Interactively Browsing Movies in terms of Action, Foreshadowing and Resolution

Stewart Greenhill, Brett Adams, Svetha Venkatesh

March 23, 2012

Abstract

We describe a novel video player that uses Temporal Semantic Compression (TSC) to present a compressed summary of a movie. Compression is based on *tempo* which is derived from film rhythms. The technique identifies periods of *action*, *drama*, *foreshadowing* and *resolution*, which can be mixed in different amounts to vary the kind of summary presented. The compression algorithm is embedded in a video player, so that the summary can be interactively recomputed during playback.

1 Introduction

Video media is being created and consumed on a scale never before seen. A recent survey shows that in the US alone, 12.7 billion videos were watched online during November 2008, an increase of 34% over the previous year (comscore.com). Video warehouse YouTube is ahead of both Google and Facebook on page views (alexa.com). The creation, propagation, and remixing of media is of the essence of Web 2.0, but currently, the consumption of video items is usually through browsing interfaces that are tied to the legacy of sequential-access tape: play, pause, rewind, fast forward, with a slider bar forming the only relative novelty.

In [2] we presented a browsing paradigm that fits with the online, interactive, and remixable context of much of the video media consumed today, termed Temporal Semantic Compression (TSC). It aimed to not only support the foraging behaviour common to today's media landscape, but take advantage of the opportunity presented by the technologies that enabled it: video is stored or delivered in the context of *computation* (server farms, desktops, or mobile devices).

TSC browsers enable the user to interactively compress and peruse a video non-linearly by resampling the displayed video according to a measure of the interest of each frame or shot. In short, they allow the user to leave out more or less of the boring bits interactively, with a simple gesture.

In the context of previous work, a TSC browser presents an interactive version of a kind of video *abstract* [4] or *skim* [5]. [4] used features like cuts, faces, dialogue, and events such as explosions to build a movie "trailer"; heuristics select important objects, dialogue and action without revealing the outcome of the movie. [5] used "film syntax" to select shots, and classified audio to identify dialogue. Constraint relaxation ensures the coherence of both video and audio, and different types of skims result by varying the objective function. In contrast, TSC uses *any* single-valued measure of interest, calculated automatically or manually for the duration of a video, and derived from any phase of a video's lifecycle–E.g., *Presence* of a particular actor or object, captured in pre-production or production [3]; *Tempo* [1] which is related to film rhythms, and other measures of Computational Media Aesthetics automatically calculated from the produced video; and *Popularity* measures calculated from view statistics collected by media warehouses like YouTube. The TSC algorithms are embedded within the browser itself giving the ability to fluidly vary the temporal detail even as a video is playing.

By way of introduction, consider the short video

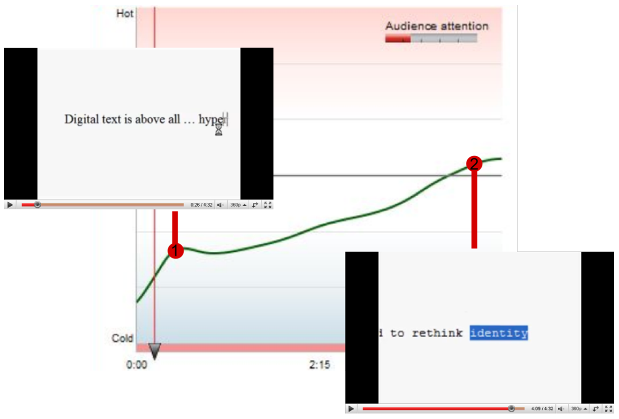


Figure 2: Popularity graph for video *Web 2.0 ... The Machine is Us/ing Us* with example frames near local maxima indicated, court. Prof. Michael Wesch.

Web 2.0 ... The Machine is Us/ing Us. The video is enormously popular, at over 10 million views, but its popularity can be analyzed in greater detail using YouTube's Hot Spots feature, which provides an estimate of the popularity of every second of the video relative to videos of the same length, pictured in Figure 2. A TSC browser can compress this video to an arbitrary length according to, say, local maxima in the popularity measure. In this case, as the video is compressed from its full length of over 4 minutes to just seconds, the portions of video that remain in the abstract contain its two most important assertions, that (1) Digital media's flexibility changes everything (left side of Figure 2), and (2) As a result, much of life needs to be rethought (right side of Figure 2). These constitute the video's turning point and climax, narratively speaking.

