Constrained Utility Maximization for Generating Visual Skims

Hari Sundaram *Shih-Fu Chang*
Dept. Of Electrical Engineering, Columbia University,
New York, New York 10027.
Email: {sundaram, sfchang}@ctr.columbia.edu

Abstract

In this paper, we present a novel algorithm to generate visual skims, that do not contain audio, from computable scenes. Visual skims are useful for browsing digital libraries, and for on-demand summaries in set-top boxes. A computable scene is a chunk of data that exhibits consistencies with respect to chromaticity, lighting and sound. First, we define visual complexity of a shot to be its Kolmogorov complexity. Then, we conduct experiments that help us map the complexity of a shot into the minimum time required for its comprehension. Second, we analyze the grammar of the film language, since it makes the shot sequence meaningful. We achieve a target skim time by minimizing a sequence utility function. It is subject to shot duration constraints, and penalty functions based on sequence rhythm, and information loss. This helps us determine individual shot durations as well as the shots to drop. Our user studies show good results on skims with compression rates up to 80%.

1. Introduction

This paper deals with the problem of systematic generation of visual skims by condensing scenes via visual analysis. At present, our skims do not incorporate audio. The problem is important because unlike the static, image based video summaries [11], video skims preserve the dynamism of the original audio-visual data. Applications of visual skims include: (a) on demand summaries of the data stored in set-top boxes (b) in interactive television (c) browsing of digital archives (d) fast-forwarding through streaming video, while maintaining the original frame rate.

There has been prior research on generating video skims. In the Informedia skimming project [1], important regions of the video were identified via a TF/IDF analysis of the transcript. Additionally, they use face detectors and motion analysis for additional cues. The MoCA project [7] worked on automatic generation of film trailers. They used heuristics on the trailers, along with a set of rules to detect certain objects (e.g. faces) or events (e.g. explosions). Work at Microsoft Research [6] dealt with informational videos; there, they looked at slide changes, user statistics and pitch activity to detect important segments.

Clearly skims differ based on the need to preserve certain semantics [2], [7], the specific domain [6] and by user needs (e.g. searching for particular content). This work focuses on three specific areas that were not investigated in earlier work: (a) the relationship between the length of a shot in a film and its comprehension time (b) analyzing the syntactical structure in the film (c) defining the notion of a sequence utility function.

We define the visual complexity of the shot to be the its Kolmogorov complexity. This measure can be bounded by using the Lempel-Ziv compression algorithm. We then conduct a series of experiments that measure the comprehension time of randomly chosen shots from six films, with respect to four questions. These are at a generic semantic level (e.g. who? where? what? why?). The timings are then analyzed to generate a bound on minimum time that must be allocated to a shot, for it to remain comprehensible.

We also investigate the use of film-syntax for reducing the content of the scene. Film-syntax refers to the arrangement of shots by the director to give meaning to the shot sequence. Examples include, specification of (a) scale (b) duration (c) order of shots, amongst many others [8]. We investigate two simple rules governing the duration of the phrase and dialogues, for content reduction. We also show how to guard against errors in shot-detection, in our framework. Then, we define a sequence utility function, and determine shot duration of each shot in the target skim using a maximization procedure subject to shot duration constraints and penalty functions based on sequence rhythm and information loss. The user studies show that the our scheme works well for compression rates up to 80%, in addition to showing statistically significant improvements over earlier algorithms [10].

The rest of this paper is organized as follows. In the next section, we review the computable scene idea. In section 3, we derive the relationship between comprehension time and complexity. In section 4, we show how to exploit film syntax for scene condensation. We introduce the maximization procedure in section 5, we present experimental results in section 6, discuss the choices made in the algorithm in section 7, and present the conclusions in section 8.

2. Computable Scenes

A *computable-scene* [9] is defined to be a chunk of audio-visual data with consistent chromaticity, lighting and ambient sound. Constraints on computable scenes stem from camera arrangement rules in film making and from the psychology of audition. We use these constraints along with analysis of five hours of commercial film data to come up with two broad categories of computable scenes. (a) N-type: show a long-term consistency with regard to chromatic composition, lighting conditions and sound. N-type scenes typically consist of shots from the same physical location. (b) M-type: these are characterized by widely different visuals that create a unity of theme by their arrangement and also have a long-term consistency to the audio.

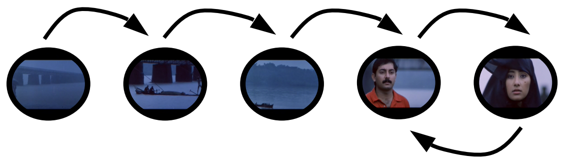

Figure 1: A progressive scene followed by a dialogue.

