Automatic Removal of Typed Keystrokes From Speech Signals

Amarnag Subramanya, *Student Member, IEEE*, Michael L. Seltzer, *Member, IEEE*, and Alex Acero, *Fellow, IEEE*

***Abstract*—Computers are increasingly being used to capture audio in various applications such as video conferencing and meeting recording. In many of these applications, user may be simultaneously typing on the keyboard, e.g., to take notes or search for information. As a result, the captured speech signals are significantly corrupted by sounds generated by the user's keystrokes. In this paper we propose an algorithm to automatically detect and remove keystrokes from speech signals. The proposed method does not require any user training or enrollment and is computationally efficient. The keystroke removal algorithm generates significantly enhanced speech as measured by both user listening tests and speech recognition experiments.**

I. Introduction

PERSONAL computers and laptops are increasingly being used as devices for sound capture in a variety of recording and communication scenarios including the recording of meetings and lectures for archival purposes, and audio/video instant messaging. Sound capture in these scenarios often faces a unique source of additive noise, that of the user typing on the keyboard. For example, if a meeting attendee takes notes while recording a meeting using the laptop's local microphone, the recorded audio will be significantly corrupted by the sound of the user's keystrokes. This can be very unpleasant for the listener and has a detrimental effect on any subsequent processing, such as automatic speech recognition or stationary noise suppression.

The enhancement of keystroke-corrupted speech can be viewed as a special case of the detection and removal of impulsive noise. There have been several algorithms proposed in the literature for this purpose, e.g., [1]–[4]. However, most of these algorithms target improved speech recognition performance, not perceptual quality.

In this letter, we present novel algorithms for the detection and removal of typed keystrokes in recorded speech that result in significant perceptual improvement. The proposed algorithms do not rely on an explicit model of the keystroke noise, which can be highly variable across different users and devices, but instead exploit several well-known properties of speech signals. The algorithms are computationally efficient, and generalize to unseen deployment environments and devices, without any training or enrollment required by the user.

Manuscript received May 7, 2006; revised August 31, 2006. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Alan McCree.

A. Subramanya is with the Department of Electrical Engineering, University of Washington, Seattle, WA 98105 USA (e-mail: asubram@u.washington.edu).

M. L. Seltzer and A. Acero are with Microsoft Research, Redmond, WA 98052 USA (e-mail: mseltzer@microsoft.com; alexac@microsoft.com).

Digital Object Identifier 10.1109/LSP.2006.888091

II. An Analysis of Keystrokes

Because keys on a keyboard are mechanical pushbutton switches, a typed keystroke appears in the signal as two closely spaced noise-like impulses, one generated by the key-down action and one by the key-up action. The duration of a keystroke is typically between between 60–80 ms but may last up to 200 ms. While keystrokes can be broadly classified as spectrally flat, they also contain a high degree of randomness due to the inherent variety of typing styles, key sequences, and the mechanics of the keys themselves. Because of this variability, traditional approaches based on noise models and stationarity assumptions, such as spectral subtraction, perform poorly for this task.

A. Effect of Keystrokes on Speech Signals

In order to systematically evaluate the effect that keystrokes have on speech signals we digitally mixed clean speech utterances with sequences of keystrokes at signal-to-noise ratios (SNRs) typical of the target applications. The resulting keystroke-corrupted utterances were processed by spectral subtraction using full *a priori* knowledge of the noise. From these enhanced magnitudes, two output waveforms were generated, one which used the phase directly from the noise-corrupted speech, and one which used the phase from the clean speech signal. Empirically, we found that these two signals were perceptually indistinguishable. From this, we concluded that a keystroke removal algorithm should concentrate primarily on enhancing the spectral magnitudes of the keystroke-corrupted speech.

III. Detection of Keystrokes in Speech

In this section, we propose a keystroke detection algorithm that exploits the local smoothness in speech signals present across time and frequency.

A. Unsupervised Keystroke Detection

Each speech utterance $s(n)$ is windowed using a Hamming window of length 20 ms with 10-ms overlap, and then converted to the frequency domain using a short-time Fourier transform (STFT). We define the magnitude of each time-frequency component of the utterance as $S(k,t)$ where t represents the frame index and k represents the spectral index. $S(t)$ represents a vector of all spectral components of frame t. We assume that the signal in each subband follows the following linear predictive model:

$$S(k,t) = \sum_{m=1}^{M} \alpha_{km} S(k, t-\tau_m) + V(k,t) \qquad (1)$$

where, $\boldsymbol{\tau} = \{\tau_1, \ldots, \tau_M\}$ defines the frames used to predict the current frame, $\boldsymbol{\alpha}_k = \{\alpha_{k1}, \ldots, \alpha_{kM}\}$ are the weights applied

