Towards content-driven intelligent authoring of mulsemedia applications

Article *in* IEEE Multimedia · July 2020

DOI: 10.1109/MMUL.2020.3011383

5 authors, including:

Raphael Abreu
Universidade Federal Fluminense
23 PUBLICATIONS **32** CITATIONS
SEE PROFILE

Douglas Paulo de Mattos
Universidade Federal Fluminense
6 PUBLICATIONS **15** CITATIONS
SEE PROFILE

Joel dos Santos
Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ)
51 PUBLICATIONS **120** CITATIONS
SEE PROFILE

Gheorghita Ghinea
Brunel University London
435 PUBLICATIONS **4,547** CITATIONS
SEE PROFILE

Towards content-driven intelligent authoring of mulsemedia applications

Raphael Abreu, Douglas Mattos, Joel dos Santos, Gheorghita Ghinea, Débora Muchaluat-Saade

***Abstract*—Synchronization of sensory effects with multimedia content is a non-trivial and error-prone task that can discourage authoring of mulsemedia applications. Although there are authoring tools that perform some automatic authoring of sensory effect metadata, the analysis techniques that they use are not general enough to identify complex components that may be related to sensory effects. In this work, we present a new method, which allows the semi-automatic definition of sensory effects in an authoring tool. We outline a software component to be integrated into authoring tools that uses content analysis assistance to indicate moments of sensory effects activation, according to author preferences. The proposed method was implemented in the STEVE 2.0 authoring tool and an evaluation was performed to assess the precision of the generated sensory effects in comparison with human authoring. This solution is expected to considerably reduce the effort of synchronizing audiovisual content with sensory effects - in particular, by easing the author's repetitive task of synchronizing recurring effects with lengthy media.**

***Index Terms*—Multimedia applications and multimedia signal processing, Mulsemedia, Content-driven Authoring, Semi-automatic Authoring**

I. Introduction

The integration of traditional multimedia (mainly audiovisual) with stimuli engaging our other senses opens opportunities for users to experience content. Thus, a new term *mulsemedia* (Multiple Sensorial Media) was coined to define such applications [1]. Such stimuli addressing additional senses are usually implemented in practice by actuator devices generating environmental effects, such as wind, fog, and heat. We refer to these as sensory effects.

To guide the activation of a sensory effect, a mulsemedia application relies on the synchronization defined by the author. This means that the author of such applications needs to carefully inspect audiovisual content to identify and annotate it with metadata defining the begin time and the end time of a given sensory effect. This is a very costly activity in terms of effort and time, besides being error-prone. Thus, accelerating and simplifying the authoring process is paramount to encourage community adoption of such applications [2].

A first attempt to reduce the burden of manual authoring is by using graphical authoring tools. These tools provide a sophisticated graphical editing interface for synchronizing a set of media objects with sensory effects. However, they still require a long and relatively complex authoring process, since the aforementioned "media content inspection" is still necessary [2].

The main contribution of this article is a novel method for integrating the automatic recognition of sensory effects in audiovisual content with mulsemedia authoring tools. Firstly raised by [3], this is still a need of the mulsemedia authoring community. Additionally, we argue that the authoring process is a highly creative task and fully automatic solutions may impede the creative process or fail to meet the author's expectations, as pointed out by [4]. To this end, we integrate Machine Learning (ML) with mulsemedia authoring tools to aid the recognition of sensory effects. To the best of our knowledge no other work delved into this integration. The approach presented here extends the one presented in [5] by redesigning and generalizing the integration of the proposed ML-component with mulsemedia authoring tools. Moreover, it affords authors enhanced flexibility in respect of choosing the effect types to be created.

As such, we raise a set of challenges for sensory effect authoring, either manual or automatic and outline a *content-driven component* (CDC) that integrates content analysis into a mulsemedia authoring tool. The proposed method places the author in the center of the authoring process by providing support to the author to tweak whatever he/she wants. Finally we present an implementation of *CDC* in STEVE 2.0 (*Spatio-TEmporal View Editor*) [6], [7], a graphical authoring tool for mulsemedia applications.

II. Mulsemedia Authoring Challenges

To synchronize a sensory effect with a particular piece of audiovisual content, a mulsemedia author must inspect that content to identify which parts may be related to sensory effects. For example, an author may relate beach scenes with a wind effect or those depicting sunny days with a heat effect. Then, the author must annotate the begin and end times of every beach and sun occurrences in the audiovisual content and, finally, use such times to specify the synchronization of those occurrences with sensory effects. In the following paragraphs, we discuss three situations where manual authoring of sensory effects is inefficient.

(1) Synchronizing recurring effects. In the authoring process, some common effects may occur multiple times in media objects (*e.g.*, vibration when explosions occur in an action movie). As the temporal length of the media increases, the number of common sensory effects needed also increases. This in turn requires more actions (and more time) of the human author to synchronize such effects.

(2) Loss of synchronization. In an audiovisual content production environment, sometimes a section of a media object content is changed or removed. In this sense, if sensory effects have already been synchronized with this content, the author should redo the synchronization process in this section and, potentially, in the latter sections if the particular modification changes the length of the media object.