In this paper we focus on scenes comprising two N-type scene structures. Progressive: a linear progression of visuals without any repetitive structure (the first part of figure 1, showing a sequence of shots of the river, is a progressive scene). Dialog: a simple repetitive visual structure amongst shots. A discussion on M-type scenes can be found in [9]. In [9], we demonstrate a framework for detecting computable scenes and dialogs. The best results: scene detection: 88% recall and 72% precision, dialog detection: 91% recall and 100% precision.

3. Visual Complexity

In this section, we discuss the relationship between visual complexity of an image and its time for comprehension.

3.1 Insights: film making and human learning

In film-making, there is a relationship between the size[1] of the shot and its apparent time (i.e. time perceived by the viewer).:

"Close-ups seem to last relatively longer on the screen than long shots. The content of the close up is immediately identified and understood. The long shot on the other hand, is usually filled with detailed information which requires eye-scanning over the entire tableau. The latter takes time to do, thus robbing it of screen time" [8].

[1] The size (long/medium/close-up/extreme close-up) refers to the size of the objects in the scene relative to the size of the image

Recent results in experimental psychology [4] indicate the existence of an empirical law: the subjective difficulty in learning a concept is directly proportional to the Boolean complexity of the concept (the shortest prepositional formula representing the concept), i.e. to its logical incompressibility. Clearly, there is empirical evidence to suggest a relationship between visual "complexity" of a shot and its comprehensibility.

3.2 Kolmogorov complexity

We define the visual complexity of an shot to be its Kolmogorov complexity. Let $\boldsymbol{x}$ be a finite length binary string of length n. Let $\mathbf{U}$(p) denote the output of an universal Turing machine[2] $\mathbf{U}$ when input with program p. Then:

$$K_U(x \mid n) \triangleq \min_{p:U(p)=x} l(p), \qquad \text{<1>}$$

where, *l(p)* is the length of the program *p,* and *n* is the length of the string $\boldsymbol{x}$ and where $K_U(x \mid n)$ is the Kolmogorov complexity of x given n. Hence, the Kolmogorov complexity of $\boldsymbol{x}$, with respect to an universal Turing machine $\mathbf{U}$ is the length of the shortest program that generates $\boldsymbol{x}$. The Kolmogorov complexity of an arbitrary string $\boldsymbol{x}$ is non-computable due to the non-existence of an algorithm to solve the halting problem [3], [5]. Hence, we must generate a reasonable upper bound on Kolmogorov complexity. Lempel-Ziv encoding is a form of universal data coding that doesn't depend on the distribution of the source [3]. We can easily show the following lemma by using results in [3], [5]. The proof has been omitted for the sake of brevity.

Lemma 1: *Let $\{X_i\}$ be a stationary, ergodic process over a finite discrete sized alphabet. Let $l_{LZ}(\mathbf{X})$ be the Lempel-Ziv codeword length of a string $\mathbf{X}$, where $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$. Then,*

$$K_U(X \mid n) \le l_{LZ}(X) + c,$$
$$\lim_{n\to\infty} \frac{1}{n} l_{LZ}(X) \to \frac{1}{n} K_U(X \mid n). \qquad \text{<2>}$$

Hence, we can use the Lempel-Ziv compression algorithm to upper bound the visual complexity of a shot.

3.3 Generating estimates of complexity

The complexity is estimated using a single key-frame[3]. Representing each shot by its key-frame is reasonable since

[2] An universal Turing machine **U** is a Turing machine that can imitate the behavior of any other Turing machine **T**. It is a fundamental result that such machines exist and can be constructed effectively [5].

[3] We choose the 5th frame after the beginning of the shot, to be its key-frame. We acknowledge that there are other more sophisticated strategies for choosing key-frames.

our shot detection algorithm [12], is sensitive to changes in color and motion. In this paper, we tested two lossless compressors — *gzip* [13] based on the Lempel-Ziv (LZ77) algorithm and *bzip2* [14] based on the recent Burrows-Wheeler transform (BWT) [1]. Algorithms based on the BWT have shown greater compression rates than those based on the well known Lempel-Ziv algorithm. In our experiments, the size of the *bzip2* sequence was typically 5 ~ 10 % smaller than the corresponding *gzip* sequence. *Bzip2* also produced a shorter sequence than *gzip* for every image in our corpus. Hence we estimated Kolmogorov complexity using *bzip2*.

The output of a compressor operating on an image *I* is a *program*. This program, when input to a universal Turing machine emulating the compressor will decompress the program to output the original image *I*. Hence the length of the BWT sequence for the image *I* is just the size of the bzip2-ped file in bytes. We normalize the complexity by dividing it by the image size (this is just # bits / pixel).

