Automatic Detection of Epileptogenic Video Content

Nelson Rodrigues N°63152
Instituto Superior Técnico
Av. Rovisco Pais, 1049-001 Lisboa, Portugal
E-mail: nelsonjsrs@gmail.com

Abstract—Epilepsy is a disease that affects around 50 million people worldwide, 3-5% of which had seizures triggered by luminance flashes or spatial patterns on images or videos. Visual contents with notable variations of luminance in short time intervals enhance the occurrence of seizures on viewers. Symmetrical patterns, including striped patterns or concentric circles, with sharp variations of luminance in its components, are also enhancers of seizures. Given the fact that often patients with photosensitive epilepsy are not aware of their condition until the triggering of an attack, it is necessary to make a risk analysis to digital content before it is distributed or displayed.

Based on studies carried out by experts, studies that later led to the elaboration of international standards, we produce two algorithms for automatic detection of epileptogenic visual contents: one for flash detection compliance with ITU-R BT. 17702; and a second one to detect and estimate the risk related to the presence of simple geometric patterns (stripes, circular, chess, etc.). To complement the risk analysis, a graphical interface was developed allowing the user to graphically verify the parts of the video that may endanger the viewers.

A careful review of the literature was made, including clinical studies and international regulations, intending to demonstrate the importance of developing evaluation tools for epileptogenic visual contents.

Index Terms—Photosensitive Epilepsy, Image Quality Assessment, Flash Detection, Pattern Detection, Automatic Video Analysis.

I. INTRODUCTION

PHOTOSENSITIVE Epilepsy (PSE) is a form of epilepsy where seizures are triggered by visual stimuli, such as flashing lights, contrasting light or geometric patterns that vary in time or space [6]. Both natural and artificial light may trigger seizures. Flashing lights or fast changing images (as in clubs, around emergency vehicles, in action movies or television programs, computer games, etc.) are the most common triggers. The seizure is generated by excessive electrical activity in the brain, that can be the a result of irregularities in the wiring of the brain and/or imbalance of neurotransmitters (chemical messengers in the brain). Individuals affected by PSE, experience what is called a generalized tonic-clonic seizure. A generalized tonic-clonic seizure is one type of seizure that involves the entire body and usually happens at the time of, or shortly after, looking at the trigger. The main symptoms of a seizure include:

- loss of conscience and fall to the ground;
- muscle contract and body stiffens;
- breathing pattern changes;
- tongue biting;
- loss of bladder control.

When the seizure finally ends, the muscles relax and the person slowly regains consciousness. It is usual that the person feels confused, tired, sore and memoryless for a short time. Around 50 million people worldwide have epilepsy, 3-5% of which had seizures triggered by luminance flashes or spatial patterns on images or videos. People with photosensitive epilepsy usually develop the condition before the age of 20, mainly between the ages of 9 and 15 years old. This condition is more likely in females than in males. Stimulus avoidance and stimulus modification can be an effective treatment in some patients and can sometimes be combined with antiepileptic drug treatment.

In the last decades, several television material was directly related to the occurrence of paroxysmal incidents:

- 1997 - The 25th episode of the anime YAT Anshin! Uchu Ryokō [8] was related to seizures in 4 children that were taken to the hospital.
- 1997 - The 38th episode of the 1st season of Pokemon [9] was broadcasted in Japan and caused 685 direct seizures.
- 2012 - The London Olympic Games promotional film [10] was blamed for triggering seizures in 4 people.

In a study made by the Spanish Society of Neurology (SEN), it was reported that in 180 hours of Spanish TV shows analysed, 1001 images where considered dangerous to susceptible viewers [11].

These incidents have led to medical studies on PSE [1]-[6] and to the formulation of national guidelines in the U.K. [11] and Japan [11], extended internationally by the International Telecommunications Union in recommendations ITU-R BT. - 1702 [12].

With the emergence of dedicated video equipment and with the improvement of viewer’s immersion, the risk is more real than ever; therefore, the implementation of guidelines preventing PSE triggers on media content production and broadcasting is mandatory. However, it is not feasible to detect images that can trigger PSE seizures using a manual quality control in real time. Hence, it is mandatory to develop automatic tools that prevent that type of visual phenomena, or at least signalize its occurrence. This paper is organized as follows:

- section II - Photosensitive epilepsy;
- section III - Flashy Video Detection;
- section IV - Red Transitions Detection;
- section V - Patterned Images Detection;
- section VI - Graphical User Interface;
- section VII - Conclusions.
II. PHOTOSENSITIVE EPILEPSY

The first guidance notes in this scope were developed between 1993 and 2001 by UK’s Independent Television Commission, ITC, further Ofcom. The final version, released in 2001 [11], was a consensus statement of a Committee set up by the ITC consisting of three medical experts: Prof G F A Harding, Prof C Binnie, and Prof A Wilkins. The Ofcom guidelines on potential harmful flashes have been adopted by ITU in 2005, in recommendation ITU-R BT. 1702 [12].

The main difference between Ofcom’s regulatory and ITU’s relies on patterned pictures characterization. Although ITU refers that regular patterns clearly discernible in normal viewing conditions [13] should be avoided, the patterns’ characteristics are not described. Both Ofcom and ITU-R also state that a sequence of flashing images lasting more than 5 seconds might constitute a risk even when it complies with the guidelines. This assumption is made considering medical opinion, that considers that the risk of seizures increases with the duration of flashing.