(3) Adjusting previous effects. After reconsideration or experimentation with the public, a mulsemedia author may decide to change or remove an effect related to certain media object content. For example, in a movie that has 90 beach scenes, one possible adjustment is to remove the heat effect associated with the beach. In this case, the author should perform actions to remove all 90 heat sensory effects related to beach occurrences.

Such issues arise from the fact that the author must scan the content of the media object several times to synchronize or re-synchronize sensory effects. Different related studies present approaches using graphical authoring tools to facilitate this process. In the following section, we will discuss them as well as describe the limitations of approaches that rely on fully automatic sensory effect authoring.

III. Tools for Mulsemedia Authoring

Focusing on the interoperability between virtual applications and real-world devices, the MPEG standardization group has defined the MPEG-V standard (ISO/IEC 23005). One of the uses of the standard is to provide metadata descriptions for audiovisual content regarding sensory effects to be rendered on physical devices. These descriptions include the effect begin and end times, their position in the environment, their intensity, and other rendering attributes specific to each type of effect. To facilitate the adherence of non-programmer users, the creation or use of applications conforming to MPEG-V is supported by the use of authoring tools.

As discussed in Covaci et al. [1], the quest for facilitating mulsemedia authoring has resulted in several authoring tools been developed by academia. One of the first is *SEVino* (*Sensory Effect Video Annotation*) [3]. In common with the surveyed tools, *SEVino* provides a graphical interface to the author that presents a video timeline to use as a basis for synchronizing sensory effects. The tool allows one to create time intervals that represent the duration of sensory effects. After the authoring phase, it generates MPEG-V-compliant descriptions indicating the temporal synchronization of sensory effects.

A more recent tool in development for mulsemedia authoring is the STEVE 2.0 authoring tool [6], [7]. Unlike other tools, STEVE 2.0 offers the author a timeline interface that is implemented by an event-based synchronization model [8]. In this model, for example, the start/end of a media object can be synchronized with an event generated by another media object. In STEVE 2.0, sensory effects can be synchronized with various traditional media (audio, image, and text) and not just with a single video. STEVE 2.0 also allows the author to create and synchronize sensory effects without the need for one main video or audio content to guide the application.

As pointed out by Waltl et al. [3], given the difficulty in authoring mulsemedia applications, an automatic form of authoring would encourage community adoption of such applications. A primary effort in this direction is the `autoExtraction` attribute in MPEG-V, which indicates whether extraction of a sensory effect is preferable. Although supported in the MPEG-V standard, it depends on the implementation of software capable of performing this automatic extraction. Tools supporting `autoExtraction` should perform it at runtime [3], *i.e.*, for the content already being played for the end-user. Thus, its temporal synchronization is completely automatic and independent of the application's author.

It is important to note that a fully automatic generation of sensory effects may be undesirable. After all, such authoring is an artistic process that depends on the preference of a human author to provide an enhanced user experience. Besides, fully automated proposals for authoring sensory effects have suffered negative repercussions from users in favor of human-generated ones. For instance, Lee et al. [4] report that authors of haptic effects disliked the completely automatic solution employed in the study. They see haptic authoring as a highly creative task and therefore believe it should be under author control. Thus, a better option for serving users and authors alike is to support automatic recognition of sensory effects at authoring time and give as much fine-tuning control to the author as possible. In this article, we refer to this approach as *semi-automatic authoring* of sensory effects.

Kim et al. [9] and Danieau et al. [10] propose algorithms to extract sensory effects at runtime and at authoring time. Both approaches consist of using objective measurements based on image or sound pro-

cessing to characterize information that enables sensory effects, such as pixel colors or loudness levels. The effects are added to the timeline of the authoring tool, which enables authors to fine-tune the results. One shortcoming of this approach is that the proposed algorithms are unable to identify complex elements in audiovisual content related to sensory effects (*e.g.*, beach, wind, rain, forest).

Amorim et al. [11] follow a different approach by employing *crowdsourcing* to gather the moments of activation of sensory effects. It also allows authors to fine-tune the time intervals of sensory effects indicated through *crowdsourcing*. The downside of [11] is the inherent cost and additional time needed to use a *crowdsourcing* platform. Our proposal resembles this work in the sense that it will also provide an indication of automatically-extracted sensory effects and enable the author to fine-tune the results. Apart from this, our work is aimed at integrating content analysis into existing authoring tools to automatically identify the moments of activation of sensory effects. This results in a faster solution without the additional cost of a *crowdsourcing* platform.

Current mulsemedia authoring tools fail to solve the challenges raised in the previous section, mainly because they offer limited or no support to automatic authoring, whilst the ones that do offer such capacity do not employ enough powerful algorithms to recognize semantic content related to sensory effects. To address this issue, our work combines the recognition capacity of Machine Learning-based (ML-based) content analysis with mulsemedia graphical editors, as will be discussed in the following section.

IV. Semi-automatic Authoring with Content Analysis Assistance

Multimedia content analysis offers a wide range of research directions and challenges. To this end, several algorithms have been developed to understand media content, bringing breakthroughs on tasks such as image recognition, classification, segmentation, detection, and video-content retrieval [12]. At the forefront of the more recent breakthroughs are Machine Learning-based methods. Especially, Deep Neural Networks (DNN) have been achieving remarkable performance in classification in many multimedia-related tasks [13]. DNNs can provide semantic descriptions to highly complex elements in the audiovisual media. Thus, they are also of high interest to the mulsemedia authoring field, since they can identify moments of sensory effects synchronization.

