34th Annual International Conference of the IEEE EMBS
San Diego, California USA, 28 August - 1 September, 2012

Quantitative Assessment of Magnetic Sensor Signal Processing Algorithms in a Wireless Tongue-Operated Assistive Technology

Abner Ayala-Acevedo and Maysam Ghovanloo, *Senior Member, IEEE*

Abstract— **In this paper, we evaluate the overall performance of various magnetic-sensor signal processing (mSSP) algorithms for the Tongue Drive System based on a comprehensive dataset collected from trials with a total of eight able-bodied subjects. More specifically, we measure the performance of nine classifiers on the two-stage classification used by the mSSP algorithm, in order to learn how to improve the current algorithm. Results show that is it possible to reduce misclassification error from 5.95% and 20.13% to 3.98% and 5.63%, from the two assessed datasets, respectively, without sacrificing correctness. Furthermore, since the mSSP algorithm must run in real time, the results show where to focus the computational resources when they are constrained by the platforms with limited resources, such as smartphones.**

I. Introduction

In the United States alone there are an estimated 259,000 persons with some type of spinal cord injury (SCI), with approximately 12,000 new cases added every year. Within this population, 47% are quadriplegics who cannot benefit from the standard input devices that able-bodied and lower body paraplegics can use to access computers, and interact with the environment. Over the years, the average age post-injury has increased from 32 years to 40 years, and currently the life expectancy for quadriplegics ranges to around their sixties. This can be translated into approximately twenty years of medical support that average ~$800,000 for the first year, and ~$143,500 for each subsequent year [1].

To significantly reduce this billion dollar annual cost and improve the quality of life for these individuals, assistive technologies (AT) for people with SCI aim to allow them as much independence as possible, thus reducing the need for constant care. Although many ATs have been developed for people with different types of motor impairments, these solutions have mostly been user-specific and focus on the limited abilities of the individual, not to mention their limited number of switch-based commands. For instance, the popular sip-and-puff system offers only four discrete commands.

By developing the Tongue Drive Systems (TDS) we aim to reduce the healthcare costs by giving the people with SCI more independence via a universal solution [2]. TDS gives its users access to their environments, powered wheelchairs (PWC), computers, and smartphones by using their tongues as an input device. The tongue has been known to have rich motor and sensory capabilities that can match the fingers of our hands. Moreover, it does not fatigue easily [3], [4].

The current external TDS (eTDS) uses four 3-axis magneto-resistive sensors in the form of a wireless headset, and read the movements of a magnetic tracer attached to the tongue, ~1 cm posterior to the tip [5]. Currently, the eTDS is able to provide up to six individual tongue commands plus a neutral command, which is the tongue resting position. At the core of this system we have the magnetic sensor signal processing (mSSP) algorithm, which translates the tongue movements into issued commands. The mSSP algorithm has evolved through the years and now uses raw data from four 3-axial magnetic sensors, mounted bilaterally near the users' cheeks. Multiple data fusion, feature extraction, and classification schemes have constantly improved the overall performance of the TDS [5].

In this paper we assess the performance of the current mSSP algorithm in order to identify its weaknesses and further improve the human-computer interaction (HCI) via tongue motion. In the following section we have detailed the current mSSP, the experimental dataset, and the basis of our evaluation. Section III describes the results of the mSSP performance assessment. Finally, we discuss potential improvements to the current mSSP algorithm.

II. Methodology

A. Magnetic Sensor Signal Processing Algorithm

In order to cancel the external magnetic interference (EMI), including the earth's magnetic field, the mSSP algorithm first calibrates the sensors and reduces the EMI by applying a stereo-differential noise cancellation method, discussed in [2]. After calibrating the raw sensors data, it uses a two-stage classification. The first stage of the mSSP algorithm simply distinguishes whether the tongue is on the left or right side of the mouth. This is accomplished by comparing the root mean square (RMS) amplitudes of the readings from the left vs. right magnetic sensors. The second stage of the classifier uses a simple majority voting scheme between nine distance-based classifiers that are listed in the following, without setting any majority threshold. It is also important to mention that the current mSSP uses PCA feature extraction of three consecutive incoming set of samples to reduce the dimensionality of the data.

B. Classifiers

1) *Linear:* multivariate normal distribution of data, with an estimated covariance matrix that is identical for each class.
2) Diagonal Linear: similar to the linear classifier, with the exception that it calculates an estimate of the diagonal covariance matrix.

