Microsoft Research Technical Report MSR-TR-2005-154, November 2005

LazyCut: Content-Aware Template Based Media Authoring

Xian-Sheng Hua, and Shipeng Li
Microsoft Research Asia
5F Sigma Center, 49 Zhichun Road, Beijing 100080, P.R.China
{xshua, spli}@microsoft.com

ABSTRACT

The rapid adoption of digital cameras and camcorders leads to an explosive growth of personal photos and home video in digital form. There is a huge demand for new tools and systems that enables average users to more efficiently and more effectively process, manage, author and share these digital media contents, in particular, a powerful video authoring tool that can dramatically reduce the users' efforts in editing home video. Though there are many commercial video authoring tools available today, video authoring remains as a tedious and extremely time consuming task that often requires trained professional skills. To tackle this problem, this paper presents a novel end-to-end system that enables fast, flexible and personalized video authoring and sharing. The novel system, called LazyCut, is based on both content analysis techniques and the proposed content-aware two-layer authoring templates: content description template and content representation template. Moreover, it is designed as an open and extensible framework that can support dynamic update of core components such as content analysis algorithms, editing methods, and the two-layer authoring templates. Furthermore, the two layers of authoring templates separate the video authoring from video representation. Once authored with LazyCut, the video contents can be easily and flexibly represented in other forms according to users' preference. LazyCut provides a semi-automatic video authoring and sharing system that significantly reduces users' efforts in video editing while preserving sufficient flexibility and personalization.

Keywords

Multimedia authoring, video editing, multimedia content analysis, multimedia management, template.

1. INTRODUCTION

The recent introduction and rapid adoption of consumer digital photo cameras and video camcorders has redefined the landscape for media (photo and video) management, authoring/editing and sharing tools. Existing video editing tools can be roughly classified into two categories. One is high-end professional editing systems, such as Adobe Premiere [1], and Sony Vegas [17]. The other is middle- or low-end editing tools, such as Ulead VideoStudio [20], Apple iMovie [2], Microsoft Movie Maker [13], and Muvee autoProducer [14]. Though we have sufficient flexibility and editing power with the first category of editing systems, complex editing skills and aesthetic sense are typically required. At the same time, it also requires great effort to learn how to use the system while the task of video editing remains time-consuming and tedious.

On the other hand, the second category of editing tools make the task of video editing much easier, thanks to many of them support automatic video editing methods which are driven by intelligent multimedia content analysis technologies [2][13][14]. Moreover, a couple of automatic or semi-automatic editing methods which tackle the same issue as these tools are also available in literature. The Hitchcock [5][6] system detected suitable clips according to numerical "unsuitability" scores. These clips are then organized by the users in a storyboard and concatenated automatically as the final video. The created video can be regarded as a dynamic summary based on users' interest. In [4], an automatic music video generation scheme is presented, in which content selection is based on calculation of video unsuitability relating to camera motion and image contrast. One system named AVE [9] provided dynamic highlights of home video content by selecting desirable high-quality clips and linking them with transition effects and incidental music. Another system also called AVE proposed in [21] is another yet similar automatic video editing method. However, these methods suffer from low editing flexibility and often unsatisfying editing results, though remarkable time and effort are saved which we will also take advantage of in this paper.

As we know, there are many templates-based text authoring tools available such as Microsoft Office (Word, Excel, PowerPoint, Project, Visio, InfoPath, etc.). These authoring templates have significantly improved the efficiency of office workers. Inspired by this, in this paper, we propose an end-to-end content-aware template based video authoring and sharing system, named *LazyCut*, which makes video authoring much more efficient and flexible. This system addresses the disadvantages of both manual and automatic video editing schemes. The goal is to provide a semi-automatic video authoring and sharing system that minimizes users' effort in the process of editing while preserving sufficient flexibility and personalization.

In fact a few "templates"-like methods for video authoring have already appeared in some commercial products [20]. However, these "templates" are only for some special video effects, such as making sepia tone and old movie effects, adding frames/borders for photos or videos, drawing graphics or animations over photos or videos, or generating DVD menus. They really have not tackled the most difficult issues in video authoring such as storyline generation, and content selection.

In the proposed LazyCut system, the primary method to accomplish fast and flexible video authoring is based on content-aware editing templates. In LazyCut, a two-layer authoring template scheme is adopted: one layer is Content Description Template (CDT), and the other one is Content Representation Template (CRT). The former one describes the temporal structure

of the video to be authored (i.e., storyline), content composition methods, rules and preferences, while the latter one specifies how to represent or render the content authored with CDTs, which might be a DVD, a website, slides for presentation, a DirectShow timeline [16], etc. In professional film editing, creating a reasonable and compelling storyline is the most important thing for the final result. As it is extremely difficult to achieve automatic storyline generation using existing multimedia content analysis techniques, we use CDT to define the temporal structure of the final video. And, editing methods (such as autoProducer [14], AVE [9] [21], Photo2Video [10] or manual editing) along with their corresponding editing parameters (such as style, duration, incidental music, and so on) are also defined for each temporal unit (we call it *slot*) in CDT. After feeding the original raw media data (video segments and/or photo collections) into slots, the content (selected video segments [9], photos with motion effects [10], and so on) that will be included in the final result is determined by the specified editing method.

With CRT, the edited content (according to selected CDT and users' inputs) can be represented in different forms, such as a streaming video, a DVD/VCD, a webpage or website, a video album[7][8], and so on.

The rest of this paper is organized as follows. Section 2 introduces the system architecture of LazyCut along with several key characteristics of the system to help better understand the architecture design. Section 3 reviews the basis of the system, Media Content Analysis, including the open and extensible framework for content analysis and content filters currently supported. The Content Description Template scheme and the Content Representation Template scheme are proposed and described in detail in Section 4 and 5, respectively. Section 6 shows how we can update and extend the LazyCut system with additional plug-ins. Experiments and their results are presented in Section 7, followed by concluding remarks in Section 8.

