Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Continuity Editing for 3D Animation

Quentin Galvane
INRIA
Univ. Grenoble Alpes & LJK

Rémi Ronfard
INRIA
Univ. Grenoble Alpes & LJK

Christophe Lino
INRIA
Univ. Grenoble Alpes & LJK

Marc Christie
IRISA
University of Rennes I

Abstract

We describe an optimization-based approach for automatically creating well-edited movies from a 3D animation. While previous work has mostly focused on the problem of placing cameras to produce nice-looking views of the action, the problem of cutting and pasting shots from all available cameras has never been addressed extensively. In this paper, we review the main causes of editing errors in literature and propose an editing model relying on a minimization of such errors. We make a plausible semi-Markov assumption, resulting in a dynamic programming solution which is computationally efficient. We also show that our method can generate movies with different editing rhythms and validate the results through a user study. Combined with state-of-the-art cinematography, our approach therefore promises to significantly extend the expressiveness and naturalness of virtual movie-making.

Introduction

The wide availability of high-resolution 3D models and the facility to create new geometrical and animated contents, using low-cost input devices, opens to many the possibility of becoming digital 3D storytellers. A convincing illustration is the rise of the machinima[1] community, consisting in creating movies by relying on contents and rendering engines borrowed from computer games through ad-hoc techniques or dedicated modeling tools[2].

To date there is however a clear lack of accessible tools to easily create the cinematography (positioning the cameras to create shots) and perform the editing of such stories (selecting appropriate cuts between the shots created by the cameras). Creating a movie requires the knowledge of a significant amount of empirical rules and established conventions. In particular *continuity editing* – the creation of a sequence of shots ensuring visual continuity – is a complex endeavor. Most 3D animation packages lack continuity editing tools, calling the need for automatic approaches that would, at least partially, support users in their creative process (Davis et al. 2013).

Previous contributions in automatic film editing have focused on generative methods mixing artificial intelligence and computer graphics techniques. However, evaluating the quality of film editing (whether generated by machines or by artists) is a notoriously difficult problem (Lino et al. 2014). Some contributions mention heuristics for choosing between multiple editing solutions without further details (Christianson et al. 1996) while other minimize a cost function which is insufficiently described to be reproduced (Elson and Riedl 2007). Furthermore, the precise timing of cuts has not been addressed, nor the problem of controling the rhythm of cutting (number of shots per minute) and its role in establishing film tempo (Adams, Dorai, and Venkatesh 2002). Most approaches yield a reactive style of editing known by professional editors as the *dragnet* style (Murch 1986), which mechanically cuts to new speakers or actions.

In this paper, we propose a continuity editing model for 3D animations that provides a general solution to the automated creation of cinematographic sequences. Our model encodes the continuity editing process as a search for the optimal path through an *editing graph*. In this *editing graph*, a node represents a time-step (a temporal fragment of a shot), and an arc represents a transition between two cameras, going from a camera to either the same camera (no cut) or another camera (cut).

Our optimization uses dynamic programming to minimize, under a semi-Markovian hypothesis, the errors made along three criteria (see Figure 1): the quality of the shots (with respect to the unfolding actions), the respect of continuity editing rules and the respect of a well-founded model of rhythm (cutting pace). Semi-Markov models (Murphy 2002; Yu 2010) have been used before in the context of information extraction (Sarawagi and Cohen 2004), speech generation (Zen et al. 2007) and computer vision (Shi et al. 2008). To the best of our knowledge, this is the first time they are suggested as a computational model for film editing.

Our contributions are: (i) a detailed formalization of continuity editing for 3D animation, encompassing a thorough number of visual properties and continuity rules (ii) an optimal algorithm for automatic editing in which parameters such as pacing can be controlled, thereby significantly increasing the expressiveness of editing tools, and (iii) a vali-

[1] contraction of machine and cinema

[2] see iClone http://www.reallusion.com/iclone/ or Moviestorm http://www.moviestorm.co.uk/ tools

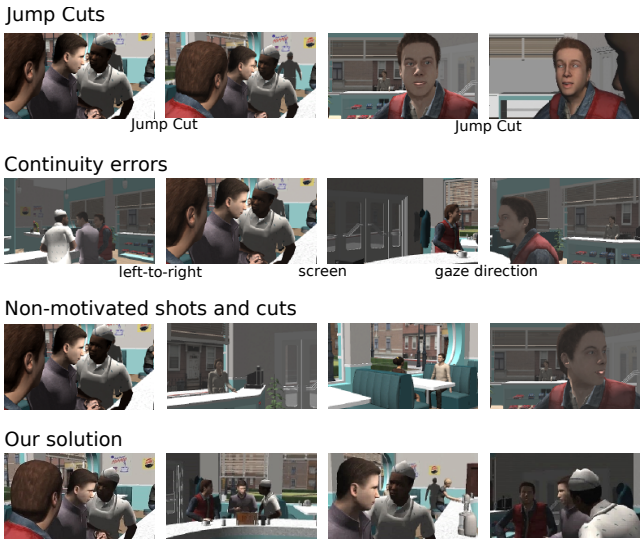


Figure 1: *Editing errors in a short sequence of a movie. From top to bottom: jump cut errors; breaking the continuity errors; non-motivated shots and cuts. The last edit is the output of our system.*

dation of our model through a user evaluation comparing the original edit of an existing movie with our optimal edit and with degraded approaches.

Related work

The problem of generating movies from 3-D animation can be decomposed into the problems of (i) choosing shots and placing cameras (the cinematography problem), and (ii) deciding when to assemble those shots into a movie (the film editing problem). In this section, we review previous work on the film editing problem chronologically.

