Concurrent Video: Versioning Concepts♣

Oleg Proskurnin

University of St.Petersburg, Russia
olegpro@acm.org

Abstract

Concurrent video is a recently proposed data model especially intended for collaborative authoring environments. Based on cooperative transactional mechanisms, it provides a solid foundation for consistent sharing and exchange of information in multi-user editing systems. The goal of this work is to extend the already presented model with versioning capabilities by utilizing major transactional properties of the original proposal.

1 Introduction

It's a long time since computers were first involved in video postproduction process and today there is a wide range of multimedia editing systems designed for this kind of tasks, e.g. [1, 16]. However, today postproduction typically represents a collaborative activity, while existing authoring applications are mostly intended for individual work.

In fact, joint editing requires efforts of various experts – editors, graphics artists and sound engineers, to name a few, – each contributing his best to the overall process. Apart from emphasizing strengths of different specialists, cooperation encourages each participant to use the right tool for his particular job and generally promises improved workflow with minimized time and resources. These benefits of joint efforts inevitably require from computers and information systems to support collaborative postproduction activities just as well as individual multimedia authoring [15].

Concurrent video data model [11, 12] is especially designed for dealing with this kind of problems, providing a formal technique for joint editing of media streams, such as video or audio. Based on the cooperative activity framework CoAct [13, 19, 10], the model has enabled consistent sharing and exchange of information in the face of concurrent modifications, which is often considered as the key problem of multi-user environments design.

The background is that co-authors in CoAct are supposed to operate in their private workspaces containing local versions of shared objects. Authors' actions are represented via activity histories, which are basically sequences of editing operations constituting individual working process. To achieve cooperation, co-authors explicitly incorporate the results of their actions into the common workspace, which reflects the current state of the collaborative effort. This incorporation is in fact realized by means of the history merging mechanism [19], which takes into account compatibility relations of constituent operations in order to detect possible conflicts and thus ensures semantically correct information sharing and exchange.

To avoid unnecessary conflicts, editing operations developed within the concurrent video model were especially designed to have good commutativity properties. Actually, according to the concepts of timeline-based video authoring [3], these operations allowed participants to work concurrently on different clips [11] and even on non-overlapping frame regions of the same clips [12], which in the end were used to form the desired video material.

The main objective of this work is to improve the original data model as it is presented in [12] with versioning capabilities, i.e. means of handling several copies of the video production, captured at certain stages of its development. In other words, this paper shows how cooperative transactions can be successfully employed for establishing versioning support in a collaborative multimedia editing system, in addition to ensuring consistency of shared data.

At this point it is worth to mention that version control is typically considered as a serious advantage in authoring environments – it helps to keep track of the editing process, provides starting points for arranging exploratory changes and enables evaluation and comparison of individual contributions to the final product made by particular participants [17]. Taken together, these benefits represent a clear motivation for the present investigation.

The rest of the paper is organized as follows. After giving a brief overview of the related work, the improved concurrent video data model is introduced. Next, various aspects of versioning support are discussed and finally conclusions and further work are outlined.

2 Related Work

This section first presents a summary of previous work on collaborative multimedia authoring and then briefly mentions approaches to establishing versioning support in the context of various editing systems.

Proceedings of the Spring Young Researcher's Colloquium on Database and Information Systems SYRCoDIS, St.-Petersburg, Russia, 2004

2.1 Collaborative Video Authoring

Although a variety of multimedia authoring tools are available in the market, e.g. [1, 16], none of them supports concurrent video editing and the related research works are also very few.

Some of these proposals investigate cooperative activities exploiting lock-based mechanisms [2, 18], while other approaches consider collaboration in real-time groupware systems only [22]. In contrast to the concurrent video model [11, 12], these works do not cover multimedia modeling issues and do not provide consistency guarantees in a transactional sense.

Similarly, investigations examining various aspects of video data management, e.g. [20, 7], turn out to be different from the model described here in its both structural and operational aspects.

Actually, the major distinction between concurrent video and former models lies in the suitability of the new proposal for the basic needs of both video authoring systems and cooperative environments, which has not been achieved ever before.

The point is that in the considered authoring systems the movie is created by altering and assembling source clips which are treated as independent units of raw material. These clips along with other media objects are placed on the timeline, which naturally represents the time flow. After the inclusion, objects can be edited, special effects and transitions can be applied and the resulting material can be arranged into a final video production. In addition, since such editing implies just referencing the original files, which are not actually modified, a special process called rendering is required to preview or export the produced movie.

