Recognizing Text Through Sound Alone

Wenzhe Li and Tracy Hammond

Sketch Recognition Lab, Department of Computer Science & Engineering, Texas A&M University, USA

3112 TAMU, College Station, TX, 77839

{liwenzhe, Hammond}@cse.tamu.edu

Abstract

This paper presents an acoustic sound recognizer to recognize what people are writing on a table or wall by utilizing the sound signal information generated from a key, pen, or fingernail moving along a textured surface. Sketching provides a natural modality to interact with text, and sound is an effective modality for distinguishing text. However, limited research has been conducted in this area. Our system uses a dynamic time- warping approach to recognize 26 hand-sketched characters (A-Z) solely through their acoustic signal. Our initial prototype system is user-dependent and relies on fixed stroke ordering. Our algorithm relied mainly on two features: mean amplitude and MFCCs (Mel-frequency cepstral coefficients). Our results showed over 80% recognition accuracy.

Introduction

HCI is a fast growing field in computer science that affords a more natural form of interaction. Many of today's systems support innovative interfaces for people to interact with, such as touch screen tablets and multi-touch surfaces. These devices typically require users to use their stylus or fingers to draw on the screen. Then the system can automatically recognize drawn shapes or gesture commands in an interactive way. While these devices are easy to use and become popular, they are still quite costly and need more time to be integrated into existing systems. Handwriting recognition systems, while gaining in accuracy, rely on users using tablet-PCs to sketch on. As computers get smaller, and smart-phones become more common, our vision is to allow people to sketch on normal pencil and paper and to provide a simple microphone, such as one from their smart-phone to interpret their writings.

In this paper, we propose our novel algorithm to recognize hand-drawn alphabet characters solely through sketch. The intuition from this algorithm is from two separate spaces. First, there is an informal but often played game without a name that friends often play where person A uses a finger or key to sketch certain shapes or word on a table without allowing person B to look at it. Then person B tries to guess what person A has drawn by carefully listening to the sound that A has made. The more accurate guesses made, the higher the score that person B gets. The game illustrates the whole process, from people sketching to final recognition.

The second intuition for this algorithm is from the field of sketch recognition. Specifically, speed has been identified as an important factor for recognizing corners in strokes (Herot 1976; Sezgin, Stahovich, and Davis 2001). To give you an example, look at the square in Figure 1. Figure 2 shows the direction curve, and Figure 3 shows the curvature data, which is the second derivative of the direction curve. Notice the three spikes representing the corners. Figure 4 shows the speed data for this square. Notice that users tend to slow down at the corners. Speed equates to sound. Thus, sound gives us an efficient profile for recognizing hand-sketched information.

The major advantage of interacting through sound is that we no longer need additional costly devices such as multi-touch screens but rather only need one cheap, built-in microphone, such as one you would find in an iPhone or Android, and built-in recognition software, such as described in this paper. As long as users draw shapes on a nearby table, the built-in microphone can capture the shape information and record it as a sound wave signal within the system. Then the signal is recognized and the system gives appropriate feedback to the end users according to the recognition result. Another important advantage is that this encourages users to fully interact directly with the physical environment.

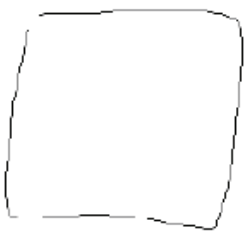

Figure 1: A square.

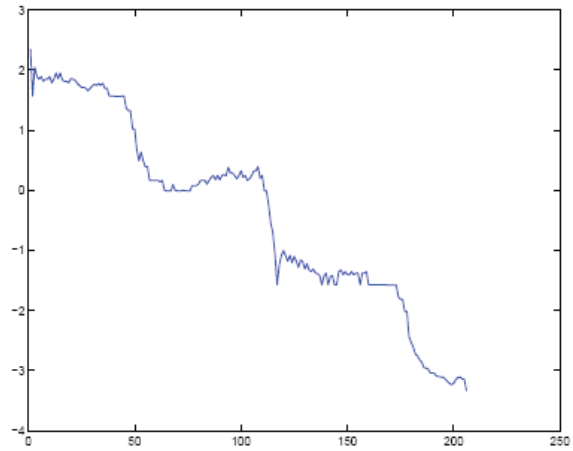

Figure 2: The direction change of a square over time.