(Christianson et al. 1996) introduce the declarative camera control language (DCCL) as a general framework for generating idiom-based solutions to cinematography and film editing problems. Film idioms (He, Cohen, and Salesin 1996) are recipes for obtaining good cinematography and editing in any given situation, similar to cases in case-based reasoning. As a result, DCCL uses a conversation idiom for filming conversations, a fighting idiom for filming fights, etc. DCCL then builds a film tree, as a result of filming each scene in all available idioms, and includes a heuristic evaluator for checking the visual quality of sequences, choosing the one with the highest quality. The evaluation criteria are only briefly mentioned, and in comparison, we offer a full description of a scoring function that can be used to find optimal editing solutions using all available cameras in a principled and reproducible fashion.

(Tomlinson, Blumberg, and Nain 2000) propose a fully automatic method for generating expressive cinematography. Their system uses autonomous cameras with emotional states and goals, that choose between a number of visual targets – such as a specific character, or two characters interacting with each other, or one character interacting with its environment. Though their system interestingly motivates the shots both emotionally and visually, the rules of montage (editing) are not enforced, and the pacing of the shots is resolved by ad hoc rules. (Kennedy and Mercer 2002) directly address the problem of planning a complete sequence of shots for a given sequence of actions which must be known in advance. Users can choose between fast or slow editing in two different styles. Authors use a depth-first forward chained planner which can only evaluate a small number of possible plans and enforce a single continuity editing rule (”not crossing the line of action”).

Unlike previous work, Darshak (Jhala and Young 2011) takes as input an explicit declarative representation of the story goals, including all actions, causal links between actions, and ordering constraints. Story goals are defined in terms of a small number of dramatic situations, and a hierarchical partial-order causal link planner computes a camera plan, consisting of a sequence of shots that achieves the desired story goals. Darshak goes a long way into motivating the shots, but the actual cinematography and editing are not evaluated. Cambot (Elson and Riedl 2007) is a movie-making system where the choice of shots is found as a solution to an optimization problem using dynamic programming. The scene is expressed as a sequence of non-overlapping *dramatic beats* and their approach evaluates different placement of characters (blockings) and camera choices for each beat. Though we also make use of dynamic programming, our method is very different from (Elson and Riedl 2007). Firstly, we search a much larger set of possible solutions, by evaluating a higher number of shot transitions at a finer level of granularity (every frame, rather than every beat). As a result, our method makes it possible to choose the precise cutting point between shots, and to maintain control over the rate of shot changes (cutting pace). Secondly, we describe in detail a cost function which implements an extensive set of film editing rules. Our cost function has a semi-Markov property which allows to efficiently optimize over this larger solution space. Thirdly, our method does not require the scene to be decomposed into a linear sequence of beats, and works equally well with overlapping actions and dialogues.

(Assa, Wolf, and Cohen-Or 2010) propose a fully automated editing process that performs cuts in real-time between multiple cameras viewing human motions. The ranking between shots is computed by measuring for each camera the correlation between human motions in the 3D scene, and the on-screen 2D projection of these motions (the larger the correlation, the better the shot). A measure of erosion of the current view is employed to motivate the cuts between the viewpoints, while enforcing continuity rules (jump-cut and crossing the line of action). In a similar context of real-time cinematography, (Lino et al. 2010) propose a system that automatically computes for each frame a collection of distinct viewpoints through spatial partitions. The editing process is then performed by encoding continuity rules as filtering operators that remove inconsistent viewpoints.

(Markowitz et al. 2011) extend the film idiom approach by replacing finite state machines with behavior trees. They enforce the 180 degree rule and the 30 degree rule, assuming a single line of interest at any given time in the scene. Simi-

lar to film idioms, their approach remains reactive, resulting in an editing style that immediately follows the action and cannot easily control the pacing or style of the editing.

System overview

In this paper, we cast the problem of film editing as an optimization problem over a space of semi-Markov chains. Our system takes as input a 3D animation scene, comprising a flow of world events, and a set of rushes taken from different cameras and covering the whole scene. We then rank possible edits on three key aspects: (i) how much shots convey unfolding actions, (ii) how much continuity editing rules are enforced and (iii) how much an input cutting rhythm is respected.

Given a 3D animated scene with arbitrarily complex actions a and a choice of rushes (*i.e.* unedited footage) from M cameras, a semi-Markov chain is a sequence of states (shots) s_j with durations d_j, chosen according to a probability model over s_j, d_j and a. The probability of the next shot s_{j+1} of duration d_{j+1} and starting at time t_{j+1} depends only on the previous shot s_j and the actions a in the segment $[t_{j+1}, t_{j+1} + d_{j+1}]$.

We here introduce the notion of *editing graph*, the graph of all possible shots and transitions. In this graph, a node represents a time-step (one frame) of a rush and an arc represents a transition from frame i of a rush (camera) to frame $i + 1$ of a rush (same or different). The output of our system is then a full edit list of the scene, computed as the continuous path through our editing graph minimizing the errors on these three key aspects.

The first input of our system is an ordered list of durative world events (that we refer to as *actions*), expressed in the form $(subject, verb, object)$. In the following, we consider that the $subject$ and $object$ of all actions are characters, and we refer to the set of all characters as $\mathcal{C}$. We use four main categories of actions: *speaking* actions performed by the character's mouth, *reacting* actions performed by the character's eyes, *manipulating* actions performed by the character's hands, and *moving* actions performed by the character's feet. As a result, a character can be the subject of at most four different actions at any given time and the object of an unlimited number of actions.