The previous incarnation of our browser offered control over some aspects of the summarisation, such as the balance between *action* and *drama*. However, it did not adequately handle the scenario where a user wants to gain an overview of a movie they have never seen. Here it is important that the system does not reveal too many of the outcomes of important episodes (i.e. "spoilers"). In [4] this is achieved by only using the first 80% of a movie. This avoids revealing the ending, but it ignores the fact that plots move in stages, and that important turning points occur throughout the movie. Instead, our approach uses the cyclic nature of a film's tempo to identify what should be included or not.

Tempo [1] measures the attention that the filmmaker asks of the audience, and can be computed from features like camera motion, sound energy, and shot length. High tempo usually indicates significant action, but changes in tempo are also important since they indicate dramatic events before and after the action. Increasing tempo often marks the suspenseful lead up to important events (*foreshadowing*), where decreasing tempo marks the outcome of these events (*resolution*).

This work extends the TSC concept by incorporating underlying interest measures and parameters that are specifically relevant to the domain of movies. The browser uses the notions of *action*, *drama*, *foreshadowing* and *resolution*, to create a version of a given movie in an interactively compressible manner. The result can be tailored to different usage scenarios: This interaction mechanism is useful for gaining a schematic of a movie the user has never seen, without resolutions, in the case where they might go on to watch it in its entirety; hunting for a remembered event in a given movie; or even sampling a remixed movie that has been seen. E.g., Rifftrax (rifftrax.com) provide humorous audio tracks that can be overlaid on existing movies. Viewing a compressed version of such a "value-added" movie will enable the user to see how the most iconic moments of the original movie have been spoofed.

To the authors' knowledge, the use of the filmic elements of action, drama, foreshadowing and resolution to enable an interactive browsing experience is novel. Given the amount of video browsing alluded to above, even small improvements to the browsing experience carry high significance. We note that not all of the long tail of online video content is of great length or high quality. However, the compression provided by TSC browsers can be of use for videos of the order of minutes, depending on the underlying interest function, and often video data is orphaned from existing metadata which would otherwise be of use in abstracting or navigating its content. Additionally, there will continue to be media of a quality that rises

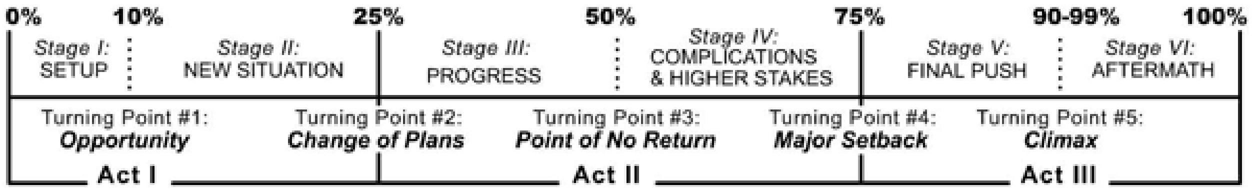

0%
10%
25%
50%
75%
90-99%
100%
Stage I: SETUP
Stage II: NEW SITUATION
Stage III: PROGRESS
Stage IV: COMPLICATIONS & HIGHER STAKES
Stage V: FINAL PUSH
Stage VI: AFTERMATH
Turning Point #1: Opportunity
Turning Point #2: Change of Plans
Turning Point #3: Point of No Return
Turning Point #4: Major Setback
Turning Point #5: Climax
Act I
Act II
Act III

Figure 1: Typical three act, six stage screenplay structure according to Michael Hauge [6].

to the top, both amateur and commercially created, such as the value adding referred to above.

2 Semantic Analysis

We begin by computing a set of features for each frame of video: The normalised difference between RGB colour histograms of the current and previous frame is calculated for use in shot segmentation. Camera motion parameters are calculated using block matching. The mean volume of audio over the frame duration is also computed.

Next, the video is analysed for structure and low-level semantics. Shot boundaries are identified by looking for frames with a difference of colour histograms above a threshold $T_s = 0.5$. If a shot boundary results in a shot of less than $T_L = 10$ frames, that shot is merged with the previous shot. This avoids over-segmentation in scenes involving transient changes in illumination like flashing lights, or lightning.

