Fig. 6. A customizable timecode can be added to the user's viewpoint.

Fig. 7. The responsive caption library restructuring the length of the captions to a maximum character length: (Left: 25 characters, Right: 12 characters)

E. *Enhanced caption file*

In order to facilitate future experiments, a custom caption file format has been implemented using a JavaScript Object Notation (JSON) structure. We have tools for importing and exporting to IMSC TTML and importing from a text based transcript. Our experimental file contains further information such as tracking information for each character in the scene. This provides a frame by frame position for the location of each person and the target can then be tied to a person or object as they move, rather than just the position they are at when the caption is first created. This allows for both a fixed position for responsive captions as we know a start position for each caption we create, or alternatively for a caption to follow a character through a scene. Where no track information is available for a character, an option is provided to interpolate between one caption target location and the next. This is reasonably successful for when a single character is moving and talking, but breaks when characters change. Therefore, there is also an option to restrict the interpolation to a single character and not include the next characters start location.

Currently, the additional track information is created manually, by defining keyframes and interpolating between them. Our framework provides a basic editor for adding the track information using a keyboard interface, shown in figure 9. In future work we will explore how computer vision techniques can be used to automatically identify the position of characters within the scene.

IV. Results and Discussion

Testing technology for usability with end users is a standard prerequisite in the development workflow. When the technology designed is related to accessibility services is a

Fig. 8. The responsive subtitles library rendering captions as speech bubbles

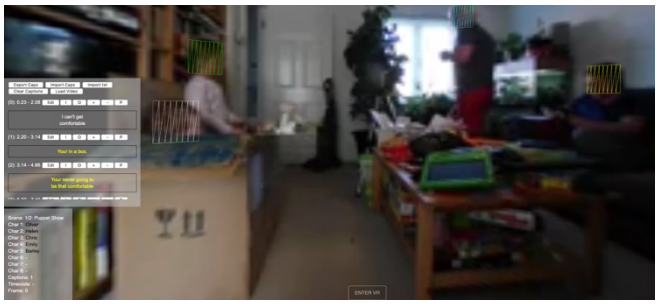

Fig. 9. The experimental caption files contain tracked location for each character and the framework provides a basic editor for creating the caption files

must. The reason is the context where any accessibility service is developed: the UN Convention of Rights of Persons with Disability (2006) with the motto “nothing about us without us”. This leads to a user-centric approach where end users express their needs and expectations. (If needed I can write more here “towards personalisation” rather than one size fits all). Experience gained from the 3-year research project (ImAc) [1, 23] showed that: 1) end users need to have real stimuli to comment, 2) testing cycles should be shorter, and to these we add 3) the new COVID-19 reality and face to face testing.

The proposed framework meets previous three issues 1) with end users having the stimuli in a real VR simulation. This is a step forward to paper prototyping which was used in ImAc. The reason for paper prototyping in ImAc was the fact that no 360º caption editor existed at the time. It was one of ImAc's objectives to develop one. Hence, the very first user requirements were generated in paper [24, 25], which might have impacted the decisions taken towards testing and further developments [8]. 2) Two reasons led to lengthy testing cycles. The first was the process of generating stimuli with different variables, since it was produced as independent 360º movies, not web based. In ImAc stimuli definition and production meant: a democratic choice of content, which then was captioned with the editor, then translated to the languages, and finally tested. The second was the number of end users required for each test [24, 25] this issue is related to the new world health context. COVID-19 has forced all communication based industries to consider existing communication technology as alternative to traditional media content production and distribution. Testing end users for IT system development is one of the many activities that needs to be redefined under the new situation. The silver lining is that with a framework as the one proposed here online testing is a reality. From the comfort of their home, and following all government health and safety regulations, end users can access stimuli from any device.

V. CONCLUSION

The need to generate a user friendly open testbed for 360º captioning is a much needed tool for both industry and academia. Previous research showed the shortcoming of paper prototyping for VR leading to a first framework development. Results from the first framework allowed for fast simulation of variables. This in turn showed the need to develop further, to allow for some unforeseen conditions which had not been considered during the previous 3 year research. The development of this second framework will allow to start a set of online tests, with many more advantages than those predicted by intuition, based on paper prototype. The number of end users and the geographical location will increase given the easy online access. The cost of testing in terms of money and time will be shed drastically. The visualisation will elicit new ideas for displays, it will also discover different needs hence higher personalisation. The framework is designed to allow structured tests to be administered following a set path which allows the test designer to integrate with questionnaires, ethical permission, and a faster processing of results.

ACKNOWLEDGMENT

The content of this article has been enriched by the comments from the W3C Immersive captions CG. This framework will be tested in H2020 957252 MEDIAVERSE, H2020 870610 TRACTION, and H2020 870939 SO-CLOSE.

