Eurographics Workshop on Intelligent Cinematography and Editing (2017)
W. Bares, V. Gandhi, Q. Galvane, and R. Ronfard (Editors)

Using ECPs for Interactive Applications in Virtual Cinematography

H-Y. Wu[1] and T-Y. Li[2] and M. Christie[1]

[1]IRISA/INRIA Rennes Bretagne Atlantique, Rennes, France
[2] National Chengchi University, Taipei, Taiwan

Abstract

This paper introduces an interactive application of our previous work on the Patterns language as creative assistant for editing cameras in 3D virtual environments. Patterns is a set of vocabulary, which was inspired by professional film practice and textbook terminology. The vocabulary allows one to define recurrent stylistic constraints on a sequence of shots, which we term "embedded constraint pattern" (ECP). In our previous work, we proposed a solver that allows us to search for occurrences of ECPs in annotated data, and showed its use in automated analysis of story and emotional elements of film. This work implements a new solver that interactively propose framing compositions from an annotated database of framings that conform to the user-applied ECPs. We envision this work to be incorporated into tools and interfaces for 3D environments in the context of film pre-visualisation, film or digital arts education, video games, and other related applications in film and multimedia.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism–Animation—I.2.10 [Artificial Intelligence]: Vision and Scene Understanding—Video Analysis

1. Introduction

3D virtual environments have become a formidable platform for pre-visualisation of films. Film companies and film schools alike use 3D environments to plan and pre-visualise film cinematography without employing real actors, scenes, and props. The popularity of 3D environments is largely due to its accessibility (often only requiring a computer), and relatively low cost (as compared to employing real scenarios with real actors). One strong topic of interest and continued challenge lies in tapping the creative potential of 3D environments, combined with AI algorithms, to be a creative assistant to the filmmaker by automated generating and proposing various camera and filming styles.

In our previous work, we have addressed the problem of computational properties of style. We first introduced *Patterns*, a set of vocabulary targeted towards describing the computable and analysable elements of film styles using terminology from actual film practice [WC15]. We then proposed embedded constraint patterns (ECPs) as a way to define recurring stylistic constraints over a sequence of shots, and designed a solver for ECPs that allows us to search annotated film data for occurrences of ECPs in order to gain insight into story and emotional elements of film [WC16a] [WC16b]. Automated analysis of actual film data is valuable in terms of learning from real examples and experience. Imagine its potential in a creative context, where a smart assistant could propose from given data, framing compositions for shots in a sequence of a 3D virtual scene that fulfil certain stylistic constraints defined by a number of ECPs, such as in Figure 1, changing the on-screen regions from same to opposite in order to replicate similar stylistic patterns observed from *Pulp Fiction*. Currently, there are limited means of incorporating the combined benefits of annotated film data and actual film knowledge into 3D environments for interactive applications targeted towards the pre-visualisation or film education.

Figure 1: *By changing small recurring constraints of on-screen style, we can create completely different emotional feelings for 3D animated scenes that replicate those observed in real movies, such as these in Pulp Fiction.*

In this paper we present the *Patterns* language as a multipurpose

DOI: 10.2312/wiced.20171076

language for virtual cinematography applications. We position *Patterns* and its accompanying ECP solver as a way to (1) describe complex recurring elements of style in shot sequences using vocabulary from actual film practice, (2) define stylistic constraints over long shot sequences for creativity contexts, and (3) link existing annotated film data to an interactive context to propose framing compositions in 3D environments.

The remaining organisation of the paper is as follows. In the next section we cover the related work. We then present an overview of the Patterns language–the vocabulary and definition of ECPs–and the framing database in which we store annotated film data. Section 6 then presents the constraint propagation and search algorithms for proposing framing solutions from the framing database to a user in an interactive context. We illustrate the contributions in this paper with a number of examples. Finally, we will conclude with a preview of the currently-in-development usage scenarios of *Patterns* and ECPs in interactive creative applications.

