Automatic Detection of 3D Quality Defects in Stereoscopic Videos Using Binocular Disparity

Sotirios Delis, Ioannis Mademlis, Nikos Nikolaidis, Ioannis Pitas
Department of Informatics
Aristotle University of Thessaloniki
Box 451, Thessaloniki 54124, GREECE
tel: +30 2310 996361
{imademlis, nikolaid, pitas}@aiia.csd.auth.gr

Abstract—3D video quality issues that may disturb the human visual system and negatively impact the 3D viewing experience are well known and become more relevant as the availability of 3D video content increases, primarily through 3D cinema, but also through 3D television. In this paper, we propose four algorithms that exploit available stereo disparity information, in order to detect disturbing stereoscopic effects, namely stereoscopic window violations (SWV), bent window effects, UFO objects and depth jump cuts on stereo videos. After detecting such issues, the proposed algorithms characterize them, based on the stress they cause to the viewer's visual system. Qualitative representative examples, quantitative experimental results on a custom-made video dataset, a parameter sensitivity study and comments on the computational complexity of the algorithms are provided, in order to assess the accuracy and performance of stereoscopic quality defect detection.

Index Terms—Visual discomfort, 3D quality, stereoscopic video, binocular disparity

I. Introduction

As the availability of 3D video content has increased in recent years, primarily in 3D cinema, certain stereoscopic effects that may confuse the human visual system and negatively impact the 3D viewing experience have been investigated. A prolonged exposure to 3D video content exhibiting quality problems can cause disturbing symptoms, such as eye strain, headaches and visual fatigue [1]. In order to deal with these problems, 3D cinematographers have set cinematographic rules which, if precisely followed during the production process, eliminate or moderate such issues. Nevertheless, in many cases, time constraints, low budget and inadequate advance planning prevent these rules from being properly followed. However, the majority of related problems can be fixed in post-production, as long as they are detected. In this paper, we propose algorithms that exploit the available stereo disparity information, in order to detect stereoscopic quality issues in videos, so that they can be fixed in a post-processing stage. Particularly, we deal with the detection of the stereoscopic window violations (SWV), bent window effects, UFO objects and depth jump cuts [1]. Moreover, the proposed algorithms try to characterize the detected effects, according to the visual stress they cause to the viewer.

The remainder of the paper is organized as follows. Section II provides an overview of state-of-the-art methods for disparity estimation, which is used later on for detecting stereoscopic quality issues, as well as information about comfortable stereoscopic vision and a presentation of existing approaches to the detection of the four stereoscopic issues studied in this paper. Section III describes the proposed detection algorithms and provides qualitative examples that demonstrate their operation. Section IV details experiments performed on a stereo video dataset, which was assembled specifically for the purpose of evaluating the proposed algorithms, and presents the respective quantitative results. Finally, conclusions are reported in Section V.

II. Disparity estimation and stereoscopic video quality assessment

A fundamental element in 3D scene interpretation is depth information. In a stereoscopic image pair, composed of a left and a right channel, a dense disparity map that assigns a depth-related disparity value to each image pixel can be estimated from detected pixel correspondences between the two channels [2]. Two different disparity maps can be extracted from a single stereo-pair, associated with the left/right image channel, respectively. When using a parallel camera setup, for each left/right-channel image point $[x, y]^\top$, in pixel coordinates, the corresponding horizontal disparity values are $d^l_{x,y} \leq 0$ and $d^r_{x,y} = -d^l_{x,y}$, while vertical disparities are zero. The closer an imaged object lies to the cameras during image acquisition, the larger is its disparity in absolute value. In contrast, objects considered to be lying at infinity, i.e., positioned very far from the cameras, are projected on pixels with near-zero disparity. When viewed in the theater space during video display, such objects appear in front of the display screen or, in the case of objects at infinity, on the display screen itself.

However, in 3DTV and 3D cinema, the stereo-pairs are typically processed in post-production, to allow a perceived placement of imaged objects, during video display, both in front of and behind the screen plane. Therefore, the disparity maps estimated from post-processed 3DTV

content typically contain both positive and negative disparity values. Pixels associated with negative left disparity are to be displayed in front of the screen, pixels with positive left disparity are to be displayed behind the screen and pixels with zero disparity will be displayed on the screen plane itself.

Disparity estimation, or "stereo matching", has been thoroughly investigated over the past three decades [2]. The disparity estimation algorithms can be classified into two main groups, namely local and global methods. Local algorithms use local neighbourhood information for matching windows, one in each stereo image channel. They generally give less accurate results than global ones, but are significantly faster, due to their reduced computational complexity. Global methods are iterative algorithms, which typically try to minimize a global energy function. They often produce quite good disparity maps, though at high computational complexity.

For our quantitative experiments described in Section IV, the state-of-the-art but computationally expensive global, variational algorithm presented in [3] was used to extract accurate and detailed dense disparity maps. It operates by exploiting temporal information and geometric constraints available in a video sequence, while the produced maps do not suffer from a common issue in disparity estimation, i.e., occasional "blank" pixels where no correspondence between channels has been detected. Additionally, two alternative algorithms were employed for the case studies in Section III, which are less computationally intensive at the cost of reduced estimation accuracy and sporadic blank pixels: a publicly available graph cuts algorithm [4], and a fast hybrid recursive matching approach [5]. In all cases, the disparity maps are stored as gray-scale images having a disparity range $[-127, 128]$. The sign indicates whether the corresponding pixel is to be displayed in front or behind the screen plane, during display.

Stereoscopic 3D video display and perception is based on a decoupling between vergence distance and accommodation distance, both defined in the theater space [6]. The distance, which the eyes must converge at, to see the same point is called vergence distance. Accommodation distance is the distance, which the eyes must focus at, in order to see a sharp image of this point. In natural viewing, the eyes accommodate and converge at the same point in 3D space. In stereoscopic image display, the eyes focus on (accommodate for) the screen plane, but converge in accordance with the value and sign of the point disparity. Only in the case of zero disparity, accommodation and convergence are in par. In its attempt to resolve the vergence - accommodation conflict and eventually see in 3D, the human visual system may suffer from eye-strain, headache and visual fatigue [6], [7].

In 1939, Fry measured the Zone of Clear Single Binocular Vision (ZCSBV), which is the set of all vergence and focal stimuli pairs that allow clear vision and successful binocular fusion. Furthermore, the vergence - accommodation conflict has been studied thoroughly by Percival [8], who set the minimum and maximum boundaries for comfortable 3D viewing, on a two-axes vergence distance vs focal distance plot. The region between these boundaries is known as Percival's zone of comfort. Sheard [9] also dealt with the same issue and set a slightly different zone of comfort. Percival's and Sheard's zones of comfort are subregions of ZCSBV. Recently, Hoffman and Banks [10] performed various experiments on how vergence - accommodation conflicts in stereo displays affect visual discomfort and fatigue. In particular, they examined the effect of viewing distance, disparity sign, exposure time and certain viewer properties, such as their refractive error (e.g., myopia, astigmatism) and age. They found that positive and negative left disparities are less comfortable at large and short viewing distances, respectively. Finally, they determined their own zone of comfort. The Banks zone of comfort is a much narrower subregion of Percival's and Sheard's comfort zones. 3D cinematographers, based on Percival's comfort zone boundaries, have set their own rules for constructing visual content that the viewer might fixate on without discomfort, which allow a negative/positive left disparity up to $2-3\%$ and up to $1-2\%$ of the frame width, respectively [10]. In order to set the appropriate thresholds for our algorithms, we adopted as starting points these simple percentage guidelines that seem to be followed by the majority of 3D content creators.

Production of quality stereoscopic video content is a difficult process that has to combine technical, perceptual and artistic aspects [1]. Certain software and hardware devices exist nowadays, that assist stereographers in avoiding annoying phenomena, such as stereoscopic window violation, UFO objects and bent window effects. Such devices are meant to be employed either during video production, or in the post-production stage. Moreover, they assist in avoiding depth jump cuts in the editing process, or smoothing them in post-production. However, much of the released stereoscopic content suffers from several of the above mentioned issues, since most of these phenomena remain undetected.

Several assistance systems have been proposed for stereo video shooting and 3DTV production. The stereoscopic analyzer (STAN) developed by HHI [11] detects stereoscopic window violation and gives a framing alert, by keeping track of several features present in both the left and right stereo image. The above approach, though it works in real-time, is of limited accuracy, as it involves sparse disparity maps and needs special hardware. Detection of stereoscopic window violations (referred to as framing violations) was proposed as a possible extension of the computational stereo camera system [12]. However, such an algorithm was neither implemented, nor tested. The same team also proposed a method that corrects a stereoscopic window violation, by pushing the object that violates the video frame border behind the screen with a disparity scaling. Lang et al proposed a non-linear disparity adaptation approach [13], in which they present an example of interpolating depth jump cuts (rapid scene cuts), in order to create smooth transitions from one shot

to the next. Kopal et al [14] proposed a viewer-centric editor for stereo cinema that gives the ability to the system operator to fix stereoscopic window violations, by adding a floating window mask (referred to as proscenium arch) to the appropriate image. Moreover, the editor gives the opportunity to fix depth jump cuts, by applying a cross fading effect, while translating images to change the convergence point before and after the cut, to have the incoming shot depth match that of the outgoing one. The above processes are manual and, thus, need the presence of an operator. Tseng et al [15] integrated simple stereoscopic window violation and depth jump cut detection algorithms within a framework for automatic optimization of stereo camera parameters, allowing a degree of direct control over the disparity range during video production. However, this approach is best suited for the correction of stereoscopic video quality issues in computer-generated imagery, when viewed by a virtual stereo camera (e.g., in 3D animated films).