3.4 Complexity and comprehension time

We conducted our experiments over a corpus of over 3600 shots from six films. A shot was chosen at random (with replacement) and then its key-frame presented to the subject (the first author). Then, we measured the time to answer the following four questions (in randomized order), in an interactive session: (a) who: [man / woman / couple / people], (b) when: [morning / evening / afternoon], (c) what: [any verb e.g. looking, walking], (d) where: [inside/ outside] [1]. The subject was expected to answer the questions in minimum time *and* get all four answers right. This prevented the subject from responding immediately. We conducted ten sessions (to avoid fatigue), where the subject was questioned on 100 key-frames. In the end, we had the reaction times to 883 unique shots (we averaged the reaction times over duplicates). For each question, "none" or "ambiguous" was an acceptable answer. The response times in such cases are valid since they measure the time to reach a decision.

3.5 Analysis of comprehension time

We generate histograms of the average comprehension time (i.e. the average of the times to answer who? where? what? and when?) after discretizing the complexity axis. The *lower-bound* on the comprehension time is generated by determining a least squares fit to the minimum time in each histogram. The distribution of times in each histogram slice, *above* the minimum time, is well modeled by a Rayleigh distribution. By using the 95th percentile cut-

[1] Questions such as "How?" or "Why?" were not used in the experiment since they cannot be answered by viewing just one image. Such questions need an extended context (at least a few shots) for an answer.

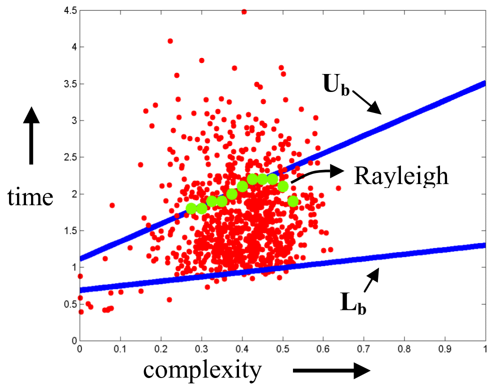

Figure 2: Avg. comprehension time (sec.) vs. normalized complexity (x-axis) showing comprehension (upper/lower) bounds. It also shows the Rayleigh (95th percentile) bounds.

off for each histogram we get an estimate of the *upper-bound* on the comprehension time. The resulting bounds are shown in figure 1. The equations for the lines are as follows:

$$U_b(c) = 2.40c + 1.11,$$
$$L_b(c) = 0.61c + 0.68, \qquad <3>$$

where c is the normalized complexity and U_b and L_b are the upper and lower bounds respectively, in sec. The lines were estimated for $c \in [0.25, 0.55]$ (since most of the data lies in this range) and then extrapolated. Hence, given a shot of duration t_o and normalized complexity c_S, we can condense it to at most $U_b(c_S)$ sec by removing the last t_o - $U_b(c_S)$ sec. Let us assume that we want to reduce a shot by 75%. If the target time of the shot is less than the upper bound U_b for that shot, we use the upper bound. If the original length of the shot is less than the upper bound it is *not* reduced any further.

Note that equation <3> indicates that both the lower and upper bounds *increase* with complexity, as they intuitively ought to. The upper bound comprehension time is actually a conservative bound. This is because of two reasons: (a) the shots in a scene in a film are highly correlated (not i.i.d) and (b) while watching a film, there is no *conscious* attempt at understanding the scene.

4. Film Syntax

In this section we shall give a brief overview of "film syntax." Then, we shall discuss its utility in films and then give syntax based reduction schemes for two syntactic elements. Finally, we show a technique to deal with shot-detection errors.

4.1 Defining film syntax

The phrase film syntax refers to the specific arrangement of shots so as to bring out their mutual relationship [8]. In practice, this takes on many forms (chapter 2, [8]) : (a) minimum number of shots in a sequence (b) varying the shot duration, to direct attention (c) changing the scale of the shot (there are "golden ratios" concerning the distribution of scale) (d) the specific ordering of the shots (this influences the meaning). These syntactical rules lack a formal basis, and have been arrived at by trial and error by film-makers. Hence, even though shots in a scene only show a small portion of the entire setting at any one time, the syntax allows the viewers to understand that these shots belong to the same scene.

4.2 Why should we use film syntax?

Let us contrast shots with words in a written document. Words have more or less fixed meanings and their position in a sentence is driven by the grammar of that language. However, in films it is the phrase (a sequence of shots) that is the fundamental semantic unit. Each shot can have a multitude of meanings, that gets clarified by its relationship to other shots.

