Thinking Like a Director: Film Editing Patterns for Virtual Cinematographic Storytelling

Hui-Yin Wu, Francesca Palù, Roberto Ranon, Marc Christie

Thinking Like a Director: Film Editing Patterns for Virtual Cinematographic Storytelling

HUI-YIN WU, North Carolina State University
FRANCESCA PALÙ, University of Udine
ROBERTO RANON, University of Udine
MARC CHRISTIE, University of Rennes 1, IRISA, INRIA

This paper introduces *Film Editing Patterns (FEP)*, a language to formalize film editing practices and stylistic choices found in movies. FEP constructs are constraints, expressed over one or more shots from a movie sequence that characterize changes in cinematographic visual properties such as shot sizes, camera angles, or layout of actors on the screen. We present the vocabulary of the FEP language, introduce its usage in analyzing styles from annotated film data, and describe how it can support users in the creative design of film sequences in 3D. More specifically, (i) we define the FEP language, (ii) we present an application to craft filmic sequences from 3D animated scenes that uses FEPs as a high level mean to select cameras and perform cuts between cameras that follow best practices in cinema and (iii) we evaluate the benefits of FEPs by performing user experiments in which professional filmmakers and amateurs had to create cinematographic sequences. The evaluation suggests that users generally appreciate the idea of FEPs, and that it can effectively help novice and medium experienced users in crafting film sequences with little training.

CCS Concepts: •**Computing methodologies** → **Animation; Virtual reality;** •**Human-centered computing** → *Interactive systems and tools;* •**Applied computing** → *Media arts;*

Additional Key Words and Phrases: film storytelling, editing, virtual cinematography, assisted creativity, 3D animation

ACM Reference format:
Hui-Yin Wu, Francesca Palù, Roberto Ranon, and Marc Christie. 2018. Thinking Like a Director: Film Editing Patterns for Virtual Cinematographic Storytelling. *ACM Trans. Multimedia Comput. Commun. Appl.* 0, 0, Article 0 (August 2018), 23 pages.
DOI: 0000001.0000001

1 INTRODUCTION

"Good artists borrow, great artists steal." Cinematography often takes inspiration from other films. Best practices for camera placement, movement, and editing have been written into film textbooks, and are used over and over again in movies in various genres, because they are effective in conveying specific actions or emotions. The same knowledge widely used in films can also benefit storytelling in 3D animated scenes, which are becoming increasingly popular, for example, to pre-visualize films for reducing the cost and time on a real set, to pitch creative ideas, and to add cinematic sequences to video games and educational media.

Cameras in films are deliberately placed to ensure spatial and temporal continuity, and guide the audience's interpretation of events. Figure 1 shows how changing the on-screen layout of characters in shots conveys different emotions. 3D animations can replicate similar stylistic patterns observed in films. Such constructs are widely used to support the director's intentions.

However, existing tools for creating 3D cinematic sequences do not ease the design and implementation of such constructs. Besides being typically quite complex to use, these tools are not able to encode elements of film language, such as shot size (sizes of characters on the screen) or on-screen regions, that are frequently used in film editing practice.

Fig. 1. By changing the camera angles, and hence the on-screen layout of characters, different emotional hints can be conveyed. Here the CG sequence replicates shots and edits observed in a real movie (film screen shots from *Pulp Fiction*).

To provide a more creative support for 3D cinematography, we propose *Film Editing Patterns* (FEP), a language to formalize film editing practices and common editing techniques (called idioms) used in film, using vocabulary from film textbooks. FEP constructs are *constraints expressed over one or more shots* that act on elements of visual style such as size, layout, and angle of actors on the screen. For example, FEPs enable one to encode typical shot-reverse-shot patterns (i.e. switching the camera between two people who are looking at each other, as in Figure 1) that are common when filming dialogues, with fine control over actors' position and size in the frame. In addition to proposing such a language, we provide an algorithm that, given a set of FEPs applied to multiple shots, is able to select the cameras for each shot that make the sequence conform to the specified FEPs.

To investigate the potential of our proposal, we have created a tool for shooting and editing 3D animated sequences (i.e., a basic animated storyboarding / previsualization tool) where users, while still having manual control over the edited sequence, can use FEPs as a way of rapidly experimenting with creative and stylistic possibilities. To assess the acceptance and practical usefulness of our approach, we have conducted an evaluation with both professional filmmakers and amateurs with little to no filmmaking experience.

The rest of the paper is organized as follows. Section 2 presents related work. We then provide in Section 3 an overview of the FEP language with examples. Section 4 introduces automated film analysis as a possible usage for FEPs, while Section 5 presents our editing tool and the algorithm

for solving FEP constraints over multiple shots. In Section 6, we present our user evaluation, and in Section 7 we discuss the limitations of our work and future improvements.

2 RELATED WORK

Various techniques have been developed to instill film knowledge into virtual cinematography systems. In general, the approaches developed so far concentrate on how actors are arranged on the scene (i.e. frame composition), basic film idioms, and continuity rules that define how shots can be ordered as to maintain spatial and temporal continuity, without confusing the viewer. Most existing approaches do not consider stylistic editing choices over multiple shots.

Drucker [7] was one of the first to propose an idiom-like language for constraint-based cinematography that allows a smooth navigation of the camera through a complex virtual environment, like a museum room. Other early examples include Christianson et al. [5], who introduced the DCCL language for planning sequences of camera movements, and He et al. [11], who encoded film idioms as finite state machines. Bares [3] introduced a constraint-based system for camera placement, gathering information about the occurring story events and making decisions on how to best show the ongoing actions. Similarly, Bares et al. [4] 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, and the system will find framings that satisfy those constraints. Notably, the Prose Storyboard Language (PSL) [19] was designed based on actual film practice on shot composition, including vocabulary for elements like size, region, or movement. In the implementation by Galvane et al. [9] for autonomous camera steering and composition, a search is conducted for camera positions based on PSL constraints. In a later work, they use semi-Markov chains to optimize a number of parameters in camera shot and cut decisions [10], motivated by film practice of evaluating narrative importance of actors, visual continuity, and rhythm in the edits. The approach developed by Lino et al. [16] is another example of a system that is able to autonomously edit a sequence by choosing where to position cameras and how to perform edits that conform to continuity rules. Recently, Leake et al.[14] proposed an automated editing system that would interactively generate an edit, from a number of clips of a same dialogue scene, based on idioms that best interpret the emphasis or desired feeling of the user. Compared to existing idiom-based systems, our approach provides a vocabulary and syntax based on elements of visual style in order to define a range of idioms potentially larger than existing work.

In interactive storytelling, planning techniques [2] and [12] have proposed camera placement algorithms that correspond to the current story context. Film idioms and their associated storytelling meanings are encoded, and a planner chooses the best camera position for each event, while optimizing transitions between shots to improve the fluency of the whole sequence. Elson and Riedl [8] adopted the idea of blockings from film practice, which involves the absolute placement of a number of characters and a camera in a non-occluded environment. The database of blockings, stages, and shots can be expanded. The camera is placed in pre-calculated relative positions in relation to the actors and stage, based on what is happening in the scene (e.g. how many actors are involved, scene type, action type). Constraints are also placed on how shots can be sequenced to avoid violating continuity rules.

The vast majority of virtual cinematography systems developed so far are non-interactive, and thus are not meant as tools to support users' creativity, with some notable exceptions, e.g. [17], which however only considers continuity rules between adjacent shots. A number of applications in the market, mainly devoted to pre-visualization, allow a user to position cameras and perform editing on 3D animated scenes. However, in these applications users need to control camera positioning and editing at low level, and no knowledge of cinematography practices exist to assist

users in their task. This kind of assistance would at least benefit novice users, as shown by a Wizard-of-Oz study conducted by Davis et al. [6], where novice filmmakers could make fewer editing errors by benefiting from guidance about violated cinematography rules.