The presented regulations will be considered along this work. The developed algorithms are prepared to evaluate if videos comply the demanded restrictions of both organizations, and even to adapt evaluation parameters if further updates arise.

A. Related Work

Harding Flash and Pattern Analyser or HFPA [21], is a tool developed in accordance to the Ofcom [11] and ITU [12] guidelines. It is capable to detect patterns, flash sequences and evaluate the corresponding risk. It displays a graphically interpretation of the video safety. The developed tools is also used by other commercial solutions, such as Cerify [23], by Tektronix and Aurora [22], by Digimetrix (which offer quality control capability on many other aspects, such as encoding errors, buffer analysis, syntax errors, etc.).

Hitachi’s Flicker Check solution detects flashing, contrast change, patterned subliminal images. It also gives relevant information about the characteristics of the detected stimuli.

III. FLASHY VIDEO DETECTION

This section describes the algorithm developed for the detection of flashy video content that can precipitate seizures, taking in account the guidelines in ITU recommendation BT.1702 [12]. The algorithm is able to detect the occurrence of luminance flashes and red flashes.

A. Screen Luminance

Although the ITU recommendation BT.1702 [12] standard is relative to brightness values of the display, expressed in cd/m², in digital video the picture elements are represented in terms of the digital values, usually luminance and chrominances. To convert from luminance (expressed in mV) to screen brightness, ITU-R suggests the use of the graph shown in Appendix 2 of [12], which can be considered as representative of the gamma characteristic, γ, for most domestic TV screens [12]. The values of Appendix 2 of [12] can be analytical related by equation (1):

\[ L(v) = 413.435 \times \left( \frac{v}{1000} + 0.0189623 \right)^{2.2} \]  

where \( v \) represents the screen voltage in mV and \( L \) is the luminance expressed in cd/m².

In order to use equation (1), it is necessary to convert the digital values to luminance voltage. ITU stipulates, through recommendation BT.601 [14], quantization levels for the black level and the peak white level corresponding to 16.00d and 235.00d, respectively. However, it happens that in several cases these limits are not respected, and the black and white levels are represented with digital values below 16.00d (in the limit, 0d) and above 235.00d (in the limit, 255.00d), respectively.

Let \( Y_W \) and \( Y_B \) represent the digital values corresponding to peak white and black levels, respectively. Assuming a linear relation between digital values and the luminance in volts, it can be written that:

\[ v(mV) = \left( \frac{700}{Y_W - Y_B} \right) \times (Y_d - Y_B) \]  

where \( v \) represents the luminance in mV and \( Y_d \) the luminance expressed in digital values.

Combining equations (1) and (2), considering that \( Y_B \) and \( Y_W \) are 0 and 255, respectively, results:

\[ Y(cd/m^2) = 413.435(0.002745 \times Y_d + 0.0189623)^{2.2} \]  

Equation (3) is implemented in the algorithm through a look-up table procedure, in order to reduce the associated processing time.

B. Algorithm for Flashy Video Detection

A flash occurs when there is a pair of opposing changes in luminance (i.e., an increase in luminance followed by a decrease, or a decrease followed by an increase) of 20 cd/m² or more [12]. This applies only when the screen luminance of the brighter image is below 160 cd/m² and when the luminance of the darker image is above 20 cd/m². Concerning the previous definition it is necessary to isolate the region of the image where the referred transition occurs. The flash evaluation will be performed only when a pre-defined fraction of the image area, \( A_{var} \), varies. When the area of variation satisfies the given threshold, several operations area made to isolate the transition area.

Initially, the frame difference between each pair of consecutive frames is computed:

\[ f_{diff} = I_{N+1} - I_N \]  

From this difference, one of the following situations can be identified for each pixel:

• positive variation: the transition is an increase in luminance;
• negative variation: the transition is a decrease in luminance;
• null variation: no change in luminance.

To evaluate the type of the transition – positive, negative or null – a thresholding operation is applied to the resulting frame difference, through conditions (5) and (6):

1) Positive transitions

\[ f_{diffPOS}(i,j) = \begin{cases}  f_{diff}(i,j), & \text{if } f_{diff}(i,j) > 0 \\ 0, & \text{otherwise} \end{cases} \]  

2) Negative transitions
TABLE I: Relevant parameters for flash detection.

| \( \Delta Y_{\text{accLocal}} \) | 40 | -30 | 50 | -50 | 50 | -60 | 40 | -50 |
| \( \text{AccFrames} \) | 5 | 3 | 3 | 3 | 4 | 3 | 2 | 2 |

\[
f_{\text{diffNEG}}(i,j) = \begin{cases} 
  \text{diff}(i,j), & \text{if } \text{diff}(i,j) < 0 \\
  0, & \text{otherwise}
\end{cases} 
\]

Applying conditions (5) and (6), results in two images with the isolated region of the image where pixels have changed their value.

Once \( f_{\text{diffPOS}} \) and \( f_{\text{diffNEG}} \) were obtained, the average luminance difference of the transition, \( \Delta y \), is computed.

