Movie Weaver

Automatic Movie Editing

Ariel Tal (036739332, arielt1985@gmail.com)

Eran Weissenstern (300489960, eranws@gmail.com)

Instructors:

Gal Elidan (HUJI, galel@cs.huji.ac.il)

Eyal Soreq (Bezalel Academy of Arts, eyal.soreq@live.bezalel.ac.il)

Michael Fink (HUJI, Google, fink@cs.huji.ac.il)

Table of Contents

Motivation

Abstract

Movie Weaver is a tool that utilizes the power of probabilistic models along with filmmaking paradigms to turn a collection of video clips into a movie.

Professional movie editors have been working for years, but the increase in non-professional video production created a need that hadn't been there before. The need for a reliable automatic tool, which will be able to handle vast amounts of video, to compile movies for fast sharing.

The algorithm pipeline is composed of several stages. The first stage is of extracting features and classifying each of the clips in the dataset, using a predefined set of classifiers. Due to the nature of video analysis this is the most resource consuming part of the process, but it can be preprocessed and optimized extensively.

Having extracted the information describing each individual clip, the second stage of the algorithm is to model the problem. We chose to model a movie as a time-sequence using a Bayesian Network, and to formulate pair-wise relationships. Deciding on the probability that two clips will follow one another is the key to our algorithm. The similarity measure takes its roots from video editing paradigms.

The third and final step is to use inference over the Bayesian Network, finding the most probable path given constraints defined by the user. This sequence is effectively the final movie.

Our model is scalable enough to handle numerous clips (input dataset) and features/classifiers (constraints). It is fast, and can easily be turned into a full product. Our results have been shown to be better than a random generated sequence, and depending on user preference can be improved significantly.

Preface

Recording video has never been easier

We would like to open this paper with a ground-breaking statistic: *"More video is uploaded to YouTube in 60 days than all 3 major US networks created in 60 years".*

In the past decade the world has experienced an enormous growth in video capture. Extensive video capturing is a phenomenon that can be explained by two parallel phenomena. The first is technological – mobile phones are getting smarter by the day, offering higher quality video capturing capabilities at the palm of the hand. The second phenomenon is a social one – sharing content has never been easier. The time between production and the viewers is shortened, and the easiness of the process causes amateur video producers to spend less time producing, and sharing more.

The majority of the video is produced by "the guy next-door" and consists of poor quality footage and with little to no editing. The fact that the footage is created by non–professionals results in un-watchable and boring content. Even in the genre of home video, family-members, which are the target audience, usually find the end result boring.

This project attempts to aid in creating higher-quality end result, which will be easy to watch, and leverage content of poor quality, with a click of a button. The average user of Movie Weaver is an amateur filmmaker, with little to no editing knowledge.

An important note is that the purpose of the project is not the compete with professional editing tools, but to create a new tool that will allow common people to take the video they already shoot daily, and create a movie that is visually pleasant and corresponds with classic editing paradigms.

We rely on assumptions taken from the cinematic field that allow us to better scope the problem. Several editors and researchers have studied these assumptions over the past century, most notably Sergei Eisenstein[1].

Those assumptions can be summarized into two basic key concepts:

1. *Context creates meaning.*
   The human mind is constantly attempting to find meaning in everything we see, we are connecting the dots as we encounter them. This is true in general but

[1] See Bibliography
[2] See appendix #2

becomes more significant in the context of films and movies. The order in which two shots appear creates a story in the viewer's mind.

Lev Kuleshov, a Soviet filmmaker and film theorist, illustrated this phenomenon by a simple example. Kuleshov filmed the most notable actor of the time Ivan Mozzhukhin, and asked him to express no emotion while being filmed. Following the expressionless face of Ivan, Kuleshov showed the viewers a bowl of soup, baby, or a beautiful woman. The response to Ivan's acting was always related to the following image. For example "How hungry Ivan looks", when his face were followed by a shot of a soup bowl.[2]

2. *Composing starts with a single pair.*

   The process of editing a professional movie can be described as the process of deciding the order of the shots, and the time in which to cut them. Placing a pair of clips on the timeline is decided taking numerous factors into account. Eisenstein put down a list of different factors that may affect a match, from color matching to symmetry in the shot and leading lines in the frame. A movie is basically of a series of pairwise matching.

Taking the above assumptions into consideration: chaining pair-wise decisions, we can create an infinitely long chain of clips, thus creating a movie. In the "Methods and material" part of this article, we describe the mathematical foundations of this process.

Why is it hard?