A different approach consists in extending the concept of aggregate video quality assessment metrics to also consider visual discomfort in the case of 3D videos. Lopez et al [16] have presented such a metric, which takes under consideration depth jump cuts and stereoscopic window violations. Ha et al [17] introduced a different quality assessment metric, which exploits more general disparity and motion distribution and variance characteristics. Their algorithm was validated on a stereo video dataset, by establishing a strong correlation between the outcome of their metric and subjective quality evaluations. Solh et al [18] calculate a quality metric by first computing an "ideal" depth map per frame, to account for distortions introduced by disparity map estimation, compression or transmission noise, and by subsequently computing three partial distortion measures related to disparity characteristics (temporal outliers, temporal inconsistencies, spatial outliers). These measures are then fused into an aggregate visual discomfort metric and the results are evaluated by comparing them to subjective quality assessment tests.

Despite the significant overlap, the end goal of such algorithms is different than that of the described defect detection algorithms, since they typically do not identify specific quality issues or artifacts, but quantitatively characterize video frames or entire video sequences. A deliberate blending of the two approaches is possible, however, as in the case of Voronov et al. [19], where stereoscopic quality defects such as color and sharpness mismatch are first detected and subsequently used to derive an aggregate quality metric. This is not the approach followed in this study, but it is an interesting field for possible future research.

Finally, few 3D video quality defect detection algorithms are fully automated, can be easily integrated into a video post-processing pipeline, are fairly accurate and cover a variety of stereoscopic effects, at the same time. The set of algorithms proposed in this work was designed with the goal of meeting the preceding criteria, provided that the dense disparity estimation is accurate enough.

III. Stereoscopic quality issues detection

In this section, we provide a description of four 3D cinematography effects, namely Stereoscopic Window Violation, UFO objects, bent window and depth jump cuts [1] and present the proposed detection algorithms. Each effect/cinematographic rule and its detection is described in a separate subsection, followed by representative detection examples. In all the provided examples, unless otherwise noted, the original videos were recorded at a resolution of 1920×1080 pixels ($W = 1920$, $H = 1080$), but were sub-sampled to 960×540 in order to reduce the disparity estimation execution time.

A. Stereoscopic Window Violation

In 3D cinematography, we observe the 3D world through the so-called Stereoscopic Window (SW) [20], namely the TV or cinema screen. In other words, the viewer watches objects floating in a space confined by the screen edges. If the left disparity of a 3D point is positive/zero/negative, the eyes converge to a point either behind the screen, on screen or within the theater space (in front of the screen), respectively. Retinal rivalry occurs on the left or right frame edges, when object regions positioned close to left image left or right border do not have correspondence (are not displayed) in the right frame and vice versa. For objects with zero disparity, no retinal rivalry is observed. When part of an object is cut off by the vertical edge of the display, it results in the so-called Stereoscopic Window Violation (SWV) and is interpreted as occlusion by the viewer.

SWV does not create any problems, when it occurs behind the screen (i.e., for objects with positive left disparity), because both disparity and occlusion cues dictate that the object lies behind the screen. However, when SWV involves objects perceived as appearing in front of the screen (i.e., they have negative left disparity), the occlusion cue conflicts the disparity one. Generally, as occlusion supersedes the disparity cue, the object is finally perceived as lying behind the screen plane [20]. The above are true for a mild SWV, where only a small object region at the left or right frame edge is missing from the other image. In a severe SWV, the missing object region is so extended that the human brain cannot fuse the images and eventually see 3D.

SWV in negative disparities is not only undesirable, but may also prove painful. The rule regarding SWV states that a cinematographer has to avoid breaking the stereoscopic window, while an object has negative left disparity. However, objects entering or exiting the video frames in no more than half a second cause no problem [1], since, by the time the brain localizes the object in front of the screen, the entire object is either fully visible in the frame or has disappeared, respectively. It must be pointed out that all the above apply to cases of mild SWV, when the rivalry region is relatively narrow. In a severe SWV, stereopsis is totally interrupted and the viewer simply sees a double image. A frequently used cinematographic tool

to fix an SWV is the so-called floating window, which is created by adding black masks on the sides of the left or right image. Masking only one image does not reduce the video frame size, but changes the perceived position of the screen window in 3D space.

A simple, yet effective, algorithm that detects the Stereoscopic Window Violation using disparity maps is presented in this work. We assume that the left and right dense disparity maps have been estimated for each stereoscopic video frame, i.e., $d^l_{x,y}$ and $d^r_{x,y}$, $x = 0, .., W-1$, $y = 0, .., H-1$, where W, H are the width and height of the video frame (in pixels). At the first step of the algorithm, pixels $[x, y]^\top$ are selected, having left disparity $d^l_{x,y} < -T_1^{SWV}$ and right disparity $d^r_{x,y} > T_1^{SWV}$. We choose a suitable threshold T_1^{SWV} and perform a per-frame connected component analysis with an 8-pixel neighbourhood to extract object regions (connected components) that are displayed significantly in front of the screen. To reduce noise, objects with small width or height (less than thresholds T_w^{SWV} or T_h^{SWV}, respectively) are rejected. The detected objects are then enclosed into rectangular regions of interest (ROIs). Thus, two sets of ROIs $\mathrm{R}^r = \{R^r_1, R^r_2, ..., R^r_N\}$ and $\mathrm{R}^l = \{R^l_1, R^l_2, ..., R^l_N\}$ are created for the left and right channel, respectively. These ROIs are represented by their upper left and lower right coordinates $[x^j_{i,min}, y^j_{i,min}]^T$ and $[x^j_{i,max}, y^j_{i,max}]^T$, where $j = \{r, l\}$ and i is the ROI index. The corresponding disparities are denoted by $d^j_{i,min}$ and $d^j_{i,max}$.

Two types of disturbing SWVs can be defined. In the first type, namely left SWV, the violation occurs on the left video frame border, since there is a region in the left image, which is missing from the right one, as shown in Figure 1. Its detection is performed as follows. If one or more object ROIs R^r_i, with disparity characteristics, such as those previously described, lie on the left border of the right image, that is, if $x^r_{i,min} = 0$, a SWV is present, because $x^l_{i,min} = x^r_{i,min} + d^r_{i,min} > 0$. Thus, the region $[0, d^r_{i,min}] \times H$ in the left image is not present in the right one. Another condition is introduced to reduce false alarms, because of disparity map inaccuracies. The object pixel number in the two leftmost ROI columns must be greater than a threshold T_2^{SWV}, expressed as a percentage of the ROI height, to decide that this object signals a SWV.

A similar procedure is followed for the detection of a right SWV. In this case, a region appearing in the rightmost border of the right image is absent from the left image, as shown in Figure 2. Thus, if one or more object ROIs detected in the left disparity map R^l_i lie on the right border of the left image, i.e., if $x^l_{i,max} = W-1$, a SWV is present. This is because $x^r_{i,max} = x^l_{i,max} + d^l_{i,max} < W-1$. Therefore, the region $[W + d^l_{i,max}, W-1] \times H$ in the right image is not present in the left one. The previously described false alarm reduction approach regarding small regions (noise) is applied to right SWV detection, as well.

When a left or right SWV of duration d_{SWV} frames is detected, the condition $d_{SWV} > \frac{fps}{2}$ is checked to

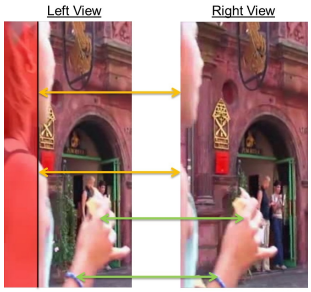

Fig. 1. Left Stereoscopic Window Violation.

determine whether the violation is annoying or not, where fps is the video frame rate. If yes, a floating window (black mask) is applied, either on the left or on the right image, depending on the SWV type, to hide regions that are visible to only one eye. The floating window width is estimated as follows. In the case of a left SWV, we first calculate a mean value $\overline{m^r_i}$ of the first three columns of right image object disparities for every object that creates a SWV, as follows:

$$\overline{m^r_i} = \left(\sum_{x=0}^{2} \sum_{y=y_{min}}^{y_{max}} d^r_{x,y} \right) / \, 3\, h_i, \quad \forall R^r_i : x^r_{i,min} = 0, \qquad (1)$$

where h_i is the height of object R^r_i and $x^r_{i,min}$ is the left vertical boundary of the object. The appropriate left floating window mask width FW_l is the mean value of all $\overline{m^r_i}$:

$$FW_l = \left(\sum_{i=1}^{N} \overline{m^r_i} \right) / \, N, \qquad (2)$$

where N is the number of objects that cause SWV, when detected in the right disparity map. This is done because the disparities of boundary ROI pixels, which are involved in a SWV, point at the boundary line of the region visible to only one eye (see the vertical line in Figures 1 and 2). The right floating window mask width FW_r is estimated using a similar approach. Although the floating window is a quick and effective way to correct mild SWVs, a strong SWV may interrupt stereopsis and result in a double image perception. To take this into account, we have set a threshold $T_{FW} = 0.03W$ on the floating window width. If this threshold is exceeded, the SWV is characterized as strong SWV and no floating window can be applied to fix it.

The computational complexity of the preceding algorithm is linear to the number of detected connected components having significant negative left disparity. Therefore, its computational requirements are dominated by those of the employed connected component analysis algorithm. Connected component labeling can efficiently run in linear time relative to the total number of video pixels [21].