The second input of our system is a list of M rushes from different cameras filming the scene for a total duration of N video frames. We are agnostic about how the rushes are obtained. Though in this paper a number of cameras were placed manually by a human expert, they could equally be computed by an automatic camera planner such as (Drucker and Zeltzer 1995; He, Cohen, and Salesin 1996; Markowitz et al. 2011; Elson and Riedl 2007) or any other method.

The output of our system is a movie, described as an edit decision list (EDL) defined as a sequence of shots s_j in the form of triplets (r_j, t_j, d_j). Note that in this paper, we only consider chronological EDLs where time is preserved ($t_{j+1} = t_j + d_j$). In this limited context, the EDL can be reconstituted using the rush selection function $r(t)$ which gives the rush index as a function of time.

We here propose to cast the editing process into a mathematical model accounting for three criteria: (i) how much shots convey unfolding actions, (ii) the continuity editing principles and (iii) the cutting rhythm of the movie. To do so, we use a log-linear model where the probability of choosing a particular sequence of shots s_j is taken to be the exponential of a linear cost function $C(s, a)$. The cost function $C(s, a)$ is further decomposed into three terms which separately measure (i) errors in conveying unfolding actions in each shot, (ii) violations of continuity editing rules in each cut and (iii) errors in choosing shot durations.

$$C(s,a) = \sum_j \sum_{t_j \le t \le t_j + d_j} C^A(r_j, t) + \sum_{1 \le j} C^T(r_{j-1}, r_j, t_j) + \sum_j C^R(d_j)$$

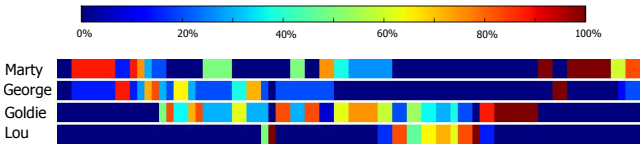
In this equation, the first term is a sum over all frames of all shots of a cost function C^A related to actions in the scene. The second term is a sum over all cuts of a cost function C^T related to transitions between shots. Those two term are further decomposed into weighted sums of features, *i.e.* $C^A(r_j, t) = \sum_k w_k^A C_k^A(r_j, t)$ and $C^T(r_i, r_j, t) = \sum_k w_k^T C_k^T(r_i, r_j, t)$. The third term is a sum over all shots of a cost function C^R related to editing rhythm. In the following sections, we explain each of those terms, and refer to the *Appendix* for mathematical details.

Symbolic projection

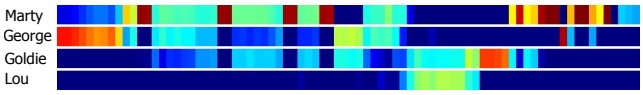
During the shooting of a virtual scene, all cameras capture images which are perspective projections of the world scene into their own frame of reference. In parallel to the computation performed by the graphics pipeline, we perform a symbolic projection of the actions which keeps a record of *how much of the action is visible* in each rush at every single frame.

This is performed as follows. First, each action is decomposed into its constituents – verb, subject and object. Based on the verb category (speaking, reacting, moving or manipulating), we then compute the bounding boxes of involved body parts (*e.g.* mouth, eyes, hands and feet) of the subject and object characters. We then compute the screen size of their projection in each frame of a rush. 0 Second, to evaluate how much of these actions are visible, we compute the visible and occluded areas of characters. To do so, for each face f of each body part b of a character, we compute its projected size (or area) $S(f, r, t)$ at time t in rush r. This projected size is measured relatively to the screen size, and comprises both the on-screen and off-screen projections of f. We then define the visible and occluded sizes of f as follows. Its occluded size $O(f, r, t)$ corresponds to the cumulative size of its areas that are either occluded or appear off-screen, while its visible size $V(f, r, t)$ is the complementary value computed such that $S(f, r, t) = V(f, r, t) + O(f, r, t)$. We finally define the projected size and the visible size of each character c as the sum of corresponding values on each face of its body parts: $V(c, r, t) = \sum_{b \in c} \sum_{f \in b} V(f, r, t)$

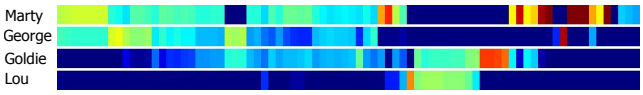
This method is further easily extended to the case of non-character objects (we use their bounding boxes) and multiple-characters.



(a) Narrative importance per character per frame



(b) On-screen character sizes in Zemeckis movie



(c) On-screen character sizes in our solution

Figure 2: *The relative narrative importance of a character is computed from all actions where it plays a role. It correlates with the screen size of the characters in both the Zemekis movie (linear correlation coefficient of 0.56) and our solution (0.73).*

Narrative importance

To each action a holding at time t, a narrative importance $I(a)$ is attributed depending on whether a is a foreground or a background action. We then firstly distribute the importance according to the different roles in the action (subject and object). Note that this defines the narrative importance of each character $I(c,t)$ at time t. We secondly distribute the importance given to each role proportionally into the individual body parts of the character filling the role. For instance, in a speak action the subject's and object's visual targets are their heads; in a manipulate action the subject's visual target is its head and chest; in a move action, the subject's visual target is its full body. We finally non-uniformly distribute the importance of a body part on its front, back and side faces to obtain the narrative importance of each face $I(f,t)$ of a character at time t.

Shot selection

Based on the symbolic and geometric data related to unfolding actions of a given frame, we rank every frame of every shot on three aspects: the action visibility, the action proximity and the action ordering (also known as the Hitchcock principle).

