

Exploring Benefits of Non-Linear Time Compression

Liwei He, Anoop Gupta
Microsoft Research
One Microsoft Way, Redmond, WA 98052
+1 (425) 703-6259
[\[{lhe,anoop}@microsoft.com\]](mailto:{lhe,anoop}@microsoft.com)

ABSTRACT

In comparison to text, audio-video content is much more challenging to browse. *Time-compression* has been suggested as a key technology that can support browsing – time compression speeds up the playback of audio-video content without causing the pitch to change. Simple forms of time-compression are starting to appear in commercial streaming-media products from Microsoft and Real Networks.

In this paper we explore the potential benefits of more recent and advanced types of time compression, called *non-linear time compression*. The most advanced of these algorithms exploit fine-grain structure of human speech (e.g., phonemes) to differentially speedup segments of speech, so that the overall speedup can be higher. In this paper we explore what are the actual gains achieved by end-users from these advanced algorithms. Our results indicate that the gains are actually quite small in common cases and come with significant system complexity and some audio/video synchronization issues.

Keywords: Time compression, Digital library, Multimedia browsing, User evaluation

1 INTRODUCTION

Digital multimedia information on the Internet is growing at an increasing rate: corporations are posting their training materials and talks online [14], universities are presenting videotaped courses online [24], and news organizations are making newscasts available online. While the network bandwidth can be a bottleneck today, newer broadband infrastructures will eventually eliminate bandwidth as a problem. The true bottleneck to address is the limited time people have to view audio-video content.

With the possibility of having so much audio-video content available online, technologies that let people browse the content quickly become very attractive. The effect of even a 10% increase in browsing speedup factor can be substantial when considering the vast number of people who will save time. Different people read documents at different rates, and people read deep

technical documents at a different rate than glossy magazines. However, today's vast body of audio-video content is provided in such a way that assumes that all people watch video and listen to audio at exactly the same speed. A major goal of time compression research is to provide people with the ability to speedup or slow down audio-video content based on their preferences.

In this paper we focus specifically on technologies that allow users to increase the playback rate of speech. While the video portion of audio-video is also important, video time compression is easier to handle than speech and is considered elsewhere [25]. Furthermore, because previous work [17] has shown that people are less likely to speedup up entertainment content (e.g., music videos and soap operas), we focus primarily on informational content (e.g., talks, lectures, and news).

1.1 From Linear to Non-Linear Time Compression

The core technology that supports the increased or decreased playback of speech while preserving pitch is called time compression [7, 10, 13, 15, 21]. Simple forms of time compression have been used before in hardware device contexts [1] and telephone voicemail systems [18]. Within the last year, we have also seen basic support for time compression in major streaming media products from Microsoft and Real Networks [6, 19].

Most of these systems today use linear time compression: the speech content is uniformly time compressed (e.g., every 100ms chunk of speech is shortened to 75ms). Using linear time compression, previous user studies [12, 17, 20] show that participants tend to achieve speedup factors of ~1.4. With this increase, users can save more than 15 minutes when watching a one hour lecture.

In this paper, we explore the additional benefits of two algorithms that employ non-linear time compression techniques. The first algorithm combines pause removal with linear time compression: it first shortens or removes pauses in the speech (which can remove 10-25% from normal speech [9]), then performs linear time compression on the remaining speech.

The second algorithm we consider is much more sophisticated. It is based on the recently proposed Mach1 algorithm [4], the best such algorithm known to us. It tries to mimic the compression strategies that people use when they talk fast in natural settings, and it tries to adapt the compression rate at a fine granularity based on low-level features (e.g., phonemes) of human speech.

1.2 Research Questions

As we will describe later, although the non-linear algorithms offer the potential for better rates time compression, they also require significantly more computing power, cause increased complexity in client-server systems for streaming media, and may result in a jerky video playback. Thus, the core questions we address in this paper are the following:

1. What are the additional benefits of the two non-linear algorithms relative to the simple linear time compression algorithm implemented in products today? While the inventors of Mach1 present some user study data about its benefits, their results deal with very high rates of playback (factors of 2.6 to 4.2), and with conditions where only a subset of the content is understood. However, most people will not listen to speech that is played back at four times its normal rate. We are interested in understanding people's preferences at more comfortable and sustainable rates of speed. Unless differences at sustainable speedups are significant, we don't believe it will be worthwhile to implement these newer algorithms in products.
2. Given the simple and more complex non-linear time compression algorithms, which one is better, and by how much? The magnitude of differences will again guide our implementation strategy in products.

Our results show that for speedup factors most likely to be used by people, the benefits of the more sophisticated non-linear time compression algorithms are quite small. Consequently, given the substantial complexity associated with these algorithms, we do not recommend adopting them in the near future.

The paper is organized as follows: Section 2 reviews various time compression algorithms evaluated in this paper and associated systems implications. Section 3 presents our user study goals, Section 4 the experimental method, and Section 5 the results of the study. We discuss results and present related work in Section 6 and conclude in Section 7.

