A Feature-Augmented Grammar for Automated Media Production

Freddy Choi, Richard Beales, Jonathan Hearn, Stuart E. Middleton and Matthew Addis
IT Innovation Centre
2 Venture Road, Chilworth Science Park
Southampton SO16 7NP
United Kingdom
{fc,rmb,jrh,sem,mja}@ it-innovation.soton.ac.uk

Christos Mangos
Cinegram Film & TV Productions
43 Gounari street
153 43 Agia Paraskevi
Athens Greece
mngs@mangos.gr

Abstract

The IST Polymnia project aims to create a fully automated system for personalised video generation. Film production involves content creation, selection and composition. Technological advances have automated parts of the process, however video editing continues to be a bottleneck in the production chain as constant human interaction is required in the process.

This paper presents a linguistically motivated solution that uses context-free feature-augmented grammar rules to describe editing tasks, thus video editing can be automated by a novel rule application algorithm. The solution is media and application independent. Performance analysis results have suggested that the solution is applicable to near real-time applications.

1. Introduction

The IST Polymnia project aims to develop a fully automated film production system that can be configured to produce personalised videos for fun park and museum visitors. The system uses cameras to record the experience of all the visitors. The video streams are analysed and segmented according to content. For each visitor, video clips of their experiences are selected and arranged to produce a personalised film which is recorded onto a DVD. The system must operate in near real-time in order to produce souvenir DVDs in time for delivery to visitors as they leave the venue. This implies automation in every step of the process. This paper focuses on the automatic selection and arrangement of video clips for personalised film production.

1.1. The roller coaster scenario

Cameras are strategically placed on a roller coaster ride to record the visitors and their experiences, i.e. (1) close up of visitor in the roller coaster chariot, (2) embarking/disembarking, (3) sitting on the chariot and , (4,5) experiencing the two main drops.

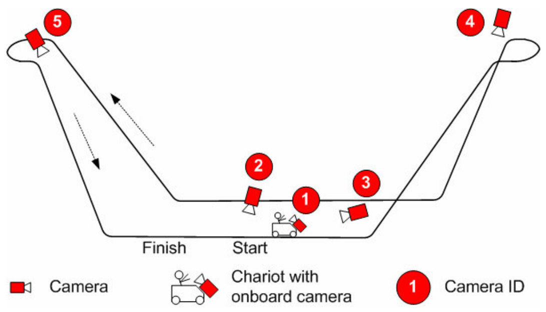


Figure 1. The roller coaster filming scenario.

Automatic content analysis is applied to the video streams to detect and identify the different visitors. Sensors are used to detect events such as the location of a roller coaster chariot on the track. The information is added to each video stream as annotation in MPEG-7 format. The MPEG-7 document is fed to an automatic media editing system that uses the information to select the relevant video segments for each visitor and produces a personalised film of their experience. The film is recorded onto a DVD and made available as a souvenir for each visitor.

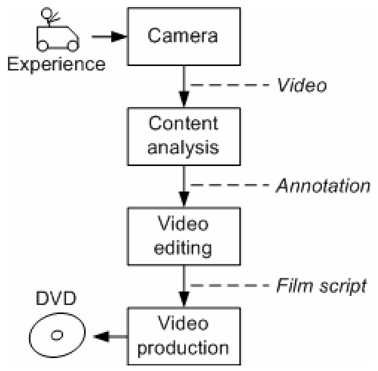


Figure 2. Automatic film production in Polymnia.

1.2. Film production

The placement of cameras, the selection of video clips and how they are arranged to produce a short film are designed and specified by a professional film director. Video clips are edited according to the time of real-world events or other shots. For instance, in the roller coaster scenario, shot 2 (S2) is a 30 seconds long video clip extracted from camera 1 (Cam 1) with start time equal to the time when a visitor was detected in Cam 1 (Event A). Shot 1 (S1) is a 20 seconds long video clip extracted from camera 2 (Cam 2) with end time equal to the start time of S2. This shows how shot generation is event triggered and can involve footage captured both before and after an event.