Filmmaking is an art, and as such - no strict guidelines exist as to what can be considered a good result, and what is a better one. Putting in scientific terms, there is no deterministic quality function. Answers vary from narrative to genre; a good movie in one context can be considered bad in another. The problem of a measuring quality introduces a sub-problem – deciding which factors are more critical for the comparison. Eisenstein helped us with a measure of how well two clips work together, but a global quality measure for an entire movie doesn't exist.

To better illustrate the last statement, we would like to remind that professional editors take many decisions while editing. The different decisions can be categorized into: narrative – what are the characters' actions and at what point is the viewer aware of these actions. Aesthetic – as illustrated by Eisenstein in his research, conjunction of different visual patterns creates a different effect upon the viewer. Rhythmic – how fast images change and how fast the story is being told. Deciding which categories are more

[2] See appendix #2

important, which underlying factors to use and what is the overall ordering among these factors, is subjective and is a matter of decision. In a top-end product the user would have control over all of these decisions.

Computational complexity – even if we were to define a quality measure similar to that of a professional editor, the computational complexity would still be very high. Choosing an ordering of 10 clips out of a pool of 50 yields an exponential explosion. We're looking to create more out of large amounts of video, and therefore need a solution that will be able to handle them.

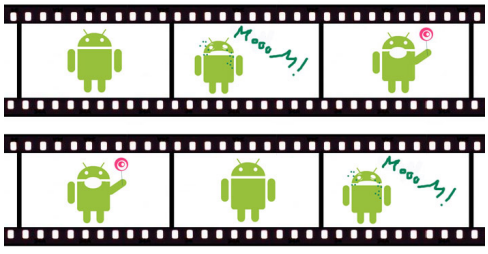
Goals and Decisions

When trying to solve the ambitious goal of automatically editing large amounts of video, we had to make several design decisions.

To begin with, we decided not to compete with the great filmmakers. Our program does not aim to create a better result than Fellini or Hitchcock, but to give an option in filmmaking to an amateur with little time to edit.

Decisions

1. Though this is an important factor in filmmaking, we decided to treat each clip as an atomic unit, and only find the best concatenation of clips. We chose not to deal with the problem of finding the best interest point in a given clip.
2. We decided not to handle narrative, but only in a semantic level. We're not dealing with generating specific stories, but rather with a consistent result. As we explained above, our mind will create the story as long as we keep the sequence consistent.
3. We realized that dataset selection is crucial for the results – a bad set of clips will create bad movies.



Decision #2: Our mind will create the story, as long as we keep the sequence consistent.

Goals

1. The program must require as little input as possible from the user, preferably only the given set of clips.
2. The program must work efficiently and fast and be scalable with the number of input clips.
3. The program must be based on editing paradigms that were written and phrased in the past century. We're not trying to reinvent the wheel.
4. The result must be consistent and better than a random compilation of clips.

Methods and Materials

The solution in a Nutshell

In this part we will describe in detail the problem and our method of dealing with it.

We will describe the way we modeled the problem, the reasons for those decisions and describe in further detail the algorithm implementation.

Modeling

As we mentioned earlier, the problem we deal with is in the field of art, where things are not always black and white. Our natural choice for a mathematical model was of a ***Probabilistic Time-Based Model,*** specifically, a Bayesian network. We decided to use a variant of HMM that will be extensible, answer the requirements and will correspond with an editor's line of thought

A Bayesian network is defined as a "probabilistic graphical model that represents a set of *random variables* and their *conditional dependencies* via a **directed acyclic graph**"[3].

A *random variable* in the model is a:

- *Video clip* chosen from the given set –
  Put simply, this is the choice of which clip appears at which point in time in the sequence.
- *Label* - a semantic tag over a clip (e.g.: dark, long, indoor/outdoor).

The *conditional dependencies* are the relations between two random variables (see figure #1):

- Clip Similarity – Defined over a pair of clips (bottom horizontal edge), determined by the properties of each clip.
  The more two clips are similar, the higher probability that they will be picked. (or dissimilar, depends on the editor's choice).
- Label Transition – Defined over a pair of Labels (top horizontal edge). The probability of an indoor scene to appear after an outdoor one. This is another factor definable by the user.

[3] http://en.wikipedia.org/wiki/Bayesian_network

- Clip Label – Defined over a single clip. The probability of a clip to be assigned a *Label* (vertical edge). This is a probability given using a classifier, but can be derived from a predefined classification table.

Mathematically, the graphic model can be squeezed into a single row of horizontal nodes. We chose to divide the properties into two rows, two categories, *aesthetic* and *semantic*. The roots of this choice lay in professional editor's terminology.

