EXHIBIT 28

Time-Compression: Systems Concerns, Usage, and Benefits

Nosa Omoigui, Liwei He, Anoop Gupta, Jonathan Grudin, and Elizabeth Sanocki

Microsoft Research, Redmond, WA 98052

{nosao,lhe,anoop,jgrudin,a-elisan}@microsoft.com

ABSTRACT

With the proliferation of online multimedia content and the popularity of multimedia streaming systems, it is increasingly useful to be able to skim and browse multimedia quickly. A key technique that enables quick browsing of multimedia is *time-compression*. Prior research has described how speech can be time-compressed (shortened in duration) while preserving the pitch of the audio. However, client-server systems providing this functionality have not been available.

In this paper, we first describe the key tradeoffs faced by designers of streaming multimedia systems deploying time-compression. The implementation tradeoffs primarily impact the granularity of time-compression supported (discrete vs. continuous) and the latency (wait-time) experienced by users after adjusting degree of time-compression. We report results of user studies showing impact of these factors on the average-compression-rate achieved. We also present data on the usage patterns and benefits of time compression. Overall, we show significant time-savings for users and that considerable flexibility is available to the designers of client-server streaming systems with time compression.

Keywords

Time-Compression, Video Browsing, Multimedia, Latency, Compression Granularity, Compression Rate.

1 INTRODUCTION

With the Internet now a mass medium, digital multimedia content is becoming pervasive both on corporate Intranets and on the Internet. For instance, Stanford University makes the video content of 15 or more courses available online every quarter [25]. Similarly, corporations are making internal seminar series available on Intranets. With so much content available, it is very desirable to be able to browse it quickly.

Several techniques exist for summarizing and skimming multimedia content [1, 2, 14]. Of these, time-compression--compressing audio and video in time, while preserving the pitch of the audio--is very promising. Time-compression allows multimedia to be viewed or listened to in less time. People read (and thus comprehend) faster than they speak; studies have shown that as long as compressed speech remains intelligible, comprehension is not affected [2, 11].

Although time-compression has been used before in hardware-device contexts [1] and telephone voicemail systems [15], it has not been available in streaming video client-server systems. This paper describes studies that can guide the design of such systems supporting time compression.

In client-server systems, designers have three choices: 1) A system with multiple pre-time-compressed server-side files, leading to discrete-granularity (e.g., 1.0, 1.25, 1.5) time-compression and long latency (wait-time) for end users. This requires essentially no client-side changes, but has large server-side storage overhead. 2) A simple real-time client-side solution, leading to continuous-granularity time compression, but still with long-latency for end-users. 3) A complex real-time client-side solution, leading to continuous granularity time-compression and negligible latency for end-users.

We have studied the impact of these choices on use patterns, and also attempted to quantify the benefits achieved. We tracked the change over time in the average speedup-factor and the number of adjustments, across users and videos. These measures are compared with the users' perception of the value of time-compression, the amount of time they saved using the feature.

At a high level, our results show time-savings of 22% for the tasks we studied. Coarse granularity and long latency do not seem to deter the benefits of time compression, but can affect user satisfaction. This suggests that considerable flexibility is available to the designers of client-server streaming systems with time compression.

The paper is organized as follows. Section 2 provides a brief introduction to time-compression. Section 3 focuses on the system-level options and tradeoffs involved in building a client-server time-compression system. Section 4 describes the prototype system used in our study, and Section 5 describes experimental procedure and task. Results are presented in Section 6, related work in Section 7, and concluding remarks in Section 8.

2 TIME COMPRESSION BASICS

Time-compression reduces the time to listen to or watch multimedia content. In general, there are two kinds of time-compression: linear time-compression and skimming [2]. With linear time-compression, compression is applied consistently across entire media streams, without regard to the multimedia information contained therein. With skimming, the content of the media streams is analyzed, and compression rates may vary from one point in time to another. Typically, skimming involves removing redundancies – such as pauses in audio –

from the original material. This paper focuses only on linear time-compression.

2.1 Time Compression of Audio

The time it takes to listen to a piece of audio content can be reduced by playing the audio at a lower sampling rate than that at which it was recorded–for instance, by dropping alternate samples. However, this results in an increase in pitch, thereby creating less intelligible and enjoyable audio. One would like to time-compress the audio and preserve pitch, to maximize the intelligibility and quality of the listening experience.

Audio content may comprise of speech and/or music. We focus on the former in this paper. The most basic technique for achieving time-compressed speech involves taking short fixed-length speech segments (e.g., 100ms segments), and discarding portions of these segments (e.g., dropping 33ms segment to get 1.5-fold compression), and abutting the retained segments [5, 7, 8, 16]. The main advantage of this technique is that it is computationally simple and very easy to implement. However, discarding segments and abutting the remnants produce discontinuities at the interval boundaries and produce clicks and other forms of signal distortion.

To improve the quality of the output signal, a windowing function or smoothing filter–such as a cross-fade–can be applied at the junctions of the abutted segments [21]. A technique called *Overlap Add (OLA)* yields signals of very good quality. OLA is relatively easy to implement and is computational inexpensive—the algorithm can be run on a Pentium 90 using only a small fraction of the CPU.