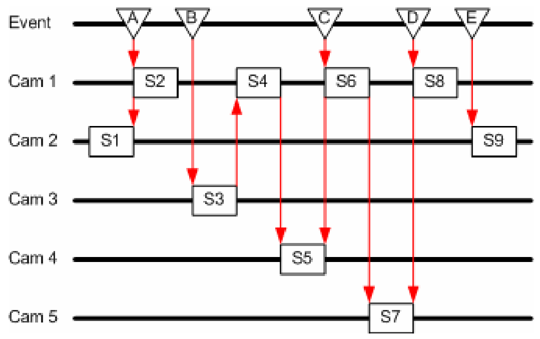


Figure 3. Video clip selection in the roller coaster scenario.

This paper is about the automatic video editing module in Polymnia. The module automates video clip selection and shot arrangement thus enabling the overall system to automate the entire film production process. It takes an automatically generated MPEG-7 document as input. The document contains annotation about the identity, location and time of appearance of each visitor and object in each video stream, e.g. visitor Andrea was detected by Cam 1 at 12:21pm. The film director's ideas are implemented as a set of editing rules that are triggered by annotation. These rules create the shots and combine the shots according to the director's specification to produce a MPEG-7 film script.

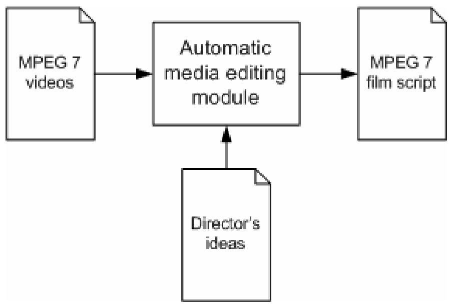


Figure 4. The automatic video editing module.

The automation of video editing implies that the film director only needs to be involved in designing the shooting scenario and the film production plan. Once the director's ideas have been implemented as editing rules, the system is able to automatically produce a personalised film for every visitor.

2. Automated video editing

A simple approach to personalised film production uses a film template with slots for inserting the personal video segments. This is the same as mail merging where the recipient name is changed to personalise a letter. A template cannot cope with exceptions, e.g. it cannot produce a film if any of the required segments is missing. This work introduces a rule-based approach that offers a level of flexibility which is crucial in any real-world system.

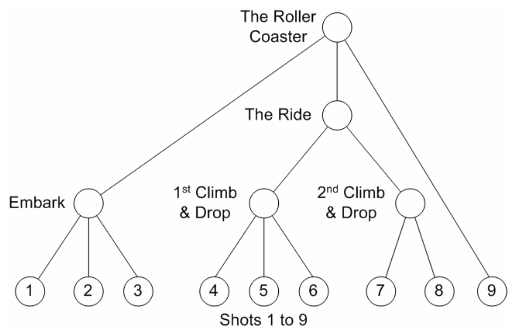


Figure 5. An aggregation approach to media editing.

For the roller coaster scenario, three shots (S1, S2, S3) are combined to produce a film about a visitor embarking and starting the ride. Two short films of the climb and drop sequences are produced by concatenating shots S4, S5, S6 and shots S7, S8. These are combined to make a film about the ride. Finally, a film about the entire roller coaster experience video is created by concatenating the short films about embarking, the ride and disembarking.

This rule-driven aggregation approach to video editing makes it easy to offer alternatives and make the video editing system more robust against unexpected input. For instance, a recursive rule can be defined to make the 'embark' short film use any combination of the three shots in the film to cope with cases where the visitor was obscured in one of the videos and thus not detected by the analysis module. Although the same can be achieved using fixed templates, one will have to create templates for all possible combinations of situations, thus making the system hard to maintain and cumbersome to manage.