- ***Aesthetic measure –*** The bottom horizontal edges – Defines the similarity of a clip to another. Specifically this is done on a 'low-level' vision features, like brightness, hue or histogram of the overall image.
  This property measures the probability of a pair of clips to follow one another aesthetically. Generally, to create a sequence that matches visually – the clips look like they were intentionally shot for the same movie. The result will be of a consistent movie.
- ***Semantic measure –*** This was computed using the metadata of a clip (length, geographic location, etc.) or extracted using more sophisticated, high-level classification (indoor/outdoor).
  The purpose of this category is to enable creating a semantic narrative (as explained in the preface).
  We want to control not only the visual seamlessness of the final result, but also the semantic one. Two clips can be similar in color, but one was shot in Paris and the other in Barcelona. Encoding high affinity in the transition between geographic tags will result in a more 'local' film that tends to stay in the same geographic location.
- ***Clip Labeling*** – The bridge between the categories – As humans we classify what we see in a semantic level. We may label a scene and categorize it (e.g. where the scene was filmed, Indoor/Outdoor, characters appearing in the scene, or the geographical location of the scene). Such classifications are easy to percept as humans and are fundamental to deciding on a matching of a pair of clips.

Using matrices to define transition probabilities

Our choice of using Matlab allows us to work with matrices. Therefore, we want to express the conditional dependencies in a matrix form.

The dimension of the matrix is set by the number of factors in the conditional dependency. Let's look at a simple example:

Matrix A

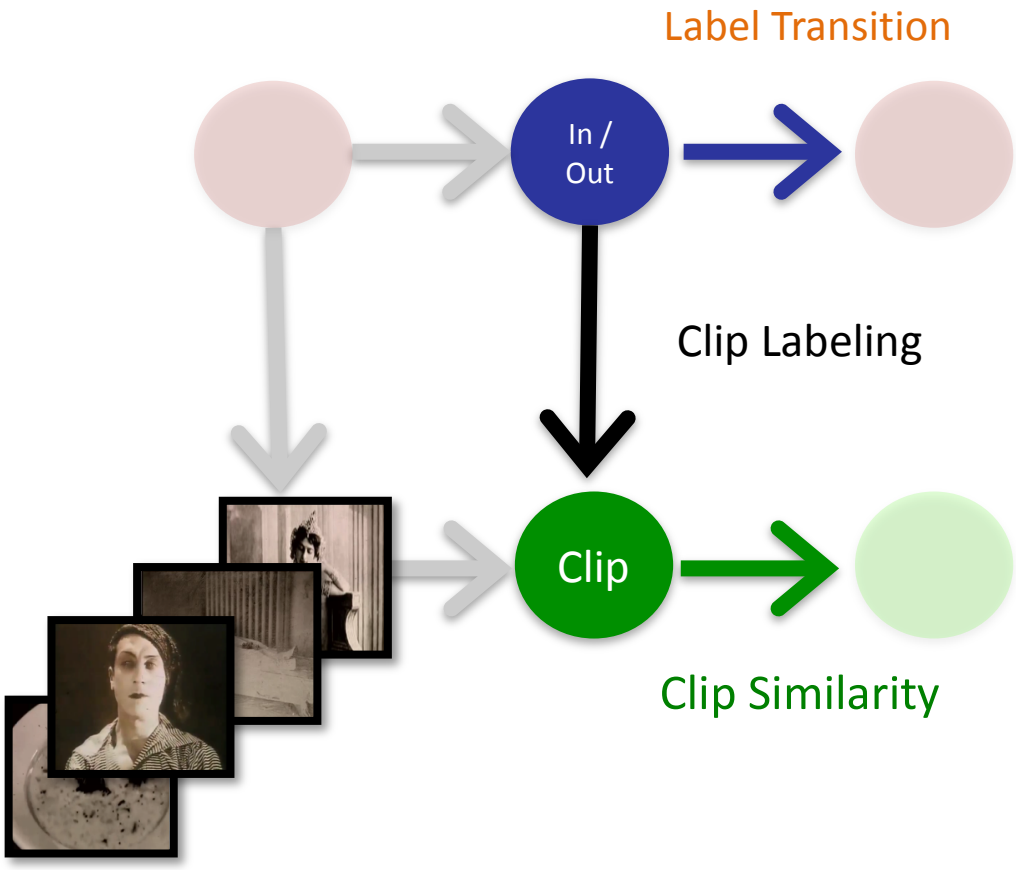
$$\begin{array}{c} \\ in \\ out \end{array} \begin{array}{c} \begin{array}{cc} in & out \end{array} \\ \begin{pmatrix} 0.8 & 0.2 \\ 0.1 & 0.9 \end{pmatrix} \end{array}$$