The Informedia [1] and the MoCA [7] projects use object detectors (e.g. face detectors etc.) over the entire video, for selecting important segments. In the Informedia project [1] the audio was selected by first performing a TF-IDF analysis of the transcript and then selecting the complete phrase surrounding the highly ranked words. An object detector based approach (e.g. Informedia, MoCA) to skims, for films, at a conceptual level, makes the analogy "shots as words." However, this is in contrast to the way film-makers create a scene, where the syntax provides the meaning of the shot sequence. Hence, while condensing films, we must honor the film syntax.

The Informedia and the MoCA projects analyze data over the *entire* video. However, they do not perform scene level analysis for skim generation. In our current work, we analyze the data *within* one scene. In future work, we plan on utilizing the interesting syntactical relationships amongst scenes that exist in the video [8], for compression.

4.3 Syntax rules for shot removal

According to the rules of cinematic syntax [8], a phrase must have at least three shots. *"Two well chosen shots will create expectations of the development of narrative; the third well-chosen shot will resolve those expectations."* Sharff [8] also notes that depicting a meaningful conversation between m people requires at least $3m$ shots. Hence in a dialogue that shows two participants, this rule implies that we must have a minimum of six shots. Let us assume that we have a scene that has k shots. Then, we perform three types of syntax reductions (break points based on heuristics) based on the on the number of shots k (Table 1). The number and the location of the dropped shots depend on k and the syntax element (i.e. dialog or progressive). In the following discussion, we use a fictional film with a character called Alice.

Table 1: Three types of syntax reductions that depend on the element (dialog/progressive) and the number of shots k.

Element	Min. phrase length	Breakpoints for each type		
		I	II	III
Dialog	6	$k \le 15$	$15 < k < 30$	$k \ge 30$
Progressive	3	$k \le 6$	$6 < k < 15$	$k \ge 15$

A phrase is a sequence of shots designed to convey a particular semantic. It is reasonable to expect that the number of phrases in a scene, increase with the number of shots. For short scenes (type I reduction) we assume that there is a single phrase, containing one principal idea, in the scene. For example, the director could show Alice, walking back to her apartment, in a short scene.

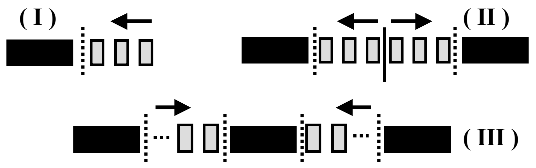

Figure 2: Three syntax reduction mechanisms. The black boxes are the minimal phrases and will not be dropped, while the gray shots can be dropped.

In scenes of medium duration (type II reduction) we assume that there are at most two phrases. For example, <1st phrase>: Alice could be shown entering her room, switching on the lights, and be shown thinking. <2nd phrase>: then, she is shown walking to the shelves looking for a book, and is then shown with the book. We assume that scenes of long duration, (type III reduction) contain at most three phrases. Modifying the previous example — <1st phrase>: Alice is shown entering the room, <2nd phrase>: she is shown searching for the book, <3rd phrase>: she walks with the book to her desk and makes a phone call. Hence, the reduction attempts to capture the phrase in the middle and the two end phrases.

In type I reduction, (figures 2 (I)–(III)) we drop shots from the right, since the director sets up the context of the scene using the initial shots. In type II, we expect an initial context, followed by a conclusion. Here, we start dropping shots from the middle, towards the ends. In type III, the scene is divided into three equal segments, and shots are dropped from the two interior segment boundaries.

Unlike written text, there are no obvious visual [1] "punctuation marks" in the shots to indicate a "phrase

[1] However, we may be able to detect a "phrase change" within a scene, by analyzing the audio and the transcript.

change." Hence our syntax reduction strategy, which will capture the phrases in scenes of short and medium duration, may cause error in scenes of long duration.

4.4 Incorporating shot detector uncertainty

In this section we show how the statistical uncertainty in shot detection algorithms affect the rules of syntax. Practical shot-detection algorithms have misses and false alarms, each of which has a different effect on skim generation. Shot misses can cause our shot condensation algorithm (ref. section 3.5) to remove a missed shot. False alarms will cause our syntax based reduction algorithm to remove more shots than permitted by the minimum requirements. In this work, we only focus on false alarms. Since false alarms manifest themselves as an increase in the number of shots detected, we can compensate for them.

Table 2: Shot detector variations. Each row show the in order: the motion and color parameter thresholds, recall, precision, prob. of false alarm $P(F_a)$, $P(F_a \mid N_d)$: prob. of false alarm give a non-dialog scene. The 97.5% confidence upper bound for $P(F_a \mid N_d)$.