C. Luminance Variation Evaluation

Consider that histo is the histogram of the positive or negative frame differences, and that avg is the difference average value computed using the highest bins of histogram that, when summed, equals the minimum flash area, \( A_{\text{var}} \times M \times N \), where M and N defines the frame width and height, in pixels. The following steps explain how avg is computed, for both \( f_{\text{diffPOS}} \) and \( f_{\text{diffNEG}} \):

1: Generate the histogram, histo.
2: Scan histo from right to left until the number of elements in the bins equals \( A_{\text{var}} \times M \times N \) elements.
3: Computes the average value of the elements scanned, \( \Delta y \), according to (7)

\[
\Delta y = \frac{\sum_{i \in B} \text{histo}(i) \times i}{\sum_{i \in B} \text{histo}(i)} \tag{7}
\]

where B is the set of bins scanned in step 2.

This algorithm evaluates the histograms of \( f_{\text{diffPOS}} \) and \( f_{\text{diffNEG}} \), and returns the average values of luminance difference, \( \Delta Y_{\text{POS}} \) and \( \Delta Y_{\text{NEG}} \), respectively. The higher absolute value is then stored in an array, and the procedure is continuously applied along the video sequence.

D. Flash Sequence Detection

The average luminance difference between each pair of consecutive frames, computed as described in the previous section, is stored in an array, \( \Delta Y \). Analysing the signal of the elements of \( \Delta Y \), in a sequential scan, it is possible to identify the variation trend; if \( \Delta Y \) has the same signal along consecutive positions, it is assumed that the trend of variation is constant (a constant increase or decrease of luminance) and the differences are accumulated in an array \( \Delta Y_{\text{acc}} \); if during the scan of \( \Delta Y \) the value in position \( i \) has a different signal of the value in position \( i+1 \), it is considered that the trend has changed (the difference is not accumulated).

Analysing table I it is observed that during the 25 frames (a time interval of 1 second) there are 4 sections with increases of luminance followed by decreases, and the accumulated variations are, in each section, above 20 cd/m². The sequence shows the occurrence of 4 flashes in 1 second. According to the guidelines [12], this video sequence does not comply with the maximum number of 3 flashes per second.

To evaluate the compliance of the video sequence with the ITU-R guidelines, \( \Delta Y_{\text{accLocal}} \) is scanned, evaluating each group of \( fps \) elements; if more than \( \text{MAXFlash}_{\text{FREQ}} \) occur, the set of frames where the video sequence do not comply the regulatory specifications (in what concerns flash frequency) are graphically signalized in the GUI (Flash Risk Analysis graph). The output generated for this procedure is shown in figure 1b. Figure 1a shows the detected flashes for a segment of the Pokemon [9] video sequence.

E. Results and Comments

To evaluate the algorithm performance, synthetic flash sequences were generated with the following characteristics (the flash area, \( A_f \), is relative to the frame area; \( ff \) is the simulated flash frequency):

- **Sequence 1**: 120 frames with luminances alternating between black (Y=0) and white (Y=255). \( A_f = 100\% \) and \( ff = 12.5 \text{ Hz} \);
- **Sequence 2**: 120 frames with luminances alternating between black (Y=16) and white (Y=235). \( A_f = 50\% \) and \( ff = 12.5 \text{ Hz} \);
- **Sequence 3**: 120 frames with luminances alternating between black (Y=0) and white (Y=255). \( A_f = 33\% \) and \( ff = 12.5 \text{ Hz} \);
- **Sequence 4**: 120 frames with luminances alternating between black (Y=0) and white (Y=255). \( A_f = 16\% \) and \( ff = 0.25 \text{ Hz} \);
- **Sequence 5**: 120 frames with increase and decrease of luminance between Y=16 and Y=235. \( A_f = 100\% \);
- **Sequence 6**: 120 frames with flashes varying between 16 and 235. \( A_f = 100\% \);
- **Sequence 7**: 120 frames with a moving white vertical bar of luminance Y=235 and background Y=16. Bar with area of 16%.

The results of the flash luminance detection are shown on table II. The input parameters used in this analysis are the following:

- maximum allowed flash area of at least 25% of the image area;
- maximum allowed flash frequency of at least 3 Hz;
- minimum flash luminance of at least 20 cd/m².

As mentioned before, the generated sequences have simple luminance flashes with noticeable luminance transitions. In sequence 7, despite no expected flashes, the
The program detects it. This happens because the algorithm keeps accumulating the luminance variations on the array \( \Delta Y_{acc} \), and consequently the flash is detected when the moving bar disappears from the screen. However, in broadcasted material, it is not a common situation having a solely bright bar moving across a dark background.

To evaluate the algorithm performance on real video sequences, the video sequences associated to seizures, have been used, together with some additional videos. The videos analysed for flash occurrence where the following:

- **Sequence 1**: “potNoodles”, advertisement of “Pot Noodle”, a sequence of synthetic images that change quickly [7].
- **Sequence 2**: “YAT”, a sequence of the 25th episode of the anime *YAT Anshin! Uchu RyokAii* [8].
- **Sequence 3**: “Pokemon”, a sequence of the 38th episode of the 1st season of *Pokemon* [9], with flashing sequences and saturated red transitions.
- **Sequence 4**: “London Olympic Games” promotional film [10], a small sequence with flash occurrences.
- **Sequence 5**: “White Stripes - Seven Nation Army”, a music video clip [19].
- **Sequence 6**: “eChannel”, a sequence gathered from E!Channel [20].

The outputs of the program for the real video sequences can be consulted on the Annex of the developed thesis [15].

The resulting outputs for the video sequences “potNoodles”, ‘YAT”, “Pokemon” and “London Olympic Games” are conclusive - all the videos fail the regulatory at some point; this explains why viewers had seizures when viewing the presented contents. In what concerns sequence “White Stripes - Seven Nation Army” and “eChannel”, there are no related incidents; however, the video sequences fail the regulatory compliance at some point, which indicates that they are potentially harmful for viewers with PSE.