2. Related Work

In virtual camera control, film idioms and continuity rules have been popularly adapted to constraint-based systems. [DZ94] was one of the first to propose an idiom-like language for constraint-based cinematography systems that would allow a smooth navigation of the camera through a complex virtual environment like a museum room. Another early example is the DCCL language for planning sequences of camera movements [CAH*96]. Bares [BGL98] introduces a real-time system for camera placement using constraint-based approaches, gathering information about the occurring story events and making decisions on how to best show the action. The system provides a nice solution to real-time context aware scenarios. Similarly, Bares et al. [BTM00] developed a constraint-based approach to framing and camera positioning for virtual cinematography, where the user can specify a number of constraints on the depth, angle, distance, region, occlusion...etc, and the camera will find shots that satisfy the decided constraints. More recently, vocabularies for cinematography have been developed apart from Patterns. Most notably, the Prose Storyboard Language (PSL) [RVB13] was designed based on actual film practice on shot composition, including vocabulary for elements like size, region, or movement. In the implementation in [GCR*13], PSL targets autonomous camera steering and composition, conducting a search for suitable camera positions based on PSL constraints. The first interactive solution was proposed by Lino and Christie [LC15] by using the toric space to calculate smooth camera transitions that fulfilled a series of user defined framing compositions. Galvane et al. [GCLR15] developed the method of virtual camera rails based on this work, which was inspired by pre-constructed rails for cameras to slide on in actual film sets, calculating smooth camera trajectories around standing or moving actors. However, these solutions do not specifically target the problem of recurring stylistic constraints in long sequences of shots, as *Patterns* aims to do.

In terms of applying camera rules or camera style constraints in editing sequences for multiple shots, [GRLC15] uses semi-Markov chains to optimise a number of parameters in camera shot and cut decisions. These decisions are mainly motivated by film practice of evaluating narrative importance (of actors), observing continuity, and creating rhythm in the edits. [MBC15] uses Hidden Markov Models to imitate a director's decision process when deciding when/how to cut between shots, what shot types to use, and which actors to focus on. However, these proposed solutions produce a single edited sequence, and was not designed to be interactive.

3. Overview

In this work we present *embedded constraint patterns* as a way to specify stylistic constraints on a sequence of multiple shots, constraints that can be complex (e.g. overlapping or embedded) and recurring (e.g. consistent relations between consecutive shots in the sequence) for interactive applications in 3D environments.

Taking as input a 3D animation that is cut into a number of shot segments, with one or more ECPs applied to any number of the shots, our solver would output recommendations for each shot that fulfils all the ECPs applied. At the core of these recommendations is a shot database with annotated framings–character positions, angles, shot sizes, etc.–of actual film clips, from which the solver filters and selects suitable framings to propose to the user. After producing an initial solution, the solver is then designed to accept user interactions such as cutting a shot in two, adding or removing shots, adjusting shot length, applying an ECP to specific shots in the sequence, or selecting a specific framing composition for a shot. On these interactions, the solver in real-time re-filters the shot database to present framing compositions for each shot that satisfy all ECPs applied to the sequence.

In the next sections, we will first provide a summary of the vocabulary and syntax of *Patterns* and ECPs, and we introduce the shot database. We then introduce our solver for interactive contexts.

4. Patterns Language and Embedded Constraint Patterns

From film textbooks, we have identified a number of visual features to describe the on-screen characteristics of actors in relation to the camera: actor position features, movement features, and size and angle features. Our selection of features closely follows the editing chapters of a filmmaking textbook [Zet07]. We then use these elements as constraints to define embedded constraint patterns (ECPs).

This section reviews our previous work on defining the *Patterns* language, focusing on those vocabulary used in later sections.

4.1. What is an Embedded Constraint Pattern (ECP)?

The definition of an ECP sets a number of constraints on the visual features of a sequence of shots. There are 3 types of constraints: (1) framing, (2) shot relations, and (3) sub-sequence constraints. Below, we describe each category in detail.

4.1.1. Framing Constraints

Framing constraints restrict how actors are arranged on-screen, mainly including four visual features: *size*, *angle*, *region*, and *movement*. Each of these parameters has strong definitions in literature. The constraints are applied to actors. We adopt a very broad definition of actors that incorporates humans, animate creatures, and objects.

