Automatic Removal of Typed Keystrokes from Speech Signals

Amarnag Subramanya *

SSLI Lab
University of Washington
Seattle, WA 98195

Michael L. Seltzer, Alex Acero

Speech Technology Group
Microsoft Research
Redmond, WA 98052

Abstract

Laptop computers are increasingly being used as recording devices to capture meetings, interviews, and lectures using the laptop's local microphone. In these scenarios, the user frequently also uses the same laptop to make notes. Because of the close proximity of the laptop's microphone to its keyboard, the captured speech signal is significantly corrupted by the impulsive sounds the user's keystrokes generate. In this paper we propose an algorithm to automatically detect and remove keystrokes from a recorded speech signal. The detection and removal stages both operate by exploiting the natural correlations present in speech signals, but do so in different ways. The proposed algorithm is computationally efficient, requires no user-specific training or enrollment, and results in significantly enhanced speech. The proposed keystroke removal algorithm was evaluated through user listening tests and speech recognition experiments on speech recordings made in a realistic environment.

Index Terms: speech enhancement, impulsive noise, keystroke removal.

1. Introduction

Laptop computers are increasingly being used as recording devices to capture meetings, interviews, and lectures. In such scenarios, the recording is typically done using the laptop's local microphone, as it is often not feasible to equip the lecturer or meeting participants with close-talking microphones. The user frequently also uses the same laptop to make notes. Because of the close proximity of the laptop's microphone to the keyboard and the significant distance between the laptop and the speaker being recorded, the captured speech signal is significantly corrupted by the sounds the user's keystrokes generate. As a result, listening to the recorded speech can be a very unpleasant experience. The presence of keystrokes in the signal also 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. Many algorithms in the literature model impulsive noise using heavy-tailed non-Gaussian distributions, e.g. [1, 2]. In [3], a HMM-based detection scheme is proposed, an approach also taken by [4]. In [5], impulsive noises in car environment are detected using the Teager Energy Operator. To remove impulsive noise, a modified median filter is used in [6], while a Bernoulli-Gaussian process model is used in [3]. However, these algorithms target improved speech recognition performance, not perceptual quality.

*This work was done at Microsoft Research

In this paper, we present novel algorithms for the detection and removal of typed keystrokes in recorded speech. The proposed algorithms require no training or enrollment on the part of the user, generalize to unseen deployment environments and laptops, and are computationally efficient. The remainder of this paper is as follows: We present the proposed detection and removal algorithms in Sections 3 and 4, respectively. The evaluation of the proposed algorithms is described in Section 5 and some conclusions are presented in Section 6.

2. Closer look at Keystrokes

Because laptop keys are mechanical pushbutton switches, a typed keystroke appears in the speech signal as two closely spaced noise-like impulses, one generated by the key down action and one by the key up action. Figure 1 shows the spectrogram of a four keystrokes produced by the same key. The spectrogram shows the significant variability across frequency and time that even the same key can have. Sources of this variability include the user's typing style, the keystroke sequence, and the actual keyboard itself.

Because of this variability, traditional approaches based on noise models and stationarity assumptions, such as spectral subtraction, perform poorly for this task. To effectively capture keystroke variability, a large, more complex model is required. However, such a model would increase the computational complexity of the enhancement algorithm. Alternatively, a model could be trained or adapted to a specific user. However, we viewed such user training as undesirable, and sought a solution that would perform well "out of the box". Because of these factors, we opted to pursue an algorithm that does not rely on an explicit model of the noise, but rather exploits well-known properties of speech.

2.1. 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 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.

3. 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.