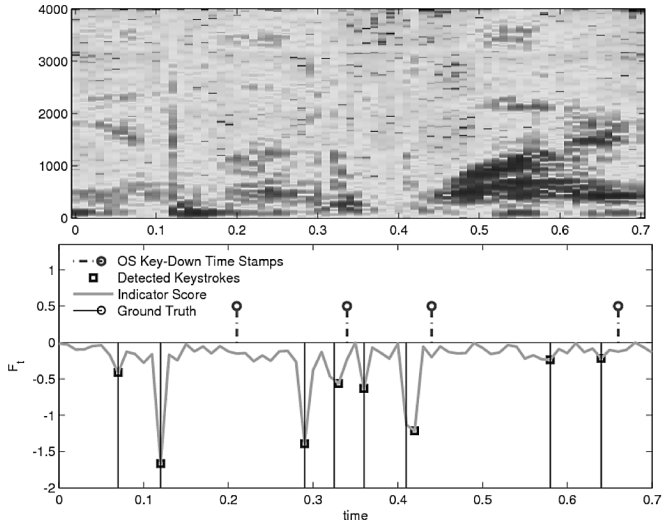

Fig. 1. Top: spectrogram of an utterance corrupted with keystrokes. Bottom: value of F_t for this utterance. The ground-truth locations of the keystrokes are shown by black solid vertical lines.

to these frames, and $V(t,k)$ is zero-mean Gaussian noise, i.e., $V(t,k) \sim \mathcal{N}(0, \sigma_{tk}^2)$. Thus, we can write

$$p(S(k,t)|S(k,t-\tau_1),\ldots,S(k,t-\tau_M)) = \mathcal{N}\left(\sum_{m=1}^{M} \alpha_{km} S(k,t-\tau_m), \sigma_{tk}^2\right). \quad (2)$$

If we assume that the frequency components in a given frame are independent, the joint probability of the frame can be written as $p(S(t)) = \prod_k p(S(k,t))$. Thus, the conditional log-likelihood F_t of the current frame $S(t)$ given the neighboring frames defined by $\boldsymbol{\tau}$ is

$$F_t = \sum_k \log\{p(S(k,t)|S(k,t-\tau_1),\ldots,S(k,t-\tau_M))\} \propto -\frac{1}{2}\sum_k \frac{1}{\sigma_{tk}^2}\left(S(k,t) - \sum_{m=1}^{M} \alpha_{km} S(k,t-\tau_m)\right)^2. \quad (3)$$

Thus, F_t measures the likelihood that frame t can be predicted by its neighbors. A frame is classified as a keystroke if $F_t < T$, where T is an appropriately chosen threshold. Empirically, we have found that keystrokes typically last three frames. As a result, we set $\boldsymbol{\tau} = \{-2, 2\}$. In addition, we use $\alpha_{km} = 1/M$, and estimate the variance in (1) as $\sigma_{tk}^2 = 1/M \sum_m (S(k,t-\tau_m))^2$.

Fig. 1 shows the spectrogram of an utterance and the corresponding score F_t. The solid black vertical lines represent ground-truth locations of keystrokes. As the figure shows, the minima of F_t give an accurate estimate of keystroke locations.

B. Event-Constrained Keystroke Detection (EKD)

While the proposed unconstrained keystroke-detection algorithm can effectively identify keystrokes in speech signals, it has the potential to generate false alarms or missed keystrokes if the likelihood threshold is improperly chosen or varies over time, users, or devices. We can make the detection algorithm more robust by exploiting information available from the computer itself. When a key is pressed, the operating system (OS) generates a *key-down* event. Similarly, when a key is released, a *key-up* event is generated. Unfortunately, there is usually a significant delay between the actual physical event and the time the OS generates the event. This delay is highly unpredictable and varies with the type of scheduling used by the OS, number of active processes, and a number of other factors. In spite of this, we can incorporate the use of OS timestamps in order to improve the keystroke detection algorithm described in Section III-A.

Event-constrained keystroke detection is performed by searching for both the key-down and the key-up events in the audio signal for every key-down event received by the operating system. Empirically, we have found this to be a more robust approach than searching for the key-down and key-up events independently. Thus, for each received key-down time stamp p, the algorithm operates as follows.

1) Find the frame t_p corresponding to system clock time p.

2) Define a search region Θ_p as all frames between previous time stamp t_{p-1} and current time stamp t_p.