2 TIME COMPRESSION ALGORITHMS USED AND SYSTEMS IMPLICATIONS

In this section, we briefly discuss the three classes of algorithms we consider in this paper and systems implications for incorporating them in client-server delivery systems. We label the three algorithms we studied Linear, PR-Lin, and Adapt.

2.1 Linear Time Compression (Linear)

In the class of linear time compression algorithms, compression is applied consistently to the entire audio stream without regard to the audio information contained therein. The most basic technique for achieving time compressed speech involves taking short, fixed-length speech segments (e.g., 100ms), discarding portions of these segments (e.g., dropping 33ms segment to get 1.5-fold compression), and abutting the retained segments [7].

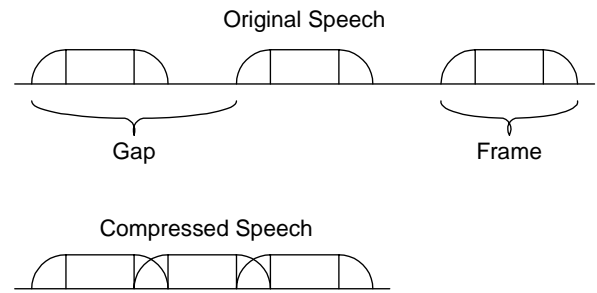


Figure 1: An illustration of Overlap Add algorithm. Windows of samples in the original audio are cross-faded at the overlap regions to produce the output.

However, discarding segments and abutting the remnants produces discontinuities at the interval boundaries and produces audible 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 good results (Figure 1). Further improvements to OLA are made in Synchronized OLA (SOLA) [22] and Pitch-Synchronized OLA [11].

The technique used in this study is SOLA, first described by Roucos and Wilgus [22]. It consists of shifting the beginning of a new speech segment over the end of the preceding segment to find the point of highest waveform similarity. This is usually accomplished by a cross-correlation computation. Once this point is found, the frames are overlapped and averaged together, as in OLA. SOLA provides a locally optimal match between successive frames and mitigates the reverberations sometimes introduced by OLA. The SOLA algorithm is labeled “Linear” in our user studies.

2.2 Pause Removal plus Linear Time Compression (PR-Lin)

Non-linear time compression is an improvement of linear compression because it analyzes the content of the audio stream. As a result, compression rates may vary from one point in time to another. Typically, non-linear time compression involves compressing redundancies (such as pauses or elongated vowels) more aggressively.

For the first non-linear algorithm we study, we use a relatively simple form of non-linear time compression: the PR-Lin algorithm. This algorithm first shortens all pauses longer than 150ms to be 150ms and then applies the SOLA linear time compression algorithm as described in the previous section.

Pause detection algorithms have been published extensively. A variety of measures can be used for detecting pauses even under noisy conditions [3]. Our algorithm uses “Energy” and “Zero crossing rate (ZCR)” features. In order to adjust changes in the background

noise level, a dynamic energy threshold is used. We use a fixed ZCR threshold of 0.4 in this study.

If the energy of a frame is below the dynamic threshold and its ZCR is under the fixed threshold, the frame is categorized as a potential pause frame; otherwise, it is labeled as a speech frame. Contiguous potential pause frames are marked as real pause frames when they exceed 150ms. Pause removal typically shortens the speech by 10-25% before linear time compression is applied.

2.3 Adaptive Time Compression (Adapt)

A variety of non-linear, adaptive algorithms have been proposed that are considerably more complex than the PR-Lin algorithm described in the previous section. For example, Lee and Kim [16] try to preserve phoneme transitions in compressed audio to improve understandability. Audio spectrum is computed first for audio frames of 10ms. If the magnitude of the spectrum difference between two successive frames is above a threshold, they are considered as a phoneme transition and not compressed.

Mach1 [4] makes additional improvements by trying to mimic the compression that takes place when people talk fast in natural settings. These strategies come from the linguistic studies of natural speech [26, 27] and include the following strategies:

- Pauses and silences are compressed the most
- Stressed vowels are compressed the least
- Schwas and other unstressed vowels are compressed by an intermediate amount
- Consonants are compressed based on the stress level of the neighboring vowels
- On average, consonants are compressed more than vowels

Mach1 estimates continuous-valued measures of local emphasis and relative speaking rate. Together, these two sequences estimate the audio tension $g(t)$: the degree to which the local speech segments resist changes in rate. High tension regions are compressed less and low tension regions are compressed more aggressively. Based on the audio tension, the local target compression rates are computed: $r(t) = \max(1, R_g + (1 - R_g)g(t))$ where R_g is the desired global compression rate. The local target compression rate $r(t)$ is then used to drive a standard time scale modification algorithm, such as SOLA.

