Anahita: A System for 3D Video Streaming with Depth Customization

Kiana Calagari[1,3] Krzysztof Templin[2] Tarek Elgamal[1] Khaled Diab[3] Piotr Didyk[2]
Wojciech Matusik[2] Mohamed Hefeeda[1]

[1]Qatar Computing Research Institute, Doha, Qatar
[2]Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA
[3]School of Computing Science, Simon Fraser University, Burnaby, BC, Canada

ABSTRACT

Producing high-quality stereoscopic 3D content requires significantly more effort than preparing regular video footage. In order to assure good depth perception and visual comfort, 3D videos need to be carefully adjusted to specific viewing conditions before they are shown to viewers. While most stereoscopic 3D content is designed for viewing in movie theaters, where viewing conditions do not vary significantly, adapting the same content for viewing on home TV-sets, desktop displays, laptops, and mobile devices requires additional adjustments. To address this challenge, we propose a new system for 3D video streaming that provides automatic depth adjustments as one of its key features. Our system takes into account both the content and the display type in order to customize 3D videos and maximize their perceived quality. We propose a novel method for depth adjustment that is well-suited for videos of field sports such as soccer, football, and tennis. Our method is computationally efficient and it does not introduce any visual artifacts. We have implemented our 3D streaming system and conducted two user studies, which show: (i) adapting stereoscopic 3D videos for different displays is beneficial, and (ii) our proposed system can achieve up to 35% improvement in the perceived quality of the stereoscopic 3D content.

Categories and Subject Descriptors

H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*video*; I.3.3 [**Computer Graphics**]: Picture/Image Generation—*display algorithms, viewing algorithms*

Keywords

Video streaming; 3D video; stereoscopic retargeting; depth optimization

MM '14, November 03 - 07 2014, Orlando, FL, USA.

http://dx.doi.org/10.1145/2647868.2654899.

1. INTRODUCTION

Stereoscopic 3D (S3D) has already become main stream in movie theaters. Many productions are released in stereoscopic format and there is a significant audience that enjoys this more immersive experience [17, 14]. Moreover, S3D is becoming increasingly popular at home. This trend is largely facilitated by widely available stereoscopic 3D content and displays—most of new TV-sets are 3D-ready. Furthermore, enabling stereoscopic viewing on mobile devices is viable and useful for a wide range of applications.

Although S3D is already well established in the movie industry, popularizing it among home and mobile users is still a challenging and unsolved problem. The most challenging problem is content adaptation. This is because both 3D perception and visual comfort highly depend on viewing conditions (e. g., display size, viewing distance) [33, 9, 30, 31, 27]. In the case of movie theaters, the same version of the content can be shown on different screens, as viewing conditions do not vary significantly among them. However, due to the wide range of TV-sets or mobile displays, such an assumption does not hold anymore. In addition, stereoscopic content should be manipulated to account for differences in user preferences. A content that provides a good experience for one user does not necessarily provide a good experience for others. As a result, providing the same content for all displays and users results in sub-optimal quality, both in terms of depth perception and visual comfort. Finally, there is a large variety of different 3D display types. Each of them needs a different video input, e. g., side-by-side, temporal interleaving (shutter glasses), spatial interleaving (e. g., polarized glasses, autostereoscopic) etc. They also have different capabilities and limitations [6]. This requires further content customization based on the display type.

To address these issues, we propose *Anahita* – a system for 3D video streaming. In contrast to previous systems, it utilizes an additional information about viewing conditions and display technology, and uses it to improve viewing experience by adjusting depth. To the best of our knowledge, this is the first 3D streaming system with such capabilities. Furthermore, the additional information enables a smart bandwidth management, i. e., sending the information that is not ultimately displayed can be avoided. In particular, the contributions of this paper are as follows:

- A new system design for adaptive 3D video streaming: The goal of this system is to optimize stereoscopic 3D videos for a wide range of display sizes, video represen-

tations, viewers' preferences, and network conditions. The system efficiently organizes the creation of various versions of 3D videos using a structure that we refer to as the 3D version tree. The system uses the Dynamic Adaptive Streaming over HTTP (DASH) to dynamically switch among different versions and optimize the quality and depth perception for different viewers.

- A novel method for depth expansion and compression for stereoscopic 3D videos: Our method performs simple image processing operations, it does not require creating accurate depth maps, and it does not introduce visual artifacts. The main target application for our method is sport videos e.g., soccer, football, tennis, and cricket. In such applications, preserving the scene structure, e.g., the straightness of the lines, is extremely important. Our method guarantees to preserve the scene structure.

- A complete end-to-end implementation and two user studies to evaluate its benefits: We implement the server side of our system and deploy it on the Amazon cloud. The server implements our depth customization method as well as several off-the-shelf video processing operations to adapt 3D videos. In addition, we provide multiple clients on 3D-enabled mobile phones, tablets, desktop displays, and large TV displays. We conduct two user studies. The first study shows the need for our system in general. The second study shows the achieved gain in depth enhancements by the proposed system, which was up to 35%.

The rest of this paper is organized as follows. In Sec. 2, we summarize previous efforts in academia and industry in designing 3D video streaming systems. We also describe different 3D content customization methods and how our method is different from them. Then, we present the design of our system (Sec. 3), and the depth customization method (Sec. 4). Sec. 5 discusses details of the implementation. In Sec. 6, we present our user studies which evaluate our system. Sec. 7 concludes the paper.

2. RELATED WORK

There has been a significant interest both in the academia and in the industry in 3D video processing. However, as discussed below, the problem of depth optimization for different displays sizes and types has not received much attention. In addition, in contrast to previous methods, our depth manipulation method does not rely on accurate depth maps, it does not introduce any significant distortions to the videos, and it is computationally inexpensive.

2.1 3D Streaming Systems

Multiple systems for 3D content streaming have been proposed in the literature. For example, Baicheng et al. [35] describe a 3D video streaming system, but their main focus is on the 3D media encoder and decoder. Whereas Diab et al. [11] focus on optimizing the storage in 3D streaming systems. Pehlivan et al. [26] propose a video streaming system that switches between 2D and 3D videos depending on the available bandwidth and display equipment. The work by Wu et al. [34] adapts 3D video quality and balances/trades off the temporal and spatial (color plus depth) quality in real-time, but it does not enhance or customize the depth signal for different display types/sizes/technologies. Our work could potentially leverage their method to provide smarter rate adaptation in our 3D streaming system. 3D teleconferencing has been also proposed. For example, early work [19] focused on extending the transport protocol to associate left and right views. Whereas the more recent ViewCast [37] enables multi-party 3D tele-immersion and prioritizes stream transmissions based on the client's viewing angle. Similarly, Telecast [5] prioritizes streams based on viewing angles, but supports a large number of non-interactive viewers, using a hybrid P2P+CDN architecture. Depth customization is not addressed in these works. In addition, multiview client-server systems, where a scene can be displayed from different viewpoints has been considered, for example in [22], [20] and [15].