Our work is also related with computational techniques for film analysis work. With respect to this, video and image processing currently have refined methods for shot boundary detection [1] and genre analysis [18]. Scene and event boundaries provide information on how a film is structured, while genre identifies crucial emotional or story-archetype features. This has strong applications to categorization and recommendation systems of streaming services. Another aspect of cinematographic analysis focuses on the aesthetics of framing composition such as detecting shot types [20] or meaningful sequences using continuity rules [23][21].

Current approaches both in analytical and generative systems already address defining film idioms and style over shot sequences. We hope to expand accessibility of film knowledge in automated film analysis and creativity tools with a filmmaking-friendly language that allows flexible definition and expansion of film idioms that operate over multiple shots.

3 FILM EDITING PATTERNS

Film Editing Patterns (FEP) is a language to formalize film editing practices that span multiple shots. More specifically, *FEP constructs* are constraints on the visual features of a sequence of shots. These constructs are assembled from simple properties or relations that act on the visual features of a single shot (*framing properties*), or on the relation between subsequent shots (*shot relations*). In addition, one can restrict the length of shot sequences or require the presence of certain sub-sequences. Hereinafter, for simplicity, we use the term *FEP* to denote either the language or a construct built with the language. The current language is extended from a previous version published in [24]. Here, we define the vocabulary of the language, and syntax of FEP.

3.1 Framing Properties

Framing properties define how actors are arranged on-screen (the on-screen layout), using four visual features: actors' *size*, *angle*, *region*, and number of *actors*. Each of these features has precise definitions in film literature [26]. We adopt a broad definition of actors that incorporates humans, animated creatures, and objects.

Size. Shorter shots (i.e., closer cameras for a given lens) display bigger actors, increasing their importance; conversely, longer shots make the actors smaller and thus less important, or more in relation with their environment. We use the 9 shot size scale [22] with the upper half body filling the frame as the median Medium Shot. Figure 2 shows all the shot sizes in the FEP language.

Angle. The camera uses horizontal, vertical, and roll angles with respect to actors to convey inner emotional states of actors, and express power relations between characters. For example, shooting actors from a lower angle creates a feeling that they are powerful, towering over the audience, whereas shooting actors from a high angle, with them looking up to the camera, gives the audience a feeling of dominance over the actors. Roll angles, more rarely used, give off a feeling of instability, since the ground is not parallel to the frame. Figure 3 shows all the angles provided in the FEP language.

Region. Framing region refers to how actors are arranged in the frame. Good arrangements improve aesthetics and can convey the inner state of actors or relations between actors. Our language provides a number of ways to describe the region of an actor in the frame:

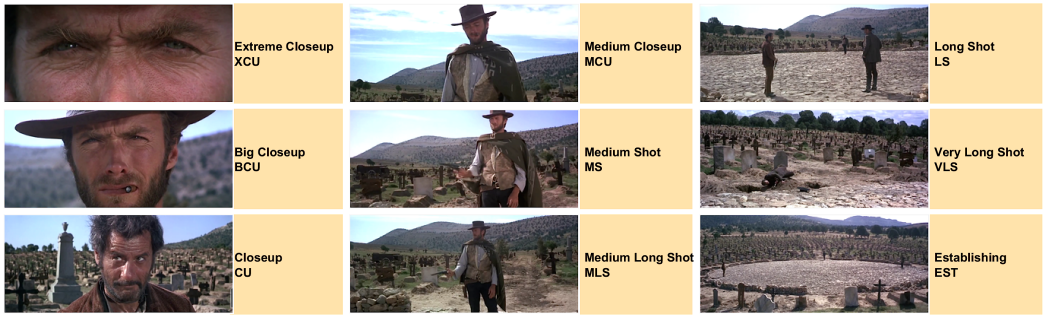

Fig. 2. The nine framing sizes in the FEP language all appear in the same sequence in *The Good, The Bad, and the Ugly*.

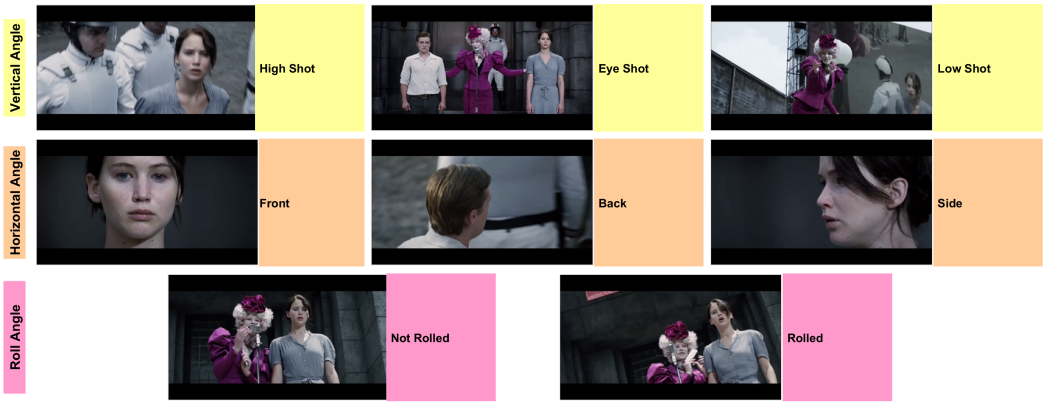

Fig. 3. Angles of characters on the screen can be categorized into vertical, horizontal, and roll angle.

- 9-split: the screen is split into three equal regions horizontally and vertically, and regions are named $R9\_1$ to 9, with $R9\_1$ being the upper-left of the frame, and $R9\_9$ being the lower-right, as labelled in red on the example in Figure 4.
- 4-split: the screen is split into two equal regions horizontally and vertically, and regions are named $R4\_1$ to $R4\_4$, with $R4\_1$ being the upper-left quarter of the frame, and $R4\_4$ being the lower-right quarter, as labelled in blue on the example Figure 4.
- 3-split horizontal/vertical: the screen is split into three equal regions horizontally/vertically, and named $R3\_UPPER$, $R3\_MIDDLE\_VER$, and $R3\_LOWER$ in a vertical split, and $R3\_LEFT$, $R3\_MIDDLE\_HOR$, and $R3\_RIGHT$ in a horizontal split, as indicated in the green labels on the outer rim of Figure 4.
- 2-split horizontal/vertical: the screen is split into two equal regions horizontally/vertically, named $R2\_UPPER$ and $R2\_LOWER$ in a vertical split, and $R2\_LEFT$ and $R2\_RIGHT$ in a horizontal split, as indicated in the yellow labels on the outer rim of Figure 4.

The regions are visualized in Figure 4. To decide the region, we focus on the most important aspects of actors: their head and eye positions, and possibly limb positions in scenes with interactions.

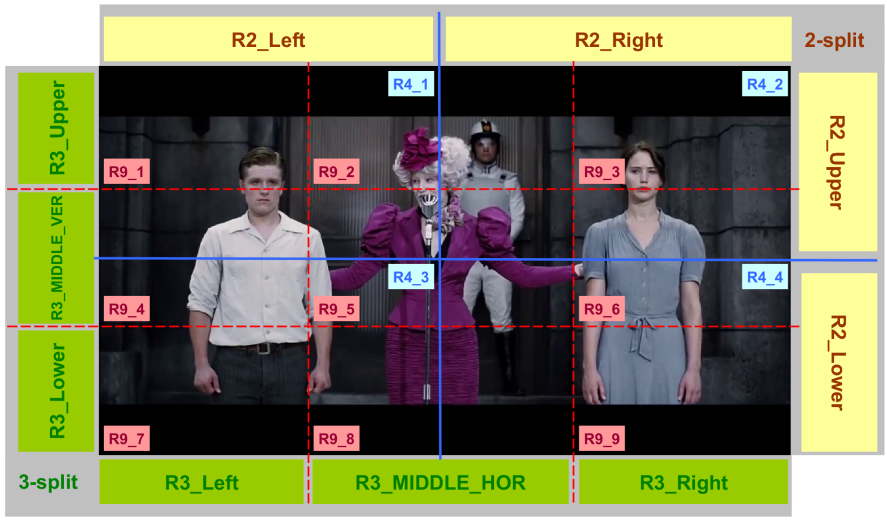

Fig. 4. The region in which actors appear can be described based on a 4-split (blue lines) or 9-split (red lines) of the screen. They can also be described in terms of the 2 or 3 split of the screen vertically and horizontally. As an example, the nose of Jennifer Lawrence (the right-most actor) can be described as appearing in the regions *R*4_2, *R*9_3, *R*2_*RIGHT*, *R*2_*UPPER*, *R*3_*UPPER*, and *R*3_*RIGHT* (shot from *Hunger Games* movie).