Because Mach1 is, to our knowledge, the best adaptive time compression technique, the second non-linear time compression algorithm we studied is based on Mach11. Unfortunately, the Mach1 executable was not available to us, thus we could not use it directly. Furthermore, the local compression rate computation in the original Mach1 cannot guarantee a specified global speedup rate (it is an “open loop” algorithm). This presented an issue because in order to compare audio clips compressed using

different algorithms, we required precise overall speedup as specified by the user. Thus, we used a “closed loop” algorithm to compute the local compression rate $r(t)$ from the audio tension $g(t)$:

1. Compute the length of the output audio from the target compression rate R_g . Given a particular window size in SOLA (see Figure 1), the number of windows W can be computed.
2. Compute the amount of audio tension per window G by integrating $g(t)$ across time then dividing by W .
3. Divide the entire audio file into W segments, each of whose audio tension integrates to G .
4. Select one window out of each segment, which can be used by the SOLA algorithm directly. In the current implementation, the selected window centers at the sample that has the highest audio tension.

Although this revised algorithm (Adapt) is based on Mach1, we wanted to ensure it was comparable in quality to the Mach1 algorithm. Thus, a preference study was run to compare our adaptive algorithm to the original Mach1 algorithm. Ten colleagues were asked to compare a time compressed speech file published on Mach1’s web site and the same source file compressed using our implementation (our colleagues were given no indication of how the file they were listening to was created). Of our ten colleagues who participated in this brief study, four preferred our implementation of the algorithm, one preferred Mach1’s implementation, and five had no preference. A Sign test¹ was conducted to assess whether the participants preferred the results from our algorithm, the published Mach1 results, or had no preference. The results of the test were not significant ($p=0.375^2$), indicating our technique is comparable to the Mach1 algorithm.

2.4 Systems Implications of Algorithms

When considering the Linear, PR-Lin, and Adapt algorithms for use in products, two important considerations arise:

- 1) What are the relative benefits? (e.g. speedup rates achievable)
- 2) What are the costs? (e.g. implementation challenges)

This section briefly discusses the costs of the algorithms; the remainder of the paper discusses the relative benefits of the algorithms, as found in our user study.

¹ The sign test is a statistical test that examines whether one item is preferred significantly more than another. It ignores the degree to which participants prefer one item over another (ex: “I like A a lot more than B” is treated the same as “I like A a little more than B”).

² A p-value is the percentage chance that there are no actual differences in the data, and that any differences one sees in the data are due to random variation. The standard practice is to consider anything with less than a 5% chance ($p < .05$) to be statistically significant.

When examining the costs of time compression algorithms, the first issue is computational complexity and CPU requirements. The first two algorithms, Linear and PR-Lin, are easily executed in real-time on any Pentium-class machine using only a small fraction of the CPU. In contrast, the Adapt algorithm has 10+ times higher CPU requirements, although it can be executed in real-time on modern desktop CPUs.

The second issue is complexity of client-server implementations. We assume people will want the time compression feature to be available with streaming media clients in such a way that they can turn a virtual knob to adjust the speed. While there are numerous issues with creating such a system [20], a key issue has to do with buffer management and flow-control between the client and server. The Linear algorithm has the simplest requirements: the server simply needs to speedup up its delivery at the same rate of time compression requested by the client. However, the non-linear algorithms (both PR-Lin and Adapt) have considerably more complex requirements due to the uneven rate of data consumption by the client. For example, if a two second pause is removed, then the associated data is instantaneously consumed by the client, and the server must compensate.

The third issue is the quality of synchronization between the audio and video. (Note that although this paper ignores most of the issues inherent when compressing video, system designers must take into account audio-video synchronization when implementing any time compression algorithms.) With the Linear algorithm, the rendering of video frames is sped up at the same rate as the audio is sped up. While everything happens at higher speed, the video remains smooth and perfect lip synchronization between audio and video can be maintained. However, this task is much more difficult with non-linear algorithms (PR-Lin and Adapt). As an example, consider removing a two-second pause from the audio track. The first option is to also remove the video frames corresponding to those two seconds. In this case the video will appear jerky to users, although the lip synchronization will be maintained. The second option is to make the video transition smoother by keeping some of the video frames from that two-second interval and removing some later ones, but with this technique lip synchronization is lost. Currently, there is no perfect solution.

The bottom line is that non-linear algorithms add significant complexity to the implementer's task. Given this complexity, we must ask whether there are significant user benefits.