In addition to the academic works mentioned above, there has been significant interest from the industry, including YouTube, 3DVisionLive, Trivido, and 3DeeCentral. YouTube [4] supports multiple 3D formats including anaglyph (red-cyan, blue-yellow or green-magenta), side by side, row and column interleaved. In addition, it supports HTML5 stereo view, which is the format for active-shutter displays that utilizes NVIDIA 3D Vision. Unlike our system, YouTube does not change or customize the depth of videos for different displays. 3DVisionLive [2] is a web channel for 3D video streaming and 3D photo sharing. It uses the Microsoft Silverlight and IIS smooth streaming technologies. It is, however, limited to one display technology. Trivido [3] is a 3D Internet video platform. It supports anaglyph, side by side, row interleaved 3D formats, and the 3D NVIDIA Vision format. However, it does not address the problem of customizing the content for different displays. 3DeeCentral [1] supports 3D content on multiple 3D-enabled devices, but it does not provide adjusted depth for different displays.

In summary, previous works on 3D streaming have addressed various problems including content acquisition, content representation, encoding, storage, and transmission of both stereoscopic and multiview content. However, we are not aware of any 3D streaming system that adaptively customizes the depth based on the display size, display technology, and viewer's preferences.

2.2 Depth Customization

There are two basic requirements stereoscopic content has to meet in order to be comfortable to watch. First, every object in the scene needs to fit within the "comfort zone", i. e., it cannot be too far from the screen plane [27]. A rule of thumb used by stereographers (*percentage rule*) is that the pixel disparity should not be greater than 2–3 percent of the screen width [27]. Second, the relative distance (in z-direction) between nearby objects (in xy-plane) cannot be too large, i. e., the disparity needs to be within the limit known as Panum's fusional area [8]; otherwise, one of the objects will not be fused by the observer. Meeting those constraints is dependent on two stereoscopic parameters that influence the depth distribution of a stereo shot – the camera separation and the convergence. The former influences the *range* of depth (and thus distances between the objects), whereas the latter influences the *placement* of that range relative to the screen plane. Oscam et al. [23] developed a method for real-time optimization of these parameters, however, it is limited to synthetic content, and cannot be applied as a post-process.

The convergence is relatively easy to modify in post-production by simply shifting either one or both views. Fixing the camera separation, however, is more difficult, since it requires synthesizing new camera views. Lang et al. [21] show how this, and even more sophisticated operations, can be accomplished via nonlinear disparity mapping. Unfortunately, their method relies to a large extent on stereo correspondences, and it requires recovering pixel values for points that have not been registered by the camera. Therefore, it is difficult to reliably generate clean, artifact-free output without manual intervention, not to mention real-time performance. Furthermore, nonlinear disparity mapping can severely degrade the quality of videos of field sports such as soccer, due to objectionable curving of the lines.

Depth can also be manipulated as a consequence of depth compression performed to limit the bandwidth of 3D content. Although some of these techniques can adapt to different viewing conditions [34, 24], their primary goal is to maintain the original depth. In contrast, our technique intentionally modifies the depth to enhance viewer experience.

Our depth customization method is a simple, yet efficient post-production technique that can expand or compress the depth in stereo content. It targets depth customization in field sports such as soccer, football, tennis, and cricket. Our method preserves lines and planes, and thus it maintains the structure of the scene, which is important in sports videos.

3. SYSTEM ARCHITECTURE

The main goal of the proposed 3D streaming system, called Anahita[1], is to provide depth-optimized videos to different 3D display types and sizes. As shown in Fig. 1, the proposed system consists of: (i) server that processes the content and creates a customized version for each display based on its size and technology, and (ii) multiple clients from which 3D devices connect to the server and receive the suitable versions. The following subsections provide more details.

3.1 Server Side

The server adjusts the depth of 3D videos and adaptively streams them to clients. It also manages the creation and storage of different versions of the 3D videos. Specifically, the server has four main components: (i) Depth Enhancer, (ii) 3D Version Manager, (iii) Storage Manager, and (iv) Adaptive Streamer.

The Depth Enhancer customizes the depth of 3D videos based on the target clients. It can either increase or decrease the amount of depth to be perceived in the video. This is done using simple image processing operations that can be performed in real time as a post-processing step (Sec. 4).

The 3D Version Manager component handles the creation of different versions of the same 3D video. This is done through what we call 3D Version Tree, which is shown in Fig. 2. We note that current 2D streaming systems, e. g., YouTube, typically store a *few* (2–4) versions of the same 2D video, but at different qualities (bitrates) to accommodate the variability in network bandwidth of different clients. The number of versions in the proposed 3D streaming system needs to be much larger than traditional 2D streaming systems. The classification of 3D displays presented in [11] suggests that more than 35 versions may be needed to accommodate current 3D displays. This is because, in addition to the variability of network bandwidth, clients requesting 3D videos use different displays in terms of size, depth rendering method, and the amount of depth that can be perceived.

The 3D version tree in Fig. 2 manages the creation of all different 3D versions. Specifically, the input video is assumed to be in the side-by-side format, which is currently the most commonly used 3D representation. Our depth enhancement method creates up to D versions with different depth values, where D is the number of different display sizes supported by the system. These D versions represent the first level of the version tree. In this level, all versions are still in the side-by-side format. In the second level of the tree, we apply various video conversion methods to support displays with different depth rendering technologies, which include anaglyph, frame interleaved, row interleaved, column interleaved, and video plus depth. In our system, we implement conversion methods known from the literature. For each display size in level 1 of the tree, up to R versions can be created in level 2, where R

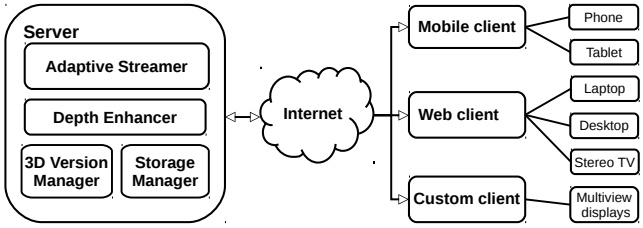


Figure 1: Design of the proposed 3D streaming system. The Depth Enhancer customizes the depth based on display sizes and viewing conditions. The creation of multiple versions of each 3D video is managed by the 3D Version Manager, and the storage of these versions is handled by the Storage Manager. The Adaptive Streamer uses DASH to serve segments of different versions to various clients.