Number of actors. The number of actors indicates the relative importance of actors in the shot. If each shot conveys equal amount of information, the more actors there are on-screen, the less important each of them is to the current event.

3.2 Shot Relations

Shot relations establish relationships between framing properties of two or more shots, like *size*, *angle*, or *region*. For example, we may describe a shot sequence that moves gradually closer to actors; or a shot sequence where actors appear in the same region in the frame. Shot relations provide FEP with the ability to define sequences following such constraints.

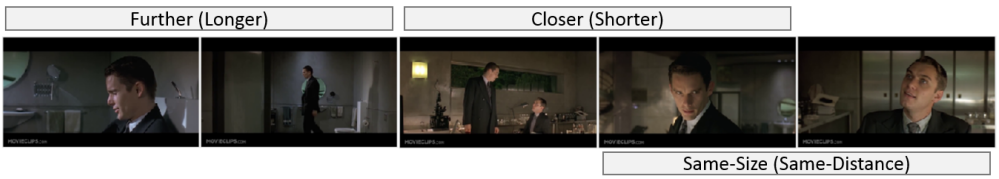

Fig. 5. A shot size relation can be relatively further (longer distance between actor and camera), closer (shorter camera-actor distance), or the same (screenshots from the film *Gattaca*).

Size. Changing the size from one shot to another can show the difference in importance of actors in the scene, as well as intensify or relax the atmosphere by moving closer or further to actors respectively. The Size relations can be *closer*, *further*, or remain the *same*. Examples of size relations can be seen in Figure 5.

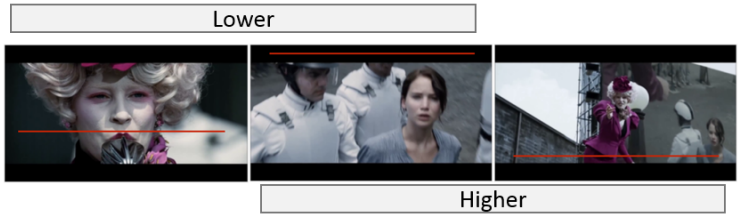

Fig. 6. Example of shot angle relation being relatively higher or lower. The red line in the figures is a rough estimate of the horizon for illustration purposes (screenshots from the film *The Hunger Games*).

Angle. Changing angles between shots can imply the relative strength of different actors, or change of emotional state for one actor. Angle relations can be either *higher*, *lower*, or the *same*. Examples of angle relations can be seen in Figure 6.

Fig. 7. Example of region relations between two successive shots (same region on the left, different regions on the right). Shots from *Infernal Affairs*.

Region. When actors appear in the same regions on the screen across shots, it often means an agreement, compassion, or mutual recognition, while laying out actors across the horizontal axis (i.e. left and right) carries the meaning of opposition. Region relations can be *same* or *diff*, with respect to the 9-*split*, 4-*split*, or *horizontal*/*vertical* in 2- and 3-split standard. Examples of horizontal region relations can be seen in Figure 7.

Actors. Two consecutive shots can show the same actors (such as to narrate a continuous action that an actor is carrying out) or different actors to show a connection between two actors. Our language defines that actor relations can either be *actor-same* or *actor-different*.

Continuity. Continuity rules ensure that the audience maintains spatial and temporal continuity from one shot to another. For example, the motion continuity rule states that if something is moving from the left to the right on the screen, it should still move left to right on the screen in the next shot. Another rule is the 180 degree rule, which states that when filming two actors, the camera should always stay on the same side of the line formed by the two actors (thus, in the same 180 degree range) maintaining their relative left/right positions. The line is referred to as the index line, and helps to ensure index continuity.

Though they are less systematically applied in modern films, we still provide a definition of most common rules as well as the vocabulary in the FEP language to express relations on *motion*/*index continuity*/*discontinuity*. Examples are provided in Figure 8.

3.3 FEP Language Syntax

Using the FEP vocabulary, we can define a variety of FEP constructs which correspond to actual filmmaking idioms. A FEP construct can contain multiple framing properties and shot relations, plus constraints on the length of a sequence (in terms of number of shots), or on the presence of a

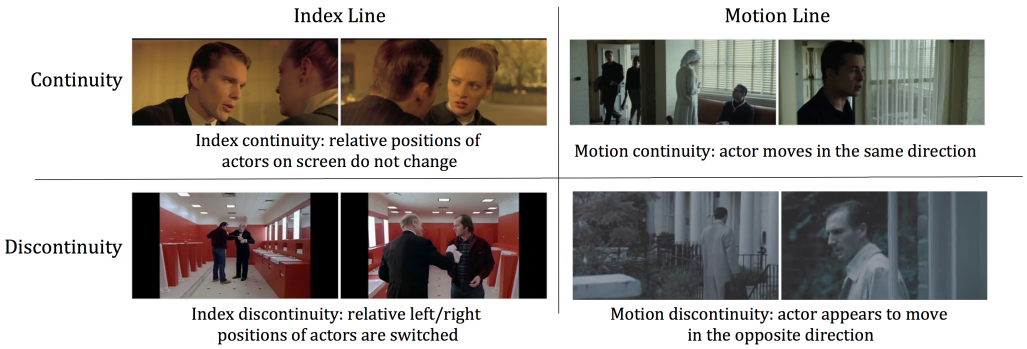

Fig. 8. Continuity relations help the audience maintain spatial and temporal continuity between two shots. *Index continuity* keeps the camera on the same side of the index line between two actors, maintaining the relative on-screen positions of actors, while *motion continuity* maintains the actor's direction of motion on-screen. In violation of these rules, the actor positions and motions are sometimes discontinuous from one shot to another (screenshots from the films *Gattaca*, *The Shining*, *The Curious Case of Benjamin Button*, and *The Constant Gardener*).

certain sub-sequence. This section presents the incremental construction of the *intensify* FEP–an editing technique where the camera gradually approaches the actors over a number of shots–to illustrate how a FEP construct is structured.

```
intensify{
  size-relation: closer
}
```

This FEP, which applies to sequences of any number of shots, states that all shots must have a shot size relatively shorter than the previous shot. This definition of *intensify* would be able to match a sequence of shot sizes like [Long][Medium][Close-up], but it would reject [Long][Medium][Medium][Close-up], since the two middle shots are not *closer* in relation. Yet, a film analyst would still consider this an intensify, since the shot size gradually increases over the whole sequence.

To overcome this limitation, we introduce the concept of *embedded sub-sequences*. Embedded sub-sequences are continuous sequences of shots that follow the constraints of some FEP, such that these shots can be grouped together in a parent FEP. In other words, individual shots in the sub-sequence are not evaluated by the relation set by the parent FEP. Using this concept, the definition of *intensify* becomes:

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

In this case, the above-described sequence would be considered as intensify. If not specified, the sub-sequence is a single shot.

