Facial Electromyogram Activation as Silent Speech Method

Lisa Rebenitsch and Charles B. Owen

Media and Entertainment Technologies Laboratory (METLAB)
Michigan State University
East Lansing, MI, 48824
rebenits@msu.edu, cbowen@cse.msu

Abstract. A wide variety of alternative speech-free input methods have been developed, including speech recognition, gestural commands, and eye typing. These methods are beneficial not only for the disabled, but for situations where the hands are preoccupied. However, many of these methods are sensitive to noise, tolerate little movement, and require it to be the primary focus of the environment. Morse code offers an alternative when background noise cannot be managed. A Morse code-inspired application was developed employing electromyograms. Several muscles were explored to determine potential electrode sites that possessed good sensitivity and were robust to normal movement. The masseter jaw muscle was selected for later testing. The prototype application demonstrated that the jaw muscle can be used as a Morse "key" while being robust to normal speech.

Keywords: Silent Speech, Human computer interaction, User interfaces.

1 Introduction

Most devices used today for nonverbal communication, such as texting and internet browsing, typically require the use of one's hands. However, there are many instances where the hands mobility is limited. These instances can be either when the hands are otherwise occupied or when the hands lack the degree of control normally assumed. A few examples are operating machinery, driving, and working with disabilities. Attempting to temporarily shift the hands from the primary task can also be dangerous. The US Department of Transportation reports nearly half a million people were injured due to a distracted driving in 2009 [1]. This includes not only cell phone usage, but stereo control and air conditioning. New interface designs and technology attempt to decrease focus time on the interface, but interfaces still routinely require use of one's hands.

Many alternatives have been developed to permit hands-free input, such as speech recognition, gesture recognition, and eye focus/typing. However, speech recognition has limited accuracy, and its performance is degraded wherever there is background noise. Gestures are severely limited when the hands must be excluded, and eye focus/typing has the "Midas touch" issue wherein normal gaze movement are perceived

M. Kurosu (Ed.): Human-Computer Interaction, Part V, HCII 2013, LNCS 8008, pp. 464–473, 2013.

as actual commands [2, 3]. Morse code-inspired systems are an option where the hands are unavailable and have the added advantage of being robust to noise in the environment. Morse decoders can employ wireless systems to allow greater range of movement, and have been developed previously for environmental control systems [4]. However, this environmental control system required the use of a traditional Morse key, and therefore, the use of one's hands. As an alternative, the tightening and relaxing of muscles not on the hands or arms and their related bioelectrical (electromyogram) signal activations can be interpreted as the Morse signal.

We present a prototype system that uses a single electromyogram (EMG) electrode with a rate adaptive Morse decoder to create a hands-free typing interface in a partially mobile environment. To produce Morse code, the selected muscle must have a fair degree of control, limited interference from normal movement, easy detection, and a location where the sensing electrode is unobtrusive. The jaw fulfills these requirements.

One inherent feature of the jaw muscle is that the signal can bleed to other nearby electrodes due to its proximity to the skin surface and relative size to other nearby muscles. This implies that if multiple muscles are to be employed, nearby EMG signals from other muscles would require decoupling. A single muscle site was chosen to avoid this issue, disallowing the "2-button" Morse switch. The resulting "tighten and relax" sequence of the EMG is then analyzed by modeling the signal as a random zero-mean Gaussian signal with amplitude proportional to clench strength [5].

The Application Environment and Morse Decoding Algorithm are discussed in Section 3, while the Signal Processing and Signal Speed are discussed in Section 4. Hardware Specifications and Discussion are in Sections 5 and 6, respectively.

2 Background

The intent of the prototype was to determine the feasibility of an EMG based command system in a larger, partially mobile environment where retuning to the computer to adjust the system would be impractical. Other requirements were that the commands were to be a secondary focus so the commands would not interfere with the primary application, would be resistant to noise from normal motion, and would avoid false positives more strongly than false negatives. An application with similar requirements would be textual input in a web browser.