4.1.1.1. Size Closer cameras create bigger actors, increasing their importance; conversely, longer camera distance makes the actor smaller and the less important. We categorise 9 sizes, as defined by [TB09] with the upper half body filling the screen as the median Medium Shot. Shot size can also be determined based on the actor's head size.

4.1.1.2. Angle Corresponding to the three axes in 3D, the camera can be rotated horizontally, vertically, and rolling to convey inner emotional states of characters, or to express relative power relations between two characters.This value is deduced from the azimuth of the head relative to the front-facing vector of the actor.

4.1.1.3. Regions Framing region refers to how actors are arranged in the screen space. Where actors appear on screen has much to do with aesthetics, but also the inner state of the actors. *Patterns* provides a simple set of vocabulary for roughly constraining the regions either by a 4-*split* or 9-*split* regions, or simply *top*/*bottom* and *left*/*right*, as shown in Figure 2.

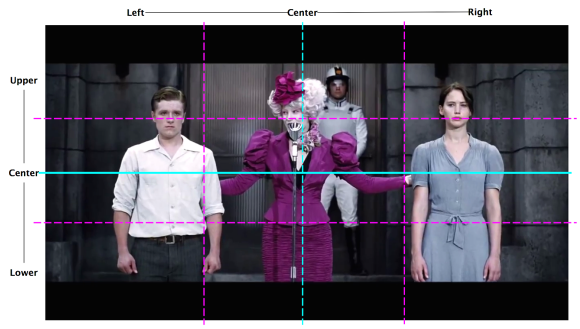

Figure 2: *There are many ways to describe on-screen positions of actors. In film literature, the most well-known one is the 9-split standard (purple dotted lines; also referred to as the rule of thirds), where the screen is split into three regions horizontally and vertically. Patterns provides vocabulary for* 9-*split*, 4-*split (blue lines in the figure; 2 regions both horizontally and vertically), and also left/right (dotted blue line) and upper/lower (solid blue line) regions.*

4.1.1.4. The number of targets The number of targets indicates the relative importance of actors or props in the scene, which are significant to the current story event.

4.1.2. Shot Relation

Constraints can be placed not only on single framing specifications, but also the relation between any two shots. Here we refer to relations as the description of properties between two shots in terms of on-screen properties: distance, angle, and framing regions. For example, we may want to search for a sequence of shots that move gradually closer to actors; or a sequence of shots in which the actors always appear in the same region in the frame. Relation constraints provides ECPs with the ability to detect full sequences following certain constraints, giving a strong advantage over other computational cinematography languages.

4.1.2.1. Size Relations Changing the distance from one shot to another can thus show the difference in importance of actors in the scene, as well as to intensify or relax the atmosphere by moving closer and further to targets respectively. The distance of the camera to actors in consecutive framings can either be closer, further, or remain the same.

4.1.2.2. Angle Relations Angles can carry the meaning of confidence or importance of actors, change of angles between shots can imply the relative strength of different actors, or change of emotional state for the same actor. Similar to size, angles can be either higher, lower, or the same angle.

4.1.2.3. Region Relations When actors appear in the same regions on the screen across shots, it often means an agreement, compassion, or mutual recognition between the actors. If actors are distributed on different sides of the horizontal axis (i.e. left and right), it often carries the meaning of opposition. *Patterns* provides the basic values *same* and its negation !*same* that can be assigned to the *region* (the same 9-*split* or 4-*split* region), the *horizontalRegion*, and the *verticalRegion*.

4.2. Defining an ECP - Language and Syntax

An ECP can contain multiple framing and relation constraints. Further constraints such as length and embedded sub-sequences can also be added to ECPs.

We will use the incremental construction of an intensifying sequence–a sequence where the camera gradually approaches the actors over a number of shots–to illustrate how an ECP is structured. We show how the combination of different vocabulary in *Patterns* can provide multiple definitions of the intensify ECP.