Concurrent video takes into account these basic concepts of timeline-based authoring and provides an efficient tree-based abstraction for referencing a sequence of media clips. It also presents a set of high-level non-destructive editing operations allowing users to insert and delete, alter and temporally combine video material to form the final production.

To support joint activities, operations on different clips and on different parts of the same clips are considered to be commutative, such that users can work on non-overlapping parts of the media stream concurrently without any conflicts. Moreover, hierarchical structure that references video data also permanently stores activity histories, providing elegant support for cooperation mechanisms of CoAct.

The latter fact concerning history storage is actively exploited in this work since it provides a proper basis for the development of versioning support on top of the already presented model.

2.2 Versioning

There exists a wealth of research proposals offering various approaches to versioning in the context of diverse fields of application, such as CAD engineering environments [4], hypermedia [9] or multimedia [2] document authoring, XML data management [21] and object-oriented database systems in general [14].

However, the crucial difference between the presented investigation and previous works on versioning lies in the area of its application. On the one hand, version control over the stream types of data, such as video or audio, is considered, and on the other hand, it is examined in the context of cooperative transactional mechanisms, which provide solid support for truly concurrent editing activities in corresponding multi-user authoring environments. In fact, versioning has never been explored within such conditions ever before.

Nevertheless, version control backed up by concurrent editing for plain text data is well-known in the area of software configuration management [5] and is implemented in numerous commercial and open source applications, such as CVS [6]. Therefore, it looks natural to employ some basic ideas found in this field in collaborative video authoring systems empowered with versioning capabilities.

Actually, a recent survey [5] to a certain extent summarizes the previous research on software configuration management. This survey provides an overview and classification of different versioning paradigms, presenting a unified terminology for major related concepts such as versions, deltas, state-based and change-based versioning, etc. Some of these concepts are introduced and explained in the following material, as they prove to be useful in cooperative video authoring environments as well.

3 Concurrent Video as a Data Model

This section outlines an improved concurrent video model, which is entirely based on its already presented counterpart [12] and has a potential for version control.

3.1 Video Segments

In the considered model video segments represent an abstraction over independent units of video data, which are used as building blocks in the authoring process.

Actually, both the definition and the purpose of video segments in this work stay the same as in the previous proposals [11, 12], however, the following facts are quoted here for convenience.

Basically, each video segment references a contiguous part of raw material via a frame sequence and has its own set of attribute-value pairs which describe the proper interpretation of the underlying media at the presentation or rendering level. Frame sequences reflect the stream-like nature of video data, while attributes support implementation of non-destructive editing operations as well as specifications of desired transitions and special effects for the segment.

Definition 1 (frame sequence) A frame sequence is a finite non-empty sequence $(f_1, \ldots, f_N)$ of video frames f_i, referring to a contiguous block of video data. N is called the length of a sequence F and is denoted by $Length(F)$.

Definition 2 (video segment) A video segment is a pair (F, A), where F is a frame sequence and A is a possibly empty set of attribute-value pairs $\{a_i{:}v_i\}$, storing various additional information about the containing segment.

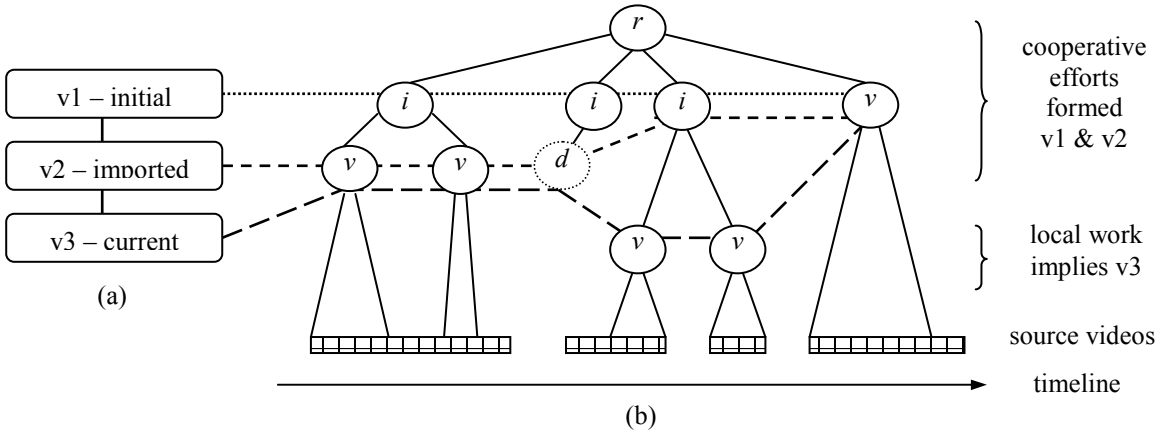

Figure 1. Concurrent video aspects: version model (a) and data model (b)