Other techniques for achieving time-compression of speech include *sampling with dichotic presentation* [18, 24], *selective sampling* [17], and improvements to OLA such as *SOLA* and *P-SOLA* [10]. Trading off output quality and computational complexity, we employed OLA in this study.

2.2 Time Compression of Video

Compared to audio, time-compressing video is more straightforward. There are two techniques to time-compressing video linearly. The first involves dropping video frames on a regular basis, consistent with the desired compression rate. For instance, to achieve a compression rate of 50% (i.e., a speedup-factor of 2.0), every other frame would be dropped. In the second technique, the rate at which video frames are rendered is changed. Thus to get a 2.0-fold speed-up, the frames are rendered at twice this rate. The main negative of this scheme is that it is computationally more expensive for the client, as the CPU has to decode twice as many frames in the same amount of time.

3 TRADEOFFS IN BUILDING CLIENT-SERVER TIME-COMPRESSION SYSTEMS

Time-compression has not been employed in client-server multimedia streaming environments. There are several ways to build time-compression into streaming systems in client-server environments, each with advantages and disadvantages.

3.1 Multiple Pre-Processed Server-Side Files

In this model, the server stores separate pre-processed media files for each speedup-factor. The author chooses a set of speedup-factors and encodes different files at each factor. As a user switches between speedup-factors, the client switches to the media file corresponding to the new factor. For example, an hour-long documentary could be time-compressed at rates of 1.0, 1.25, 1.5, 1.75, and 2.0. Users would then have the option of choosing among the resulting files.

This technique has several advantages: (1) Minimal changes to the client and server. (2) No extra bandwidth is required because the time-compressed media files are also encoded at the appropriate bit-rate. (3) It does not affect server scalability, since no complex processing is done on the server. (4) Because time-compression is performed offline, computationally expensive and high quality time-compression algorithms can be used.

The disadvantages are: (1) Latency is incurred when switching between files (when user changes time-compression) and if video is not at a key-frame boundary. (2) Additional storage is required at the server for the different speed media files. (3) The time-compression feature cannot be provided with existing media files. New files have to be encoded. (4) It allows for only discrete speedup-factors, forcing all users to rely on the author's judgment as to speedup granularity.

3.2 Simple Real-Time Client-Side Solution

In this scheme, the client time-compresses the incoming data in real time. Changes to the server are minimal: it must be able to accept a speedup-factor request from the client. To achieve time-compression for a specified speedup-factor, the client sends a message to the server, informing it to stream data to the client at N times the bit-rate at which the data was encoded, where N is the speedup factor. The client then time-compresses the data on the fly.

This technique has several advantages: (1) Time-compression can be achieved with existing media files. (2) No additional storage is needed on the server. (3) It allows for both discrete and continuous speedup-factors. (4) It does not affect scalability, since no complex processing is done on the server. (5) Most importantly, it is simple to implement, as complex buffering and flow-control to eliminate latency are not needed.

The disadvantages are: (1) It requires extra network bandwidth, since the server must send data at a faster rate than that at which it was encoded. This might be feasible on corporate/LAN environments, but not over existing dial-up networks. (2) Time-compression is performed after the audio is decompressed on the client, leading to worse audio quality than in the previous scheme, where time-compression occurs before encoding audio.

3.3 Sophisticated Real-Time Client-Side Solution

The scheme described above can be improved by having the client perform flow-control in order to drive the rate at which the server streams data to it. For example, the client can monitor its buffer and have the server send data at a rate such that its buffer remains in steady state. The client could also have the server tag incoming data samples with the rate at which they were sent. Then, when the user switches speedup-factors, the client tells the server to send at the new rate, and only invoke time-compression when it receives the data for that rate. In addition, the client would track I-frames so that speedup-factor transitions occur at "clean" boundaries.

The net effect of these optimizations is that the client would eliminate – or at least minimize – startup latency that results from buffering. This technique shares all the advantages of the previous method. However, it is much more complicated than the simple client-side solution; potentially complex changes to both the client and the server are required. The characteristics of the three methods are summarized in Table 1 below:

Table 1: Alternatives for Building Time-Compression into Client-Server Multimedia Streaming Systems.

	Multiple Pre-Processed Server-Side	Simple Real-Time Client-Side Solution	Sophisticated Real-Time Client-Side Solution
Allowed Speed-up Factor Granularity	Discrete only	Discrete and Continuous	Discrete and Continuous
Additional Storage Demands?	Yes	No	No
Additional Bandwidth Demands?	No	Yes	Yes
Added Complexity?	Minimal	Yes, on client	Yes, significantly, on client
Limits Scalability?	No	No	No
Works with Existing Media Files?	No	Yes	Yes
Preserves audio signal quality?	Yes, very well	Yes, reasonably well	Yes, reasonably well
Latency while Switching Speedup-Factors?	Yes	Yes	No

4 TIME-COMPRESSION SYSTEM USED IN STUDY

We built the time-compression system by modifying an existing multimedia streaming system, the Microsoft® NetShow™ product. To enable user control of time-compression, the client was changed to correspond to the "simple real-time client-side" solution. The interface and implementation were changed to support full control of the granularity of time-compression and the latency experienced by the user. Figure 1 and Figure 2 show the user interface.