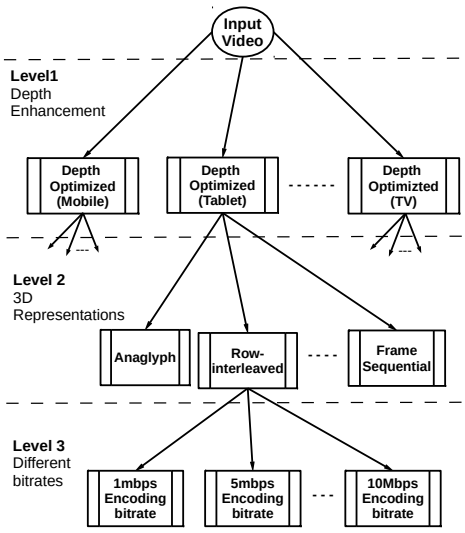


Figure 2: 3D Version Tree. It serves as an execution or master plan to create different versions of each 3D video.

[1]In ancient times, Anahita was the source of all clean water streams that flowed through golden channels to all lands and oceans on Earth. In the Internet age, Anahita is the source of all high-quality 3D video streams that flow through network channels to all types of displays.

is the number of different 3D video representations supported by the system. In the third level of the 3D version tree, we create up to B versions for each 3D representation in level 2 to accommodate clients with heterogeneous and varying network bandwidth, where B is the number of different bitrates (qualities) offered by the system. We note that the depth enhancement method is applied in the first level of the tree because it is more expensive than other operations. Since the first level handles only a few number of display sizes, the number of times that the depth enhancement method is invoked is minimized. In addition, the depth enhancement method should be applied on the original, side-by-side, video in order to create consistent and optimized depth perception regardless of the different 3D video representations.

The third component of our system is the Storage Manager, which organizes the storage of different versions. Notice, that the 3D version tree can have up to $D \times R \times B$ different versions for each video to support optimized 3D video quality on displays with different sizes, 3D video representations, and dynamic network conditions. With current display technologies and sizes, the parameters D, R, and B are roughly in the ranges $[4-6], [5-7]$, and $[2-4]$. That is, more than 100 versions of the same 3D video could be needed to support the large diversity of clients. Creating and storing all versions for all videos may waste storage and processing resources of the system, especially for videos with low popularity. A recent work [11] studied this problem and proposed an algorithm to store some versions while creating others on demand, based on video popularity, video size, and processing requirements to create each version. This algorithm, or similar ones, can be used in our system.

The final component of our system is the Adaptive Streamer, which adopts the Dynamic and Adaptive Streaming over HTTP (DASH) protocol [18, 29, 28]. DASH enables the server to scale to many concurrent sessions, uses off-the-shelf HTTP servers, facilitates switching among different versions, and provides wide client accessibility because it uses the standard HTTP protocol which is allowed in most networks. The server side of the DASH protocol divides a video into a sequence of small segments. Current 2D streaming systems that use DASH create few versions of each segment at different bitrates. In our 3D streaming system, however, we create different versions of each segment using the 3D version tree to adapt not only to the network bandwidth, but also to different display sizes and technologies.

In addition to the depth customization feature that supports different 3D displays, our 3D streaming system offers another feature: *depth personalization*, which is adjusting the depth of a video based on the preferences of individual viewers. This is a desirable feature, as it improves the engagement of viewers and allows them to choose the most visually comfortable depth perception. This feature is realized as follows. The user interface at the client side displays multiple depth options. The default selection is our depth-optimized version, created for the viewer's display. The viewer is allowed to choose other versions with more or less depth. After selection, the client side translates this request to the corresponding segment ID and submits it to the DASH server.

3.2 Client Side

As shown in Fig. 1, the client side of our streaming system supports various platforms, including: (i) mobile, (ii) web-based, and (iii) custom. All three platforms implement the client side of the DASH protocol to adaptively request segments of the different versions of 3D videos stored at the streaming server. Specifically, the client starts by requesting the manifest file from the server, which contains meta data about the requested 3D video and its available versions. Then, it decides on the most suitable 3D version based on the display characteristics and the current network conditions. The client then starts requesting videos segments from the server and the media player is notified to start playing these segments.

For the mobile client, we developed an application to work on various mobile devices such as smartphones and tablets. Most 3D-enabled mobile devices use autostereoscopic displays, which do not require glasses. Rendering of 3D content on these displays depend on five main parameters including: pitch (spatial frequency or the period of the parallax barrier), duty cycle (parallax barrier ratio of opacity to transparency), optical thickness (distance between the parallax barrier and the display), angle (barrier degree of rotation from vertical), and shift (horizontal offset of the parallax barrier relative to the display). Since different manufacturers can choose different values for these parameters, we built a mobile client to adjust the content according to the display characteristics provided by the manufacturer. We designed the mobile client application to have two parts. The first (lower) part calls specific APIs supplied by the manufacturer of the mobile device. The second part is independent of the manufacturer and implements all high-level functions such as the adaptive streaming.

The web-based client uses standard web interfaces to render the encoded 3D video (e.g., side-by-side or frame interleaved) and the 3D display as well as its associated glasses (if any) create the 3D perception to the viewer. Finally, there are displays that require special handling of 3D videos, such as the multi-view displays. Such displays usually require the 3D video in the form of video-plus-depth (2D images and the associated depth map). From the video-plus-depth input, view synthesis methods are used to create multiple virtual views. In our system, we implemented a custom client for such displays, which includes a component to generate depth maps that are needed for the view synthesis method.

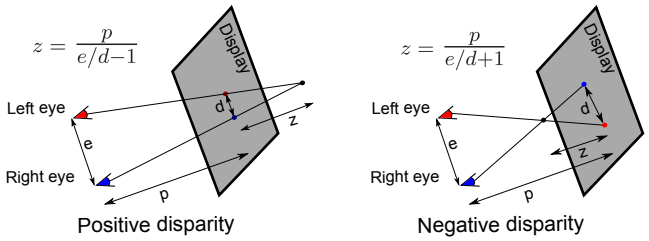


Figure 3: Disparity (d) vs. perceived depth (z). p is the viewing distance, and e is the inter-ocular distance.

4. DEPTH CUSTOMIZATION

In order to perceive binocular stereopsis, we need to display a different view to each eye. The two views show the same scene but from two slightly different viewing points. For each pixel in the left view, its corresponding pixel in the right view is located a few pixels away. Given a pair of corresponding pixels, the signed distance $d = x_r - x_l$ between their x-positions is called *disparity*. The disparities are detected

by the human visual system and interpreted as depth. As shown in Fig. 3, an object is perceived behind the screen plane if the disparity is positive, and in front of the screen plane if the disparity is negative. In the special case of zero disparity, the object appears on the screen plane. The amount of perceived depth (z) is a function of disparity (d), viewing distance (p), and inter-ocular distance (e). Perceived virtual depth, both for positive and negative disparities, is commonly approximated as in Fig. 3, with larger disparities resulting in perception of larger distances [16].

In addition to viewing conditions, the perception of depth varies from person to person [10, 12], and is a matter of preference. Thus, using the same content in all situations is sub-optimal, and clearly there is a need for techniques enabling customization of the depth distribution in 3D videos. In the context of streaming sports videos, such techniques need to meet three requirements: (i) they need to work as a post-process, since we do not have influence on the recording process, (ii) they need to be fast, and (iii) they need to automatically produce high quality results, without objectionable artifacts. We propose a new method of depth expansion/compression that meets these requirements. It targets videos of various field sports such as soccer, football, and tennis.