2. SYSTME ARCHITECTURE

In this section, we will first review a typical personal video authoring process, then introduce the architecture of the proposed system, LazyCut, and finally present the four primary and unique characteristics of the system.

A typical manual video authoring process may be divided into the following three major steps, as illustrated by Figure 1.

1. **Raw Media Data Acquisition**: Raw media data are acquired through recoding devices, for example, video captured by camcorders and photos captured by digital cameras or mobile phones, or through downloading from a media server or the Internet, and then imported into a computer system (encoding may be required). For an automatic or semi-automatic editing system, media content analysis may also be embedded in this step.

2. **Content Composition/Authoring**: In this step, editors select appropriate video clips and/or photos from the personal media library, and then put them onto a timeline to generate a storyline. Captions, credits, transitions and video effects may be also added in this step. This is the very essential step for video authoring, while it is also the most time-consuming and tedious step. As to be discussed in details later, CDT templates are designed for this step to remarkably reduce the workload while at the same time preserving high flexibility.

3. **Representation and Rendering**: This step will decide how to render the results created in step 2. For example, based on the authored result, we may generate a streaming video, burn a DVD, create a website, export to slides, print on an album, send out as an email, or just save the result into a description XML file. As we will discuss later, CRT templates are designed for this step to significantly improve the work efficiency and flexibility.

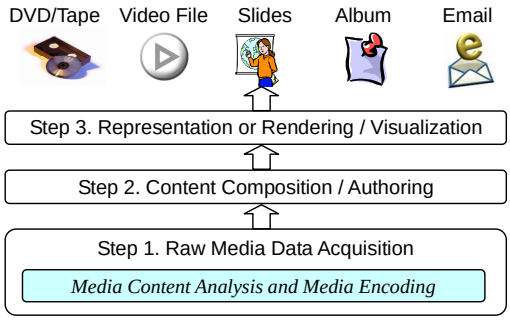


Figure 1. A Typical Process of Video Authoring.

In accordance with this typical process, LazyCut consists of four primary modules, including Media Library Building (MLB), Media Management and Browsing (MMB), Media Authoring (MA), and Media Sharing (MS), as illustrated by Figure 2. MLB provides an open and extensible framework for media content analysis. A set of basic content analysis algorithms, called content (analysis) filters (or filters in brief), are currently supported in LazyCut. Besides, other filters can be easily plugged into the system. Based on these filters, a set of metadata are extracted to enable efficient browsing of the media library and also serve as the basis of content-aware template-based video authoring and sharing.

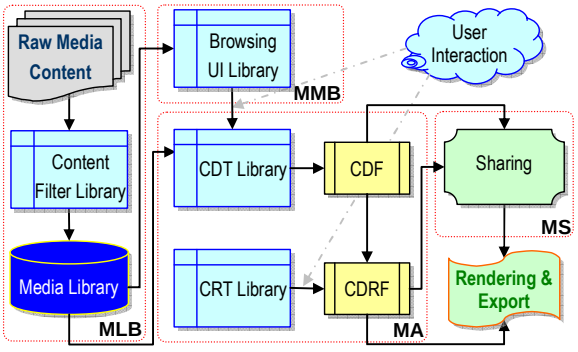


Figure 2. Architecture of LazyCut.

Based on the metadata obtained from MLB module, MMB fosters an open media content browsing and visualization framework. Besides the currently-supported thumbnail and temporal structure based media browsing interface (similar to the ones in [18]), other metadata based visualization and browsing tools can be easily adopted into the system. Efficient browsing also facilitates high-quality media authoring, as it helps users to effectively find appropriate video segments and photo collections in the media

library. Since this module is not the focus of this paper, it will not be presented in detail.

MA is based on a two-layer template scheme which enables semi-automatic, fast, flexible and personalized video authoring. MS is a P2P network based sharing solution built on top of template-based video authoring. When authoring, a template UI (user interface) engine will render a CDT (Content Description Template) template (defined by a XML file) into a timeline to input the corresponding source media content (refer to Figure 9). Then the system will generate a resultant XML file (Content Description File, or CDF file in brief) based on the CDT template and the input data. And finally, a composition engine combines the CDF and a CRT (Content Representation Template) file (also defined by XML) to generate and render the final result, i.e., a CDRF file (Content Description and Representation File, also in XML format), which could be directly used to create a DVD, a website, a video file, etc. In addition, one can also extract CDT from CDF file, or both CDT and CRT from CDRF file, as well as replace CDT and CRT in CDF and CDRF files. The relationships of CDT, CDF, CRT and CDRF files are illustrated in Figure 3.

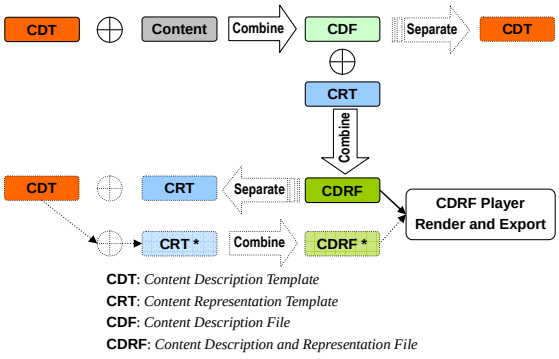


Figure 3. The Relationships of CDT, CRT, CDF and CDRF.

Our system has the following unique characteristics compared with existing video authoring tools and systems.

First, LazyCut is based on media content analysis. In particular, the automatic or semi-automatic editing methods specified in CDTs fully utilize the metadata extracted by content filters. These editing methods significantly speed up the authoring process. Examples of this type of methods can be found in [9], [10], [14] and [21] (to be addressed in details in Section 4.3).

Second, LazyCut is an open and extensible system for video authoring and sharing, which supports dynamic component updating, including adding new content analysis filters, browsing interfaces, editing methods, CDTs and CRTs. This means that besides authoring videos and designing new CDTs/CRTs, advanced users are able to develop their own automatic or semi-automatic editing methods (together with possibly required content filters and supported browsing interfaces), and then plug them back into the system. SDK (Software Development Kit) for developing these new functionalities are available as an advanced feature of LazyCut. These new updates can also be shared with other LazyCut users. With this feature, CDT designers have more choices to select editing methods, and general users may have more choices to use appropriate templates, as well as friendly browsing interfaces. More about system updating is discussed in details in Section 6.