DNNs for sensory effects recognition are underway, with some studies in the academia already using DNN architectures obtaining encouraging results. For instance, Siadari et al. [12], [14] propose a DNN framework for classification of sensory effects in videos. In the work of Zhou et al. [15], a mixture of DNN methods is presented to detect sensory effect activation times and other rendering attributes. In addition, in our previous work, we built a DNN architecture that utilizes both audio and visual information to infer sensory effects activation times [16]. Our architecture was able to identify effects such as explosion, wind, thunder, rain and gunshots.

The DNN architectures for recognizing audiovisual media objects can vary dramatically, depending mainly on the task to be performed [17]. Moreover, depending on the particular architecture, networks can have different input modalities as well. For example, a network can perform the recognition only with video input while another may use both audio and video data as input. However, to perform sensory effects recognition, one similarity that such networks must have is the ability to perform classification. This task usually returns a set of labels that indicate the likelihood that certain content elements will be present at each moment in the media. In this context, it is necessary for a mulsemedia authoring tool to both be able to send the desired media to be recognized as well as be ready to parse the response description labels.

Figure 1 illustrates labels returned from a video classification task. In the figure, a 4-second video is shown and, for each second, a set of labels is presented. For brevity, only the most relevant 3 labels (top-3) are presented and their occurrence probabilities have been omitted. In the figure, we can see that the returned labels change as video content changes. At 1s from the start, the sun appears in the video and therefore the label `sun` starts to be returned.

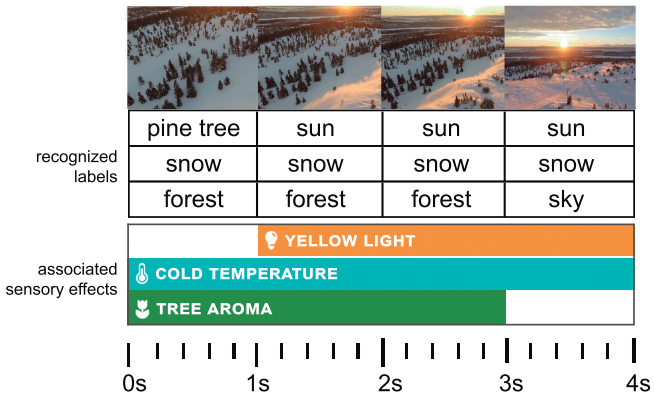

Fig. 1: Sensory effect synchronization based on labels returned by DNN.

Labels returned from the classification can be associated with sensory effects such that, for example, whenever the label `sun` occurs, there will be a light effect. Figure 1 also presents a timeline of sensory effect synchronization based on labels recognized during

classification. In the example, it is desired to synchronize the labels `sun`, `snow` and `forest` with the sensory effects of yellow light, cold temperature and tree aroma respectively.

The main issue preventing ML-based content analysis methods from being used for mulsemedia authoring is the lack of description standards dedicated to relating label naming with sensory effects. One question to exemplify this issue is: *which labels can be defined to activate a wind effect?* The label `wind` itself, or a more complex description like `explosion` or `beach`? Another point to keep in mind is the variety of possible labels that can be returned from a classification task. Labels are dependent on the dataset used, the training process and the preference of the ML architect. Furthermore, deciding where to place sensory effects is an often subjective decision-making process that involves an author's preference. Therefore, we argue that the mulsemedia authoring tool should also allow the author to choose which of the returned labels should be used to generate the sensory effects.

Moreover, one of the main drawbacks of ML-based content analysis approaches for mulsemedia authoring is that they are often domain-specific. That is, they only excel in the classification task in the context that they have been trained. For instance, a DNN that was trained to classify content in daylight videos, once embedded into an authoring tool, may become unable to classify future content in darker videos. Thus, the DNN has to be decoupled from the authoring tool. Also, authoring sensory effects is an artistic endeavor that depends on its author's preferences, who may dislike the classification method of a given DNN. So, an effective solution is to enable the author to select which DNN should be used to recognize sensory effects as well as the labels to be related to a given effect.

As seen in this section, ML-based content analysis can be used to aid the authoring of sensory effects, but there are several challenges when pairing such methods with mulsemedia authoring tools. For the reasons discussed previously, a more efficient approach could be the integration of different content analysis to give the user (*i.e.*, the mulsemedia author) a flexible tool able to work efficiently across different contexts. At the same time, mulsemedia tools incorporating content analysis should provide a mechanism for adapting the response of such networks to annotate sensory effects on the timeline. Therefore, we outline a method to build a component inside a mulsemedia authoring tool, which allows this tool to employ content analysis for sensory effect recognition. In this work, we call such module a *content-driven component* (CDC), which is discussed in detail in the next section.