3) Find $\hat{t}_D = \arg\min_t\{F_t, \forall\, t \in \Theta_p\}$, classify frames $\Psi_D = \{\hat{t}_D - l, \ldots, \hat{t}_D + l\}$ as keystroke-corrupted frames corresponding to the key-down action.

4) Find $\hat{t}_U = \arg\min_t\{F_t, \forall\, t \in \Theta_p,\ t \notin \Psi_D\}$, classify frames $\Psi_U = \{\hat{t}_U - l, \ldots, \hat{t}_U + l\}$ as keystroke-corrupted frames corresponding to the key-up action.

We have found that because keystrokes typically last three frames, setting $l = 1$ gives good performance.

In Fig. 1, the OS keydown time-stamps are shown as dotted stems. The centers of keystrokes (t_D or t_U) detected using event-constrained keystroke detection algorithm are shown as squares on the F_t curve. This figure illustrates the significant variability in the lag between the physical occurrence of the keystroke and the OS time stamp. The figure also shows that the proposed detection algorithm can accurately detect the location of keystrokes. By using the time stamps from the OS, we have created a threshold-free keystroke detection algorithm. One potential pitfall of the above algorithm is that it would fail if the user-produced multiple keystrokes between a pair of received OS events. However, in our experience, we have noticed that this rarely, if ever, occurs in practice.

IV. Removal of Keystrokes From Speech

In this section, we present a method for removing keystrokes from speech, once the corrupted frames have been identified. The proposed method employs a "missing feature" approach to the enhancement of keystroke-corrupted speech. In missing feature methods, e.g., [5], components of log spectral feature vectors with low local SNR are removed and replaced with new estimates generated using data imputation techniques.

One of the main difficulties of missing feature methods is determining which spectral components to remove and impute. In this work, because keystrokes are spectrally flat and keystroke-corrupted frames have a low local SNR due to the proximity of the microphone to the laptop keyboard, we assume that *all* spectral components of a keystroke-corrupted frame are missing. While this assumption is not strictly true, it allows us to recast the keystroke removal problem to one of reconstructing a sequence of frames from its neighbors.

A. MAP Estimation of Keystroke-Corrupted Frames

To reconstruct the keystroke-corrupted frames, we employ the correlation-based reconstruction technique in [5]. In this algorithm, a sequence of log-spectral vectors of a speech utterance is assumed to be generated by a stationary Gaussian random process. The statistical parameters of this process, its mean and covariance, are estimated from a clean training corpus. By modeling the *sequence* of vectors in this manner, we estimate covariances not just across frequency, but across time as well. Because we assume the process is stationary, the estimated mean vector is independent of time and the covariance between any two components is only a function of the time difference between them. In order for the data to better fit the Gaussian assumption of the model, we operate on log-magnitude spectra rather than on the magnitude directly. Thus, we define $X(t) = \log(S(t))$, where $S(t)$ represents the magnitude spectrum as before.

We now define X_o and X_m to be vectors of clean ("observed") and keystroke-corrupted ("missing") speech, respectively. Under the assumption that $p(X_m, X_o(t))$ is Gaussian with mean μ and covariance Σ, it can be shown that the posterior distribution $p(X_m|X_o(t))$ is also Gaussian, and therefore, the MAP estimate of X_m is the posterior mean, given by

$$\hat{X}_m(t) = E[X_m|X_o(t)] = \mu_m + \Sigma_{mo}\Sigma_{oo}^{-1}(X_o(t) - \mu_o) \quad (4)$$

where μ_o and μ_m are the components of μ corresponding to the clean and keystroke-corrupted speech, respectively. Similarly, Σ_{mo} and Σ_{oo} are the appropriate partitions of the covariance matrix Σ that was learned in training. For a derivation of (4), see [6]. Thus, for each keystroke-corrupted frame in $\Psi = \{\Psi_D, \Psi_U\}$, the keystroke removal algorithm operates as follows.

1) Set $\begin{cases} X_m(t) & = [X(\hat{t}_D - l)^T \ldots X(\hat{t}_D + l)^T]^T \\ X_o(t) & = [X(\hat{t}_D - l - 1)^T\, X(\hat{t}_D + l + 1)^T]^T \end{cases}$

2) Compute the MAP estimate $\hat{X}_m(t)$ according to **(4)**.

3) Repeat steps 1–2 for Ψ_U.

Note that $\hat{X}_m(t)$ is a MAP estimator as the posterior is Gaussian and thus unimodal with its mode occuring at its mean. The experimental setup and results obtained using this algorithm are presented in Section V. However, some shortcomings in the overall performance of the reconstruction algorithm for this application were discovered. Most notably, the large dimensionality of the vectors required computationally expensive matrix operations and the mismatch in noise and reverberation between the training and test environments resulted in estimation errors which produced artifacts in the resulting audio signal.