The action visibility term evaluates how much of unfolding actions is visible. To fully satisfy this criteria, each important body part of a character taking part in an unfolding action should be on-screen and fully visible. The cost associated to action visibility is computed as the sum, on each face f of each body part b of each character c, of the occluded proportion of the face weighted by its narrative importance:

$$C_V^A(r,t) = \sum_{c\in\mathcal{C}} \sum_{b\in c} \sum_{f\in b} I(f,t) \cdot \frac{O(f,r,t)}{S(f,r,t)}$$

The action proximity term evaluates how immersed the camera is in the unfolding actions, *i.e.* how much the screen is filled by actions. The cost (or penalty) associated to poor action proximity is then given by the proportion of the screen filled by the characters:

$$C_P^A(r,t) = 1 - \sum_c V(c,r,t)$$

The action ordering term evaluates how much the on-screen importance of a character matches its narrative importance. This is also known as the Hitchcock principle, which states that the size of a character should be proportional to its narrative importance in the story (Truffaut and Scott 1967; Hawkins 2005; DeLoura 2009). Our implementation considers all characters present in the scene, not just the characters present in each shot, or the characters participating in the main action. This has the benefit to easily rule out prominent shots of unimportant characters and favor prominent shots of important characters, focusing on its important body parts (mouth while speaking, eyes while reacting, hands while manipulating, feet while moving). The cost associated to the Hitchcock principle is computed as the sum of all deviations of the on-screen visibility of a character compared to its narrative importance:

$$C_H^A(r,t) = \sum_{c\in\mathcal{C}} \left| \frac{I(c,t)}{\sum_{c'} I(c',t)} - \frac{V(c,r,t)}{\sum_{c'} V(c',r,t)} \right|$$

Continuity editing

We rank all possible transitions between rushes based on a computational model of continuity-editing. Continuity-editing is the most commonly used editing style in filmmaking. It relies on a set of well established rules: avoiding jump cuts, enforcing the screen, motion and gaze continuity and maintaining the left-to-right ordering of on-screen characters (Dmytryk 1984; Thompson and Bowen 2009).

Jump cuts When the same character appears in two consecutive shots (*i.e.* before and after a cut), there must be sufficient change in either its apparent size or its profile angle in the two shots, so that the audience perceives the cut as a change in viewpoint, rather than a sudden change in the character's pose, also known as a *jump cut*. To prevent such cuts, we penalize such errors through a cost function over all characters appearing in successive shots, whose formula is detailed in Section *Jump cuts* of the appendix.

Screen, motion and gaze continuity The main goal of continuity editing is to enforce continuity on screen positions, motion directions and gaze directions of all characters across the cut. To prevent discontinuities, we penalize them through a non-linear combination of the differences in screen position, gaze direction and motion direction of all characters appearing in successive shots. The formulas of these cost functions are detailed in Sections *Screen continuity*, *Motion continuity* and *Gaze continuity* of the appendix.

Left-to-right ordering The left-to-right ordering of characters is another important factor to enforce visual continuity. Characters whose relative screen positions are reversed after a cut appear to be jumping around, which attracts attention to the cut (Smith 2005) – this criteria is also known as the 180 degree rule. For every pair of character (c, c') appearing on-screen before and after the cut, we then penalize the reversion (cost=1) iff their oriented difference of on-screen position $\Delta P_x(c, c')$ (taken on the x axis) is of opposite sign. The overall cost on left-to-right continuity is then computed as the sum of this penalty on all pairs of character appearing on-screen before and after the cut. The formula of this cost function is detailed in Section *Left-to-right ordering* of the appendix.

Cutting rhythm

Cutting rhythm is an important element of film editing style (Bordwell 1998). Cutting between cameras produces visual rhythm. Fast cutting as in an action scene can change cameras as often as every half second. The cutting rhythm has been studied extensively by film scholars (Salt 2009; Cutting, DeLong, and Nothelfer 2010) who have shown that it is well approximated with a time-varying log-normal distribution of shot durations (Limpert, Stahel, and Abbt 2001). Parameters of the log-normal distribution are the mean μ and standard deviation σ of the log-transformed durations $\log d_j$, which result in a skewed distribution of durations with average shot length $ASL = \exp\left(\mu + \frac{\sigma}{2}\right)$ and variance $Var = \exp\left(2\mu + \sigma^2\right)\left(\exp\sigma^2 - 1\right)$. Rather than making automatic decisions, our system is designed to let the user/director choose the average shot length (ASL) which dictates the rhythm of editing, and hence the editing style. To enforce those values, we compute a cost measuring, for each shot s_j of duration d_j, the deviation of its duration from the log-normal distribution

$$C^R(d_j) = \frac{(\log d_j - \mu)^2}{2\sigma^2} + \log d_j$$

Optimization-based film editing

We evaluate the cost of an arbitrary edit decision list of shots s_j with a weighted sum of simple feature functions. To do so, we use a dynamic programming approach to find the minimum cost solution by storing partial solutions (Mitchell, Harper, and Jamieson 1995; Murphy 2002).

We define $B(r, t)$ to be the cost of the best sequence of shots ending at time t with a shot using rush r. One important result that follows from our choice of cost functions is the following recurrence relation

$$B(r,t) = \min_{\substack{t_0 < t \\ r_0 \neq r}} \Big[B(r_0, t_0) + C^T(r_0, r, t_0 + 1) + \sum_{t'=t_0+1}^{t} C^S(r, t') + C^R(t - t_0) \Big]$$

In plain words, the best sequence ending on rush r at time (frame) t can be computed by comparing all combinations of a shot ending on rush r_0 at time $t_0 < t$, followed by a cut from rush r_0 to rush r between frames t_0 and t_0+1, followed by a shot of duration $t - t_0$ using rush r (see Figure 3).