EMGs are inherently noisy signals. Morse code has been used as a communications medium for nearly two centuries due to its robustness in the presence of noise. Automatic decoding with reasonable accuracy has been available since as early as 1959 [6, 7]. More recently, machine decoding has demonstrated 96-98% accuracy rates, which is sufficiently accurate for command tasks [7]. Morse code by experienced operators has been reported at 35 words per minute (wmp) [7]. This is faster than reports on eye typing which often report speeds of 5-10 wmp with higher rates requiring word prediction methods [2, 3]. This is too slow for a command system and a much higher wmp with a "one-finger typing" restriction of eye typing should not be expected. Soukoreff and MacKenzie calculated the theoretical wmp bounds of using a stylus on a QWERTY keyboard, or "one-finger typing," to be 8.9 to 30.1 wmp [8]. Eye typing will slower to compensate for the Midas effect, although there are attempts to mitigate it. Isokoski explains that a primary reason for the Midas effect is

that the textual input and the application overlap, and therefore, both have primary focus at all times [9]. Isokoski moved the gaze switches to the sides of a monitor to remove this overlap which allowed for a dwell time under 100ms. While this method permits textual input to be a secondary task, it does not permit partial mobility.

The bounds calculate by Soukoreff and MacKenzie can also can be applied to Morse code since the bounds were calculated using the bits (choices/clicks) per second [8]. Those "clicks" can be considered to be Morse key presses, and given the shorter distance between "clicks," the slightly higher than theoretical upper bounds of 35 wmp for Morse code is unsurprising. Therefore, these theoretical bounds were used as an objective feasibility measure of our Morse code-inspired system. However, to keep the hands free, a different input source is required. We used the jaw's EMG in our system. The selection criteria of muscle are discussed more in 3.1.

Other methods have been used for hand-free interfaces. MacKenzie provides an overview of one key (button) keyboard interfaces and their issues [10]. These interfaces are often intended for the disabled who cannot use a normal keyboard or for small devices such as some cell phones that cannot support a full keyboard. There are two primary categories: scanning the keys until the wanted character is reached, and a hierarchal method where a subset of options is repeatedly selected until only one character/command remains. Morse code, while consisting of one or two switches, does not require a keyboard counterpart and thus allows direct input into the device. As a consequence the character codes must be memorized rather than searched for visually. The advantage to this method is that it can be much faster overall.

Systems similar to our own use EMGs as input. Felzer and Nordmann created a Morse code-inspired system that uses the eyebrow's EMG [11]. Their encoding uses the direction of the cursor, which is set by the number of EMG pulses. While the system is fairly intuitive in that the state is easily seen by cursor motion, it requires more muscle clenches than Morse code. For example, the letter A would require 4 clenches while Morse code would require 2. The worst case would require 8 clenches. Nilas, Rani, and Sarka present another option that uses the eyebrow EMG as the Morse code signal [12]. They restricted the set of characters to decrease learning time and mapped the characters to high level robot commands such as "move left." However, their method would require it to be the primary focus to avoid false positives. Park et. al. developed a system similar to our own designed for disabled users [13]. An EMG on the masseter was used as input into a Morse decoder. However, there are sparse details on the implementation, and whether it could be used as a secondary input in a partially mobile environment is ambiguous.

3 Application Environment

Our application is a GUI coded in C# and C using a single EMG signal. The Morse decoding application is comprised of several state variables, such as the letters decoded thus far, the approximate words per minute, and a feedback character of the current state as an aid to new users. An example of the decoder window is available in Fig. 1. The Morse element categories of the decoder are `dot(dit)`, `dash`, the different spaces, and `undefined`. A chart of the feedback character for each element is available in Table 1. A detailed explanation of the Hardware Specifications is available in section 5.