B. Improved MAP Estimation Using Locality Constraints

1) Reconstruction Using a Block-Diagonal Covariance: In the log spectral domain, each frame consists of N components, where $2N$ is the DFT size. Consequently, Σ_{oo} is $cN \times cN$, where c is the number of frames of observed speech used to estimate the missing frames. Typically, $N \geq 128$ and $c \geq 2$, making the matrix inversion required in (4) computationally expensive. To reduce the complexity of the operations, we assume that covariance matrix has a block-diagonal structure, preserving only local correlations. If we use a block size B, then we need to compute the inverse of N/B matrices of size $cB \times cB$ thus reducing the number of computations. In our experiments, we set $B = 5$.

Using a block-diagonal covariance structure also improves the environmental robustness for farfield speech. There can be long-span correlations across time and frequency in close-talking speech. However, these correlations can be significantly weaker in a farfield audio. This mismatch results in reconstruction errors, producing artifacts in the resulting audio. By using a block-diagonal structure, we utlize short-span correlations only, making the reconstruction more robust in unseen farfield conditions.

2) Locally Adapting the Gaussian Mean: The Gaussian model described in Section IV-A uses a single mean vector to represent all speech. This model, though weak, worked reasonably well in [5] because the training and test data were both from a close-talking microphone, and the algorithm operated on smoothed spectral vectors, i.e., log mel spectra. Because our algorithm reconstructs the full magnitude spectrum, and operates on farfield audio, there is considerably more variation in the observed spectra. As a result, using a single pre-trained mean vector to compute the MAP estimate results in significant reconstruction artifacts.

To improve the model's accuracy, but still keep the computational cost low, we maintain the use of a single mean vector but locally adapt its value. To do so, we utilize a linear predictive framework similar to that proposed for detection in Section III. The mean vector is estimated as a linear combination of the neighboring clean frames surrounding the keystroke-corrupted segment. In our experiments, we estimated $\hat{\mu}_k$ simply as the sample mean of the frames used for reconstruction.

V. Experimental Setup and Results

In order to evaluate the proposed algorithms, we performed two experiments. In the first experiment, we compared the performance of a simple frame relacement algorithm described below (REPLACE), the original MAP reconstruction algorithm described in Section IV-A (MAP), and the proposed locally-constrained MAP reconstruction algorithm described in Section IV-B (LMAP). The REPLACE algorithm simply replaces a missing frame with the closest occuring observed frame (either in the past or future). In the case of a tie, we simply replace with the mean of the frames in question. To evaluate these algorithms, we simulated keystrokes by randomly dropping frames in an utterance. The missing frames were then reconstructed using the REPLACE, MAP, and LMAP algorithms. The Gaussian statistics for the MAP and LMAP algorithms were trained using the WSJ0 SI84 training set [7]. As keystrokes occur in clusters of two or more frames, we first randomly dropped 10% of the frames in the utterance. In order to simulate higher drop percentages, we dropped frames surrounding the missing frames, i.e., to simulate a 30% drop, we randomly dropped 10% of the frames, and then deterministically dropped one frame on either side of randomly dropped frames. Fig. 2 shows the log spectral distortion (LSD) between the original and reconstructed frames for various percentages

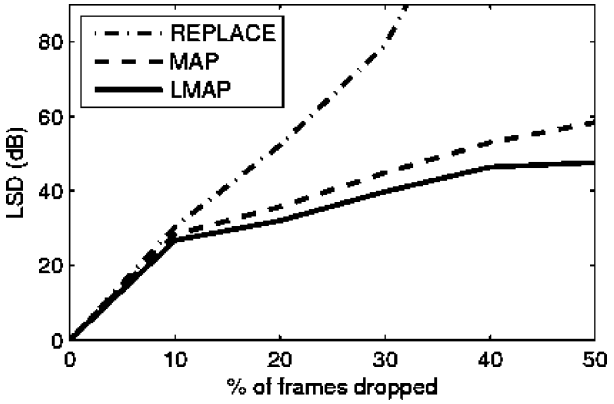

Fig. 2. LSD between original and reconstructed frames for REPLACE, MAP, and LMAP techniques.

TABLE I
DMOS EVALUATION CRITERIA AND RESULTS

Score	A vs. B
3	B much better than A
2	B somewhat better than A
1	B slightly better than A
0	B nearly identical to A
-1	B slightly worse than A
-2	B somewhat worse than A
-3	B much worse than A

	KS vs. MAP	KS vs. LMAP	MAP vs. LMAP
Mean	-1.9231	**1.7713**	2.0128
STD	0.4523	0.3402	0.3203

of dropped frames for all the algorithms. The figure shows that a naive algorithm such as REPLACE results in large reconstruction errors, while for small percentages of dropped frames ($< 20\%$), the performance of MAP and LMAP is similar. However, when 30% or more of the frames are missing (typical of an actual keystroke-corrupted utterance), LMAP achieves significantly lower distortion than the original MAP reconstruction algorithm.