5 EXPERIMENTAL METHOD

5.1 Subjects

To explore user responses to time-compression and the tradeoffs described, fifteen subjects participated in two study sessions in the Microsoft Usability Labs. They were recruited from a pool of participants previously indicating interest in participating in usability testing at Microsoft. Subjects were intermediate or better Windows users who indicated interest in the topic areas of the videos to be presented. They were given software products for their participation.

5.2 Experimental Procedure

5.2.1 Conditions Tested

All subjects completed five conditions. The first was the control condition, where no time-compression was available. The other four were derived from two values for each of two control parameters. The first parameter was *latency*, i.e., the time following a speedup adjustment before the video resumed playing. The values used for this were 0 and 7.5 seconds, the latter chosen to reflect typical latency for NetShow today. The second parameter was *granularity*, representing the step-size for possible speedup adjustments. The two settings used were continuous and discrete. For the continuous case we use granularity of 0.01, and for the discrete case granularity of 0.25 (allowing speedup factors of 1.0, 1.25, 1.5, etc).

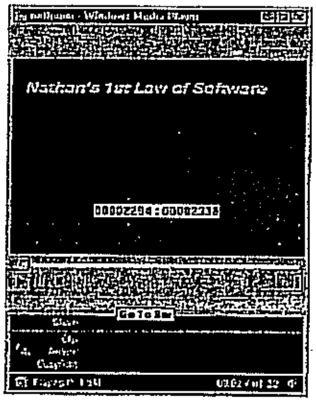

Figure 1. The modified interface for Microsoft® NetShow™ with time-compression UI elements. The status bar shows the current speedup-factor.

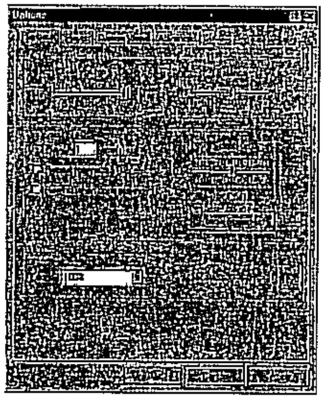

Figure 2. The interface for the "Options" dialog box. Note the slider control for speedup-factor and the "normal speed-up" button that returns users quickly to the regular viewing speed.

The five conditions we study thus are: *CG-LL, CG-NL, DG-LL, DG-NL* (CG/DG for continuous vs. discrete granularity, and LL/NL for long-latency vs. no-latency), and *no-TC* (no time-compression). Based on Section 3, the three conditions of primary interest are CG-LL, CG-NL, and DG-LL.

5.2.2 Subject Tasks

The subjects watched five 25 - 40 minute videos. Two were Discovery Channel™ videos on sharks and grizzly bears, and three were talks from ACM's 50th Anniversary Conference "The Next 50 Years of Computing" held in March 1997. We used talks by Raj Reddy, Bran Ferren and Elliot Soloway. The videos ranged from being easy to watch and visually stimulating to being more intellectually challenging and requiring concentration.

Subjects were asked to assume that they were in a hurry and would have to summarize the videos' contents during a meeting scheduled for later that day. After watching each video, a subject did give a 3-5 minute verbal summary. The videos were viewed in the same sequence by all subjects, but we counterbalanced the four latency-granularity and control conditions; subjects experienced conditions in different orders.

The subjects watched the videos over two days/sessions. The first session began by filling out a background questionnaire. After completing a training session where they familiarized themselves with the operation of the software, they watched the first two videos (the ACM talk by Raj Reddy and the Discovery Channel™ video on sharks). During the second study session, the subjects watched the remaining three videos: an ACM talk by Bran Ferren, a Discovery Channel™ documentary on grizzly bears, and another ACM talk by Elliot Soloway. The second study session ended with the subjects completing a post-study questionnaire and participating in a debriefing session where they discussed their impressions of the time compression feature.

While watching the videos, subjects had full control. They could play, pause, stop, adjust the volume, and move to specific parts of the video via a "seek" bar. The client computer logged these actions: "Open," "Play," "Pause," "Stop," "Seek," "Change Speedup-Factor," and "Close." Also recorded were the "Seek" positions and the "Change Speedup-Factor" rates.

A few caveats: We did not specifically focus on subjective experience of fatigue or confidence in their comprehension, and only obtained general impressions of their preferences. And we recognize that the preliminary interface to the prototype could have some effect on performance.

6 RESULTS

We now report on how the use of time-compression varied with the control conditions, the subjects' usage behavior across time and videos, number of adjustments made, and the savings in task-time.