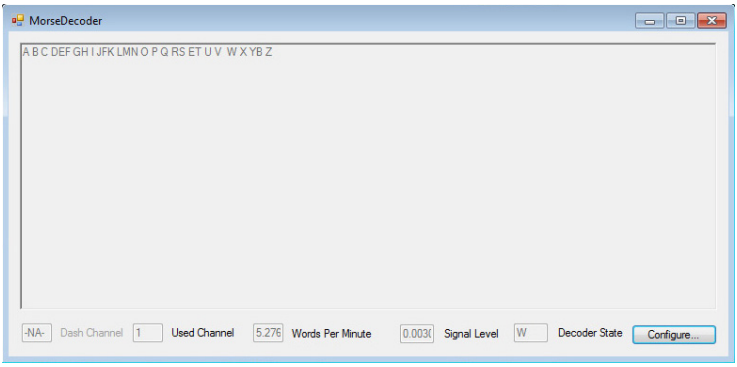

Fig. 1. Morse decoder application screen shot after approximately an hour of practice

Table 1. Feedback Character for each Morse Element

Element	Feedback Character
Dot	.
Dash	_
Inter-element_space	I
Character_space	C
Word_space	W
Undefined	no feedback

3.1 Electrode Placement

When selecting potential muscles for the sensing electrode placement, a few requirements are enforced: the muscle may not be on the arms or hands, the muscle is large enough to allow for minor electrode displacement, the electrode is not inhibitive to task, the electrode causes minimal discomfort, the electrode is robust to minor movement, and the muscle must be capable of a fair degree of control and strength.

Clenching the jaw muscle (masseter) fit these requirements. It has a fair degree of control, yields a strong signal, and the electrode placed on the rear of the jaw is less obtrusive. Other candidates were the neck, eyebrow, and eye movement. However, the neck and eye movement resulted in numerous false positives from normal motion, and thus would have required textual input to be the primary focus at all times. The activations from normal speech were sufficiently lower than that of a clenched jaw to avoid most of these false positives. The eyebrow had fewer, but still many, false positives, and was the most obtrusive of all candidate locations. The positive electrode was placed on the rear of the left jaw on the masseter muscle with the negative electrode placed immediately below, as shown in Fig. 2. The ground electrode was placed on the back of the right hand or forearm for convenience.

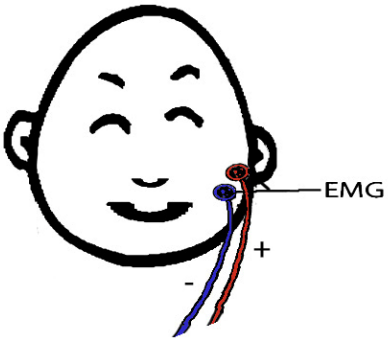

Fig. 2. Placement of negative and positive electrodes

3.2 Morse Decoding Algorithm

The Morse decoder is a multistate system derived from Kyriazis "xdemorse" program, and is distributed under the terms of the GNU General Public License [14]. The decoder code was extracted and adapted for use in our system. The input signal of original project is replaced by muscle activity to represent the "on/off" switch. The training and dot-dash detection sections were then modified to better accommodate the jaw EMG signal. The dot-dash detection and training algorithms were isolated to later allow potential substitution of other Morse decoding algorithms and contextual rules.

The decoder breaks the incoming signal into a string of 250 bits per second, called "fragments." Each fragment is encoded as a true (jaw clenched) or false (jaw relaxed). Each fragment is then fed to the decoder to update the decoder state. A broad overview of the state machine is provided in Fig. 2. Each state has the ability to either wait or progress to a new state depending on the incoming fragment, and return a decoded letter or feedback, if required.

Each fragment is parsed individually. If the fragment does not cause a change in state, feedback is provided as an aid for those new to Morse code. In the `MARK_SIGNAL`, `CHAR_SPACE`, and `WAIT_WORD_SPACE` states, the length of the sequence is checked to determine the current element (e.g. dot, character space) and is then displayed to the user as feedback. If the fragment does cause a change in state, the sequence is decoded. In the feedback states mentioned before and `ELEM_SPACE` and `WORD_SPACE`, the length of the current input string is compared against the stored threshold lengths for each Morse element.

Once an element is decided, it is encoded into a binary string and then read as a hexadecimal value. The initial string is 0x01, and then if the element is a dot, a 1 is shifted in. If it is a dash, a 0 is shifted in. For example, an "E" (single dot), is encoded as a 0x03 and a "T" (single dash), is encoded as a 0x02. This encoding is then indexed into an array of characters to decode the element set. A space element is stored separately as the length of time.