### IV. Red Transitions Detection

A potential harmful red transition occurs when there are opposing changes to or from a saturated red. A saturated red, in the RGB colorspace, occurs when the R component is at its maximum (255 for a 8 bit representation, or even 235 if the ITU recommendation BT.601-7 [14] is respected) and the remaining components, B and G, are on their minimum (0 or 16 if the ITU recommendation BT.601-7 [14] is respected). However, different hues of red can be perceived by the viewers as saturated reds, as can be inferred in figure 2. Therefore, the developed routine allows the user to define a range of “saturated red” colours, to be detected along the video analysis, as can be perceived on figure 3.

The methodology used to implement this routine is based in the work developed for luminance flash detection, presented in subsection III-C. For the luminance flash detection, only the luminance channel - Y - is analysed; however, for red detection the remaining chroma channels - Cr and Cb - have also to be considered.

#### A. Colour Space Conversion

The user can insert, in the developed GUI, 2 RGB values which create a detection zone, as shown in figure 3. These 2 values, are the minimum and maximum of the colour intensity, allowing to detect a range of hues. In other words, the colors detected inside the defined range will trigger the red detection, considering the area of flash.

![Fig. 2: Comparison between different red hues. The central stripe represents the pure saturated red (R=255, G=0, B=0).](image)

![Fig. 3: Colour detected - C - inside the detection zone.](image)

As can be seen in figure 3, the inserted RGB values - A and B - determine the colour detection zone. As the developed program only analyses videos in *yuv* format, a color conversion is needed; for this conversion, the formula presented in recommendation ITU-R. BT. 601 [14] was used.

#### B. Description of the algorithm

The developed algorithm computes the mean intensities of each colour channel and evaluates if the resulting averaged colour lies inside the detection zone. The user can define, in the input parameters, the detection zone in the RGB colorspace, for the darker, \( \text{color}_{\text{min}} \), and the saturated colour, \( \text{color}_{\text{max}} \). Both \( \text{color}_{\text{min}} \) and \( \text{color}_{\text{max}} \) are then converted to the YCbCr colorspace in order to comply with the developed flash detection routines. Let the defined variables for red transitions detection be:

- \( \text{color}_{\text{min}} = [Y_{\text{min}}, C_{R\text{min}}, C_{B\text{min}}] \) - array that sets the minimum color values;
- \( \text{color}_{\text{max}} = [Y_{\text{max}}, C_{R\text{max}}, C_{B\text{max}}] \) - array that sets the maximum color values;
- \( \text{avg}_{\text{frame}} = [Y_{\text{frame}}, C_{R\text{frame}}, C_{B\text{frame}}] \) - array that stores average intensity values for each frame channel.

Algorithm 1 illustrates the stages of the developed algorithm for red transitions. As mentioned before, the solution is generic, i.e., it is able to identify transitions of different colors. For the following examples, the colour that makes more sense to detect, for safety reasons, is red.

1 The developed graphical user interface, for flash and pattern detection, is described in section VI.

### TABLE II: Detected flashes for generated video sequences.

<table>
<thead>
<tr>
<th>Generated Sequence</th>
<th>Expected Flashes</th>
<th>Detected Flashes</th>
<th>ITU-R BT. 1702 compliment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>63</td>
<td>63</td>
<td>FAIL</td>
</tr>
<tr>
<td>2</td>
<td>59</td>
<td>59</td>
<td>FAIL</td>
</tr>
<tr>
<td>3</td>
<td>59</td>
<td>58</td>
<td>FAIL</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>PASS</td>
</tr>
<tr>
<td>5</td>
<td>28</td>
<td>28</td>
<td>FAIL</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>4</td>
<td>PASS</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>24</td>
<td>FAIL</td>
</tr>
</tbody>
</table>

*Image 320x769 to 536x810*
**Data:** YCrCb Frame; color$_{min}$; color$_{max}$

**Result:** returns color detection

- isolation of the flash area;
- calculation of the mean value of area isolated for each channel;

if avg$_{frame} >$ color$_{min}$ and avg$_{frame} <$ color$_{max}$ then color transition detected;

**Algorithm 1:** Algorithm for colour detection.

Once a harmful colour is detected, the GUI presents graphically the frame where the color transition occurred, allowing the user to identify the frame and take further actions in order to minimize the risk. The described colour detection process is performed for each frame along the video sequence.

V. PATTERNED IMAGES DETECTION

According to the studies referred on section I, patterns that are prone to induce seizures are characterized by its spatial frequency, area occupied on the screen, average luminance and luminance contrast. Considering the variety of possible patterns, it was aimed to identify only a reduced set of patterns, that includes those considered most harmful on the studies reported in [2]. Thereby, the main objective of the pattern detection algorithm was to correctly identify striped and circular patterns.

The algorithm is composed by the three main stages, as shown on figure 4:

1. Line Detection
2. Pattern Analysis
3. Risk Estimation

**Algorithm for colour detection.**

The algorithm starts by computing the level-line angle at each pixel to produce a level-line field, i.e., a unit vector field such that all vectors are tangent to the level line going through their base point. Then, this field is segmented into connected regions of pixels that share the same level-line angle up to a defined tolerance, \( \tau \). These connected regions are called line support regions, and each line support region (a set of pixels) is a candidate for a line segment.