3 USER STUDY GOALS

Numerous studies have compared compressed and uncompressed audio (for a comprehensive survey of those studies, see Barry Arons Ph.D. thesis [2]), thus we chose to only compare the three time compression algorithms

presented above amongst themselves: Linear Time Compression (Linear), Pause Removal plus Linear Time Compression (PR-Lin), and Adaptive Time Compression (Adapt). We used the following four metrics to explore the differences between these algorithms:

1. **Highest intelligible speedup factor.** What is the highest speedup factor at which the user still understands the majority of the content? This metric tells us which algorithms perform best when users push the limits of time compression technology for short segments of speech.
2. **Comprehension.** Holding the speedup rate constant, which compression technique results in the most understandable audio? This metric is indicative of the relative quality of speech produced by the compression algorithms. When observed for multiple speedup factors, measures of comprehension can also indicate when users are listening to audio at a speedup factor that isn't sustainable.
3. **Subjective preference.** When given the same audio clip compressed using different techniques at the same speedup factor, which one do users prefer? This metric is directly indicative of the relative quality of speech produced by the algorithms. Since people are very sensitive to subtle distortions that are not computationally understood, subjective preference is an important way to measure issues of quality.
4. **Sustainable speedup factor.** If users have to listen to a long piece of content (like a lecture) under some time pressure, what speedup factor will users choose, and will the speedup factor differ between the different algorithms? We believe this metric is the most indicative of benefits to users in natural settings.

4 EXPERIMENTAL METHOD

Twenty-four people participated our study in exchange for a free software package. Participants came from a variety of backgrounds ranging from professionals in local firms to retirees to homemakers. All participants had some computer experience and some used computers on a daily basis. The participants were invited to our usability lab to do the study.

The entire study was web-based: all the instructions were presented to the subjects via web pages. The study consisted of the following four tasks, corresponding to the four goals outlined in the previous section:

Highest Intelligible Speedup Task: Participants were given three clips compressed by the Linear, PR-Lin, and Adapt algorithms, and were asked to find the fastest speedup at which the audio was still intelligible. We asked participants to choose their own definition of what "intelligible" meant (e.g. understanding 90-95% of words in the audio) but asked them to use a consistent definition throughout the task.

Table 1: Information about tasks and test materials.

	Task	Audio source	WPM	Approx. Length
1	Highest intelligible speed	3 technical talks	99-169	In 10 sec segments
2	Comprehension	6 conversations from Kaplan’s TOEFL program	185-204	28-50 sec
3	Preference	3 clips from an ACM’97 talk by Brenda Laurel	178	30 sec
4	Sustainable speed	3 clips from “Don’t know much about geography”	169	8 min

For each algorithm, short segments of an audio clip were presented to participants in sequence. Participants used five buttons (much faster, faster, same, slower, much slower) to control the speedup at which the next segment was played. The buttons increased or decreased the speedup by a discrete level of either 0.1 or 0.3. When participants found their highest intelligible speedup for the clip, they clicked the “Done” button.

The audio clips used in this task were from 3 talks. The natural speech speed, as measured by words per minute (WPM), had a fairly wide range among the chosen clips (see Table 1): the WPM of the fastest speaker is 71% greater than the slowest speaker. However, the experiments were all counterbalanced among participants, as we will discuss later.

Comprehension Task: We gave each participant six clips of conversations compressed by the three algorithms at 1.5x and at 2.5x. Participants listened to each conversation once (repeats were not allowed) and then answered four multiple-choice questions about the conversation. The conversation clips and questions were taken from audio CDs of Kaplan’s TOEFL (Test of English as Foreign Language) study program [23]. When answering questions, participants were encouraged to guess if they were not sure of the answer. We note that the two chosen speedup factors, 1.5x and 2.5x, represent points on each side of the sustained speedup factor for users.

Subjective Preference Task: Participants were instructed to compare six pairs of clips compressed by the three algorithms at 1.5x and at 2.5x and indicate their preference on a three point scale: prefer clip 1, no preference, prefer clip 2. The audio clips in this task were captured live from an ACM’97 talk given by Brenda Laurel.

Table 2: An example task list for a participant. Condition (the order in which participants experienced the different algorithms) changed for each participant.

	Task	Condition	TC factor
1	Highest intelligible speedup	Linear	(User adjusted)
		PR-Lin	
		Adapt	
2	Comprehension	Linear	1.5
		PR-Lin	
		Adapt	
		Linear	2.5
		PR-Lin	
		Adapt	
3	Preference	Linear vs. Adapt	1.5
		PR-Lin vs. Linear	
		Adapt vs. PR-Lin	
		Linear vs. Adapt	2.5
		PR-Lin vs. Linear	
		Adapt vs. PR-Lin	
4	Sustainable speed	Linear	(User adjusted)
		PR-Lin	
		Adapt	

Sustainable Speedup Task: We gave participants clips compressed by the three algorithms and asked them to imagine that they were in a hurry but still wanted to listen to the clips. Participants adjusted the speedup of the clips until they found a maximum speedup for each clip that was sustainable for the duration of the clips (the clips were about 8 minutes long uncompressed). After listening to the clips, participants were asked to write four or five sentences to summarize what they just heard. However, these textual summaries were used only to motivate people to pay more attention; they were not analyzed for correctness. Audio clips in this task were taken from the audio book “Don’t Know Much About Geography” [5].

