AutoDub: Automatic Redubbing for Voiceover Editing

Shrikant Venkataramani
University of Illinois at
Urbana-Champaign, IL
svnktrm2@illinois.edu

Paris Smaragdis
University of Illinois at
Urbana-Champaign, IL
Adobe Research
paris@illinois.edu

Gautham Mysore
Adobe Research
San Francisco, CA
gmysore@adobe.com

ABSTRACT

Redubbing is an extensively used technique to correct errors in voiceover recordings. It involves re-recording a part of a voiceover, identifying the corresponding section of audio in the original recording that needs to be replaced, and using low level audio tools to replace the audio. Although this sequence of steps can be performed using traditional audio editing tools, the process can be tedious when dealing with long voiceover recordings and prohibitively difficult for users not familiar with such tools. To address this issue, we present AutoDub, a novel system for redubbing voiceover recordings. Using our system, a user simply needs to re-record the part of the voiceover that needs to be replaced. Our system automatically locates the corresponding part in the original recording and performs the low level audio processing to replace it. The system can be easily incorporated in any existing sophisticated audio editor or can be employed as a functionality in an audio-guided user interface. User studies involving participation from novice, knowledgeable and expert users indicate that our tool is preferred to a traditional audio editor based redubbing approach by all categories of users due to its faster and easier redubbing capabilities.

ACM Classification Keywords

H.5.5. Sound and Music Computing: Methodologies and techniques; I.5.2. Interfaces: Voice I/O; I.5.4. Applications: Signal processing

Author Keywords

Redubbing; Voiceover; Overdub; Dynamic time warping.

INTRODUCTION

Narration track recordings form an integral component of audio and audio-visual content in the form of speeches, podcasts, advertisements, films, tutorial and demo videos etc. These recordings exhibit a large variability in duration, ranging from a few seconds to over multiple hours. Audio editing tools have enabled the user to manipulate audio signals by using a multitude of operations to create a high quality narration audio. Content-based editing tools and tools that provide immediate feedback about the recorded speech have been proposed for efficiently recording narration tracks [7, 6].

UIST 2017, October 22–25, 2017, Quebec City, QC, Canada
© 2017 ACM. ISBN 978-1-4503-4981-9/17/10…$15.00
DOI: https://doi.org/10.1145/3126594.3126661

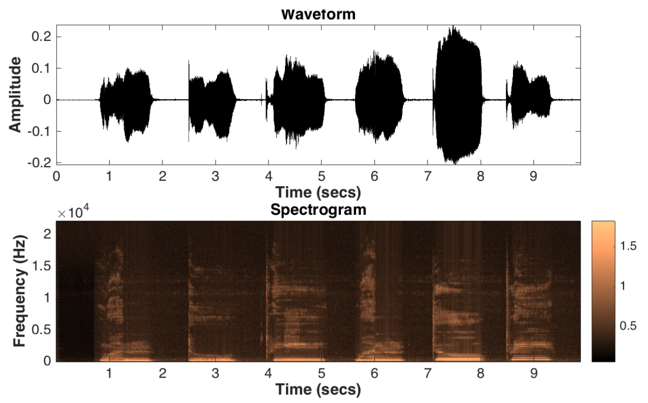


Figure 1: The waveform and spectrogram of an utterance "one-two-three-one-two-three". The repeated utterances are subjected to a pitch increase, loudness increase and a falsetto respectively. These variations significantly alter the waveforms of the utterances while the spectrograms are relatively unchanged.

However, the creation of narration tracks is an error-prone process. Unintentional pronunciation errors are commonly encountered in recording narration tracks. In addition, sudden transient events in uncontrolled environments (sneeze in a lecture) can also obscure one or more words of the narration audio. "Redubbing" enables us to correct such errors without having to re-record the complete narration track. We will refer to the narration track as the "voiceover" that includes an erroneous sentence desired to be corrected. The new correct recording of the erroneous sentence would be referred to as the "overdub". Thus, redubbing replaces the error in the voiceover by the overdub so that it sounds seamless and there are no noticeable artifacts.