Subsequently, we provide representative examples of SWV detection. The segments were extracted from a

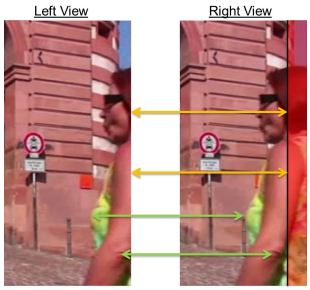

Fig. 2. Right Stereoscopic Window Violation.

stereo video that is available on the Internet. The video resolution is 640×480 pixels. The disparity maps for these examples were estimated using algorithm [5].

Case study 1: A left stereoscopic window violation is illustrated in Figure 3. It is obvious from Figure 3d that the right disparity map signals the beginning of the violation, when the lady hits the right image border in the second video frame ($n = 2$), while being in front of the screen. In the left and right disparity maps shown in Figure 3c and 3d, respectively, pixels with absolute disparity values greater than the disparity threshold T_1^{SWV} are marked yellow. There are additional pixels in the first frame ($n = 1$) that signal violation, but the T_2^{SWV} threshold, which is set to 30% of the ROI height, prevents false alarm. Therefore, the algorithm detects a SWV that starts at the second frame and ends when the lady disappears at frame $n = 17$. It must be pointed out that a small part of the object is still visible in the left image, even when it has disappeared from the right one, which means that a SWV still exists for one more video frame. Such a situation cannot be detected accurately through the left disparity map, because there is no correspondence regarding the object between the left and right images at the same time instance and the disparities in this region are interpolated using their neighbouring values. The SWV duration is 16 frames or $16/25 = 0.64$ seconds (the video fps is 25). Thus, the SWV duration exceeds the threshold $T_d = 25/2 \approx 13$ and the violation is labeled as annoying. Therefore, a floating window is needed to fix it, having width ranging from 28 to 30 pixels. This means that the violation is mild and can be fixed by applying a floating window mask on every left image, where the SWV occurs.

Case study 2: A right stereoscopic window violation is demonstrated in Figure 4. Here the left disparity map, shown in Figure 4c signals the beginning of the SWV, when the lady enters the frame, while being in front of the screen. The algorithm starts signaling SWV at the second video frame ($n = 2$), since, at $n = 1$, the size thresholds T_w^{SWV} and T_h^{SWV}, meant to filter out small artifacts caused by noise, are not exceeded. The algorithm stops signaling SWV, when the lady does no longer hit the right image border (frame $n = 9$). As before, this happens because the T_2^{SWV} threshold is not exceeded and, thus, these pixels are not taken into account. The SWV duration of 7 video frames is lower than the duration threshold of $fps/2 = 13$ frames. Therefore, the violation is not considered annoying. However, since the violation width ranges from 36 to 44 pixels, the SWV is mild, but significant. Thus, a floating window mask to every right image involved in the SWV can fix the problem.

B. The Bent Window effect

A stereoscopic window violation can involve any of the four video frame borders (left, right, top, bottom ones). Although the most distracting SWVs are those that occur at the left or right border of the screen, because they cause retinal rivalry, a violation can happen even at the top or bottom frame borders. Typically, top or bottom window violations cause less discomfort to the human brain, but may ruin the 3D effect and change depth perception. Figure 5 depicts a car that has positive left disparity, thus appearing behind the screen plane, and a street pole having significant negative left disparity, thus appearing in front of the screen. Although neither the house nor the tree interferes with the left or right edge of the screen, the shot framing cuts off the top and bottom of the tree. In such a case, the viewer's brain has to decide what to do with the contradictory cues stemming from the tree position, since the top and bottom sides of the tree cannot be in front of the screen, as they are cut off by the frame top and bottom borders, while the rest of the tree lies clearly in front of the screen, due to its strongly negative left disparity. The brain's solution to this conflict is to decide, in most cases, that the stereoscopic window is bent towards the viewer, since the top and bottom window violation locks the tree behind the screen plane. It has been observed that top screen edge violations have more significant impact in the creation of the bent window effect than the bottom screen edge violation. This could be explained by the fact that we are used to seeing the entire heads, but not necessarily the feet, of people standing in front of us [1].

The proposed bent window effect detection algorithm takes as input the left disparity map of a stereo video frame. Initially, we detect objects that have significantly negative disparity. To do so, we perform connected component analysis only on pixels with negative disparity that is lower than a threshold $-T_1^{BW}$. Then we enclose every such object in a rectangular ROI, whose upper-left and lower-right corner coordinates are $[x_{i,min}, y_{i,min}]^\top$ and $[x_{i,max}, y_{i,max}]^\top$. The final output of this step is a set of ROIs $\overline{\mathcal{R}} = \{R_1, R_2, ..., R_N\}$. Subsequently, the algorithm checks if any of the objects R_i found in the previous step is in contact with the upper and lower video frame boundary. If this is the case, i.e., when $y_{i,min} = 0$ and $y_{i,max} = H-1$, the object is marked as the cause of a bent window effect.

As in the case of SWV detection, the computational complexity of the algorithm is linear to the number of detected connected components having significant negative left disparity. Therefore, its computational requirements are dominated by those of the connected component analysis process, which is linear to the total number of video pixels [21].

Fig. 3. Example of left stereoscopic violation: a) left video frames, and corresponding b) right video frames, c) left disparity maps, d) right disparity maps.

Fig. 4. Example of right stereoscopic violation: a) left video frames, and corresponding b) right video frames, c) left disparity maps.

Subsequently, we provide a representative example of bent window detection. The disparity maps for this example were estimated using algorithm [4].

Case study 3: In the example depicted in Figure 5, the camera is located right behind a thin pole, i.e., a road sign shown in frames 1 and 202 and a metal pole shown in frames 712 and 945. In both cases, the pole is located very close to the camera and has a strong negative left disparity of about -35 pixels. The algorithm detects the pole and includes it in a ROI R. Since the pole ROI intersects both the upper and lower frame edges, a bent window effect is declared.

C. UFO object detection

In 3D cinematography, a UFO is an object that is improperly displayed inside the theater space [20]. The corresponding cinematographic rule states that an object reaching far inside the theater space must be brought there in a proper way, e.g., by smooth motion. For example, an object flying at a plausible speed towards the audience is not declared as UFO. Additionally, when the object position in the theater space can be justified by the image structure, no UFO is declared either. For example, an object held on a hand that extends in front of the screen is not a UFO. Technically speaking, a UFO is an object that a) appears and disappears suddenly, b) has significantly negative left disparity and c) is not justified by the image structure [1]. UFOs cause visual discomfort and fatigue, due to rapid changes in eye convergence. For that reason it is highly important that UFOs are identified, in order to be dealt with in post-production.

There are inherent difficulties in detecting whether image construction justifies the existence of an object appearing close to the viewer, since this is directly related

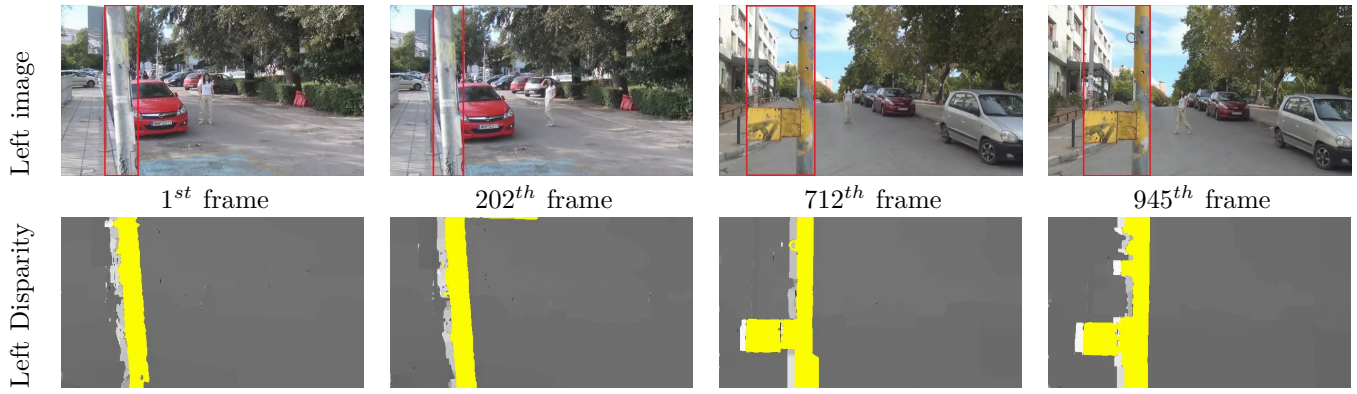

Fig. 5. Bent window effect caused by a thin object (road pole).

to rich prior knowledge about 3D set construction and object placement in the 3D space. Therefore, we did not deal with the previously mentioned condition (c) and devised an algorithm that detects a UFO by analysing object motion along the depth axis. As the 3D cinematographic rule states, a UFO object appears and disappears suddenly near the viewer. Reversely, a non-UFO object either never moves close to the viewer in the theater space, or begins its motion away from the viewer, approaches him/her in the theater space and, subsequently, returns on or behind the screen. For example, a bird flying from its nest (appearing on or behind the screen) towards the viewer and then flying back to its nest is not a UFO. Additionally, a non-UFO object may follow half the trajectory described above. Examples of non-UFOs include a knife flying from the screen to the viewer and then disappearing, or a ball appearing just in front of the viewer flying to the screen and falling to the ground, while being behind the screen. In summary, a non-UFO object with transiently negative left disparity, has either to move from the screen to the viewer or from the viewer to the screen at a sustainable speed. The proposed UFO detection algorithm is presented below.