Third, the use of a two-layer editing template scheme, i.e., CDT and CRT, makes video authoring very flexible. In fact, separating CDT and CRT enables the users to obtain a variety of authoring results with only one-round users' input and interaction. More details are presented in Section 4 and 5.

Fourth, XML format is used to describe the results in each step, which facilitates fast and convenient content sharing, as discussed in details later.

3. MEDIA CONTENT ANALYSIS

Video, image and music content analysis is the basis of LazyCut (it is also the underlying reason that we can be "lazy" when doing authoring). LazyCut, as aforementioned, is an open and extensible framework which future new content algorithms can be easily integrated into. More content analysis tools mean that we are able to obtain more compelling authoring results with less effort.

3.1 Open Framework for Content Analysis

In LazyCut, media content analysis algorithms are regarded as content filters (working in a manner similar to DirectShow filters [14]). For example, a shot detection filter is used for shot detection, and a histogram extraction filter is applied for color histogram. There are two types of content filters: one is *online filters*, and the other one is *offline filters*. Online filters require parsing the entire video or audio stream, or photo files, to extract the corresponding basic metadata from the stream. While offline filters extract higher level metadata directly from these basic metadata without parsing the original media data. Examples of online filters include shot detection, timestamp extraction, and histogram extraction filters. Offline filters include scene detection, quality assessment filters, etc.

XML is used to store all extracted metadata. If a new metadata needs be extracted, we only need to write a new filter based on a filter template (a base C++ class) and then plug it into the system. In particular, advanced users can inherit a predefined online or offline base class, and then implement the real metadata extraction functions.

3.2 Content Filters Currently Supported

This section briefly introduces the media content analysis filters currently supported, for video, photo and music, as illustrated in Figure 4.

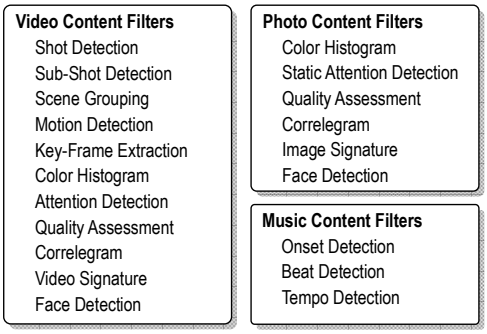


Figure 4. Content Filters Currently Supported in LazyCut.

For video, a shot detection filter with several feature extraction filters is included, such as key-frame selection filter, quantized

color histogram filter [9], ordinal measure based signature filter [8], correlegram filter, face detection filter, attention detection filter, and camera/object motion filter [11]. For photos, we support the same set of filters as videos except shot detection, key-frame extraction and motion detection filters [12]. For music, onset, beat, tempo detection filters are supported [9].

As the detailed content analysis algorithms are not the focus of this paper, we only list these filters with corresponding references here. It should be emphasized here that content filters (to be exact, the metadata extracted by these filters) will be applied by the editing methods, which are the sub-elements of CDTs to be introduced in Section 4. In addition, these metadata also facilitate users to find appropriate raw media segments and then feed them into pre-selected CDTs.

4. CONTENT DESCRIPTION TEMPLATE

As aforementioned, there are many templates based text authoring systems available, such as Microsoft Office templates. Content Description Template plays a similar role but in video authoring. Our goal is to provide a uniform template schema for non-professional video authoring, so that LazyCut users can efficiently and easily modify existing templates, design new ones, and share their templates with others.

CDT is described in XML. The basic unit of CDT is "MSeg" (also called "Slot" when it is rendered – refer to Figure 9), which stands for "Media Segment". MSeg could be a chapter, a scene or a shot, or whatever temporal segment of a video. For a specific template, MSegs may be arranged hierarchically. All MSegs share the same definition and structure. The default settings for a child MSeg are inherited from its parent MSeg, while a child MSeg can have its own settings which have higher priority. A typical hierarchical structure could be "Chapter – Scene", which is similar to a general DVD content menu. In this paper, we will use this structure to present our idea.

A template should at least contain one chapter (MSeg). A Chapter may contain several Scenes (also MSegs), while a Scene can contain one or more smaller scenes, and so on.

4.1 CDT Samples

A good example of a real CDT template is similar to the comprehensive shot list for a typical birthday party proposed by Jan Ozer [15], which includes *establishing*, *pre-party*, *guests arriving*, *meeting and greeting*, *environment*, *lighting candles*, *singing happy birthday*, *eating cake*, *giving and opening gifts*, etc. Figure 5 shows the structure of a sample CDT template based on Ozer's shot list (some shots are removed or merged into one scene or chapter). It contains six chapters, including one leader chapter (*Introduction*), four body chapters (*Guests arriving and greeting*, *The party*, *Guests leaving and giving favors*, and *Final words of the birthday child*), and one tail chapter. And the second body chapter contains three scenes.

Figure 6 is the XML description of MSeg "Leader" in the above sample. It shows that the source data of this MSeg is a collection of photographs, the final duration of this segment is 60 seconds, the incidental music is "abc.mp3", and the automatic editing method is "Photo2Video". Figure 7 shows the XML description of MSeg "Chapter 2" who contains three child MSegs. XML syntax of CDT templates will be introduced in next sub-section.