4.2.1. Sub-Sequences

Embedded sub-sequences are continuous sequences of shots that follow the constraints of some other ECP, such that the sequence of shots can be grouped together in the parent ECP. Individual shots in the sub-sequence are not evaluated by the relation constraints set by the parent ECP. Suppose we defined a very simplified definition of intensify as:

```
intensify{
    relation
        constraint: closer
    sub-sequence
        constraint: 'shot'
}
```

meaning that all shots must have a shot distance relatively closer than the previous shot, and intensify should be solely comprised of single shots. Then intensify would be able to match a sequence of shot sizes [Long shot][Medium shot][Close-up], but it would not consider [Long shot][Medium shot][Medium shot][Close-up] as intensify since the two middle shots are not *closer* in relation. Yet, as a film analyst, this may still be considered an intensify, since the shot size gradually increases over the whole sequence. To overcome the limitation, one solution is to allow same-sized shot sequences

embedded in the larger intensify sequence. That is, we allow sequences of 1 or more same-sized shots that ignore the relation constraint of *closer*.

```
intensify{
    relation
        constraint: closer
    sub-sequence
        constraint: ECP{
            relation
                constraint: same-size
        }
}
```

Only the first and last shot in the sub-sequence are restricted by the relation constraint. In this case, the above-described sequence would still validate as intensify.

4.2.2. Ranges

If the user would like a constraint to only be evaluated once, such as at the first or last shot of the sequence, or interweave shot sizes where shots 1, 3, and 5 are CU, and 2, 4, and 6 are LS, it can be achieved through setting ranges for constraints. A range parameter that can either be a continuous range expressed as $[x\text{-}y]$, a discrete list $< x, y, z... >$, or it can be one of the keywords of *initial* (the first container in the sequence, equal to $< 1 >$ in the discrete list representation), *all* (all containers in the sequence), *none* (a strong negating constraint on all containers in the sequence), or *end* (the last container of the sequence). In the intensify ECP, we can add range restrictions to each constraint. By default, range is set to *all*.

```
intensify{
    framing
        constraint: size>=MCU
        range: initial
    relation
        constraint: closer
        range: all
    sub-sequence
        constraint: ECP{
            relation
                constraint: same-size
                range: all
        }
        range: all
}
```

4.2.2.1. Length We can also restrict the length of an ECP, which indirectly affects the number of shots an ECP can match. The length parameter is targeted towards the number of sub-sequences, and not the number of shots. Here we add a length constraint to intensify:

```
intensify{
    length
        constraint: length>=3
    relation
        constraint: closer
        range: all
    sub-sequence
        constraint: ECP{
            relation
                constraint: same-size
                range: all
        }
        range: all
}
```

The reason for setting a length requirement on sub-sequences instead of number of shots is central to our goal of using *Patterns* to define ECPs that capture recurring changes over sequences. Since the evaluation of relational constraints allows observations only between sub-sequences, the number of sub-sequences on which a relation constraint is evaluated is much more meaningful than the actual number of shots in the ECP.

5. A Framing Database

Patterns was designed to work with real film data, and in our previous work, we established a framing database of framing compositions. This database contains annotations of 22 film clips from 18 different films of roughly 5 to 10 minutes in length. These clips were selected from famous scenes in each film based on a YouTube search. In each selected clip, at least one framing is recorded for each shot with specific features, namely, when available:

- list of all the actors in the framing
- the head, left eye, and right eye positions of each character in the framing is annotated
- other significant body parts, such as hand, foot, when the head and eye positions are not available
- non-animate objects crucial to the story are also annotated
- the angle of the camera relative to the main character
- a frame number (from the actual film clip)

The head is annotated in terms of its size, position, and azimuth angle while other elements are annotated for their on-screen x and y positions respectively as a ratio of the width and height of the screen space.

The above information is sufficient for extracting framings for most visual constraints included in the *Patterns* vocabulary that we described in the previous section, and will remain constant if applied to a 3D environment. The only exception is shot distance, which is variable according to the current positions of the actors in the 3D scene to which we wish to apply a specific framing.

We use the Toric manifold method introduced by [LC15] to calculate a camera position that can realise a framing to two on-screen targets. When the framing only contains one actor, the two eyes are designated as the two targets. When the framing contains two or more actors, the head position of the two most important actors (based on head size on the screen) are considered as the targets. With the Toric manifold, we can recalculate a camera position relative to the virtual actors that realises the framing, and assigns a new distance value to the framing according to the distance between the camera and the actor.