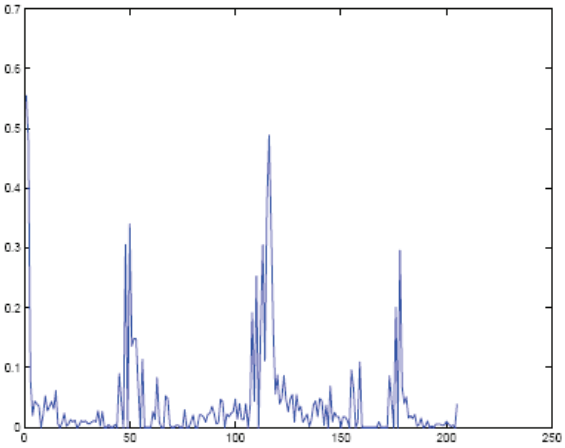

Figure 3: The change of curvature of a change over time. Notice the spikes at the corners

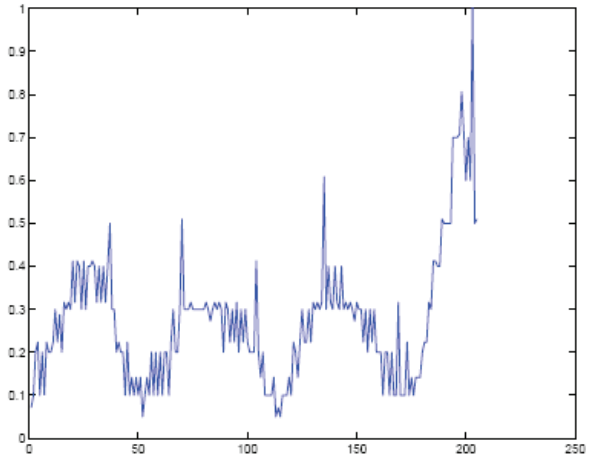

Figure 4: The speed graph of a square. Notice that people slow down at the corners.

In order to demonstrate our idea, we have built a lightweight novel sound recognizer to recognize a limited set of shapes. We use only the built-in microphone of a MacAir to detect and record the sound waves, and have tested our system also with standard iPhone and Android phones. We limited the distance range to 0.5m from our Mac computer in order to sufficiently capture the sound signal. Multi-sensor or multiple microphones could possibly be used to increase the distance or range and improve the system performance. For our initial algorithm, we restricted the stroke ordering to be fixed, although future implementations might allow various stroke orderings. Different stroke orderings will produce different sound waves that require additional training to account for the variations. For our initial implementation we noticed that users tend to draw each character in a similar style and stroke order each time.

Previous Work

Sratch Input (Harrison and Hudson 2007) is an acoustic based input technique that relies on the unique sound produced when a fingernail is dragged over the surface of a textured material and employs a digital stethoscope for sound recording. They conducted a study that shows users can perform six different gestures at about 90% accuracy. They employed a shallow decision tree primarily based on peak count and amplitude variation.

Another system has been developed to detect gestures drawn on non-electronic surfaces and focuses on the use of a textured pad and a wrist-mounted microphone to recognize different gestures drawn with a user's finger (Kim et al. 2007). The system is designed for small input surfaces, such as the back of an mp3 player. Their system is able to detect single-stroke representations of Arabic numbers, triangles, rectangles, and bent lines. The authors also outline ways to create wearable input surfaces such as wrist-mounted textured pads.

While the previous work listed focuses on developing novel user interfaces using only a few gestures, our research focuses on recognizing a larger set of 26 gestures. Our algorithm combines traditional gesture research with geometrical sketch recognition research that recognizes shapes by composing higher-level shapes from their primitive parts (Hammond 2005) through stroke segmentation, segment identification, and dynamic time warping. Sound analysis was performed from multiple features extracted from the input sound signal. Additionally, we use only one built-in microphone instead of using additional sensors or multiple microphones.