In the second experiment, we collected a corpus of keystroke-corrupted speech data in a conference room environment. Three different laptops were placed on a conference room table. Users were asked to take notes on each laptop while a loudspeaker located across the table played utterances from the WSJ0 test set. Approximately one-third of the test set was recorded on each laptop. This recording session was then repeated in the same environment without any typing on the laptops. This yielded two corpora of 300 utterances, one corrupted by keystrokes (KS), and one that was clean (CL). Note that both corpora contained farfield speech data.

The proposed keystroke removal algorithm was then performed on all utterances of the KS corpus. In order to evaluate the performance, user listening tests were conducted using a differential mean opinion score (DMOS) criterion. Test subjects were asked to make A/B comparisons of a series of utterances processed using different algorithms, using the criteria shown in Table I. The ordering of the utterances presented to each user was randomized. Pairwise comparisons were made across three algorithms: the unprocessed keystroke-corrupted speech (KS), MAP, and LMAP.

The results of the DMOS tests averaged over 36 subjects are shown in Table I. As the results indicate, users showed a strong preference for unprocessed KS utterances over the MAP processing. This indicates that MAP generates artifacts more annoying to users than the keystrokes themselves. On the other hand, users showed a strong preference for the LMAP utterances compared to the KS utterances, with an average DMOS score of 1.77. Thus, the proposed locality constraints significantly improve the reconstruction algorithm and create minimal artifacts or distortion. Finally, the table shows that users preferred LMAP over MAP, as expected given the previous results. The results were found to be significant to the 95% level.

We also performed speech recognitions experiments on the processed LMAP utterances. The HTK speech recognizer was trained using the WSJ0 SI84 training set (close-talking speech). The resulting HMMs were then adapted via supervised MLLR using 100 utterances of farfield speech from the CL corpus. Speech recognition was then performed on the remaining 200 utterances from CL, as well as the same utterances from the KS, and LMAP test sets.

The CL corpus obtained a Word Error Rate (WER) of 66.2%. Because this data was not corrupted by keystrokes, this represents the upper bound on recognition performance. The WER of the KS speech increased to 81.6%, showing that keystrokes degrade recognition performance significantly. The WER of the LMAP corpus, processed by our keystroke removal algorithm, improved to WER of 76.6%, closing the gap in performance between KS and CL by 32%.

VI. CONCLUSIONS

In this paper, we have proposed effective and efficient algorithms to detect and remove keystroke noise from speech signals. The proposed removal algorithm aims to leverage the natural correlations in speech. Further, the algorithm does not require any thresholds that might hinder its generalization ability or any noise statistics for keystroke removal.

REFERENCES

[1] G. A. Tsihrintzis and C. L. Nikias, "Data-adaptive algorithms for signal detection in impulsive noise modeled as a sub-Gaussian, alpha-stable process," in *Proc. IEEE Signal Processing Workshop on Statistical Signal and Array Processing*, Corfu, Greece, Jun. 1996.

[2] P. Ding, "Soft decision strategy and adaptive compensation for robust speech recognition against impulsive noise," in *Proc. Interspeech*, Lisbon, Portugal, Sep. 2005.

[3] M. S. Moore, C. Chandra, and S. K. Mitra, "An efficient method for the removal of impulse noise from speech and audio signals," in *Proc. IEEE Int. Symp. Circuits and Systems*, Monterey, CA, 1998.

[4] S. V. Vaseghi and B. P. Milner, "Noise compensation methods for hidden Markov model speech recognition in adverse environments," *IEEE Trans. Speech Audio Process.*, vol. 5, no. 1, pp. 11–21, Jan. 1997.

[5] M. L. Seltzer, B. Raj, and R. M. Stern, "Reconstruction of missing features for robust speech recognition," *Speech Commun.*, vol. 43, no. 4, pp. 275–296, Dec. 2004.

[6] J. M. Mendel, *Lessons in Estimation Theory for Signal Processing, Communications, and Control*. Upper Saddle River, NJ: Prentice-Hall, 1995.

[7] D. B. Paul and J. M. Baker, "The design of the wall street journal-based CSR corpus," in *Proc. ARPA Speech and Nat. Language Workshop*, Harriman, NY, Feb. 1992, pp. 357–362.