Initially, objects with strongly negative left disparity are detected. Let us consider a left disparity sequence $d^l_{x,y}(i)$, $i = 1, .., N_t$ for a stereo video. By performing a connected component analysis on each video frame only for pixels having negative disparities that are lower than a threshold $-T_1$, we detect objects that appear inside the theater space.

Having detected an object that is a UFO candidate, we track it in previous and subsequent video frames, in order to acquire its entire motion trajectory. To this end, we compute the bounding rectangle enclosing the object and employ a tracking algorithm, e.g., the one described in [22] or [23]. The tracker output is a sequence of ROIs $\mathbf{R} = \{R_1, R_2, ..., R_{N_t}\}$. A mean left disparity value $\overline{d}_i$ for every R_i is calculated, after having previously trimmed out the background pixels from each ROI R_i, $i = 1, .., N_t$, i.e., pixels not belonging to the tracked object, using the following approach. For each ROI R_i, the initial mean disparity $\overline{m_i}$ and standard deviation σ_i are calculated. Subsequently, all ROI pixels $[x, y]^\top$ with disparity $d_{x,y} > \overline{m_i} + 2\sigma_i$ or $d_{x,y} < \overline{m}_i - 2\sigma_i$ are discarded [24]. The trimmed mean ROI disparity $\overline{d}_i$ is computed using only the remaining pixels.

After background pixel trimming for the entire set $\mathbf{R}$, a new sequence of mean disparity values $\overline{\mathbf{D}} = \{\overline{d}_1, \overline{d}_2, ..., \overline{d}_{N_t}\}$ is created. A median filter of length $M = 5$ is then applied to the sequence $\overline{\mathbf{D}}$, in order to eliminate abrupt mean disparity changes among consecutive frames, possibly caused by noise. Thus, a UFO candidate moving ROI is detected and represented by its depth trajectory, as a sequence of the filtered mean disparity values $\overline{\mathcal{D}'} = \{\overline{d}'_1, \overline{d}'_2, ..., \overline{d}'_{N_t}\}$. After finding the time instance when the object is closest to the viewer, we check whether its motion in the previous frames is smooth and whether the object comes from the screen or behind it. The same steps are repeated for video frames following the aforementioned time point. Rapid changes in the object depth or a final position close to the viewer, mean that, most likely, the object is a UFO.

To determine whether this is the case, the minimum disparity frame number *imin*, i.e., the frame number for which $\overline{d}'_{imin} < \overline{d}'_i$, $\forall i \in \{1, ..., N_t\}$, is needed. If at this point the object is not too close to the viewer, that is, if $\overline{d}'_{imin} > -T^{UFO}_{min}$, where T^{UFO}_{min} is an appropriate threshold, the object is removed from the set of candidate UFOs. Then, the algorithm returns to the previous step and another moving ROI is considered. Otherwise, subsequently, the trajectory before and after *imin* is checked for depth discontinuities and for whether the final object position lies close to the screen. That is, for every $j \in \{imin, .., 1\}$, if $\overline{d}'_j < -T^{UFO}_{max}$, where T^{UFO}_{max} is a threshold for determining whether the object is displayed on the screen plane, the "depth speed", defined as the first derivative of the disparity signal, is calculated for time instance j: $V_j = |\overline{d}'_{j-1} - \overline{d}'_j|$. If $V_j > T^{UFO}_v$, where T^{UFO}_v is the speed threshold, the object is labeled as "UFO" due to depth speed discontinuities and the algorithm terminates. If the starting point of the sequence $\overline{d}'_1$ is reached and processed, i.e., if $\overline{d}'_1 < -T^{UFO}_{max}$, the object is labeled as "potential UFO", since the initial position of its depth trajectory lies in front of the screen,

suggesting sudden appearance near the viewer. The same process is followed for time instances after $imin$, i.e., for instances $j \in \{imax, .., N\}$. The object depth speed at frame number j is now numerically computed using $V_j = |\overline{d'_{j+1}} - \overline{d'_j}|$. If the end point of the sequence $\overline{d'_{N_t}}$ is reached and processed, i.e., if $\overline{d'_{N_t}} < -T^{UFO}_{max}$, meaning that the object suddenly disappears while being displayed in front of the screen, and the moving ROI has already been declared as "potential UFO", then it is labeled as "UFO" and the algorithm terminates. Otherwise, the object is marked as "non-UFO" and the algorithm ends normally.

At the pre-processing stage of the algorithm, its computational complexity is dominated by the connected component analysis and the tracking processes, which both can run at linear time relative to the total number of video pixels [21], [25]. However, the tracking algorithm must be employed separately for each detected connected component with strongly negative left disparity, resulting in a time complexity of $\mathcal{O}(N_c HWN_t)$, assuming N_c discovered connected components. Subsequently, the main part of the algorithm starts with the pixel trimming and median filtering processes, having computational complexities of $\mathcal{O}(N_c N_t)$ and $\mathcal{O}(N_c MN_t)$, respectively, if a traditional median filter is used [26]. Finally, the ROI motion smoothness and position checks run in $\mathcal{O}(N_c N_t)$ time. The total time complexity is $\mathcal{O}(N_c N_t(HW + M + 2)) = \mathcal{O}(N_c HWN_t)$, i.e., it is dominated by the employed tracking algorithm.

Below, we provide a representative example of UFO detection. The disparity maps for this example were estimated using algorithm [4].

Case study 4: In this example, a UFO object is present for the entire duration of a video segment shown in Figure 6 and having 608 video frames. The object is detected in the first frame of the sequence, as the T_1 threshold is set to 10 pixels. Subsequently, the object is tracked forwards and the entire object ROI sequence is extracted. The trimmed mean disparity value is calculated for the ROI in each video frame, thus producing the time series $\overline{d}_n,\ n = 1, .., 608$ depicted in Figure 6c. The blue line depicts the trimmed mean disparity signal, without any post-processing. The red line is the smoothed signal, after applying a 5-point median filter [24]. The algorithm then detects the minimum mean disparity value, which, in this example, occurs at video frame $n = 373$ and has a value of -20.317 pixels. This corresponds to the time instance the object is in front of the screen and closest to the viewer. At this time, the mean disparity value exceeds threshold $T^{UFO}_{min} = 15$ pixels. The algorithm creates two lists of points $L_1 = \{373, .., 1\}$ and $L_2 = \{373, .., 608\}$. The two criteria for speed and position are checked for all points in L_1. The speed threshold $T_U = 0.5$ is not exceeded. Therefore, the object depth motion is smooth. However, since $\overline{d'_i} < -T^{UFO}_{max}$, $i = 373, .., 1$ ($T^{UFO}_{max} = 5$), the object is always displayed in front of the screen. Thus, the object is labeled as a "potential UFO". The same criteria are checked for time instances in list L_2. Since the object has the same depth motion behaviour in both time intervals L_1 and L_2, the algorithm indeed labels this object as a UFO.

D. Depth Jump Cuts

During the editing process, which is part of the post-production stage, individually recorded shots are assembled into a sequential order. This process is more complex in 3D cinematography, compared to the 2D one, because the editor has to take into consideration, among other factors, the depth continuity rule. This rule states that one should not cut between two shots, if their depth does not match [1].

There is no objective definition for the "matching depth" concept between two shots. Nevertheless, a cut from a long shot, where objects are positioned behind the screen to a close-up inside the theater space is a good example of non-matching depth cut, as the eye convergence point in the close-up shot is too far away from the convergence point in the long shot. The viewer loses 3D perception, until his/her visual system adapts to the new convergence point and the left and right images are fused together to produce a proper 3D scene perception again. This phenomenon is called a depth jump cut. A forward jump cut is much more disturbing than a backward one. In a forward jump cut, the new convergence point is closer to the viewer. Thus, the viewer's eye has to squint to restore stereopsis. On the contrary, in a backward jump cut, the eye convergence point is farther away from the previous one and the viewer has to relax his eye muscles, which is an easier task.

In 3D cinematography, there is another type of depth cut, the so-called active depth cut. It is used when a cut between two shots with "non-matching" depth is absolutely necessary, e.g., in a live music band concert, where shots depicting the band are interchanged with shots depicting the audience. In an active depth cut, the eye vergence point of the long shot is moved to the screen plane, the cut to the close-up shot is performed and the close-up shot vergence point keeps moving towards the viewer, till it takes its correct position. Other types of transitions, very common in 2D cinematography, like cross fades, wipes and split screens can be adapted to fit 3D cinematography. However, their use is limited, because their implementation is much more difficult than in the 2D case.