If the user would like a constraint to be enforced on specific shots or sub-sequences (e.g. shots 1, 3 should be Close Ups, and 2, 4 should be Long Shots), this can be achieved by setting ranges. The range parameter can either be continuous $[x\text{-}y]$, discrete $< x, y, z... >$, or one of the keywords

between *initial* (equal to < 1 > in the discrete list representation), *all* (all sub-sequences), *none* (a strong negating constraint on all sub-sequences), or *end* (the last sub-sequence). By default, the range of a constraint is *all*. With ranges, we can refine our definition of *intensify* to add the requirement that the first shot is a Medium Shot (MS):

```
intensify{
  framing-size: MS (range: initial)
  size-relation: closer (range: all)
  sub-sequence: same-size{
      size-relation: same (range: all)
  }
}
```

Length. We can also restrict the length of an FEP, which checks the number of sub-sequences–and not the number of shots–in the sequence. Here we add a length constraint to intensify:

```
intensify{
  length: >= 3
  sub-sequence: same-size{
      length: >= 1
      sub-sequence: shot{
          size-relation: same (range: all)
      }
  }
  size-relation: closer (range: all)
}
```

We can flexibly limit a sub-sequence to a single shot or an embedded FEP, creating complex FEPs that capture recurring changes over sequences. Since the evaluation of relations 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 FEP.

3.4 Examples of Common FEPs

The FEP language provides a rich and flexible vocabulary to define a number of stylistic rules or conventions directors commonly use in their films. Here we present some examples of filming and editing techniques from film textbooks defined using our FEP constructs.

3.4.1 Composition and Continuity Rules.

Rule of thirds. A common framing composition rule for both film and photography that states the main character's head (or main target object) must appear in the upper-third of the frame.

```
rule-of-thirds{
  length: >= 1
  framing-region: R3_UPPER  (range: all)
}
```

Figure 4 is a good demonstration of the rule of thirds, where all three main characters' heads appearing on the upper third of the framing.

180 degree rule. The classical continuity rule of not crossing the index line formed by two characters can be applied directly.

```
180-degree{
  length: >= 2
  index-relation: match-index  (range: all)
}
```

The visual effect of following and violating the rule are shown in Figure 8.

3.4.2 Stylistic FEPs. From film textbooks, we identified five commonly used FEPs that are used in film storytelling, to invoke certain emotions among viewers, such as *intensify*, introduced in the previous section. Examples of all five FEPs can be seen in a clip from *Hunger Games* in Figure 9.

Fig. 9. Five FEPs with examples from an extracted film clip from *Hunger Games*: intensify, frameshare, opposition, shot-reverse-shot, and same size.

Here we provide the definitions of the four stylistic FEPs using the syntax described above.

Same-size. Actors appear the same size on screen over a sequence of 2+ shots. Used in dialogues and calm situations.

```
same-size{
  length: >= 2
  size-relation: same  (range: all)
}
```

Opposition. Actors appear individually in opposite horizontal regions over a sequence of 2+ shots. Used to express enmity, disagreement between the actors.

```
opposition{
  length: >= 2
  region-relation: R2_HOR_Diff  (range: all)
  framing-actor-num: ==1 (range: all)
}
```

Frameshare. Actors appear individually in the same horizontal regions over a sequence of 2+ shots. Used to express agreement or compassion between the actors.

```
frameshare{
  length >= 2
  region-relation: R2_HOR_Same  (range: all)
  framing-actor-num: ==1  (range: all)
}
```

Shot-reverse-shot. Two different actors appear individually in two consecutive shots of the same size with both actors looking off screen in opposite directions to indicate that they perceive each other. Shot-reverse-shot is a complex pattern with a strong psychological aspect (i.e. gaze, perception). Without story context or a 3D reconstruction of the scene, there is no way to know if two actors are actually looking at each other, but the framing aspect of the technique can be modelled as:

```
shot-reverse-shot{
  length: == 2
  size-relation: same (range: all)
  actor-relation: actor-different (range: all)
  framing-actor-num: == 1 (range: all)
}
```

The example in Figure 9 shows that directors frequently overlap FEPs to communicate complex emotions and meanings throughout the film. Table 1 summarizes these five FEPs.

Table 1. Summary of the 5 stylistic FEPs.

FEP	length	description	usage
Same-size	2+ shots	actors are the same-size across shots	dialogue, calm scenes
Intensify	3+ sub-sequences	a sequence of shots moving gradually closer	build emotion
Frameshare	2+ shots	actors in the same horizontal region	agreement, compassion
Opposition	2+ shots	actors in opposite horizontal regions	enimity, disagreement
Shot-Reverse-Shot	2 shots	two actors looking at each other in two shots	perception

4 FEP FOR FILM ANALYSIS

A first usage of FEP is film analysis on annotated film data. As an example, we can investigate how much the FEPs presented in the previous section are employed in cinematography. To do that, we have constructed a database of 22 clips, each of roughly 5 to 10 minutes in length, from 18 well-known movies, spanning different genres and a time frame between 1955 and 2012. The clips were usually the most memorable or famous sequences from each movie, based on YouTube searches. In total, our database contains 1018 annotated shots, and is publicly available at: https://github.com/husky-helen/FilmAnnotation. This data set was first published in [24] with detailed analysis on FEP related features in [25]. Here we provide a broad overview of what the database contains and how we have used FEPs to analyze these movie clips.

For each shot in the database, at least one framing was annotated, reporting:

- the frame number (from the film clip)
- list of all the actors in the framing
- the head, left eye, and right eye positions of each character in the framing

- other significant body parts, such as hand, foot, when the head and eye positions were not available
- non-animate objects crucial to the story
- the angle of the camera relative to the main character

The head was annotated in terms of its size, position, and azimuth angle with respect to the camera, while other elements were annotated for their on-screen x and y positions, respectively as a ratio of the width and height of the frame space.

We can then use FEPs as a mean to analyze film style. We analyzed the database for occurrences of four FEPs presented in the previous section: same-size, intensify, opposition, and frameshare. In this analysis, shot-reverse-shot was left out. Due to the difficulty in determining if two characters are indeed looking at each other (as explained in the previous section), the FEP's matches are not that meaningful in this context, since they cannot confirm nor indicate the psychological connection between two targets.

As defined before, each FEP has (i) a set of framing properties, e.g. `framing-size: MCU`, (ii) a set of shot relations, e.g. `size-relation: closer`, and (iii) a set of sub-sequence constraints, e.g. `sub-sequence: FEP`, and represents a query that a user can perform over an annotated film clip.

The solving process is an extension of the Knuth-Morris-Pratt algorithm [13], expressed as a search over a sequence **S** of annotated frames. Since in our annotated data, multiple keyframes are annotated in each shot, the first keyframe is selected to represent the shot in **S**. The search iterates through **S** and tries at each iteration to match the given FEP starting from frame i (where $1 < i < Card(S)$). The solver returns a set $\mathbf{R} = \{\mathbf{r}_1, .., \mathbf{r}_n\}$ of sub-sequences such that $\mathbf{r}_n = [f_I, f_F]$ where f_I and f_F represent respectively the starting and ending frames of the sequence that match the FEP.

For the sake of performance (i.e. avoiding re-evaluation of the satisfaction of framing properties, shot relations and sequence constraints), the search is run in two stages. The first stage builds a cache of valid solutions as three sets of frame sequences **FC**, **RC** and **SC**. The first represents the sets of frames that satisfy each of the framing properties mentioned in the FEP. The second represents the sets of successive frames $[f_i, f_{i+1}]$ that satisfy the shot relations, and the last represents the set of frame sequences $\mathbf{SC} = \{\mathbf{s}_1, .., \mathbf{s}_m\}$ where $\mathbf{s}_i = [f_I, f_F]$ and where f_I and f_F represent respectively the starting and ending frames that satisfy the specified sub-sequence.

The search algorithm (`FEPRecurse`) relies on the function `isValidFrame()` to evaluate whether the frame f_i is within the set of valid frames **FC**. The same process occurs with `isValidRelation()` and `isValidSubsequence()`. The function `isValidSequence()` simply checks that the given length of the sequence is valid (since all frames, relations and subsequences are valid).