1) *Straight Segment Detection:* OpenCV provides a very effective function to detect straight lines - Line Segment Detector (LSD) [28]. LSD aims to detect straight contours on images, where the gray level is changing fast enough from dark to light or the opposite. The algorithm takes a gray-level image as input and returns a list of detected line segments. As this function was designed as an automatic image analysis tool, it works without requiring any parameter tuning; therefore, all the refinements, if required, have to be done *a posteriori*.

The algorithm starts by computing the level-line angle at each pixel to produce a level-line field, i.e., a unit vector field such that all vectors are tangent to the level line going through their base point. Then, this field is segmented into connected regions of pixels that share the same level-line angle up to a defined tolerance, \( \tau \). These connected regions are called line support regions, and each line support region (a set of pixels) is a candidate for a line segment.

![Fig. 5: Line Segment Detector - detected segments on a synthetic image.](image)

When the same procedure is applied to real video images (figure 6) the outcome becomes noisier, demanding further line screening to discard lines that do not comply with the line selection criteria, that will be described on the next subsection.

![Fig. 6: (6a)Real image; (6b)Detected segments; (6b)Segments filtered.](image)

2) *Line Selection Criteria:* The lines detected by OpenCV’s LSD [28] function must be evaluated and discarded or grouped in order to achieve better pattern detection results, as can be verified on figure 6: the red green lines in figure 6-b) represent the detected line segments; the green ones, on figure 6-c) are the resulting segments after applying the selection procedure. The following criteria defines which lines are considered as relevant:

- length - a minimal length is required, linETHRESH;
- straight segments - establish angular relations between lines;
- midpoint deviation between parallel lines;
- minimum number of parallel lines that have to comply with the previous conditions, BinSIZE.

The length of each detected line is calculated through the euclidean distance:

\[
\text{length} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}
\]
where  \( p = (x_1, y_1) \) and  \( q = (x_2, y_2) \) represent the terminal points of the line segment, retrieved by the LSD function.

After discarding the line segments that do not satisfy the minimum length, the next step relies on finding sets of line segments that share the same orientation, \( \text{line}_\alpha \), in the image. The orientation of each line segment can be calculated through equation (9):

\[
\text{line}_\alpha = \arctan\left( \frac{y_2 - y_1}{x_2 - x_1} \right)
\]

Grouping all the segments with similar orientations in angular bins, allows to discard angular bins that do not have a pre-defined \( \text{bin}_\text{SIZE} \). Each bin has a resolution of \( \phi_{\text{bin}} \).

The last screening operation decides which line segments are related according to the deviation lines’ midpoint. This is done by computing the deviation of the midpoint in each bin group. The deviation is calculated through the distance that the midpoint of each line segment has to a perpendicular line defined from a specific pivot line. Figure 7 presents, graphically, the considered method.

![Fig. 7: Line selection by midpoint deviation.](image)

Figure 7 aims to explain how line segments are discarded by showing an evaluation round with line \( \text{L}_1 \) as pivot. In each verification, all the lines have the same orientation, as all the lines are on the same angular bin. A perpendicular line to the midpoint of \( \text{L}_1 \) is considered and all the remaining lines are evaluated relatively to it. The perpendicular line - \( \text{P}_1 \) - has intersection points with all the represented lines, which are computed to measure the distance to the midpoint of each one of the remaining lines. If the intersection point is outside the selection region, \( \text{selection}_{\text{region}} \), the line in question will be discarded from the sub group defined with \( \text{L}_1 \) as pivot. In this case, lines \( \text{L}_3 \) and \( \text{L}_5 \) are discarded. The procedure is repeated, having each line as a pivot. The subgroup constituted by \( \text{L}_3 \) and \( \text{L}_5 \) is also stored in another data structure, another line subgroup, since both comply with the described considerations. When all the stored lines have their midpoint inside the selection zone, for each line subgroup, the final set of segments is encountered.

After evaluating the lines in every single orientation, one final verification has to be performed. Each bin of retained lines has to be equal or greater than \( \text{Bin}_\text{SIZE} \), otherwise the set is discarded. The minimum number of segments - \( \text{Bin}_\text{SIZE} \) - is an input parameter and it is relevant to correctly identify a harmful striped pattern. If each subgroup set of segments satisfy the minimum number of segments, it is probable that there is more than one pattern, or a more complex one. However, all the screened subgroups are stored in order to estimate pattern risk. Concerning the information inferred from the literature [2], a five-stripe pair pattern represents a substantial increase of risk. Therefore, \( \text{Bin}_{\text{SIZE}} \) should be not less than 9 line segments.

3) Circular Segments Detection: Concentric circular patterns are prone to induce seizures, depending of the pattern’s spatial frequency and luminance contrast, as shown in the literature [2]. To identify circular patterns on noiseless synthetic images may be a simple task. However, in a noisy environment false positive/negative detections may arise.

The method used to detect the shape contours is the OpenCV’s Findcontours [29] function, which performs contours detection through the Suzuki algorithm [36]; each detected contour is stored as a vector of points. The function receives as input a grayscale or a binary image. Once all the contours are defined and stored, screening operations are performed in order to select the correct circular shapes. The first operation is to detect the moments of the shape, trough the OpenCV Moments [30] function. Moments of order \((i,j)\) are computed using equation (10)

\[
m_{i,j} = \sum_{x,y} \text{contour}(x,y) \cdot x^i \cdot y^j
\]

where \((x,y)\) are the image coordinates of the contour points.

\[
\overline{m} = \frac{m_{10}}{m_{00}}, \quad \overline{m} = \frac{m_{01}}{m_{00}}
\]

Equation (11) allows to determine the position of each contour mass center. This information allows to discard shapes were the center of mass is not inside the shape are discarded.