This work was supported in part by the National Institute of Biomedical Imaging and Bioengineering grant 1RC1EB010915, the National Science Foundation awards CBET-0828882 and IIS-0803184, and the National GEM Consortium Fellowship, sponsored by IBM.

Abner Ayala-Acevedo and Maysam Ghovanloo are with the GT-Bionics Lab, School of Electrical and Computer Engineering at the Georgia Institute of Technology, Atlanta GA, USA (Email: abner.ayala@gatech.edu, mgh@gatech.edu).

3) Quadratic: multivariate normal distribution of data, but unlike linear classifier, it estimates a different covariance matrix for each class.
4) Diagonal Quadratic: similar to the quadratic classifier, except that it utilizes the diagonal covariance matrix estimate.
5) Mahalanobis: similar to quadratic classifier, except that it assumes Gaussian distribution and utilizes Mahalanobis distance to calculate the covariance matrix.
6) *KNN Euclidean:* k-nearest neighbor (KNN) classifier that utilizes Euclidean distance.
7) *KNN City Block:* KNN classifier that calculates absolute value distance.
8) KNN Cosine: KNN classifier that measures the angle between the points.
9) KNN Correlation: KNN classifier that utilizes correlation distance.

C. Classifiers Combinations

1) *Stacking:* using multiple classifier stages, where the output of a preceding classifier becomes the input for the succeeding classifier, and making the final decision on the last stage [6]. In this method, since correct classification of the succeeding level is entirely dependent on the correctness of the preceding level(s), the earlier classifications must be quite reliable in order not to introduce errors, which would then propagate throughout the following stages and change the final decision.

2) Majority Voting: the final decision is made based on the class that receives majority of the votes among classifiers. Results in [7] showed that this scheme was the most robust one by achieving lower misclassification error at the cost of a slight reduction in correct classification.

D. Correctness, Misclassification, and Rejection

To assess the performance of the mSSP algorithm and the majority voting scheme, we should distinguish between correctness, misclassification error, and rejection error.

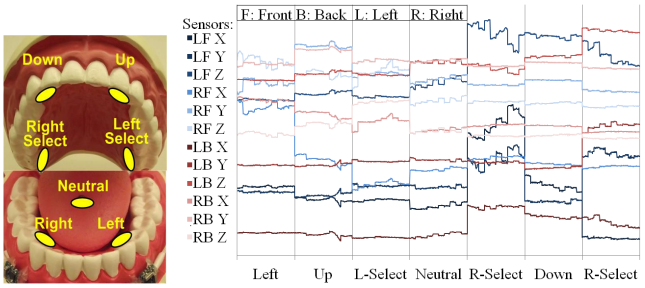
1) *Rejection Error:* this error occurs when there is a tie on majority voting scheme or when the final voted class does not reach the majority voting threshold. In this case, the mSSP classifies the incoming command as resting/neutral.

2) *Misclassification Error:* this error occurs when a wrong class achieves the majority of classifier votes, and a wrong command (i.e. not intended by the TDS user) ensues. These misclassification errors are considered false negatives. It should be noted that the individual classifier outputs (0 to 6) are either right or wrong, but not "rejection".

3) Correctness: correctly classified commands.

E. Dataset

In order to assess the performance of the current mSSP algorithm, we have analyzed the collected data from human subject trials with eight able-bodied volunteers at the Northwestern University (NUB) and Georgia Institute of Technology (GTB). Details of the TDS trials with this group of subjects, who have been referred to as "Group-B", can be found in our prior publication [8].



(a) (b)

Figure 1. (a) Recommended tongue positions for six TDS commands plus the tongue resting position, which is considered neutral [8]. (b) Sample magnetic sensors waveforms from each command (raw data).

Each of the eight participants took part in five sessions, accounting for a total of 40 experiments that were analyzed. The data used in this paper was taken from the Information Transfer Rate (ITR) test, in which participants were cued to issue a randomly selected command as fast and accurately as possible [8]. It should be noted that during this test no resting commands was requested. Therefore, we do not consider the issuance of the resting as a correct command.

III. Experimental Results

Figs. 1a and 1b show the recommended tongue positions for six TDS commands and a sample of the resulting waveforms from magnetic sensors for each command (raw data), respectively. Following training, the mSSP is tasked with processing the raw data, sampled at 50 Hz, in real time and identify the TDS command that is intended by the user. All experiments, except for A. Left vs. Right Classification, use a majority voting scheme for their evaluation.