Motion	Color	Recall	Precision	$P(F_a)$	$P(F_a \mid N_d)$	97.5%
L	L	0.97	0.72	0.28	0.55	0.64
L	M	0.96	0.71	0.29	0.55	0.65
L	H	0.96	0.71	0.29	0.55	0.65
M	**L**	**0.92**	**0.93**	**0.07**	**0.14**	**0.23**
M	M	0.91	0.94	0.06	0.12	0.21
M	H	0.91	0.94	0.06	0.12	0.21
H	L	0.90	0.94	0.06	0.11	0.19
H	M	0.86	0.95	0.05	0.10	0.18
H	H	0.84	0.95	0.05	0.10	0.18

We conducted a test with nine variations of our shot detection algorithm [12]. The algorithm had two adjustable thresholds for motion and color, and each parameter was set to three possible values: {low, medium, high}. Table 2 shows the results of the tests. We used 112 shots from four scenes, each from a different film. The test set had 54 dialog shots and 58 non-dialog (i.e. progressive) shots. The shot detectors had no false alarms in the section which had dialogs. Hence we are interested in the probability of false alarm given a non-dialog scene: $P(F_a \mid N_d)$. We use standard statistical techniques to compute the 97.5% confidence upper bound for $P(F_a \mid N_d)$. In this current work we use the following shot detector: {motion: M, color: L}. The table implies that for this shot detector, $P(F_a \mid N_d) \leq 0.23$, with 97.5% confidence, over unseen data sets.

The upper bound on $P(F_a \mid N_d)$ is used to modify the minimum number of shots that must remain in a progressive scene. If the progressive scene contains N shots, with m minimum number of shots (from the rules in section 4.3), then m is modified as:

$$m' = m + \lfloor N \square P(F_a \mid N_d) + 0.5 \rfloor \qquad <4>$$

where m□ is the new minimum $\lfloor \ \rfloor$ is the floor function. function. The second part of equation <4>, is just the expected number of false alarms. For example, in a scene with type II reduction, assume that N = 12, then, m = 6 and Δm = 3. Hence the minimum number of shots to be retained is 9. Modifying the minimum number of shots to drop ensures that we do not violate the minimum shot requirements of the syntactical elements of the scene.

5. Skim Generation Constraints

In this section, we derive the shot utility function using constraints on comprehension due to viewing time and shot complexity. We discuss penalty functions for skim rhythm and for dropping shots. We formulate a constrained minimization problem to determine the duration of each shot in sequence as well as the shots to be dropped, for a given target skim duration.

In the sections that follow we assume the following notation. N is the total number of shots in the original sequence, T_o is the original duration of the sequence, the target duration is T_f, $t_{o,n}$ represents the original duration of shot n in the scene. Define indicator sequence $\phi(n) = 1$ iff. n^{th} shot is present in the condensed scene. Define $N_\phi = \square\, \phi(n)$, the number of shots in the condensed scene.

5.1 The need for an utility function

The shot utility function, models the comprehensibility of a shot as a continuous function of its duration and its visual complexity. This idea is connected to the results in section 3.5 in following way. Let us assume for the sake of definitiveness, that we have a 10 sec. shot of complexity 0.5. Then the upper bound duration U_b = 2.23 sec. We have argued that that if we reduce the shot duration to its upper bound, then there is a high probability that it will still be comprehensible. However, the results in section 3.5 do not tell us how the comprehensibility of a shot *changes* when we decrease its duration. Hence the need for a shot utility function. The sequence of shots in a scene are assumed to be i.i.d. allowing us to model the utility of a shot independently of other shots.

5.2 Defining the utility of a shot

The non-negative utility function of a shot $S(t, c)$, where t is the duration of the shot and c is its complexity, must satisfy the following constraints:

1. For fixed c, $S(t, c)$ must be a non-decreasing function of the duration t. i.e. $\forall\, t_1 \leq t_2, S(t_1, c) \leq S(t_2, c)$. This is intuitive

since decreasing the shot duration by dropping frames from the end of the shot (section 3.5), cannot increase its comprehensibility.

2. $\forall t\ S(t,0)=0, S(t,1)=0$. This is because complexity c = 0 implies the complete absence of any information, while c = 1 implies that the shot is purely random.

We model the shot utility function to be a bounded, differentiable, separable, concave function:

$$S(t,c) = \beta c(1-c)\cdot(1-\exp(-\alpha t)). \qquad <5>$$

The exponential is due to the first constraint and the fact that the utility function is assumed to bounded. Symmetry with respect to complexity is again reasonable and the functional form stems from second constraint and concavity. We wished to avoid choosing a more general separable model (e.g. higher order polynomials) satisfying the constraints to avoid the possibility of over fitting.