Upon decoding the sequence, the current lengths of the elements are stored for later use by the adaptation algorithm, and the lengths are reset to 0. Upon reaching the end of a word, the adaptation algorithm is called. The algorithm is described in more detail in section 4.3.

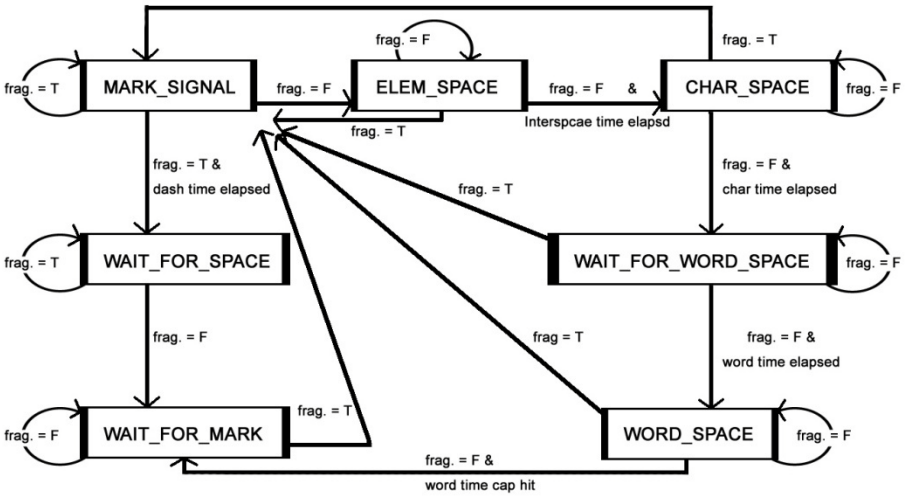

Fig. 3. State change diagram Morse decoder algorithm

3.3 Further Concerns

When assessing the training algorithm, the jaw demonstrated a slower response time and greater variance in signal lengths then those expected from hand generated code. In an effort to counter the short Morse spacing, the jaw tended to remain slightly tensed which quickly resulted in fatigue. Thus, the algorithm was modified to permit greater spacing between the dots and dashes, and the threshold lengths were modified to be the midpoint between two element lengths. In addition, previously trained threshold lengths were loaded rather than beginning with default settings in which the operator was trying to "force" a certain response.

4 Signal Processing

The incoming signal is an EMG signal at 250 samples per second, with a gain of 1000. The EMG signal is modeled as a zero-mean Gaussian distribution noise signal with amplitude proportional to the clench strength [5]. However, cyclical noise from neighboring wires, poor electrode contact, signal shifts, speech, and other noise sources remain in the incoming signal resulting in the raw amplitude being unreliable. Therefore, the signal must be preprocessed to remove the outside noise, and then smoothed for detection of activation time and length.

4.1 Outside Noise Removal

Temporal whitening has been shown to improve the amplitude estimation for EMG signals [5, 15] and has the added benefit of removing cyclical noise. The outside cyclical 60 Hz information from nearby wires and electrode drift is first removed from

the EMG signal. This is performed through an adaptive Wiener whitening filter derived from the algorithm described by Baldwin [16]. For a 250 Hz signal, the parameters were a window size of 35 and a filter feedback constant of .05. This filter adapts from a zeroed initial state to remove stable cyclical noise from signal within a few seconds. Afterwards, mild to moderate noise shifts in the incoming signal are removed rapidly. This whitening filter has the added advantage of making the signal have a mean of 0, which effectively removes electrode drift in one sweep of the signal. An image of the initial signal and the whitening filter output is shown in Fig. 4.