4.1 Structure Preserving Scene Shifting

Adjusting depth is not a straightforward task. As discussed in Sec. 2, it can be achieved using a disparity remapping. This is, however, a viable option only in off-line applications with some form of supervision. The only safe automatic adjustment for general scenes is convergence manipulation, which can be easily performed using a horizontal shift of the two views. We observed, however, that for some scenes, especially in sports videos, the geometry has approximately planar structure. In such cases, disparity maps can be well described by a single disparity gradient $g = (g_x, g_y)$.

This observation led us to devise a Structure Preserving Scene Shifting (SPSS) method for depth expansion/compression, which adjusts the depth range of the scene by means of 2D geometric transformations. The basic idea behind the SPSS method is to estimate the disparity gradient g and adjust its magnitude. The gradient is estimated by fitting a plane to the scene's disparity map which is obtained using a stereo-correspondence algorithm. In contrast to the depth remapping approach, we use stereo correspondences only to estimate the single depth gradient, hence the accuracy of the correspondences is not critical.

Modification of the gradient is achieved via a remapping operation, in which a parallelogram-shaped region of the input image is selected and mapped back onto the whole image area. Such a mapping can be applied to one of the views or both, and in the latter case, the depth modifications caused by each transformation will add up. To minimize visibility of the distortion, we split the desired depth adjustment between the two views, so that each mapping of one view is accompanied by a complementary mapping of the other. The top and bottom edges of the mapping parallelogram are always kept horizontal, and its shape is described by a combination of two parameters: the *slant* and the *stretch*.

The *slant* operation regulates the skewness of the mapping region by horizontally moving its top and bottom edges in opposite directions. This operation modifies g_y. An example is given in Fig. 4. The *stretch* operation re-scales the mapping region horizontally. The stretch is done linearly to assure that any plane structure remains planar. This transformation modifies g_x. An example is given in Fig. 5.

Following the convention of the percentage rule, we express the amount of slant and stretch as a fraction of the image width, and denote it as σ_{sl} and σ_{st}, respectively. Additionally, a *shift* operation, that moves the mapping region to the left or right can be used, to adjust the convergence. Depending on the direction of the shift, the scene disparities are uniformly increased or decreased, and as a result, the scene appears closer or farther from the user. As previously, we express the size of the shift as a fraction of the image width, and denote it as σ_{sh}.

Assuming, that image x- and y-coordinates are in the range $[-\frac{1}{2}, \frac{1}{2}]$, the three operations are combined into one operator, mapping a pair of old coordinates (x, y) to a pair of new coordinates $(\hat{x}, \hat{y})$ as shown in Eq. 1. To accommodate slanting, stretching, and shifting, the mapping region has to be slightly smaller than the input image, therefore the factor r (typically 0.95) is used to re-scale the coordinates. Recall, that the depth transformation is split equally between the two views of the stereo image, hence the factor ± 0.5.

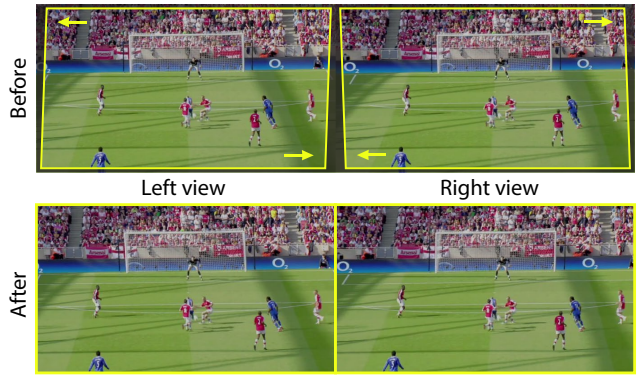


Figure 4: Example of the slant operation. A parallelogram-shaped region is mapped back onto the whole image area, modifying the vertical component g_y of the disparity gradient g. Note, how the opposing slants in the left and right view complement each other to minimize the distortion.

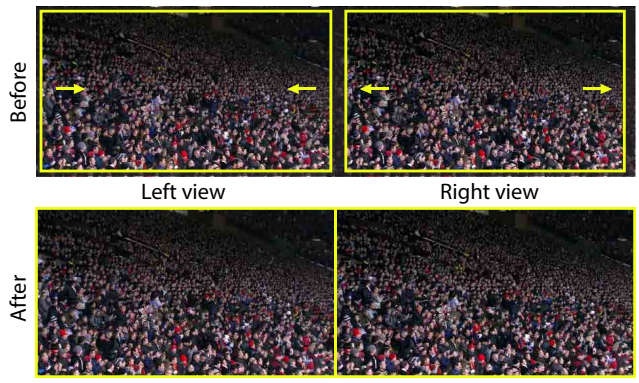


Figure 5: Example of the stretch operation. The left and right views are horizontally scaled. In effect, the horizontal component g_x of the disparity gradient g is changed.

$$(\hat{x}, \hat{y}) = (x \pm 0.5 \cdot (\sigma_{\text{sl}} \cdot y + \sigma_{\text{st}} \cdot x + \sigma_{\text{sh}}),\ y) \cdot \frac{1}{r}. \qquad (1)$$

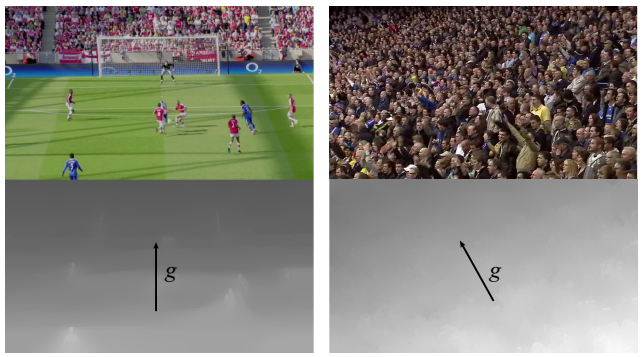
Formally, for any scene classified as suitable for the SPSS method, the following steps are taken:

1. The gradient $g = (g_x, g_y)$ of the scene's disparity map is calculated.

2. The slant and stretch are calculated as follows, where e is the *expansion value*, dependent on the viewing conditions and the content.

