1 Introduction

The goal of the proposed project is to create a system that will assist people with photosensitive epilepsy navigate the world wide web safely.

Photosensitive epilepsy is a type of epilepsy that can be triggered by flashing lights. The exact estimations of the number of people who suffer from it vary, but all are in the millions. Like every type of epilepsy it is extremely dangerous, as each attack can be fatal due to injuries sustained during the incident, and every attack causes brain damage that can be permanent.

With the advent of film, people have realised that TV and other bright screens that display video content have the potential to cause seizures in people suffering from photosensitive epilepsy. Most countries developed legal countermeasures to protect people from this content. In the UK, for instance, the Ofcom Broadcasting Code states that "Television broadcasters must take precautions to maintain a low level of risk to viewers who have photosensitive epilepsy". However, no such regulations exist for the world wide web, which can also contain triggering content that is not being monitored. Furthermore there exist numerous reports of such content being deliberately sent to people with the intent to cause them physical harm. What is more, and difficult to understand, such malicious acts are not illegal under UK law. These circumstances make a client-side automated content filter crucial for people with photosensitive epilepsy to be safe online.

While it is not possible to perfectly identify all triggers in online content, one can apply guidelines pertaining to TV and film to the content being displayed on personal computer screens. There are very specific instructions laid out by the W3C that describe the maximum levels of flashing content that is acceptable, and it is possible to implement a program that analyses videos to see if they pass the criteria.

Two such programs currently exist: PEAT (free but extremely outdated - as stated on its own website), and Harding FPA (proprietary, expensive, for film industry use). Neither is suitable for ordinary people to use, or developers to incorporate in their systems.

I would like to program a video analysis engine that implements the guidelines set out in W3C, and is able to analyse videos, screen captures, and potentially other dynamic elements. It will detect specific patterns of blinks and flashes that occur at the frequency, brightness and size which could make them a trigger, and report potential dangers.

A possible extension to the project that I will investigate is editing the videos so that the returned video contains no such triggers, and passes the W3C guidelines, and minimises the changes that need to be made to the video.

This engine could be used to create systems that an end user could use to shield themselves from exposure to triggering content, both malicious and accidental. I will also investigate creating such a system - a web browser extension that would screen web content for potential triggers.

2 Project Structure

1. Video Analysis Engine

   (a) The main part of the Engine part of the project will be implementing the guidelines, and efficiently analysing the videos to screen for any parts that do not meet the criteria.

   (b) Those criteria are available in the W3C: SC 2.3.1 (www.w3.org)

      i. "Three Flashes or Below Threshold: Web pages do not contain anything that flashes more than three times in any one second period, or the flash is below the general flash and red flash thresholds."

      ii. Where "Flash" is defined as "a pair of opposing changes in relative luminance that can cause seizures in some people if it is large enough and in the right frequency range".

      iii. Specific definitions of general flashes, red flashes, frequency ranges, luminance, viewing size, and exceptions are also provided in the W3C: SC 2.3.1 document, and will be implemented in the system.

      iv. "There can be no more than three general flashes and/or no more than three red flashes within any one-second period" [www.w3.org]. This needs to be tracked, and evaluated live, with as little delay as possible - possibly displaying a warning when a second flash occurs in cases where the next 1 second of video has not yet been analysed.

      v. "The combined area of flashes occurring concurrently occupies no more than a total of .006 steradians within any 10 degree visual field on the at typical viewing distance - using a 341 x 256 pixel rectangle anywhere on the displayed screen area when the content is viewed at 1024 x 768 pixels will provide a good estimate of a 10 degree visual field for standard screen sizes and viewing distances" [www.w3.org]. This poses a challenge, as a 1024x768 pixel matrix contains as many as 349696 possible distinct 341x256 rectangles. Evaluating all of them would be impossible to do in real time, and therefore various search and exclusion techniques have to be devised.

      vi. "A general flash is defined as a pair of opposing changes in relative luminance of 10% or more of the maximum relative luminance where the relative luminance of the darker image is below 0.80" [www.w3.org] - in digital terms, the relative luminance of a colour is defined as L = 0.2126 * R + 0.7152 * G + 0.0722 * B where R, G and B are defined as: if RsRGB ¡= 0.03928 then R = RsRGB/12.92 else R = ((RsRGB+0.055)/1.055) $\hat{2}$.4, and RsRGB = R8bit/255 (with analogous calculations for G and B) [www.w3.org].

vii. Tracking transitions - their relative luminance differences, transition directions, and locations, in order to pair them with others and record them as flashes (in very low computational complexity) will be a crucial and challenging task.

viii. A red flash is defined as any pair of opposing transitions involving a saturated red." - in digital terms: "for either or both states involved in each transition R/(R+ G + B) ¿= 0.8, and the change in the value of (R-G-B)x320 is ¿ 20" [www.w3.org].

ix. "Exception: Flashing that is a fine, balanced, pattern such as white noise or an alternating checkerboard pattern with "squares" smaller than 0.1 degree (of visual field at typical viewing distance) on a side does not violate the thresholds." [www.w3.org] - this allows for a faster analysis that does not need to consider every pixel individually, but considers only the average luminance and colour values of smaller squares, as fine, complementing flickers inside squares are exempt.