Implementation

We choose to recognize the 26 English characters listed in Figure 5 along with how each individual character is drawn. We collected samples using an ordinary wooden desk (1-2m width and 1-1.5m height) and a standard house key, pen and fingernail. The sound was collected by a built-in MacAir microphone. Figure 6 shows the overall system architecture.

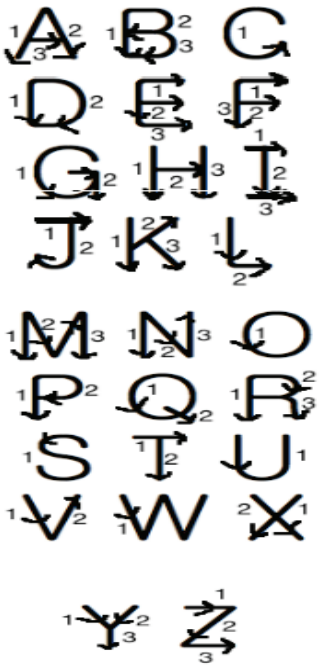

Figure 5. English characters and their drawing orders

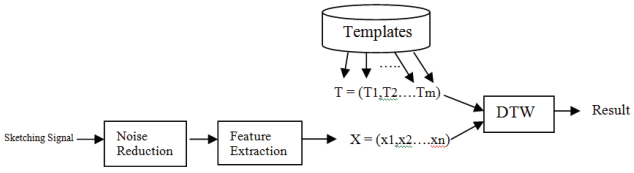

Figure 6. System Architecture

Noise Reduction

Although we expect our environment to be relatively quiet, there is undoubtedly noise in our sample data. We employed two different types of noise reduction methods, and the comparative results will be given in the result section.

Endpoint Noise Removal

Just as removing tails is important for recognizing strokes on a tablet PC (Paulson 2008), removing start and end noise is just as important in sound recognition. Endpoint noise removal provides a valid sound wave between the start point and end point of each sketching character. In order to detect the start and end points, we calculate the signal energy for the first 150ms using equation (1). We assume that this value represents the general environmental noise E_e. We then scan each frame 45ms at a time and calculate the mean energy for this frame, E_f. If $E_f > E_e * T$, for a fixed threshold, T[1], we set the current frame as start point. The same process applies to the end point detection.

$$E = \frac{1}{n}\sum_{i=1}^{n} S_i^2 \quad (1)$$

The simple process assumes that the environmental background noise is relatively constant. Figure 7 shows the original sound wave and final one for character 'A'.

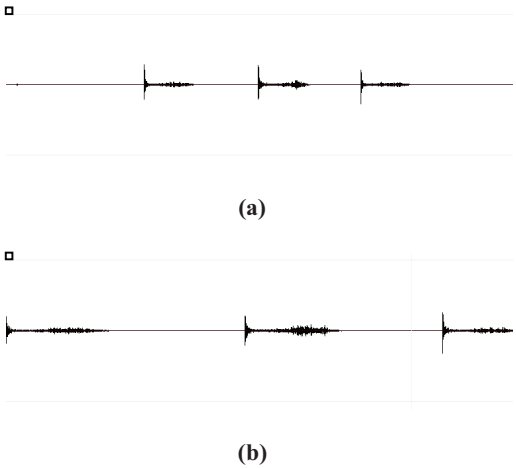

Figure 7. a) is original sound wave and (b) is the result wave after endpoint noise is removed.

Endpoint Noise Removal

A second noise reduction algorithm we use in our algorithm is similar to that proposed by (Saha, Chakroborty, and Senapati 2005). The algorithm assumes that the start of a signal input is environmental noise or silence. The algorithm calculates a Gaussian probability function for the first 20ms and obtains the mean μ and variance σ for this segment. Each subsequent 10ms segment is then examined and marked as either valid or silence if it is with in the Gaussian threshold, 3, using the formula $|x- \mu|/ \sigma < 3$. If at least half of all segments within this frame are valid, then the entire frame is marked as "valid", else it is marked as "silence". The algorithm then concatenates all valid frames together to form the resulting sound wave. We modified the algorithm to add an additional step before deleting and concatenating the frames. Specifically, for sketch (and not speech), we