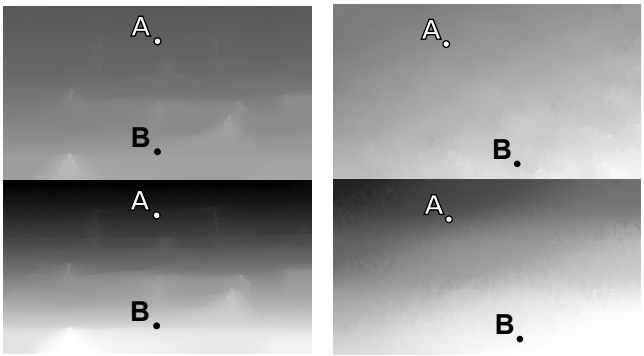
$$\sigma_{\text{sl}} = e \cdot \frac{g_y}{|g_x| + |g_y|} - g_y, \quad \sigma_{\text{st}} = e \cdot \frac{g_x}{|g_x| + |g_y|} - g_x,$$

3. The shift σ_{sh} is set in such a way that there are no negative disparities on the edges of the image.



(a) (b)

Figure 6: Examples of shots in a soccer video that have planar depth structure. The depth maps were determined using optical flow estimation methods [32, 7], and were further enhanced by cross bilateral filtering [25]. Vectors in the maps visualize depth gradients. Note, that they are provided for visualization purposes, and our method does not require computationally expensive estimation of accurate, dense stereo correspondences.



(a) (b)

Figure 7: Our method has increased the disparity range while preserving the depth structure and orientation (the top images are the original disparity maps, while the bottom ones show depth after our optimization.)

4. In order to maintain temporal coherency, $\sigma_{\text{sl}}, \sigma_{\text{st}}$, and σ_{sh} are temporally smoothed, using the history of their values in the n previous frames.

5. The views are remapped according to Eq. (1) using linear interpolation.

Ekin et al. distinguish four types of camera shots in soccer games: long shots, medium shots, close-ups, and out-of-field shots [13]. Long shots provide a global view, in which the field takes most of the screen-space, and multiple small player silhouettes are visible. Medium shots show a smaller portion of the field, usually with couple larger silhouettes, while close-ups show the above-waist portion of one player. Finally, out-of-field shots show the audience, coaches, etc.

Close-ups, out-of-field, and medium shots usually have quite complicated geometry. Long shots, however, are different, because their geometry can be very well approximated by a plane, therefore, they are perfect candidates for the SPSS (see Fig. 6a for an example of a long shot). Occasionally, some medium or out-of-field shots, such as the one shown in Fig. 6b, are also well fitted by a plane, and thus can benefit from the proposed depth adjustment. In the evaluation section, we analyze the shot types in different sports videos and show that the proposed SPSS method can cover a significant portion (60–70%) of the shots in field sports.

Fig. 7 shows the depth maps of the two samples from Fig. 6, before and after SPSS. It can be seen that SPSS enhances the depth contrast, while preserving the original scene structure. In both samples, the depth difference between points A and B has increased, while, the direction of the depth gradient has remained unchanged. In the following, we focus exclusively on long shots, which are the most important use case of SPSS. In Sec. 6, we describe a perceptual study, in which optimal expansion values are found.

4.2 Shot Classifier

Our depth adjustment technique assumes that the scene can be well approximated by a plane. This makes the technique suitable for long shots. Therefore, in order to decide which frame can be manipulated, the type of a shot needs to be determined.[2] Although some techniques performing this exist [13], our scenario is specific. First, we need to detect long shots only. Second, if the classification reports false positives our method should provide acceptable quality. Moreover, previous techniques considered only a single image as an input, whereas our classification can take an advantage of having stereoscopic version of the content.

To detect shots suitable for our technique, we estimate how well a scene can be approximated by a single plane. To this end, we first recover depth using [36]. Next, we fit a plane to it using least squares method and compute the coefficient of determination (R^2) to measure its goodness. To detect scenes suitable for our depth manipulation we construct a binary classifier, which classifies a scene based on the R^2 value, i. e., if the goodness is above a certain threshold q, the scene is considered suitable for our manipulations. Otherwise, it remains unchanged. In order to determine a good threshold q, we processed 1,015 randomly chosen frames from one soccer game and classified them manually into two groups: long shots and the rest. Then, we analyzed how our binary

[2] A shot in our definition is just a pair of images, and we do not need any shot boundary detection algorithms.

classifier performs for different threshold values using the relation between true positive and false positive ratios. Fig. 8 presents the ROC curve. Based on the analysis we chose $q = 0.693$, which gives true positive ratio = 0.8981, and false positive ratio = 0.1925.

5. SYSTEM IMPLEMENTATION

The server-side operations of our system have been developed using C++ and OpenCV. The system is implemented in an extensible and modular manner. Specifically, all video processing operations, e.g., depth expansion, frame interleaving, etc., are implemented as independent components with standard interfaces. This enables us to chain any sequence of these video operations to create different 3D versions. The sequence of operations to create any 3D version is pre-defined using our 3D version tree as illustrated in Fig. 2. Only one 3D version tree is maintained in the system, which is consulted upon the creation of any 3D version. For example, to serve a viewer using a stereoscopic 3D TV with polarized glasses, the system creates the corresponding version by performing three operations: depth optimization, row-interleaving, and scaling. We note that to create a version in Level 3 in the version tree (Fig. 2), the ancestor versions at Levels 1 and 2 have to be created first. The system is designed to have the most time-consuming operations in Level 1. Therefore, these time-consuming operations are executed once instead of being executed in every branch of the tree.

In our prototype, we implemented the anaglyph, row-interleaving, column-interleaving, frame-sequential, depth optimization, and depth estimation operations. Additionally, we implemented some auxiliary operations, such as scale, which resizes the images uniformly, and split, which splits a side-by-side image into separate left and right images.

During initialization, the system decides which versions to create from each 3D video in the system. This decision depends on the storage requirements and available processing capacity to create any missing version on demand. The chosen versions are not necessarily leaf nodes, but they can be at any level of the 3D version tree. To create a version, the system recursively traverses up the 3D version tree starting from the node corresponding to that version until it reaches an already-created version or it hits the root node (the original 3D video). The system pushes the non-created versions to a stack. Then, it creates these versions using the order in the stack. We note that this version creation process is highly parallelizable, and a separate thread can be spawned for each version. However, multiple versions may share an ancestor that should be created. To avoid multiple creations of the same version, we chose to parallelize Level-1 branches. With this setup, one thread is spawned for each branch, and versions inside each branch are generated in series.

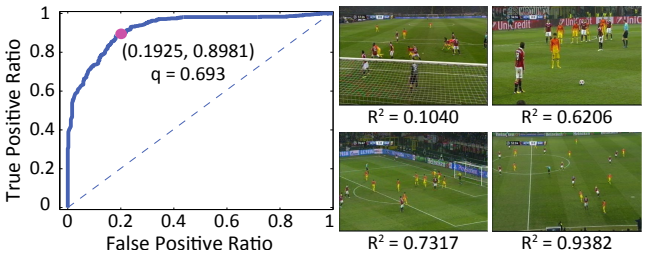


Figure 8: Left: ROC curve for our binary shot classifier. Right: Four examples of shots with corresponding R^2 values. Top two have low R^2 value, therefore, their depth will be unchanged. On the other hand, bottom two are long shots and due to the high R^2 value (above the threshold) their depth will be modified.