The utility function for the sequence of shots is the sum of the utilities of the individual shots. This follows from the i.i.d. assumption.

$$U(\vec{t},\vec{c},\phi) = \frac{1}{N_\phi} \sum_{i:\phi(i)=1} S(t_i, c_i) \qquad <6>$$

where, $\vec{t}: t_0, t_1 \dots t_N$ and $\vec{c}: c_0, c_1 \dots c_N$. represent the durations and complexities of the shot sequence.

5.3 Rhythm penalty

The original sequence of shots have their duration arranged in a specific proportion according to the aesthetic wishes of the director of the film. Clearly, while condensing a scene, it is desirable to maintain this "film rhythm." For example, in a scene with three shots of durations 5 sec. 10 sec. and 5 sec. maintaining the scene rhythm would imply that we preserve the ratios of the duration (i.e. 1:2:1) of the shots. We define the rhythm penalty function as follows:

$$R(\vec{t},\vec{t}_o,\phi) = \sum_{i:\phi(i)=1} f_{o,i} \ln\left(\frac{f_{o,i}}{f_i}\right),$$
$$f_{o,i} = \frac{t_{o,i}}{\sum_{i:\phi(i)=1} t_{o,i}}, f_i = \frac{t_i}{\sum_{i:\phi(i)=1} t_i}. \qquad <7>$$

where R is the penalty function, and where, t_i is the duration of the ith shot in the current sequence, while $t_{o,i}$ is the duration of the ith shot in the original sequence. The ratios are recalculated with respect to only those shots that are not dropped, since the rhythm will change when we drop the shots. The penalty function is the familiar Kullback-Liebler distance function [3].

5.4 Shot dropping penalty

We deem the penalty due to dropping a shot to be proportional to its utility evaluated at its lower bound. This is because the lower bound represents the minimum amount time required to understand the shot. The sequence penalty $P(\phi)$ given the indicator sequence ϕ, is defined as follows:

$$P(\phi) = \frac{1}{N} \sum_{i:\phi(i)=0} \lambda(t_{p,i})\cdot S(t_{Lb,i}, c_i),$$
$$\lambda(t_{p,i}) = \begin{cases} 1 & t_{p,i} \geq t_{Ub,i} \\ \dfrac{t_{p,i}-t_{Lb,i}}{t_{Ub,i}-t_{Lb,i}} & t_{Lb,i} \leq t_{p,i} < t_{Ub,i} \\ 0 & t_{p,i} < t_{Lb,i} \end{cases} \qquad <8>$$

where, λ modulates the shot utility, $t_{p,i}$ is the proportional time for shot i i.e. $t_{o,i}\cdot T_f / T_o$, $t_{Lb,i}$ and $t_{Ub,i}$ are the lower and upper time bounds for shot i, and $S(t,c)$ is the shot utility function. The intuition for the modulation function λ, comes from observations in an earlier user study [10]. Users do not seem to mind dropped shots when average shot duration of the shots retained in the condensed sequence is large. However, when the average shot duration in the condensed sequence is close to the upper bound, they prefer to see the whole sequence.

5.5 The sequence objective function

We now define the objective function $O(\vec{t},\vec{c})$ to determine the duration and number of shots allowed in the sequence given the target duration T_f as follows:

$$O(\vec{t},\vec{c},\phi) \equiv 1 - U(\vec{t},\vec{c},\phi) + \gamma_o R(\vec{t},\phi) + \gamma_1 P(\phi) \qquad <9>$$

where γ_o and γ_1 are constant weights. Then, the individual shot durations and the indicator sequence is determined in the following way:

$$(\vec{t}_{opt}, \phi_{opt}) = \arg\min_{\vec{t},\phi} O(\vec{t},\vec{c},\phi)$$
subject to:
$$t_{L_b,i} \leq t_i \leq t_{o,i} \quad i:\phi(i)=1,$$
$$\sum_{i:\phi(i)=1} t_i = T_f, \qquad N_\phi \geq N_o, \qquad <10>$$

where, N_o is minimum number of shots to be retained in the scene. This value is obtained from Table 1. Note that $\vec{c}, \vec{t}_{Lb}, \vec{t}_{Ub}, \vec{t}_o$ and $\vec{t}_p$ are constants during the minimization.

Also, the shots can be dropped only in a constrained manner using the rules in section 4.3. Due to constraints on N_ϕ, there are some target times not achievable with our constraints. Then the algorithm will generate a best-effort skim, with maximal number of shots dropped, with each retained shot reduced to its lower bound.