The algorithm proposed for the detection of depth jump cuts begins by calculating the mean positive and negative disparity values for the entire disparity map for every video frame $n = 1, .., N_t$. Given a set of disparity maps $\mathbf{d} = \{d_1, d_2, ..., d_{N_t}\}$, for every disparity map d_i, two subsets are defined, $\mathcal{A}^+_i = \{(x, y)|\ \ d_i(x, y) > 0\}$ and $\mathcal{A}^-_i = \{(x, y)|\ \ d_i(x, y) < 0\}$ and the average positive and negative disparity values are calculated as:

$$\overline{d^+_i} = \frac{1}{|\mathcal{A}^+_i|} \sum_{(j,k)\in\mathcal{A}^+_i} d_i(j, k), \tag{3}$$

and

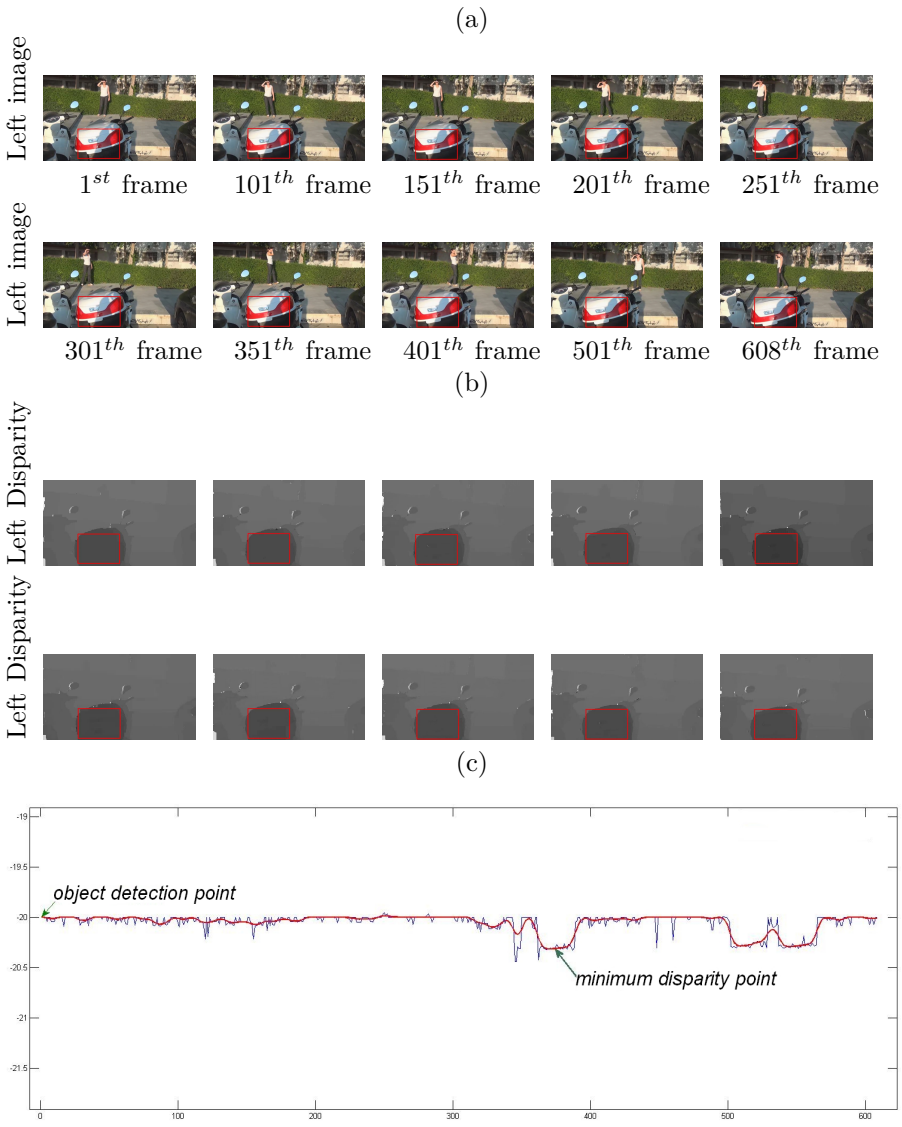

Fig. 6. A static UFO object. An object is dispayed far inside the theater space, while the object of attention (a woman walking) appears behind the screen plane. The thin blue line corresponds to the mean ROI disparity, while the thick red line corresponds to the filtered mean ROI disparity.

$$\overline{d_i^-} = \frac{1}{|\mathcal{A}_i^-|} \sum_{(j,k)\in\mathcal{A}_i^-} d_i(j,k), \tag{4}$$

where $|\mathcal{A}|$ denotes the cardinality of set $\mathcal{A}$. This way, two mean disparity signals $\overline{d_i^+}$ and $\overline{d_i^-}$, $i = 1,..,N_t$ are created, as shown in Figure 7b. Since disparity maps are generally noisy, a median filter is applied on both the positive and negative mean disparity signals. Positive disparities suffer from noise more severely than negative ones, since they usually refer to the background, which is customarily displayed behind the screen, is often blurred and covers a much bigger region than foreground. Thus, disparity estimation is harder on the background than on the foreground. Taking the above into account, we use median filter masks of length $M^- = 5$ and $M^+ = 15$ to filter the negative/positive mean disparity signals, respectively. Thus, two filtered disparity signals are constructed, namely $\overline{d_i^+}'$, $\overline{d_i^-}'$, $i = 1,..,N_t$. Then, their first derivative, which is related to the speed of change of the average positive/negative disparity, is estimated numerically by $V_i^+ = \overline{d_{i+1}^+}' - \overline{d_i^+}'$ and $V_i^- = \overline{d_{i+1}^-}' - \overline{d_i^-}'$. In order to determine whether a depth jump cut is present, we set the thresholds T_{dsn}^{DJC}, T_{dsp}^{DJC} for the negative / positive disparity derivative, respectively. The above thresholds cannot be fused to one, because the range of positive disparity values ($1-2\%$ of the screen width) is smaller than negative ones ($2-3\%$ of the screen width), so that the object is displayed in the comfort zone. At each time instance, $i = 1,..,N_t$, we compare the signals $|V_i^+|$ and $|V_i^-|$ against the T_{dsp}^{DJC}, T_{dsn}^{DJC} thresholds, respectively. If either of the thresholds is exceeded, then we label a depth jump cut in the background and in the foreground,

respectively. The derivative sign indicates a positive or negative depth jump cut. Note that there are cases, where both depth jump cuts in positive and in negative disparities are present at the same time instance, as is the case in Figure 7b.

It has been proven experimentally that rapid changes in depth, such as depth jump cuts are annoying for the human visual system and cause visual fatigue and discomfort [27], [28], [29]. Thus, in its final step, our algorithm tries to rate a depth jump cut, according to the stress it causes. We define three discomfort characterizations for depth jump cuts, namely "mildly uncomfortable", "uncomfortable", "highly uncomfortable". A characterization is given to a depth jump cut, according to the sign of positive and negative depth derivatives V_i^+ and V_i^-, as shown in Table I. Rows 1,3,7 and 9 in Table I refer to cases, where a jump cut in positive disparities is combined with a jump cut in negative disparities. In rows 2, 4, 6 and 8, a jump cut is present only in the negative or in the positive disparities. In row 5, no depth jump cut occurs. The "-", "0", "+" sign means negative, zero, positive disparity derivative, respectively. The labeling criteria have been established as follows. After a negative jump cut in the foreground (negative disparities), the viewer has to squint the eyes to adapt to the new viewing position, since the object is much closer to him. This is a painful process that needs much time. On the other hand, after a positive depth jump cut in the foreground, the viewer has to relax the eyes, which is a quicker and less painful process. When a positive jump cut happens in background (positive) disparities, the viewer has to diverge the eyes to see the background clearly, which is a tiring process for the human visual system. In a negative jump cut in background disparities, the viewer has to converge his eyes, which is a more comfortable process. Moreover, a jump cut in negative disparities is much more annoying and uncomfortable for the human visual system, than a jump cut in positive disparities. At this point, it must be stated that the perceived intensity of a depth jump cut depends on the viewer's point of attention at the time the jump cut happens. Thus, it is highly subjective. Nevertheless, usually the viewer attention is on the foreground objects, which simplifies the problem and justifies the presented characterization scheme to a large extent. However, it may be possible to employ more complicated 3D saliency models in order to find the viewer's point of attention [30] [31].

The computational complexity of the proposed algorithm is dominated by the requirements of the mean disparity signals calculation and filtering. The first one may run in linear time relative to the total number of video pixels, i.e., $\mathcal{O}(HWN_t)$, while the second one is given by $\mathcal{O}(N_tM^- + N_tM^+)$. The total time complexity of the algorithm is $\mathcal{O}(N_tM^- + N_tM^+ + N_tHW) = \mathcal{O}(N_tHW)$.

Subsequently, we provide examples of the performance of our algorithm in depth jump cut detection. The video segments are taken from the short film "The Magician". The disparity maps for these examples were estimated using algorithm [5].

Case study 5: As shown in Figure 7, an on-the-screen medium close-up shot is followed by a shot with large depth that starts at frame $n = 170$, which is followed by a shallow depth shot, beginning at frame $n = 902$. The disparity range in the first shot is $[-2, .., 3]$ pixels. In the second shot, the disparity range increases to $[-24, .., 25]$ pixels. The third shot, though, is a close-up and has shallow depth, since its disparity range is $[-8, .., 2]$ pixels.

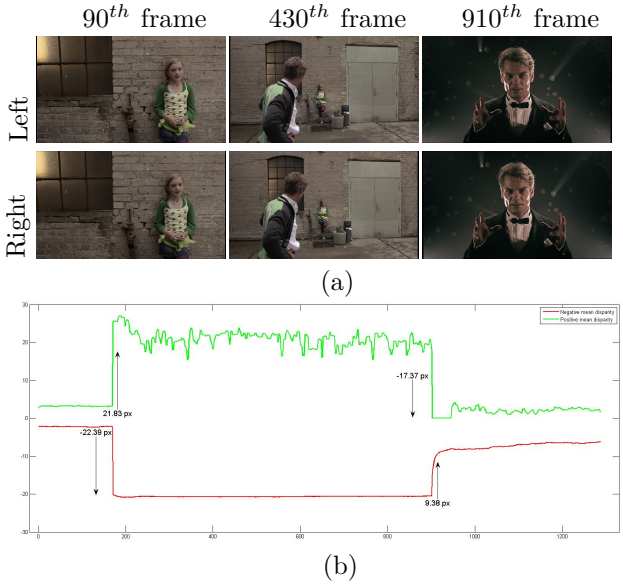

Fig. 7. A "highly uncomfortable" depth jump cut detected at frame 170 and a "mildly uncomfortable" one at frame 902: a) sample frames from the three shots, b) filtered disparity signals $\overline{d_i^+}'$, $\overline{d_i^-}'$. Arrows show time instances where the first derivative of the filtered positive and negative mean disparities is large.