V. *Content-driven component* design

The proposed *CDC* offers the possibility of semi-automatic sensory effect authoring in a mulsemedia authoring tool. It enables the author to request the recognition of the moments that sensory effects should occur according to certain audiovisual content. *CDC* abstracts from the authoring tool the specificity of the content analysis algorithm. It is important to note that any content analysis method can be utilized with *CDC*, as long as labels are returned. In this work, we will focus on ML-based content analysis, more specifically, DNNs. In the following paragraphs, we will outline the *CDC* design for integrating recognition capacities of DNNs into mulsemedia authoring tools.

The first step is the communication with a given DNN, which is performed using the DNN's available Web APIs. The information of a specific service is defined in *CDC*'s configuration file. For each media type, the file indicates the API to be used and, for each available API, its URL, a request template, and (if required) an authorization header. If the DNN API enables some form of customization, the configuration file may also contain specific API parameters such as the time interval between each returned set of labels. By default, we assume that this time interval is 1 second. Whenever specific media have to be recognized, *CDC* converts it to *base64*, appends it to the request template available in its configuration file, and sends a POST request to the DNN API URL. The above communication method was selected because it provides easy access, easy configuration and reusability of several services for recognition. For example, the authoring tool can access a DNN API in the same machine, in a different machine, in a local network, or in a cloud service.

When using *CDC*, the mulsemedia author should initially choose which media object and which sensory effect types to recognize. Having received the recognized set of labels, *CDC* correlates them with the chosen sensory effect types. Then, a sensory effect is added to the temporal view of the tool when a related label is found. As presented in the previous sections, it is important to allow the author to choose which labels returned from the API should generate sensory effects. Therefore, *CDC* provides a dictionary of labels related to sensory effect types. A dictionary snippet can be seen in Listing 1.

Listing 1: Dictionary of labels for sensory effects.

```
{"WIND": ["air", "storm", "flight"],
"VIBRATION": ["action", "explosion",
    "crash", "calamity", "motion"],
"TEMPERATURE": ["heat", "cold", "sun",
    "snow", "summer", "winter"],
"AROMA": ["trees", "garden", "forest",
```

```
7      "flower"],
8  "FLASH": ["lightning", "gunshot"],
9  "FOG": ["fog","smoke"]}
```

The relationship between labels and sensory effects can be changed by the author, by editing the dictionary file. One should note, however, that the author has to know beforehand the possible labels used by a given DNN. The decision for the dictionary solves two problems related to sensory effect authoring using ML-based content analysis. The first is to give the application author greater control over which labels represent sensory effects according to his/her preference. The second purpose of this dictionary is to enable tool interoperability, as the file can be adapted to match any labels that are returned from the chosen DNN; additionally, each DNN architecture may follow a different label nomenclature.

Figure 2 presents an activity diagram illustrating the semi-automatic authoring process supported by *CDC*. The process begins with the author placing media objects in the temporal view of the authoring tool. Then the author can select which media to use for sensory effect extraction and the desired sensory effect types to be recognized. *CDC* will communicate with a DNN API, taking into account the defined parameters in the configuration file. The chosen DNN will analyze the media and return labels. *CDC* will then create sensory effect instances according to the labels found and author's preferences.

The crucial part of this process is when *CDC* sets the sensory effect instance activation times. This is performed as presented in Algorithm 1. The algorithm takes as input a type of sensory effect chosen by the author to be recognized (*effectType*), the dictionary that maps sensory effect types to labels (dictionary), and a list of identified labels received from the scene recognition software organized by time intervals (*returnedLabels*). Here we refer to each item in *returnedLabels* as an *instant* of recognition of the DNN API. Firstly *CDC* converts *returnedLabels* to a time interval of 1 second between *instants*. This is achieved by merging or duplicating *instants* if the original time intervals were, respectively, lower or greater than 1 second.

The algorithm checks, for each *instant*, if at least one of the labels corresponding to the sensory effect type, as defined in the dictionary, is found. The first time a label is found (Lines 4 and 5), its corresponding time is stored in *startTime*. The algorithm continues analyzing *instants* until no more corresponding labels are found. If *startTime* is defined (Line 9), which means that a sensory effect instance was found, *endTime* is defined as the current time when no more labels were found. Consequently a new sensory effect instance is created (Line 11). This process continues until the last *instant* in *returnedLabels*, possibly creating more sensory effects of the same type in the process.

This algorithm is called for each effect type chosen by the author. After that, the authoring tool timeline is populated with sensory effects. From this moment onward, the author can make any necessary adjustments or finish the authoring process.

Algorithm 1: CDC sensory effect creation process.

```
   Input:
       effectType : Sensory effect type
       dictionary : Dictionary of labels related to
       sensory effects
       returnedLabels : List of identified labels
1  startTime ← endTime ← -1;
2  effectLabels ← dictionary.getLabels(effectType);
3  for t ∈ [0, returnedLabels.duration] do
4      if returnedLabels[t] ∈ effectLabels then
5          if startTime = -1 then
6              startTime ← t;
7          end
8      end
9      else if startTime ≠ -1 then
10         endTime ← t;
11         new sensoryEffect(effectType, startTime,
            endTime);
12         startTime ← endTime ← -1;
13     end
14 end
```

VI. Implementation of CDC in STEVE 2.0

The *CDC* design was implemented in the STEVE 2.0 mulsemedia authoring tool. STEVE 2.0 was selected because it provides an API for creating new elements in its timeline. In this implementation, *CDC* is called STEVEML (STEVE Machine Learning). This section presents how manual authoring is done in the STEVE 2.0 tool as well as the semi-automatic authoring proposal using STEVEML.