6.1 Use of Time Compression

The first measure of interest is the average-compression-rate used by the subjects as a function of the conditions. It is calculated based on the amount of time spent at each compression factor:

$$average\_compression\_rate = \frac{\sum_i usertime(i) * speedup\_factor}{\sum_i usertime(i)}$$

Equation 1: Average compression rate.

usertime(i) is length of the i^{th} contigious playing time at a given compression factor. A new interval begins when the compression-factor is changed. All pause times to take notes, etc are excluded in this measure (we will look at them later, when we consider the total-task-time).

Our thinking before the study was as follows:

Continuous vs. Discrete granularity: On one hand we felt that continuous granularity would lead to greater savings in time, because subjects would move to the highest intelligible speedup factor for that specific video segment. E.g., if a video segment was not understandable at 1.5-fold speedup (feasible in the discrete case), they could watch it at 1.4 rather than having to drop to 1.25 (the next option in the discrete case). On the other hand, we could see how that discrete granularity could lead to greater savings in time. If a video-segment could be watched with extra-concentration at 1.5-fold speedup, then in the discrete case users might continue to watch at that higher speed rather than switch down to 1.25. In contrast, for the continuous case, they might drop to 1.4.

No-latency vs. Long latency: Here also, our intuition was conflicting. On one side, we felt that no-latency would lead to higher overall speed-up, as subjects would be more prone to making frequent adjustments to match the current video segment. As in the continuous-vs.-discrete case, however, the fixed speed that the long-latency subjects use could be faster than what the no-latency subjects were using (at the cost of more concentration), so the outcome is unclear.

Table 2 presents the average-compression-rate across all subjects and conditions. The first thing that we observe is that the average-compression-rate, across all subjects and conditions, is quite substantial (avg=1.42). If one considers the total length of all five videos (~2.5 hours), this implies a savings of about 45 minutes.

Table 2. Average Compression Rate Across Subjects and Conditions.

Subject No.	CG-LL	CG-NL	DG-LL	DG-NL	Average	Std Dev
1	1.45	1.31	1.48	1.37	1.40	0.08
2	1.47	1.5	1.34	1.4	1.43	0.07
3	1.68	1.71	1.5	1.5	1.60	0.11
4	1.32	1.37	1.36	1.25	1.33	0.05
5	1.42	1.35	1.33	1.25	1.34	0.07
6	1.57	1.71	1.58	1.51	1.59	0.08
7	1.06	1.18	1.36	1.14	1.19	0.13
8	1.37	1.43	1.42	1.41	1.41	0.03
9	1.43	1.43	1.46	1.48	1.45	0.02
10	1.41	1.42	1.46	1.27	1.39	0.08
11	1.44	1.39	1.48	1.42	1.43	0.04
12	1.52	1.44	1.35	1.4	1.43	0.07
13	1.28	1.24	1.26	0.92	1.18	0.17
14	1.36	1.61	1.49	1.71	1.54	0.15
15	1.61	1.82	1.7	1.46	1.65	0.15
Avg.	1.43	1.46	1.44	1.37		
Std Dev	0.15	0.18	0.11	0.18		

Quite to our surprise, we found that there are no significant differences in the average-compression-rate achieved (repeated measures ANOVA, p = n.s.).[1] We found the subjects to be quite diverse in their usage patterns. For example, considering latency effects, while 6 of the 15 subjects (1, 5, 9, 11, 12, 13) perform faster under CG-LL, the rest operate faster under CG-NL. Similarly, considering granularity affects, while

[1] Statistically, we do find a significant interaction between latency and granularity factors (repeated measures ANOVA, F = 6.286, p = 0.025). The average-compression-rate for DG-NL was lower than that for other conditions, but since that condition is not of interest to us (based on Section 3) we do not comment here on the result.

5 of the 15 subjects perform faster under CG-LL, the rest are faster under DG-LL. It appears that the counter-acting factors considered before the study seem to be balance out in practice.

Looking at the individual subjects, we see considerable variation in the speed-up factors they used (averaged across all conditions). The fastest three averaged 1.65, 1.60, and 1.59, while the slowest three averaged 1.18, 1.18, and 1.32. This is not too surprising given the variation in the subjects—e.g., the 16-year old high-school student (subject 3) averaging 1.60 speed-up to the 60-year old retired person (subject 13) averaging 1.18 speed-up.

So, what are the implications for designers? The key implication is that implementers should feel free to choose the simplest solution, DG-LL, barring the storage overhead on the server side. If this storage overhead is not acceptable, then CG-LL should provide similar benefits to end-users at much less complexity than CG-NL.

6.2 Usage Over Time and Across Videos

Another question for us was "How does users' behavior change as they watch a video?" Previous work [19, 29] suggests that training on time-compressed speech increases people's ability to use higher speed-up factors. We wanted to see if those observations would apply in our case within the same video, and also across videos (i.e., greater speed-up factor used for videos later in the sequence).

Figure 3 shows the speed-up factor across time for the five videos. The videos appear in the same order in which they were watched by the subjects.