10.21437/Interspeech.2006-72

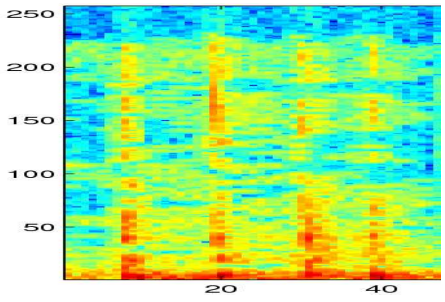


Figure 1: Variability of Key-Strokes

3.1. Unsupervised keystroke detection (UKD)

Each speech utterance $s(n)$ is segmented into 20 ms frames with 10 ms overlap 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

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

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

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

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

$$F_t = \log \prod_k p(S(k,t)|S(k,t-\tau_1), \ldots, S(k,t-\tau_M)) \quad (3)$$

$$= \sum_k \log \{p(S(k,t)|S(k,t-\tau_1), \ldots, S(k,t-\tau_M))\} \quad (4)$$

$$= -\frac{1}{2} \sum_k \frac{1}{\sigma_{tk}^2} \left(S(k,t) - \sum_{m=1}^{M} \alpha_{km} S(k,t-\tau_m) \right)^2 + C_{tk} \quad (5)$$

where C_{tk} is a constant. 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 Eq. (1) simply as $\sigma_{tk}^2 = \frac{1}{M} \sum_m (S(k,t-\tau_m))^2$.

3.2. Event-constrained keystroke detection (EKD)

We can make the detection algorithm more robust by exploiting information available from the laptop 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 into the keystroke detection algorithm described in Section 3.1.

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. 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 $t_{p-1} + l$ (t_{p-1} is the previous time stamp) and the current time stamp t_p.
3. Find $\hat{t}_D = \underset{t}{\text{argmin}} \{F_t, \forall\, t \in \Theta_p\}$, 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 $\hat{t}_U = \underset{t}{\text{argmin}} \{F_t, \forall\, t \in \Theta_p,\ t \notin \Psi_D\}$, 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. We note that by incorporating the OS time stamps into the keystroke detection algorithm, we have removed the necessity for a threshold in the detection process, while also significantly reducing the chance of false alarms in detection.

4. Removal of Keystrokes From Speech

As stated in Section 2, we do not want our keystroke removal algorithm to rely on a prior model of keystrokes. We instead employ an enhancement scheme that uses a prior model of speech. Specifically, we take a "missing feature" approach to keystroke removal. In missing feature methods, e.g. [7], components of log spectral feature vectors with a low local SNR are removed and replaced with new estimates generated using data imputation techniques.

One of the main difficulties in missing feature methods is determining which spectral components to remove. 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 Figure 1 shows that assumption is not strictly true, it considerably simplifies the missing feature problem. In essence, it allows us to recast the keystroke removal problem to one of reconstructing a sequence of frames from its neighbors.

4.1. MAP estimation of keystroke-corrupted frames

We employ the correlation-based reconstruction technique proposed in [7]. In this algorithm, a sequence of log-spectral vectors of a speech utterance is assumed to be a sample of a stationary Gaussian random process. The statistical parameters of this process, its mean and covariance, represent prior knowledge about clean speech, and are estimated from an uncorrupted training corpus. By modeling the *sequence* of vectors, we estimate covariances not just

across frequency, but across time as well. Furthermore, 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 difference in time between them. In order for the data to better fit the Gaussian assumption, we perform reconstruction on log-magnitude spectra rather than on the magnitude spectra 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 Gaussian process assumption, we can now express the MAP estimate of X_m as

$$\hat{X}_m = E[X_m|X_o] = \mu_m + \Sigma_{mo}\Sigma_{oo}^{-1}(X_o - \mu_o) \quad (6)$$

where Σ_{mo} and Σ_{oo} are the appropriate partitions of the covariance matrix learned in training. Thus, for each keystroke-corrupted frame in Ψ_D, the keystroke removal algorithm operates as follows.

