Sketch-On-Video

Manuel Diogo do Nascimento
Instituto Superior Tecnico, Portugal
manueldiogo@gmail.com

Abstract. Sketch-On-Video is a tool inside Mobymedia project and it was developed with a HP grant in collaboration with VIMMI Group to eLerning environment. Although its name sounds like a pen-based application with a simple goal like to manipulate digital ink and to permit to sketch over video, it’s more than that. Sketch-On-Video also named as SOV is a post-production tool to enrich the classes throw annotations and manipulation of many media streams and to publish these classes on the Web to be accessible to the students. Besides, SOV can be seemed also as a hypermedia editor or a authoring tool that permits to create and to publish riches multimedia’s presentations. It is being implemented in C# using Visual Studio during 12 months approximately and we already had a prototype being used in second semester of scholar year on PCM subject.

Key words: Sketch-On-Video, Mobymedia, eLearning, Post-Production Tool, Hypermedia Editor

1 Introduction

Each day we interact with a rapidly growing amount of digital information, of various data types. The computer applications for viewing, manipulating, and annotating some of these data types, such as text and images, have become quite established among the average computer user. Video, however, is a data type that has only recently moved to digital form and it’s the crucial media on SOV execution.

There are definite advantages to emulating the annotation ability on a computer. While real ink annotations often end up in the recycle bin, digital annotations can persist throughout the lifetime of a document. They can be filtered and organized, and, like digital documents, they can easily be shared. Annotations provide a way to add more information on existing documents and it’s useful to many goals: to highlight important pieces of a document, to add notes when we are displaying a document (e.g. in a lecture or presentation event) and to reshape documents for some domains. Annotations can be used to personalize the learning content on the learners as well as on the pedagogical level and by personalization we mean the ability on the part of the learner to interact with

\[1\] Production of multimedia contents.
the content, the way she deems fit.

These are some reasons that motivates the creation of a tool that permits even a simple teacher to enrich his classes with notes of many nature: text, links, video demos and sketches, and to structure them in order to be more convenient to his students learning.

One of biggest goals behind SOV implementation consists on permit teachers enriches study materials provided to students without the need to perform high complexity operations.

![Fig. 1. Sketch-On-Video inputs and output.](image)

We can see in the following diagram (figure 1) what are needed as inputs and what we can obtain as output using SOV. First, we need the class video. Teacher have to record his classes in order to publish it on Web and it will be the main media on SOV. Second, we need the slides displayed in the classroom. These are the slides that were talked about in the class and that we need to synchronize with the class video. Finally, we need the records of slides transition. The timestamps of slides transitions are recorded in an own file (eventstream.xml) through Show’Em application that is used by the teacher in the classroom. Having these inputs we are ready to start the post-production adventure.

\[2\] Application similar to Power Point which teacher can present his slides in the class. Show’Em records the class slides transitions in a file and usually upload that on the server.
At the end, after performing many operations over the input components, there is the publish phase where results a Web archive accessible by Learn’It\(^3\).

It should be noted that the classes have been published even before SOV creation by using ePresence Producer and to replace that tool is another SOV biggest goal. The existing workflow until SOV requeried much more patience and time from user or teacher to be able to publish a single class. With operations like single insertion or repositioning of slides and constant configuration of publishing parameters, Producer demonstrated failures in terms of temporal effectiveness, flexibility and robustness. These are the failures that we hope SOV can bridge and additionally to provide more functionalities to help the user to easily perform a post-production of his presentations or classes.

In the next section of this paper, we discuss related work. In section 3, we provide a detailed description of SOV presenting its main caracteristics. We also discuss the success of Sketch-On-Video based on evaluation studies in section 4. In section 5, we conclude and present our plans for future work.

## 2 Related Work

We have been experienced a constant growth of annotation’s systems number and these ones provide a support for many daily task we do that’s requires notetaking and sketching.