The first step of redubbing is to locate the error in the voiceover using the overdub. In short voiceovers, we may attempt to do this manually by listening to the voiceover and the overdub signals. However, longer voiceovers and overdubs require the use of audio visualization and editing tools. A common representation of an audio signal is the waveform shown in figure 1. It can be challenging to see useful information in a waveform beyond the presence of speech and silence, particularly for novices. Also, speech waveforms exhibit significant variation over multiple instances of the same utterance by the same speaker. Interactions with novice users revealed that they of-

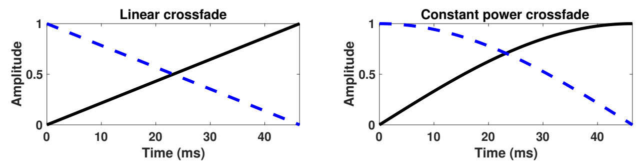


Figure 2: Linear and constant power crossfade windows. The dotted line shows the fade out window and the solid line shows the corresponding fade-in window. The crossfade windows are defined in the range $[0,1]$. over a domain duration of 46.4 milliseconds.

ten attempt to locate the error by trying to match the overdub and the voiceover waveforms, which is futile. A better approach to represent an audio signal is the use of spectrograms. Spectrograms form a 2-dimensional time-frequency representation that displays the energy of each frequency component of the speech signal over time [5]. Spectrograms are more robust compared to waveforms and are preferred by knowledgeable and expert users. However, they do not provide a good mapping between the representation of sounds and their interpretation. Thus, locating the error can still be daunting when long voiceovers are involved.

The voiceover and the overdub may often be recorded in different rooms. Consequently, the background in the two recordings may be significantly different. There could also be differences in the loudness of the two recordings. Resolving these differences forms the second step of redubbing viz., acoustics matching (AM). AM uses the notion of "filtering" an audio signal. In signal processing terms, a filter modifies the energy of each frequency component of its input signal by using a separate multiplicative factor for each component. Constructing an optimal filter for AM is a trial and error based procedure. We incrementally modify the multiplicative factors of the filter and check if the filtered overdub sounds similar in loudness and background to the voiceover. This can be tedious for novice users.

The final step of redubbing is to replace the error in the voiceover by the acoustically matched overdub. To avoid undesirable clicks and pops, we use a crossfade between the voiceover and the overdub. In this step, the volume of the overdub is slowly increased over a predefined crossfade duration while simultaneously decreasing the voiceover volume by a proportionate amount. In other words, we fade in the overdub while simultaneously fading out the voiceover. At the other end of the replacement, to revert back to the voiceover, we fade in the voiceover and fade out the overdub. The rate of change of volume during the crossfade is controlled by symmetric fade in and fade out windows. The shape of the windows are so chosen that the loudness of the mixture over the crossfade duration is equal to the loudness of the rest of the voiceover. Crossfading is often straightforward for experts. But, novices may be unaware of its importance and the procedure to implement a crossfade using an audio editing tool.

Given these considerations, recording narration tracks for demo videos and podcasts can be frustrating. Small spoken mistakes often necessitate either re-recording the voiceover multiple times until it is error-free, or, tedious audio editing to add, remove, of replace words. The current graphical approaches used to visualize sound are neither intuitive nor useful for novice users. The accessibility of audio editors can be improved by developing Audio-guided User Interfaces (AUIs) wherein, the interactions are in the form of spoken text. Such an interface would enable novices to perform complex editing tasks without deep domain experience.