As it is well known, the mass center of a circular shape lies inside the shape and it also corresponds to its center. Accordingly, shapes where the center of mass is not inside the shape, are discarded. Figure 8 shows a circular pattern, its contours, and the center of mass for each contour. To verify if the mass center points are inside the shape, OpenCV’s pointPolygonTest [31] function determines whether the point is inside a contour, outside, or if it lies on an edge. It returns +1 if inside, -1 if outside, or 0 if on an edge. Considering figure 8a and discarding the contours whose center of mass if outside the contour, the outcome is shown in figure 8b.

![Fig. 8: Detection of center of mass and circle screening in a circular patterned image.](image)

Despite of losing the external circles of the original image 8a, and consequently decreasing the expectable induced risk, this approach aims to avoid contours that could trigger false circle detections, even at the cost of losing some accuracy in the risk estimation.
Another characteristic of circular shapes is the absence of vertices. Polygonal shapes of specific geometries can deteriorate a satisfactory circle detection. OpenCV’s approxPolyDP [32] function approximates a curve or a polygon with another curve/polygon with less vertices so that the distance between vertices is less or equal to a specified precision, \( \epsilon \) - parameter specifying the approximation accuracy that is the maximum distance between the original curve and the approximation made by the function. ApproxPolyDP uses the Douglas-Peucker algorithm [33]. If the found contour is circular it will have a huge number of edges. This verification will discard only contours with a small number of edges, as a circular shape, according to the precision of the approxPolyDP will have, for sure, a considerable amount of edges. Therefore, after testing, it was considered that the shapes should have more than 10 edges; the precision value that, along the course of simulations, achieved the best results was \( \epsilon = 0.02 \).

B. Isolation of Pattern Regions

The line screening, performed in section V-A, is not enough by itself to define a pattern. With all the line groups gathered, each line group is a probable structure of a pattern. In order to apply the connected components function to each line group, OpenCV provides the ConvexHull [34] function, that gathers a point set and finds its convex hull.

The convex hull of a polygon \( P \) is the smallest-area convex polygon which encloses \( P \). Informally, it is the shape of a rubber-band stretched around \( P \). The convex hull of a convex polygon \( P \) is \( P \) itself. Figure 9 shows the output generated by the OpenCV ConvexHull function, when applied to an image with 2 striped patterns.

![Original image and detected segments](image)

![Isolated region of bigger pattern](image)

![Isolated region of smaller pattern](image)

Fig. 9: Isolation of pattern region through convexHull function.

With the pattern’s regions determined according to each set of lines, the region is extracted from the original image. In the defined region, the mask will preserve the image original pixel values.

C. Extracted Pattern Analysis

Having the pattern area isolated, its risk is then evaluated. Since the patterns have periodic variations of luminance - components of bright luminance and components of dark luminance - along the stripes or circles, OpenCV’s connectedComponents [26] function isolates components with the same luminance and retrieves data about the components: number, area and average luminance. This data will allow to estimate the risk of the detected pattern.

OpenCV’s connectedComponents [26] function analyses the image pixel-by-pixel (from top to bottom and left to right) in order to identify connected pixel regions, i.e., regions of adjacent pixels which share the same set of intensity values. The function receives as input a binary or graylevel image and a measure of connectivity. For the following explanation, a binary image is assumed with 8 way connectivity.

Connected components labeling works by scanning an image, pixel-by-pixel (from top to bottom and left to right) in order to identify connected pixel regions, i.e., regions of adjacent pixels which share the same set of intensity values \( V \). For a binary image \( V=0 \) or \( 1 \); for a graylevel image, \( V \) may take a range of values. For the case of binary images, the connected components labeling operator scans the image by moving along a row until it comes to a point \( P \) (where \( P \) denotes the pixel to be labeled at any stage in the scanning process) for which \( V=1 \). Then, it examines the four neighbours of \( P \) which have already been scanned. The labelling of \( P \) occurs as follows:

1) if all four neighbours are 0, assign a new label to \( P \),
2) if only one neighbour has \( V=1 \), assign its label to \( P \), else
3) if more than one of the neighbours have \( V=1 \), and they have different labels, assign one of the labels to \( P \) and make a note of the equivalences.

After completing the scan, the equivalent label pairs are sorted into equivalence classes and a unique label is assigned to each class. As a final step, a second scan is made through the image, during which each label is replaced by the label assigned to its equivalence classes. With connectedComponents function it is possible to easily obtain relevant information of the component, such as:

- area of the component, \( area_{COMP} \);
- averaged luminance of the component, \( averageY_{COMP} \).

The first step is to generate a binary image of the extracted region derived from section V-B. Considering that this works aims to detect patterns with bright and dark stripes (2 opposed luminance values), i.e., patterns that have notorious luminance transitions, two binary images are generated. One represents the components with high pixel values or positive components, \( positive_{comp} \), and the other the low pixel values or negative components, \( negative_{comp} \). The generation of 2 images permits that the connectedComponents function is used without any modification to the source code of the function provided by OpenCV. In order to create the referred binary images an Otsu’s [27] thresholding operation is performed. Through Otsu’s method, a threshold \( th \) is computed, which is
the value of luminance that maximizes the separability between higher and lower values of luminance on the region of interest:

- **Positive Components** - pixel with luminance values above $th$.
- **Negative Components** - pixel with luminance values below $th$.