It’s possible to divide these systems in three categories of annotations:

- Private annotation which the objectives are tipically to support active reading\(^4\) and to help to retrieve content after (it includes summaring, searching and tagging).
- Collaborative annotation which is usefull when we need to indicate interisting parts of a document or instructions as a method to cooordinate activities or to perform a real-time notes sharing in a work situation.
- Public or Social annotation which is less directed for team work than collaborative annotations. It consists in public comments or notes that others can see and comment\(^5\).

There are a few tools to support the development of pen-based systems. We will present some toolkits with that goal and after that we present some annotation’s systems.

---

\(^3\) A web application that permits students to access the published classes where they can see besides teacher notes, annotations made by themselves in the class and to perform new ones. It’s also a part of Mobymedia Project.

\(^4\) A combination of reading with critical thinking and learning. It involves not just reading per se, but also underlining, highlighting and scribbling comments.

\(^5\) Similar to a blog. You can post a public text and others can read and comment it.
2.1 Toolkits

Tablet PC SDK[10] was designed to run on TabletPCs and it’s implemented in C#. It manipulates digital ink as a primary object class and contains methods to determinate intersections, ends and when strokes are contained in others. It uses Microsoft handwriting reconigzer and has a recognition module for gestures. That toolkit permits also a rudimentary analysis in order to determinate if a stroke set is a word or just a sketching. Tablet PC SDK is very simple to use and has a support to events control and interact with others Windows components.

TORCH is a machine learning library developed in C++. Although it is not connect directly with real pen-based applications’s development, it provides many learned algorithms by the machine that can be used for recognition. It can be downloaded in [13].

SATIN [7] is a Java-based toolkit designed to support the creation of applications that leverage the informal nature of pens. This support includes a scene graph for manipulating and rendering objects; support for zooming and rotating objects, switching between multiple views of an object, integration of pen input with interpreters, libraries for manipulating ink strokes, widgets optimized for pens, and compatibility with Java’s Swing toolkit. SATIN includes a generalized architecture for handling pen input, consisting of recognizers, interpreters, and multi-interpreters.

2.2 Annotation Systems

In this section, we will refer diverse types of annotation systems, since these that only annotates tagging parts of information till others that permits sketching over many surfaces.

Family Video Archive [1] is presented as a tool to give consumers the ability to annotate and browse large collections of informal family movies. The informal nature of home movies makes it difficult to use fully-automated techniques for scene detection and annotation. This system explores the symbiosis between automated and manual techniques for annotation. It also explore the use of a zooming interaction paradigm for browsing and filtering large collections of video scenes. Similar to this, there is Video Annotator[6], the antV[5] sucessor, a video annotation tool that gives several perspectives over the same video content and Videotater, an experimental tool for a Tablet PC that supports the efficient and intuitive navigation, selection, segmentation, and tagging of video. Also in the video annotation scope we have LEAN[9] that explores a variety of interaction and visualization techniques for fluid navigation, segmentation, linking, and annotation of digital video.

There are some systems more oriented to collaborative tasks like Livenotes[8]
Sketch-On-Video, a authoring tool

a shared whiteboard system and educational practice that uses wireless communication and tablet computing to support real-time conversations within small groups of students during lectures, independent of class size. It enables group members to interact with one another by taking lecture notes cooperatively, as well as to augment student note-taking by providing instructor slides in the background to annotate over. Another system for collaborative tasks but not in eLearning scope is VideoDraw[12], a video-based prototype tool that provides a shared virtual sketchbooks among two or more collaborators. It not only allows the collaborators to see each others drawings, but also conveys the accompanying hand gestures and the process of creating and using those drawings.

Some systems permits public or social annotations important for group discussions. Madcow[2] is one of them, is digital annotation system organized in a client-server architecture, where the client is a plug-in for a standard web browser and the servers are repositories of annotations to which different clients can login. Notelook[4] is very similar to MADCOW since it’s also a client-server system and designed to support multimedia note taking in meetings with digital video and ink. SOV inserts in that kind of annotation tool since teacher’s notes are public to his students.