The algorithm detects a negative depth jump cut at frame 170, as $V_{170}^- = -22.39 > T_{dsn}^{DJC}$. A positive depth jump cut is also detected, since the depth speed $V_{170}^+ = 21.83$ pixels exceeds the positive depth change speed threshold T_{dsp}^{DJC}. The overall depth cut is labeled as "highly uncomfortable", because $V_{170}^- < 0$, and $V_{170}^+ > 0$. Thus, viewer must squint his/her eyes considerably, to see the man appearing in front of screen plane, while he has to diverge his/her eyes too much to explore the background.

On the other hand, the depth cut at frame $n = 902$ is labeled as "mildly uncomfortable", as only the T_{dsp}^{DJC} is exceeded. As $V_{902}^+ < 0$, viewer has to relax his eyes, so that they converge to a point closer to him to explore the background. It must be pointed out that a positive depth cut at frame 902 ($V_{902}^+ = 9.56$ pixels) is not considered as a depth jump cut, since it does not exceed the T_{dsn}^{DJC} threshold.

Case study 6: In this case study, a depth jump cut in negative disparities is demonstrated. Specifically, as it is obvious from the diagram in Figure 8, a close-up shot engaging low depth range is followed by a shot with highly negative disparity and then by a shot with almost zero disparity. Thresholds remain the same as in the previous example, i.e., T_{dsn}^{DJC} and T_{dsp}^{DJC} are 15 and 10 pixels, respectively. The algorithm detects the two jump cuts

TABLE I
Depth jump cuts characterization according to disparity derivative sign. U: uncomfortable, MU: mildly uncomfortable, HU: highly uncomfortable, NO: no depth jump cut

Case	$\mathbf{V^-}$**sign**	$\mathbf{V^+}$**sign**	Label	Explanation
1	-	-	U	negative jump cut in both negative and positive disparities
2		0	U	negative jump cut in negative disparities
3		+	HU	negative jump cut in negative disparities and positive jump cut in positive disparities
4	0	-	MU	negative jump cut in positive disparities
5		0	NO	no depth jump cut present
6		+	U	positive jump cut in positive disparities
7	+	-	MU	positive jump cut in negative disparities and negative jump cut in positive disparities
8		0	MU	positive jump cut in negative disparities
9		+	U	positive jump cut in both negative and positive disparities

at frames $n = 185$ and $n = 412$. The depth jump cut at frame $n = 185$ is negative, as $V^-_{185} < 0$. Thus, it is labeled as "uncomfortable". The depth jump cut at frame $n = 412$ is labeled "mildly uncomfortable" as $V^-_{412} > 0$. The other depth cuts detected at frames $n = 359$, $n = 407$ and $n = 730$ are discarded, because the algorithm considers them negligible, since thresholds T^{DJC}_{dsn} and T^{DJC}_{dsp} are not exceeded.

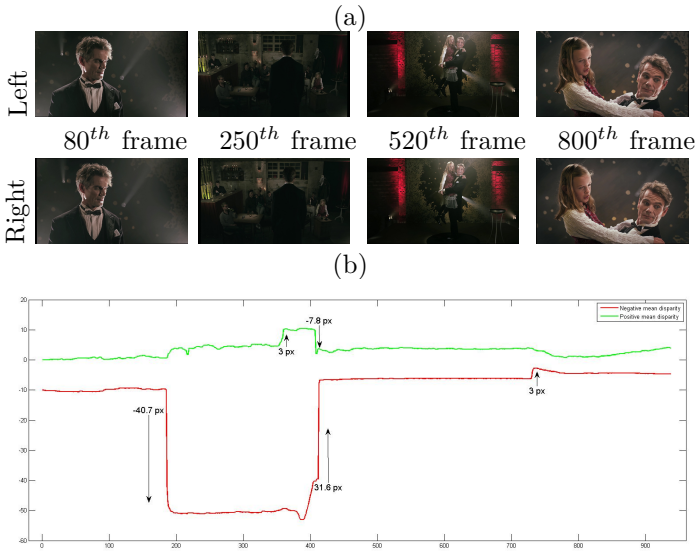

Fig. 8. An "uncomfortable" depth jump cut at video frame 185 and a "mildly uncomfortable" one at video frame 412: a) sample frames of the four shots, b) filtered disparity signals $\overline{d_i^-}'$, $\overline{d_i^+}'$. Arrows show time instances where the first derivative in positive and negative disparities is large.

IV. Experimental evaluation

There is no common and publicly available stereoscopic video dataset oriented specifically towards the detection of the four quality defects of interest, although more generic stereo quality assessment datasets do exist (e.g., [32]). Thus, a video database was assembled in order to quantitatively evaluate the proposed algorithms. It is composed of four videos, one for each of the quality defect types under examination, with every video consisting of multiple consecutive shots. A percentage of these shots exhibit the corresponding quality defect, while others are defectless.

The four videos were subjectively evaluated in an annotation process, in order to construct the ground truth of this study. The Single Stimulus Continuous Quality Evaluation method was employed in a setting conforming to Recommendation ITU-R BT.1438, a standard protocol for subjective stereoscopic video quality assessment [33]. This protocol specifies an integer five-grade scale (with 5 corresponding to "imperceptible defect" and 1 corresponding to "very annoying defect"). After the evaluation process, each video frame score S_i was thresholded to become compatible with a binary quality defect ground truth: $S_i \geq 4.5$ means that no defect is present, while $S_i < 4.5$ signals a defect. Subsequently, each defective video frame range was visually inspected to rule out cases where the viewer discomfort is not caused by the defect type under examination (e.g., in the "SWV" video, if discomfort can be solely attributed to excessive disparities, without any SWV present). Such frames were annotated as defectless.

The "SWV" video is composed of 26 shots containing 1368 frames, where a stereoscopic window violation is present, out of a total of 1986 video frames. The "BW" video is composed of 21 shots containing 999 frames, where a bent window effect is present, out of a total of 1650 frames. The "DJC" video is composed of 52 shots containing 46 depth jump cuts, out of a total of 9798 frames. The "UFO" video is composed of 27 shots containing 1087 frames where a UFO effect is present, out of a total of 2570 frames. The original videos were

Fig. 9. Example of the video dataset used for the experimental evaluation: a left-channel frame and its left disparity map.

recorded at a resolution of 1920×1080 pixels ($W = 1920$, $H = 1080$), but were sub-sampled to 960×540 pixels, in order to reduce the execution time of left and right disparity estimation for each frame. The employed stereo matching method was the state-of-the-art algorithm [3], in order to get accurate disparity maps. Figure 9 shows an example left-channel frame and its corresponding disparity map from the "DJC" video. All of the proposed algorithms were implemented in C++ and were executed in real-time, achieving processing rate greater than 25 frames per second on a high-end desktop PC (Quad Core i7 @ 3.4 GHz, 16 GB RAM).

With the exception of depth jump cuts, all of the examined quality defect instances are characterized by a specific duration (typically 15 frames or more). Therefore the temporal overlap between the detected defects and the actual defects known from the ground truth is of high importance. In order to account for this overlap, the experimental results were evaluated on a per-frame basis and each frame was deemed as a true positive, a true negative, a false positive or a false negative, based on the ground truth. Given these characterizations, a set of popular metrics were evaluated for the results of each algorithm, i.e., precision, recall, F-Measure, specificity (true negative rate) and accuracy.

Figures 10a-d show the experimental results for the case of stereoscopic window violation, bent window, UFO and depth jump cut detection, respectively, using the aforementioned metrics. As can be seen, the corresponding F-Measures are 94.05%, 90.40%, 96.15% and 84.54%, while the respective specificity rates (SPC) are 79.00%, 98.31%, 98.79% and 99.89%. Therefore, the algorithms successfully detect the majority of the defects in all cases, while the false positive rate ($FPR = 1 - SPC$) is negligible in all cases, except for the stereoscopic window violations. Note, however, that the specificity and accuracy metrics are not very informative in the case of depth jump cuts, since their computation is dominated by the number of true negatives TN. Due to depth jump cuts being of momentary duration, only 46 out of 9798 frames contain depth jump cuts and TN is unavoidably very large, resulting in values near 100% for both specificity and accuracy.

The preceding results were obtained with the algorithm parameter values shown in Table II, where W, H are the frame width, height in pixels, respectively, and h_{ROI} is the ROI height. These values were derived through a sensitivity analysis performed on the parameters, by

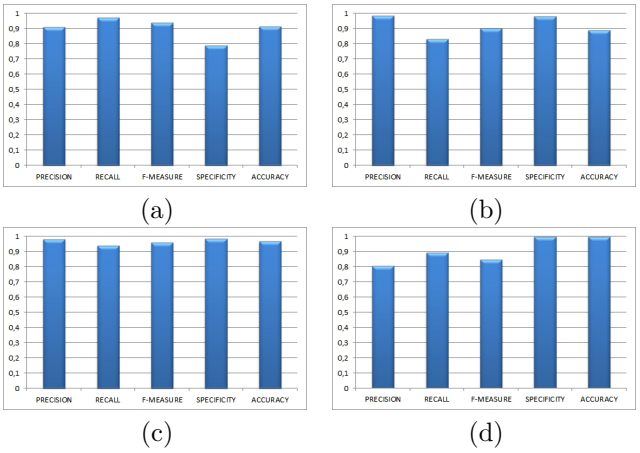

Fig. 10. Results of the proposed detection algorithms for a) stereoscopic window violations, b) bent window effects, c) UFO effects, d) depth jump cuts.