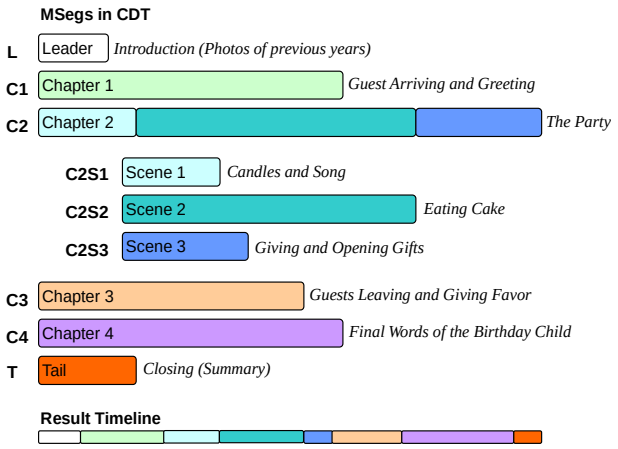


Figure 5. Temporal Structure of a CDT Sample.

```
<MSeg level="1" mtype="Photo">
  <Title>Old Pictures</Title>
  <Duration fixed="false">60</Duration>
  <Method name=Photo2Video ver=1.0 clsid="{BBBBBBBB-2dC1-4081-9BdB-20D7030234EF}">
    <param name="ChannelNum" value="1" />
    <param name="Style" value="Romantic Sepia" />
    <param name="SyncMusic" value="True" />
  </Method>
  <Music><clip src="abc.mp3" start="0" stop="120" mstart="0" /></Music>
</MSeg>
```

Figure 6. XML Description of a Sample MSeg.

```
<MSeg level="1" mtype="Phodeo">
  <Title>The Party</Title>
  <Duration fixed="false">600</Duration>
  <Method name="AVE" ver="1.0" clsid="{AAAAAAAA-2dC1-4081-9BdB-20D7030234EF}">
    <param name="ChannelNum" value="1" />
    <param name="Style" value="Music Video" />
    <param name="SyncMusic" value="true" />
  </Method>
  <MSeg level="2" mtype="Video">
    <Title>Candles and Song</Title>
    <Duration fixed="false">240</Duration>
    <Method name="AVE" ver="1.0" clsid="{AAAAAAAA-2dC1-4081-9BdB-20D7030234EF}">
      <param name="ChannelNum" value="1" />
      <param name="Style" value="Highlight" />
      <param name="SyncMusic" value="true" />
    </Method>
    <Music>
      <clip src="123.mp3" start="0" stop="201" mstart="0" />
      <clip src="456.mp3" start="201" stop="240" mstart="0" />
    </Music>
  </MSeg>
  <MSeg level="2" mtype="Video">
    <Title>Eating Cake</Title>
    <Duration fixed="true">180</Duration>
    <Music><clip src="123.mp3" start="1" stop="180" mstart="0"/></Music>
  </MSeg>
  <MSeg level="2" mtype="Video">
    <Title>Gifts</Title>
    <Duration fixed="false">180</Duration>
    <Music><clip src="abc.mp3" start="0" stop="180" mstart="0" /></Music>
  </MSeg>
</MSeg>
```

Figure 7. XML Description of an MSeg with 3 Sub-MSegs.

4.2 CDT XML Syntax

In this section, we describe the primary elements and the corresponding syntax of CDT templates. Typically a CDT file contains one root element which includes a sub-element called "CDTInfo", as well as a series of "flat" or hierarchical MSegs. CDTInfo provides the basic information of the CDT, including five basic sub-elements, as listed in Table 1.

Table 1. Sub-Elements of "CDTInfo".

Name	Description
title	The title/name of the CDT template.
author	The author of this template.
email	The email of the template author.
url	The URL of the relevant website.
Description	Description of the template.

As aforementioned, MSeg is the basic element of a CDT, and all MSegs share the same definition and structure. MSeg has two primary attributes and four sub-elements, as listed in Table 2 and Table 3, respectively.

Table 2. Attributes of Element "MSeg".

Name	Description
level	The structure level. The first level is "1". MSeg may contain multiple child MSegs, the level of a child MSeg is the level of its parent MSeg plus 1.
mtype	Specify media type of the source data. May be "Video", "Photo" or "Phodeo" (stands for Photo and Video).

Table 3. Sub-Elements of "MSeg".

Name	Description
Title	The title of the MSeg, e.g., the caption of a chapter or a scene.
Duration	The desired duration of this MSeg in the output video. It has only one attribute called "fixed", which specifies whether the duration is fixed, or can be adjusted by the algorithm (<*Method*>).
Music	The incidental music for this MSeg. It is an optional element. The sub-elements of <*Music*> is <*clip*>, which is defined the same as DirectShow DES [14] (*src* is the URL or path of the original raw videos/photos; *start*/*stop* means the start/stop time in the output video segments, while *mstart*/*mstop* means the start/stop time of the original input source video clips).
Method	The method/algorithm that will be used to compose the output timeline of the corresponding slot from the raw media data specified by the users. It has three basic attributes as listed in Table 4, as well as a set of method-dependant parameters (refer to Section 4.3).

Table 4 shows the attributes for the most essential sub-element of "MSeg", i.e., "Method", which stands for the editing method that will be applied to the raw media content that is fed into this slot.

Table 4. Attributes of Element "Method".

Name	Description
name	The name of the editing method, e.g., "AVE", which stands for the method of Automatic Video Editing [9].
ver	The version of the method. This attribute enables users to apply different versions of the same editing method at the same time (within one CDT).
clsid	The GUID (Globally Unique Identifier) of the method. GUID is the unique identification of a certain method.

4.3 Editing Methods

Editing method is the crucial part of a CDT, in which the metadata extracted by content filters are applied. Below is a set of existing methods (some are available in literature or commercial products) we may use. LazyCut users or commercial organizations can develop more editing methods and integrate into the system as plug-ins.

***Muvee autoProducer and Microsoft Movie Maker*:** autoProducer [14] provides an automatic home video editing method with a number of fancy editing styles. Similar functionality called AutoMovie is supported in Movie Maker [13]. Though details of the methods are unavailable to public, these editing methods can be adopted into LazyCut.

***AVE*:** AVE is an optimization-based automatic home video editing method proposed in [9]. It automatically selects suitable or desirable highlight segments from a set of raw home video segments and aligns them with a given piece of incidental music to create an edited video segment based on the contents of the video and music. Four typical editing styles, including *Music Video*, *Highlights*, *Old Movie*, and *Day by Day*, are supported.