The contributions of this paper are thus, two-fold: (i). In this paper, we demonstrate an alternative to the redubbing workflow that allows us to seamlessly correct errors in the voiceover. We will demonstrate a system in which a user will only need to record small snippets with the correct content, and it will automatically replace the errors in the original recording. This new workflow can hence be easily implemented as an AUI. (ii). We achieve this by applying dynamic time warping (DTW) in a novel way to automatically locate and replace the error. The proposed approach can be easily incorporated into any existing audio editor without any modification to the editors themselves. In section 2, we discuss the approach used to develop our simplified redubbing procedure. In section 3, we evaluate the performance of AutoDub.

AUTOMATIC VOICEOVER REDUBBING: AutoDub

In this section, we present AutoDub, our approach to automate each step of the redubbing process. We assume that the voiceover and the overdub signals have already been recorded by the user.

Automatic AM

We use the algorithm proposed by Francois and Mysore, in [1] for AM. The first stage of the algorithm is to separate the voiceover and overdub into their respective speech and noise signals. This is performed using the denoising procedure described in [9]. The next stage is to acoustically match the speech/noise signal of the overdub to the speech/noise signal of the voiceover. We construct suitable equalization filters for both speech and noise. These filters automatically modify each frequency component of the overdub speech/noise to have the same energy as the corresponding frequency component of the voiceover speech/noise. We incorporate the modifications suggested in [1] to construct realizable filters to avoid undesirable aliasing effects. The matched speech signal is now added to the matched noise signal to form the matched overdub.

Locating the error using dynamic time warping (DTW)

The next step is to automatically locate the erroneous sentence in the voiceover, given the acoustically matched overdub. Locating the error using spoken text and context has been previously employed [10, 11] using language models and confusion networks. In this paper, we develop a novel language-independent method by leveraging the available context that is common in both recordings. Consider two sequences $A = [a_1 \ a_2 ... \ a_m]$ and $B = [b_1 \ b_2 ... \ b_n]$. Here, a_i's and b_j's are vectors of N real numbers each. Also, the sequence B is a scaled (stretched or compressed) version of A. Applying DTW on the sequences returns a warping path that maps each element of B to its closest equivalent element in A [2]. Now, consider an extension to this scenario wherein, the sequence A is a long sequence and B is a shorter subsequence ($m > n$). Also, the subsequence B appears in A and the position is unknown. As before, DTW on the sequences A and B returns a warping path that maps each element of B to its closest equivalent in A. Thus, the warping path includes information about the position of the subsequence B in A. This extension to DTW is known as subsequence DTW [2]. Now consider a further extension wherein, the subsequence B appears with a small error in its middle, somewhere in A. The error could be an addition, deletion or substitution of a few elements in B. It is reasonable to expect that DTW would still locate the position of B in A, provided that there is sufficient context on either side of the error which matches exactly with the elements of B. We use this novel setting for DTW to locate the position of the erroneous sentence in the voiceover when there exists sufficient context on either side of the error that matches exactly with the overdub utterance.

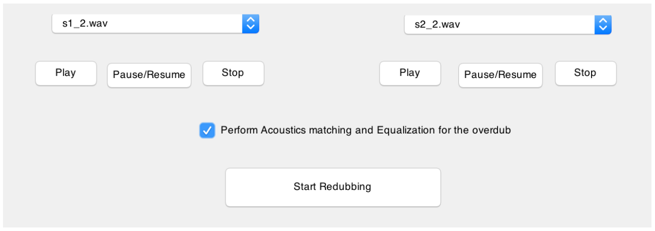
Preprocessing: Multiple instances of the same utterance may exhibit intentional and unintentional pitch variations in the form of jitter and shimmer, even when recorded by the same speaker. A commonly used approach to achieve pitch invariance is the use of Mel spectrograms [4]. The Mel filter bank is a collection of 40 perceptually motivated filters. The filters are concentrated near the lower frequencies and become increasingly spaced out at higher frequencies. The higher frequency filters allow a wider range of frequency components to pass through. We filter the audio signal by each filter of the Mel filter bank and compute the energies of each filter output at several instants in time. Stacking these energies one over the other for the 40 Mel filters gives the Mel spectrogram matrix representation of an audio signal. We compute the Mel spectrograms for the voiceover and the overdub.