All participants completed the tasks in the same order, but for each task, the order in which participants experienced the three algorithms was counterbalanced. Thus, as shown in Table 2, for the first task, a participant may have done task 1 using the Linear algorithm, then the PR-Lin algorithm, and then the Adapt algorithm. However, other participants may have done the first task using the Adapt algorithm first, then the Linear algorithm, and then the PR-Lin algorithm. Mixing up the order in which the algorithms were presented was done to minimize order effects, as it’s possible that a person may become more accustomed to listening to time-compressed speech over time.

5 STUDY RESULTS

For each of the metrics presented in the previous sections, our goal is to determine if there are any differences

Table 3: Average speedups selected by participants for the highest intelligible speedup task. Higher speedups are better.

Condition (n = 24)	Avg Speedup	Std Dev
Linear	1.41	0.25
PR-Lin	1.71	0.32
Adapt	1.56	0.30

Table 4: Results of paired-samples t-tests examining differences between the algorithms. The t-value is the output of the t-test, and df is the number of degrees of freedom for the test (one fewer than the number of participants). The t-value and the number of degrees of freedom combine to provide the p-value.

	t	df	p
Linear vs. PR-Lin	6.66	23	0.001
Linear vs. Adapt	4.12	23	0.001
PR-Lin vs. Adapt	3.83	23	0.001

between the Linear, PR-Lin, and Adapt algorithms. Below we present data from each of the tasks that show the differences between the algorithms.

5.1 Highest Intelligible Speedup

The first task measures the highest speedup at which the clips are still intelligible. Table 3 shows the results from the task. Clearly the results for each algorithm are different, but our next question is to determine if the results are significantly different from a statistical test. To answer this question, we performed an ANOVA (analysis of variance) test, which tells us whether there is an overall difference between the conditions. In this case, the ANOVA resulted in $p < .001$ ($F = 22.0$), indicating that there is an overall, statistically significant difference between the three algorithms. The value of F in ANOVA gives the power of the ANOVA test: the higher the F value, the more trustworthy the test is.

The next step is to test each pair of algorithm combinations to see if each condition is significantly different from the other two. This follow-up test was conducted using paired samples t-tests, each of which yielded a $p < .001$ (Table 4).

Thus, for this task, the PR-Lin algorithm is significantly better than both the Adapt and Linear algorithms, and the Adapt algorithm is significantly better than the Linear algorithm. We speculate that this is due to the fact that with removal of pauses (~15-20% time savings upfront), PR-Lin has to compress the audible speech much less than the other two algorithms to reach the same speedup factor, thus makes words easier to recognize. We reflect on this in the discussion section.

5.2 Comprehension Task

In this task, listener comprehension was tested under different algorithms at the speedup factors of 1.5x and

Table 5: Quiz score results from the comprehension task. Higher scores are better. Max possible score was 5.

Condition	1.5x (StDev)	2.5x (StDev)	Average
Linear	3.1 (1.2)	2.0 (1.2)	2.6
PR-Lin	2.7 (1.5)	2.3 (1.3)	2.5
Adapt	3.1 (1.2)	2.8 (1.3)	3.0
Average	3.0	2.4	2.7

2.5x. We expected Adapt to do best, followed by PR-Lin and Linear. We also expected the comprehension differences to become more pronounced at higher rates of speed. Note that 1.5x and 2.5x represent points on the two sides of the highest intelligible speedup up factor for users.

The quiz scores from the comprehension task are listed in Table 5. Once again, we use an ANOVA to examine whether the numbers in the table are significantly different from each other. In these case, we examine three questions: are the numbers significantly different between each speedup (1.5x and 2.5x), are the numbers significantly different between each algorithm (Linear, PR-Lin, and Adapt), and is there an interaction effect where the numbers are significantly different taking both algorithm and speedup into account?

The answer to the first question is yes: scores go down significantly when speedup is increased ($F = 21$, $p = .001$), which makes sense: it's more difficult to answer questions about the clips when they're played faster.

The answer to the second question is no, although the results are almost statistically significant ($F = 20$; $p = .063$).

The answer to the third question is yes: as speedup increases, scores decrease in significantly different manners ($F = 20$; $p = .006$). Looking at the data, we see that when speedup increases, scores go down a lot for the Linear algorithm, somewhat for PR-Lin, and not as much for Adapt. These results come in line with our expectations. One might extrapolate, given the differences in the average scores of 2.5x and 1.5x, that higher compression rates would make the differences between the algorithms even more pronounced.

5.3 Preference Task

In this task, subjective preference was tested under different algorithms at the speedup factors of 1.5x and 2.5x. The motivation was that minor artifacts caused by time compression which might not affect comprehension may still change a listener's preference. The average preference ranks are listed in Table 6.

As noted earlier, for this task, participants were given three separate pairs of clips (Linear vs. PR-Lin, Linear vs. Adapt, and PR-Lin vs. Adapt) and asked to pick their favorite. To analyze these data, we wanted to transform the results of the comparisons to a 1-3 ranking where 1 is

Table 6: The average rankings of the algorithms from the preference task (lower is better).