Our current database holds 1018 shots, with 1184 framings annotated, including a variety of shot types from different camera distances, angles, and target types.

In the next section, we present the solver whose goal is, given a list of ECPs applied to specific shots, for each shot propose a selection of framings from the database that would fulfil all the ECPs applied to the sequence.

6. ECP Solver

There are two ways in which we use *Patterns* in cinematographic contexts. The first is to search for matching ECP sequences in annotated film data, which is the main contribution in our previous work [WC16a] for the purpose of automated analysis of film style. We show how ECPs have a close link to elements of storytelling and emotion. In extension to this, we beleive ECPs can be then used in creative contexts for virtual cinematography, where one could imagine generating camera sequences, with the help of annotated film data, in which the shots pertain to the constraints of an ECP to create a certain emotion that is linked to the ECP–e.g. intensification of emotion using the intensify ECP. This contribution focuses on this creative aspect. In this section we explain the mechanisms of the interactive solver, the algorithm for arriving at an initial solution, and how it responds to user interactions such as cuts, edits, or adding ECPs to a sequence.

6.1. Problem Overview

The aim of the solver is to allow users to apply ECPs to a number of selected shots in the sequence, and in return the solver would propose suitable framings for each shot that can fulfill the ECPs. By applying an instance of an ECP to the selected shots, the user is setting a constraint to fulfil the ECP on all the shots selected, as a sequence. The job of the solver component is to remove the framing specifications (from the shot database of each selected shot) that violate or cannot fulfil the ECP that the user applies, arriving at an initial solution: all framing propositions from the framing database for each shot that fulfills all ECPs if there exists one. Thus, the idea is to limit the framing specifications that the user can choose from in each selected shot to only those that can fulfil the ECP instance.

6.1.1. Initial Solution

Algorithm 1 ConstraintPropagate (ECPInstanceList P)

```
1:  ECPListCopy P'=P
2:  while P'.count!=0 do
3:      ECPInstance e=P'.pop()
4:      ECP p = e.ECP
5:      ShotSequence S = e.S_[x,y]
6:      ECPFilter(p,EmptySet,0,S)
7:      for all ECPInstance ei ∈ P do
8:          for all Shots s ∈ S do
9:              if s∈ei.S_[x,y] and ei∉P' then
10:                 P'.add(ei)
11:                 break
```

The solver initialises each shot with a list of candidate framings (from the shot database). The main body of the algorithm (Algorithm 1) is a constraint propagation method that maintains and updates an active list of ECP instances that need to be solved. Each ECP instance contains an ECP and an ordered list of shots $S_{[x,y]}$ to which the ECP is applied. Algorithm 1 iterates on the list of ECP instances, and incrementally filters the framing candidates for each shot in the ECP instance by calling Algorithm 2. Thus at each iteration, Algorithm 1 produces the subset of framing candidates that fulfils this ECP instance and all the preceding ECP instances. If the ECP instance removes framings in shots that overlap with other solved ECP instances, the constraints must be propagated by adding those affected instances back to the active list of ECP instances so that they can be re-solved.

Algorithm 2 ECPFilter (ECP p, FramingSeqeunce F, Position pos, ShotSequence **S**)

```
1:  if pos<=S.count then
2:      for all Framings f ∈ S_pos.candidateFramings do
3:          F.add(f)
4:          ECPFilter(p,F,pos+1,S)
5:          F.remove(f)
6:  else
7:      if IsValidECP(p, EmptySet, F) then
8:          for all Framings f in F do
9:              f.validate()
```

Algorithm 2 filters the framings among the candidate framings for all shots $s_j \in S_{[x,y]}$. Each possible framing f_i from the candidate framings must be either validated or rejected as a candidate for s_j in this ECP instance based on the following condition: if there exists a sequence of framings $f_x, f_{x+1}, ...f_i, ...f_y$ for each shot $S_{[x,y]}$ that fulfils the constraints set by the ECP (Algorithm 3), then f_i is validated, which means the framing should be available for the next ECP instance. If no combination containing f_i can be validated, then f_i is rejected. At the end of the process, the remaining framings for each shot are made available to the user for interactive selection.