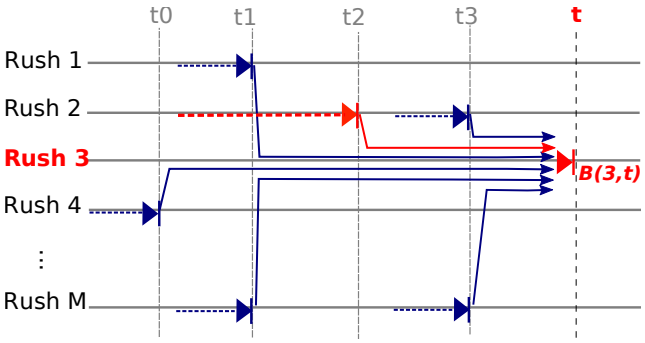


Figure 3: *Illustration of our dynamic programming algorithm, using semi-Markov decisions. The best edit in which a shot ends in Rush 3 at time t is computed as the best combination (drawn in red) of an edit in which a shot ends in Rush $r_0 \neq 3$ at a prior time $t_i < t$ then a shot using Rush 3 between t_i and t.*

Using the above recurrence, we build the table of $B(r, t)$ in time M^2N^2 and space MN, where M is the number of rushes and N is the number of video frames. We additionally store a back-pointer to retrieve the actual path yielding the minimum-cost solution. Note that the method can also be used to retrieve sub-optimal solutions ending in other cameras. This can be useful in cases where the final shot is chosen by the director. To improve our system efficiency, we further restrain the search in our algorithm to a constant horizon H (longest allowable shot duration) of 30 seconds. This leads to compute a solution in time M^2NH (Mitchell, Harper, and Jamieson 1995; Murphy 2002), which is sufficient for our purpose. For longer sequences with more rushes, an algorithm by (Datta, Hu, and Ray 2008) can be used to compute an exact and unbounded solution in time $MN(M + N)$.

Experimental results

To validate our approach, we have recreated the animation of a well-known scene from Robert Zemeckis' movie "Back to the future". The average shot length (ASL) in the original version is 6.6 seconds. The distribution of shot lengths in the original version is well approximated with a log-normal law of mode $m = 2.28$ and standard deviation $\sigma = 0.82$. This short (80 seconds) scene is a moderately complex interior scene, with four main characters, all engaging in a variety of actions, including two-way and three-way dialogues, physical contacts, and everyday activities such as sweeping the floor and serving food. All animations were manually annotated to provide $(subject, verb, object)$ descriptions at the right time-codes. Twenty-five cameras were manually placed for the whole duration of the sequence (sixteen of them closely approximating the actual cameras from the original movie, and nine providing alternative angles).

The evaluation of editing is a general and challenging problem (Lino et al. 2014) since no ground truth is available

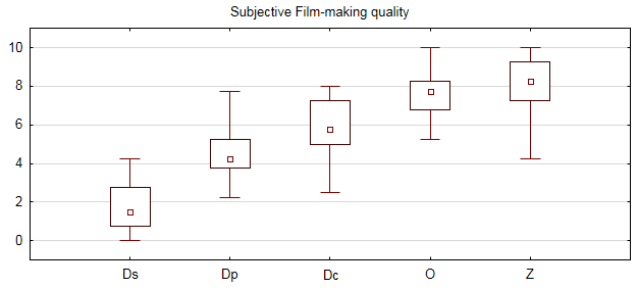


Figure 4: The scores obtained by the five stimuli, each represented with a whisker plot. The central point represents the median score, the box represent the scores between the first and third quartiles, and the bottom and top lines represent the minimum and maximum scores.

for objective comparisons. As a consequence, the quality of an edit can only be measured subjectively through indirect user evaluations. Therefore, to demonstrate the soundness of our model, we have experimentally compared our method (O) to the original edit of the scene (Z) reproduced from Zemeckis' movie (which serves as a reference for comparison with expert cinematographers) and to three degraded versions: a degraded version (Ds) where the content of shots is not considered (*i.e.* the shot cost is removed), a degraded version (Dp) where the enforcement of the specified cutting rhythm is not considered (*i.e.* the rhythm cost is removed), a degraded version (Dc) where visual discontinuities are enforced (*i.e.* the cut cost is reversed).

We performed a subjective evaluation of our method by designing a perceptual user-study. Twenty-one participants volunteered for this experiment. They were 27.5 ($\pm$ 5.7) years old (range: 20 to 42). They were naive with respect to the purpose of the experiment. All had normal or corrected-to-normal vision. They gave written and informed consent and the study conformed to the declaration of Helsinki. We prepared 5 stimuli (a fully edited version of 80 seconds per method). Participants were asked to observe the video stimuli while seated in front of a desk. After each stimulus viewing, participants were asked to rank the global film-making quality[3] on a discrete scale ranging from 0 (very bad) to 10 (very good). In total, they repeated this task 20 times (5$\times$4 repetitions). Stimuli were presented in a randomized order. The total duration of the experiment was about 30 minutes.

We tested three hypotheses. **H1:** editing has an impact on the perceived quality of the observed video stimulus; **H2:** each of the three terms in our cost function (shot content, continuity rules and cutting rhythm) has a positive impact on perceived quality; **H3:** the perceived quality of the version done by an expert cinematographer is significantly higher than our method. The dependent variable in this study was the participants' score given to each version. Figure 4 illustrates the scores obtained for each version.

Hypothesis H1 was confirmed using a non-parametric Friedman test ($p < 0.05$). Post-hoc comparisons summarized in Figure 4 confirm hypothesis **H2** that the consideration of each of the three key aspects has a positive impact on subjective quality, but discard hypothesis **H3** that the version done by an expert cinematographer is significantly better scored than the one generated by our system (though Zemeckis' version globally obtained better scores).