***AVE* (CMU)**: It is a method that analyzes a home video and edits it automatically into a condensed and interesting mini-movie with some special effects learned from expert human editors [21].

***Photo2Video*:** Photo2Video, proposed in [10], is a method to automatically convert a photographic series into a motion video by detecting attention areas in the photos and accordingly simulating camera motions on them. The input data of this method are a collection of photographs.

***Full Manual Editing*:** This method supports an manual editing user interface. Users can select and cut appropriate video segments and then drag onto the timeline, as well as adding transitions, effects, captions and incidental music manually. It works in a manner like a traditional video editing system.

Depending on the methods used, there will be different set of editing parameters or preferences. For example, for AVE, the typical parameters are:

***ChannelNum*:** The number of channels that raw media data will be fed in (by users). Typically the number is set to "1", while "2" or a higher value might be applied to generate fancier results. For example, two source video streams may be combined into one stream by "patching" two frames (one from each video) into one frame, as shown by Figure 8.

***Style*:** The editing style [9][14]. It defines the editing preference.

***SyncMusic*:** Whether to sync the beats of the music with shot transitions or not. It is set to "true" or "false".

Figure 8. Sample of Multiple Channel Inputs and Result.

4.4 CDT UI Engine

In LazyCut, a UI engine will parse the user selected CDT (XML file), and then construct an interface to get users' inputs (to be exact, the raw video segments or photos). As CDT is defined by well-structured XML, a uniform UI (user interface) engine is designed. We will use the CDT for birthday video introduced in Section 4.1 as an example to better illustrate how the UI engine works.

Firstly, the UI engine parses the hierarchical structure of the CDT, and draws a series of corresponding "lattices" representing the MSegs (such as L, C1, C2, C2S1, etc.) in the CDT as a timeline (Figure 9). The titles of the slots (i.e. MSegs) will be displayed in the corresponding lattices, and the details (title, duration, method, etc.) of a certain slot will be displayed in a window when the corresponding slot is clicked or got focused. The users are able to adjust the parameters (say, the duration, caption, or editing method) for a certain MSeg in the detail window. Adding, deleting, copying, pasting and moving a slot are also supported in a manner similar to typical editing software.

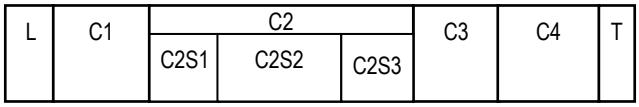


Figure 9. Interface Rendered by CDT UI Engine.

During the process of authoring, users are able to feed raw video segments or photo collection into a specific slot by directly dragging the content from the media library (refer to Figure 12) onto the corresponding slot. Then the UI engine will respond to this action and display the thumbnails of the content in the slot. For a certain slot that requires two or more input sequences, it will be separated into multiple parallel "lattices". For slots where manual editing is specified, a new interface similar to typical manual editing tools will popup to facilitate users' manual editing process.

4.5 CDT With Source Content (CDF)

After inputting raw media content into a certain CDT template, the resultant file, named Content Description File (CDF), is in the same form as the CDT file, except that we add a sub-element called "CDFinfo" into the root element, and a sub-element called "Content" into element "MSeg". Figure 10 shows an example of "CDFInfo" and an example of a certain MSeg (showed in Figure 7) with source content. It shows that two source video clips ("aaa.mpg" and "bbb.mpg") are put into the MSeg and the total duration of the source video is 810.5 seconds. These two clips will be passed to editing method "AVE" and the output video is 240 seconds.

5. CONTENT REPRESENTATION TEMPLATE

The CDF files obtained from previous section are only the descriptions of the authored contents. It is CRT templates that will determine how to represent the contents. As mentioned earlier, CRT templates can be used to render authored contents in many forms, such as DVD, VCD, website, video file, slides, and so on. Each type of CRTs consists of three key components, a XML template definition scheme, a template rendering engine, and a content composition engine. Similar to CDT, in LazyCut, CRT is designed as an independent component. LazyCut provides a uniform programming interface for advanced users to design and develop new types of CRTs. General users also can easily modify existing CRTs and design new CRTs based on the definitions of existing types of CRTs.

```
<CDFInfo>
  <Title>My Daughter's 8 Year's Old Birthday Party </Title>
  <Author>Tom</Author>
  <Affiliation>ABCD</Affiliation>
  <Email>abcd@abcd.com</Email>
  <URL>http://www.abcd.com/ </URL>
  <Description>This CDF XML file is for ...</Description>
</CDFInfo>

<MSeg level="2" mtype="Video">
    <Title>Candles and Song</Title>
    <Duration fixed="false">240</Duration>
    <Method name="AVE" ver="1.0" clsid="{AAAAAAAA-2dC1-4081-9BdB-20D7030234EF}">
      <param name="ChannelNum" value="1" />
      <param name="Style" value="Highlight" />
      <param name="SyncMusic" value="true" />
    </Method>
    <Music>
      <clip src="123.mp3" start="0" stop="201" mstart="0" />
      <clip src="456.mp3" start="201" stop="240" mstart="0" />
    </Music>
  <Content>
    <clip src="aaa.mpg" mstart="0" mstop="600" />
    <clip src="bbb.mpg" mstart="810" mstop="1020.5" />
  </Content>
</MSeg>
```

Figure 10. MSeg Description with Source Media Data.

5.1 Sample CRT Types

Unlike CDT templates which have a uniform definition, CRT definition depends on the type of the CRT. Different type of CRTs will have different definition schemes, rendering engines and composition engines. In this section, we will take DVD, website, and video CRTs as samples to briefly present how to define CRTs, as well as how CRTs work.

For DVD CRT, a XML scheme for describing the look-and-feel, as well as necessary parameters for the output DVD is defined (such as the background image of the menu page and menu button styles). A UI rendering engine is designed to display the look-and-feel of a specific CRT, and a content composition engine is developed to generate the corresponding CDRF file and compose the authored content into DVD format (i.e., to render CDRF file – refer to next sub-section) thus a DVD burning program can burn it directly to DVD disc or a simulated DVD software player can play it in a manner like playing a typical DVD.