Condition	Average Rank	Std Dev
Linear at 1.5x	1.88	0.80
PR-Lin at 1.5x	1.50	0.59
Adapt at 1.5x	1.88	0.74
Linear at 2.5x	2.46	0.59
PR-Lin at 2.5x	1.54	0.59
Adapt at 2.5x	1.17	0.38

Table 7: Wilcoxon tests that show whether the differences between the pairs of algorithms are statistically significantly different. The output of the Wilcoxon test is the z-value, which is then translated to a p-value.

Condition	z	p
Linear vs. PR-Lin at 1.5x	-1.7	0.10
PR-Lin vs. Adapt at 1.5x	-1.8	0.07
Adapt vs. Linear at 1.5x	0.0	1.00
Linear vs. PR-Lin at 2.5x	-3.7	< 0.001
PR-Lin vs. Adapt at 2.5x	-2.1	0.04
Adapt vs. Linear at 2.5x	-4.1	< 0.001

best. Thus, for each participant and speedup factor, we started by giving each algorithm a rank of 1. Then, each time an algorithm lost in a pair-wise comparison, the algorithms rank dropped by 1. Note that this method can result in tied rankings if users are inconsistent in their preferences (for example, preferring Linear over PR-Lin and preferring PR-Lin over Adapt, but preferring Adapt over Linear).

The average ranking scores and their standard deviations are listed in Table 6. As with the previous sections, we turn to statistical analyses to examine whether the numbers in the tables are significantly different. However, because these data were measured using subjective preference scales, instead of using an ANOVA test, we use a Friedman test to look for overall differences in the data. Similarly, we use Wilcoxon tests (not t-tests) to examine differences between pairs of algorithms.

When examining the data for the speedup factor of 1.5, a Friedman test finds that there are no overall significant differences in the rankings ($X^2 = 3.6$; $df = 2$; $p = 0.17$).

However, a Friedman test of the data at the speedup factor of 2.5 finds that the data are significantly different ($X^2 = 30.0$; $df = 2$; $p < 0.001$). Wilcoxon tests examining differences between pairs of algorithms are shown in Table 7.

As Table 7 shows, at the 2.5x speedup factor, all of the algorithms are ranked significantly different from each other, with Adapt ranked the highest on average, PR-Lin ranked second-highest, and Linear ranked lowest. Similar

Table 8: Sustainable speedup by conditions.

Condition	Average Speedup	Std Dev
Linear	1.62	0.28
PR-Lin	1.69	0.38
Adapt	1.76	0.40
Average	1.69	0.36

Table 9: The results of t-tests to look for significant differences between pairs of algorithms in the highest sustainable speedup test. The output of the t-test is the t-value, and df is the number of the degrees of freedom for the test (one fewer than the number of participants). The t-value and number of degrees of freedom are combined to provide the p-value.

Contrast test	t	df	p
Linear vs. PR-Lin	1.6	23	0.12
Linear vs. Adapt	3.7	23	0.001
PR-Lin vs. Adapt	1.5	23	0.15

to the results from the comprehension test, the benefits of the non-uniform algorithms are more pronounced at a higher compression rate.

5.4 Sustainable Speed factor

Data in this section describe the highest speedup at which participants could listen to audio for a sustained period of time. The average speedup factors at which the listeners eventually settled are summarized in Table 8. The Linear method gives the lowest speedup factor at 1.62, followed by PR-Lin's 1.69, which is 4% better than Linear. The Adapt method gives the highest speedup factor at 1.76, which is 8% better than Linear.

The ordering of the average speedup rates is what we expected. Again an ANOVA was conducted to see if there was an overall significant difference in these data. In this case, the ANOVA did find a significant overall difference ($F = 6.4$; $p = 0.006$), and we turn to t-tests to examine the pair-wise differences between the algorithms. Table 9 shows the results of these tests. The only significant difference among the pair-wise comparisons is between the Linear and Adapt algorithms: the highest sustainable speedup at which participants settled was significantly higher when using the Adapt algorithm relative to the Linear algorithm.

6 DISCUSSION AND RELATED WORK

Before discussing our results at length, we briefly summarize the results from the Mach1 paper [4]. The user study reported in the Mach1 paper included listener comprehension and preference tasks comparing Mach1 and linear time compression algorithm. Clips that were two to fifteen sentences long were compressed at factors of 2.6 to 4.2. These are very high speedups, as the

resulting word rates ranged from 390 wpm to an astonishing 673 wpm. Listener comprehension for Mach1 compressed speech was found to improve on average 17% over that for linear time-compressed speech. In the preference test, Mach1 compressed speech was chosen 95% of the time. The difference between Mach1 and linear time compression was found to increase with the speedup factor.

6.1 Comparing the Mach1 Results with this Paper's Results

To best understand Mach1's results and the results presented in this paper, it is useful to divide the observations into two sets: a) results for low to medium speedup factors (e.g., 1.5x), and b) results for high speedup factors (e.g., 2.5x).