Next, we derive a *tempo* function, as described in [1]. Let F be a set of frame-level features. In this study, $F = \{pan, tilt, volume\}$ where *pan* and *tilt* are camera motion parameters, and *volume* is the mean audio volume. For each feature, we define the global mean μ_i and standard deviation σ_i. The tempo T is defined as:

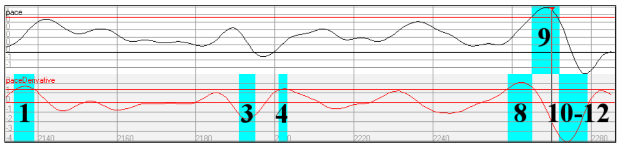
$$T(n) = \alpha(W(s(n))) + \sum_{i \in F} \frac{\beta_i(\mu_i(n) - \mu_i)}{\sigma_i}$$

Where $s(n)$ is the length of shot n, and $\mu_i(n)$ is the mean value of feature i over the duration of shot n. W is the shot normalisation function described in [1], which tapers linearly from 1 at 0 to 0 at the median shot length, then asymptotically approaches -1 at a shot length below which 95% of the shots lengths are distributed. α and β_i are weights for the individual features. Nominally, $\alpha = 1$, and $\beta_i = \frac{1}{3}$. The tempo function $T(n)$ is smoothed using a Gaussian filter ($\sigma = 3$) and the derivative $T'(n)$ is computed using a recursive filter. Note that filtering is done in the *shot* domain, not the time domain.

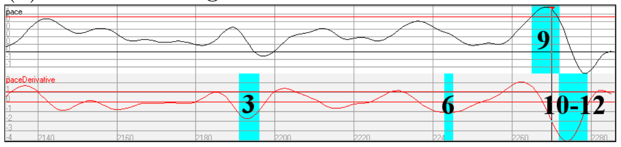
We define *semantic compression* as the process of using semantic information to control the amount of information presented to the viewer. The user defines a *compression factor* $0 < f \leq 1$, and the system selectively discards video frames so that if there were initially N frames, there are fN frames remaining after compression. A *compression function* alters the video time-base, mapping each frame in the compressed video to a frame in the original video. One approach is to vary the playback rate, frequently dropping single frames of video. Alternatively, we can drop contiguous sequences of frames, even entire shots, based on interest. This tends to preserve intelligibility and is our focus here.

Given a compression factor f, we compute the duration of the compressed video $\tau = fN$. Given an interest function E, we rank M shots in order of interest, $S_{E,1}, ..., S_{E,M}$. The compressed video is simply the sequence of shots satisfying $\sum_{i=1}^{k} dur(S_{E,i}) = \tau$, ordered according to their original relative position.

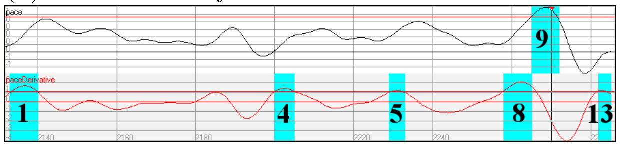
While tempo T can be used as an interest function, the resulting video would be dominated by fast-paced action sequences. Regions of changing tempo are dramatically significant too since they mark important transitions in the narrative. Thus, we use T as an interest function for *action*, and $|T'|$ for *drama*. An action-drama balance factor $0 \leq \delta \leq 1$ controls how much of the final duration is allocated to each inter-



(a) Foreshadowing + Resolution



(b) Resolution only



(c) Foreshadowing only

Figure 3: Each chart shows T (top) and T' (bottom) for the final few minutes of "The Matrix". Shaded areas show sequences of shots included based on action (top) and drama (bottom).

est function: $\delta\tau$ for *action*, and $(1 - \delta)\tau$ for *drama*. *Drama* can be subdivided into *foreshadowing* ($T' > 0$, tempo increasing) and *resolution* ($T' < 0$, tempo decreasing). We can switch these elements on and off by filtering according to the sign of T', or we can use a balance parameter to specify a mixture (as with δ). One final parameter is the *scene limit* γ, which defines the maximum length of any sequence as a proportion of the total duration. This prevents unusually long shots from dominating the result at high levels of compression.

Action, foreshadowing and resolution can be selectively mixed to vary the style of the result. For example, if the aim is present a trailer-like "teaser", we favour foreshadowing and action over resolution. If the aim is to summarise the story, we favour action and resolution. Note that while feature extraction is a potentially time-consuming, the compression function can be calculated almost instantaneously, allowing it to be interactively recomputed during playback.