Then, in a second stage, the process iterates over all frames f_i of the sequence **S** (see Algorithm 2). At each iteration a double recursive depth search is performed from the current frame with a simple idea: the next frame is retrieved from the sequence S, and if valid, is either considered as part of a subsequence (see line 4) or part of the sequence (see line 6).

The algorithm is then applied to our annotated database of clips. We found that all clips used at least one of the four FEPs, with more than half of the shots in each clip using one or more FEPs.

ALGORITHM 1: FEPSearch (FEP p, Sequence **S**)

1 ResultSet $\mathbf{R} = \emptyset$; **while** **S** *not empty* **do**
2 | $f_i = first(\mathbf{S})$; FEPRecurse(p, $\emptyset$, **S**, i, **R**); **S** = **S** $\setminus f_i$;
3 **end**
4 **return R**

ALGORITHM 2: FEPRecurse (FEP p, CurrentSequence **C**, Sequence **S**, Index s, ResultSet **R**)

```
if S not empty then
    f_i = first(S); if isValidFrame(f_i) AND isValidRelation(f_i) then
        if isValidSubsequence(p, C ∪ {f_i}) then
            FEPRecurse(p, C ∪ {f_i}, S \ {f_i}, s, R );
        end
        if isValidSequence(p, C ∪ {f_i}) then
            FEPRecurse(p, {f_i}, S \ {f_i} s, R ∪ {[s, i]} );
        end
    end
end
```

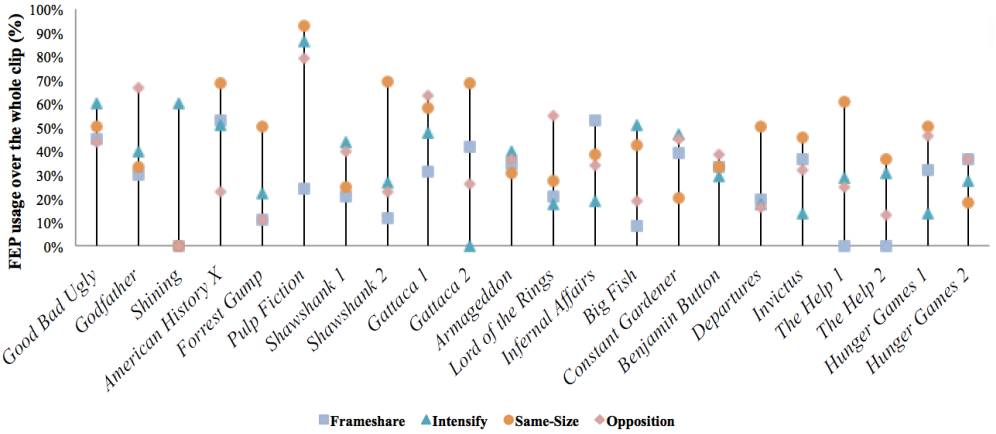

Fig. 10. The percentage of shots that use each of the FEPs throughout the whole clip.

This algorithm allows us to make quantitative observations on the evolution of average FEP lengths (in terms of the number of shots in the sequence) over these film clips such as in Figure 11 or the usage of these techniques over the entire dataset, as shown in Figure 10. On average, intensify and same-size sequences can be much longer than any other FEP, applied to more than 5 shots in the sequence, as seen in *The Good, the Bad, and the Ugly*, *American History X*, and *Pulp Fiction*.

In the clip from *Pulp Fiction*, three types of sequences (intensify, same-size, and opposition) also cover more than 80% of the shots in the clip, which clearly shows that all three techniques can be used in parallel throughout a whole scene.

From this analysis, we can see that FEPs are very frequently used by film directors, and our definition is flexible enough to successfully detect their occurrences in annotated film data.

5 FEP FOR INTERACTIVE EDITING

Having confirmed that FEPs are indeed widely used in movies, we then designed an editing application that assists the user in creating an edited sequence from a 3D animated scene, and allows them to apply FEPs. More specifically, the application allows the user to:

- cut the animation sequence into a number of shots;

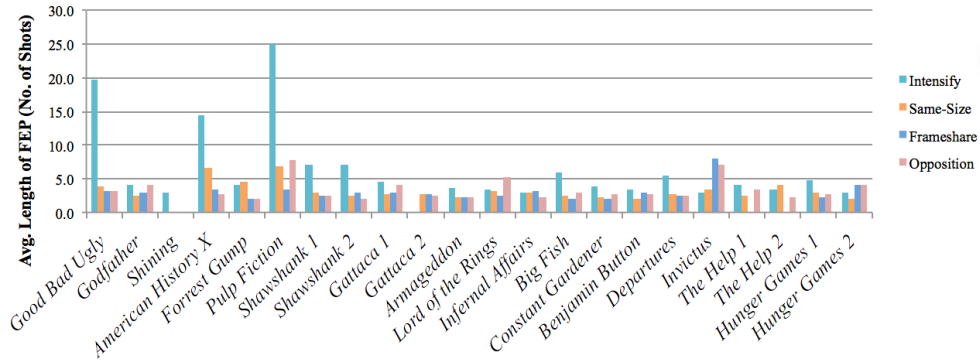

Fig. 11. Average length (in seconds) for all FEPs (embedded constraint pattern) used through shot sequences in the film clips.

- for each shot, select a framing (i.e., a camera) among a framing database, which is dynamically computed by the system on the basis of the 3D position of actors at the cut point (or at any time inside the shot, chosen by the user);
- apply a FEP to a selection of shots in the sequence, and as a result, reduce the choice of available framings to the ones that satisfy the FEP.

Any action of the user, for example selecting a certain framing for a shot, results in the system re-applying the FEPs to ensure that the resulting sequence always satisfies them.

The idea is that the user can creatively experiment with different editing choices by thinking in terms of stylistic choices, emotion building, and relations between actors, instead of worrying about lower level aspects such as camera position and basic continuity rules.

Figure 13 shows the interface of the system, developed in Unity, which resembles popular editing applications like Adobe Premiere of Final Cut Pro. For a demonstration of the application, please view the accompanying video.

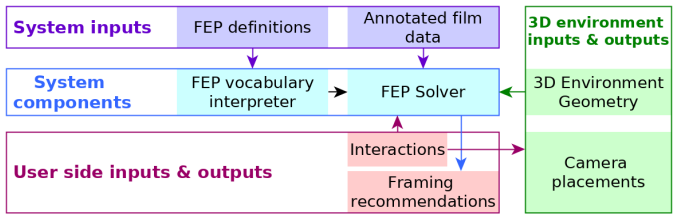

Fig. 12. Our system is composed of the interpreter for FEP vocabulary and the interactive solver. On the system side, the inputs are the definitions for the FEPs and also annotated film data. Based on interactions from the user, the solver then proposes framings for each shot in the sequence. The 3D environment geometry is analyzed by the solver, and framing recommendations are returned to the user via the application interface.

At an architectural level, the system is composed of:

- the interpreter for the FEP language;
- a database of film framings, which considers situations with one or two actors at all possible camera lengths and angles;

- a solver that: (i) computes actual framings based on the positions of actors in the scene at a certain instant; (ii) restricts the possible framings to the ones that respect the constraints defined in a specified FEP.

Figure 12 provides an overview of the architecture of our system. In the following section, we explain how the FEP solver works.

Fig. 13. The application we developed contains basic functions to edit a sequence and view the result in real time. Framings are proposed for each shot (upper right corner), dynamically calculated from our framing database and generally covering each actor in the scene, or their combination, at different sizes and angles. FEPs can be applied to a number of shots in the sequence in order to filter the framings to those that can fulfill the applied pattern. In the figure, an intensify pattern has been applied to three shots.