Locating the error: We now describe the DTW based approach to locate the overdub position in the voiceover. Here, the sequences A and B are given by the Mel spectrogram representations of the voiceover and the overdub respectively. We also know that the subsequence B appears alongwith an error in A. We first construct a distance matrix d of size $m \times n$ such that $(i, j)^{\text{th}}$ element of d is the Euclidean distance between a_i and b_j. We use this distance matrix to compute the DTW cost matrix D of size $m \times n$. The $(i, j)^{\text{th}}$ element of D is given as,

$$D(i,j) = \begin{cases} d(i,j), & \text{if } i = 1 \text{ or } j = 1 \\ d(i,j) \quad + & \\ \quad min\{D(i-1,j), D(i,j-1), & \\ \qquad D(i-1,j-1)\} & \text{otherwise.} \end{cases} \tag{1}$$

Here, the $min(.)$ operator returns the minimum input value. Next, we begin at (m,n) and find the warping path to $(1,1)$ by iteratively using the following backtracking rule:

$$(i,j) = \begin{cases} (1, j-1), & \text{if } i = 1 \\ (i-1, j), & \text{if } j = 1 \\ argmin\{D(i-1,j-1), & \\ \quad D(i-1,j), D(i,j-1)\}, & \text{otherwise.} \end{cases} \tag{2}$$



✓ Perform Acoustics matching and Equalization for the overdub

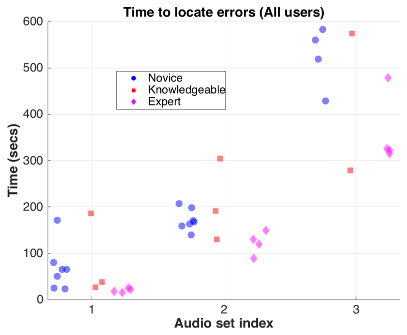
Figure 3: AutoDub GUI used for the user-study. The point of the simple GUI was to show how an automated process can replace a tedious workflow âĂŞ requiring the user to only provide two recordings. Thus, AutoDub can be incorporated into any existing editor with minimal modifications to the interfaces.

Here, (i, j) give the coordinates of the warping path through D. We denote by P_i the sequence of values of i obtained using the backtracking rule, starting at 1 and ending at m. We compute the gradient ΔP_i by subtracting the previous element from every element of P_i. The element of P_i corresponding to the first non-zero value of ΔP_i gives the start position of the match. Similarly, the element of P_i corresponding to the last non-zero value of ΔP_i gives the end position of the match. A point (i_1, j_1) on the warping path such that i_1 lies between the start and end positions implies that the element b_{j_1} in B maps to the element a_{i_1} in A. To deal with multiple matches of the overdub in the voiceover, we iteratively apply DTW based error location multiple times, after setting the vectors in the previous match to zeros.

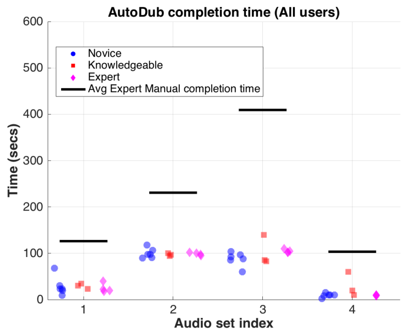
Clearly, the context on either side of the error plays a crucial role in identifying these positions using DTW. Informal experiments show that an error of n seconds requires a minimum context of approximately n seconds on either side. This results in a few considerations to be followed while recording the overdub. The error should occur at approximately the centre of the overdub. Also, in order to avoid abrupt changes in the rhythm of the voice and abrupt crossfades, it is helpful for the overdub to end with some kind of pause (such as a period in the corresponding text). Consequently, if the error occurs at the end of a sentence, it is helpful to include overdub context up to the next natural pause such as the end of the next sentence.