Matrix B

$$\begin{array}{c} \\ in \\ out \end{array} \begin{array}{c} \begin{array}{cc} in & out \end{array} \\ \begin{pmatrix} 0.1 & 0.9 \\ 0.8 & 0.2 \end{pmatrix} \end{array}$$

To find the probability of the current scene to be indoor, given that the previous one is outdoor, or P(in|out), is 0.2. The probabilities in the transition matrix A can result in a stable movie (the tendency is to stay), whereas the probabilities in matrix B will result in an unstable movie (tendency to switch).

Figure #1 – Our graphical model of the problem



Encoding Clip Similarity

In order to achieve a result, which is aesthetically pleasant it is important to encapsulate the visual properties of a clip.

In the preprocessing stage we calculated each of the visual properties mentioned above (histogram, etc.). We defined a ***similarity function*** specifically for each factor, returning a probability between 0 and 1. It's important to note that if the desired result is dissimilarity, we can define the function to return 1 for the most different, rather than the most similar. For example, for histograms we chose to use L1 distance between the histograms, and returned 1 for the closest pair of histograms.

After calculating the similarity over each pair of clips we get a matrix of size n^2 (n being the number of clips in the input set) representing the overall "histogram-similarity" or "hue-similarity".

Next we use ***weights*** and a normalized linear sum over the different matrices to create a single n^2 matrix representing the overall aesthetic similarity over the entire set. Similar to changing the function itself to get "least similar" instead of "most similar", one can choose to use a negative weight to force a "least similar" measure for a specific factor.

Both the weights, and the similarity functions can be configured by the user.

The resulting matrix is used as a metric in the computation of the likelihood of matching a pair of clips.

Encoding Semantic Classification/Label

Using tagged databases of scenes from movies we can train classifiers to recognize several different classes of scenes. For example: Torralba[4] showed an algorithm for "scene recognition", where he can differentiate between 8 classes of scenery. We used the same classifier to distinguish between Indoor/Outdoor scenes, a distinction that is fundamental in filmmaking.

It's important to note that a classification is not deterministic, it may be that a clip is 80% indoor, and 20% outdoor (e.g. a clip shot on a balcony). Rather than being a drawback, this non-determinism fits well in our model.

[4] See Reference #2

Using various classifiers we determine the relation of a clip to each label, we utilize the uncertainty of some of those classifiers to construct probabilistic *affinity* of a clip to a given label.

Encoding Semantic Behavior

We calculate semantic similarity transitions in a similar manner to the way we calculate aesthetic similarity: we prioritize the semantic labels by their cinematic importance. For example, the importance of geographical location could be set to precede the one of Indoor/Outdoor consistency. The result will be of a movie that is generally taking place in the same location, but tends to hop more between indoor and outdoor scenes

Algorithm

Now that we have a mathematical formulation of the problem, we use probabilistic methods, specifically ***inference*** to calculate the most probable sequence of clips - the 'best' film under the given constraints.

During our work process we initially used a naive implementation of the Viterbi algorithm, which is common for such problems. This was sufficient for the beginning but since we dealt with a bank of hundreds of clips, and as our mathematical model evolved, we decided to use a different method of inference.

Our model is time-invariant, i.e. the model structure does not change, and the parameters do not change. Such a model is called a "Dynamic Bayesian Network" (DBN), and can be solved more efficiently by other methods than viterbi.

Materials

Matlab

The most significant technological choice we made was the decision to develop in Matlab, a decision that affected our work entirely.

Matlab allows a developer/researcher to test ideas and refine algorithms very quickly. Many libraries for image manipulation and Bayesian network modeling are offered for the Matlab environment, making coding much quicker and easier.

The downside of this choice is that the running time of an algorithm in Matlab is much slower than that of a native application, but at an initial stage, when the mathematical foundations need to be laid out, this is a major advantage.

We used a Matlab Toolbox written for dealing with probabilistic graphs called "the Bayes Network Toolbox"[5].

Working with Video

There is no one single format to handle video in the market. There are numerous compressing methods and formats by different distributers, and working with all of them is near impossible. For the scope of this project we decided to use MOV as our video format of choice, and wrote wrappers to play and analyze the video in Matlab. Moving from project to product, the biggest overhead would be handling various video formats.

[5] See bibliography #3

Results

In this article we described a method of automatic editing. Our method produces results that are better visually than a random sequence and corresponds to tendencies, or preferences, defined as input.

