A Virtual Director Inspired by Real Directors

Billal Merabti
Polytechnic Military School
Algiers
Algeria

Kadi Bouatouch and **Marc Christie**
IRISA, University of Rennes 1
Campus de Beaulieu
France

Abstract

Automatically computing a cinematographically consistent sequence of shots over a set of actions occurring in a 3D world is a complex task that requires not only the computation of appropriate shots (viewpoints) and appropriate transitions between shots (cuts), but more importantly the ability to encode and reproduce elements of cinematographic style that exist in real movies. In this paper, we propose an expressive automated cinematography model that learns some elements of style from real movies and reproduces them in synthetic movies. The model relies on a Hidden Markov Model representation of the editing process. The proposed model is more general than existing representations that encode cinematographic idioms and proves to be more expressive in the possible variations of style it offers.

Introduction

The recent possibilities in rendering increasingly realistic 3D virtual environments in real-time contexts urges the need for novel techniques to automatically and efficiently convey these contents through the use of appropriate cinematographic techniques. Indeed, there is a clear shift towards a more and more cinematographic experience of 3D environments especially in gaming contexts. More specifically, the reproduction of elements of cinematographic genre and style (war scenes, investigation scenes, etc.) plays a key role in the immersion of users. However most applications, including games, rely on manually preset camera viewpoints, pre-authored camera paths and triggered cuts between viewpoints.

In attempts to replace this manual endeavor, the research community has proposed a number of automated approaches. The computation of automated cinematographic sequences either relies on search techniques in film-tree representations (Christianson et al. 1996) (a film-tree is the representation of a film in a hierarchical structure of scenes, shots, and frames), dynamic programming over collections of template shots (Elson and Riedl 2007), hierarchical planning (Jhala and Young 2005), or finite state machine representations (He, Cohen, and Salesin 1996) (in which a state is a shot, and a transition is a cut between shots). While appraised for their generality, such representations display limitations when trying to perform variations in the way the sequence is shot and edited. Finite state machines encode filmic idioms (which are prototypical ways of shooting actions and actor blockings), and require the manual design of new idioms to perform variations. Film-tree techniques require the design of new heuristics to perform a different search in the film-tree. And dynamic programming techniques over collections of shots require the specification of different costs to perform variations in the generated results. Therefore, the task of mimicking elements of style from real movies is not straightforward.

In this paper, we propose a new expressive model which can compute significant variations in terms of cinematographic style, with the ability to control the duration of shots, and adding specific constraints on the computed sequence. Furthermore, the model is parameterized in a way that facilitates the application of learning techniques so as to reproduce elements of style extracted from sequences of real movies. The approach is founded on a Hidden Markov Model representation of the editing process, where the states represent the shots and the observations are the events related to the shots.

Related work

A number of approaches have been proposed in the literature to partially or fully automate the computation of viewpoints and edits. The seminal work of He etal. (He, Cohen, and Salesin 1996) proposes to encode the directorial process (placing the cameras and selecting the cuts between the cameras) as a finite state machine (FSM). A state in the FSM represents a canonical shot (eg. apex shot, over-the-shoulder shot or panoramic), while a transition between states represents a cut. The transitions are either triggered by scene events (eg. when the distance between two characters reaches a threshold) or temporal events (duration of the current shot). The FSM representation is then used in a real-time context to compute a cinematographic sequence: as the 3D scene evolves, the FSM places the camera corresponding to the shot expressed in the current state, and performs the cut when events occur. By relying on Arijon's cookbook (Arijon 1976) of camera setups, the authors encode different filmic idioms (stereotypical ways of shooting

character configurations and actions) as different finite state machines, organized in a hierarchical representation that enables transitions between FSMs. This hierarchical representation can be viewed as a mean to encode some aspects of filmic style (choice in shots, rhythm of cuts). However, the nature of the triggers on the transitions together with the complexity of associating idioms to all possible character configurations restricts the applicability of the technique to well-known scenarios. Furthermore, it is necessary to specify different FSMs to encode different filmic styles.

Other approaches rely on a combinatorial branching system (see the film-tree proposed by Christianson etal. (Christianson et al. 1996)) that represents a movie as a hierarchical structure of sequences decomposed into scenes, candidate idioms and candidate frames. Computing the best sequence of frames consists in searching the best frames, the best idioms and the best transitions between idioms by exploring and evaluating all the possibilities in the film tree.

A more general process that includes the tasks of blocking (placing the characters and the scene), shooting and editing has been proposed by Elson and Reidl (Elson and Riedl 2007). The authors rely on a combination of search and dynamic programming to identity the best shots and best blockings, by processing a library of canonical shots associated to canonical blockings. The search step explores the best locations where to stage the scene, while the dynamic programming step searches for the best sequence of shots to convey the beats (where a beat is a sequence of actions, representing a unit in the narrative). While the approach enables variations in the style (through directorial heuristics), the control of these variations seems to be performed through manual parameter weighting.

Jhala and Young (Jhala and Young 2005) propose to encode filmic idioms as a hierarchy of plan operators and rely on a partial order causal link planning system to generate a cinematographic sequence that fulfills directorial goals and conveys a sequence of actions. The planner recursively explores the plan operators, adds and propagates constraints until reaching primitive camera actions. To perform variations in the generated sequence, the specification of a heuristic search function is necessary (cost function).