Optimal crossfade windows

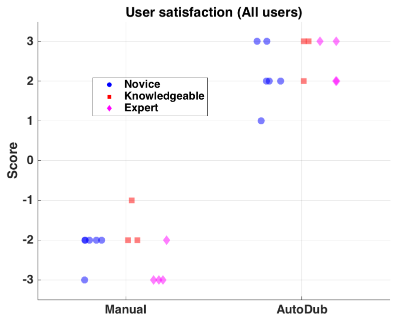
We set the crossfade duration to 46.4 milliseconds. The shape of the crossfade windows is determined by the correlation (a measure of extent of interdependence) between the voiceover and the overdub in the duration of the crossfade. For highly correlated signals, a linear crossfade, as shown in figure 2, satisfies the uniform loudness criterion [8]. Likewise, for uncorrelated signals, the crossfade windows must satisfy the equal power criterion, i.e., the square of the crossfade windows should sum to unity. We observe that an equal power crossfade, shown in figure 2, performs better as the background noises in the voiceover and the overdub are typically uncorrelated.



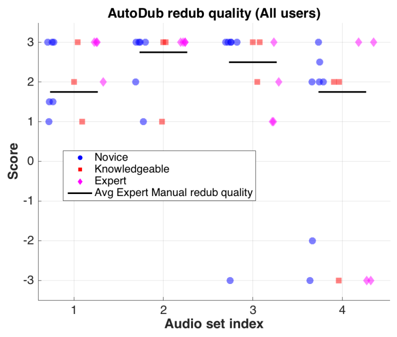
(a) Scatter plot of time to locate the error given the voiceover and overdub.



(b) Scatter plot of AutoDub completion time.



(c) Scatter plot of user-satisfaction scores for manual redub procedure vs AutoDub



(d) Sctter plot of redub quality user-scores obtained using AutoDub

Figure 4: User study results displayed as a scatter diagram. We introduce a small jitter on the x-axis while plotting to improve the visibility of overlapping points.

EVALUATION AND USER STUDY

To the best of our knowledge, there have not been any previous attempts towards building specialized redubbing tools. We evaluate AutoDub in two steps. First, to verify that the proposed method works, we evaluate the accuracy of AutoDub in locating the error. Next, we design a user study to compare AutoDub with the manual approach using an existing audio-editing tool.

Evaluating Error Location

To demonstrate how well AutoDub behaves in locating the errors, we constructed 30 voiceover/overdub pairs and in each pair, determined how well our system could locate the part of the original voiceover that corresponds to the overdub. We constructed these pairs from the DAPS dataset [3], which consists of real recordings in real environments (conference rooms, bedrooms, balcony etc). The dataset includes 5 utterances by 10 male and 10 female speakers each. Each of these recordings are available in 12 different settings characterized by a recording device and a recording environment. Since narration tracks are typically recorded in low-reverberation, low noise environments, we restrict our test set to ipad recordings in 5 environments (2 conference rooms, 1 bedroom and 2 offices). This gives us 500 recordings to choose from with 25 recordings per speaker (5 utterances $\times$ 5 environments).

30 voiceover recordings were randomly selected from these 500 recordings. To simulate errors in the voiceover, we used word additions, deletions or substitutions in the original recording. In the case of word additions and substitutions, we inserted words from the same speaker. These were introduced with proper crossfades so that the transitions and loudness sounded natural, which makes for a plausible erroneous sentence. The length of the error was restricted to a maximum of 4 words and the errors were placed at a randomly selected location in the original recording. The original recording was available in 4 other acoustic environments. One of these 4 recordings was randomly selected to construct the overdub. Thus, the voiceover and overdub signals were obtained from different acoustic environments. To avoid unnatural crossfades in the middle of a sentence, the overdubs were always rounded-off to a period/comma.