5.1 FEP Solver

The solver uses the Toric manifold method [15] which provides an algebraic computation of virtual camera positions based on a number of desired visual features, such as size of actors on screen, vantage angles, and on-screen position of one or two targets (e.g. two actors, or the left and right eye of the same actor). For example, we can compute a camera showing two actors, where one is on the left side of the frame, while the other is on the right side, and their heads are in the same vertical region. We refer the reader to [15] for details on the calculation.

In this paper, we concentrate on the other function of the solver: given a list of FEPs applied to specific shots, propose for each shot a selection of framings from the database that fulfill all the FEPs applied to the sequence. First, the solver initializes each shot with a list of candidate framings.

The main algorithm of the solver (Algorithm 3) is a constraint propagation method that maintains and updates an active list of FEP instances that are yet to be solved. Each FEP instance contains an FEP and an ordered list of shots $S_{[x,y]}$ to which the FEP is applied. Algorithm 3 iterates on the list of FEP instances, and incrementally filters the framing candidates for each shot in the FEP instance by calling Algorithm 4. Thus at each iteration, Algorithm 3 produces the subset of framing candidates that fulfills the considered FEP instance and all the preceding FEP instances. If Algorithm 4 removes framings in shots that overlap with other solved FEP instances, the constraints must be propagated by adding those affected instances back to the active list of FEP instances so that they can be re-solved. In this case, all overlapping FEP instances with p must be re-evaluated

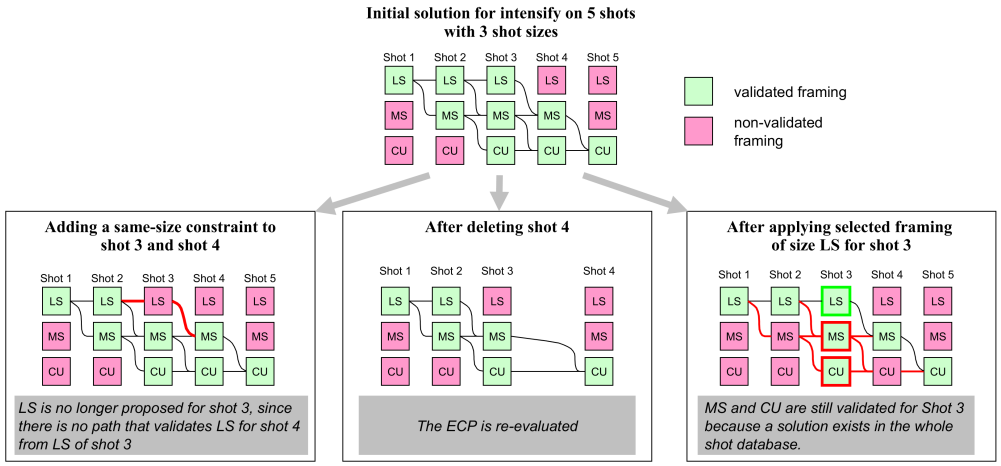

Fig. 14. The solver is called to calculate an initial solution and in response to a user's action. This figure shows how the solver filters the available framings for intensify on a five shot sequence, where each shot has the choice of a long (LS), medium (MS), and close-up shot (CU). In the initial solution, all framings that have a path from the first shot to the last are validated. Three types of interactions can follow the initial solution: (1) *applying another FEP*, same-size, to shots 3 and 4, the LS option is removed from shot 3 (2) *deleting shot 4 from the sequence*, the solver re-evaluates intensify for the remaining shots. (3) *selecting 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.

ALGORITHM 3: ConstraintPropagate (FEPInstanceList P)

```
1  FEPListCopy P'=P;
2  while P'.count!=0 do
3      FEPInstance e=P'.pop();
4      FEP p = e.FEP;
5      ShotSequence S = e.S[x,y];
6      FEPFilter(p,EmptySet,0,S);
7      forall FEPInstance ei ∈ P do
8          forall Shots s ∈ S do
9              if s∈ei.S[x,y] and ei∉P' then
10                 P'.add(ei);
11             end
12             break;
13         end
14     end
15 end
```

for each shot defined in $S_{[x,y]}$ to ensure that the changes made by Algorithm 4 still uphold for the other overlapping FEPs. Algorithm 3 iterates either until there are no more FEPs in the list, indicating all FEPs are solved, or until Algorithm 4 returns false, indicating that the combination of FEPs cannot be solved.

ALGORITHM 4: ReduceFramingSelection (FEP p, FramingSequence F, Position pos, ShotSequence S)

```
1  if pos_i=S.count then
2      forall Framings f ∈ S_pos.candidateFramings do
3          F.add(f);
4          ReduceFramingSelection(p,F,pos+1,S);
5          F.remove(f);
6      end
7  end
8  else if ValidateFEP(p, EmptySet, F) then
9      forall Framings f in F do
10         f.validate();
11     end
12 end
```

Algorithm 4 evaluates each FEP, and reduces the framings selection among the candidate framings for all shots $s_j \in S_{[x,y]}$. The actual validation of constraints defined in FEP p is carried out by Algorithm 5. We solve each FEP p by evaluating whether each frame in the whole sequence $S_{[x,y]}$ fulfills the framing and relation constraints, and whether the sequence can be split into a number of sub-sequences that fulfill the sub-sequence constraints (described in Section 3.3). Each possible framing f_i from the candidate framings must be either validated or rejected as a candidate for s_j in the considered FEP instance based on the following condition: if there exists a sequence of framings $f_x, f_{x+1}, \ldots f_i, \ldots f_y$ for each shot $S_{[x,y]}$ that fulfills the constraints set by the FEP (Algorithm 5), then f_i is validated, which means the framing should be available for selection. 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.

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. However, typically users work with 2-3 shots at a time, making the algorithm able to respond at interactive rates.

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

A new FEP instance is added: When a new FEP e is added to a number of selected shots, e is added to the FEPInstanceList P of Algorithm 3. If when solving e, the solver removes framing propositions from shots with another FEP instance e', then e' is added to the FEPInstanceList P. In this manner, the constraints of e are propagated to all other overlapping FEPs 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.

Adding/Removing of a cut or a shot: The shot s is added/removed from the range of all overlapping FEP instances, and all the FEP instances are pushed into the FEPInstanceList P in Algorithm 3 to be re-evaluated. If no solution exists for an FEP, the FEP is removed. Each time an FEP is removed or if the FEP removes proposed framings from a shot, all overlapping FEPs are pushed into the FEPInstanceList P to be re-evaluated.

A framing F_m is selected for a shot S_n: F_m is set as a new constraint and propagated to overlapping FEP instances. All framing proposals as well as selected framings for other shots must have a

ALGORITHM 5: ValidateFEP (FEP p, CurrentSequence **C**, Sequence **F**)

```
if F not empty then
    f_i = first(F);
    if isValidFrame(f_i) AND isValidRelation(f_i) then
        if isValidSubsequence(p, C ∪ {f_i}) then
            return ValidateFEP(p, C ∪ {f_i}, F \ {f_i});
        end
        else if isValidSequence(p, C ∪ {f_i}) then
            return ValidateFEP(p, {f_i}, F \ {f_i});
        end
        else
            return False;
        end
    end
end
if ValidateLength(C) then
    return True
else
    return
end
False
```

validated path through F_m for S_n, which means in Algorithm 4, 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, preventing the user from changing to another framing that can also be validated by the whole sequence. Instead, 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 FEPs and constraints. To accommodate this, we simply add an additional condition to Algorithm 5 to return true only when the FEP 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 5 will only return true when all other framings apart from the one at Position *pos* of Algorithm 4 must be a selected framing, if the user has selected one. Figure 14 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 FEPs on the new sequence.