Algorithm 3 IsValidECP (ECP p, CurrentSequence **C**, Sequence **F**)

```
1:  if F not empty then
2:      f_i = first(F)
3:      if isValidFrame(f_i) AND isValidRelation(f_i) then
4:          if isValidSubsequence(p, C ∪ {f_i}) then
5:              return IsValidECP(p, C ∪ {f_i}, F \ {f_i})
6:          else if isValidSequence(p, C ∪ {f_i}) then
7:              return IsValidECP(p, {f_i}, F \ {f_i} )
8:          else
9:              return False
10: else if ValidateLength(C) then return True
    return False
```

Suppose there is a framing database of n framing specifications over a sequence of m shots, the complexity of the algorithm would be at the worst case n^m, which makes the algorithm quite slow, since it is a full search over all possibilities.

6.1.2. Interactive Solving

The solver is called under all circumstances when a new ECP instance is added by the user, when shots are added or removed from one or more ECP instances, or when a framing is selected for a shot in one ore more ECP instances. The solver reacts to these three types of actions as following:

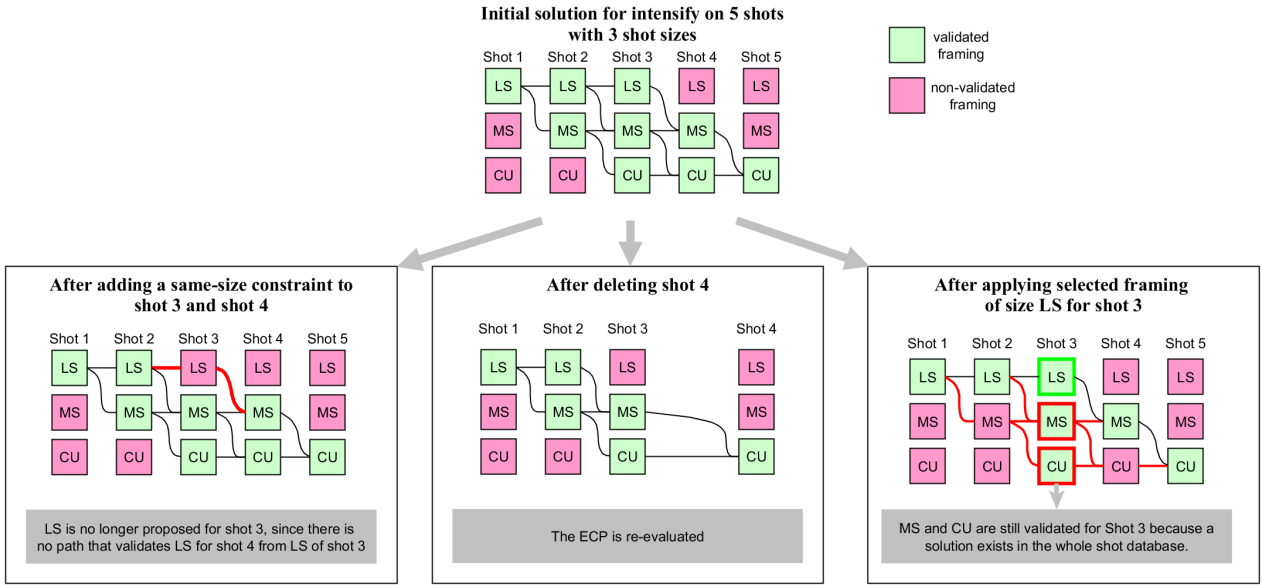

Figure 3: ***The solver for Patterns interface*** *is called both to calculate an initial solution and when a user's action affects an existing ECP. This figure shows how the solver filters the available framings for intensify on a five shot sequence, where each shot has the choice of a long shot (LS), medium shot (MS), and close-up shot (CU). In the initial solution, all framings that have a path from the first shot to the last are validated. Three possible interactions are then shown below the initial solution. The first is when another ECP, same-szie, is applied to shots 3 and 4, effectively filtering the LS shot from shot 3. The second interaction is when the shot 4 is deleted from the sequence, the solver re-evaluates intensify for the remaining shots. The third interaction is when the user selects a LS framing for Shot 3, shots 2 and 4 are filtered again to only allow framings that have a valid path through the LS framing of Shot 3.*