Therefore, we believe there is a need for a more general representation that is both expressive enough to represent different cinematographic styles, and is able to be driven by data extracted from real movies so as to reproduce elements of style.

Beyond the Markovian assumption

Let X be the set of all possible shots in a cinematographic language and let E be the set of all event types existing in a given scene script. The objective of automated cinematography process is to associate the best sequence of shots $\dot{s} = s_1, s_2, \cdots, s_N$ with $s_i \in X$ within all possible shot sequences $\hat{s}$ to a given sequence of events $\hat{e} = e_1, e_2, \cdots, e_N$ with $e_i \in E$ (see equation 1).

$$\dot{s} = \arg\max_{\hat{s}}(p(\hat{s}|\hat{e})) \tag{1}$$

The easiest way to compute the sequence $\dot{s}$ for a given script $\hat{e}$ would be simply to ignore the sequential aspects between the shots and treat them as Independent and Identical Distributed Shots (IIDS), corresponding to Figure 1. In such case, each event e_i is treated independently and each shot s_i depends only on the event e_i. Therefore, the probability of a sequence of shots $\hat{s}$ given a sequence of events $\hat{e}$ is given in equation 2.

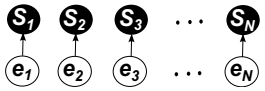


Figure 1: Independent and identical distributed shots.

$$p(\hat{s}|\hat{e}) = p(s_1, s_2, \cdots, s_N|e_1, e_2, \cdots, e_N) = \prod_{i=1}^{N} p(s_i|e_i) \tag{2}$$

Such an approach, however, fails to exploit sequential patterns that appear in the scene, such as transition patterns between shots. Furthermore, if we treat the script considering this hypothesis, then the only information we can extract from the learned data is the relative frequency of shots used to shoot a single event. However, practice in cinematography shows that there is a strong dependency between shots (Mascelli 1965).

To express such dependencies in a probabilistic model, we need to relax the independent distribution IIDS assumption, and one of the simplest ways to do this is to consider a Markov Model (Bishop 2006). First of all we note that, without loss of generality, we can use the *chain rule* to express the joint distribution for a sequence of observations in the form:

$$p(\hat{s}) = p(s_1, s_2, \cdots, s_N) = p(s_1) \prod_{i=1}^{N} p(s_i|s_1, \cdots, s_{i-1}) \tag{3}$$

If we, now, assume that each of the conditional distributions on the right-hand side is independent of all previous observations except the most recent one, we obtain the first-order Markov chain (Bishop 2006), which is depicted as a graphical model in Figure 2. From the d-separation property, we see that the conditional distribution for the shot s_n, given all of the observations up to time n, is given by:

$$p(s_n|s_1, \cdots, s_{n-1}) = p(s_n|s_{n-1}) \tag{4}$$

Joint distribution for a sequence of N observations under this model is given by:

$$p(\hat{s}) = p(s_1, s_2, \cdots, s_N) = p(s_1) \prod_{i=1}^{N} p(s_i|s_{i-1}) \tag{5}$$

From these equations and considering that each shot s_i depends on one single event e_i, we can represent the joint probability as:

$$p(\hat{s}|\hat{e}) = p(s_1|e_1) \prod_{i=2}^{N} p(s_i|s_{i-1}, e_i) \tag{6}$$

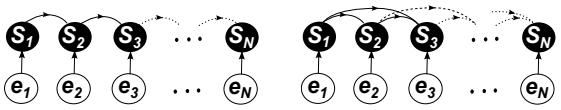


Figure 2: Shot dependency.

which is easily verified by direct evaluation starting from equation 3 and using the product rule of probability. Thus if we use such a model to predict the next shot in a sequence, the distribution of predictions will depend only on the the value of the corresponding event and the value of the immediately preceding shot and will be independent of all earlier shots.

Transitions between shots are represented as transition matrix, each element of which is determined by an event. The matrix defines a first order Markov model. Let us assume a shot s takes values from a set $X = \{x_1, x_2, x_3\}$, the transition matrix can be represented as a Markov automaton (see Figure3). A sequence $\hat{s}$ is simply a feasible path in the automaton.

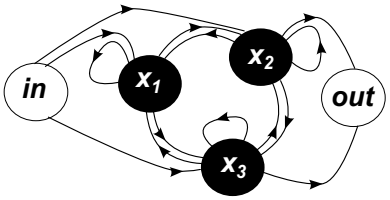


Figure 3: Finite state Markov automaton Illustrating transitions between three shots x_1, x_2 and x_3

Although this is more general than the IIDS model, it is still very restrictive. In many cinematographic shot sequences, the choice of a shot depends on a number of previous shots. One way to allow earlier shots to have an influence is to move to higher-order Markov chains. If we allow the predictions to depend also on the previous-but-one value, we obtain a second-order Markov chain, represented by the graph in Figure 2.b. The joint distribution is now given by:

$$p(s_1, s_2, \cdots, s_N) = p(s_1)p(s_2|s_1)\prod_{i=2}^{N} p(s_i|s_{i-2}, s_{i-1}) \tag{7}$$

Similarly to equation 4 and from the equations 3 and 5 the joint probability will be:

$$p(\hat{s}|\hat{e}) = p(s_1|e_1)p(s_2|s_1, e_1)\prod_{i=2}^{N} p(s_i|s_{i-2}, s_{i-1}, e_i) \tag{8}$$

Each shot is now influenced by two previous shots. We can similarly consider extensions to an M^{th} order Markov chain in which the conditional distribution for a particular variable depends on the previous M variables. However, for this increased flexibility, the cost will be much larger due to the number of parameters in the model.