6 USER EVALUATION

To assess the effectiveness and acceptability of using FEPs in editing tasks, we designed an experiment in which participants were asked to use our application to produce an edited sequence of a given animation. The considered animation is a 3D reconstruction of a scene of 80 seconds taken from the Robert Zemeckisfi movie *Back to the Future*, composed of four main characters (Marty, George, Goldie, Lou). The animated scene faithfully reproduced the movements of the characters, and included original audio from the movie, which was converted from stereo to mono, to avoid

any inconsistencies between audio cues (a character heard on the left) and visual cues (the same character framed on the right), hence enabling any editing choice.

There were a total of 17 participants: 6 experienced film professionals (4 of which had 10+ years of experience in filmmaking), and 11 amateur participants with only basic video editing experience. The participants' ages were between 20 and 46. Two of the film professionals had experience with 3D animation software, while most participants in the amateur group had some experience with 3D animation and graphics software. All participants had previously seen the *Back to the Future* movie, and could re-watch the selected clip before starting with the experiment.

6.1 Experimental procedure

First, participants were introduced to the experiment and were informally trained about FEPs by showing examples of their application in well-known movies. Then, the functioning of the system was briefly explained, and a pre-evaluation survey, to collect demographic and user experience information, was administered. We then showed them once the film original of the clip that they were to edit. This first part took about 15 minutes.

Then, participants were asked to produce their best edited sequence by starting from an initial editing of the *Back to the Future* clip comprised of *23* shots (approximately one every 4-5 seconds), where, for each shot, a random framing among the ones in the database was chosen. The users were asked to meet three criteria within the limitations of the tool:

(1) A framing must be selected for each shot
(2) the selected framing should express what is currently happening in the scene
(3) the overall edit should be coherent and aesthetically pleasing

Throughout the task, they could remove cuts, add extra cuts, select framings for each shot, and apply or remove FEPs. There was no time limit set for the task. Participants could choose whether or not to use FEPs for the task.

After the task, a post-test questionnaire was filled to gather feedback on the application, and on the usefulness of FEPs.

6.2 Results

Some edited sequences produced by participants can be seen in the accompanying video. On average, users completed their task in about half an hour, with a maximum time of about 40 minutes. The sequences produced by the participants varied greatly from the original film clip, generally with shorter shot lengths, and a larger variety of framing choices.

In the post-task questionnaire, we asked participants to rank on a Likert scale of 1 (strongly disagree) to 5 (strongly agree) the following statements:

- A. Concerning the application: (1) I'm satisfied with the output (2) The tool is easy to use (3) It's easy to predict the effect of commands (4) The tool is effective in creating sequences.
- B. Concerning FEPs: (5) It's easy to work with FEPs (6) FEPs did not limit creativity (7) FEPs produced good results (8) I understand how to apply FEPs (9) FEPs helped convey emotions.

The overall impression of the tool was mostly positive. Figure 15 and 16 summarize the results of the post-survey questionnaire. One film professional chose not to use the FEPs and felt that he was thus unable to evaluate statements (5) to (9). Table 2 shows how many times each FEP was applied in total for each group, and in parentheses, the number of people from the group that used the FEP.

The amateur group was more enthusiastic in terms of applying FEPs, each person using an average of 4.6 FEPs for a sequence of 23 shots. Though two participants commented on the difficulty to understand how to apply the FEPs, the general feedback for this group was positive, appreciating

Table 2. Number of times each FEP was used by each group (and in parentheses, the number of participants in the group that used the FEP).

FEP	Professionals	Amateurs
Same-size	1 (1)	6 (5)
Intensify	0 (0)	13 (8)
Frameshare	0 (0)	8 (5)
Opposition	2 (2)	18 (5)
Shot-Reverse-Shot	4 (2)	6 (4)

the pedagogical aspect of FEPs, and mentioning that they influenced the framings they chose and allowed them to explore ideas they never had before. It was noted that a large number of participants did not find it easy to work with FEPs. This could be expected, as the concept of film cinematography language would be relatively new to them. Nevertheless, as shown in the accompanying demo video, we found that with the assistance of the application, amateur video editors were able to make use of FEPs to create sequences that had similar shots as professionals, but with their own distinct styles. Users with some professional 3D animation and graphics backgrounds found the interface accessible and easy to use, and were moderately pleased with the tool, though many recommended adding more advanced functions to edit and create their own framings.

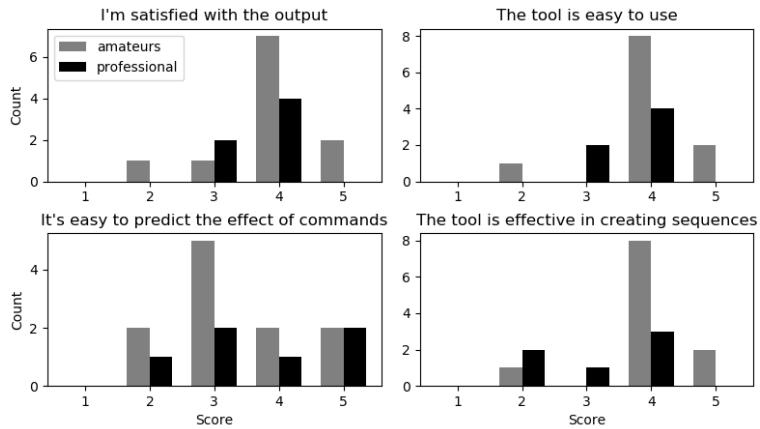

Fig. 15. Users general impressions on the tool (easy of use, effectiveness) and on the perceived quality of their produced sequence.

Opinions of professional filmmakers were more polarized. Most of them felt that the editing functions were limited, and the provided FEPs too rigid for professional use, and as a result, they relied on them much less than the amateurs (Table 2). The feedback was within our expectations, as professional users with sufficient film knowledge would probably already have a desired edit to the sequence without aid from the system, and as the system was designed to be as simple as possible, it provides limited editing functions for fine-tuning each shot. Despite this, all the participants in this group reacted warmly towards the idea of incorporating film knowledge as FEPs, and the ease of creating an acceptable edit. In the post-task survey, one film director reflected positively that the FEPs *"represent some of the most used cinematographic conventions since the beginning of the story of editing."* It was also mentioned multiple times how easy it was to make a edit and the experimental

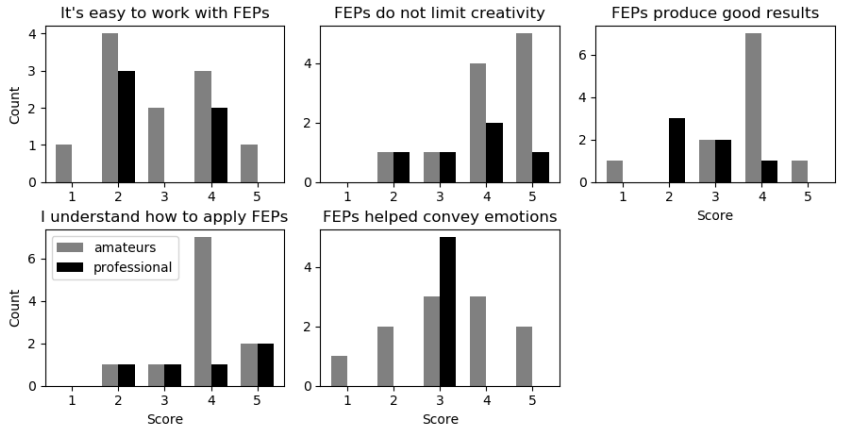

Fig. 16. Users general impressions on FEPs.

aspect of FEPs, which would especially benefit novices of film editing. Their response validated that FEPs is a good representation of film knowledge for the purpose of assisted creativity, and that FEPs can express common film idioms and editing techniques that these professionals frequently use in their films.

Indeed, the number and types of FEPs provided was probably too limited to cover all user styles and preferences. This suggests that an application should provide users with the possibility of designing their own FEPs, or maybe even automatically detect them from user-provided examples.