For website CRT, a XML scheme for describing the structure, layout, and format of the website is defined, which is similar to HTML language. Similar to DVD CRTs, a UI rendering engine renders the look-and-feel of a specific CRT of this type, and a composition engine generates the CDRF file and the real website according to the specific CDF and CRT.

The type of video CRT includes a XML scheme that defines the transitions, effects and caption format of the output video, a UI rendering engine to display the appearance of a specific video CRT, and a composition engine to generate CDRF file and the final output video.

5.2 CDRF File and CDRF Player

As shown in Figure 3, CDRF file is the combination of CDF and CRT. Figure 11 shows an example CDRF file. Similar to CDF, CDRF is also a description file (including the content and the corresponding representation form). CDRF file can be directly used to generate the final results (DVD, video, website, etc.), and can be directly played back by a CDRF player (to be explained later), and shared with friends or relatives who are connected to the CDRF owner and also have a CDRF player.

CDRF player is a component of LazyCut. It also can work as a standalone application. The CDT parser, editing method plug-ins, CRT composition engines are also linked into CDRF player.

In fact, CDRF player will generate an intermediate XML file similar to the XTL file format in DirectShow Editing Service [16], which is a description of the final results but not the real content. And, according to this description file, the final result can be created, say, a video stream in DVD format, an actual video file, etc. In addition, we can also regard these description files as virtual video files and do not produce any actual video files.

```
<?xml version="1.0" ?>
<CDRF>
  <CDRFInfo>
    <Title>My Daughter's 8 Year's Old Birthday Party</Title>
    <Author>Tom</Author>
    <Affiliation>ABCD</Affiliation>
    <Email>abcd@abcd.com</Email>
    <URL>http://www.abcd.com</URL>
    <Description>This CDRF XML file is for ...</Description>
  </CDRFInfo>
  <CDF>
    ... // the content of the CDF is put here
  </CDF>
  <CRT>
    ... // the content of the CRT is put here
  </CRT>
</CDRF>
```

Figure 11. XML Description of a Sample CDRF File.

6. SYSTEM MAINTANANCE

LazyCut is an open system that allows convenient and flexible extension of new functionalities. For clarity, we divide system extensions into two types. One type of extensions is for general users, who are able to modify and personalize existing templates (both CDTs and CRTs) and create new templates based on existing editing methods and existing CRT types in the system. The other one is for advanced users, who are able to develop new editing methods, new content filters and new types of CRTs.

For the first group of users, though modifying existing CDTs is supported within LazyCut system (refer to Section 4.4), a standalone CDT editor is also provided. Similar to the build-in functionalities supported within LazyCut, the CDT editor supports adding, deleting, copying, pasting, and moving a slot, as well as parameter tuning. The standalone CDT editor shares the same CDT library with the LazyCut system. As for the CRT, the corresponding editor is much more complex since different types of CRTs have different syntaxes. Therefore, for each type of CRTs, a CRT editing interface is required and all interfaces are put into one CRT editor. Similar UI engines and supported editing operations as in CDT are adopted for each CRT editing interface.

For the second group of users, a software development kit (SDK) is available to enable fast implementation of new content filters, editing methods and CRT schemes. Similar to the base classes defined for content filters mentioned in Section 3.1, base classes for editing methods, CRT render engines, and CRT composition engines are also well-defined in the SDK.

New extensions can be compiled and packed into a self-installation package or a package that can be imported through LazyCut or corresponding template editors. Users sharing the media contents can download a package and install or import it into local LazyCut system. After that, the new extensions are available to be used by LazyCut, template editors and CDRF player.

7. RESULTS AND EXPERIMENTS

Next, we will demonstrate the LazyCut framework and the concepts of content-aware template based video authoring, and then present a comparative study on LazyCut and several editing tools in terms of video quality generated and time spent.

Figure 12 shows the main interface of the LazyCut system, consisting of five primary function areas:

Library Tree (A): It shows the currently imported media data (video clips, photos and music), CDTs and CDFs in a tree structure.

Library View (B): It shows the details and content of the selected item in the library tree, such as thumbnails of scenes/shots in a video/photo collection, and look-and-feel of CDT/CRT.

Player (C): A CDRF player that is able to show source pictures, play source videos, and play CDRF files.

Property List (D): It shows the detail information/properties of the selected items in Library View area (B) or a selected slot in a CDT in CDT Timeline area (E).

CDT Timeline (E): It renders the timeline of the selected CDT.

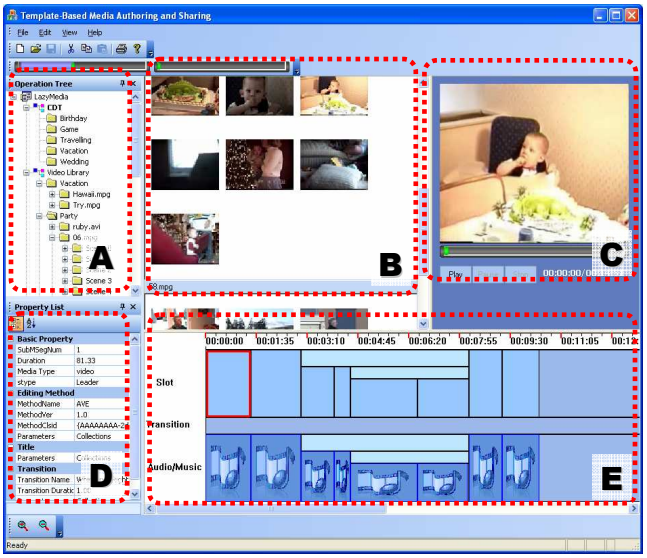


Figure 12. Main UI of LazyCut.