In our work, to address this issue, we propose to reverse this vision: the relationship between events and shots is expressed in the inverse direction. We consider that the shot informs us of the type of event. The information that we need to learn now becomes: $p(e_i|s_i)$. Modeled this way (Figure 4), our model looks like a fairly known model in the literature: Hidden Markov Model.

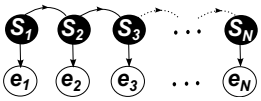


Figure 4: Hidden Markov Model representation of a cinematographic sequence.

Hidden Markov Models (quick reference)

A hidden Markov model (HMM) is a statistical model in which the system being modeled is assumed to be a Markov process with unknown parameters. HMMs are widely used especially in pattern recognition, artificial intelligence or automatic natural language processing (Boudaren et al. 2008)(Rabiner 1989). A HMM is defined as a tuple $\{X, O, \Pi, A, B\}$ where :

- X is a set of states which here denotes the set of possible shots: $X = \{x_1, x_2, ..., x_k\}$
- O is a set of observations which here denotes the set of possible events: $O = \{o_1, o_2, ..., o_l\}$
- $\Pi = \{\pi_1, \pi_2, ..., \pi_k\}$ where π_i is the probability for which the shot s_1 is equal to x_i: $\pi_i = p(s_1 = x_i)$
- A is the transition matrix. Each element $a_{ij}|i, j < k$ is the probability of transition from a state x_i to another state x_j

$$a_{ij} = p(s_n = x_j|s_{n-1} = x_i)$$

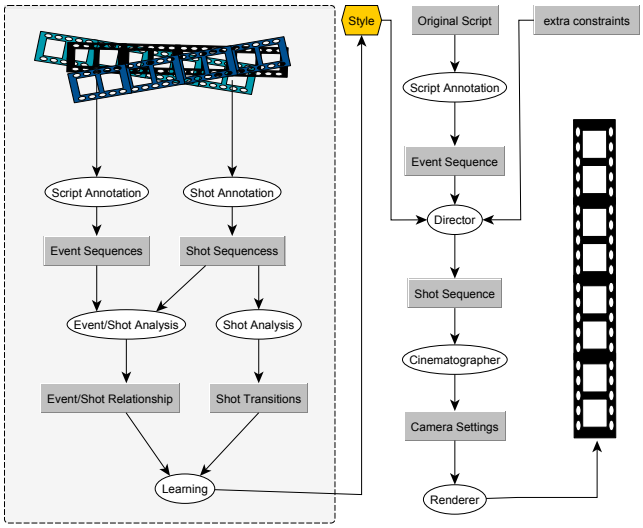
- B is called the emission matrix. Each element is the probability of an event knowing the shot:

$$b_{ij} = p(e_n = o_j|s_n = x_i)$$

with the constraints:

$$\sum_{i=1}^{k} \pi_i = 1, \qquad \forall i, \sum_j aij = 1, \qquad \forall i, \sum_j bij = 1$$

There are three typical examples of problems that can be resolved with an HMM: evaluating, decoding and learning (Rabiner 1989). The decoding problem consists, given a HMM, to find out the most probable sequence of hidden states (shots in our case) which generates a given sequence of observations (Input script). This problem is, in general, solved using a dynamic algorithm of quadratic complexity well known as the *Viterbi algorithm* (Viterbi 1967). The learning problem, which consists in building and parameterizing a HMM is often addressed using the iterative "Baume-Welch" algorithm (Baum et al. 1970).



On the left, cinematographic style is learned from sequences of annotated movies. On the right, the learned cinematographic style is applied to generate a sequence of shots for a new script.

Figure 5: Overall process.

Label	Details
HIGH_HIGH	The actor of the highest hierarchy exit the stage and replaced by another with a highest hierarchy than the remaining one
HIGH_LOW	The actor of the highest hierarchy exit the stage and replaced by another with a lower hierarchy than the remaining one
LOW_LOW	The actor of the lower hierarchy exit the stage and replaced by another with a lower hierarchy than the remaining one
LOW_HIGH	The actor of the lower hierarchy exit the stage and replaced by another with a highest hierarchy than the remaining one
BOTH	Both actors of the current stage were not participating in the last one

Table 1: List of meta-events related to the actors importance and presence.

Overview

The main objective of our work is to propose an expressive editing model that is able to generate a sequence of viewpoints from a script describing events occurring in a 3D environment. We first propose a *Director* that has the ability to reproduce a given cinematographic style. Learning cinematographic style consists in extracting cinematographic elements such as shot description: which type of shot, which actors are composed in the shot and which ones are in focus; shot transitions: towards which shots the cuts are performed; and relations between shot descriptions and events (see section).