2.1. What is video editing?

Given a collection of videos, video editing is about the selection and arrangement of video clips according to a set of artistic preferences to produce a coherent film. The selection of video clips involves watching a video to analyse the content such as what appears in the video and at what time. The selection of video clips is based on this semantic information, e.g. the roller coaster film is made up of shots about a particular visitor sitting on a particular roller coaster chariot.

The arrangement of shots is also based on the semantic information such as camera location and recording time. More specifically, the arrangement of shots is based on the relationships between different video clips. For instance, in the roller coaster scenario, the shots are defined and arranged according to time constraints between the shots to ensure that only video clips from the same ride are used to produce a short film, i.e. the visitor can go on the ride twice to produce two films, one for each ride, as opposed to a single film that uses video clips from two different rides.

In abstract terms, the selection and arrangement of video clips is entirely feature driven. A feature, in this context, can refer to any property about a video segment, e.g. the start/end time of a video, the name of the visitor/camera or the lighting condition. A feature value is anything that is comparable, i.e. name, time and number. Selection is therefore automated by pattern matching, e.g. the existence of an event with label ='A' means the creation of a shot with label='S2', end time='start time of event A' and start time='end time – 20 seconds'. Arrangement is similar except the pattern matching condition is more complex, e.g. the existence of three shots with labels 'S1', 'S2' and 'S3' implies the creation of a short film with label 'EMBARK' if S2 starts when S1 ends and S3 starts within 30 seconds of S2 ending.

2.2. A linguistically motivated solution

The challenge presented here is similar to syntactic parsing and sentence generation in computational linguistics [1] where the grammatical structure of a sentence is described by context-free rules that define valid word and phrase combinations according to their part-of-speech tags, e.g. verb, noun, article, verb phrase, noun phrase. A grammatically correct sentence is generated by applying the rules to a set of words to produce phrases which are aggregated to produce a complete sentence. This is the same as the video editing problem except text generation uses non-overlapping complete words to produce a sentence whereas video editing can use overlapping video clips in the final result.

The solution presented in this paper was inspired by previous work in computational linguistics [1] and expert systems [7,9]. The primary distinctions are, firstly, the proposed grammar is designed for video clip aggregation as opposed to word ordering as in text generation. A video clip can be parts of multiple aggregated short films. Secondly, the grammar allows the use of mutually dependent conditions as opposed to independent conditions as in most expert systems. Two conditions can define tests that reference the properties of elements that are matched by the other condition.

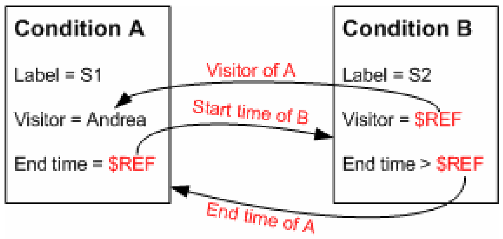


Figure 6. Describing a combination of two elements with mutually dependent conditions.

2.3. Context-free feature-augmented grammar

The proposed grammar operates on elements. The grammar describes the creation of elements based on other elements. An element is a collection of properties where a property is a key-value pair. The key is a textual name and a value can be anything that is comparable, e.g. number, text and time. For instance, a video clip is an element with properties such as camera name, start/end time, duration, object type and object name.

```
<elements>
    <element>
        <property key="type" value="event"/>
        <property key="camera" value="Cam1"/>
        <property key="objectType" value="Visitor"/>
        <property key="object" value="Andrea"/>
        <property key="startTime" value="12:10:36"/>
        <property key="endTime" value="12:10:39"/>
        <property key="duration" value="00:00:03"/>
    </element>

    <element>
        <property key="type" value="event"/>
        <property key="camera" value="Cam3"/>
        <property key="objectType" value="Chariot"/>
        <property key="object" value="Chariot_A"/>
        <property key="startTime" value="12:12:09"/>
        <property key="endTime" value="12:12:14"/>
        <property key="duration" value="00:00:05"/>
    </element>

    <element>
        <property key="type" value="event"/>
        <property key="camera" value="Cam4"/>
        <property key="objectType" value="Chariot"/>
        <property key="object" value="Chariot_A"/>
        <property key="startTime" value="12:12:14"/>
        <property key="endTime" value="12:12:20"/>
        <property key="duration" value="00:00:06"/>
    </element>
</elements>
```

Figure 7. Example video elements.