We have also u-Annotate[3] system in eLearning area as SOV, a user-driven freeform digital ink annotation application for web eLearning content which aims at facilitating the learner to annotate the online content with the aid of pen computing devices such as graphic tablets, etc. Learners can freely mark up the content, save the annotations for recall at a later date, as well as share these with other learners. 

SOV differentiates of all this systems because of its main characteristics: permits to edit video, to synchronize many streams of medias and to make many kind of annotations since text notes, links and video demos as well sketches over slides and video.

3 Implementation

How it was already said before SOV presents caracteristics that make it unique of his kind. SOV, besides of many others important aspects integrates requirements from three kinds of systems: annotation and sketching systems, video editor systems and synchronization systems.

3.1 Sketching Feature

In SOV implementation, we started developing a simple sketch player which allowed sketching over videos and to control digital ink properties like color,

---

6 It’s very similar with Learn’It, a Mobymedia subproject refered before.
thickness, time position and duration. After that, we integrate its functionalities in SOV application. To sketch over video it was needed a transparent panel over video panel, where the strokes could be designed and displayed. It permits to separate video from digital ink that it contains and the video frames remains intact. That’s why time position of these video strokes are important, it permits us to know which strokes need to be displayed on the panel in a determined instant.

When sketching over slides we don’t use the transparent panel, the active slide behaves like a panel background where we can easily draw over. Another difference between video and slide sketching consists in ink properties. Slides’s strokes don’t need to keep time position like video strokes, they save the slide information (name) in which they are over and when the slide is displayed also his sketches are.

All sketches created by the teacher (over slide and video) are saved in XML format. We used Microsoft Ink Library to create and to manipulate digital ink. This library comes from the package Tablet PC SDK, already refered as one toolkit for the development of pen-based applications. Microsoft Ink in addition of capturing the points and allowing us to draw with ink desired attributes as color, thickness or other special attributes created by us (time to start, stop and so on), provides us operations to manipulate strokes that contains (move, resize, delete and select).

### 3.2 Video Edition Feature

SOV allow us to move or delete class segments to better organization of its contents, to insert new pieces of video and also to replace existing audio of a existing chunk of video. The biggest difficulty implementing these operations was performing them taking into account the dependency relationship existing between the class video and others medias connected to it. Each one of these operation had to be made taking in account the existing synchronization between different medias in post-production. For example, it can be inconsistent to delete or move only a portion of video segment that explains a slide. And what about the remaining portion of video that explain only a part of that slide? The explanation to that slide would become insufficient so we choose to consider the duration of the slide as the smallest fragment possible of a class and therefore capable of being erased and moved.

Besides of that, insertion of video for example brings up the problem of different videos’s "physiognomy". It have to be account the codecs, resolutions and ratios of refered videos to know if it’s possible to compose both. Splicer[11] comes to solve not just that problem as also allowed to integrate all the video edition operations. It is a known library to compose video and that permits to add transitions to chunks, a usefull feature to SOV ⁷.

---

⁷ SOV automatically insert fade-in and fade-out transition when a segment is moved, deleted or insert to avoid abrupt frame transitions.
3.3 Multimedia Synchronization Feature

This module is based on ePresence Producer operations, basically that system permits to synchronize slides with the class video. SOV, besides of that synchronizes also notes and sketches. The synchronization could be made using SMIL but it presents restrictions on the number of web players that can support it. Besides of that, ePresence format was already used by us, it’s easy to configure, it’s becoming known and it offers different pages layouts already structured for many kinds of presentations. Each slide, note and video stroke has a timestamp that corresponds to its time position in the class video. SOV obtains slides transitions times recorded by Show’Em and just have to synchronize the first timestamp with the time when it happens in class video and automatically all the following slides are synchronized. Unlike Producer, SOV allows to manipulate slides individually and in group which inserts a certain efficiency and flexibility.