The overall process is the organized as follows (see figure 5): first, a *Script Analyzer* transforms the original script of the scene into an augmented script by applying a semantic segmentation and attaching meta-information to the different segments. The meta information is the annotation of the events to be performed by the actors. Second, the *Director*, encoded using a hidden Markov model representation, relies on this event sequence to produce the the resulting sequence of shot descriptions. This sequence of shot descriptions will then be used by a *Cinematographer* (Lino and Christie 2012) to position the camera and set its parameters in a 3D environment.

Director

In this paper, we propose to tackle our problem of reproducing cinematographic style by using a probabilistic learning technique. We first describe how to model the problem.

Modeling

Augmented Script (Observations) The first step consists in providing a formalization of the input script as a sequence of real events and meta-events (see Figure 6). Real events specify the nature of dialogues in the script and we propose to classify these dialogues into three categories: symbolic, moral and narrative as described in (Boussion 2013). Meta-events describe changes of actors between real events such as when an important actor in a scene is replaced by another one of less importance (see Table 1). Meta-events are modeled using discrete variables which provide information about the actors entering and leaving the stage. This information is inserted into the event list as a meta-event before the real event occurs in the script. These meta-events therefore describe the role of an actor expressed as (i)his importance with respect to other actors in the current event, and (ii)his presence or not in the related events. The presence, which is the information about the evolution of participating actors in the event sequence, helps shooting scenes with more than two actors.

The classification of the semantics in dialogues (symbolic, moral and narrative) provides a more fine-grained representation than most approaches (Lino et al. 2010), and is better suited to the way filmmakers construct dialogues. Second, the introduction of the varying actor importance allows modeling sequences with multiple actors entering and leaving the scene, an aspect only partially addressed by previous works.

Shot specification (Hidden States) Constructing shots for cinematography and photography consists in specifying a number of features: the *shot type*, the *composition*, the *focus*, the lighting and the color (Ward 2003). In our work, we only consider the three first features. These features are the minimal set for a virtual cinematographer to shoot scenes comprising two actors (Lino and Christie 2012), i.e. to decide camera position, orientation, zoom and depth of field.

In our representation, a shot s_n is considered as a generic shot independently of the actors involved. However, it encompasses the relative importance (highest or

```
Jack: Gentlemen, what do keys do?
Turkish Pirate: Keys unlock things?
Gibbs: And whatever this key unlocks, inside there's something
     valuable. So we're setting out to find whatever this key
     unlocks.
Jack: No. If we don't have the key, we can't open whatever we
     don't have that it unlocks. So what purpose would be
     served in finding whatever need be unlocked, which we
     don't have, without first having found the key what
     unlocks it?
Gibbs: So we're going after this key.
Jack: You're not making any sense at all.
Jack: Any more questions?
```

a. Original script scene

```
\meta_shot{new actor (Jack)}
Jack to All:" \mor{Gentlemen,} \que{what do keys do?}"
\meta_shot{new secondary actor(TP)}
Turkish Pirate to Jack: " \que{Keys unlock things?}"
\meta_shot{TP replaced by secondary actor(Gibbs)}
Gibbs to Jack: "\nar{And whatever this key unlocks, inside
     there's something valuable.} \nar{So we're setting out to
     find whatever this key unlocks.}"
Jack to Gibbs: " \mpl{No.} \mgl {If we don't have the key, we
     can't open whatever we don't have that it unlocks.}
     \mpl{So what purpose would be served in finding whatever
     need be unlocked,} \nar{which we don't have,} \mpl{without
     first having found the key what unlocks it?}"
Gibbs to Jack: "\nar{So we're going after this key.}"
Jack to Gibbs: "\mpl{You're not making any sense at all.}"
Jack to All: "\que{Any more questions?}"
```

b. Augmented script scene

Figure 6: Original and augmented (annotated) script

lowest) of each actor in the shot. Let us take an example of a shot s_n corresponding to a shot type "*Medium Over the Shoulder*" with the focus set on an actor of highest importance, denoted H, in the corresponding event e_n. The description of the state (shot) is denoted $s_n = MediumOverShoulder(H)$. Additional information on the position of the two actors on the screen, can be specified using a set of possible compositions (ex. $s_n = MediumOverShoulder(H, Hleft\_Lright)$ where $Hleft\_Lright$ specifies that actor of lowest importance L is in the right of the screen and H is on the left). At this point, the shot s_n does not precisely specify which are the actors composed in the shot.

Now, in order to determine which actors are composed in the shot, a second type of hidden state is introduced: the *prep-shot*. A prep-shot is a state that helps to prepare a subsequence of shot states by precisely defining who is the actor of highest, respectively lowest, importance in the shot. Prep-shot states are associated to meta-events observations in the script. Meta-events define changes between the actors as well as changes in their relative importance and prep-shot states reflect these changes in the shots.

Matrix representation We propose to represent an instance of an HMM (i.e. a cinematographic style) as two matrices (see Figure 7). The left and right matrices respectively represent the transition and the emission matrices of the HMM. The transition matrix is a square matrix (figure 7.left) where each element (i, j) provides the probability of transition from a shot x_i to a shot x_j: $p(s_n = x_i | s_{n-1} = x_j)$. The highlighted bottom-right square in the transition matrix represents transitions between shots, while other areas represent the transitions including prep-shots.