At the client side, we have implemented three applications. First, a web application using HTTP and DASH-JS, which is a JavaScript DASH library for the Google Chrome browser. Second, a mobile application for the autostereoscopic mobile devices using Java and the Android SDK. We tested the application on HTC and LG 3D smartphones, and on a Gadmei 3D tablet. Third, we implemented a MS Windows application for a multiview display (Dimenco 55" TV). We implemented the DASH client using Java. The application has been developed to perform view-synthesis before presenting the content to the multiview display.

6. EVALUATION

We start our evaluation by showing the need for depth customization through a subjective study. Next, we briefly describe our implementation of a complete, end-to-end, prototype of the proposed 3D streaming system. Then, we use this prototype to conduct another subjective study to show that the proposed system significantly improves the perceived depth in 3D videos. Finally, we analyze the percentage of video shots in field sports that can benefit from the proposed depth customization method.

6.1 The Need for Depth Customization

In this subjective study, we displayed 3D videos to multiple subjects using different displays. We manipulated depth with different degrees to study its impact on the subjects.

Subjects: Ten subjects took part in this study. They were all members of a computer graphics group. They had normal/corrected-to-normal vision, and could perceive stereoscopic 3D effect.

Experimental Setup: We tested two trial conditions: smartphone and TV. In the smartphone condition, the content was presented using an LG Optimus 3D MAX P725 phone (the stereo effect is achieved by means of a parallax barrier). The observation distance was about 30 cm. In the TV condition, the content was shown using a 55-inch Sony Bravia XBR-55HX929 TV-set, and a pair of active shutter glasses. The observation distance was about 2 m.

Content: We used a series of short clips taken from the Manchester United vs. Wigan Athletic FA Community Shield game (2–0, 12 August, 2013) as the test sequence. Each clip was a few seconds long, and the entire sequence was 54 seconds long. The initial depth range of the test sequence was small, and the whole image appeared flat.

Protocol: The subjects were shown six versions of the test sequence, with increasing expansion values, in the two above-mentioned conditions. The values used were: −0.5 % (depth compression), 0 % (original sequence, no slant), 1 %, 2 %, 3 %, and 4 % (depth expansion). Half of the subjects saw the TV condition first, and the other half saw the smartphone condition first. In each condition, the subjects were asked to choose the preferred version, assuming that they were to watch the entire 90-minute soccer game.

Results and Discussion: The average preferred values for expansion were: 1.4 % in the TV condition and 2.3 % in the smartphone condition. The results together with

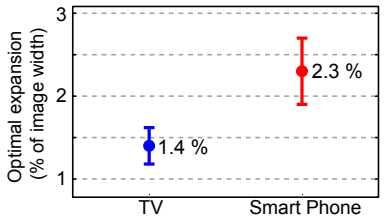


Figure 9: Mean expansion value and standard error of the mean for both conditions.

the standard error of the mean are shown in Fig. 9. The experiment shows that the original 3D videos were not the best choice for subjects. Instead, the depth-manipulated 3D videos scored higher than the original ones. In addition, the level of depth manipulation depended on the used display. Therefore, we can conclude that for optimized viewing of 3D videos on different displays, the depth of the videos needs to be adjusted.

In addition, we use the results of this experiment to set the expansion values in the rest of our evaluation. That is, the expansion values for the TV and smartphone are set to 1.4 % and 2.3 %, respectively. Our experiment in this section tested only two viewing conditions – a smartphone and a TV. We used linear interpolation of the expansion value for the other display sizes used in the experiments in Sec. 6.2.

We assumed that displays are always viewed from a particular, standard distance which depends on the display size. Therefore, the size is the only parameter that our depth adjustment technique accounts for. For non-standard viewing distances, the amount of depth expansion/compression our method needs to perform could be derived in a similar experiment, and the server could adjust the content accordingly also in such cases. The viewing distance could be indicated manually by the user or inferred using an embedded camera.

Existing studies demonstrate that there is a significant variation in depth perception among observers [10, 12]. We did not explicitly analyze how much the ideal depth varies across subjects, however, it is unclear how the system could take such variability into account without some prior knowledge about the user. Therefore, we decided to use by default the mean expansion value, and to provide the user with the possibility of overriding that setting.

6.2 Depth Improvement

We conduct a subjective study to measure the perceptual depth enhancements achieved by our system, for different videos and viewing conditions. The system was deployed on Amazon cloud and fully tested over the Internet. For our subjective studies, however, the experiments were done over a LAN to assess the effectiveness of the proposed depth adjustment method. For handling network dynamics, our system supports encoding the video in different bitrates and it can switch among them in real time using the standard DASH protocol. Our depth adjustment method does not increase the bitrate and it is independent of the rate adaptation method.

Subjects: Fifteen new subjects took part in this experiment. They were all computer science students and researchers, had normal/corrected-to-normal vision, and could perceive stereoscopic 3D effect.

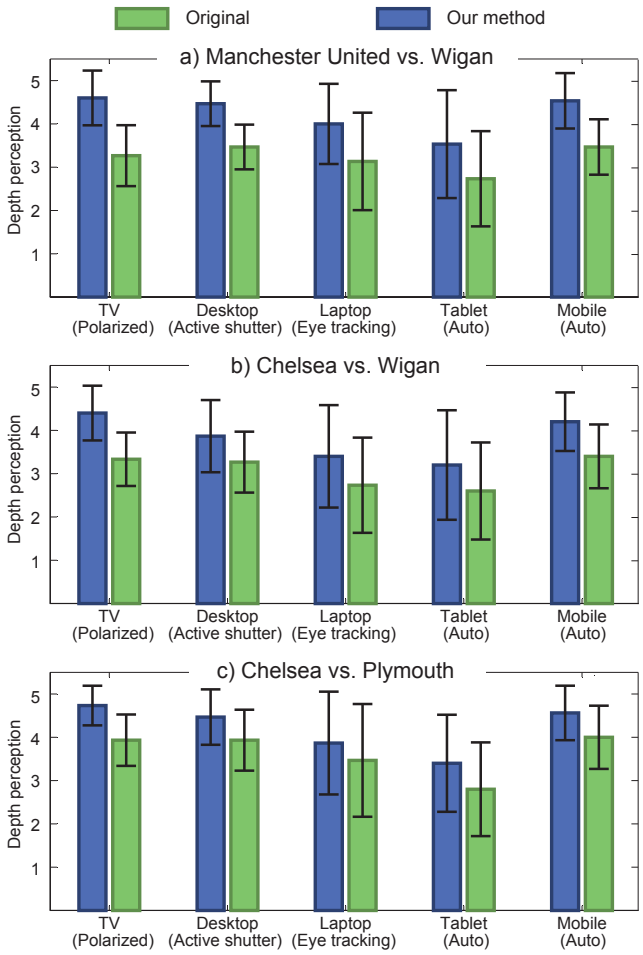


Figure 10: Our method improves depth perception for all tested videos on all displays. Error bars represent the standard deviation of the user ranking.