(c) The analysis should be computationally reasonable, and happen in quasi real time, with minimal pre-processing delay.

(d) Videos will be analysed using python. Computations will be done on a time series of frames, using NumPy matrix operations. Individual frames will be transformed to calculate the sRGB values, relative luminance, red saturation, red colour majority, and other crucial criteria for each pixel. These values will be compared with neighbouring frames to detect transitions. These transitions will be analysed to detect and classify potential flashes.

(e) This will be using the CPU. The impact of the plugin on the CPU load will be evaluated to ensure it's suitable for end users. Running calculations using the GPU is a possible extension.

2. Editing Out Triggers

   (a) A possible extension to the project is masking the detected flashes by modifying content, so that the returned video can be safely viewed.

   (b) This would be an area of research, as no such standard tools exist (Harding FPA does include recommendations on how to change the video, but does not do it itself). However, the end result is verifiable using standard tools.

   (c) Possible implementations will be investigated, and possibly added to the engine. They would rely on information from the analysis engine for information on which frames and fragments are problematic, as well as for re-evaluation after the video is modified.

3. Investigating Applications

   (a) I have chosen a web browser extension as the most appropriate way of investigating creating a system that screens for dangerous content. This is due to their popularity and ease of use.

   (b) The browser extension will likely scrape the website content for videos and possibly other dynamic elements. These will be passed to an analysis engine. Analysing rendered website content is a possible project extension.
   (c) The browser extension will use JavaScript code. Communicating with the analysis engine (running in Python) will not be easy, as Python is not generally supported with JavaScript, and may require major changes to the project plan - reworking the engine code into JavaScript, either manually or using a transpiler, or implementing a server where the Python code can be run.

4. Data Structures, Algorithms, and Evaluation
   (a) Analysing videos means working with that kind of data. Data structures will include those for video storage, and a timestamp-indexed array of events for the analysed videos. Algorithms and libraries typical for video editing are also going to be used in the project.
   (b) Test data will include captures of typical internet content – popular videos, and screen captures of popular websites that contain dynamic elements. Content will be evaluated with PEAT and compared to the results obtained by the system. Due to the limitations of PEAT (in terms of video format, codec, resolution, etc.), test data will have to be created specifically for this purpose.

5. Project Extensions and Directions
   (a) It would be extremely inefficient to re-analyse the same content every time a page is refreshed or accessed again. In order to combat this, the system will include a way of remembering the results of all analysed videos using their metadata and hash value as keys. This opens up the possibility of a centralised system that gathers lookup table data from everyone (subject to an opt-in setting), and shares it with others, allowing for even greater reductions to the computational loads. Data gathered this way could be an extremely interesting insight into the prevalence of photosensitivity triggering content in various websites across the internet, as well as some general patterns present.
   (b) Allowing user input data, such as personalised frequency sensitivity, visual field estimations, to conduct analyses more suited to individual needs.
   (c) Using wavelets or other techniques to detect movement artifacts - differences between frames that do not result from flashing, but from ordinary movement, yet are hard to exclude from the analysis.
   (d) Analysing the entire screen while browsing, and tracking the risk to the user, with possible warnings.
   (e) Improving quasi-real time behaviour by using the GPU, especially if CPU-solutions underperform.

3 Success Criteria

The analysis engine will be a success if it is able to screen for photosensitivity triggers well enough that it approximates the results achieved by the lengthy and expensive computations done by the inaccessible standard tools used in industry (PEAT and/or Harding FPA – the latter is a tool meant for use in TV and film, and follows different guidelines than those set by W3C, so differing results are to be expected). Another goal is for it to be able to do it efficiently enough that it could be used for potential applications working with live content.

The investigation into potential applications will be a success if a prototype web browser extension that screens web content for flashing images using the engine or it's transpiled version is created.

The part of the project tasked with editing out triggers will be a success if the program is able to return edited videos that closely resemble the originals, while being free from dangerous triggers.

4 Plan of Work

1. October 5th – October 17th
   Project preparation and design. Planning the project, investigating the feasibility of plans. Discussions with Supervisor and DoS. Submitting Phase 1 and Phase 2 of the proposal. Writing the final project proposal.

2. October 18th
   Project proposal submission.

3. October 19th - November 3rd
   Deciding on the project configuration elements such as IDE, setting up backups and version control.
   Specifying an implementation of the W3C guidelines that will be as close to them as possible while also computationally realistic for the purpose of client side real time applications.
   Creating a "proof of concept" video analysis program with a basic blueprint of the functionality.

4. November 5th - December 2nd
   Work on implementing W3C guidelines in the engine - calculating the relative luminance, red saturation and other relevant metrics for each pixel, as well as comparing those values between frames to detect transitions.
   Investigating applications by setting up an empty template browser add on, and researching possible ways of running the video analysis in the browser. Researching ways of accessing video elements, desktop capture, or both.