Regarding the emission matrix (the rightmost image), each couple (i, k) provides the probability of an event o_k given the shot x_i: $p(e_n = o_k | s_n = x_i)$. The highlighted bottom-right square in the emission matrix represents probabilities of real events regarding shots. While the top-left framed area represents the relationships between meta-events and prep-shots. Probabilities in other areas are set to zero, meaning that there is no relation between meta-events and shots or between real events and prep-shots.

Learning

The learning process consists in determining the parameters of hidden Markov model. Each HMM can be initialized manually so as to reproduce academic cinematography style that will express all the constraints and techniques empirically designed and taught in art schools (see Figure 7). However, the aim of our work is to be able to reproduce a specific style of a real film director. We therefore rely on some of their movies to decide the HMM's parameters by resorting to an iterative learning technique (Rabiner 1989).

In our approach, the parameters of a HMM are initialized manually basing on an academic cinematography style. We fulfill all the possible transitions, authorized in cinematography (Mascelli 1965), in the transition matrix with regular values. The emission matrix is initialized in the same way with all the doable shot/event. This way for initialization permit to our HMM to find a path (shot sequence) in all cases.

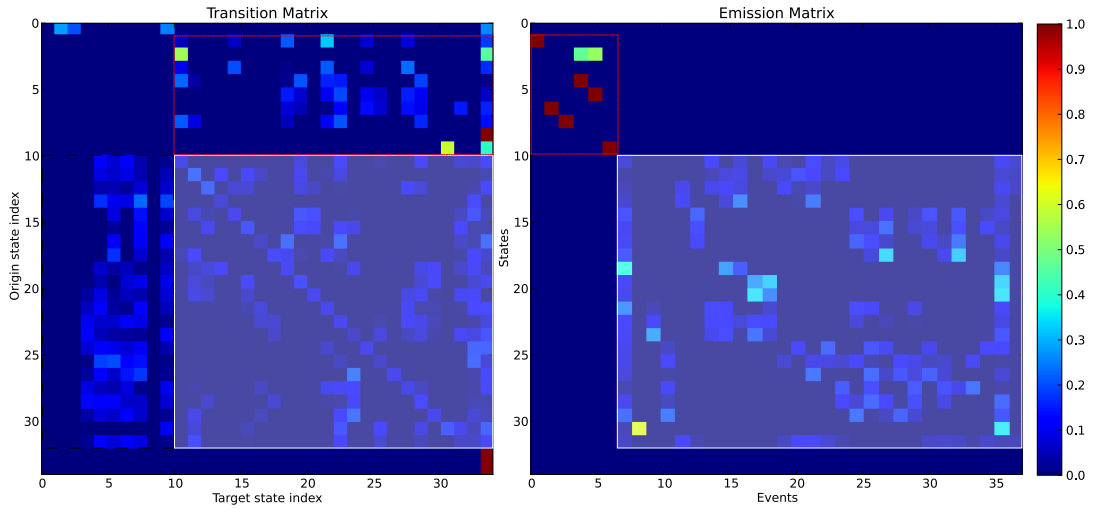
Then, we adjust them iteratively using the Baume-Welch algorithm for every sequence retrieved from a movie produced by the director for which we hope to learn the style. Each sequence is obtained from dialog scenes. These scenes are first annotated to get two sequences: (i) a sequence of events and (ii) the corresponding sequence of shots. Then, the sequences are automatically analyzed to insert meta-events and prep-shots. This annotation is performed using a manual process. Shots are easily observed and classified to get the needed features (shot type, composition and focus). By contrast, event annotation still more complicated to be processed due to its subjective aspect. This annotation is based on the semantics of the script which can be interpreted differently from an operator to another.

Once the sequences of shots and events are annotated and meta-events and prep-shots are inserted, we apply the Baume-Welch algorithm (Rabiner 1989) to update our HMM parameters to converge to better represent the style that we want to learn.

Cinematographer

Automatically positioning a virtual camera in a 3D environment given the specification of our shot's properties to be satisfied (on-screen layout of subjects, visibility, shot type) is a complex problem. Most approaches address the problem by expressing visual properties as constraints or functions to optimize over the camera parameters, and rely on computationally expensive search techniques to explore the solution space.

In our approach, we rely on the manifold representation introduced by (Lino and Christie 2012) to express and solve