3.4 Others Aspects

There are two important SOV characteristics that have to be mentioned because they permits an improvement in flexibility, efficiency and time to perform tasks as we will show in Study and Discussion section. These two aspects are the implementation of user profiles and SOV predictive behaviour. Using Producer the user have to constantly configure parameters to its publications while SOV with profiles, user just need to create once a profile where he configures all publishing parameters and after it use it whenever he wants to publish with that specifications.

We were able to identify more used functionalities and to detect how to make them more efficient and fast. Because of that, the predictive behaviour was born. Unlike Producer, SOV automatically detects the video actual position and associate that to the created object which reduces time needed to perform the task and avoids errors. The same happens with all time related position (reposition). Besides of automatic detection of actual time, SOV tries to predict which slide will be added next. Slides are numbered and SOV uses that fact to automatically select the next potential slide to add taking in account the last one added.

4 Study and Discussion

We performed three evaluation studies. In the first, SOV was integrated in PCM classes exercises which the students realized some tasks and answered a questionnaire about the experience\(^8\). We obtained very positives feedbacks about the experience with 94% of answers confirming that would use it again for that kind

\(^{8}\) The exercises were performed by 40 students and we get 32 answers to the questionnaire.
Table 1. Measures recorded in each class post-production made by teacher.

<table>
<thead>
<tr>
<th>Class</th>
<th>Errors</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>7</td>
<td>20 minutes</td>
</tr>
<tr>
<td>11</td>
<td>6</td>
<td>18 minutes</td>
</tr>
<tr>
<td>12</td>
<td>5</td>
<td>16 minutes</td>
</tr>
<tr>
<td>15</td>
<td>4</td>
<td>17 minutes</td>
</tr>
<tr>
<td>18</td>
<td>4</td>
<td>15 minutes</td>
</tr>
<tr>
<td>19</td>
<td>3</td>
<td>15 minutes</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>13 minutes</td>
</tr>
<tr>
<td>21</td>
<td>2</td>
<td>9 minutes</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>10 minutes</td>
</tr>
</tbody>
</table>

of presentations. This session of tests was useful to detect existing bugs, to detect interface improvements needed to be made and to obtain new suggestions of functionalities that would be useful to user.

Another evaluation study was made using SOV execution logs that was implemented in the program and recorded during the SOV utilization in PCM classes post-production. We focused in metrics like average time till the publication, total errors made and tasks frequency.

As it is possible to observe in the table 1 both time and errors in a post-production session were reducing during the semester which demonstrates growing familiarity with the system beyond the improvements made in the operations.

More important, logs gave us values to make a benchmarking study with Pro-
ducer. SOV, besides of improving all the Producer functionalities, it provides new ones allowing more flexibility, more efficiency and less time to publish. In the figure 2 we can see a comparison in terms of existing functionalities between SOV and Producer. SOV features refered in section before make it greater than Producer in quality and also in quantity levels allowing a minimization of time needed until publish in order of five to ten minutes.

5 Conclusion

We think that SOV achieved the goals we build it for. It incorporates the concept of an existing tool already used to publish classes, ePresence Producer, improved these tool’s functionalities and in addiction it creates new ones to offer more post-production options to the users. These new functionalities differentiate SOV in eLearning context without adding more complexity to its execution. Taking in account the results obtained in SOV evaluation, we believe that we are in the right path to get a tool with good quality and really usefull to eLearning.

5.1 Future Work

Besides of good results obtained in SOV evaluation we know that there are some future work to SOV improvement as such: to allow execution of multiples instances of SOV, to permit direct manipulation of slide besides manipulation through menus, to implement zoom mechanism over slides for sketching, to add new functionalities to video edition, to perform automatically detection of presentation begging to slides synchronization, to provide a global view of existing sketches in the class and finnaly to be able to perform retrieval of existing notes.

References