A grammar is a set of rules where a rule defines the creation of an element based on the existence of other elements. A rule has two parts, an element definition and a list of conditions. The former defines the properties of an element that is created by the rule. The latter defines the set of elements that must exist in order to activate the rule. For instance, the rule for creating a short film about a visitor embarking on the roller coaster has three conditions describing the three shots required to produce the film.

A condition describes a group of elements by their properties. A condition consists of a collection of property tests where a test is a triplet of property name, test operator and test value, e.g. (label, equals, S1). An element satisfies a condition if it satisfies all the tests. A condition can optionally define a set of references to the property values of matched elements which may be used by other conditions in their test statements or in the element definition.

The following example presents a grammar with four rules. The first rule defines an 'Embark' shot based on the detection of a visitor in Cam 1. The shot uses video from Cam 2 and starts 3 seconds before the detection. The bind statements define references for use in the element definition. The second rule defines a 'Start' shot with start time equal to the end time of the 'Embark' shot and end time defined by the detection of the roller coaster chariot in Cam 3. The third rule define the 'Drop' shot based on the end time of the 'Climb' shot and the detection of the roller coaster chariot dropping in Cam 4. The last rule concatenates the three shots to produce a film.

```
<grammar>
    <rule>
        <define key="type" value="shot"/>
        <define key="name" value="Embark"/>
        <define key="camera" value="Cam2"/>
        <define key="startTime" value="$ST-3s"/>
        <define key="endTime" value="$ET"/>
        <define key="duration" value="$D+3s"/>
        <condition>
            <test key="type" op="eq" value="event"/>
            <test key="camera" op="eq" value="Cam1"/>
            <bind key="startTime" ref="$ST"/>
            <bind key="endTime" ref="$ET"/>
            <bind key="duration" ref="$D"/>
        </condition>
    </rule>

    <rule>
        <define key="type" value="shot"/>
        <define key="name" value="Start"/>
        <define key="camera" value="Cam3"/>
        <define key="startTime" value="$ET2"/>
        <define key="endTime" value="$ET1"/>
        <define key="duration" value="$ET2-$ET1"/>
        <condition>
            <test key="type" op="eq" value="event"/>
            <test key="camera" op="eq" value="Cam3"/>
            <bind key="endTime" ref="$ET1"/>
        </condition>
        <condition>
            <test key="type" op="eq" value="shot"/>
            <test key="name" op="eq" value="Embark"/>
            <bind key="endTime" ref="$ET2"/>
        </condition>
    </rule>

    <rule>
        <define key="type" value="shot"/>
        <define key="name" value="Drop"/>
        <define key="camera" value="Cam4"/>
        <define key="startTime" value="$ET2"/>
        <define key="endTime" value="$ET1"/>
        <define key="duration" value="$ET2-$ET1"/>
        <condition>
            <test key="type" op="eq" value="event"/>
            <test key="camera" op="eq" value="Cam4"/>
            <bind key="endTime" ref="$ET1"/>
        </condition>
        <condition>
            <test key="type" op="eq" value="shot"/>
            <test key="name" op="eq" value="Start"/>
            <bind key="endTime" ref="$ET2"/>
        </condition>
    </rule>

    <rule>
        <define key="type" value="film"/>
        <define key="name" value="RollerCoaster"/>
        <condition>
            <test key="type" op="eq" value="shot"/>
            <test key="name" op="eq" value="Embark"/>
        </condition>
        <condition>
            <test key="type" op="eq" value="shot"/>
            <test key="name" op="eq" value="Start"/>
        </condition>
        <condition>
            <test key="type" op="eq" value="shot"/>
            <test key="name" op="eq" value="Drop"/>
        </condition>
    </rule>
</grammar>
```

Figure 8. Example video editing rules.