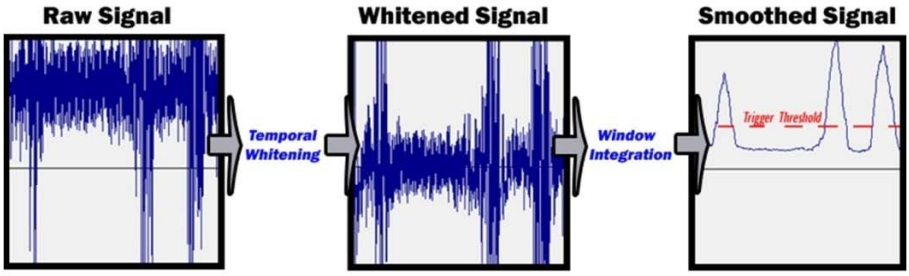

Fig. 4. Signal processing progression (The threshold line is added for visual clarity)

4.2 Threshold Smoothing

After the signal is temporally whitened, it is smoothed so that it can be compared against a threshold limit. To permit greater time for improved decoding algorithms, the EMG signal is smoothed via a fast window integration function with a width of approximately 150 milliseconds, or 35 samples at the 250 Hz sample rate. The window size was chosen empirically applying the tradeoff of signal-to-noise ratio as described by St-Amant, Rancourt, and Clancy [17]. The output signal is then compared directly to an activation threshold. Should the signal amplitude surpass the threshold, the muscle state is changed to "on" until the amplitude drops below the threshold. The threshold is static and is manually chosen by the user. The threshold normally only changes if the skill level of the user changes. For example, when training there is a tendency to strongly clench the jaw. After the Morse decoder has been trained, there is not the perceived need to be as forceful. While a changing threshold, such as the method provided by Nilas, Rani, and Sarka, which reacts much more strongly to the initial increases could be used, it carries a risk of many false positives during normal speech [12].

4.3 Signal Speed Adaptation

A user's Morse rate changes with practice, fatigue, and focus on the Morse code versus the environment. Although the system provides an option for static element lengths for practice, a rate adapting algorithm is essential. The original algorithm adjusted the basic element threshold length (`inter-element_space`) at the end of characters and words. If both dots and dashes were present, the basic element

threshold is updated by analyzing the fragments since the last threshold update. The higher the estimated character/word count, the shorter the next threshold. The other element thresholds were then set to multiples of the `inter-element_space`.

However, this algorithm was found to have poor accuracy for the jaw EMG signal. The algorithm is modified so that it is performed at the end of the word, but only when both dots and dashes were present to assure a comparison of the elements. The adaptation algorithm then determines the average length of the all Morse elements in the current word. It then uses the weighted average of the original element thresholds and current averaged element thresholds to update. However, to prevent the elements from overlapping, each element threshold is given a minimum and maximum cap based on the next smallest related element. These lengths are then used in the decoding process as described in section 3.

5 Hardware Specifications

The Morse decoder application was run on a Toshiba Satellite 505 with 4GB of memory running Windows 7. The electromyogram amplifier (EMG100C) of a BIOPAC MP150 system, a bioelectric suite from Biopac Systems, Inc., was used. The gain on the BIOPAC module was set to 1000, the low pass band filer was set to 100 Hz, the high pass filter was set to 1Hz, and the 100Hz high pass filter was turned off. Foam electrodes with clip leads from Biopac Systems were used. The operator had the option to sit or stand in front of the monitor at a distance of their choosing.

6 Discussion

A short informal experiment was performed to examine the feasibility of the system. The electrodes were attached as explained in section 3.1, and the applicant was trained for a few minutes by inputting SOS, and then the element lengths were saved. After approximately one hour of practice with a reference chart, the novice attempted printing the alphabet while stopping approximately 3/4 through to talk normally. An example result is available in screen shot in Fig. 1. A few mistakes were made (some not shown due to an added erase command). Surprisingly, normal speech had little effect on the output beyond a rare "E". A novice's input is slow at only about 5 wmp which is below the lower bound of 8.9 calculated by Soukoreff and MacKenzie [8]. This lower value may be due to the slower response time of the jaw, and the user's unfamiliarity with Morse code. There were many delays due to referencing the Morse code chart. Had the codes been memorized, the wmp would likely have doubled.

Learning Morse code takes time and practice, but was chosen since it is a well known standard code that handles noisy environments well. The time can be dramatically decreased when using a restricted set of input codes as suggested by Nilas, Rani, and Sarka [12]. A set of five to ten input commands would be easier to learn and could consist of commands such as stop, start new file, previous, next, select, and other application specific commands. These commands are available in most standard computer software, but are often not available in non-traditional interfaces.