To further illustrate and validate our method, we have also generated two new versions of the scene with ASLs of resp. 2 seconds (fast cutting style) and 10 seconds (slow cutting style). The choice of shots is very different in those different styles. In fast cutting style, there is a preference for close-ups and medium shots. In slow cutting style, there is a preference for medium long shots and full shots. For evaluation purposes, we are making our experimental data (including rushes and their annotations) and our experimental results publicly available [4].

Limitations and future work

Our model is currently limited to the case of linear editing, where the chronology of events is maintained. In future work, we would like to remove this limitation by allowing temporal ellipses and re-ordering of events. Another limitation is that we are restricted to a limited choice of cameras. Future work is needed to optimize over camera positions and framings. In addition, the proposed model only enables to control the pacing. Other style parameters such as shot composition (flat vs. deep staging), camera movements (static shots, dolly shots, crane shots), transitions (reverse shots) and lighting preferences would favor user creativity. Our model of shot selection is based on bounding boxes of the character's body parts and a primitive classification of their actions. Objects and places in the scene, as well as character's moods and intentions, should also play a part. Finally, we should note that the semi-Markov assumption has limitations of its own. Important film editing patterns such as book-ending, separation or parallel action (Sharff 1982) cannot be taken into account by such a memory-less model. The investigation of higher-order Markov models or context-free grammars will be pursued to overcome such limitations.

Conclusion

In this paper, we have presented a continuity-editing approach to the automated creation of cinematographic sequences for 3D animations. We have introduced the notion of editing graph and showed how dynamic programming can be used to compute an optimal edit under a semi-Markov hypothesis. We have provided a thorough description of means to rank shots and edits, and to measure the distance to a specified cutting rhythm. Our solution is supported by subjective evaluations obtained in a perceptual user study. The proposed approach performs a clear shift from existing techniques such as idiom-based representations, with a level of expressiveness not addressed by previous contributions. Finally, this work provides the foundations to address novel

[3] We additionally proposed a number of criteria that participants could consider to score each version: the enhancement of characters performance, the synchronization of cuts with the scene content, the aesthetic of shots.

[4] https://team.inria.fr/imagine/continuity-editing/

challenges in automated cinematography, such as learning and reproducing cinematic styles from real-movies.

Acknowledgments

The research presented in this paper benefited from the assistance of many people. Laura Paiardini and Estelle Charleroy created the assets and the animation for the example sequence. Adela Barbulescu helped create the facial animation. Anne-Hélène Olivier and Michael Gelicher gave precious help on experimental design and evaluation of the user study. We also thank the volunteers who participated in the user study.

A preliminary version of this work was developed by the second author as part of the patented Text-to-Movie application by Xtranormal Technologies [5].

This work has been funded by the French Agency for Research (ANR) through projects CHROME and CINECITTA and the European Research Council (ERC) through the project EXPRESSIVE.

Appendix

We here provide the formulas of our cost functions related to continuity-editing rules. We first introduce $v(c)$, a weighting factor defined as the minimum visible area of character c in two consecutive frames. In details, it is computed as follows

$$v(c) = \min(V(c,i,t-1), V(c,j,t))$$

This factor is used to weight costs on each single character. We then give no importance to off-screen characters, little importance to background characters, and more importance to foreground characters (since they are the focus of attention).

Jump cuts

An illustration of the avoidance of jump cuts is given in Figure 5. We penalize a jump cut by summing, on each single character, the degree of similarity between the two frames before and after the cut. Practically, this penalty is computed as follows

$$C_J^T(i,j,t) = \sum_c v(c) \cdot \phi_J\left(\Delta S(c), \Delta\theta(c)\right)$$

where $\Delta S(c)$ and $\Delta\theta(c)$ are the differences in resp. apparent size and view angle of character c between the two frames. ϕ_J is a non-linear function taking these two parameters as input. It considers a minimum acceptable change in apparent size ΔS_{min}, as well as a minimum acceptable change in view angle θ_{min} (often set to 30 degree). ϕ_J then returns the maximum penalty when no change occurs neither in apparent size nor view angle of character c, and the penalty decreases as the change in either apparent size or view angle increases.

Screen continuity

An illustration of the screen continuity rule is given in Figure 6. We penalize such a discontinuity by summing, on each single character, its screen position change. Practically, this penalty is computed as follows

$$C_S^T(i,j,t) = \sum_c v(c) \cdot \phi_S\left(P(c,j) - P(c,i)\right)$$

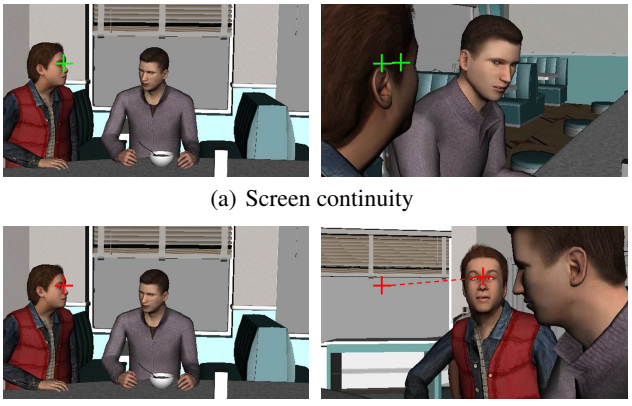
where $P(c,i)$ and $P(c,j)$ represent the 2D screen position of character c resp. before and after the cut. ϕ_S is a non-linear function which takes as input the distance between both positions. It then returns the minimum penalty (0) for two identical positions, and the penalty increases with their distance.