It was observed that the DTW algorithm was able to locate the position of the replacement correctly for all the 30 examples. This demonstrates that the DTW algorithm is not a source of error when a suitable overdub recording is available.

Finally, when understanding the nature of demo videos and podcasts, we did not come across significant instances of repetitions of sentences. Even when repetitions of short phrases occurred, they were easily delineated by the fact that their extended context (adjoining sentences/words) was different. Thus, in the above experiments, we restricted the evaluation to single-match examples only.

User Study

We now describe the user-study aimed at comparing AutoDub to the manual approach. We classified the participants of the user study into three categories viz., novice, knowledgeable and expert. Participants familiar with either signal processing or audio editing were classified as knowledgeable. Knowledgeable participants with prior experience of redubbing were classified as experts. Participants unfamiliar with signal processing and audio editing were classified as novice. We recruited 7 novice (6 male and 1 female), 3 knowledgeable (1 male and 2 female) and 4 expert (3 male and 1 female) participants.

When comparing AutoDub to the manual approach, this resulted in a new challenge. Novice and knowledgeable users were typically only capable of performing simple tasks like playing audio, selection and deletion in the editor. The more complex tasks of scrubbing audio, crossfades and multi-track settings were often difficult for them. Given that these are necessary for redubbing, these participants could locate the error at best. Experts, however, could complete the task. Thus, in the user study, we compared 'error location' for novice and knowledgeable participants and, 'error location' and 'redub completion' for experts. This was done in terms of completion times and user satisfaction scores. To test the accessibility of AutoDub in an AUI setting, the participants were provided with a simple interface as shown in figure 3. We note here that AutoDub allowed novice users to complete a redubbing procedure, something that was previously infeasible.

Dataset

The tasks involved in the user study were performed on a dataset of four voiceover/overdub pairs (3 pre-recorded pairs and 1 custom-pair recorded by the participant). For the pre-recorded sets, the overdub signals had an approximate duration of 15 seconds. The voiceover signals had an approximate duration of 1, 5 and 10 minutes respectively. The errors incorporated into the voiceovers were three missing words, three word additions and additive noise respectively. The participants were also provided with an additional trial pair of voiceover and overdub recordings to familiarize themselves with the interfaces involved in the user study. The length of the voiceover is proportional to the difficulty of the set. Thus, the audio sets were used sequentially without counter-balancing during the user study to avoid starting with a relatively difficult task for a novice user.

Tasks and discussion

For the first task, all participants were asked to locate the error in the voiceover for each of the three pre-recorded sets. This task was not performed on the fourth customized set as participants preferred small example recordings. The users were allowed to choose between Adobe Audition and Audacity for this task. The task completion times are shown in figure 4a. We observe that expert users are adept at locating the error given their familiarity with the task. However, half of the novice and knowledgeable participants were unsuccessful in locating the error for the third set. According to some participants, this can be attributed to fatigue induced by the length of the voiceover and the difficulty of use of audio editing tools. As informed by some participants, novice and knowledgeable users often needed to listen to the overdub and the voiceover multiple times to locate the error. In addition, increasing the voiceover length increased the task difficulty significantly. This can be seen from the unanimously negative user-satisfaction ratings for this task as shown in figure 4c (Manual). Thus, AutoDub would be a qualitative improvement to existing audio editors.

The second task of the user study was aimed at comparing the completion times (time to locate the error and time to replace the error) for the entire redubbing procedure. As stated previously, the manual redubbing procedure could be completed by experts only. The black line in figure 4b gives the average manual completion time, averaged across all the expert participants for each set. The redubbing procedure using AutoDub was performed by all the users on all four sets. The AutoDub completion times are also shown in figure 4b. By comparing these completion times with the average expert completion times (black lines), we see that AutoDub allows novice users to perform the redubbing process approximately as fast as an expert user. Even for experts, AutoDub provides a significant advantage in terms of completion times over the manual redubbing procedure. This advantage becomes increasingly pronounced as the length of the voiceover increases.