States: 1.init 2.set(2actors) 3.set(1actor) 4.update(2act) 5.update(Xact) 6.update(Yact) 7.update(Up) 8.update(Down) 9.invert() 10.update(Symb) 11.fullBody(X) 12.medium(X) 13.close(X) 14.extCloseUp(X) 15.fullBody(Y) 16.medium(Y) 17.close(Y) 18.extCloseUp(Y) 19.fullBody2Shots(X) 20.Medium2Shots(X) 21.CloseUp2Shots(X) 22.FullBodyOverShoulder(X) 23.medOverShoulder(X) 24.closeOverShoulder(X) 25.fullBody2Shots(Y) 26.medium2Shots(Y) 27.close2Shots(Y) 28.fullBodyOverShoulder(Y) 29.medOverShoulder(y) 30.closeOver-Shoulder(Y) 31.symbShot(A) 32.overAllShot() 33.final
Events: 1.update(X,Y) 2.updateUp(X,Y) 3.updateDown(X,Y) 4.update(X) 5.update(Y) 6.update(S) 7.neutral 8.symbolic(S) 9.sementic(X,Y) 10.semantic(Y,X) 11.semantic(X,All) 12.semantic(Y,All) 13.narration(X,Y) 14.moralGle(X,Y) 15.moralPle(X,Y) 16.moralOrder(X,Y) 17.question(X,Y) 18.narration(X,All) 19.moralGle(X,All) 20.moralPle(X,All) 21.moralOrder(X,All) 22.question(X,All) 23.narration(Y,X) 24.moralGle(Y,X) 25.moralPle(Y,X) 26.moralOrder(Y,X) 27.question(Y,X) 28.narration(Y,All) 29.moralGle(Y,All) 30.moralPle(Y,All) 31.moralOrder(Y,All) 32.question(Y,All) 33.indetermined()

Figure 7: A representation for film style encoding probability of transitions between shots, and probability of events knowing shots

the exact on-screen positioning of two or three subjects. The manifold representation relies on an efficient technique that consists in expressing the solution space as a 2D manifold surface for a specified composition with two subjects. We therefore express a given type of shot (eg, medium over the shoulder shot) as a point on the manifold surface. The representation then enables the computation of a full camera configuration (position, orientation and field of view).

Shot Generation

Given a sequence of events $\hat{e}$ (real events or meta-events), output shots are generated by the Director as described in Figure 5 using a *decoding* process. Decoding consists in associating a shot s_i for each event e_i such as $\dot{s} = s_1, s_2, \cdots, s_N$ is the solution of the equation 1 using the learnt style represented as a HMM.

Before determining the sequence $\dot{s}$ in equation 1, let us develop this equation. We first recall the Kolmogorov definition applied to our parameters:

$$p(\hat{s}|\hat{e}) = \frac{p(\hat{s}) \cdot p(\hat{e}|\hat{s})}{p(\hat{e})} \tag{9}$$

Now, if we replace $p(\hat{s}|\hat{e})$ in the equation 1, we obtain:

$$\dot{s} = \arg\max_{\hat{s}} \frac{p(\hat{s}) \cdot p(\hat{e}|\hat{s})}{p(\hat{e})} \tag{10}$$

The denominator of equation 9 or 10 is independent from the argument to maximize. Therefore, equation (1) becomes:

$$\dot{s} = \arg\max_{\hat{s}} p(\hat{s}) \cdot p(\hat{e}|\hat{s}) \tag{11}$$

We know that there is no dependency between any shot s_i and any event e_j except if $i = j$ (see Figure 7). Furthermore, there are no dependencies between events e_i. Therefore:

$$p(\hat{e}|\hat{s}) = \prod_{i=1}^{N} p(e_i|s_i)$$

On the other hand, the Markov assumption on the sequentiality of shots gives us:

$$p(\hat{s}) = p(s_1) \prod_{i=2}^{N} p(s_i|s_{i-1})$$

Now, we replace these values of $p(\hat{s})$ and $p(\hat{e}|\hat{s})$ by their respective expressions in equation 11. Then we can deduce that:

$$\dot{s} = \arg\max_{\hat{s}=s_1 \cdots s_N} p(s_1)p(e_1|s_1) \prod_{i=2}^{N} p(s_i|s_{i-1})p(e_i|s_i) \tag{12}$$

To compute $\dot{s}$ in the equation 12, we can use the well-known Viterbi algorithm which is exactly adapted to this problem (Forney Jr 1973). The Viterbi algorithm is very efficient and its complexity is equal to $O(N \times |X|)$. It ensures, given the sequence $\hat{e}$, the provision of the best sequence of shots $\dot{s}$ according to the learnt style.

Viterbi Algorithm

The Viterbi algorithm was introduced by Andrew Viterbi in (Forney Jr 1973) as a dynamic programming algorithm which computes the optimal sequence without using an exhaustive search process. It determines from an observation

sequence $\hat{e}$ the most likely sequence of hidden states $\dot{s}$ (also called: *Viterbi path*), that might generate it.

To compute $\dot{s} = s_1, s_2, \cdots, s_N$ using the Viterbi algorithm, let:

- $\hat{e}_{u:v}$ denotes a sub-sequence of events from $\hat{e}$ beginning from e_u and finishing at e_v included.
- $\tilde{s}_t(x_i)$ denotes the best shot sub-sequence of length t and finishing by the shot $s_t = x_i$.

$$\tilde{s}_t(x_i) = \underset{\hat{s}=s_1 \cdots s_t}{\arg\max}\, p(\hat{s}|\hat{e}_{1:t-1}) \cdot p(s_t = x_i|\hat{s}_{t-1}) \quad (13)$$

- $\sigma_t(x_i)$ denotes the best partial probability of reaching the intermediate state x_i at the time t.

$$\sigma_t(x_i) = p(\tilde{s}_t(x_i)|\hat{e}_{1:t}) \quad (14)$$

From the definitions below, we can easily see that:

$$\dot{s} = \underset{k}{\arg\max}\, \tilde{s}_N(x_k) \quad (15)$$

Calculating sub-sequences

σ can be determined through a recursive computation. For this, we compute partial probabilities σ as the most probable route to our current position given a known Hmm.