6.1.2.1. A new ECP instance is added: When a new ECP e is added to the sequence, e is added to the ECPInstanceList P of Algorithm 1. If when solving e, the solver removes framing propositions from shots with another ECP instance e', then e' is added to the ECPInstanceList P. In this manner, the constraints of e are propagated to all other overlapping ECPs as well as those indirectly overlapping. Thus the solver continues to solve for each instance in P until the algorithm converges, or until no solution can be found for a specific instance.

6.1.2.2. Adding/Removing of a cut or a shot: The shot s is added/removed from the range of all overlapping ECP instances, and all the ECP instances are pushed into the ECPInstanceList P in Algorithm 1 to be re-evaluated. If no solution exists for an ECP, the ECP is removed. Each time an ECP is removed or if the ECP removes proposed framings from a shot, all overlapping ECPs are pushed into the ECPInstanceList P to be re-evaluated.

6.1.2.3. A framing F_m is selected for a shot S_n: F_m is set as a new constraint and propagated to overlapping ECP instances. All framing proposals as well as selected framings for other shots must have a validated path through F_m for S_n, which means in Algorithm 2, the list of candidate framings can include only the user-selected framing when other shots would like to validate their candidate framings. However, if we simply set F_m as a hard constraint by removing all other framings of S_n from the candidate framings list, this would result in only one available framing F_m to select from for S_n, avoiding the user from changing to another framing that can also be validated by the whole sequence. Thus, we would still want to be able to propose for S_n an augmented set of framings where, despite not the selected framing of the user, still have a valid solution through framings of other shots that have been validated by all other ECPs and constraints. To accomodate this, we simply add an additional condition to Algorithm 3 to return true only when the ECP is validated, and when all other framings in Sequence F are selected framings (where available). This allows us to "validate" a framing that is not selected by the user, but still contains a valid solution in the sequence. This also prevents other shots to use this augmented set to validate their own framings, since Algorithm 3 will only return true when all other framings apart from the one at Position *pos* of Algorithm 2 must be a selected framing if the user has selected one. Figure 3 shows how a user action of selecting a framing for a single shot would trigger the solver, and what the solver would do to uphold the ECPs on the new sequence.

7. Examples

We use a 3D animation of the *Back to the Future* scene for our sample scenario. The 79 second scene shows Marty McFly talking to his father George McFly. In this example, we have cut the scene into five shots, and applied the intensify ECP to the first three shots, and the frameshare ECP to the fourth and fifth shot. We then selected a proposed framing for each of the five shots to have an edited sequence that fulfils both ECPs on the selected shots. The

(a)

(b)

Figure 4: *Here we show (a) a screenshot of our editing tool and (b) four possible framing selections for the same sequence of five shots with two ECPs output by the interface, intensify and frameshare, applied respectively to shots 1 to 3 and shot 4+5. For all sequences, notice how the shot size increases for the first three shots–fulfilling the intensify ECP–and how the actors appear on the same side horizontally of the framing for shots 4 and 5–fulfilling the frameshare ECP.*

accompanying video shows how the interactive solver would arrive at an initial solution and propose further filtered framings for each interaction. After each interaction, we can see how the framing database for each shot is filtered. Figure 4 shows four possible sequences. All the sequences conform to the constraints posed by the two ECPs.

In using the editing tool, the user can carry out a number of actions, including:

- carry out basic editing actions such as cuts and adjusting shot lengths
- browse the framing database and assign a framing to a shot
- select a number of shots and apply a pattern

On applying a pattern, the available framing selections for each shot will be filtered and reduced to only those that can fulfill the pattern. Likewise, when a framing is selected for a shot with patterns applied, the framings available to other neighboring shots with the same overlapping patterns instance will be filtered to conform to the user's selected framing. When no solution is found by the solver for applying a pattern, an error is shown to the users, and the pattern is not applied.