(a) Change in apparent size

(b) Change in view angle

(c) Insufficient change in both size and view angle

Figure 5: *Examples of cuts with sufficient change in size (top), sufficient change in view angle (middle), and an example of jump cut (bottom).*



(a) Screen continuity

(b) Screen discontinuity

Figure 6: *Examples of cuts respecting (top) or violating (bottom) the screen continuity.*

[5] Rémi Ronfard, automated cinematographic editing tool, Xtranormal Technologies, May 2009.

Motion continuity

An illustration of the motion continuity rule is given in Figure 7. We penalize such a discontinuity by summing, on each single character, its change of apparent motion direction. Practically, this penalty is computed as follows

$$C_M^T(i,j,t) = \sum_c v(c) \cdot \phi_M \left(M(c,i), M(c,j)\right)$$

where $M(c,i)$ and $M(c,j)$ are 2D vectors representing the on-screen motion direction of character c resp. before and after the cut. ϕ_M is a non-linear function which takes as input these two consecutive motion directions. It then returns the minimum penalty (0) when the two vectors are close enough (*e.g.* a character moving in a given direction keeps moving in a similar direction after the cut), and the penalty increases as these vectors differ from each other.

Gaze continuity

An illustration of the gaze continuity rule is given in Figure 8. In a similar way to the motion continuity, we penalize such a discontinuity by summing, on each single character, its change of apparent gaze direction. Practically, this penalty is computed as follows

$$C_G^T(i,j,t) = \sum_c v(c) \cdot \phi_G \left(G(c,i), G(c,j)\right)$$

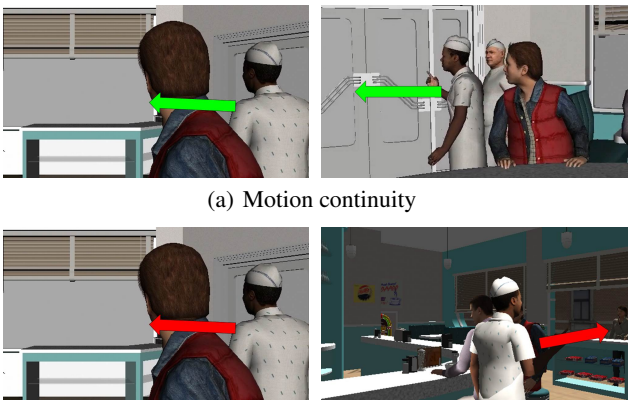
where $G(c,i)$ and $G(c,j)$ are 2D vectors representing the on-screen gaze direction of character c resp. before and after the cut. ϕ_G is a non-linear function which takes as input these two consecutive gaze directions. It then returns the minimum penalty (0) when the two vectors are close enough (*e.g.* a character looking in a given direction keeps looking in a similar direction after the cut), and the penalty increases as these vectors differ from each other.

Left-to-right ordering

An illustration of the left-to-right continuity rule is given in Figure 9. We penalize such a discontinuity by summing, on each pair of characters (c,c'), their change in relative on-screen position (from left to right, this is also known as the 180 degree rule). To do so, we define a new weighting factor $v(c,c')$ computed as the product $v(c) \cdot v(c')$ of the weights of both characters. We then give no importance to a pair of characters where at least one is off-screen either before or after the cut, little importance to a pair of background characters, and much importance to a pair of foreground characters. Practically, this penalty is computed as follows

$$C_L^T(i,j,t) = \sum_{c,c'} v(c,c') \cdot \phi_L \left(L(c,c',i), L(c,c',j)\right)$$

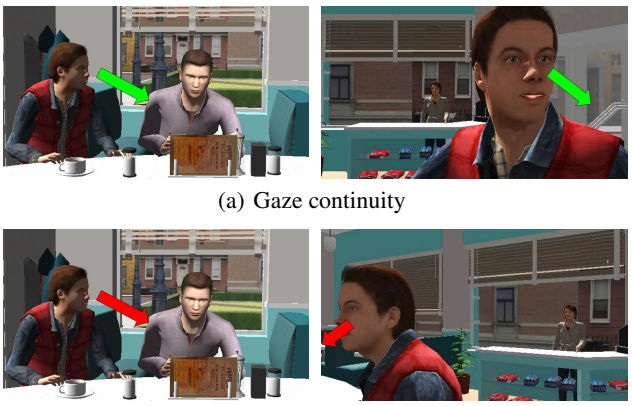
where $L(c,c',i)$ and $L(c,c',j)$ are two real values representing the relative position of characters c and c' resp. before and after the cut (practically, this relative position is computed as the signed difference of their on-screen horizontal coordinates). ϕ_L is a non-linear function taking as input these two reals. It then returns the minimum penalty (0) when both values are of same sign (*i.e.* the relative position of characters is enforced) and the maximum penalty (1) when the two values are of opposite sign (*i.e.* the relative position of characters is reversed).



(a) Motion continuity

(b) Motion discontinuity

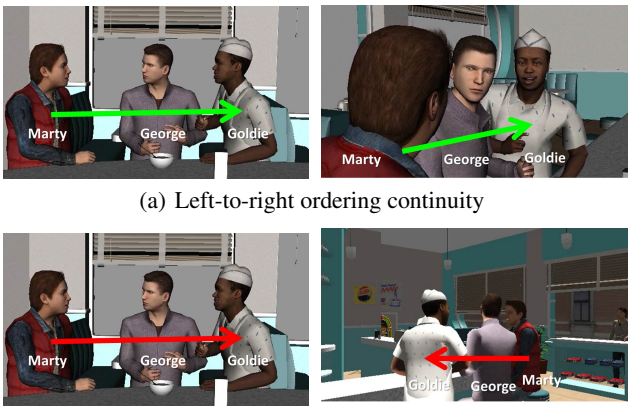
Figure 7: *Examples of cuts respecting (top) or violating (bottom) the motion continuity.*



(a) Gaze continuity

(b) Gaze discontinuity

Figure 8: *Examples of cuts respecting (top) or violating (bottom) the gaze continuity.*



(a) Left-to-right ordering continuity

(b) Left-to-right ordering discontinuity

Figure 9: *Examples of cuts respecting (top) or violating (bottom) the left-to-right ordering continuity (also known as 180 degree rule).*

References

Adams, B.; Dorai, C.; and Venkatesh, S. 2002. Toward automatic extraction of expressive elements from motion pictures: tempo. *IEEE Transactions on Multimedia* 4(4):472–481.