TABLE II
Parameter values used to obtain the experimental results.

Parameter	Value	Parameter	Value
T_1^{SWV}	$0.0030W$	T_{min}^{UFO}	$0.0100W$
T_h^{SWV}	$0.2000H$	T_{max}^{UFO}	$0.0030W$
T_w^{SWV}	$0.0100W$	T_v^{UFO}	$0.0010W$
T_2^{SWV}	$0.3000h_{ROI}$	T_{dsn}^{DJC}	$0.0009W$
T_1^{BW}	$0.0045W$	T_{dsp}^{DJC}	$0.0016W$

selecting the value that leads to the highest F-Measure, with the exception of T_{max}^{UFO}. The latter was always kept at a constant value of $0.0030W$, since it is a threshold that determines whether a ROI is displayed on the screen plane and it must be as low as possible. For video width $W = 960$ pixels, $T_{max}^{UFO} = 0.0030W = 2.88$ pixels. Figures 11a-i show a plot of F-Measure and specificity rate for different parameter values, for each of the other 9 parameters that were employed in the algorithms.

As can be seen, the proposed algorithms are fairly robust with regard to the parameter values, since an extended, continuous range of possible parameter values lead to high quality defect detection performance. The most sensitive parameter is T_{min}^{UFO}, for which a very specific parameter value seems to provide the best performance in terms of the F-Measure metric.

In order to compare the proposed methods against similar algorithms, rival stereoscopic quality defect detection methods had to be selected, implemented and tested on the assembled dataset. However, the few such algorithms present in the literature mostly ignore bent window and UFO defects. In fact, comparisons were only possible against the methods described in [15], regarding SWV and DJC detection. The best obtained F-Measures are 78.79% and 63.16%, for SWV and DJC respectively, while the corresponding specificity rates (SPC) are 81.26%

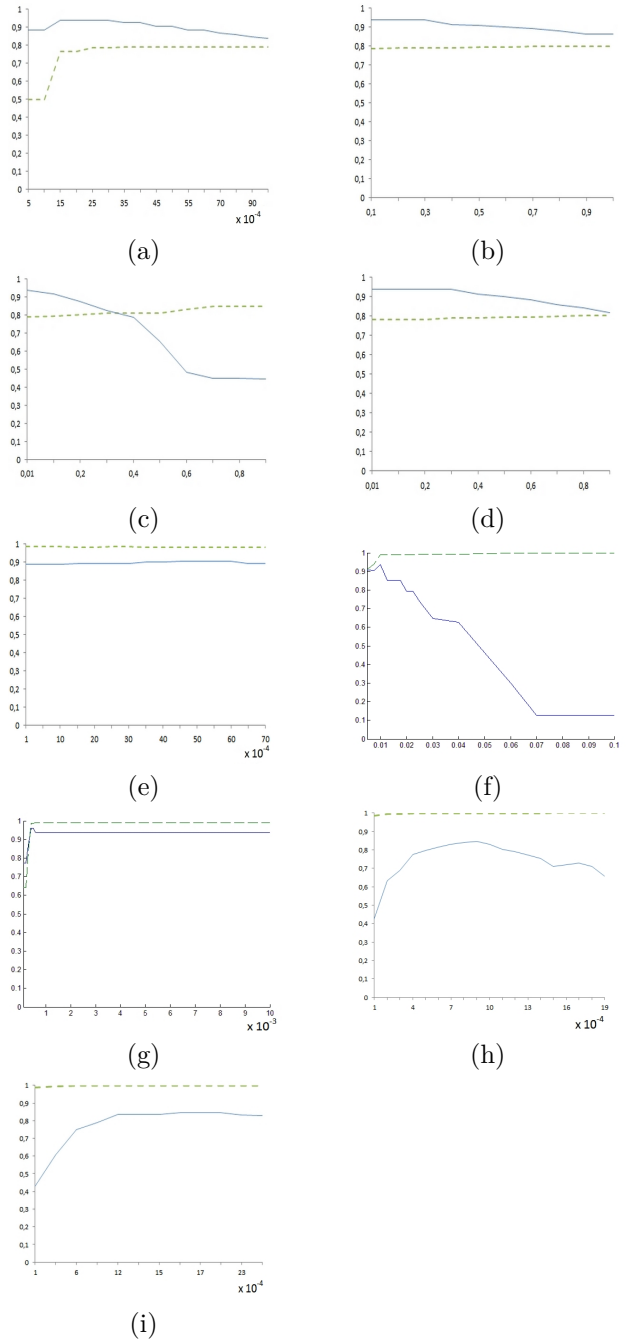

Fig. 11. Sensitivity analysis results for a) T_1^{SWV}, b) T_h^{SWV}, c) T_w^{SWV}, d) T_2^{SWV}, e) T_1^{BW}, f) T_{min}^{UFO}, g) T_v^{UFO}, h) T_{dsn}^{DJC}, i) T_{dsp}^{DJC}. The horizontal axis indicates different parameter values, while the vertical one indicates evaluation result metric values, in percentage form. The solid/dotted lines represent the F-Measure/specificity rate metrics, respectively.

and 99.51%, respectively. Thus, the proposed methods outperform these competing algorithms by 15.26% (SWV) and 21.38% (DJC), in terms of the F-Measure metric, while achieving comparable specificity/false positive rate. Additionally, the proposed methods rate the detected defects according to the visual stress they cause.

It should be noted that, during preliminary testing, tweaking the parameters affecting the operation of the employed, off-the-shelf connected component analysis library [34] led to slight performance increases. This indicates that the performance of the proposed algorithms depends not only on the noise of the inputs (the estimated disparity maps), but also on the accuracy of the employed blob extraction method. This was to be expected, since the detected connected components on the disparity channel located in front of the screen (during stereoscopic display) are the image regions being checked for any stereo quality defects.

Additionally, a method implementation parameter (initially thought to be of secondary importance) was modified after initial testing, thus contributing to slight performance increases. More specifically, on our initial SWV detection software, if two different left (respectively, right) SWVs were observed, with less than 5 frames separating them in time, they were considered to be one and the same SWV. The entire video duration from the start of the first up to the end of the second defect, was then marked as suffering from left (respectively, right) SWV. This precaution was meant to account for transient disparity estimation and blob detection errors. In the final software, this parameter was changed from 5 frames to 2 frames, thus leading to improved results.

V. Conclusions

The popularity of 3D movies makes the investigation of stereo quality issues all the more important. Certain stereoscopic effects found in the 3D video content may confuse the human visual system, affect viewing experience in a negative way and eventually cause unpleasant symptoms, such as eye strain, visual fatigue and headaches. In this paper, new algorithms are presented that detect four such stereoscopic effects, namely, stereoscopic window violations (SWV), bent window effects, UFO objects and depth jump cuts automatically, by exploiting disparity information. The algorithms also try to characterize these stereoscopic effects according to the stress they cause to the viewer. Representative qualitative examples, quantitative experimental results on a custom-made video dataset, a parameter sensitivity study and comments on the computational complexity of the algorithms are provided, proving effectiveness of the proposed methods in detecting the four above mentioned stereo quality defects.

The assembled video dataset may be useful in future stereo quality studies, by providing positive and negative examples of the four quality defects under examination.

VI. Acknowledgement

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement number 287674 (3DTVS). This publication reflects only the author's views. The European Union is not liable for any use that may be made of the information contained herein. The authors would like to thank the Fraunhofer Heinrich Hertz Institute for providing stereo videos used as case studies.

References

[1] B. Mendiburu, 3D movie making. Stereoscopic digital cinema from script to screen. Focal Press, 2009.

[2] D. Scharstein and R. Szeleiski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithm," International Journal of Computer Vision, vol. 47, no. 1-3, pp. 7–42, 2002.

[3] J. Ralli, J. Diaz, and E. Ros, "Spatial and temporal constraints in variational correspondence methods," Machine Vision and Applications, vol. 24, no. 2, pp. 275–287, 2013.

[4] V. Kolmogorov and R. Zabih, "Computing visual correspondence with occlusions using graph cuts," in Proceedings of IEEE International Conference of Computer Vision, 2001.

[5] N. Atzpadin, P. Kauff, and O. Schreer, "Stereo analysis by hybrid recursive matching for real-time immersive video conferencing," IEEE Transactions on Circuits and Systems for Video Technology, vol. 14, no. 3, pp. 321–334, 2004.

[6] M. Lambooij, W. IJsselsteijn, M. Fortuin, and I. Heynderickx, "Visual discomfort and visual fatigue of stereoscopic displays: A review," Journal of Imaging Science and Technology, vol. 53, no. 3, pp. 30 201–1, 2009.

[7] D. M. Hoffman, A. R. Girshick, K. Akeley, and M. S. Banks, "Vergence – accommodation conflicts hinder visual performance and cause visual fatigue," Journal of vision, vol. 8, no. 3, pp. 33–33, 2008.