The graphical interface of STEVE 2.0 can be seen in Figure 3. In the interface, the media repository at the upper left corner allows the author to import media objects into the graphic environment. In the center, we see the panel to edit properties of the media objects and sensory effects. In the upper right, there is the preview screen for mulsemedia applications displaying their audiovisual content. The temporal view is presented at the bottom of the screen . This temporal view corresponds to an event-based timeline where nodes are synchronized using event-based causal relationships. These relationships and the entities that represent the mulsemedia application in STEVE 2.0 are defined by the MultiSEM [6] mulsemedia model.

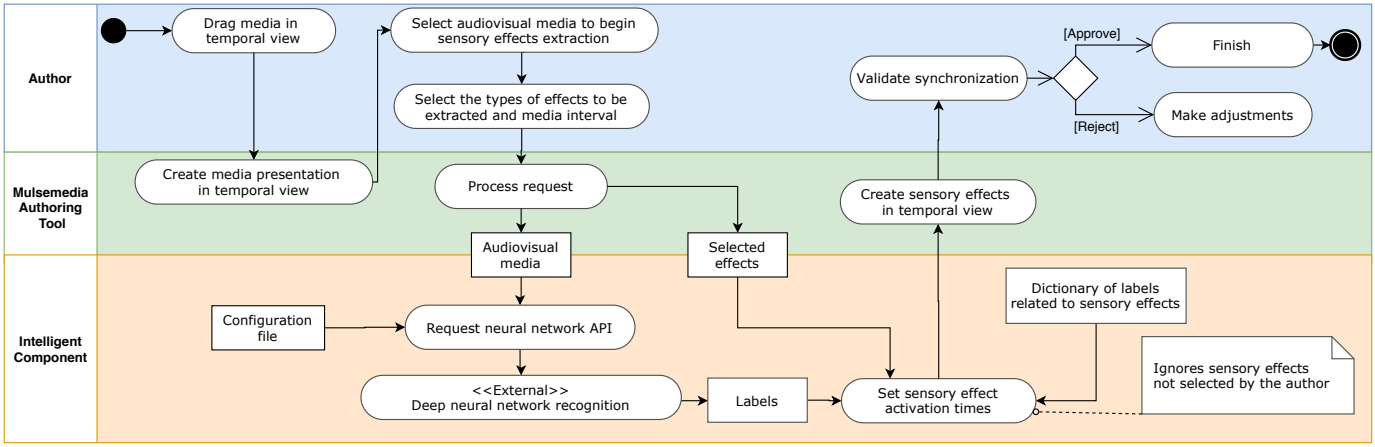

Fig. 2: Activity diagram of the sensory effect authoring process in a mulsemedia authoring tool using *CDC*.