[1] Empirical evidence shows 9 to be a good threshold.

consider each stroke to be a single block, and the silence frame within a sketch block shouldn't be removed, but rather kept in its entirety. For this purpose, we calculate the number of frames within sufficiently large window size (50 fragments) using a similar process to the second phase mentioned above, and require the entire window to be marked as silence for it to be removed

Normalization

Just as in sketch recognition, normalization is crucial to recognition. Due to the different sketching materials used as well as individual sketching habits, the input sound can have a wide variety of different amplitude values while the wave form remains the same. Thus, we need to normalize the input signal before extracting the features. We choose mean amplitude value as the normalization factor.

Feature Extraction

Feature extraction follows the normalization. Here, we use two different kinds of features, namely, the mean amplitude feature and MFCCs feature, which represent different aspects of sound.

Mean Amplitude

We calculate the mean amplitude value for each time frame. We set each frame size at 45ms, with an overlapping and size of 30ms, shown in Figure 8. The purpose of overlapping two consecutive frames is to make the features more consistent.

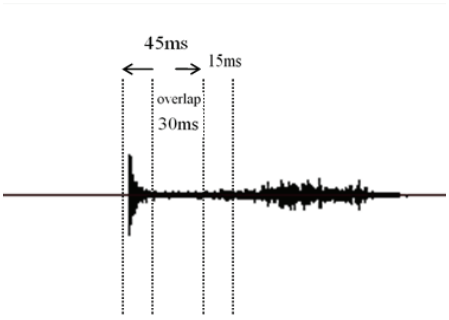

Figure 8 shows the mean amplitude used for feature extraction. Each frame size is specified at 45ms, whereas each overlapping size is specified as 30ms.

MFCCs

MFCCs (Mel-frequency cepstral coefficients) is one of the most widely used features in signal processing, and characterizes the dynamic change of the digital signal (Min 2004). We apply this to our sketching sound signal. The calculation process of MFCCs is out of this paper's scope, but the overall algorithm is as follows: 1.) Take the Fourier transform of (a windowed excerpt of) a signal. 2.) Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows. 3.) Take the logs of the powers at each of the mel frequencies. 4.) Take the discrete cosine transform of the list of mel log powers, as if it were a signal. 5.) The MFCCs are the amplitudes of the resulting spectrum (WP 2011). In our implementation, there are two major issues we should consider: a) We set the window size to 256. b) We chose the number of coefficient values to be the first 12 values excluding the first one, which proves to be harmful to the result. After applying this feature extraction process, the input signal is represented as vector $V = (v_1, v_2, \ldots, v_n)$, where each v_i is an MFCC feature which consists of 12 values.

Note that this feature extraction is essentially corner finding, and searches for periods of slow movement in the stroke. The output of feature extraction is a number of partial strokes that are segmented based on silence/slow movement that likely represents strong corners.

Template Matching

We then use dynamic time warping to calculate the distance between query $A = a_1, a_2, \ldots a_i, \ldots a_n$ and each template $B = b_1, b_2, \ldots b_i, \ldots b_n$ using equation (2) where c(k) is the actual mapping between candidate and template at time index k, w(k) is the weighting function, and N is the normalized value. Dynamic time warping will find the optimal path F to minimize D(A,B) (Myers, Rabiner, and Rosenberg 1980). Additionally, for the DTW constraints, we define the following:

$$D(A,B) = \frac{1}{N} \min_{F} \left[\sum_{k=1}^{K} d(c(k)) * w(k) \right] \quad (2)$$