A. First Stage: Left vs. Right Classification

The first stage of the classifier identifies whether the tongue is on the left or right side of the mouth [5]. Identifying the tongue resting position (the neutral command) in the midline is postponed to the second stage. The results obtained in Fig. 2 shows that the current algorithm, which only uses RMS distance for left vs. right classification, has introduced a significant amount of error in the first stage, i.e. 2.02% and 20.1% for GTB and NUB, respectively, which has consequently affected the overall accuracy of the mSSP.

On the other hand, we observed that if the majority voting method among the nine classifiers used in the 2nd stage were also used in the first stage, this error would have significantly dropped to 0.13% and 1.6% for GTB and NUB, respectively. On the down side, this would have almost doubled the computational cost. However, it can also be seen in Fig. 2 that using a simple quadratic classifier can lead to classification accuracy close to the majority voting scheme (0.13% for GTB and 2.13% for NUB). Also note the poor performance of both linear and diagonal linear distance classifiers.

B. Second Stage: TDS Command Classification

This stage uses the majority voting of all nine classifiers to make the final class assignment. However, since we have already identified whether the tongue is on the left or right side of the mouth in the first stage, the classification will only be between the three commands that are on that side of the

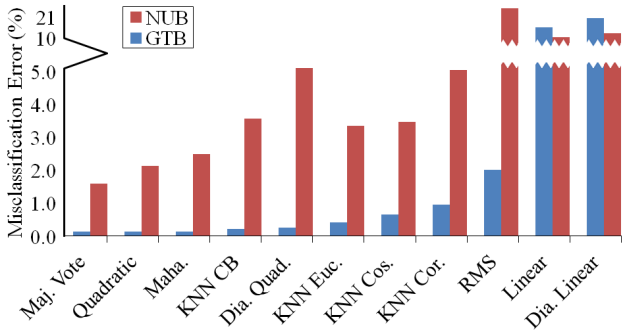


Figure 2. First stage (left vs. right) classification performance.

mouth (see Fig. 1a), plus the neutral command which will be classified on either side, i.e. one out of four classes.

Results in Fig. 4 show that the majority voting does represent the lowest misclassification error with 3.98% and 5.63% for GTB and NUB, respectively. Among the nine classifiers, KNN City Block was the best on average. However, it was not consistently the best classifier for every experiment, and its overall performance was considerably worse that the majority voting results.

C. *Majority Voting Threshold*

Here we intend to determine the optimal threshold for the majority voting scheme, i.e. what minimum percentage of the classifiers should vote for a class for it to be selected as the final decision. The higher the voting threshold, the higher the accuracy of the selected command (lower misclassification error), but also the higher would be the rejection error rate. A trade-off between correctness and robustness of the system has to be made, to optimally choose a majority voting threshold. We consider the TDS more robust if we lower the number of misclassification errors, even if it leads to classifying some of the issued commands as neutral.

In order to identify the optimal voting threshold, one should also consider the task being accomplished. For instance, when driving a wheelchair, issuance of a wrong command, e.g. turning left instead of right, can have dire consequences. Therefore, we need a very robust mSSP algorithm. In other words, it is better to ignore a questionable command, and have the user try it again than making a wrong decision. On the other hand, when surfing the web, the user may want the TDS to be very responsive, while making a wrong choice would not be very costly.

In Fig. 3, we have shown the effect of voting threshold on

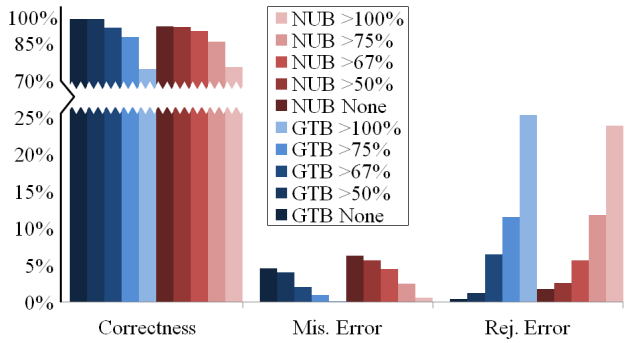


Figure 3. Majority voting results with various threshold levels.