When $t = 1$ the most probable path to a state does not exist. We however use the probability of being in that state given $t = 1$ and the observable event e_1 of $\hat{e}$.

$$\forall k, \quad \tilde{s}_1(x_k) = (x_k), \quad (16a)$$
$$\sigma_1(x_k) = \pi_k \times b_{k,e_1}, \quad (16b)$$

Now, we show that the partial probabilities σ_t at time t can be calculated in terms of the vector σ_{t-1} which denotes the σ's at time $t-1$.

$$\tilde{s}_t(x_k) = (\tilde{s}_{t-1}(x_j), x_k) \quad (17a)$$
$$\sigma_t(x_k) = \max_i(\sigma_{t-1}(x_i) \times a_{ik} b_{k,e_t}) \quad (17b)$$

where :

$$j = \underset{i}{\arg\max}(\sigma_{t-1}(x_i) \times a_{ik})$$

Results

To present the results, we built a rough 3D model mimicking a sequence from the "what do keys do?" scene from Pirates of Carribeans (referred to as "POC").

In all the following figures, the first row represents the script to which are applied different styles. The second row shows the shot distances: Extreme close-up, Close-up, Medium, Full-body, Over-all. The last three rows show the presence of each actor: white (actor not present in the shot), gray (partially visible) and black (actor on focus). Each row, except the first one, is divided into two sub-rows corresponding to the results obtained with two different styles, except for figure 9 where the first sub row corresponds to the style of the real movie.

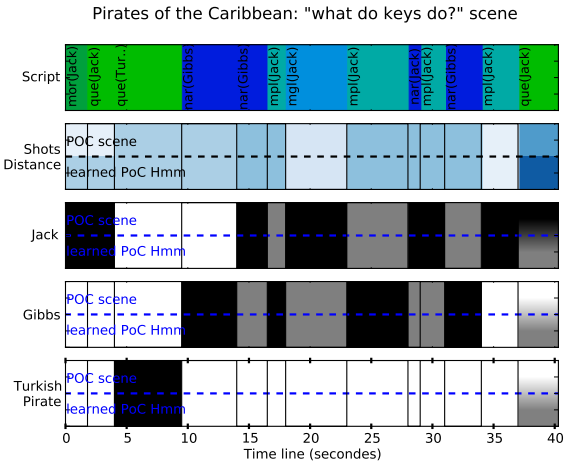


Figure 9: Learned POC HMM Style applied to the "What do keys do?" actions.

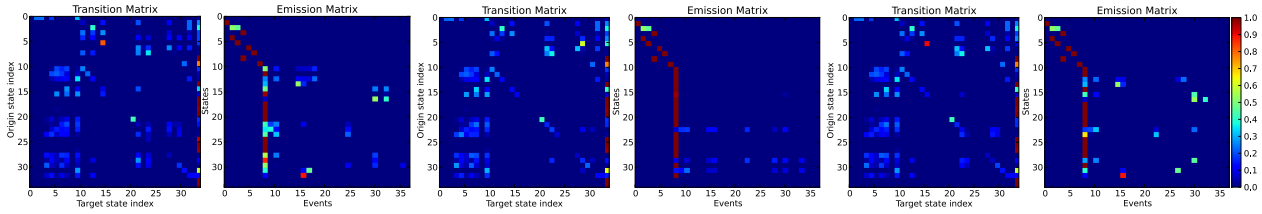
Applying a style learned from POC scenes Figure 9 shows the sequence of shots of the real POC movie as well as those obtained with the 3D model after applying the style learned from 60 shots of POC taken from 5 distinct scenes.

The sequence obtained with our method is very similar to the real one, which demonstrates the capability of our system to at least reproduce some aspects of the editing process (viewpoints, and transitions are mostly the same).

Applying other styles To demonstrate the flexibility of our system, we confront in Figure 10 the sequence learned from Pirates of the Caribbean and the one learned from a significantly different style (One piece, a Japanese manga by Eiichiro Oda).

While distances in shots are quite similar (only 2 shot lengths out of 13 are different), the way the main characters are being framed in these shots differ significantly (38% of the shots are different).

FSM vs HMM We compare our HMM-based approach to a classical finite state machine (FSM) representation ((He, Cohen, and Salesin 1996)) on the same dataset. In a FSM representation, each type of shot is encoded as a state, and probabilities of transitions between shots are specified by a transition matrix which is similar to that of an HMM. FSM representations make a decision to select a shot for each new event. In other words, it chooses the next shot depending only the current shot and the next event (local decision). In comparison, our method selects the optimal sequence of shots while considering all the sequence of events. The necessity to objectively compare the two methods requires some changes. Indeed, for FSM, we propose to take into account the whole sequence of events using the Viterbi algorithm and use the same learning dataset as the one used for HMM. The resulting shots are compared to those obtained with our method as shown in figure 11. The FSM approach fails to reproduce the learned style between the 13th and the 23rd second, whereas our approach follows it faithfully.



(a) learned HMM OP. (b) Edited Basic 3-states HMM. (c) learned HMM POC.