- Endpoint constraint: The endpoint constraint decides where the mapping starts and ends. We start mapping at (1,1) and end at (N,M).
- Local continuity constraint: In order to avoid excessive compression or expansion of the time scales, neither the query nor the template can skip more than two frames at a time when matching. The five possible movements to get to (m, n) are (m-1, n), (m, n-1), (m-1, n-1), (m-2, n-1), (m-1, n02) as shown in Figure 9.
- Global path constraint: Because the signal is time-dependent, we set the boundary to control for each mapping range at time index k such that |A(k)-B(k)|<= R. we choose R equals to 20.
- Axis orientation. There are two variations. We can either put a candidate on the X-axis and the template on the Y-axis or put a template on the X-axis and a template on the Y-axis. However, (Myers, Rabiner and Rosenberg 2005) show that putting the query on the X-axis generally gives better result, thus we adopted that technique for this paper.
- Weighting function: We choose a symmetric weighting function, w(k) = A(k) – A(k-1) + B(k) – B(k-1), so that the

weighting value characterizes how many steps are from the current point to the previous one.

• Distance Measure. We use Euclidean distance for our distance measurement. One tricky part is to properly calculate the distance between two MFCCs features, which are multi-dimensional values. There are 12 dimensions for each feature value in MFCCs. In order to provide different scaling factors for each dimension within each feature, we define a normalization factor as the variance of each dimension.

In summary, our DTW algorithm can be written as follows:

D(n,m) = MIN(
D(n-1,m-1) + 2d(n,m),
D(n,m-1) + d(n,m),
D(n-1,m) + d(n,m),
D(n-2, n-1) + 3d(n,m),
D(n-1,m-2) + 3d(n,m))

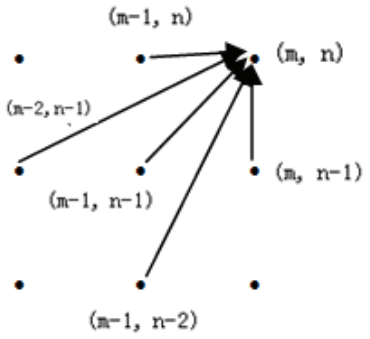

Figure 9. Local path constraint and five possible movements.

Results

We collected a total of 2340 samples from six persons. Each person produced five samples of each character using their fingernail, pen, and then key, totaling 390 samples per person (26*5*3 = 390).

Experiment 1

Our algorithm is user dependent, thus for each user we has a separate group of templates. For experiment 1, we had three example templates of each character; one template drawn through key scratch, one through pen, and the third with the fingernail. This provided a total total of 26 * 3 = 78 candidate templates. The remaining four samples were used for for testing. We repeat the same process five times for each user, each time using a different sample set as the templates. We tested our recognizer accuracy by using different noise reduction methods as well as different features. Because our recognizer is user-dependent, for the recognizer accuracy, we average through all six persons, and the results are shown in Table 1. From Table 1, we can see that MFCCs feature gives higher accurate result than pure mean amplitude feature.

	Noise reduction		*Feature*		*Accuracy*
Case	Endpoint noise removal	Guassian noise removal	Mean	Mfcc	
Case1	✓		✓		0.761
Case 2	✓			✓	0.834
Case 3		✓	✓		0.779
Case 4		✓		✓	0.812

***Table 1.* Recognition accuracy for different noise reduction algorithms and different features.**

Experiment 2

We performed a second test. In this case, rather than grouping templates of different input types, we separated them out. Rather than each query having 78 templates to compare against as in the earlier scenario, in this case, each query only had 26 templates to match against. The recognition is still user dependent, and query/templates from different users were kept completely separate, and as mentioned above, in this scenario only, the different input types were also kept completely separate. Again our templates included one sample of each character, and the other four were used for testing. We repeated this five times for each user, and then again for each user. The averaged results for all six users are shown in Table 2. For this experiment, we chose MFCCs for our default feature and endpoint noise removal for our default noise reduction method. The result shows that writing by fingernail gives a lower accuracy rate compared to using a pen or key.

	Key	*Pen*	*Finger*
Key	0.859	x	x
Pen	x	0.868	x
Finger	x	x	0.783

Table 2. Recognition accuracy for each input tool

Discussion

In this paper, we described our novel sound recognizer and have performed successful preliminary experimental studies. One interesting result is that the simpler endpoint noise removal approach yields more accurate results than the more complicated Gaussian noise removal approach.