We also compare AutoDub with the manual approach in terms of resulting voiceover quality. This is shown in figure 4d. The participants have rated AutoDub to have a comparable quality to an expert redub procedure in most cases. Thus, there is a significant economy in terms of completion times without compromises in redub quality. We also note that the redub quality ratings for AutoDub fall for the customised set. The reason for this is the difficulty of a few users in recording a suitable voiceover with sufficient context. However, most users (five novices, one knowlegeable and three experts) were able to correctly record a suitable overdub when informed about the context requirements. The ease of using AutoDub can be seen in the overwhelmingly positive user satisfaction ratings as shown in 4c (AutoDub)

Although, AutoDub is significantly faster than manual redubbing, it could potentially be made faster. The computational bottleneck for AutoDub is the computation of the distance matrix for DTW. Optimizing this step and using GPUs for computation can potentially make the use of AutoDub significantly faster.

CONCLUSION

In this paper, we presented a novel approach to automate and simplify the task of voiceover redubbing, a process known to be tedious and time-consuming when done on an audio editor. Using AutoDub, the user is asked to record the correct version of the sentence in error in the form of an overdub and an automatic process locates and replaces the error in the voiceover. The evaluation and user studies indicate that the availability of a streamlined redubbing tool makes the process easier and accessible to novices and experts alike. Thus, the computational core developed can be a potentially powerful tool that can be employed in audio-driven user interfaces or in existing sophisticated audio editors.

Acknowledgement
This material is partially based upon work supported by the National Science Foundation under Grant No. 1451380. The authors would also like to thank Prof. Karrie Karahalios and the reviewers for their valuable suggestions and guidance.

REFERENCES

1. Francois G. Germain, Gautham J. Mysore, and Takako Fujioka. 2016. Equalization matching of speech recordings in real-world environments. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 609–613.
2. Meinard Muller. 2007. *Dynamic Time Warping*. Springer Berlin Heidelberg, 69–84.
3. Gautham J Mysore. 2015. Can we automatically transform speech recorded on common consumer devices in real-world environments into professional production quality speech?–a dataset, insights, and challenges. *IEEE Signal Processing Letters* 22, 8 (2015), 1006–1010.
4. Lawrence R. Rabiner and Biing-Hwang Juang. 1993. *Fundamentals of Speech Recognition*. Prentice-Hall, Inc.
5. Lawrence R. Rabiner and Ronald W. Schafer. 1978. *Digital processing of speech signals*. N.J. Prentice-Hall.
6. Steve Rubin, Floraine Berthouzoz, Gautham J. Mysore, and Maneesh Agrawala. 2015. Capture-Time Feedback for Recording Scripted Narration.. In *UIST*, Celine Latulipe, Bjoern Hartmann, and Tovi Grossman (Eds.). ACM, 191–199.
7. Steve Rubin, Floraine Berthouzoz, Gautham J. Mysore, Wilmot Li, and Maneesh Agrawala. 2013. Content-based tools for editing audio stories.. In *UIST*, Shahram Izadi, Aaron J. Quigley, Ivan Poupyrev, and Takeo Igarashi (Eds.). ACM, 113–122.
8. Francis Rumsey. 2008. *Digital Audio Recording Formats and Editing Principles*. Springer New York, 703–729.
9. Pascal Scalart and Jose V. Filho. 1996. Speech enhancement based on a priori signal to noise estimation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 629–632.
10. K. Vertanen and P. O. Kristensson. 2009. Automatic selection of recognition errors by respeaking the intended text. In *2009 IEEE Workshop on Automatic Speech Recognition Understanding*. 130–135.
11. K. Vertanen and P. O. Kristensson. 2010. Getting it right the second time: Recognition of spoken corrections. In *2010 IEEE Spoken Language Technology Workshop*. 289–294.