1. Set $\begin{cases} X_m &= [X(\hat{t}_D - l)^T \ldots X(\hat{t}_D + l)^T]^T \\ X_o &= [X(\hat{t}_D - l - 1)^T\, X(\hat{t}_D + l + 1)^T]^T \end{cases}$
2. Compute the MAP estimate $\hat{X}_m$ according to Eq. (6).
3. Repeat steps 1-2 for Ψ_U

The experimental setup and results obtained using this algorithm are presented in Section 5. An analysis of these results highlighted some shortcomings in the algorithm for the target application. Most notably, the large dimensionality of the vectors required computationally expensive matrix operations to be performed 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.

4.2. Improved MAP estimation using locality constraints

We propose to improve the performance of the MAP estimation algorithm by imposing locality constraints on both the mean and the covariance in the Gaussian model used. These improvements enable us to improve both the computational efficiency and the robustness of the keystroke removal algorithm.

4.2.1. 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 size of the context window, i.e. the number of frames of observed speech used to estimate the missing frames. Typically, $N \geq 128$ and $c \geq 2$, making the matrix operations required in Eq. (6) 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 have found that setting $B = 5$ works reasonably well.

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 farfield audio. This mismatch results in reconstruction errors, producing artifacts in the resulting audio. By using a block-diagonal structure, we rely only on short-span correlations, making the reconstruction more robust in unseen farfield conditions. To incorporate this change into the MAP estimation algorithm, the single MAP estimation for the keystroke-corrupted frames is simply replaced with multiple estimations, one for each block in the covariance matrix.

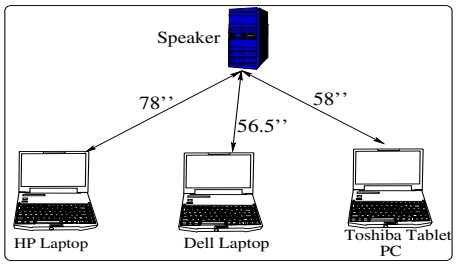


Figure 2: Recording setup used in experiments. The height of the speaker was $52''$. The laptops were placed on a conference room table.

4.2.2. Locally adapting the Gaussian mean

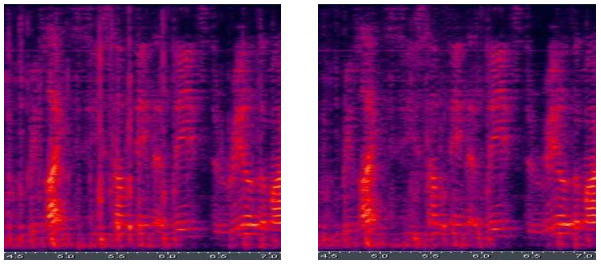
The Gaussian model described in Section 4.1 uses a single mean vector to represent all speech. This model, though weak, worked reasonably well in [7] because the training and test data were both from a close-talking microphone, and the algorithm operated on smoothed spectral vectors, i.e. log mel spectra. Because our algorithm reconstructs the full magnitude spectrum rather than the mel spectrum, and operates on farfield audio, there is considerably more variation in the observed features. As a result, using a single pre-trained mean vector in the MAP estimation process 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 3. The mean vector is estimated as a linear combination of the neighboring clean frames surrounding the keystroke-corrupted segment. If we define μ_k to be the kth spectral component of the mean vector μ, we define the adapted value of this component as $\hat{\mu}_k = \sum_{\tau \in \Gamma} \beta_\tau X(t - \tau, k)$ where Γ defines the indices of the neighboring clean frames, and β_τ is the weight applied to the observation at time $t - \tau$. Because the mean is computed online, it can easily adapt to different environmental conditions. In our experiments we have found that setting Γ to the indices of frames in X_o and $\beta_\tau = 1/|\Gamma|$ resulted in good performance.

5. Experimental Setup and Results

In order to evaluate the proposed keystroke removal algorithm, we collected a corpus of keystroke-corrupted speech data in a conference room environment. Three different laptops (Dell, HP, Toshiba) were placed on a conference room table in the configuration shown in Figure 2. A loudspeaker located across the table from the laptops was used to play utterances from the WSJ0 test set. Users were asked to take notes on each laptop while the audio from the speaker was being recorded using the laptop's local microphone. 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 that was corrupted by keystrokes (KS), and one that was clean (CL). Note that both corpora contained farfield speech data. We trained the means and covariances using the WSJ0 SI84 training set.