One clue to this result is that it is hard to effectively control the Gaussian threshold to ensure that each sample is classified accurately. It is not robust enough for noise residing within a sound wave. Once such a misclassification occurs, the recognition almost always fails. Another important reason is that the beginning part of a sound wave often does not properly represent the silence part within the whole sound wave, which causes the sampling process to fail.

Another interesting problem that we found during the experiments is that the design variations of the DTW algorithm, such as choosing different local continuity constraints or weighting functions do not affect the result significantly. However, the number of coefficient values and windows size can greatly affect the recognition result. The parameters we chose in this paper are all tend to be optimal values that we found during our experiments.

An important question is, is 80% accuracy high enough for use in a real world system? On a per character basis it is hard to get above 90% accuracy simply because the sound profile from certain pairs of characters are almost the same. Table 3 shows some character pairs that have similar sound profiles. However, the ambiguity can be relieved by recognizing sequences of characters in context through the use of a dictionary. We did a preliminary study with a 500+ word dictionary that included the 500 most common English words plus others that were added manually. We assume each character within word is pre-segmented correctly using an extremely long space. By using Naive Bayes model, we correctly recognized 91% of collected words.

C	U
D	P
A	H
J	T
X	Y

Table 3. Pairs of characters that can be easily misclassified by each other

Future Work

In our current work we recognize only single characters at a time, requiring significant space or a click between characters. Future work directions include continuous character recognition. Second we would like to build a user independent system. One important observation is that sound signal variation is affected more by the material than by user behaviors. Proper clustering methods could be used to categorize all these variations of sketching environment and combine the automatic process for learning the environment before recognizing the input sound, which is similar to speaker identification in speech recognition. Third, we would like to use a hybrid feature set to improve the recognition accuracy. Fourth, a robust noise removing method is needed to make system work in public noisy situation. Future work will allow more fluid interaction. Finally, we would like to apply our results to a real and applied domain.

Conclusion

In this paper, we introduce a novel algorithm to recognize 26 English hand-sketched characters (A-Z) solely through sound with greater than 80% accuracy.

Acknowledgements

This work was supported in part by NSF, DARPA, and Google. Thanks to members of SRL.

References

Harrison, C. and Hudson, S, E. 2008. Scratch input: creating large, inexpensive, unpowered and mobile finger input surfaces. In *Proceedings of the 21st annual ACM symposium on User interface software and technology (UIST'08),* New York, NY.

Hammond, T., and Davis, R. 2005. LADDER: A Sketching Language for User Interface Developers, Computer and Graphics, *Elsevier*, pp. 518-532.

Herot, C.F. 1976. Graphical input through machine recognition of sketches. In *Proceedings of the 3rd Annual Conference on Computer Graphics and interactive Techniques*, Philadelphia, Pennsylvania, ACM, New York, NY, 97-102.

Kim, J. E., Sunwoo, J., Son, Y. K., Lee, D. W. and Cho, I. Y. 2007. A gestural input through finger writing on a textured pad. In *Proceedings of CHI'07 extended abstracts on Human factors in computing systems (CHI'07),* New York, NY.

Logan, B. 2000 Mel frequency cepstral coefficients for music modeling. *International Symposium on Music Information*. Plymouth, MA.

Min Xu et al. 2004. HMM-based audio keyword generation. In *Advances in Multimedia Information Processing - PCM 2004: 5th Pacific Rim Conference on Multimedia. Springer.*

Myers, C., Rabiner, L. and Rosenberg, A. 1980 Performance tradeoffs in dynamic time warping algorithms for isolated word recognition. In *IEEE Transactions on Acoustics, Speech and Signal Processing*.

Paulson, B. and Hammond, T. 2008. PaleoSketch: Accurate primitive sketch recognition and beautification. In 13th International Conference on Intelligent User Interfaces, pp. 1-10.

Saha, G., Chakroborty, I. and Senapati, S. 2005. A new silence removal and endpoint detection algorithms for speech and speaker recognition applications. In *Proceedings of NCC.* 56-61.

Sezgin, T.M., Stahovich, T., and Davis, R. 2001. Sketch based interfaces: Early processing for sketch understanding. In Proceedings of PUI '01, pp. 1-8.