To summarise, the proposed grammar is based on the manipulation of elements where an element is simply a bundle of properties. A property is described by a name and a constant value. New elements are created by a rule if all its conditions are satisfied by a set of elements. A new element is created for every combination of matching elements. Property references can be defined in a condition to facilitate the expression of cross-referencing conditions and property inheritance in the element definition. Values in the element definition and tests can be constants, variables or a combination of the two (e.g. addition and subtraction).

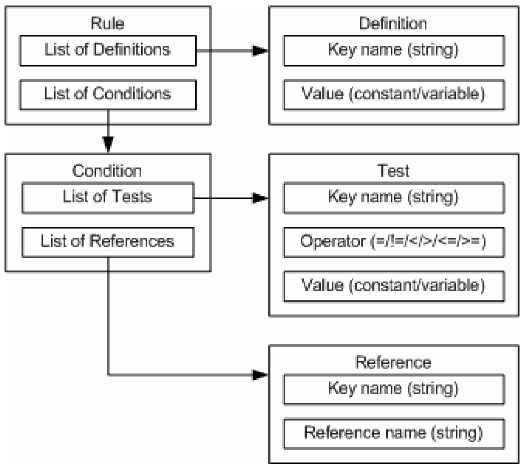


Figure 9. A feature-augmented grammar for video editing.

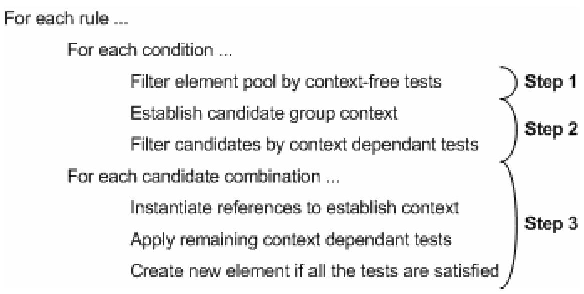
2.4. Rule application algorithm

The rule application algorithm has an element pool which is initialised with elements describing the observed events, e.g. visitor Andrea was detected by Cam 1. Every rule is applied to the pool to create a new set of elements which are added back to the pool, e.g. the shots, the short films and finally the complete film. The process repeats until either no new elements are created or the process has repeated too many times, i.e. all the possible films have been created or an erroneous recursive rule is continuously creating elements that trigger the rule again.

Rule application consists of three distinct steps. First, candidate elements for each of the rule conditions are discovered by filtering the element pool using only the tests that refer to constant values, e.g. find all the video clips about Andrea.

Second, the candidate set for each condition is further reduced by examining the candidate group context and applying tests that only refer to group variables, e.g. given a set of candidates, select the longest video clip.

Finally, every combination of candidates for each condition is examined to instantiate all the references and apply all the remaining tests to identify the combinations that satisfy all the conditions. A complete match implies the creation of a new element based on the element definition using the reference values for the particular candidate combination, e.g. given a particular combination of shots 1, 2 and 3, apply tests to ensure that each shot starts just after the previous shot; if all the tests are satisfied, create the 'embark' short film with property 'visitor' equal to that of shot 2.


```
For each rule ...
    For each condition ...
        Filter element pool by context-free tests            } Step 1
        Establish candidate group context                    }
        Filter candidates by context dependant tests         } Step 2
    For each candidate combination ...
        Instantiate references to establish context          }
        Apply remaining context dependant tests              } Step 3
        Create new element if all the tests are satisfied    }
```
Figure 10. A naïve algorithm for rule application.