7 LIMITATIONS AND FUTURE WORK

The Film Editing Patterns language provides a simple way to define complex editing constraints for automatic detection and enforcement of cinematographic patterns. However, the visual features currently available in the vocabulary are limited to the size, position, angle and number of actors on-screen, and there are many other features that are essential to cinematographic storytelling, including lighting, sound, and staging, that we currently do not take into account when designing patterns. With respect to camerawork, another notable limitation is the absence of properties and relations that refer to camera movement. While, for previsualization tasks and rough edits, camera movement might not be fundamental, we plan to overcome this limitation by expanding the FEP vocabulary with common movement properties, and incorporating the idea of camera path keyframes in the framing database that can be implemented by various camera movement algorithms.

Our interactive application was designed with ease of use and familiarity of the film language in mind. As a prerequisite though, the application requires pre-designed 3D animations and assets, which may be hard to come by. However, given the growing availability of 3D content in asset stores, this seems less and less of a concern.

The editing patterns presented in this paper are just examples that mainly come from film textbooks. We foresee that our techniques can be improved and extended greatly with the collection of data through video processing techniques, and learning techniques to discover other common editing patterns that directors use in films. By releasing both our dataset and the animated scene, we hope that easily accessible resources for this application can be expanded on in the future.

Technically, extra FEPs and framings can be easily achieved by modifying the XML files that define our framing and FEP database.

In our evaluation, we targeted a diverse user group incorporating people of various levels of editing experience. In the post-task questionnaire, we found that our tool is best targeted towards people with mid- to low-professional experience in filmmaking: students of film, amateur filmmakers, 3D animators. Our professional users, though the response towards the tool was mostly positive, commented also on the restrictions of the tool, notably the inability to design their own camera positions, or design different FEPs. Others suggestions mentioned elements non-related to the editing (e.g. audio, animation, staging). The addition of non-editing features will require closer future collaboration with film educators and practitioners to tailor the tool to their specific needs.

8 CONCLUSION

In this paper, we have proposed the concept of *Film Editing Patterns*, which are evolving elements of visual style over (long) sequences of shots. We have shown how the language and design of FEPs is linked to actual film theory and studies both by detecting the film editing patterns in annotated data, and by creating an editing application that uses them for smart editing guidance. Our application was evaluated by both film professionals and amateurs. The evaluation has provided encouraging results, especially for non-expert users.

REFERENCES

[1] 1998. Structured Representation and Automatic Indexing of Movie Information Content. *Pattern Recognition* 31, 12 (1998), 2027–2045.

[2] Dan Amerson and Shaun Kime. 2005. Real-time cinematic camera control for interactive narratives. In *ACM SIGCHI International Conference on Advances in computer entertainment technology*. ACM Press, 369–369.

[3] William H. Bares, Joël P. Grégoire, and James C. Lester. 1998. Realtime constraint-based cinematography for complex interactive 3D worlds. In *The National Conference On Artificial Intelligence*. Citeseer, 1101–1106.

[4] William H Bares, Somying Thainimit, and Scott Mcdermott. 2000. A Model for Constraint-Based Camera Planning. In *AAAI Spring Symposium*. Stanford.

[5] David B. Christianson, Sean E. Anderson, Li-wei He, David H. Salesin, Daniel S. Weld, and Michael F. Cohen. 1996. Declarative camera control for automatic cinematography. *AAAI Conference on Artificial Intelligence* (1996).

[6] Nicolas Davis, Alexander Zook, Brian O'Neill, Brandon Headrick, Mark Riedl, Ashton Grosz, and Nitsche Michael. 2013. Creativity support for novice digital filmmaking. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2013), 651–660.

[7] Steven M. Drucker and David Zeltzer. 1994. Intelligent camera control in a virtual environment. In *Graphics Interface fi94*. 190–199.

[8] David K. Elson and Mark O. Riedl. 2007. A Lightweight Intelligent Virtual Cinematography System for Machinima Production. In *3rd Conference on Artificial Intelligence and Interactive Digital Entertainment*. Palo Alto, California, USA.

[9] Quentin Galvane, Marc Christie, Rémi Ronfard, Chen-Kim Lim, and Marie-Paule Cani. 2013. Steering Behaviors for Autonomous Cameras. *Proceedings of Motion on Games - MIG '13* (2013), 93–102.

[10] Quentin Galvane, Rémi Ronfard, Christophe Lino, and Marc Christie. 2015. Continuity Editing for 3D Animation. In *AAAI Conference on Artificial Intelligence (AAAI Press)*. Austin, Texas, United States.

[11] Li-Wei He, Michael F. Cohen, and David H. Salesin. 1996. The virtual cinematographer: a paradigm for automatic real-time camera control and directing. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM Press, 217–224. https://doi.org/10.1145/237170.237259

[12] Arnav Jhala and R. Michael Young. 2010. Cinematic Visual Discourse : Representation , Generation , and Evaluation. *IEEE Transactions on Computational Intelligence and AI in Games* 2, 2 (2010), 69–81.

[13] Donald E. Knuth, James H. Morris, and Vaughan R. Pratt. 1977. Fast Pattern Matching in Strings. In *SIAM Journal on Computing*. 323–350.

[14] Mackenzie Leake, Abe Davis, Anh Truong, and Maneesh Agrawala. 2017. Computational Video Editing for Dialogue-Driven Scenes. *ACM Transactions on Graphics* 36, 4 (2017). https://doi.org/10.1145/3072959.3073653

[15] Christophe Lino and Marc Christie. 2015. Intuitive and Efficient Camera Control with the Toric Space. *Transactions on Graphics* 34, 4 (2015).

[16] Christophe Lino, Marc Christie, Fabrice Lamarche, G Schofield, and Patrick Olivier. 2010. A Real-time Cinematography System for Interactive 3D Environments. In *2010 ACM SIGGRAPH Eurographics Symposium on Computer Animation*. 139–148.

[17] Christophe Lino, Marc Christie, Roberto Ranon, and William H Bares. 2011. The director's lens: an intelligent assistant for virtual cinematography. In *19th ACM International Conference on Multimedia*. 323–332.

[18] Zeeshan Rasheed, Yaser Sheikh, and Mubarak Shah. 2005. On the use of computable features for film classification. *IEEE Transactions on Circuits and Systems for Video Technology* 15, 1 (2005), 52–63.

[19] Rémi Ronfard, Gandhi Vineet, and Laurent Boiron. 2013. The Prose Storyboard Language. In *AAAI Workshop on Intelligent Cinematography and Editing*.

[20] M. Svanera, S. Benini, N. Adami, R. Leonardi, and A. B. Kovcs. 2015. Over-the-shoulder shot detection in art films. In *International Workshop on Content-Based Multimedia Indexing*, Vol. 2015-July.

[21] Wallapak Tavanapong and Junyu Zhou. 2004. Shot Clustering Techniques for Story Browsing. *IEEE Transactions on Multimedia* 6(4) (2004), 517–527.

[22] Roy Thompson and Christopher J Bowen. 2009. *Grammar of the Shot*.

[23] Jihua Wang and Tat-Seng Chua. 2003. A cinematic-based framework for scene boundary detection in video. *The Visual Computer* 19, 5 (2003), 329–341.

[24] Hui-Yin Wu and Marc Christie. 2016. Analysing Cinematography with Embedded Constrained Patterns. In *Proceedings of 2016 Eurographics Workshop on Intelligent Cinematography and Editing*.

[25] Hui-Yin Wu, Quentin Galvane, Christophe Lino, and Marc Christie. 2017. Analyzing Elements of Style in Annotated Film Clips. In *Proceedings of 2017 Eurographics Workshop on Intelligent Cinematography and Editing*. Lyon, France, 7. https://doi.org/10.2312/wiced.20171068

[26] Herbert Zettl. 2007. *Sight, sound, motion: Applied media aesthetics*. Wadsworth Publishing Company.