3.2 Video Activity Tree

Concurrent video employs a single hierarchical data structure, called video activity tree, for modeling both underlying media and cooperative transactions.

Basically, the structure of an activity tree in this work remains the same as before [12]: leaf nodes, called valid, are associated with video clips that authors currently see on a timeline, dead nodes stand for previously deleted clips and intermediate nodes hold related parts of the activity history. Also, activity history elements are marked whether they are private or shared to indicate what part of the tree is present in the common database and what part exists only in the considered local workspace, as illustrated in figure 1-b.

However, for the purpose of versioning support it is necessary to introduce two corrections into the structure of the video activity tree. First, not only valid, but also all intermediate nodes should reference corresponding parts of the underlying raw material by means of video segments. Second, no node is allowed to contain more than one operation instance. These rules are reflected below in the redefinition of node's structures.

Definition 3 (video activity tree node) A valid node *V* of a video activity tree is a tuple (*NID*, *Range, Segment*, *Instance*), where *NID-Range* pair uniquely identifies *V*'s associated video data, *Segment* is the video segment representing this data and *Instance* is an operation instance (if any) related to the given node *V*.

Self-evident modifications are also required in algorithms describing clip editing and deletion methods in order to avoid storage of multiple operation instances in a single node – basically, in such cases these algorithms should create a child node to store extra information.

4 Concurrent Video as a Version Model

As a matter of fact, concurrent video naturally reflects the evolution of the video production from its initial version to the final stage, capturing all details of the editing process. This is achieved due to activity history, which is stored within the video tree in a way preserving inter-operation dependencies [11].

As a result, particular versions of the video production can be formed by linking the set of leaf nodes of the activity tree with a usual list structure at desired time points, as illustrated in figure 1. During further editing the tree inevitably "grows" downward, but any previous version can be easily reconstructed by mere walking through the corresponding list and looking at the encountered segments.

Typically, a new revision will be formed in the common workspace each time somebody commits the results of his individual work, while the initial version will come from the initialization method [11].

Accordingly, information exchange between workspaces [10] in an environment with versioning support will be optimized by taking into account the common part of the activity history, i.e. the common version, present in both workspaces. And subhistory extraction algorithm from [11] can be readily adapted to such conditions by forcing to end its work after reaching a certain common revision instead of the root node.

It is worth to mention, that in terms of [5] concurrent video provides both state-based and change-based version control. The first one is realized via the lists mentioned above, which fix particular states of the video production. The second one occurs since the differences between any two versions can be described by a portion of an activity history, located in the video tree between the appropriate versions lists.

Moreover, concurrent video can be seen as a version model [5], which contains an explicitly specified sequence of revisions, as illustrated in figure 1-a, and provides various operations for retrieving old versions of media production or constructing the new ones (of course as a result of editing activities).

Finally, there are no reasons why concurrent video can not be extended to support branches [5], i.e. separate lines of development having their own revision history. Branches may be helpful in arranging dynamic subgroups of co-authors who wish to share intermediate results of their work. Eventually, all changes made within the branched revisions can be merged back into the mainstream line of development with the help of the regular CoAct merging mechanisms.

5 Conclusions and Further Work

In this paper, the concurrent video data model has been extended with basic versioning support, thus introducing benefits of version control to the field of collaborative video editing environments.

It is worth to mention that the results of this work are applicable to stream data in general. For instance, by renaming frame sequences and video segments to sample sequences and audio segments, the concurrent video model is transformed into concurrent audio, which may be found useful in certain authoring systems.

As a possible direction of further research, investigation of compensation and differencing facilities on the basis of this work is considered.

Besides, development of a prototype system demonstrating feasibility of the presented approach is regarded as an essential part of the future work.