To summarise, the rule application algorithm uses set refinement to find all possible combinations of elements that satisfy the rule conditions in three steps. The first step applies the context-free tests to eliminate clearly non-matching elements. The second step analyses the candidates for each condition to establish the range of property values in each group, thus enabling the application of context-dependant tests to further reduce the candidate set. The final step applies the remaining tests to each candidate combination to identify complete matches which result in the creation of new elements. The application of rules and the creation of elements are recorded by the algorithm to ensure that a rule is only applied once to the same set of elements thus creating only one new element.

2.5. Implementation

The algorithm adopts a naïve approach to rule application. Its implementation incorporates several optimisation methods that make the algorithm practically applicable. Every rule, rule condition and element is associated with a unique integer identifier. This facilitates the use of a memory and computationally efficient hash tree structure for caching the analysis results from step one and three of the algorithm thus reducing redundant computation.

One instance of the cache structure is used to record the result of applying the context-free tests of a condition to each element, i.e. step one of the algorithm. The information is reused in subsequent applications of a rule to bypass the first step for known elements. Another instance of the cache structure is used for recording combinations of elements that satisfy a rule, i.e. the creation of a new element. The information is reused in subsequent applications of a rule to bypass the third step of the algorithm. The information is also used to ensure that only one new element is created for a combination of elements that satisfy a rule.

3. Evaluation

The proposed solution has been applied to a real problem, personalised video souvenir generation for a fun park. The evaluation procedure assumes that information about the presence of a visitor or object in a video footage is complete and accurate. A grammar was developed to generate personalised films about a roller coaster ride (Figure 1).

The grammar uses five rules to define the five key events (Figure 3). Nine rules are used to define the nine shots (S1 to S9) based on these events or other shots. Five rules are used to concatenate these nine shots into a complete film (Figure 5), thus the grammar has a total of 19 rules.

The scalability of the proposed solution was assessed by applying the grammar to an increasing number of events and measuring the computation time. Every ride on the roller coaster generates five events which trigger the generation of a film. The addition of an event increases computational complexity as it introduces ambiguity in associating events from different rides to a film. A theoretical analysis of the algorithm has revealed that it has a complexity of order $\mathbf{O(n^c)}$ where **n** refers to the number of input elements and **c** is the average number of conditions in each rule.

A quantitative analysis of speed performance used varying number of events from 5 to 120 in steps of 5, representing 1 to 24 rides on the roller coaster. Each ride increases the number of pool elements by 24, thus the maximum number of elements was 576. The test was repeated 35 times to estimate the average computation time for different number of events. All the experiments were carried out on an AMD Athlon XP 1700+ (1.47GHz) desktop PC with 1GB RAM (a maximum of 250MB for the Java virtual machine).

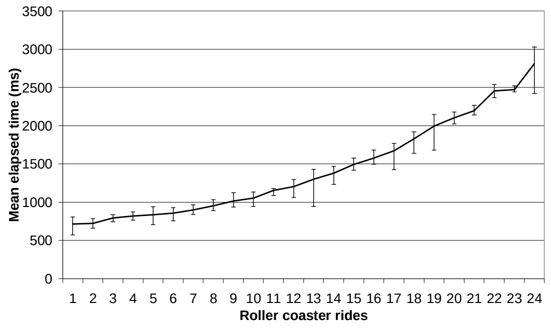


Figure 11. Speed performance analysis.

The test results show that the computation time of the optimised implementation increases exponentially with order $\mathbf{O(e^{0.06n})}$ where **n** refers to the number of events. The 19 grammar rules has on average 1.44 conditions per rule, i.e. **c = 1.44**. The experimental system took less than one second to analyse 40 ambiguous events and automatically select 72 video clips to produce 8 films. In three seconds, the system was able to analyse 120 ambiguous events and select 576 video clips to produce 24 films.

The speed performance analysis results suggest that the automated solution can carry out video editing at a speed that makes it applicable to near real-time applications. For instance, a low-resolution streamed version of a personalised video can be made available to remote families and friends immediately after the visitor has disembarked from the roller coaster. The video can also be presented at a terminal at the venue in much the same way as making recently captured photos available for preview and purchase.