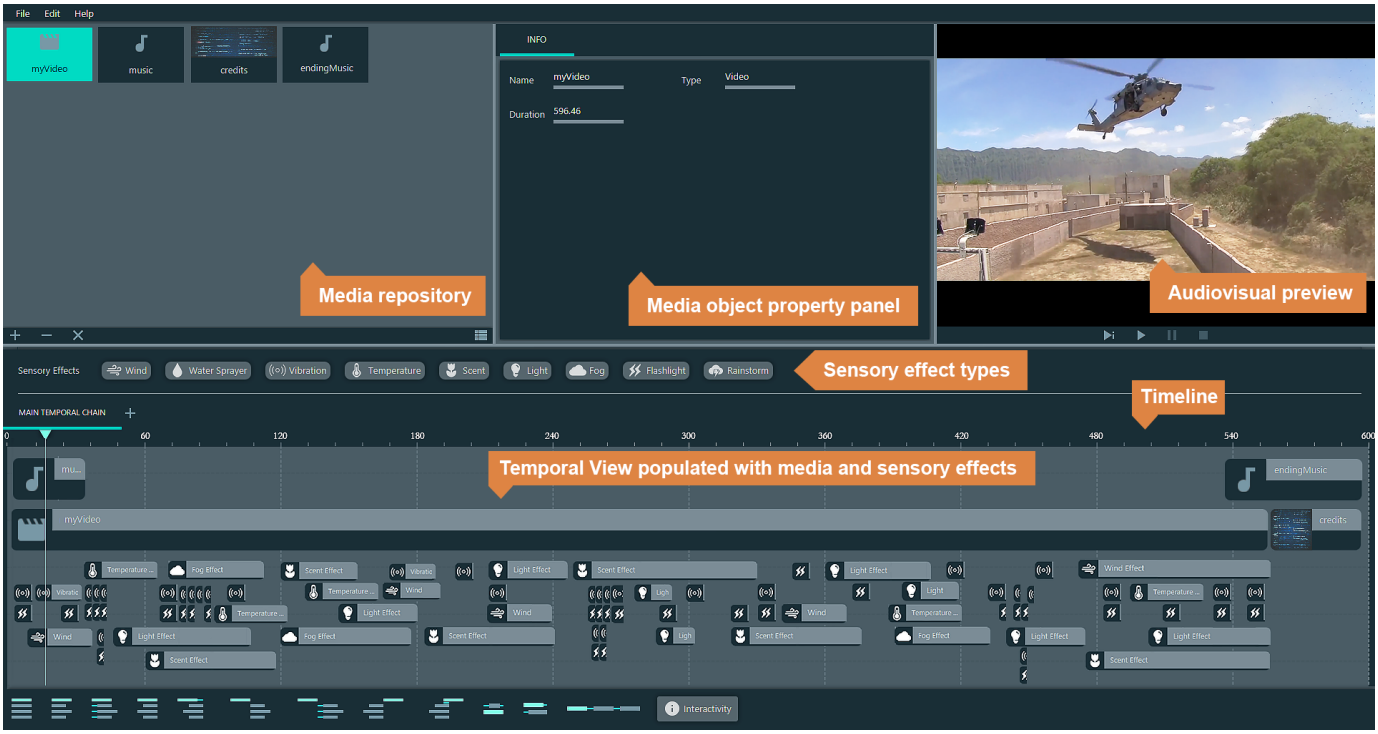

Fig. 3: STEVE 2.0 tool interface and result of STEVEML execution with a main video.

From the media repository, the author can select a particular media, drag it into the temporal view and create temporal relationships with other nodes present in the timeline. To support sensory effects, STEVE 2.0 presents a list of sensory effect types above the temporal view so that authors can also drag a certain type of effect into the temporal view to create a new instance for the selected sensory effect. STEVE 2.0 allows the addition of wind, water sprayer, vibration, temperature, aroma, light, fog, flashlight and the composite storm effect (*rainstorm*). The storm effect encompasses the effects of spray, flashlight, and smoke.

The process of authoring sensory effects manually is carried out by dragging the sensory effect icons and placing them at the timeline. As soon as the author drags the icon to the timeline, a standard-duration sensory effect is inserted. The author can click on the effect icon in the timeline and change its properties, *e.g.*, its duration.

The process of semi-automatic authoring using STEVEML is the following. First, the author selects the media objects and selects the option “AutoExtract Sensory Effects” in the STEVE mouse context menu. Then a pop-up window allows the author to select which sensory effect types should be recognized in the current media object. The pop-up window also allows the author to select which part of the media should be sent for content analysis. After the recognition, the

corresponding sensory effect instances are added to the timeline.[1]

In the current implementation of STEVEML, a cloud-based DNN service for video recognition was used.[2] The chosen neural network API provides a free plan that allows the recognition of 5,000 seconds of video per month. It used the *general* recognition model which can return over 11,000 different labels. Notice that no neural network training was required. The DNN service returns a set of labels for every single second of video. That is, the DNN recognizes labels in the video for every second of content. According to the work in [2], 1 second can be considered reasonable for the synchronization of most sensory effects. Besides, given that the focus of this paper is to provide authoring support, manual editing after automatic recognition is expected.

VII. Usage Scenario

A usage scenario has been set up to illustrate the benefit that semi-automatic authoring brings to a mulsemedia application. In this scenario, the author wants to synchronize sensory effects with a video of approximately 9 minutes. This video is composed of action scenes in various environments, spanning forests, fields, deserts, and snow. To enhance the user experience, the author wants to add sensory effects of vibration, flash, temperature, aroma, wind, and fog.

A timeline representation of the video content and its synchronization with sensory effects is presented in Figure 3. It shows the STEVE 2.0 temporal view with the sensory effects already generated using the STEVEML component. This figure shows a total of 89 sensory effects synchronized with the main video.

All 89 sensory effects were identified within an ≈ 8.0 seconds response-time window. This time is primarily composed of the time to send the video content to the API, the processing time of the DNN, and the time to return the set of labels to STEVEML. Sensory effects were identified following the corresponding labels defined in the dictionary shown in Listing 1. In the following paragraphs, we discuss how STEVEML handles the authoring challenges raised in Section II.

(1) Synchronizing recurring effects. In this example scenario, there are 58 vibration and flash effects related to gunshots, explosions, and crashes. The DNN returns labels related to these events every time they are recognized in the video, *e.g.*, labels `explosion`, `gunshot` and `crash`. These labels are used to create and synchronize the vibration and flashlight sensory effects in the authoring tool temporal view.

[1]We invite the reader to watch the accompanying video showcasing STEVEML in https://youtu.be/0OziKkuMeVQ

[2]https://clarifai.com

(2) Loss of synchronization. Suppose a change in the media content inserted a 16-second snippet beginning at the 3-minute mark. Therefore, all previously annotated effects from this point onward lose their synchronization with the media object. To get the initial synchronization of the sensory effects again, the author should rerun STEVEML. That is, with STEVEML, synchronization does not depend on the occurrence times, but rather on the media content. Once the content is identified (*i.e.*, obtained from the labels), the corresponding sensory effects are inserted in the timeline. Also, with STEVEML the author can select only a part of the media content that was changed to perform effect recognition again.

(3) Changing effects. Suppose the author wants to make a change to the mulsemedia application. In addition to flash and vibration effects, the author also wants to synchronize temperature (heat) effects whenever explosions and gunshots occur. In this case, the author needs to go through a two-fold process. First, the author has to modify the JSON file to relate the labels `explosion` and `gunshot` to the temperature sensory effect. Then the author starts the process once again for the desired media object, after selecting the temperature effect to be recognized. Therefore, STEVEML automatically extracts and synchronizes temperature effects to the timeline along with previously created effects.