We consider our biggest strength to be the algorithmic depth of our solution. A similar solution, using a different variety of classifiers and quality factors can be used in other fields, as described in the “Future Plans” appendix.

Strengths

- Scalability –

    Our algorithm is fast enough to generate satisfying results even using a large number of classifiers, and over a large set of clips.

    The scalability property of our algorithm will allow it to easily turn into a fully-grown product.

- Semantic classifiers can be added easily – Tempo, Geographical, etc.

    The graphic model in the design phase and the code later on, were both written thinking of an infinite number of classifiers. A professional movie editor takes numerous factors into consideration, and we aimed at being able to do exactly that.

- Several levels of abstraction - low-level similarity up to a semantic-based narrative

    By carefully selecting which classifiers to use, we can use the same model to generate a sequence that is compiled using color or histogram similarity. Using sematic classifications, for example using character classifications, we can create a narrative completely derived from the probabilities.

- User preferences are projected as probabilities –

    Going back to one of our initial statements, this project is about art, and therefore any preference the user may have is a preference and not a deterministic yes/no requirement. Though we allow binary requirements, using probabilities, our model gives a greater degree of choice by specifying a hierarchy of importance between the user’s desires.

- Fast –

  The pre-processing phase of our algorithm is the part that consumes largest amount of resources. This is a problem that even high-end, professional editing tools on the market suffer from, and the problem only grows with higher video quality. Having that in mind, we designed our model so that the pre-processing part can happen only once, which allows a very fast inference time, allowing the user to try out different parameters and test different results.

  The inference running time on our local environments took a few seconds to run. In our test environment, we used a dataset composed of hundreds of low-resolution, several seconds long clips and four different classifiers in our graph.

Weaknesses

- Tendency to repeat selected clip sequences

  In several cases the model showed a tendency to hang on to a small number of clips as anchors, and to repeat sub-sequences of clips which include the anchors. This is a problem that can be overcome by different methods of flattening the probability matrices. By forcing smaller differences between the maximal/minimal similarity, we can de-facto eliminate the anchors.

- Result may not always correspond with the user's preferences

  Depending on the user's input, a sequence, which corresponds with all of the user's requests, may not be available, thus the most-likely solution may not fall entirely within the user's given requirements.

Sample Results

Unfortunately, we cannot show an example of a video created by our algorithm. We can give a little taste by showing a screen capture from each clip (see appendix #1) to give a simple sense of the algorithm in action. We can compare the outputs of our results with the results of the random editor.

In the movie we generated using the algorithm, the brightness differences between scenes is very smooth, whereas the random clips do not keep a visual smoothness. Colors alternate more frequently. This result was achieved using aesthetic factors that tend more towards color consistency.

Bibliography and References

1. A Dialectic Approach to Film Form – Sergei Eisenstein – 1949
2. Using the Forest to See the Trees: A Graphical Model Relating Features, Objects, and Scenes – Murphy, Torralba, Freeman – 2003
3. FullBNT – [bnt.sourceforge.net]

Appendices

1. Sample results strips

A sequence generated by our algorithm:

A random sequence of clips:

2. Kuleshov's Experiment

3. Future Plans

1. More features/classifiers
   The most obvious step is to add more features and classifiers to the existing model. Extending and refining the quality measure can increase our ability to generate a better result.
2. Trim the clips (pick interesting points)
   Even though this is the primary work of a professional editor, for the scope of this project, we decided not the cut clips into sub-clips, only to concatenate them. Picking the interest point, the most suitable position in which to cut each clip is a different problem to study. Incorporating a solution to both problems can significantly improve the result.
3. Edit according to Soundtrack
   Another possible path to take, is editing according to a given soundtrack. An audio track can be analyzed, and the transition matrix can incorporate the information into the inference stage.
4. More than pairwise relations
   In our model only adjacent nodes affect one another directly (the rest affect only indirectly, through the neighboring nodes). Creating more direct relations, for example deciding that the 3rd node should affect the 1st may change the result. This is exactly the point where our project can transform from an automatic editing tool, to a "pencil" in the hands of an artist.
5. Learning the weights from existing films
   The pipeline we described works from the clips to a movie. We can do the opposite process, learning the weights and the probabilities from existing movies, to create a MovieWeaver representation of a certain genre of movies.
6. Transfer the learned model and create movies BBC-style, MTV-style, and more.
   If we have a representation as described in #6, we can transfer use the same model to create a movie in the given style, using a different set of clips. The result can be for example: "Grandma's birthday – MTV style".