6. Experiments

The scenes used for creating the skims were from four films: *Blade Runner (bla), Bombay (bom), Farewell my Concubine (far), Four Weddings and a Funeral (fou)*. The films were chosen for their diversity in film-making styles. While we arbitrarily picked one scene from each film, we ensured that each scene had one dialog segment. We detected shots using the algorithm to be found in chapter 2, [12], and with the shot-detector parameters {motion: M, color: L} (ref. Table 2).

We used two different skim generation mechanisms: the algorithm presented in this paper and the algorithm created in an earlier work [10]. Briefly, in the earlier algorithm, we attempted reduction to the target time by proportionately reducing the duration of each shot. If the target time could not be met in this manner, we would then drop shots according to rules of syntax. In that algorithm, we did not incorporate shot detector uncertainty, nor did we define the notion of sequence utility.

We created two skims (one from each algorithm) at each of the four different compression rates (90%, 80%, 70% and 50%). Ideally, we would liked to have created one skim per compression rate, per film. However, this would have meant that the user would have to rate 32 skims, an exhausting task. We ordered the scenes according the number of shots, and the scene with the maximum number of shots was compressed at 90%; the scene with the next highest number of shots at 80% and so on. A little thought indicates that this is a difficult test set for testing our algorithm. The parameters α and β in $U(t, c, \phi)$ were estimated using the results of the user study in [10]. The parameters in this algorithm are: $\alpha = 2.397$ and $\beta = 4.0$, $\gamma_o = 0.2$, $\gamma_1 = 0.25$.

We conducted a pilot user study with five graduate students. Each film was on the average, familiar to 2.25 students. The testers were expected to evaluate (Table 3) each skim, on a scale of 1 - 7 (strongly disagree – strongly agree), on the following metric: Does the sequence tell a story? They were additionally asked to rate their confidence in answering the four generic questions of who? where? when? what? for the four skims (ref. section 3.4). Each user watched the skims in random order. After the viewers had evaluated all the eight skims, we also asked the users to evaluate the two skims at each compression rate. For each pair, they were asked indicate (Table 4) degree of agreement (scale 1 - 7) with the statement: skim A is better than skim B.

Table 3: Test scores from five users. The columns: the algorithm used, the film, compression rate, and the five questions.

Algo.	Film	Rate	Story?	where?	what?	who?	when?
new / old	bla	90	4.8 / 4.0	6.8 / 6.6	5.8 / 5.6	6.6 / 6.8	5.2 / 5.0
change			+ 0.8	+ 0.2	+ 0.2	- 0.2	+ 0.2
new / old	far	80	5.8 / 5.2	7.0 / 6.8	6.4 / 6.2	7.0 / 6.8	6.4 / 6.4
change			+ 0.6	+ 0.2	+ 0.2	+ 0.2	0.0
new / old	bom	70	6.4 / 5.6	7.0 / 6.8	6.4 / 6.2	7.0 / 7.0	6.2 / 6.2
change			+ 0.8	+ 0.2	+ 0.2	0.0	0.0
new / old	fou	50	6.4 / 5.8	7.0 / 7.0	6.4 / 6.0	7.0 / 6.8	5.8 / 5.4
change			+ 0.6	0.0	+ 0.4	+ 0.2	+ 0.4

The test scores indicates that the new improved algorithm outperforms the old one at almost every compression rate and at every question. This improvement is statistically significant. We computed the students t-test on the result of the direct skim comparison. The null hypothesis that the two skim algorithms were identical was rejected at confidence level better than 99.99%. Also, the excellent user feedback tapers off at the 80% compression rate (the story? column in Table 3), indicating that perhaps the syntax based reduction is too restrictive; at 90% compression rate it may be better to drop more shots thus increasing the average shot duration of the remaining shots.

Table 4

Rate	New > Old?
90 %	5.0
80 %	5.2
70 %	6.2
50 %	5.6
All	5.50

7. Discussion

In this section, we discuss some of the choices made in our skim generation algorithm in greater detail.

7.1 The comprehension time experiment

We now discuss two aspects of the comprehension time experiment that are important: (a) use of a single still image and (b) the number of subjects in the experiment.

There are two simplifying assumptions in this experiment: (a) the general level semantics of the shot (in terms of the four questions) are adequately captured by our key-frame, and (b) the semantics do *not* change during the course of the shot. (i.e. the answers to who / where / when / what do not change during the course of the shot). Clearly, in complex shots, both of these assumptions may be violated.

Only the first author participated in the experiment. This is problematic and clearly, a more comprehensive study is needed. However, in order to compensate for this deficiency, the subject was tested on 1000 images, picked at random from a very diverse corpus, so as to generate reliable statistics. The user study results from our current set of experiments as well as well as those found in [10] are encouraging. They indicate that the viewers find the skims resulting from the shots condensed to their upper bounds to be highly coherent, thus validating our comprehension experiment.