VIII. Evaluation

An evaluation experiment was set up to compare automatic recognition to manual authoring of sensory effects. The evaluation was performed on the sensory effects *dataset* presented in [18]. The *dataset* contains videos and related annotations of sensory effects according to the MPEG-V standard. The sensory effects present in the *dataset* are wind, light, and vibration. Light effects are defined by the `autoExtraction` attribute and wind and vibration effects are annotated manually. From this *dataset*, the *Action* subset containing 38 videos was selected. Among these videos, 3 videos were excluded. One was excluded because it is a video without related manual annotation. Two videos were excluded because they represent animations, which are not handled by the current DNN used in the evaluation. The remaining 35 videos last between 6 and 135 seconds.

The DNN service API already available in STEVEML was used in this evaluation. Since this API only identifies the visual part of a video, the evaluation focused only on the vibration effect. The labels defined as related to the vibration effect were *calamity, motion*, and *action*.

Following the approach proposed here, the recognition of a label related to a sensory effect is indicative

that a certain video segment can be synchronized with a sensory effect. Therefore this evaluation assesses which manually annotated sensory effects were also recognized by the DNN. After the authoring done by STEVEML, the intersection of automatically generated effects with the manual authored effects is assessed. As discussed in [2], it is estimated that 1 second after the video scene, the vibration effect can still be considered in sync by users. In light of this, we considered that an intersection can occur from 1 second before or after the end of manual vibration annotation.

Figure 4 shows the ratio between the match of automatic authored sensory effects and the manual authoring for each analyzed video[3]. A match is defined as an intersection between a given automatic annotation with one or more manual annotations. The processing of the videos through the DNN (including the API call) took an average response time of 8 seconds, and the average match rate was ≈ 61.4%. It is important to note that, of the 10 videos that got 100% matches, 5 contained more automatic annotated effects than those manually annotated. These annotations can be false positives of some neural network label. It is also possible that a label has been identified correctly, but the authors in [18] have not annotated the corresponding effect.

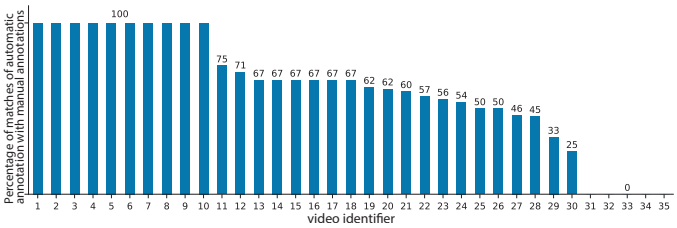

Fig. 4: Percentage of matches of automatic annotation with manual annotations for each video of the dataset.

The remaining 5 videos that got 100% matches had fewer effects than manually annotated. This is because a single effect found by STEVEML may encompass a time frame in which the human author wishes to annotate 2 or more effects (*e.g.,* with different intensities). This difference refers to the need for fine-tuning by the author after automatic recognition. To highlight this need, we see an example in Figure 5. The figure shows the comparison of manual and automatic annotations in video number 18 ("babylonad 1 d"). In this example, there was ≈ 67% match. That is, of 9 manually annotated sensory effects, 6 have an intersection with the annotation obtained from the DNN. Indeed, it was identified that the DNN recognizes explosions and destruction events well by categorizing them as '*calamity*'. In second 2 of the video, a bomb

[3]We invite the reader to explore the name of the videos and additional data at http://bit.do/e3zWa

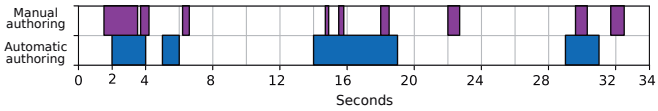

Fig. 5: Timeline of manual and automatic annotation of vibration effects in video "babylonad 1 d".

blast occurs and the DNN correctly identifies that a vibration effect could be placed at that time. In contrast, manual authoring split this effect into 2 vibration effects. On the other hand, some effects present in the manual annotation were not identified by the DNN API. For example, the end of the video has a vibration annotation associated with a submarine emerging from the ice, but the neural network did not identify any labels related to the vibration effect.

Experimental results show that the authoring approach of using automatic sensory effect recognition is a viable alternative to support the mulsemedia content authoring process. Using an automatic method to indicate sensory effect activation intervals can be a starting point for mulsemedia content annotation. Particularly, such an approach avoids the need for the author to undertake the repetitive task of marking sensory effects.

IX. Conclusion

This article presented key challenges for mulsemedia authoring tools. The novel approach proposed here addresses them by allowing semi-automatic sensory effect authoring. In particular, the main challenge addressed is synchronizing recurring effects, as the proposed *CDC* provides the time intervals where a given effect was found independent of media duration. Changes in the audiovisual content do not require great authoring effort to correct the application synchronization, just a rerun of *CDC*, thus addressing the other challenges of loss of synchronization and adjusting previous effects. The *CDC* implementation in the STEVE 2.0 authoring tool shows promising results in respect to reducing the authoring effort.