Each one of the resulting images is then eroded in order to smooth transitions between pixels borders. Erosion can be applied several times. In case of multi-channel images, each channel is processed independently. The erosion is done by using a structuring element that determines the shape of a pixel neighbourhood over which the minimum is taken.

The structuring element used for erosion has a shape of a cross, with a kernel of size 3. This option was the one that achieved better results.

Figure 11b shows the detected, positive and negative, components of figure 11a merged in the same image, for $th = 100$.

![Figure 11: Components detection on a circular pattern.](image)

The detection of components may not be as simple as shown in figure 11. If for some reason, the group of detected components, numCC, has residual components, in other words, components of area $< th_{area}$, those components are discarded. For this reason, after the detection of the connected components, the area of each component, $CC_{area}$, is evaluated. In the developed User Interface, the user is able to insert the desired threshold area, $th_{area}$, that each component, positive or negative, should satisfy; otherwise, the component is discarded.

Once the component is eroded, the region of the original image that matches the component is accessed. Then, through OpenCV’s `meanStdDev` [35] function, the mean, $Y_{mean}$, and standard deviation, $Y_{dev}$, of the pixels are computed for each component. The referred function receives as input the original image and the component under evaluation. At last, it returns the mean and standard deviation of the component’s luminance.

Once the mean and standard deviation of each connected component are computed, the algorithm evaluates which components may be part of the sought pattern. Positive components, $positive_{comp}$, and negative components, $negative_{comp}$, are evaluated individually in order to determine the similarity between components of the same group. Considering, as example, the positive components, the difference of the mean luminance amongst each pair, should not differ more than an maximum acceptable difference $Y_{maxCC_{diff}}$. The total number of detected components, $numCC_{total}$, is evaluated and the number of similar components, $similar_{CC}$, accounted for each iteration. At last, the number of similar components of each iteration, $similarCC_{total}$, the total area of the similar components, $CumulativeCC_{AREA}$, and the mean luminance of the components, $YSimilarCC_{mean}$, is returned.

Let $similarCC_{Pos_{total}}$ and $similarCC_{Neg_{total}}$ refer to positive and negative number of similar components, respectively; if the obtained number of similar components, in each group, satisfies a given minimum value, $minComp$, a risk estimation is performed. According to the ITU-R recommendations [11], a potentially harmful regular pattern contains clearly discernible stripes when there are more than 5 light-dark pairs of stripes, 10 stripes, in any orientation. Therefore, it is advisable a value of $minComp \geq 5$ for both positive, lightstripes, and negative, darkstripes, components. Through the sum of the areas of both positive, $AreaSimilarCC_{pos}$, and negative, $AreaSimilarCC_{neg}$, components, results the pattern area, $pattern_{area}$. Conditions are now met to correctly estimate the associated risk of the detected pattern.

### D. Risk Estimation

The pattern risk estimation requires several information obtained in the subsection V-C:

- pattern area, $pattern_{area}$;
- number of stripes, $N_{stripes} = light_{stripes} + dark_{stripes}$;
- mean luminance of the light stripes, $y_{L\text{-mean}}$;
- mean luminance of the dark stripes, $y_{D\text{-mean}}$.

It is also necessary to remind some parameters from the literature [2]:

- distance of visualization from the screen, $distance_{VIS}$;
- screen aspect ratio, $A = W/H$, where $W$ and $H$ are, respectively, the width and height of the screen;
- angle of vision, $\theta$.

Considering the equations (12), (14), (15), (16), (17) and (18):

$$\theta(degrees) = 2 \times \arctan \left( \frac{A \times H/2}{distance_{VIS}} \right)(degrees) \quad (12)$$

$$\text{spatialFreq}(cpd) = \frac{N}{\theta}. \quad (13)$$

$$y(x) = 0.382 \times \log(x)^3 - 2.020 \times \log(x)^2 + 1.285 \times \log(x) + 0.839 \quad (14)$$

$$y(x) = -0.708 \times x^2 + 1.792 \times x - 0.083; \quad (15)$$

$$y(x) = 0.336 \times \log(x) - 0.745 \quad (16)$$

$$C_{Michelson} = \frac{Y_{white} - Y_{black}}{Y_{white} + Y_{black}} \quad (17)$$

$$y(x) = -0.7874 \times x^{-0.3136} + 2.309 \quad (18)$$

The risk of a pattern can be estimated through the following steps:

**Step 1:** Obtain $\theta$ with equation (12);

**Step 2:** Obtain the spatial frequency, $\text{spatialFreq}$, with equation (13), considering $N_{stripes}$;

**Step 3:** Obtain the proportion of patients affected based on the number of cycles of the pattern, $p_{CPD}$, considering the $\text{spatialFreq}$ in equation (14);
Step 4: Obtain the proportion of patients affected as function of the area of the pattern, $p\text{AREA}$, considering pattern area in equation (15);

Step 5: Combining Steps 3 and 4, by multiplication, the risk associated to the pattern area is given by, $risk\text{AREA} = p\text{CPD} \times p\text{AREA}$;

Step 6: Considering $yL_{mean}$ and $yD_{mean}$, the Michelson contrast is computed with equation (17), and the associated risk is estimated with equation (18), $risk_{MICHELSON}$:

Step 7: Obtain the risk associated to the space averaged luminance of the pattern, $risk_{Y_{avg}}$, with equation (16);

Step 8: Combining the proportion at risk, resulting from the average luminance and contrast, of the pattern results in the following risk estimation: $risk_{LumaContrast} = risk_{MICHELSON} \times risk_{Y_{avg}}$.