7.1 A Typical Authoring Process

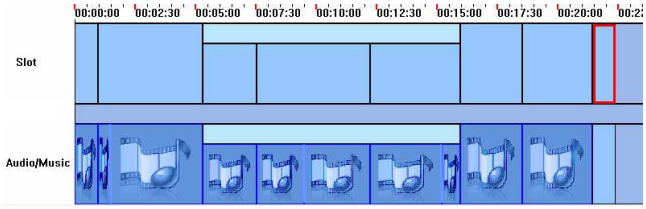
To demonstrate how LazyCut works, we will edit a 10-minute birthday video based on a 2-hour raw home video and a number of digital photos. Suppose we have imported all raw video footages, photos and a set of necessary incidental music into LazyCut. The following features will be demonstrated:

1. Download and install CDT package.
2. Modify selected CDT on the rendered timeline.
3. Feed raw media data into CDT slots.
4. Share CDRF file.
5. Export edited result, and render the CDRF using another CRT.

Firstly we check whether there is any suitable CDT template in the CDT template library. There are several built-in CDTs, but we try to search for a new one over the Internet (simulated by an internal website currently). While a desired CDT package, titled *Elegant Movie* is found on the web, we download the package and imported it into LazyCut. The package includes an offline content filter (a new measure for selecting informative segments), a new editing method (AVE Version 2, which is an automatic editing method similar to AVE proposed in [9]), and a set of CDTs based on the new and existing editing methods. We choose a specific CDT called *Elegant Child Birthday Party*, and then it is rendered in area E (Figure 13(a)). The temporal structure of this CDT is the same as the example we showed in Section 4.1, which contains one leader chapter, three body chapters and one tail. And body chapter 2 has three scenes.

Next, after right clicking on the lattice representing the tail chapter and on the popup window, we change the duration of tail chapter from 1 minute into 2 minutes, and the editing style from "*Default*" into "*Slow Motion*". The modified CDT can be saved (or saved as a separate CDT) in the CDT library for later use.

Then we drag related source video clips and photo collections from the library view area into every slot. For example, photos of previous years are put into leader slot, and clips about candles and birthday song are dragged into scene 1 of the second body chapter. We also drag certain music pieces from music library into the music track of the CDT timeline to replace the default music specified by the original CDT. Similar browsing and searching functionalities as the ones in [18][19] are supported in LazyCut, which facilitate users to efficiently find appropriate source media data. Figure 13(b) shows the CDT after source content are fed in.



(a) Empty CDT.

(b) CDT with source media content.

Figure 13. CDT Timeline of *Elegant Child Birthday Party*.

Next step is selecting a CRT type, as well as a specific CRT and then rendering the final result. In this process, a video CRT is selected, and then we are able to generate the playable CDRF file by selecting menu or toolbar "Build". Then the edited timeline is generated. If the CDT requires more metadata than the system currently has, content analysis may be required in this step.

We may share the generated results, i.e., CDRF file, after applying CRT, or we may also share the intermediate description file, i.e., CDF file. To demonstrate this feature, we send the CDRF file to a friend using instant messaging software. When the receiver opens the shared CDRF file using CDRF player, the edited content is played while the necessary raw media data is streamed to the player by a P2P network connection.

The last step is exporting or saving edited result. Firstly we export the above CDRF into an actual video file in MPEG2 format. Next we select another CRT of website type (may also search and download from the Internet), and then export the result as a website (Figure 14). Please note that no video actually are created during this process. When a certain thumbnail is clicked, the CDRF player will locate the corresponding position in the CDRF file and then instantly compose the timeline and then play the virtual video.

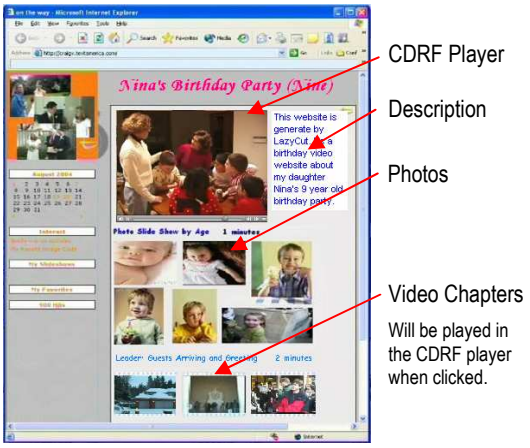


Figure 14. A Website Output of the Edited Result.

7.2 User Study

Processing time for video content analysis in LazyCut is about 1/4 real time for MPEG2 video on a Dell 2.8GHz computer (including decoding time). Photo and music analysis is much faster: One 4M-pixel photo takes about 0.5 seconds, while five-minute music only takes about 4 seconds for analyzing. Final rendering or encoding to a video file from CDRF files is processed in real time, while most of the CPU usage is for video decoding and encoding and less than 5 seconds of it is for timeline composition.

It is difficult to objectively evaluate the quality of the videos created by LazyCut results (with a default video CRT – other CRT types cannot be compared as none of the other systems is able to produce similar results currently). To subjectively evaluate LazyCut results, we compare the edited video using LazyCut with the video produced by connecting randomly selected video segments (sub-shots) and AVE results [9], and the video manually edited by a *nonprofessional* user (using the professional video editing tool, *Adobe Premiere*), who is fond of and familiar with editing home videos but knows nothing about how LazyCut works.

Ten evaluators were invited to do the user study. To obtain more reasonable result, three different sets of randomly edited video (R1, R2 and R3) are applied in this evaluation, while only one set of manually edited video is used due to the intensive labors required for manually editing. Five samples are created for each scheme, and accordingly there are 30 video clips in total. Each set of edited video clips generated from the same media data source are randomly ordered and renamed, thus both the authors and evaluators don't know the producers of the video just by looking at the names of the video clips (so we may say the evaluation is *double blinded*). All users are required to give a satisfaction score ranging from zero to one for each edited video clip, which reflects both "informativeness" and "enjoyabililty" [11]. The scores of the first set of video clips generated by random content selection are fixed to 0.50 thus the users can take them as an example to giving scores for other results.