Some limitations are to be noted, though. The current DNN used for evaluation does not recognize labels in various scenarios, such as nighttime and animated videos, making it inefficient to enhance sensory effect authoring in those scenarios. Also, the integration with CDC supports only the identification of moments of activation of sensory effects. However, effects also have specific characteristics that may be recognized automatically, such as intensity, type and position. Thus, a valuable future work is to integrate in mulsemedia authoring tools DNN architectures tailored to specifically recognizing sensory effects and its characteristics in

various scenarios. To enable such integration, the proposed CDC should be improved to be able to parse the response description labels to define sensory effects characteristics.

Finally we propose two promising directions in ML-based content analysis with mulsemedia authoring tools. The first one is utilize DNNs to predict the Quality of Experience (QoE) of the sensory effects experience. The main need in this regard is to build *datasets* of QoE responses to sensory effect for various videos. The second promising direction is to utilize reinforcement learning to adapt and tailor the classification to the author preference. This implies a greater level of complexity, requiring that the authoring tool also responds to API requests, but would greatly improve the outcome of the semi-automatic authoring process.

X. Acknowledgments

This work was carried out with the support of the Coordination for the Improvement of Higher Education Personnel - Brazil (CAPES) - Financing Code 001. The authors also thank FAPERJ, CNPq and CAPES PRINT Program for partially funding this work.

References

[1] A. Covaci, L. Zou, I. Tal, G.-M. Muntean, and G. Ghinea, "Is multimedia multisensorial?-a review of mulsemedia systems," *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, p. 91, 2018.

[2] R. Abreu and J. dos Santos, "Using abstract anchors to aid the development of multimedia applications with sensory effects," in *DocEng '17*. New York, NY, USA: ACM, 2017, pp. 211–218.

[3] M. Waltl, B. Rainer, C. Timmerer, and H. Hellwagner, "An end-to-end tool chain for Sensory Experience based on MPEG-V," *Signal Processing: Image Communication*, vol. 28, no. 2, pp. 136–150, 2013.

[4] J. Lee, B. Han, and C. Seungmoon, "Interactive motion effects design for a moving object in 4d films," in *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology*. ACM, 2016, pp. 219–228.

[5] R. Abreu, D. Mattos, J. A. F. d. Santos, and D. C. Muchaluat-Saade, "Semi-automatic synchronization of sensory effects in mulsemedia authoring tools," in *Proceedings of the 25th Brazillian Symposium on Multimedia and the Web*, ser. WebMedia '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 201–208. [Online]. Available: https://doi.org/10.1145/3323503.3360302

[6] D. P. de Mattos, D. C. Muchaluat-Saade, and G. Guinea, "An approach for authoring mulsemedia documents based on events," in *International Conference on Computing, Networking and Communications, 2020*. IEEE, 2020, p. 7.

[7] D. P. de Mattos and D. C. Muchaluat-Saade, "Steve: a hypermedia authoring tool based on the simple interactive multimedia model," in *Proceedings of the ACM Symposium on Document Engineering 2018*, 2018, pp. 1–10.

[8] G. Blakowski and R. Steinmetz, "A media synchronization survey: Reference model, specification, and case studies," *IEEE journal on selected areas in communications*, vol. 14, no. 1, pp. 5–35, 1996.

[9] S. K. Kim, S. J. Yang, C. H. Ahn, and Y. S. Joo, "Sensorial information extraction and mapping to generate temperature sensory effects," *ETRI Journal*, vol. 36, no. 2, pp. 224–231, 2014.

[10] F. Danieau, J. Fleureau, P. Guillotel, N. Mollet, M. Christie, and A. Lécuyer, "Toward haptic cinematography: Enhancing movie experiences with camera-based haptic effects," *IEEE MultiMedia*, vol. 21, no. 2, pp. 11–21, Apr 2014.

[11] M. N. de Amorim, E. B. Saleme, F. R. de Assis Neto, C. A. S. Santos, and G. Ghinea, "Crowdsourcing authoring of sensory effects on videos," *Multimedia Tools and Applications*, pp. 1–27, 2019.

[12] T. S. Siadari, M. Han, and H. Yoon, "4d effect classification by encoding CNN features," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 1812–1816.

[13] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. S. Iyengar, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Comput. Surv.*, vol. 51, no. 5, pp. 92:1–92:36, Sep. 2018.

[14] T. S. Siadari, M. Han, and H. Yoon, "4d effect video classification with shot-aware frame selection and deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 1148–1155.

[15] Y. Zhou, M. Tapaswi, and S. Fidler, "Now you shake me: Towards automatic 4D cinema," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7425–7434.

[16] R. Abreu, J. dos Santos, and E. Bezerra, "A bimodal learning approach to assist multi-sensory effects synchronization," in *IJCNN '18*. IEEE, 2018.

[17] S. Chen, "Multimedia deep learning," *IEEE MultiMedia*, vol. 26, no. 1, pp. 5–7, Jan 2019.

[18] M. Waltl, C. Timmerer, B. Rainer, and H. Hellwagner, "Sensory effect dataset and test setups," in *QoMEX*. IEEE, 2012, pp. 115–120.