Experimental Setup: We tested 5 conditions: (i) Smartphone: LG Optimus 3D MAX P725 phone (autostereoscopic), with observation distance about 30 cm. (ii) Tablet: GADMEI tablet (autostereoscopic), with observation distance about 40 cm. (iii) Laptop: 15.6" Toshiba Qosmio F755-3D350 laptop (autostereoscopic with eye tracking), with observation distance about 60 cm. (iv) Desktop: 27" Samsung desktop (active shutter glasses), with observation distance about 1 m. (v) Big TV: 55" Philips TV-set (passive polarized glasses), with observation distance about 3 m.

Content: We used three soccer 3D video clips downloaded from YouTube. From each video, we extracted a series of shots with the following total lengths. (i) Manchester United vs. Wigan: 60 sec. (ii) Chelsea vs. Wigan: 24 sec. (iii) Chelsea vs. Plymouth: 20 sec. All shots were of long-view nature, thus suitable for our method. In our system, these shots are automatically identified by the classifier in Sec. 4.2. Other shots are not subjected to our depth customization.

Protocol: All subjects viewed the five 3D displays. For each display, the subjects were shown the original sequence and the optimized version. Both the order in which participants saw the displays and the video sequences were randomized. The subjects were asked to rank the depth perception of

each version between 1 and 5 as follows: 5–Excellent, 4–Very Good, 3–Good, 2–Fair, and 1–Poor.

Results and Discussion: We plot the ranking results for each 3D video in Fig. 10. The figure shows the mean ranking values for the depth-optimized 3D videos created by our method as well as the original 3D videos. The error bars visualize the standard deviation of the rankings. In order to analyze the obtained data we computed a series of t-tests. All differences reported in the figure are statistically significant (p-value < 0.05) except the tablet version of the second game (Fig. 10b). Regardless of the original depth perception which depends on filming style, depth range of the scene, artistic choice, and shooting angles, the study demonstrates that our method always improves the depth perception. The improvement can be as high as 35% in the case of the TV Fig. 10a.

6.3 Coverage

To demonstrate how many shots can benefit from our depth customization method, we analyze two full 3D soccer games (each is more than 90 minutes), and a 10-min segment of 3D tennis game. The two soccer games were from different broadcasters and different competitions. We watched each video and manually classified shots. In Table 1, we summarize our analysis of the first full 3D soccer game, Milan vs. Barcelona, showing the percentage of long, medium, close-ups, and out-of-field shots. Similar to any usual soccer game, the game included goals, injuries, player changes, offsides, etc. It can be seen that the percentage of shots are similar throughout the two halves of the game. During the second half, two injuries, two player changes, and two goals happened, which decreased the percentage of long shots. This is because these events typically have more close-up shots. However, it can be seen that, on average, over 70 % of full 3D soccer games are long shots. Table 2 shows the analysis of another full 3D soccer game (Manchester United vs. Liverpool). The main difference between the two games is the percentage of close-up shots. Nevertheless, the shot percentage is still dominated by long shots in both games.

Time/Shot Type	Long	Medium	Close-up	Out-of-field
00:00:00 - 00:02:10	*Introduction*			
00:02:10 - 00:48:17	73.1 %	24.4 %	1.3 %	1.2 %
00:48:17 - 1:03:20	*Half-time*			
1:03:20 - 1:53:52	67.1 %	28.4 %	1.4 %	3.1 %

Table 1: Shot analysis of the Milan vs. Barcelona full 3D soccer game on 20/2/2013 in the UEFA Champions League.

Time/Shot Type	Long	Medium	Close-up	Out-of-field
00:00:00 - 00:04:30	*Introduction*			
00:04:30 - 00:55:10	68.25 %	26.25 %	4.1 %	1.4 %
00:55:10 - 1:06:56	*Half-time*			
1:06:56 - 1:58:10	61.3 %	29.4 %	6.6 %	2.7 %

Table 2: Shot analysis of the Manchester United vs. Liverpool full 3D soccer game on 16/3/2014 in the English Premiere League.

Time/Shot Type	Long	Medium	Close-up	Out-of-field
00:00:00 - 00:00:28	*Introduction*			
00:00:28 - 00:09:41	64.4 %	16.8 %	2 %	16.8 %

Table 3: Shot analysis of the R. Nadal vs. N. Djokovic 3D tennis clip on 3/7/2011 in the Wimbledon competition.

Our analysis of the 3D segment from the tennis game between Rafael Nadal and Novak Djokovic is summarized in Table 3. The table shows that more than 64 % of the game can be enhanced by our method.

In summary, our analysis of various 3D games shows that our method can enhance between 60 % to 70 % of the shots in 3D videos in field sports, such as soccer and tennis.

7. CONCLUSIONS

It is challenging to stream and render 3D stereoscopic videos for different displays. This is not only because of the different technologies used to render depth, but also because the perceived depth depends on the size of the display, viewing distance, and the viewer's preferences. This is further complicated by the dynamic nature of the Internet used to stream 3D videos from a server to clients. Through a user study, we have shown that there is indeed a need to customize 3D videos for different displays. In our study, subjects ranked our depth-manipulated versions of the videos higher than the original ones. We have also shown that the amount of manipulation is dependent on the display size. We have presented a novel system that allows for an adaptive streaming of 3D videos to heterogeneous receivers in terms of display sizes, 3D video representations, viewers' preferences, and network conditions. Our system employs the DASH protocol for adaptive streaming and switching among different versions of 3D videos in order to improve the user viewing experience. In addition, we have proposed a new method for depth customization of 3D sport videos such as soccer, football, and tennis. Our method is computationally inexpensive and it maintains the scene structure of 3D videos. We have implemented a complete prototype of the proposed system and assessed its performance through another user study with 15 subjects viewing several 3D videos on five different 3D displays, ranging from a mobile phone to a large 55" TV set. Our results have shown that the proposed system significantly improves the depth perception in all scenarios.

8. REFERENCES

[1] 3DeeCentral. `http://www.3deecentral.com/`.

[2] 3DVisionLive. `https://www.3dvisionlive.com/`.

[3] Trivido. `http://www.trivido.com/`.

[4] Youtube. `http://www.youtube.com/`.

[5] A. Arefin, Z. Huang, K. Nahrstedt, and P. Agarwal. 4D TeleCast: Towards large scale multi-site and multi-view dissemination of 3DTI contents. In *Proc. of IEEE Conference on Distributed Computing Systems (ICDCS'12)*, pages 82–91, Macau, China, June 2012.

[6] P. Benzie, J. Watson, P. Surman, I. Rakkolainen, K. Hopf, H. Urey, V. Sainov, and C. von Kopylow. A survey of 3DTV displays: Techniques and technologies. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(11):1647–1658, 2007.