Looking first at change in speed-up used within a video, we see some interesting results. For the Reddy and Shark videos (these two videos were watched on the first day of the subjects' visit to the Usability lab) we clearly see that the subjects are watching them faster as they get deeper into the video. There is some slowdown right at the end, an area that corresponds to the concluding remarks.

Surprisingly, for the latter three videos, which were watched on a subsequent day (their second visit), the pattern is quite different. The subjects start watching the video at a higher speed-up factor (between 1.35-1.4, in contrast to 1.23-1.28), but overall there is no consistent pattern over the duration of the session. Our hypothesis is that on the first day, time-compression was a novel feature for the subjects, and they tried to push their limits. As indicated by past literature, they started conservatively and by end got to quite high speed-up factors. In contrast, on the second day, time-compression was already a familiar feature. The subjects started at a higher compression-rate based on their previous day's experience, and only made local adjustments over the duration of the session. This suggests that in the long-term, when time-compression feature is more universally available, we are more likely to observe the latter behavior.

We look next at change in speed-up across videos. From Figure 3, the numbers are 1.43, 1.46, 1.44, 1.43, and 1.34 respectively. Clearly, there is no increase across videos (repeated means ANOVA = n.s.), as may have been predicted based on the literature [18, 29].

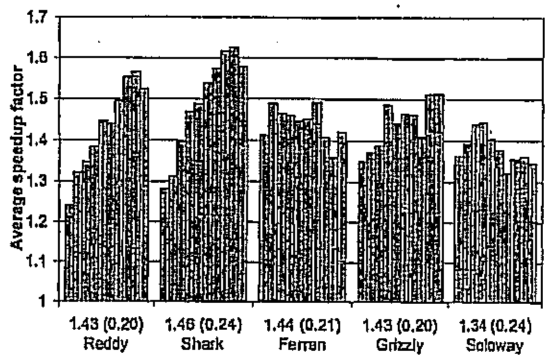

Figure 3: Average speed-up factor as function of time offset within the video. Each bar corresponds to 10% of the length of the video. The average speedup factors and standard deviations for each video are shown below the x-axis.

6.3 Number of Adjustments

One of the things we wanted to learn from the study was "How many adjustments do the subjects make?" Will they just make 2-3 in the beginning and settle in with no more adjustments for the rest of the talk, or will they make tens of adjustments, fine-tuning all along the talk. We did not have any strong predictions before the study (other than there are likely to be more at the beginning of the talk), and we did not know of any previous work to guide our thinking.

Figure 4 shows the distribution of adjustments made by subjects (averaged across conditions) over the length of the session for each video. The mean number of adjustments is quite small (between 2.5 and 4.5). As expected, they tend to occur more in the beginning, though subjects do adjust throughout a session. If almost all adjustments are near the beginning, a design implication might have been to avoid all client modifications, and just provide the end-users with multiple URLs for different speed videos. The data indicate that it is indeed important to allow adjustments throughout the video rather than just a pre-selection mechanism.

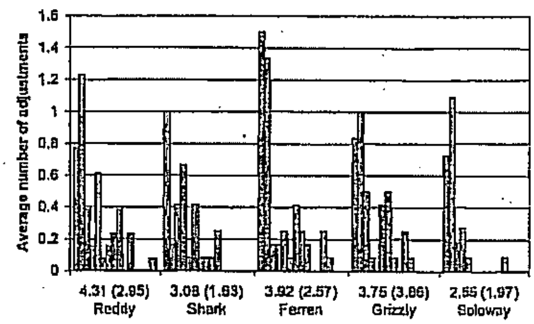

Figure 4: Average number of adjustments to speed-up factor over time. **Each bar corresponds to 10% of the length of the video.**

At a finer level, we were also interested in understanding how these numbers changed with the latency-granularity conditions. Here we expected more adjustments for the lower-latency condition, as it was less disrupting to the viewer. The continuous-versus-discrete granularity cases present counter-acting factors--continuous provides opportunity for fine-grain adjustments, whereas discrete could cause people to switch back-and-forth frequently when pushing their limits.

Table 3 shows the average number of adjustments the subjects made as a function of the conditions. The averages are quite similar (3.1, 3.7, 3.5, and 3.9 respectively) and we found no statistically significant differences (repeated measures ANOVA, p = n.s.). At least for the limited number of subjects we used, our expectation that no-latency condition would lead to higher adjustments is not borne out here. On the whole, we see no particular systems design implications, as the magnitudes are small and similar (3-4 adjustments over a period of 45 minutes).