Detailed evaluation results are listed in Table 5, including the average editing time (*T*, in minutes, excluding the time used for content analysis – as typically it is done when importing the raw media data into LazyCut). The results show that LazyCut has higher satisfaction score than the random, manual and AVE results. The manual and AVE results have comparable evaluation scores, while they are all better than random results. The main reason is that, random editing loses more important segments than AVE (sometimes even low quality segments are selected), and does not align the music with the shot boundaries. AVE has lower flexibility and variations and it cannot preserve the temporal structure very well, while LazyCut supports well-defined temporal structure and a variety of editing methods and styles. From Table 5, we can also see that LazyCut has higher satisfaction score than the manually edited results. Although manual editing could choose relatively more "important" or representative segments from the source media data, it is not easy for an unprofessional user to compose a good storyline, as well as to manually synchronize music beats with the shot boundaries, and align motions with music tempos (which are supported by the automatic editing methods included in the selected CDT). This may be the main reason that LazyCut results are better than manually edited results. A better evaluation would be to compare LazyCut results with professionally edited videos.

Table 5 also shows that the editing time using LazyCut is a little more than AVE, but much less than manual editing. According to the editing process with LazyCut, feeding appropriate media segments into CDT timeline takes most of the time.

Table 5. Subjective Evaluation.

Methods	***R1***	***R2***	***R3***	***Manual***	***AVE***	***LazyCut***
Score	0.50	0.61	0.43	0.73	0.71	0.81
T (minute)	<1	<1	<1	130	<1	6

8. CONCLUSION AND FUTURE WORK

In this paper, we present an end-to-end system, LazyCut, which enables fast, flexible and personalized video authoring and sharing based on content-aware authoring templates. LazyCut is an open and extensible video authoring and sharing system, which supports convenient component updating, including adding new content analysis filters, browsing interfaces, editing methods, and authoring templates. LazyCut is based on media content analysis, in which the metadata extracted by content filters is fully utilized by the automatic or semi-automatic editing methods specified in CDTs. These editing methods significantly speed up the authoring process. The adoption of the two layers of editing templates, i.e., CDT and CRT, separates the video authoring from video representation, and further facilitates flexible and rapid video authoring and sharing.

It should be also mentioned here that CDT templates actually also guide users to capture higher-quality raw media data. With the structure and titles of the CDT slots in mind, users have clearer objectives when shooting appropriate scenes, people, events and so on. In the future, this type of template can even be put into capturing devices to provide a timely guidance to the shooting process.

One limitation of LazyCut is that currently we only support temporal editing though video effects can be regarded as a type of spatial operations. A quick solution to tackle this issue is to integrate spatial editing scheme, such as the one in [3], as an editing method that could be applied on the output results.

Possible improvements and future works on LazyCut includes: (1) to support more well-designed CDTs and CRTs in which the viewpoints of film makers or artists may also be adopted; (2) to support a central media server that enables video publishing; (3) to design a better user interface; (4) to develop more compelling editing methods based on multimedia content analyses.

9. REFERENCES

[1] Adobe Systems Inc. http://www.adobe.com/.

[2] Apple Computer Inc. http://www.apple.com/.

[3] Bennett, E.P., and McMillan, L. Proscenium: A Framework for Spatio-Temporal Video Editing. In *Proceeding of ACM Multimedia 2003*. Berkeley, CA, USA, Nov 2003, 177-184.

[4] Foote, J., Cooper, M., and Girgensohn, A. Creating Music Videos Using Automatic Media Analysis. *ACM Multimedia 2002*.

[5] Girgensohn, A., et al. Home Video Made Easy – Balancing Automation and Use Control. In *Proceedings of INTERACT*, pp. 464-471, 2001.

[6] Girgensohn, A., et al. A Semi-automatic Approach to Home Video Editing. In *Proceedings of ACM Symposium on User Interface Software and Technology*, Vol. 2, pp. 81-89, 2000.

[7] Graham, J., *et al*. The Video Paper Multimedia Playback System. *ACM Multimedia 2003*.

[8] Hua, X.-S., Li, S., and Zhang, H.-J. Video Booklet. In *Proceeding of IEEE Intl Conf on Multimedia and Expo* 2005. Amsterdam, Netherlands, July 2005.

[9] Hua, X.-S., Lu, L., and Zhang, H.-J. AVE – Automated Home Video Editing. In *Proceeding of ACM Multimedia 2003*. Berkeley, CA, USA, Nov 2003, 490-497.

[10] Hua, X.-S., Lu, L., and Zhang, H.-J. Automatically Converting Photographic Series into Video. In *Proceeding of ACM Multimedia 2004*. New York, NY, USA, Oct 2004, 708-715.

[11] Ma, Y.-F., et al. A User Attention Model for Video Summarization. In *Proceeding of ACM Multimedia 2002*. Juan-les-Pins, France, 2002, 533-542.

[12] Ma, Y.-F., et al. Contrast-based Image Attention Analysis by Using Fussy Growing. In *Proceeding of ACM Multimedia 2003*. Berkeley, CA, USA, Nov. 2-8, 2003.

[13] Microsoft Movie Maker. http://www.microsft.com/.

[14] Muvee Technologies. http://www.muvee.com/.

[15] Ozer, Jan. Scripting the Birthday Party Video. *Doceo Publishing*. http://www.doceo.com/bday script.htm.

[16] Pesce, Mark D. Programming Microsoft DirectShow for Digital Video and Television. *Microsoft Press*. Feb 12, 2003.

[17] Sony Pictures Digital Inc. http://mediasoftware.sonypictures.com/

[18] Wang, Y, et al. MyVideos - A system for home video management. In *Proceeding of ACM Multimedia 2002*.

[19] Sun, Y., et al. MyPhotos - A system for home photo management and processing. In *Proceeding of ACM Multimedia 2002*.

[20] Ulead VideoStudio. http://www.ulead.com/.

[21] Yip, S., Leu, E., and Howe, H. The Automatic Video Editor. In *Proceeding of ACM Multimedia 2003*. Berkeley, CA, USA. Nov. 2-8, 2003.