[7] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *Proc. of European Conference on Computer Vision (ECCV'04)*, pages 25–36, Prague, Czech Republic, May 2004.

[8] P. Burt and B. Julesz. Modifications of the classical notion of Panum's fusional area. *Perception*, 9(6):671–682, 1980.

[9] L. Chauvier, K. Murray, S. Parnall, R. Taylor, and J. Walker. Does size matter? The impact of screen size on stereoscopic 3DTV. In *IBC conference*, 2010.

[10] B. E. Coutant and G. Westheimer. Population distribution of stereoscopic ability. *Ophthalmic and Physiological Optics*, 13(1):3–7, 1993.

[11] K. Diab, T. Elgamal, K. Calagari, and M. Hefeeda. Storage optimization for 3D streaming systems. In *Proc. of ACM Conference on Multimedia Systems (MMSys'14)*, pages 59–69, Singapore, March 2014.

[12] P. Didyk, T. Ritschel, E. Eisemann, K. Myszkowski, and H.-P. Seidel. A perceptual model for disparity. *ACM Transactions on Graphics*, 30(4), 2011.

[13] A. Ekin, A. M. Tekalp, and R. Mehrotra. Automatic soccer video analysis and summarization. *IEEE Transactions on Image Processing*, 12(7):796–807, 2003.

[14] J. Freeman and S. E. Avons. Focus group exploration of presence through advanced broadcast services. In *Proc. of SPIE Human Vision and Electronic Imaging Conference*, pages 530–539, San Jose, CA, June 2000.

[15] A. Hamza and M. Hefeeda. A DASH-based Free-Viewpoint Video Streaming System. In *Proc. of ACM NOSSDAV'14 workshop, in conjunction with ACM Multimedia Systems (MMSys'14) Conference*, pages 55–60, Singapore, March 2014.

[16] N. S. Holliman. Mapping perceived depth to regions of interest in stereoscopic images. In *Proc. of SPIE Stereoscopic Displays and Virtual Reality Systems Conference*, pages 117–128, San Jose, CA, May 2004.

[17] W. IJsselsteijn, H. de Ridder, R. Hamberg, D. Bouwhuis, and J. Freeman. Perceived depth and the feeling of presence in 3DTV. *Displays*, 18(4):207–214, 1998.

[18] ISO/IEC 23009-1:2012. Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats.

[19] M. Johanson. Stereoscopic video transmission over the internet. In *Proc. of IEEE Workshop on Internet Applications (WIAPP'01)*, pages 12–19, Washington, DC, USA, July 2001.

[20] H. Kimata, K. Fukazawa, A. Kameda, Y. Yamaguchi, and N. Matsuura. Interactive 3D multi-angle live streaming system. In *Proc. of IEEE International Symposium on Consumer Electronics (ISCE'01)*, pages 576–579, Singapore, June 2011.

[21] M. Lang, A. Hornung, O. Wang, S. Poulakos, A. Smolic, and M. Gross. Nonlinear disparity mapping for stereoscopic 3D. *ACM Transactions on Graphics*, 29(3):10, 2010.

[22] J.-G. Lou, H. Cai, and J. Li. A real-time interactive multi-view video system. In *Proc. of ACM Multimedia Conference (ACM MM'05)*, pages 161–170, Singapore, November 2005.

[23] T. Oskam, A. Hornung, H. Bowles, K. Mitchell, and M. Gross. OSCAM - Optimized stereoscopic camera control for interactive 3D. *ACM Transactions on Graphics*, 30(6):189:1–189:8, 2011.

[24] D. Pająk, R. Herzog, R. Mantiuk, P. Didyk, E. Eisemann, K. Myszkowski, and K. Pulli. Perceptual depth compression for stereo applications. *Computer Graphics Forum*, 33(2):195–204, 2014.

[25] S. Paris and F. Durand. A fast approximation of the bilateral filter using a signal processing approach. *International Journal of Computer Vision*, 81(1):24–52, 2009.

[26] S. Pehlivan, A. Aksay, C. Bilen, G. B. Akar, and M. R. Civanlar. End-to-end stereoscopic video streaming system. In *Proc. of IEEE International Conference on Multimedia and Expo (ICME'06)*, pages 2169–2172, Toronto, Canada, July 2006.

[27] T. Shibata, J. Kim, D. M. Hoffman, and M. S. Banks. The zone of comfort: Predicting visual discomfort with stereo displays. *Journal of Vision*, 11(8):11, 2011.

[28] I. Sodagar. The mpeg-dash standard for multimedia streaming over the internet. *IEEE Multimedia Magazine*, 18(4):62–67, 2011.

[29] T. Stockhammer. Dynamic adaptive streaming over HTTP: Standards and design principles. In *Proc. of ACM Conference on Multimedia Systems (MMSys'11)*, pages 133–144, San Jose, CA, USA, February 2011.

[30] W. J. Tam, F. Speranza, S. Yano, K. Shimono, and H. Ono. Stereoscopic 3D-TV: Visual comfort. *IEEE Transactions on Broadcasting*, 57(2):335–346, 2011.

[31] J. R. Thorpe and M. J. Russell. Perceptual effects when scaling screen size of stereo 3D presentations. In *Proc. of Society of Motion Picture and Television Engineers Conferences(SMPTE'11)*, pages 1–10, Hollywood, CA, USA, October 2011.

[32] Ugo Capeto. Depth Map Automatic Generator. `http://3dstereophoto.blogspot.com/2013/04/depth-map-automatic-generator-dmag.html`.

[33] A. J. Woods, T. Docherty, and R. Koch. Image distortions in stereoscopic video systems. In *Proc. of SPIE Stereoscopic Displays and Applications Conference*, pages 36–48, San Jose, CA, September 1993.

[34] W. Wu, A. Arefin, G. Kurillo, P. Agarwal, K. Nahrstedt, and R. Bajcsy. Color-plus-depth level-of-detail in 3D tele-immersive video: A psychophysical approach. In *Proc. of ACM Multimedia Conference (ACM MM'11)*, pages 13–22, Scottsdale, Arizona, USA, November 2011.

[35] B. Xin, R. Wang, Z. Wang, W. Wang, C. Gu, Q. Zheng, and W. Gao. Avs 3D video streaming system over internet. In *Proc. of IEEE International Conference on Signal Processing, Communication and Computing (ICSPCC'12)*, pages 286–289, Hong Kong, August 2012.

[36] Q. Yang, L. Wang, and N. Ahuja. A constant-space belief propagation algorithm for stereo matching. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'10)*, pages 1458–1465, San Francisco, CA, USA, June 2010.

[37] Z. Yang, W. Wu, K. Nahrstedt, G. Kurillo, and R. Bajcsy. Viewcast: View dissemination and management for multi-party 3D tele-immersive environments. In *Proc. of the ACM Multimedia Conference (ACM MM'07)*, pages 882–891, Augsburg, Bavaria, Germany, September 2007.