Figure 8: Used HMM styles

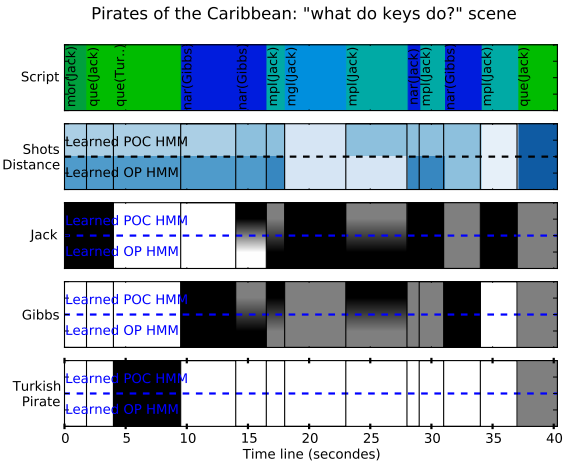


Figure 10: Learned OP HMM vs Learned POC HMM on POC actions.

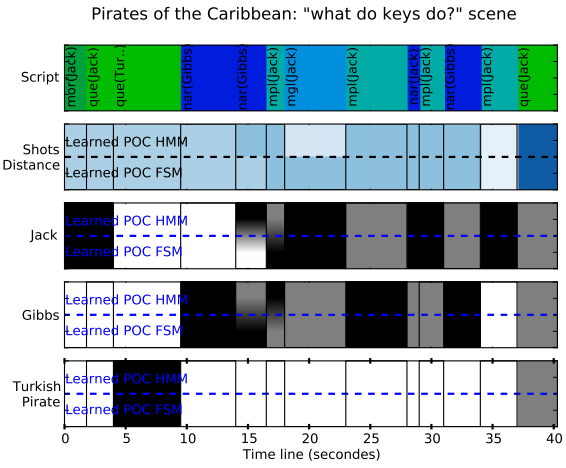


Figure 11: Learned POC Hmm VS learnd POC Fsm on the "What do keys do?" actions.

Conclusion and Future work

We proposed a virtual director based on a hidden Markov model in order to tackle the problem of shooting a virtual scene with cinematographic styles either learnt from real movies or manually edited by a user. To this end, we modeled the problem as an HMM in which the hidden states represent the shots while the observations correspond to the script events of the scene to shoot. Determining a sequence of shots (hidden states) from the input sequence of the events (observations) is performed through a decoding operation. We have shown that after learning a given style from real movies, our method is able to reproduce it. We also demonstrated that our HMM-based approach performs better than finite state machine representations in terms of style reproduction.

In future work, we propose to study the combination and means to perform transitions between different learnt styles, as well a proposing more evolved representations to handle discourse structures such as parallel stories, flashbacks and foreshadows. Besides its application to automated cinematography, this work may also be used as a way to help analyzing styles and discourse of real movies.

References

Arijon, D. 1976. *Grammar of the Film Language*. Hastings House Publishers.

Baum, L. E.; Petrie, T.; Soules, G.; and Weiss, N. 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics* 164–171.

Bishop, C. M. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Boudaren, M. E. Y.; Labed, A.; Boulfekhar, A. A.; and Amara, Y. 2008. Hidden markov model based classification of natural objects in aerial pictures. In *Proceedings of IAENG, London*.

Boussion, T. 2013. Construire un dialogue hollywoodien (apocalypse now, 300, eyes wide shut...). http://analyse-scenarios.over-blog.com/. Internet Blog.

Christianson, D. B.; Anderson, S. E.; He, L.-W.; Salesin, D. H.; Weld, D. S.; and Cohen, M. F. 1996. Declarative camera Control for Automatic Cinematography. In *In proceedings of AAAI'96*, volume 1, 148–155.

Elson, D. K., and Riedl, M. O. 2007. A Lightweight Intelligent Virtual Cinematography System for Machinima Production. In *3rd Annual Conference on Artificial Intelligence and Interactive Digital Entertainment*, 8–13.

Forney Jr, G. D. 1973. The viterbi algorithm. *Proceedings of the IEEE* 61(3):268–278.

He, L. W.; Cohen, M. F.; and Salesin, D. H. 1996. The Virtual Cinematographer: A Paradigm for Automatic Real-time Camera Control and Directingh. In *SIGGRAPH '96 Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, 217–224.

Jhala, A., and Young, R. M. 2005. A Discourse Planning Approach to Cinematic Camera Control for Narratives in Virtual Environments. In *AAAI'05 Proceedings of the 20th National Conference on Artificial Intelligence*, volume 1, 307–312.

Lino, C., and Christie, M. 2012. Efficient composition for virtual camera control. In *Proceedings of the ACM SIGGRAPH/Eurographics SCA*, 65–70.

Lino, C.; Christie, M.; Lamarche, F.; Schofield, G.; and Olivier, P. 2010. A real-time cinematography system for interactive 3d environments. In *Proceedings of the ACM SIGGRAPH/Eurographics SCA*, 139–148.

Mascelli, J. 1965. *The Five C's of Cinematography: Motion Picture Filming Techniques*. Hollywood: Cine/Grafic Publications.

Rabiner, L. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77(2):257–286.

Viterbi, A. J. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on* 13(2):260–269.

Ward, P. 2003. *Picture Composition for Film and Television*. Media production. Focal Press.