Table 3: Average Adjustments across Subjects and Conditions

Subject No.	CG-LL	CG-NL	DG-LL	DG-NL	Average	Std Dev
1	3	3	4	9	4.8	2.9
2	3	13	12	9	9.3	4.5
3	5	3	1	3	3.0	1.6
4	3	2	3	2	2.5	0.6
5	5	8	6	3	5.5	2.1
6	6	8	6	6	6.5	1.0
7	2	1	4	3	2.5	1.3
8	5	7	1	6	4.8	2.6
9	2	2	3	5	3.0	1.4
10	2	1	2	2	1.8	0.5
11	3	2	1	6	3.0	2.2
12	3	1	2	1	1.8	1.0
13	2	2	2	2	2.0	0.0
14	1	1	1	1	1.0	0.0
15	2	1	4	1	2.0	1.4
Avg.	3.1	3.7	3.5	3.9		
Std Dev	1.5	3.6	2.9	2.7		

Interestingly, although the data indicate that neither latency nor speedup-factor granularity affected user behavior, several subjects commented in post-study debriefing that the long latency and discrete granularity conditions had affected their use of the time compression feature. The subjects felt that they made fewer adjustments and watched at a lower compression rate when long latency and discrete granularity were used. This indicates that from a product-design (marketing) perspective, these psychological factors may be the primary driving forces to push for the lower-latency continuous-granularity functionality.

6.4 Savings in Task Time

A bottom-line measure of the utility of the time-compression feature is the amount of time it saves in performing the task. For example, a subject using time-compression may find himself/herself reviewing the content more often due to decreased comprehension, thus negating some of the benefits from the use of time compression. In this subsection we quantify these factors.

We decompose task-time into five components: view-time, review-time, pause-time, seek-time, and latency-time. View-time is when a user is watching the video content for the first time. Review-time is the time a user spends reviewing already watched portions of video (this was time spent throughout the session rather than just at the end of the session). Pause-time is when the player is paused, e.g., while taking notes. Seek-time is due to the stall (e.g., for buffer fill) that occurs each time the subject seeks to a different point in the video. Latency-time is due to stall after each change in time-compression adjustment.

Table 4 lists the components of task time for the different granularity-latency conditions. As we expected, we find that the review time does go up when time-compression is used—mean of 157 seconds across all conditions with time-compression versus 126 seconds with no time-compression. Overall, subjects seemed to be spending about 9-11% of their time reviewing the videos with time compression. The data show that pause time was also quite substantial (4-13%), but varied widely across the conditions. The contribution of the buffering latency to overall task-time was minor (for both video-seeks and time-compression adjustments).

Table 4: Components of task time under different conditions.

Time	CG-LL		CG-NL		DG-LL		DG-NL		NO-TC	
	seconds	%	seconds	%	seconds	%	seconds	%	seconds	%
View	1289	81	1325	77	1349	81	1393	85	1883	88
Review	150	9	173	10	175	11	161	10	126	6
Pause	92	6	216	13	68	4	92	6	122	6
Seek/Play	20	1	0	0	37	2	0	0	0	0
Latency	40	3	0	0	35	2	0	0	0	0
Total	1591		1714		1664		1646		2131	
Speedup	1.34		1.24		1.28		1.29		1.00	

The need to review content also brought us some valuable user-interface feedback from the subjects. When using high speed-up factors, the subjects would find that they had just gone past some interesting statement that they did not follow. They would want to back-up in the video (say 15 seconds), but the seek-bar in the interface provided only a *very* blunt control for that (e.g., 30 minutes represented over 3 inches). As a result, users would end-up backing too much most of the time. Specific controls/buttons that would say back-up 5/10/15/30/60 seconds would have been quite valuable.

6.5 User Feedback and Comments

6.5.1 Perceived Value of Time-Compression

In a post-study questionnaire, subjects rated several aspects of the time compression feature. Table 5 summarizes the results of these questions.

Table 5: Average subject ratings for time compression feature.

[illegible]	[illegible]
6.53	I liked having the time compression feature.
6.67	I found the time compression feature useful.
6.40	I would use this software to watch videos again.
6.33	I feel that I saved a significant amount of time by using the time compression feature.

* where 1 = not useful, strongly disagree, etc., and 7 = very useful, strongly agree.

The questionnaire suggested that most subjects liked the feature very much, a finding reported by others. One subject noted "I think it will become a necessity if introduced on a large scale; once people have experienced time compression they will never want to go back. Makes viewing long videos much, much easier." Another said "Many times you spend a lot of time wading through information that is not related to your needs. This speeds up that process." Yet another wrote "Sure, it saves time and people are always short on time."

In our survey, 87% of the subjects reported that they either loved the feature or found it very useful. However, several subjects wrote that they would use the time compression feature at work or at school for information-related content but not at home for entertainment.

Two subjects mentioned that paradoxically, at rates they paid more attention to the videos than at lower rates. One subject noted "My attention span was kept intact. With the slower pace, my attention span actually wavered, and I focused on too much detail. For summarizing, the faster pace is helpful and forces me to concentrate on the major points."

6.5.2 *Perceived Time Savings*

On the questionnaire, we asked the subjects whether they actually saved time by using the time-compression feature. Surprisingly, most subjects said they were not sure of whether they saved time or not. One wrote "I'm not sure if I actually saved a significant amount of time, but it sure felt like I did."

Possibly, once users get used to the time-compression feature, they regard the compressed time as though it were normal time. This is supported anecdotally by the fact that one subject insisted that the time compression feature was broken and he had just viewed the video at the recorded speed.