Assa, J.; Wolf, L.; and Cohen-Or, D. 2010. The virtual director: a correlation-based online viewing of human motion. *Computer Graphics Forum* 29(2):595–604.

Bordwell, D. 1998. *On the History of Film Style*. Harvard University Press.

Christianson, D. B.; Anderson, S. E.; He, L.-W.; Weld, D. S.; Cohen, M. F.; and Salesin, D. H. 1996. Declarative camera control for automatic cinematography. In *AAAI*, 148–155.

Cutting, J. E.; DeLong, J. E.; and Nothelfer, C. E. 2010. Attention and the Evolution of Hollywood Film. *Psychological Science* 21(3):432–439.

Datta, R.; Hu, J.; and Ray, B. 2008. On efficient viterbi decoding for hidden semi-markov models. In *19th International Conference on Pattern Recognition.*, 1–4.

Davis, N.; Zook, A.; O'Neill, B.; Headrick, B.; Riedl, M.; Grosz, A.; and Nitsche, M. 2013. Creativity support for novice digital filmmaking. In *SIGCHI Conference on Human Factors in Computing Systems*, 651–660. ACM.

DeLoura, M. 2009. *Real Time Cameras, A Guide for Game Designers and Developers*. Morgan Kaufman.

Dmytryk, E. 1984. *On Film Editing: An Introduction to the Art of Film Construction*. Focal Press.

Drucker, S. M., and Zeltzer, D. 1995. Camdroid: A system for implementing intelligent camera control. In *Symposium on Interactive 3D Graphics*, I3D '95, 139–144. ACM.

Elson, D. K., and Riedl, M. O. 2007. A lightweight intelligent virtual cinematography system for machinima generation. In *Artificial Intelligence and Interactive Digital Entertainment (AIIDE '07)*.

Hawkins, B. 2005. *Real-Time Cinematography for Games*. Charles River Media.

He, L.-w.; Cohen, M. F.; and Salesin, D. H. 1996. The virtual cinematographer: a paradigm for automatic real-time camera control and directing. In *SIGGRAPH*, 217–224. ACM.

Jhala, A., and Young, R. M. 2011. Intelligent machinima generation for visual storytelling. In *Artificial Intelligence for Computer Games*. Springer New York. 151–170.

Kennedy, K., and Mercer, R. E. 2002. Planning animation cinematography and shot structure to communicate theme and mood. In *Smart Graphics*, 1–8. ACM.

Limpert, E.; Stahel, W. A.; and Abbt, M. 2001. Log-normal distributions across the sciences: Keys and clues. *BioScience* 51(5):341–352.

Lino, C.; Christie, M.; Lamarche, F.; Schofield, G.; and Olivier, P. 2010. A real-time cinematography system for interactive 3d environments. In *Symposium on Computer Animation*, 139–148. ACM.

Lino, C.; Ronfard, R.; Galvane, Q.; and Gleicher, M. 2014. How Do We Evaluate the Quality of Computational Editing Systems? In *AAAI Workshop on Intelligent Cinematography And Editing*.

Markowitz, D.; Jr., J. T. K.; Shoulson, A.; and Badler, N. I. 2011. Intelligent camera control using behavior trees. In *Motion in Games (MIG)*, 156–167.

Mitchell, C.; Harper, M.; and Jamieson, L. 1995. On the complexity of explicit duration hmm's. *IEEE Transactions on Speech and Audio Processing* 3(3):213–217.

Murch, W. 1986. *In the blink of an eye*. Silman-James Press.

Murphy, K. P. 2002. Hidden semi-markov models. Technical report, MIT AI Lab.

Salt, B. 2009. *Film Style and Technology: History and Analysis (3 ed.)*. Starword.

Sarawagi, S., and Cohen, W. W. 2004. Semi-markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems*.

Sharff, S. 1982. *The elements of cinema. Towards a theory of cinesthetic impact.* Columbia University Press.

Shi, Q.; Wang, L.; Cheng, L.; and Smola, A. 2008. Discriminative human action segmentation and recognition using. semi-markov model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Smith, T. J. 2005. *An Attentional Theory of Continuity Editing*. Ph.D. Dissertation, University of Edinburgh.

Thompson, R., and Bowen, C. 2009. *Grammar of the Edit*. Focal Press.

Tomlinson, B.; Blumberg, B.; and Nain, D. 2000. Expressive autonomous cinematography for interactive virtual environments. In *International Conference on Autonomous Agents*, AGENTS '00, 317–324. ACM.

Truffaut, F., and Scott, H. G. 1967. *Truffaut/Hitchcock*. Simon & Schuster.

Yu, S.-Z. 2010. Hidden semi-markov models. *Artificial Intelligence* 174(2):215 – 243.

Zen, H.; Tokuda, K.; Masuko, T.; Kobayasih, T.; and Kitamura, T. 2007. A hidden semi-markov model-based speech synthesis system. *IEICE - Trans. Inf. Syst.* E90-D(5):825–834.