	Maj. Vote	KNN CB	KNN Euc.	Dia. Linear	KNN Cos.	KNN Cor.	Linear	Quad.	Maha.	Dia. Quad.
GTB	3.98%	5.12%	5.28%	6.67%	7.27%	7.86%	9.07%	10.93	11.18%	13.92
NUB	5.63%	8.78%	8.86%	11.47%	8.74%	11.33%	10.82	11.18%	11.24%	10.80

Figure 4. Second stage classifiers' rate of misclassification. Majority voting is reported at 50% threshold.

the two types of error and correctly selected commands. As mentioned above, this threshold can be adjustable, depending on the task. When driving a PWC, where safety is paramount, misclassification error can be reduced by choosing 100% threshold. Meaning that all classifiers should agree on the final command. Misclassification in this case was only 0.04% and 0.57% for GTB and NUB, respectively, at the cost of decreasing the rate of correctly selected commands to 74.7% and 75.6%, respectively. On the other hand, when we sat no voting threshold, correctness increased to 95% and 92% for GTB and NUB, respectively, at the cost of 4.58% and 6.23% misclassification, respectively. Fig. 4 shows the rate of misclassification for different classifiers.

D. *PCA vs. Raw Data*

We observed considerable variability between raw data associated with specific commands issued by each subject between different experiments, while the variability of the raw data within each experiment was quite low. This could be attributed to the positioning of the TDS headset and re-calibration, which was done at the beginning of each session, but rarely within the session (unless the headset shifted) [8].

This finding raised the question of whether there is a need for the principal component analysis (PCA), while applying the stereo-differential noise cancelation readily reduces the dimensionality of the raw data by half (from 12 to 6). Fig. 5 shows that in fact there is no need for applying the PCA feature extraction scheme, especially when it does not have a significant impact on lowering the computational load.

E. *Number of Samples per Command*

The current algorithm utilizes 40 middle samples collected per command during training to construct each

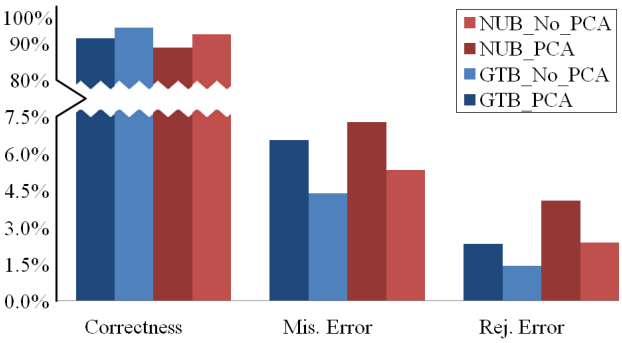


Figure 5. The mSSP performance with and without PCA.

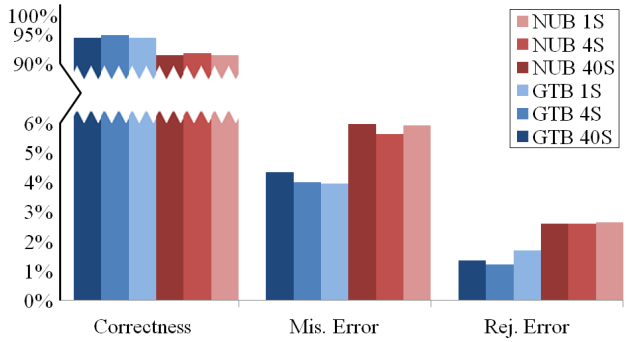


Figure 6. The effect of number of samples maintained per command to construct each class during training.

class. However, because the KNN classifiers require all data points to be saved, this translates to more required memory and higher computational burden. These are not suitable for a future version of the mSSP, which is going to be embedded within the headset. Therefore, we evaluated the effect of reducing the number of samples that we collect and associate to each class during the training phase.

Fig. 6 shows that because of the low variability in raw data points within each experiment, representing the data points for each class with only 4 training samples result in about the same accuracy as 40 samples in the current mSSP, while still providing sufficient redundancy in the system. This can also lead to a faster training in our enhanced mSSP.

F. Number of Repetitions per Command

In another test we studied the relationship between the numbers of repetitions required per command during training and the overall mSSP accuracy. As expected, Fig. 7 shows that as we decrease the number of repetitions per command, there is a reduction in the mSSP accuracy. However, this experiment shows that the amount of loss in accuracy from 5 to 10 repetitions is small. Therefore, as the TDS users become more familiar with the system, the training phase can be shortened without major negative consequences.

IV. Discussion

The results obtained, show that simple distance based classifiers, such as the quadratic classifier and Mahalanobis, perform well under conditions where the distances between classes are large and well defined. When the distances are short, the KNNs classifiers offer a better solution.