The proposed keystroke removal algorithm was then performed on all utterances of the KS corpus. Figure 3 shows a spectrogram of one utterance before and after processing. In order to evalu-



(a) (b)

Figure 3: Spectrograms of a keystroke-corrupted utterance (a) before processing and (b) after the proposed algorithm has been applied.

ate 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 1. The ordering of the utterances presented to each user was randomized. Pairwise comparisons were made across three algorithms: the unprocessed keystroke-corrupted speech (KS), the MAP reconstruction algorithm described in Section 4.1 (MAP), and the locally-constrained MAP reconstruction algorithm proposed in Section 4.2 (LMAP).

The results of the DMOS tests averaged over 36 subjects are shown in Table 2. As the results indicate, users showed a strong preference for unprocessed KS utterances over the enhanced MAP utterances. This indicates that MAP generates artifacts that are more annoying to users than the keystrokes themselves. On the other hand, users showed a strong preference for the LMAP utterances compared to the unprocessed KS utterances, with an average DMOS score of 1.77. This demonstrates that the proposed locality constraints significantly improve the reconstruction algorithm and create minimal artifacts or distortion. Finally, we also confirmed that users preferred KS-LMAP over KS-MAP, which is expected given the previous results.

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. Because this data was not corrupted by keystrokes, this performance represents the upper bound on recognition performance. Recognition was then performed on the same set of 200 utterances from the KS and KS-LMAP corpora.

The CL corpus obtained a Word Error Rate (WER) of 66.2%. This is extremely poor baseline performance for this task, but demonstrates the difficulty of recognizing farfield audio in a real environment with suboptimal microphone placement, especially when the acoustic models are trained from close-talking data. The WER obtained on the KS speech is 81.6%, showing that keystrokes degrade recognition performance significantly. The LMAP corpus, processed by our keystroke removal algorithm, obtained a WER of 76.6%. Thus, the proposed algorithm was able to close the gap in performance between KS and CL by 32%.

Table 1: DMOS Evaluation Criteria

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

Table 2: DMOS Evaluation Results

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

6. Conclusions and Future Work

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 is devoid of any thresholds that might hinder its generalization capabilities and does not require noise statistics for keystroke removal. Although we have presented an algorithm tailored to the keystroke removal application, we believe it can be applied to any impulsive denoising problem with little or no modification.

In the future, we plan on investigating others ways of enforcing local continuity constraints within the proposed framework. In addition, the algorithm presented in this paper assumes that keystrokes corrupt all frequency components of the speech signal. However, as Figure 1 shows, there are spectral components that are in fact unaffected by keystrokes. We plan to improve the keystroke removal algorithm by utilizing these uncorrupted components.

7. References

[1] M. J. Coates and E. E. Kuruoglu, "ime-frequency based detection in impulsive noise environments using alpha-stable noise models," *Proc. ACM Sig. Proc.*, 1999.

[2] G. A. Tsihrintzis and C. L. Nikias, "Data-adaptive algorithms for signal detection in impulsive noise modeled as a subgaussian, alpha-stable process," in *Proc. of IEEE Sig. Proc. Workshop on Stat. Sig. and Array Proc.*, 1996.

[3] S.V. Vaseghi and B.P. Milner, "Noise compensation methods for hidden markov model speech recognition in adverse environments," *IEEE Trans. on Speech and Audio Proc.*, 1997.

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

[5] M. Akbacak and J. L. Hansen, "Environmental sniffing: Noise knowledge estimation for robust speech systems," in *Proc. ICASSP*, 2003.

[6] 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. on Circuits and Systems*, Monterey, CA, 1998.

[7] M. L. Seltzer B. Raj and R. M. Stern, "Reconstruction of missing features for robust speech recognition," *Speech Communication*, 2004.