[8] A. S. Percival, "The relation of convergence to accommodation and its practical bearing," Ophthalmological Review, vol. 11, pp. 313–328, 1892.

[9] C. Sheard, "The prescription of prisms," American Journal of Optometry, vol. 11, no. 10, pp. 364–378, 1934.

[10] T. Shibata, J. Kim, D. M. Hoffman, and M. S. Banks, "The zone of comfort: Predicting visual discomfort with stereo displays," Journal of vision, vol. 11, no. 8, pp. 11–11, 2011.

[11] F. Zilly, M. Müller, P. Eisert, and P. Kauff, "The stereoscopic analyzer – an image-based assistance tool for stereo shooting and 3d production," in Proceedings of IEEE International Conference on Image Processing, 2010.

[12] S. Heinzle, P. Greisen, D. Gallup, C. Chen, D. Saner, A. Smolic, A. Burg, W. Matusik, and M. Gross, "Computational stereo camera system with programmable control loop," ACM Transactions on Graphics, vol. 30, no. 4, p. 94, 2011.

[13] M. Lang, A. Hornung, O. Wang, S. Poulakos, A. Smolic, and M. Gross, "Nonlinear disparity mapping for stereoscopic 3D," ACM Transactions on Graphics, vol. 29, no. 3, p. 10, 2010.

[14] S. J. Koppal, C. L. Zitnick, M. F. Cohen, S. B. Kang, B. Ressler, and A. Colburn, "A viewer-centric editor for stereoscopic cinema," IEEE Computer Graphics and Applications, vol. 31, no. 1, 2011.

[15] K. L. Tseng, W. J. Huang, A. C. Luo, W. H. Huang, Y. C. Yeh, and W. C. Chen, "Automatically optimizing stereo camera system based on 3D cinematography principles," in Proceedings of IEEE 3DTV-Conference, 2012.

[16] J. P. Lopez, J. A. Rodrigo, D. Jimenez, and J. M. Menendez, "Stereoscopic 3D video quality assessment based on depth maps and video motion," EURASIP Journal on Image and Video Processing, vol. 2013, no. 1, p. 62, 2013.

[17] K. Ha and M. Kim, "A perceptual quality assessment metric using temporal complexity and disparity information for stereoscopic video," in Proceedings of the IEEE International Conference on Image Processing, 2011.

[18] M. Sohl, G. AlRegib, and J. M. Bauza, "3VQM: A vision-based quality measure for DIBR-based 3D videos," ser. Proceedings of the IEEE International Conference on Multimedia and Expo, 2011.

[19] A. Voronov, D. Vatolin, D. Sumin, V. Napadovsky, and A. Borisov, "Towards automatic stereo-video quality assessment and detection of color and sharpness mismatch," in Proceedings of the IEEE International Conference on 3D Imaging, 2012.

[20] B. Mendiburu, Y. Pupulin, and S. Schklair, 3D-TV and 3D cinema. Tools and processes for creative stereoscopy. Focal Press, 2012.

[21] K. Suzuki, I. Horiba, and N. Sugie, "Linear-time connected-component labeling based on sequential local operations," Computer Vision and Image Understanding, vol. 89, no. 1, pp. 1–23, 2003.

[22] O. Zoidi, A. Tefas, and I. Pitas, "Visual object tracking based on local steering kernels and color histograms," IEEE Transactions on Circuits and Systems for Video Technology, vol. 23, no. 5, pp. 870–882, 2013.

[23] O. Zoidi, N. Nikolaidis, A. Tefas, and I. Pitas, "Stereo object tracking with fusion of texture, color and disparity information," Signal Processing: Image Communication, vol. 29, no. 5, pp. 573–589, 2014.

[24] I. Pitas and A. N. Venetsanopulos, Nonlinear Digital Filters: Principles and Applications. Boston:Kluwer, 1990.

[25] S. Gu, Y. Zheng, and C. Tomasi, "Linear time offline tracking and lower envelope algorithms," in IEEE International Conference on Computer Vision, 2011.

[26] S. Perreault and P. Hebert, "Median filtering in constant time," IEEE Transactions on Image Processing, vol. 16, no. 9, pp. 2389–2394, 2007.

[27] S. Yanoa, M. Emotoa, and T. Mitsuhashia, "Two factors in visual fatigue caused by stereoscopic HDTV images," Displays, vol. 25, no. 4, pp. 141–150, November 2004.

[28] S. Yanoa, S. Idea, T. Mitsuhashib, and H. Thwaitesc, "A study of visual fatigue and visual comfort for 3D HDTV/HDTV images," Displays, vol. 23, no. 4, pp. 191–201, 2002.

[29] K. Ukai and P. A. Howarth, "Visual fatigue caused by viewing stereoscopic motion images: Background, theories and observations," Displays, vol. 29, no. 2, pp. 106–116, 2008.

[30] Y. Niu, Y. Geng, X. Li, and F. Liu, "Leveraging stereopsis for saliency analysis," IEEE Conference on Computer Vision and Pattern Recognition, 2012.

[31] J. Wang, M. P. D. Silva, P. L. Callet, and V. Ricordel, "Computational model of stereoscopic 3D visual saliency," IEEE Transactions on Image Processing, vol. 22, no. 6, pp. 2151–2165, 2013.

[32] L. Goldman, F. De Simone, and T. Ebrahimi, "A comprehensive database and subjective evaluation methodology for quality of experience in stereoscopic video," in Proceedings of the Three-Dimensional Image Processing (3DIP) and Applications. International Society for Optics and Photonics, 2010.

[33] ITU, "Subjective assessment of stereoscope television pictures," 2000.

[34] C. C. Linan, "cvBlob," http://cvblob.googlecode.com.

VII. Author Biographies

Sotirios Delis was born in Alexandroupolis, Greece in 1983. He received his Electrical Engineering Diploma in 2006 and his M.Sc degree in 2008, both from the Electrical and Computer Engineering Department of Democritus University of Thrace, Xanthi, Greece. He worked as a researcher in the Artificial Intelligence and Information Analysis laboratory at the Department of Informatics of Aristotle University of Thessaloniki till 2012. He is currently working as a software engineer in Nokia Networks. His main research interests are in stereo image

and video analysis and document image analysis.

Ioannis Mademlis is a computer scientist, specialized in artificial intelligence. He received a B.Sc. (2007) and a M.Sc. degree (2010) from the Department of Computer Science at the University of Ioannina, Greece. He also received a M.Sc. degree in cognitive science from the School of Electrical and Computer Engineering at the Aristotle University of Thessaloniki, Greece (AUTH, 2014). His background includes parallel/distributed systems and evolutionary robotics. Presently, he is a Ph.D. candidate, teaching assistant and research assistant at the Artificial Intelligence and Information Analysis Laboratory (AIIA) in the Department of Informatics of AUTH, participating in European Union-funded R&D projects. His current research interests include machine learning, computer vision, natural computing, autonomous robotics and intelligent cinematography.

Nikos Nikolaidis received the Diploma of Electrical Engineering and the Ph.D. degree in Electrical Engineering from the Aristotle University of Thessaloniki, Thessaloniki, Greece, in 1991 and 1997, respectively. From 1992 to 1996, he was a Teaching Assistant at the Departments of Electrical Engineering and Informatics at the Aristotle University of Thessaloniki. From 1998 to 2002, he was a Postdoctoral Researcher and Teaching Assistant at the Department of Informatics, Aristotle University of Thessaloniki, where he is currently an Assistant Professor. He has co-authored 1 book, 15 book chapters, 40 journal papers and 136 conference papers and co-edited one book and two special issues in journals. Moreover he has co-organized 6 special sessions in international conferences. The number of citations to his work by third authors exceeds 3000 (h-index 24). He has participated into 24 research projects funded by the EU and national funds. His areas of interest/expertise include stereoscopic/multiview video processing/analysis, anthropocentric video analysis (human detection and tracking, activity recognition), computer vision, digital image/video processing, computer graphics and visualization, multimedia copyright protection. Dr. Nikolaidis is currently serving as associate editor for the EURASIP Journal on Image and Video Processing, the International Journal of Innovative Computing Information and Control, the Innovative Computing, Information and Control Express Letters and the Journal of Information Hiding and Multimedia Signal Processing. Furthermore, he is a member of the Editorial Board of the International Journal of Multimedia Intelligence and Security. He served as Exhibits chair of IEEE ICIP 2001, Technical Program chair of IEEE IVMSP 2013 workshop and is currently serving as Publicity co-chair of EUSIPCO 2015. He is an IEEE Senior Member.

Prof. Ioannis Pitas (IEEE fellow, IEEE Distinguished Lecturer, EURASIP fellow) received the Diploma and PhD degree in Electrical Engineering, both from the Aristotle University of Thessaloniki, Greece. Since 1994, he has been a Professor at the Department of Informatics of the same University. He served as a Visiting Professor at several Universities. His current interests are in the areas of image/video processing, intelligent digital media, machine learning, human centered interfaces, affective computing, computer vision, 3D imaging and biomedical imaging. He has published over 750 papers, contributed in 39 books in his areas of interest and edited or (co-)authored another 9 books. He has also been an invited speaker and/or member of the program committee of many scientific conferences and workshops. In the past he served as Associate Editor or co-Editor of eight international journals and General or Technical Chair of four international conferences (including ICIP2001). He participated in 68 R&D projects, primarily funded by the European Union and is/was principal investigator/researcher in 40 such projects. He has 17900+ citations (Source Publish and Perish), 6250+ (Scopus) to his work and h-index 64+ (Source Publish and Perish), 38+ (Scopus).