6.5.3 *Features Requested by Subjects to Complement Time-Compression*

About a third of the users said that they also needed bookmarks or a table of contents in order to quickly browse the videos. In general, they implied that time-compression in and of itself is not enough to give users the ability to browse and skim videos effectively.

This suggests that time-compression should be employed in concert with other features to give users the power to quickly interact with multimedia content.

7 RELATED WORK

Signal processing aspects of time-compression algorithms, such as *OLA*, *SOLA*, and *P-SOLA*, have been studied since the 1950s [1, 5, 7, 8, 14, 16, 17]. These studies are complementary to our work, which focuses on issues that arise in integrating these algorithms into client-server systems. They do not address issues of latency, granularity, scalability of servers, and constraints of constant-bandwidth channels for multiple streams.

Considerable research addresses intelligibility and comprehension of time-compressed speech [1, 3, 6, 11, 12, 28]. In a discrete-granularity study, Harrigan [11] found that students used average speed-up of ~1.3, similar to our ~1.4. Tarquin et al. [28] found that for compression-rates up to 70% (corresponding to a speedup-factor of about 1.4), student performance with time-compressed tutorial tapes was at least as good as that with tapes played at normal-speed. Some researchers have suggested that the limiting factor in comprehension and intelligibility is the word rate, not the compression rate or speedup-factor. Foulke and Sticht [6] discovered that the mean preferred compression rate was 82% (i.e., a speedup-factor of 1.25) corresponding to a word rate of 212 wpm. Although we have not measured the word-rate in our videos, they are quite diverse, yet we saw a comparable compression-rate for the videos, a finding supported by Heiman et al [13].

It has been observed in other studies that exposure to time-compressed speech increases both intelligibility and comprehension. Orr [18] noticed that listeners with no prior exposure to time-compression could tolerate speedup-factors of up to 2.0 but that with 8-10 hours of training, significantly higher speedup-factors are possible. Voor [29] also found that comprehension levels of speech increased with practice. Our results are somewhat different. For the first day subjects used higher speed-ups within a video as time progressed. On the second day, however, no such trends were observed. The subjects seemed to find their preferred speedup range quickly and failed to move much from there.

There have been several studies on applications of time-compression technology as small hardware devices (like voice-mail systems) [23]. More recently, work has also been done on speech-skimming hardware devices [1, 2, 4, 22, 27]. In addition, several classroom educational studies have been performed [11, 26]. Of these, the closest study to ours is Harrigan's [11] wherein he offered students time-compressed lectures at three distinct speedups, 1.0, 1.18, and 1.36 and found that 75% of the time, the students preferred the lectures at 1.36 times speed. Stifelman's [26] study included an examination of the educational use of time-compression but the goal of her work was less on time-compression and more on issues relating to speech annotations. None of the studies have looked at the tradeoffs in use of time-compression from a latency and granularity perspective, as done here.

8 CONCLUDING REMARKS

A key feature in future client-server streaming-media solutions will be time-compression. From an implementation perspective, designers of such systems will have three choices. First, a simple system with multiple pre-processed server-side files, leading to discrete-granularity and long latency access (DG-LL) for end users. Second, a simple real-time client-side solution, leading to continuous granularity, but long latency (CG-LL) for end-users. Third, a complex real-time client-side solution, leading to continuous granularity, but negligible latency (CG-NL) for end-users. In this paper we presented results that will enable designers to make these tradeoffs.

Our data show that under all three conditions, users obtain a substantial compression rate of ~1.4. Quite surprisingly though, there are no significant differences in the time-savings under the three conditions. Thus implementers are free to choose the simplest solution, DG-LL, barring the storage overhead on the server side. If this storage overhead is not acceptable, then CG-LL should provide similar benefits to end-users at much less complexity than CG-NL. While some may feel that the results are negative from a study perspective (in that there are no significant differences across conditions), the news is very good for the implementers.

We also presented results regarding usage patterns and benefits of time compression. Across all five videos, the savings in task-time was 22%. The subjects made only a small number (3-4) of time-compression adjustments during the course of the video, the majority made towards the beginning of video. Overall, the subjects liked the time-compression feature very much (47% voting "loved it" and 40% voting "very useful"). One subject quoted "I think it will become a necessity if introduced on a large scale; once people have experienced time compression they will never want to go back. Makes viewing long videos much, much easier."

ACKNOWLEDGMENTS

Thanks to Microsoft Usability Laboratory for use of their facilities, our user study subjects for their time and efforts, and Mary Czerwinski for her assistence in our study designs.