From this work we are able to see the strengths of the ECP solver as a creative assistant in an interactive context. First of all, it allows experimentation of various observed camera styles on a 3D sequence, defined using the *Patterns* vocabulary. Second, the solver proposes framings in an educational way by hiding framings that violate constraints, while making them available again when ECPs are removed or when the user changes selected framings for a given shot. Finally, the solver preserves the user's creative autonomy in filtering and proposing multiple possible framings for each shot, but not selecting a solution, and instead, leaves the final decision of which framing to use for each shot up to the user.

8. Conclusion

In this work we have introduced a way of specifying recurring patterns of style over film sequences for interactive applications. There are many directions of work that can be extended from the current status of *Patterns*. We are at the final stages of carrying out experiments for the assisted creativity application for 3D camera editing that uses the interactive solver to provide the user smart feedback during the editing process. In the future we envision developing this work for educational purposes as well as pre-visualisation of films. The film database will also be released and made available for other creative or research applications.

References

[BGL98] BARES W. H., GRÉGOIRE J. P., LESTER J. C.: Realtime constraint-based cinematography for complex interactive 3D worlds. In *The National Conference On Artificial Intelligence* (1998), Citeseer, pp. 1101–1106. 2

[BTM00] BARES W. H., THAINIMIT S., MCDERMOTT S.: A Model for Constraint-Based Camera Planning. In *AAAI Spring Symposium* (Stanford, 2000). 2

[CAH*96] CHRISTIANSON D. B., ANDERSON S. E., HE L.-W., SALESIN D. H., WELD D. S., COHEN M. F.: Declarative camera control for automatic cinematography. *AAAI Conference on Artificial Intelligence* (1996). 2

[DZ94] DRUCKER S. M., ZELTZER D.: Intelligent camera control in a virtual environment. In *Graphics Interface âĂŹ94* (1994), pp. 190–199. 2

[GCLR15] GALVANE Q., CHRISTIE M., LINO C., RONFARD R.: Camera-on-rails : Automated Computation of Constrained Camera Paths. *ACM SIGGRAPH Conference on Motion in Games 2015 (MIG2015)* (2015), 151–157. 2

[GCR*13] GALVANE Q., CHRISTIE M., RONFARD R., LIM C.-K., CANI M.-P.: Steering Behaviors for Autonomous Cameras. *Proceedings of Motion on Games - MIG '13* (2013), 93–102. 2

[GRLC15] GALVANE Q., RONFARD R., LINO C., CHRISTIE M.: Continuity Editing for 3D Animation. In *AAAI Conference on Artificial Intelligence* (Austin, Texas, United States, jan 2015), AAAI Press. 2

[LC15] LINO C., CHRISTIE M.: Intuitive and Efficient Camera Control with the Toric Space. *Transactions on Graphics 34*, 4 (2015). 2, 4

[MBC15] MERABTI B., BOUATOUCH K., CHRISTIE M.: A Virtual Director Using Hidden Markov Models. *Computer Graphics Forum* (2015). 2

[RVB13] RONFARD R., VINEET G., BOIRON L.: The Prose Storyboard Language. In *AAAI Workshop on Intelligent Cinematography and Editing* (2013). 2

[TB09] THOMPSON R., BOWEN C. J.: *Grammar of the Shot*. 2009. 3

[WC15] WU H.-Y., CHRISTIE M.: Stylistic Patterns for Generating Cinematographic Sequences. In *Proceedings of Eurographics Workshop on Intelligent Cinematography and Editing* (2015). 1

[WC16a] WU H.-Y., CHRISTIE M.: Analysing Cinematography with Embedded Constrained Patterns. In *Proceeding of Eurographics Workshop on Intelligent Cinematography and Editing* (2016). 1, 5

[WC16b] WU H.-Y., CHRISTIE M.: Detecting Cinematographic Patterns. In *Conference on Cognitive Science and Moving Images 2016 (SCSMI2016)* (Ithaca, 2016). 1

[Zet07] ZETTL H.: *Sight, sound, motion: Applied media aesthetics*. Wadsworth Publishing Company, 2007. 2