5. By December 3rd (end of Michaelmas term)
   Implementing a large part of the computations needed to screen videos for triggers that violate the W3C guidelines.

Understanding how a browser extension would be able to conduct analysis of online content, and the limitations it may pose on the video analysis engine.

6. December 4th - December 17th
   Fully implementing the W3C guidelines in the engine by incorporating calculations of visual fields, matching transitions into pairs that are to be considered flashes, recording flashes and the frequencies at which they occur.
   Investigating possible ways of removing flashes from videos that could be incorporated in the engine, or a standalone program.

7. December 18th - January 1st
   Buffer period. Time for potential leeway, low intensity work and rest.

8. January 2nd - January 17th
   Testing the engine, comparing results of the analysis with results returned by the PEAT and/or Harding FPA tools. Evaluating performance and computational complexity.
   Applying changes to the engine that may result from testing outcomes, possibly adding options of setting personalised analysis criteria.
   Implementing a basic form of removing flashes, that at this point will not concern itself with the extent of changes it applies to the video.

9. By January 18th (start of Lent term)
   Finished Video Analysis Engine, tested and evaluated.
   Implemented a basic form of removing flashes from videos.

10. January 19th - February 3rd
    Gathering project documentation and results, so that they can be included in the progress report, and later the dissertation.
    Writing the progress report and preparing presentation. Evaluating progress and accomplishments so far, considering difficulties and challenges.
    If the project is not on track, discussing with Supervisor about measures to be taken.

11. February 4th
    Progress report submission.

12. February 10th
    Progress report presentation.

13. February 11th - February 28th
    Working on removing flashes from videos in a way that limits the negative impact on the video. Evaluating the returned videos using the Video Analysis Engine, as well as standard tools.
    Preparing a dissertation plan, compiling a list of relevant documentation, notes, references, and figures. This includes code that is written and manual notes on the progress of the project.

14. March 1st - March 17th
    Further investigating applications of the engine by coding the browser extension to perform some form of video analysis with the help of the engine, or its transpiled version (due to JavaScript limitations on performance, this could be a limited version of the engine).
    Spare time for any additional work on video analysis and video editing that may be necessary, or possibly enhancing the functionality of the browser extension to be able to return a video that is free from potential triggers.
    Preparing a dissertation plan, compiling a list of relevant documentation (continued).

15. By March 18th (end of Lent term)
    Finished Analysis Engine, Editing Out Triggers, and Investigating Applications parts of the project.
    Prepared dissertation plan based on the project, compiled lists of relevant documentation, notes, references, and figures. Documenting the process will be done during the entirety of the project.

16. March 19th - April 1st
    Buffer period. Time for potential leeway, low intensity work and rest.

17. 01 April - 25 April
    Writing the dissertation based on the project. Dividing content into chapters. Creating the cover and proforma pages, necessary declarations, bibliographies, including the project proposal and appendices.

18. By April 26th (start of Easter term)
    Final draft of the dissertation. Submission to DoS and Supervisor for proofreading.

19. 27 April - 12 May
    Discussions with DoS and Supervisor, work on the dissertation to reflect relevant feedback.
    Final changes to the dissertation.

20. May 13th
    Dissertation and source code submission

The workload is heavier at the beginning of the project due to uneven course load. The plan includes buffers, as well as the following considerations for other coursework:

1. Lower intensity project work due to high intensity Units of Assessment work between November 5th and December 1st.

2. Lower intensity project work between January 18th and March 18th due to high Lent term lecture load and Units of Assessment work.

3. Only final dissertation changes and dissertation submission scheduled for Easter term due to time needed for preparation before the Tripos examinations.

5 Starting Point Statement

I have not written any significant bodies of code or other material for this project before the start of the 2021/2022 Academic Year.

To the best of my recollection I have never created video analysis tools, video editing tools or web browser extensions, and I do not intend to reuse any such code.

Creating this analysis engine will involve knowledge about human perception, colour spaces, visual fields, and understanding of the conversions between physical and digital units which I acquired during the Introduction to Graphics and Further Graphics courses. I acquired other crucial skills, such as familiarity with Python, most notably NumPy matrix computations during the Data Science course, and some personal projects.

6 Resource declaration

My project will be developed using my own machine (personal laptop). I accept full responsibility for this machine and I have made contingency plans to protect myself against hardware and/or software failure. These contingency plans include access to another machine, automatic cloud backup (Microsoft OneDrive), regular commits to online version control (GitHub), and periodic physical backups on an external hard drive.

My project will involve a commonly used browser for investigating applications. Google Chrome is preferred, as it is the most widely used browser. Alternatively Mozilla Firefox may be used.

My project will involve existing photosensitive epilepsy analysis tools: the PEAT (free), and possibly Harding FPA or another commercial tool. They will be used for ground-truth approximation, and testing.