By using a two-stage classification we reduce the number

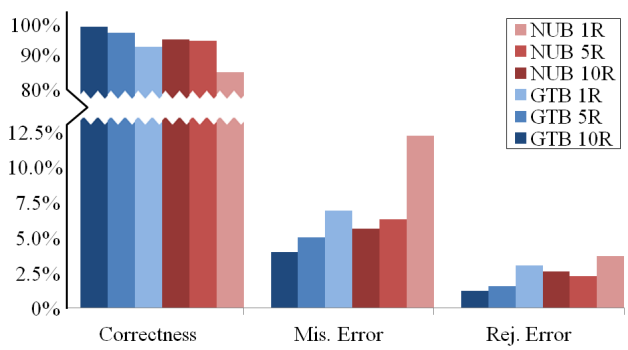


Figure 7. The effects of number of repetitions per command in the training phase.

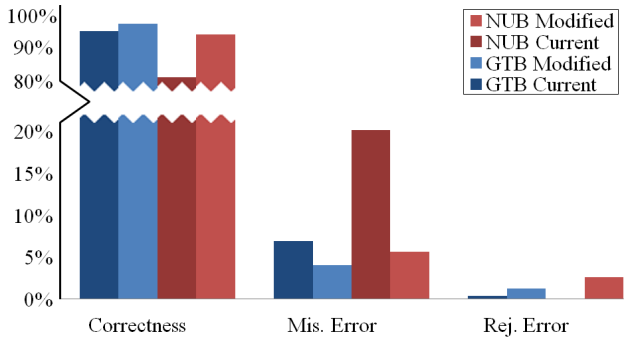


Figure 8. Improved mSSP algorithm vs. current mSSP algorithm.

of classes in the second stage classes by half, reducing the computational load if compared to older versions of the mSSP. Alternatively the number of individual commands can be increased in the future with comparable performance.

By applying an adjustable task-based threshold on the majority voting scheme we can tune the system robustness and responsiveness for the optimal TDS user experience.

We should point out that in the ITR test subjects are not requested to issue the resting commands. Hence, the presented results might be slightly different from the exact measurements of the current mSSP and system accuracy.

V. Conclusion

Quantitative assessment of the current mSSP clearly shows the key areas where we can improve the current mSSP algorithm, and where the limited computational resources of the system should be focused. We can understand the effects of applying new constraints or simplifying an algorithm on the overall performance of the TDS using real field data. Fig. 8 shows the impact of applying some of these conclusions to the current mSSP algorithm, resulting in an improvement from 5.95% and 20.13% to a 3.98% and 5.63% for GTB and NUB subjects in misclassification errors, respectively.

References

[1] National Spinal Cord Injury Statistical Center, "Spinal Cord Injuries Facts & Figures at Glance," 2009, [Online], Available: https://www.nscisc.uab.edu/public_content/facts_figures_2009.aspx

[2] X. Huo, J. Wang, and M. Ghovanloo, "A magneto-inductive sensor based wireless tongue-computer interface," *IEEE Trans. on Neural Sys. Rehab. Eng.*, vol. 16, no. 5, pp. 497-504, Oct. 2008.

[3] R.G. Oliver, and S.P. Evans, "Tongue size, oral cavity size, and speech," *The Angle Orthodontist*, pp. 234-243, 1986.

[4] E.R. Kandel, J.H. Schwartz, and T.M. Jessell, *Principles of Neural Science*, 4th ed., New York: McGraw-Hill, 2000.

[5] E.B. Sadeghian, X. Huo, and M. Ghovanloo, "Command detection and classification in tongue drive assistive technology," *Proc. IEEE 33rd Eng. in Med. Biol. Conf.*, pp. 5465-5468, Sep. 2011.

[6] F. Lotte, M. Congedo, A. Lécuyer, F. Lamarche, and B. Arnaldi, "A review of classification algorithms for EEG-based brain-computer interfaces," *J. of Neural Eng.,* vol4, no. 4, pp. R1-R13, Jun. 2007.

[7] M. Mace, K. Abdullah-Al-Mamun, S. Wang, L. Gupta, and R. Vaidyanathan, "Ensemble classification for robust discrimination of multi-channel, multi-class tongue-movement ear pressure signals," *Proc. IEEE 33rd Eng. in Med. Biol. Conf.*, pp.1733-1736, Aug. 2011.

[8] B. Yousefi, X. Huo, E. Veledar, and M. Ghovanloo, "Quantitative and comparative assessment of learning in a tongue-operated computer input device," *IEEE Trans. Info. Tech. in Biomedicine*, vol. 15, no. 5, pp. 747-757, Sep. 2011.