7.2 Video segment selection

We choose to keep the initial segment for the following reason — the director "sets-up" the shot (i.e. provides the context in which the action takes place) in the first few frames. Hence, by preserving the initial segment, we hoped to preserve the context. In a shot of long duration, the semantics of the shot can change considerably, and in such cases, our strategy will not capture the important information within the shot.

We did consider several alternate strategies including sub-sampling and alternate methods for segment selection. Sub-sampling the shot will cause the data will appear too quickly, with additional loss of lip synchronization. Another strategy is to sub-divide the original shot using a heuristic. This will create new shot boundaries and we conjecture that this will prove to be disconcerting to the viewers (particularly at high compression rates).

7.3 Syntax analysis and object detection

This work focuses on syntax analysis as a method to determine which shots should be removed. We did so since the content within film-data is completely unconstrained. However, object detectors and syntax analysis can complement each other. For example, when we are faced with the task of searching for specific information, the object detectors could serve to highlight important shots in scenes, in the video. Then, syntax analysis could be used to extract a visual phrase around the highlighted shot.

8. Conclusions

In this paper, we've presented a novel framework for condensing computable scenes. The solution has three parts: (a) visual complexity, (b) film syntax analysis and (c) determining the duration of the shots via an constrained utility maximization. We define the visual complexity of a shot to be its Kolmogorov complexity. Then, we showed how to effectively estimate this measure via the Lempel-Ziv algorithm. Then we conducted an experiment that allowed us to map visual complexity of a shot to its comprehension time. After noting that the syntax of the shots influences the semantics of a scene, we devised algorithms based on simple rules governing the length of the progressive phrase and the dialog. We showed how we could guard against the errors in practical shot-detectors in our syntax reduction rules. Then, we developed a sequence utility function that was maximized subject to shot duration constraints and the two penalty functions.

We conducted a pilot user study on four scenes, generating skims using two skim generation algorithms, at four different compression rates. The results of the user study shows that while all skims are perceived as coherent, the skims generated at less than 80% compression work well. We also showed that the improvement in skim comprehension, due to the new algorithm over the old one is statistically significant.

The algorithms presented here leave much room for improvement: (a) we are working on incorporating audio into the skims. (b) incorporating other elements of syntax such as scale and time distribution of shots and differential changes in scale. (c) additional experiments to verify the time-complexity curves as well as a large user study (>25 subjects) are also needed.

9. References

[1] M. Burrows, D.J. Wheeler *A Block-sorting Lossless Data Compression Algorithm,* Digital Systems Research Center Research Report #124, 1994.

[2] M.G. Christel et. al *Evolving Video Skims into Useful Multimedia Abstractions,* ACM CHI '98, pp. 171-78, Los Angeles, CA, Apr. 1998.

[3] T.M. Cover, J.A. Thomas *Elements of Information Theory,* 1991, John Wiley and Sons.

[4] J. Feldman *Minimization of Boolean complexity in human concept learning,* Nature, pp. 630-633, vol. 407, Oct. 2000.

[5] M. Li, P. Vitányi An Introduction to Kolmogorov Complexity and its Applications, 2nd ed 1997, Springer Verlag New York.

[6] H. Liwei et. al. *Auto-Summarization of Audio-Video Presentations,* ACM MM '99, Orlando FL, Nov. 1999.

[7] S. Pfeiffer et. al. *Abstracting Digital Movies Automatically,* J. of Visual Communication and Image Representation, pp. 345-53, vol. **7**, No. 4, Dec. 1996.

[8] S. Sharff *The Elements of Cinema: Towards a Theory of Cinesthetic Impact,* 1982, Columbia University Press.

[9] H. Sundaram, Shih-Fu Chang *Determining Computable Scenes in Films and their Structures using Audio-Visual Memory Models,* ACM Multimedia 2000, pp. 95-104, Los Angeles, CA, Nov. 2000.

[10] H. Sundaram, Shih-Fu Chang *Condensing Computable Scenes via Visual Complexity and Film Syntax Analysis,* ICME 2001, pp. 389-92, Tokyo, Japan, Aug. 2001.

[11] S. Uchihashi et. al. *Video Manga: Generating Semantically Meaningful Video Summaries* Proc. ACM Multimedia '99, pp. 383-92, Orlando FL, Nov. 1999.

[12] D. Zhong *Segmentation, Indexing and Summarization of Digital Video Content* PhD Thesis, Dept. Of Electrical Eng. Columbia University, NY, Jan. 2001.

[13] http://www.gzip.org (*gzip* version. 1.2.4)

[14] http://sources.redhat.com/bzip2/index.html