4. Related work

4.1. Video editing description languages

Previous works [5,10] have acknowledged the importance of adopting an abstract representation for media editing. The W3C synchronized multimedia integration language (SMIL) standard [5] is a XML-based description language for interactive multimedia presentations. It defines a set of XML elements for describing the temporal behaviour and the visual layout of a multimedia presentation and also for associating hyperlinks with media objects. SMIL has been created as a media, platform and application independent solution for multimedia authoring. It can be used to describe a fixed film script but it cannot describe alternative strategies such as what to do if a video clip is missing.

Existing languages can describe fixed presentations with predefined behaviour and known actions. Polymnia requires a solution that can automatically adapt a film script according to what video clips are available. The work presented here uses simple rules to describe different parts of a presentation. Multiple rules can be used to create different short films using different video clips. This rule-based approach makes the encoding of alternative editing strategies manageable. The same approach has been used in computational linguistics to facilitate the description of very large complex grammars.

4.2. Automatic media editing

Advances in content creation and storage technologies have created an urgent need for automation in content organisation and editing. Commercial automatic video editing solutions for home users [12,15,17] typically adopt a music video editing approach where shots are selected, organised and combined according to a piece of music. In general, the process involves first selecting a video and manually marking up the interesting and irrelevant parts. A piece of music is then selected. These are analysed by an automatic algorithm to determine the tempo and scene changes. An edited video is created by concatenating the interesting scenes and inserting scene transitions according to the tempo. Video editing packages for professionals and enthusiasts [2,14] tend to offer simple functions for automating mundane tasks such as scene detection and transition insertion thus enabling the user to focus on film production.

Experimental automatic video editing systems are typically developed for specific scenario such as multiparty conversation in meetings [16] and documentary generation [3,4,6,8,11]. The former adopts a simple editing strategy that uses speaker identification and head orientation detection to determine how the video should be edited to show the speaker and the reaction of the listeners. Documentary generation is based on discourse theories that stem from computational linguistics. Existing systems are similar to that for textual story generation. Stock video clips are manually annotated with semantic information to enable an automatic system to generate a tailored video by selecting and concatenating the clips according to the annotation.

Previous work in the area of automatic media editing and automatic video generation is progressing naturally towards the creation of a domain independent solution [3,13,18]. These systems aim to encode common film editing and story generation rules to make them generally applicable. The main challenge is creating the complementary content analysis system that can automatically extract the rich semantic information required to activate the rules.

The work presented in this paper offers a generic platform for encoding and executing media editing rules. It introduces the concept of feature-based media editing which stems from work in computational linguistics. The method has been applied to a wide range of problems in text processing and text generation which have many parallels with media editing.

5. Conclusions

Film production is a labour intensive and expensive process. Even the production of a simple home video involves many hours of manual labour to capture the video clips, review the content, select the highlights, edit the shots, compose the video, insert scene transition effects, review the video and publish the result. The process requires constant human input. Tools have been developed to automate and speed up most of the processes such as video content analysis for object detection and identification, scene change detection for storyboard generation, transition effect insertion for media composition. However, video editing is still a manual process that involves constant human interaction.

The IST Polymnia project aims to create a fully automated film production system. It uses automatic content capture and analysis to acquire videos and annotate the videos with semantic information. The solution presented in this paper offers a rule-based grammar for encoding a film director’s ideas about how a film should be made and a rule-application algorithm for implementing the ideas according to real situations. It generates a personalised film script which describes what video clips are used in what order to produce a personalised film. An automatic video production module fetches the video clips and combines them according to the script. The result is written to a DVD for delivery to the visitor, thus the entire production chain is automated.

Automation does not remove the human element from media production. It enables a human operator to provide all the information about a task which is then executed and completed automatically, thus removing the need for continuous interaction. The proposed solution has been designed for video editing, however it is applicable to any kind of media editing that involves the selection, arrangement and aggregation of content such as music clips, photographs and text. The solution has been evaluated using a real scenario. The speed performance analysis results suggest that the solution is applicable to near real-time applications.