Story Elements	R—F	R	F
1 Agents pursue Neo	F		F
2 Agent Smith kills Neo			
3 Trinity declares love	R	R	
4 Trinity revives Neo	F		F
5 Neo fights agents			F
6 Neo fights agents		R	
7 Neo kills Agent Smith			
8 Neo escapes Matrix	F		F
9 Machines breach cabin	A	A	A
10 Morpheus uses EMP	R	R	
11 Neo wakes up	R	R	
12 Neo's ultimatum	R	R	
13 Neo flys			F

Figure 4: Story elements in "The Matrix" selected with action (A), *and* resolution (R), foreshadowing (F), or both (R—F).

3 Results

To demonstrate, we compressed the movie "The Matrix" to 10% ($f = 0.1$, $\sigma = 3$, $\gamma = 0.08$, $\delta = 0.15$). Figure 3 shows T (top) and T' (bottom), and the shaded areas indicate included shots for combinations of foreshadowing and resolution. In the story, Neo battles the Agents in the virtual world of the Matrix, while in the real world he and his comrades are attacked by the Machines (see Figure 4). Initially, Neo is defeated by the Agents (1-3) but is then revived and triumphs against them (4-8), escaping in time to prevail against the Machines (9-11).

The high compression level means that many significant scenes must be dropped, including several important events (2, 7). While the lengthy fight scenes are not included, the significant events that bracket the major sections are retained. It is not until after the climactic scenes (9–10) that we realise whether Neo has survived or not. This is indicated in the relatively low paced following scene (11) which is identified as a resolution shot. Foreshadowing shots introduce the major sections (1, 4, 8), and include the final scene (12) which hints at Neo's future exploits.

Reducing the compression level enables additional scenes to be included, but also increases the length of existing scenes. To demonstrate this, we compressed the movie "Shrek" using $\sigma = 3$, $\gamma = 0.05$, $\delta = 0.15$ and varying the compression factor: $f = 0.05$, 0.1, and 0.2. Figure 5, shows the major plot points that are included at each level. We compare the identified elements to the typical six-stage, three act plot structure as outlined by screenwriting guru Michael Hauge([6], Figure 1). In Stage 1 (Setup: 1–2) we meet Shrek, an ogre who happily lives alone in a swamp. In Stage 2 (New Situation: 3–14) Shrek's home is invaded by fairytale creatures by decree of Lord Farquad. Shrek meets Donkey, and they resolve to journey to Duloc to put matters right (Part A: 3–8). Farquad decides to marry Princess Fiona, and as Shrek arrives at Duloc he is sent on a quest to rescue her from the dragon's keep (Part B: 9–14). In Stage 3 (Progress) Shrek rescues the princess, but is revealed as an ogre and not the handsome saviour she was expecting (15–19). In Stage 4 (Complications & Higher Stakes: 20–27) they journey back toward Duloc, and we learn that Fiona has been enchanted, and becomes an ogre herself at sunset. Despite signs of romance, a misunderstanding causes Shrek and Fiona to quarrel and Fiona leaves to marry Farquad. Shrek and Donkey argue, and the outcome looks grim. In Stage 5 (Final Push: 28–25) Donkey and Shrek are reconciled. Donkey summons the dragon and they fly to Duloc to stop the wedding. In the ensuing battle, Farquad is eaten by the dragon and Fiona discovers her true form. In Stage 6 (Aftermath: 36) Shrek and Fiona are married.

At extreme compression levels ($f = 0.05$) we are left with only the skeleton of the story. This includes important parts of each major stage, but also most of the important turning points: 8 and 14 (Change of Plans), 18–19 (Point of No Return), and 33–34 (Climax). Some important details are missing such as the back story around Farquad's quest (9–10), and parts around the "Major Setback" turning point. In the movie this period of melancholy is highlighted by a lengthy montage accompanied by Leonard Cohen's "Hallelujah". This turns out to be unexceptional in terms of either Action or Drama, even at lower compression levels ($f = 0.2$). Further details of included shots can be seen in the example videos, online at (`http://computing.edu.au/~stewartg/files/browser`).

While the generated summary captures key elements of the plot, it has problems. There are two types of "errors" we can make using this tempo analysis to identify significant plot events. Errors of *omission* occur when significant events are emphasised using low tempo. As we have seen this includes the setback scenes in Shrek, but also the death of Neo and Agent Smith in The Matrix. Errors of *inclusion* occur when insignificant events are treated with high tempo. An example of this is Fiona's duet with the bluebird (Shrek: 21), which is not only loud but includes rapid cuts between the two characters. While it is here used for comedic effect, the same sort of pattern can also occur in dialogue sequences. Whether such sequences are significant or not depends on the details of the story. While we did not attempt to identify dialogue sequences, this would be relatively simple to do by looking for alternating sequences of similar shots (eg. using colour histograms as was used for shot segmentation).