For this purpose, there is an intention to utilize the recently developed Advanced Authoring Format [8, 15] and its open source SDK. Basically, the AAF file format is an industry-driven standard especially designed for interchange of compositional meta-data between various multimedia authoring tools. At the moment of writing this paper, several video editing applications have already claimed a certain level of compliance with AAF. In particular, Adobe Premiere Pro [1] is able to export its projects in this novel form.

Acknowledgements

Special thanks are expressed to Boris Novikov for encouragement and discussions on the topic.

References

[1]. Adobe, Adobe Premiere Pro 7.0, 2004 http://www.adobe.com/premiere

[2]. Borghoff, U.M., Teege, G.: Structure management in the collaborative multimedia editing system IRIS. In Proc. of the International Conf. on Multi-Media Modeling, pages 159-173, Singapore, 1993

[3]. Bulterman, D.C.A., Hardman, L.: Multimedia authoring tools: State of the art and research challenges. In Lecture Notes in Computer Science #1000, Springer-Verlag, 1995

[4]. Chou, H.T., Kim, W.: A unifying framework for version control in a CAD environment. In Proc. of the 12th International Conference on Very Large Databases, pages 336-344, Kyoto, August 1986

[5]. Conradi, R., Westfechtel, B.: Version models for software configuration management. In ACM Computing Surveys, Vol.30, No.2, 1998

[6]. CVS, Concurrent Versions System, 2004 http://www.cvshome.org

[7]. Dumas, M., Lozano, R., Fauvet, M.-C., Martin, H., Scholl, P.-C.: A sequence-based object-oriented model for video databases. Multimedia Tools and Applications, Vol.18, Issue 3, pages 249-277, 2002

[8]. Gilmer, B.: AAF – The Advanced Authoring Format, 2002 http://www.aafassociation.org/html/techinfo

[9]. Haake A., Haake J.M.: Take CoVer: exploiting version support in cooperative systems. In Proc. of the SIGCHI Conference on Human Factors in Computing Systems, pages 406-413, Amsterdam, The Netherlands, April 1993

[10]. Klingemann, J., Tesch, T., Wasch, J.: Semantics-based transaction management for cooperative applications. In Proc. of the International Workshop on Advanced Transaction Models and Architectures, pages 234-252, Goa, India, August - September 1996

[11]. Novikov, B., Proskurnin, O.: Towards collaborative video authoring. In Proc. of the 7th East-European Conference on Advances in Databases and Information Systems, pages 370-384, Dresden, Germany, September 2003

[12]. Proskurnin, O.: Concurrent video: operational extensions. To be published In Proc. of the 6th International Baltic Conference on Databases and Information Systems, Riga, Latvia, June 2004

[13]. Rusinkiewicz, M., Klas, W., Tesch, T., Wasch, J., Muth, P.: Towards a cooperative transaction model - The cooperative activity model. In Proc. of the 21st International Conference on Very Large Databases, pages 194-205, Zurich, Switzerland, September 1995

[14]. Sciore, E.: Versioning and configuration management in an object-oriented data model. VLDB Journal, Vol.3, pages 77-106, 1994

[15]. Turner, B.: A new take on teamwork. Video Systems, December 2002 http://videosystems.com/ar/video_new_teamwork

[16]. Ulead, MediaStudio Pro 7.0, 2004 http://www.ulead.com/msp

[17]. Vitali, F.: Versioning hypermedia. In ACM Computing Surveys, Vol.31, No.4es, 1999

[18]. Wang, K.: The design of extending individual multimedia authoring to cooperative multimedia authoring. In Proc. of the Fourth International Conference for Young Computer Scientists (ICYCS'95), Beijing, China, July 1995

[19]. Wasch, J., Klas, W.: History merging as a mechanism for concurrency control in cooperative environments. In Proc. of RIDE-Interoperability of Non-traditional Database Systems, pages 76-85, New Orleans, USA, February 1996

[20]. Weiss, R., Duda, A., Gifford, D.: Composition and search with a video algebra. IEEE Multimedia, Vol.2, No.1, pages 12-25, 1995

[21]. Wong, R.K., Lam, N.: Managing and querying multi-version XML data with update logging. In Proc. of the ACM Symposium on Document Engineering, pages 74-81, McLean, USA, 2002

[22]. Xiao, B.: Collaborative multimedia authoring: scenarios and consistency maintenance. In Proc. of the Fourth International Workshop on Collaborative Editing Systems, New Orleans, USA, November 2002

♣ This work is partially supported by the Russian Foundation for Basic Research under grant No. 04-01-00173