Areas where the hands are otherwise engaged would receive the greatest benefit, such as in cockpit simulators, machine operation where the hands must stay on the controls and noise limits speech recognition, and disabled users whose hands do not have sufficient control over traditional interfaces.

The Morse decoder for the jaw muscle currently employs an exact pattern match for character retrieval and a fast weighting scheme for updating. Better decoding algorithms, such as fuzzy logic, word completion, and spell checking for code sequences with errors would increase accuracy. Changing the update algorithm to store previous examples of elements and/or characters should also improve accuracy and usability of the system.

References

1. US Department of Transportation, National Highway Traffic Safety Administration, Distracted Driving 2009 (September 2010), http://www.distraction.gov/research/PDF-Files/Distracted-Driving-2009.pdf (accessed January 2012)
2. Majaranta, P., Räihä, K.-J.: Twenty Years of Eye Typing: Systems and Design Issues. In: Proceedings of Eye Tracking Research and Applications, pp. 15–22 (2002)
3. Hansen, D.W., Hansen, J.P., Niels, M., Johansen, A.S.: Eye Typing using Markov and Active Appearance Models. In: Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision, pp. 132–136 (2002)
4. Yang, C.-H., Chuang, L.-Y., Yang, C.-H., Luo, C.-H.: Morse Code Application for Wireless Environmental Control Systems for Severely Disabled Individuals. IEEE Transactions on Rehabilitation Engineering 11(4), 463–469 (2003)
5. Clancy, E.A., Hogan, N.: Single Site Electromyograph Amplitude Estimation. IEEE Transactions on Biomedical Engineering 41(2), 159–167 (1994)
6. Gold, B.: Machine Recognition of Hand-Sent Morse Code. IRE Transactions on Information Theory 5(1), 17–24 (1959)
7. Blair, C.R.: On Computer Transcription of Manual Morse. Journal of the ACM 6(3), 429–442 (1959)
8. Soukoreff, R.W., MacKenzie, I.S.: Theoretical Upper and Lower Bounds on Typing Speed Using a Stylus and Soft Keyboard. Behaviour & Information Technology 14, 370–379 (1995)
9. Isokoski, P.: Text Input Methods for Eye Trackers Using. In: Proceedings of the 2000 Symposium on Eye Tracking Research & Applications, pp. 15–21 (November 2000)
10. MacKenzie, I.S.: The One-Key Challenge: Searching for a Fast One-Key Text Entry Method. In: Proceedings of the 11th International ACM SIGACCESS Conference on Computers and Accessibility, New York (2009)
11. Felzer, T., Nordmann, R.: Alternative Text Entry Using Different Input Methods. In: Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility, pp. 10–17 (October 2006)
12. Nilas, P., Rani, P., Sarkar, N.: An innovative high-level human-robot interaction for disabled persons. In: Proceedings of Internation. Conference on Robotics and Automation, New Orleans (2004)
13. Park, H.-J., Kwon, S.-H., Kim, H.-C., Park, K.-S.: Adaptive EMG-driven communication for the disabled. In: Proceedings of BMES/EMBS Conference Serving Humanity, Advancing Technology, Atlanta (1999)

14. Kyriazis, N.: Linux Morse Code Decoding Software (2002), http://www.qsl.net/5b4az/pages/morse.html (accessed 2011)
15. Clancy, E.A., Farry, K.A.: Adaptive Whitening of the Electromyogram to Improve Amplitude Estimation. IEEE Transactions on Biomedical Engineering 47(6), 709–719 (2000)
16. Baldwin, R.G.: An Adaptive Whitening Filter in Java (November 1, 2005), http://www.developer.com/java/other/article.php/3560501 (accessed 2010)
17. St-Amant, Y., Rancourt, D., Clancy, E.A.: Influence of Smoothing Window Length on Electromyogram Amplitude Estimates. IEEE Transactions on Biomedical Engineering 45(6), 795–799 (1998)