6. Future work

A novel rule development environment is currently being created to offer an intuitive graphical interface for rule authoring, testing and management. The interface has been designed specifically for adaptive video editing. The tool will streamline the rule development cycle thus reducing the cost of configuring, deploying and managing the Polymnia system.

7. Acknowledgements

The POLYMNIA project is funded by the European Commission in the Information Society Technologies (IST) Programme under FP6-2003-IST-2 strategic objective 2.3.2.7 Cross-media content for leisure and entertainment - Project No. IST-2-004357. The partners in Polymnia include ICCS/NTUA, IT Innovation, Telefonica I+D, Hewlett Packard, Joanneum Institute, Giunti Interactive Labs, Cinegram S.A., Università Cattolica del Sacro Cuore Description, and Hellenic Entertainment Park - ALLOU. Full details of the Polymnia project can be found at http://polymnia.pc.unicatt.it/.

8. References

[1] Allen, J. *Natural Language Understanding*, The Benjamin/Cummings Publishing Company, Redwood City, California, 1995.

[2] Apple Computer Incorporated, “iLife’06: iMovie HD”, http://www.apple.com/ilife/imovie, 2006.

[3] Bocconi, S., “Semantic-Aware Automatic Video Editing”, in *Proceedings of MM’04*, New York, USA, 2004.

[4] Bocconi, S. and Nack, F., “VOX POPULI: Automatic Generation of Biased Video Sequences”, in *Proceedings of SRMC’04*, New York, USA, 2004.

[5] Bulterman, D., Grassel, G., Jansen, J., Koivisto, A., Layaïda, N., Michel, T., Mullender, S. and Zucker, D. (editors) “Synchronized Multimedia Integration Language (SMIL 2.1) W3C Recommendation”, The World Wide Web Consortium (W3C), http://www.w3.org/TR/SMIL2, 2005.

[6] Davenport, G. and Murtaugh, M., “ConText: Towards the Evolving Documentary”, in *Proceedings of ACM Multimedia’95*, 1995.

[7] Friedman-Hill, E., “Jess, the Rule Engine for the Java Platform”, Sandia National Laboratories, http://herzberg.ca.sandia.gov/jess, 2001.

[8] Houbart, G., *Viewpoints on Demand: Tailoring the Presentation of Opinions in Video*, PhD Thesis, MIT, 1994.

[9] Jackson, P. *Introduction to Expert Systems*, Addison Wesley, 1999.

[10] Macromedia Incorporated, “Macromedia Director MX 2004 Product Datasheet”, Adobe Systems Incorporated, http://www.macromedia.com/software/director, 2003.

[11] Mateas, M. “Generation of Ideologically-Biased Historical Documentaries”, in *Proceedings of AAAI’00*, 2000.

[12] Muvee, “Muvee autoProducer 5”, http://www.muvee.com, 2006.

[13] Nack, F., *AUTEUR: The Application of Video Semantics and Theme Representation in Automatied Video Editing*, PhD Thesis, Lancaster University, 1998.

[14] Pinnacle Systems Incorporated, “Pinnacle Studio 10.5”, http://www.pinnaclesys.com, 2006.

[15] Roxio, “Roxio VideoWave 7”, http://www.roxio.com, 2006.

[16] Takemae, Y., Otsuka, K. and Yamato, J. “Automatic Video Editing System Using Stereo-Based Head Tracking for Multiparty Conversation”, in *Proceedings of CHI’05*, Oregon, USA, 2005.

[17] Ulead Systems Incorporated, “Ulead VideoStudio 9”, http://www.ulead.com, 2006.

[18] Yip, S., Leu, E. and Howe, H., “The Automatic Video Editor”, in *Proceedings of MM’03*, California, USA, 2003.