Step 9: Assuming that all the computed risks are independent, the final risk is obtained by combining, by multiplication, $risk_{\text{AREA}}$ and $risk_{\text{LumaContrast}}$: $Risk_{FINAL} = risk_{\text{AREA}} \times risk_{\text{LumaContrast}}$.

Once the characteristics of the patterns are independent in what concerns risk estimation, the independent risks are combined by multiplication in to final global risk.

E. Results and Comments

The algorithm for pattern detection was tested with two generated and one real, video sequences:

**Pattern Sequence 1**: Generated video sequence, with 5 striped patterns and 2 circular patterns.

**Pattern Sequence 2**: Generated video sequence, with 3 striped patterns inserted in a noisy background and 2 circular patterns.

**Pattern Sequence 3**: Real video sequence with 22 seconds, with striped patterns inserted in the middle of the video sequence.

More detailed information about the tested video sequences is presented on table III.

<table>
<thead>
<tr>
<th>Pattern Sequence</th>
<th>#Frames</th>
<th>Striped Patterns</th>
<th>Circular Patterns</th>
<th>Area of the Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>1 to 5</td>
<td>6 to 7</td>
<td>100%</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
<td>1 to 129</td>
<td>130 to 200</td>
<td>25% to 100%</td>
</tr>
<tr>
<td>3</td>
<td>669</td>
<td>121 to 133</td>
<td>none</td>
<td>100%</td>
</tr>
</tbody>
</table>

TABLE III: Detailed information about the tested patterned video sequences.

The comparison between the estimated risk and the algorithm’s computed risk for specific frames can be observed on table IV.

<table>
<thead>
<tr>
<th>Video Sequence</th>
<th>Frame</th>
<th>Estimated Risk</th>
<th>Computed Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.71</td>
<td>0.8</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>0.75</td>
<td>0.9</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.429</td>
<td>0.48</td>
</tr>
<tr>
<td>2</td>
<td>32</td>
<td>0.95</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>87</td>
<td>0.031</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>129</td>
<td>0.901</td>
<td>0.88</td>
</tr>
</tbody>
</table>

TABLE IV: Comparison between the estimated risk and the algorithm’s computed risk for some frames of the tested video sequences.

The values of the computed risk for the different frames, shown in table IV, present small differences when compared to the estimated risk values (calculated concerning pattern properties). The program is able to correctly identify the different patterns displayed in the video sequences, since the patterns are not inserted in a noisy environment (as shown in the discrepancy of the computed risk of frame 87 from video sequence 2).

VI. Graphical User Interface

The GUI was developed on Qt Creator 5.3 framework, a C++ development environment designed to streamline the creation of applications and user interfaces for desktop, embedded, and mobile platforms. Figure 12 shows the layout of the main screen of the application.
Block 7: Graphical presentation of the estimated risk and regulatory compliance:

a) Flash Detection - displays generic flash and red flash occurrence.

b) Pattern Risk Analysis - displays the risk of the detected patterns: striped or circular.

c) Flash Risk Analysis - displays the flash risk analysis in compliance with the input flash properties parameters.

Block 8: Displays the current visualized frame and the analysis progress (if activated).

Besides the input parameters for a correct video decoding (Block 5), the parameters of the Flash Properties and Pattern Properties tabs have extreme importance for an accurate video analysis.

VII. Conclusions

This paperpurposes an automatic system to detect video content potentially harmful to people who are prone to Photosensitive Epilepsy. The developed algorithm for flash detection, detects flash sequences concerning the changes in luminance along the video sequence and the flash frequency. It was also implemented a routine to identify the occurrence of harmful colors in the image, such as saturated red. The solution showed credible results for the tested video sequences; the main noticed constraint relies on the fact that the algorithm does not analyse the video in real-time, which is a factor that needs future improvements.

The developed algorithm for pattern detection has notorious constraints in the detection. Since there are numerous types of patterned images, it was aimed to detect only a few set of striped and circular patterns, which are considered the most harmful pattern type. The whole procedure was developed in accordance to the existing recommendations [11], which resulted from the clinical studies described in section II. The algorithm can evaluate videos in real-time, a improvement in comparison with the flash detection algorithm.

The graphical user interface, described in section VI, presents a friendly interface that allows the user to define all the required input parameters. The GUI is able to evaluate videos of different dimensions, chroma samplings and frame rates. However, there is a need of improvement in what concerns memory efficiency.

Photosensitive epilepsy is a disease that requires attention and should not be neglected. Although in Portugal there are no reported PSE related incidents, the fact that none of the international guidelines [12] [11] are applied, should be alarming. The release of the developed application, like others with a similar scope, would enable broadcasters and content providers to evaluate the risk of the digital video content that people (and in particular, children) watch. This will provide, in what concerns harmful video content, awareness and the improvement of the quality of the broadcasted video content, and of the viewers quality of experience (QoE).

REFERENCES


[29] FindContours, OpenCV, URL: http://tinyurl.com/bwzzf0g, visited on 10/2014.