REFERENCES

1. Arons, B. "Techniques, Perception, and Applications of Time-Compressed Speech." In *Proceedings of 1992 Conference*, American Voice I/O Society, Sep. 1992, pp. 169-177.
2. Arons, B. "SpeechSkimmer: A System for Interactively Skimming Recorded Speech." *ACM Transactions on Computer Human Interaction, 4*, 1, 1997, 3-38.
3. Beasley, D.S. & Maki, J.E. "Time- and Frequency-Altered Speech." In N.J. Lass (Ed.), *Contemporary Issues in Experimental Phonetics*, 419-458. NY: Academic Press, 1976.
4. Degen, L., Mander, R., & Salomon, G. "Working with Audio: Integrating Personal Tape recorders and Desktop Computers." *Proc. CHI '92*, ACM, Apr. 1992, pp. 413-418.
5. Fairbanks, G., Everitt, W.L., & Jaeger, R.P. "Method for Time or Frequency Compression-Expansion of Speech." *Transactions of the Institute of Radio Engineers, Professional Group on Audio AU-2* (1954): 7-12. Reprinted in G. Fairbanks, Experimental Phonetics: Selected Articles, University of Illinois Press, 1966.
6. Foulke, W. & Sticht, T.G. "Review of research on the intelligibility and comprehension of accelerated speech." *Psychological Bulletin*, 72: 50-62, 1969.
7. Garvey, W.D. "The intelligibility of abbreviated speech patterns." *Quarterly Journal of Speech*, 39: 296-306, 1953. Reprinted in J. S. Lim (Ed.) *Speech Enhancement*, Prentice-Hall, Inc., 1983.
8. Garvey, W.D. "The intelligibility of speeded speech." *Journal of Experimental Psychology*, 45:102-108, 1953.
9. Gerber, S.E. "Limits of speech time compression." In S. Duker (Ed.), Time-Compressed Speech, 456-465. Scarecrow, 1974.
10. Griffin, D.W. & Lim, J.S. "Signal estimation from modified short-time fourier transform." *IEEE* Transactions *on Acoustics, Speech, and Signal Processing*, ASSP-32 (2): 236-243, 1984.
11. Harrigan, K. "The SPECIAL System: Self-Paced Education with Compressed Interactive Audio Learning," *Journal of Research on Computing in Education*, 27, 3, Spring 1995.
12. Harrigan, K.A. "Just Noticeable Difference and Effects of Searching of User-Controlled Time-Compressed Digital-Video. Ph.D. Thesis, University of Toronto, 1996.
13. Heiman, G.W., Leo, R.J., Leighbody, G., & Bowler, K. "Word Intelligibility Decrements and the Comprehension of Time-Compressed Speech." *Perception and Psychophysics 40*, 6 (1986): 407-411.
14. Hejna Jr, D.J. "Real-Time Time-Scale Modification of Speech via the Synchronized Overlap-Add Algorithm." MS thesis, *MIT, 1990. Electrical Engineering and Computer Science.*
15. Maxemchuk, N. "An Experimental Speech Storage and Editing Facility." *Bell System Technical Journal 59*, 8 (1980): 1383-1395.
16. Miller, G.A. & Licklider, J.C.R. "The intelligibility of interrupted speech." *Journal of the Acoustic Society of America*, 22(2): 167-173, 1950.
17. Neuburg, E.P. "Simple Pitch-Dependent Algorithm for High Quality Speech Rate Changing." *Journal of the Acoustic Society of America 63*, 2 (1978): 624-625.
18. Orr, D.B. "A perspective on the perception of time compressed speech." In P. M. Kjeldergaard, D. L. Horton, & J. J. Jenkins, (Eds.) *Perception of Language*, 108-119. Merrill, 1971.
19. Orr, D. B. Friedman, H.L., & Williams, J.C. "Trainability of listening comprehension of speeded discourse." *Journal of Educational Psychology*, 56: 148-156, 1965.
20. Portnoff, M.R. "Time-scale modification of speech based on short-time fourier analysis." *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-29 (3): 374-390, 1981.
21. Quereshi, S.U.H. "Speech compression by computer." In S. Duker (Ed.), *Time-Compressed Speech*, 618-623. Scarecrow, 1974.
22. Resnick, P. & Virzi, R.A. "Skip and Scan: Cleaning Up Telephone Interfaces." *Proc. CHI'92 (May 1992)*, ACM.
23. Schmandt, C. & Arons, B. "A Conversational Telephone Messaging System." IEEE Transactions on Consumer Electronics CE-30, 3 (1984): xxi-xxiv.
24. Scott, R.J. "Time Adjustment in Speech Synthesis." *Journal of the Acoustic Society of America 41*, 1 (1967): 60-65.
25. Stanford Online: Masters in Electrical Engineering, 1998. http://scpd.stanford.edu/cee/telecom/onlinedegree.html
26. Stifelman, L. "The Audio Notebook: Paper and Pen Interaction with Structured Speech" *Ph.D. dissertation, MIT Media Laboratory*, 1997.
27. Stifelman, L.J., Arons, B., Schmandt, C. & Hulteen, E.A. "VoiceNotes: A Speech Interface for a Hand-Held Voice Notetaker." *Proc. INTERCHI'93 (Amsterdam, 1993)*, ACM.
28. Tarquin, A., Craver, L., & Schroder, D. "Time-Compression Effects of Video-tapes on Students," *Journal of Professional Issues in Engineering*, Vol. 110, No. 1, January 1984.
29. Voor, J.B. & Miller, J.M. "The effect of practice upon the comprehension of time-compressed speech." *Speech Monographs*, 32: 452-455, 1965.