4 Conclusion

We described a browser that uses Temporal Semantic Compression (TSC) to present a compressed abstract of a movie. Compression is based on *tempo* derived from film rhythms. The compression algorithm identifies periods of *action*, *drama*, *foreshadowing* and *resolution*, which can be mixed in different amounts to vary the kind of summary presented. Once features have been extracted, the compression function can be rapidly recomputed even during playback.

References

[1] B. Adams, C. Dorai, and S. Venkatesh. Towards automatic extraction of expressive elements from motion pictures: Tempo. *IEEE Transactions on Multimedia*, 4(4):472–481, December 2002.

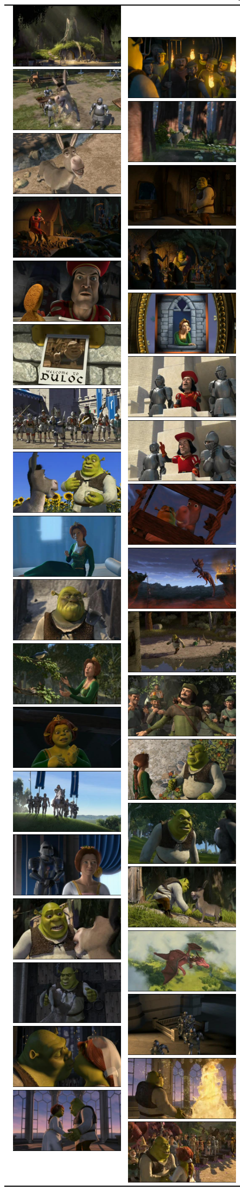
[2] B. Adams, S. Greenhill, and S. Venkatesh. Temporal semantic compression for video browsing. In *Int. conf on Intelligent user interfaces, Gran Canaria, Spain*, p. 293–296, 2008.

[3] L. Hardman, Z. Obrenovic, F. Nack, B. Kerherv, and K. Piersol. Canonical processes of semantically annotated media production. *Multimedia Syst.*, 14(6):327–340, 2008.

[4] R. Lienhart, S. Pfeiffer and W. Effelsberg. Video abstracting. *Communications of the ACM*, 40(12):54–63, 1997.

[5] H. Sundaram, L. Xie, and S. Chang. A utility framework for the automatic generation of audio-visual skims. In *ACM Multimedia*, p. 189–198, 2002.

[6] C. Vogler and M. Hauge. *The Hero's 2 Journeys*. 2004. `http://www.screenplaymastery.com/`.



Story Elements	20	10	5
1 Home in the swamp	X	X	X
2 Villagers	X	X	X
3 Donkey flys	F		
4 Donkey escapes	A	A	A
5 Donkey meets Shrek	R		
6 Wolf inside	F		
7 Creatures invade swamp	A	A	A
8 Shrek leaves for Duloc	F	F	F
9 Farquad interrogates	X	X	
10 You have chosen ...	R		
11 Arrive at Duloc	R	R	
12 Farquad's competition	F	F	F
13 Fight with knights	A	A	A
14 Farquad explains quest	R	R	
15 Ogres have layers	R		
16 Rescue at castle	X	X	X
17 Fiona in the tower	F	F	F
18 Escape from dragon	X	X	R
19 Shrek revealed to Fiona	R		
20 ... Farquad's stature	R	R	R
21 Fiona finds breakfast	X	X	X
22 Robin Hood	X	X	X
23 Fiona's curse revealed	X	X	X
24 Shrek & Fiona quarrel	F		
25 Farquad claims Fiona	A		
26 Shrek & Donkey argue			
27 Hallelujah			
28 Shrek & Donkey friends	F		
29 We'll never make it	F	F	F
30 Dragon ride to Duloc	X	X	R
31 There's a line...	X	X	X
32 Guards!	X	X	X
33 True love's first kiss	F	F	F
34 Fiona transforms	A	A	A
35 Loves true form	R	R	
36 Wedding	X		

Figure 5: Story elements detected in "Shrek" at varying levels of semantic compression. Tags indicate action (A), resolution (R), foreshadowing (F), or a mixture of at least two types (X).