REFERENCES

[1] Montagud M, Orero P, Matamala A (2020) “Culture 4 all: accessibility-enabled cultural experiences through immersive VR360 content.”, Pers Ubiquit Comput. https://doi.org/10.1007/s00779-019-01357-3

[2] Louis Netter (2020) “The importance of art in the time of coronavirus”, retrieved from: https://theconversation.com/the-importance-of-art-in-the-time-of-coronavirus-135225

[3] Papachristos NM, Vrellis I, Mikropoulos TA (2017) “A Comparison between Oculus Rift and a Low-Cost Smartphone VR Headset: Immersive User Experience and Learning.”, 2017 IEEE 17th International Conference on Advanced Learning Technologies (ICALT), Timisoara, 2017, pp. 477-481. doi: 10.1109/ICALT.2017.145

[4] Srivastava P, Rimzhim A, Vijay P, Singh S, Chandra S (2019) “Desktop VR Is Better Than Nonambulatory HMD VR for Spatial Learning.”, Front. Robot. AI 6:50. doi: 10.3389/frobt.2019.00050

[5] M. Montagud, I. Fraile, J. A. Núñez, S. Fernández (2018). “ImAc: Enabling Immersive, Accessible and Personalized Media Experiences.” ACM TVX 2018, Seoul (South Korea)

[6] Chris Hughes, & Mario Montagud. (2020) “Accessibility in 360-degree video players.”, https://arxiv.org/abs/2005.03373

[7] Brown A, Turner J, Patterson J, Schmitz A, Armstrong M, Glancy M (2017). “Subtitles in 360-degree Video.”, In Adjunct Publication of the 2017 ACM International Conference on Interactive Experiences for TV and Online Video (TVX ’17 Adjunct). Association for Computing Machinery, New York, NY, USA, 3–8.

[8] Hughes CJ, Montagud M, tho Pesch, P (2019) “Disruptive Approaches for Subtitling in Immersive Environments.”, Proceedings of the 2019 ACM International Conference on Interactive Experiences for TV and Online Video – TVX ’19. doi: 10.1145/3317697.3325123

[9] Myers, Maddy (June 12, 2020). "The Last of Us Part 2 review: We're better than this". Polygon. Vox Media. Archived from the original on June 12, 2020. Retrieved July 8, 2020.

[10] S. Rothe, K.Tran, H. Hussmann (2018), “Positioning of Subtitles in Cinematic Virtual Reality”, ICATEGVE 2018 - International Conference on Artificial Reality and Telexistence and Eurographics Symposium on Virtual Environments, November 2018

[11] W3C Immersive Captions Community Group, https://www.w3.org/community/immersive-captions

[12] Three.js (2020) GitHub, https://github.com/mrdoob/three.js

[13] WebGL 2.0 Specification (2020), retrieved from https://www.khronos.org/registry/webgl/specs/latest/2.0/

[14] WebVR Polyfill (2020) GitHub, https://github.com/immersive-web/webvr-polyfill

[15] WebVR 1.1 Specification (2017), retrieved from: https://immersive-web.github.io/webvr/spec/1.1

[16] TTML Profiles for Internet Media Subtitles and Captions 1.0.1 (IMSC1), W3C Recommendation 24 April 2018, retrieved from: https://www.w3.org/TR/ttml-imsc1.0.1/

[17] Pantos, R., May, W. (2017) "Playlists", HTTP Live Streaming. IETF. p. 9. sec. 4. doi:10.17487/RFC8216. ISSN 2070-1721. RFC 8216.

[18] HLS.js (2020) GitHub, https://github.com/video-dev/hls.js/

[19] Spiteri, Sitaraman and Sparacio (2018), “From Theory to Practice: Improving Bitrate Adaptation in the DASH Reference Player”,ACM Multimedia Systems Conference, June 2018

[20] DASH.js (2020) GitHub: https://github.com/Dash-Industry-Forum/dash.js

[21] Reeves, William (1983) “Particle Systems—A Technique for Modeling a Class of Fuzzy Objects”, ACM Transactions on Graphics. 2 (2): 91–108.

[22] Hughes, CJ , Armstrong, M, Jones, R and Crabb, M (2015), “Responsive design for personalised subtitles”, in: The 12th Web for All Conference, 18-20 May 2015, Florence, Italy.

[23] Agulló B, Montagud M, Fraile I (2019), “Making interaction with virtual reality accessible : rendering and guiding methods for subtitles.”, AI EDAM (Artificial Intelligence for Engineering Design, Analysis and Manufacturing) doi: 10.1017/S0890060419000362

[24] Agulló B, Matamala A, and Orero P (2018), “From Disabilities to Capabilities: testing subtitles in immersive environments with end users”, HIKMA 17: 195-220. doi: 10.21071/hikma.v17i0.11167

[25] Agulló B, Matamala A (2019), “The challenge of subtitling for the deaf and hard-of-hearing in immersive environments: results from a focus group”, The Journal of Specialised Translation 32, 217–235 http://www.jostrans.org/issue32/art_agullo.php