For low to medium speedup factors, we have no data from the Mach1 paper. However, our data for speedup factors of 1.5x (looking at comprehension and preference metrics) shows that there are no significant differences in favor of the Adapt algorithm.

At high speedup factors (e.g., 2.5x), the results favor the Adapt algorithm. The Adapt algorithm was ranked significantly higher than both the PR-Lin and the Linear algorithm at the 2.5x speedup factor, and quiz scores went down less for the Adapt algorithm when the speedups increased from 1.5x to 2.5x in the comprehension test. Our intuition is that as audio is compressed at even higher speedup factors (beyond 2.5x), the benefits of the Adapt algorithm will likely become even more pronounced.

6.2 Implications for System Designers

So what do these results imply for system designers? It depends on the task for which time compression is used. For sustainable listening, the first question to ask is the speedup factor most likely to be used by users. Our data from the sustainable speedup task indicates that speedup factors will fall in the 1.6-1.7 range when people are in a hurry. Past results from Harrigan [12], Omoigui et al [20] and Li et al [17] indicate comfortable speedup factors of ~1.4. Results from Foulke and Sticht [8] indicate speedup factors of ~1.25 corresponding to a word rate of 212 WPM.

These data indicate that users will most often utilize low to medium speedup factors. Consequently, for most purposes the Linear algorithm should suffice. As discussed in Section 2.4, it is computationally efficient, simpler for client-server systems, and no jerky video results. More aggressive algorithms like Adapt/Mach1 may help increase the comfortable speedup factor to some extent, but they are limited to applications where client-server system and jerky video issues do not matter: e.g. listening speeches from a local storage.

For the tasks where only word intelligibility is required (e.g. searching for a segment in fast forward mode), our highest intelligible speedup test seems to suggest PR-Lin is the best solution. The PR-Lin algorithm had a

significantly higher average speedup chosen by users. We speculate that this is due to the fact that with removal of pauses (~15-20% time savings upfront), PR-Lin has to compress the audible speech much less than the other two algorithms to reach the same speedup factor. Barry Arons wrote in his Ph.D. thesis [2] that pauses in speech carry significant semantic meanings and can help understand the speech. Occasional pauses also give the listener mental breaks so he can parse the words that he just heard and extract their meanings. Our experiment also supports this finding: that is why the Adapt algorithm, which does not totally delete pauses as in PR-Lin, performs better in the comprehension, preference, and highest sustainable speedup tasks, but people do not care as much about pauses when only word intelligibility is required.

6.3 Comparing Adapt against Naturally Fast Speeches

As we were wrapping up these studies and thinking about the results showing no substantial benefits from sophisticated algorithms like Mach1/Adapt at sustainable speedups, we were left wondering whether it is the case that these state-of-the-art algorithms are still not too good or whether we were hitting some more inherent human limits. With even the best algorithms, participants reached a sustainable speedup of only 1.76x. Is the limit due to the technology or to fundamental limits on human listening capability? Assuming humans are most adept at listening to natural human speech, this question can be tested by comparing naturally sped up speech with artificially compressed speech. We ran two such comparisons in a quick user study.

A colleague of ours with significant background in public speaking was asked to read two articles (each around 700 words) and three short sentences at two speeds: a regular speaking speed and a fast speed. His fast speedup was approximately 1.4 times his regular speed. Both the slow readings (SR) and fast readings (FR) were digitized and time compressed using our Adapt algorithm. Fifteen colleagues participated in a web-based experiment comparing the resulting clips.

In the first comparison, subjects compared the slow readings speedup by Adapt at 1.4x versus the fast readings (which were naturally 1.4 times faster than the slow reading). Of the 45 total comparisons (15 people x 3 short clips) 19 preferred FR, 18 preferred speeded-up SR, and 8 expressed no preference. Clearly, there is no significant preference one way or the other.

Our second comparison was a sustainable speedup test where participants speeded-up both SRs and FRs until comfortable. If naturally sped-up speech is qualitatively different from that generated by Adapt, we would expect the benefits of each to be somewhat additive. Using Adapt, participants should be able to speedup up the FR clips to a rate faster than that of the SR clips. However, this was not the case. When normalized to the speech rate

of the slow readings, the sustainable speedup up for SR was 1.63 and 1.68 for FR. There were no statistical differences, suggesting that the algorithm is a reasonable substitute for natural human speech "compression."

For low to medium speedup factors (speedups that users will typically utilize), the results indicate that the current crop of algorithms is indeed very good, effectively substitutable for natural speech sped-up. It appears that the limits are on the human listening side rather than on how we generate time compressed speech.

7 CONCLUDING REMARKS

We are faced with an information glut, both of textual information and, increasingly, audio-visual information. The most precious commodity today is human attention and time. Time compression, in some sense, can be a magical technology that helps us generate extra time by allowing us to watch audio-visual content at faster rates of speed. Simple forms of time compression technology are already appearing in commercial streaming media products from Microsoft and Real Networks. The question explored in this paper is whether newer, more advanced time compression algorithms have the potential of providing significantly more user benefits (time savings) and to develop an understanding of the associated implementation costs.

Our results show that for speedup factors most likely to be used by people, the more sophisticated non-linear time compression algorithms do offer a significant but small advantage. Given the substantial implementation complexity associated with these algorithms in client-server streaming media systems and issues with audio/video synchronization, we may only see them adopted in some limited applications. Based on a preliminary study, we speculate that the reason the benefits of non-linear time compression are small is not due to poor algorithms, but rather due to limitations on the human side, i.e. how fast people can listen to speech, and today's relatively simple algorithms bring us close to this limit. Thus, delivering significantly larger time compression benefits to users remains an open challenge for researchers.

ACKNOWLEDGMENTS

The authors would like to thank Scott LeeTiernan for his help in statistical analysis. Thanks also go to JJ Cadiz for his initial implementation of the experiment code and his voice for the fast and slow reading experiment and Marc Smith for his valuable comments on the paper.

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. "Interactively Skimming Recorded Speech." In Ph.D. thesis, MIT, 1994.
3. Atal, B.S. & Rabiner, L.R. "A Pattern Recognition Approach to Voiced-Unvoiced-Silence Classification with Applications to Speech Recognition," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, ASSP-24, 3 (June 1976), 201-212.
4. Covell, M., Withgott, M., & Slaney, M. "Mach1: Nonuniform Time-Scale Modification of Speech," *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*. Seattle, WA, May 12-15 1998.
5. Davis, K.C. "Don't Know Much About Geography," *Bantam Doubleday Dell Audio Publishing*, New York, 1992.
6. Enounce, 2xAV Plug-in for RealPlayer
<http://www.enounce.com/products/real/2xav/index.htm>
7. 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.
8. Foulke, W. & Sticht, T.G. "Review of research on the intelligibility and comprehension of accelerated speech." *Psychological Bulletin*, 72: 50-62, 1969.
9. Gan, C.K. & Donaldson, R.W. Adaptive Silence Deletion for Speech Storage and Voice Mail Applications. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 36, 6 (Jun. 1988), pp 924-927.
10. Gerber, S.E. "Limits of speech time compression." In S. Duker (Ed.), *Time-Compressed Speech*, 456-465. Scarecrow, 1974.
11. 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.
12. Harrigan, K. "The SPECIAL System: Self-Paced Education with Compressed Interactive Audio Learning." *Journal of Research on Computing in Education*, 27, 3, Spring 1995.
13. Harrigan, K.A. "Just Noticeable Difference and Effects of Searching of User-Controlled Time-Compressed Digital-Video. Ph.D. Thesis, University of Toronto, 1996.
14. He, L., Grudin J. & Gupta, A., 2000. "Designing Presentations for On-demand Viewing," In *Proc.CSCW'00*. ACM.
15. 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.
16. Lee, S. & Kim, H. "Variable Time-Scale Modification of Speech Using Transient Information," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 2, pp 1319-1322, Munich, 1997.
17. Li, F.C., Gupta, A., Sanocki, E., He, L. & Rui Y. "Browsing digital video," *Proc. CHI'00*, Pages 169 - 176, ACM.
18. Maxemchuk, N. "An Experimental Speech Storage and Editing Facility." *Bell System Technical Journal* 59, 8 (1980): 1383-1395.

19. Microsoft Corporation, Windows Media Encoder 7.0
<http://www.microsoft.com/windows/windowsmedia/en/wm7/Encoder.asp>
20. Omoigui, N., He, L., Gupta, A., Grudin, J. & Sanocki, E. Time-compression: System Concerns, Usage, and Benefits. *Proceedings of ACM Conference on Computer Human Interaction*, 1999.
21. Quereshi, S.U.H. "Speech compression by computer." In S. Duker (Ed.), *Time-Compressed Speech*, 618-623. Scarecrow, 1974.
22. Roucos, S. & Wilgus, A. "High Quality Time-Scale Modification for Speech," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 2, pp 493-496, Tampa, FL, 1985.
23. Rymniak, M., Kurlandski, G, et al. "The Essential Review: TOEFL (Test of English as a Foreign Language)," *Kaplan Educational Centers* and *Simon & Schuster*, New York.
24. Stanford Online: Masters in Electrical Engineering, 1998.
<http://scpd.stanford.edu/cee/telecom/onlinedegree.html>
25. Tavanapong, W., Hua, K.A. & Wang J.Z. "A Framework for Supporting Previewing and VCR Operations in a Low Bandwidth Environment," In *Proc. Multimedia'97*, 303-312, ACM.
26. van Santen, J. "Assignment of Segmental Duration in Text-to-Speech Synthesis," *Computer Speech and Language*, 8(2): 95-128, 1994.
27. Withgott, M. & Chen, F. "Computational Models of American Speech," CSLI Lecture Notes #32, *Center for the Study of Language and Information*, Stanford, CA.