An Online Adaptive Filtering Algorithm for the Vocal Joystick

[1]*Xiao Li,* [1]*Jonathan Malkin,* [2]*Susumu Harada,* [1]*Jeff Bilmes,* [3]*Richard Wright and* [2]*James Landay*

[1]Dept. of Electrical Engineering [2]Dept. of Computer Science [3]Dept. of Linguistics
University of Washington, Seattle

Abstract

This paper introduces a novel adaptive direction filtering algorithm in the Vocal Joystick (VJ) setting that utilizes context information and applies real-time inference in a continuous space. The VJ system using this algorithm is endowed with the ability to produce movements in arbitrary directions and the ability to draw smooth curves. This is in contrast to previous VJ settings whereby vowel quality was used to determine mouse movement in only a finite discrete set of directions [1].

Index Terms: computer interface, voice control, adaptive filtering

1. Introduction

As human-computer interaction becomes ubiquitous, the concept of continuous interaction [2] increasingly impacts the way we interact with everyday objects and appliances, and ultimately on the way we live in the modern world. Many emerging technologies, including vision-based and haptic interfaces, require new design approaches that allow for a continuous exchange of information between the user and the system at a high resolution, Moreover, many applications that utilize such interfaces, *e.g.* mouse and robotic arm control, additionally require real-time reasoning in a continuous space.

The Vocal Joystick (VJ), introduced in our earlier work [1], is a voice-based assistive technology originally designed for individuals with motor impairments. Unlike standard ASR, our system goes beyond the capabilities of sequences of discrete speech sounds, and exploits continuous vocal characteristics such as pitch, vowel quality, and loudness which are then mapped to continuous control parameters. We have utilized this interface to control computer mouse movement using the following scheme [1]: a mouse movement is triggered when vowel activity is detected, and continues until the vowel activity stops. At each time frame, the movement direction is determined by vowel quality, while the step size is determined by loudness. Finally, a small set of consonants are used to execute mouse clicks. Several video demonstrations of this VJ mouse system are available online at `http://ssli.ee.washington.edu/vj`.

One challenge faced by our earlier VJ systems is that a mouse movement was constrained to be in a number of (four or eight) principle directions. As shown in Figure 1, we categorized vowel qualities and mapped them to a discrete set of directions. Although we were using soft classification, which theoretically could produce movements in arbitrary directions if given "interpolated" vowels as input, in practice the chance of having such outputs is very small due to the nature of the classifier (as will be explained in the next section). Additionally, we have found that it can be difficult for a user to articulate interpolated vowels in a continuous and precise manner. Therefore, tasks like drawing a smooth curve was beyond the ability of the early VJ system.

This material is based on work supported by the National Science Foundation under grant IIS-0326382

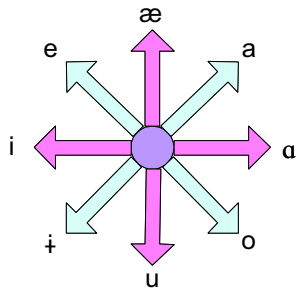


Figure 1: Mapping from vowels (in IPA symbols) to directions

This paper introduces an adaptive filtering algorithm in the Vocal Joystick setting that utilizes context information and applies real-time inference in a continuous space. The VJ system using this algorithm is endowed with the ability to produce movements in arbitrary directions and the ability to draw smooth curves. The rest of the paper is organized as follows: Section 2 introduces the background and formulates the problem; Section 3 describes and compares two early VJ systems. Section 4 presents in detail our proposed algorithm; and the last section discusses several qualitative tests and provides comments.

2. Problem Formulation

In the VJ mouse control system, we want to produce a 2-D motion vector $v_t = (\Delta x, \Delta y)^T$ (relative movement) given a vowel articulation o_t at each time frame. Here we only focus on mapping vowel quality to the angle of the motion vector. In other words, we assume $\Delta x^2 + \Delta y^2 = 1$. (How to map loudness to the norm of the motion vector has been investigated in [3].) There are two strategies for mapping the vowel space to a unit-motion-vector space. The first strategy is *regression*. For example, we can characterize the vowel space by the first and second formants (or their equivalent measures), and transform the formants to the X- and Y-coordinates of a unit motion vector. Though this strategy, theoretically, enables the mouse to move in arbitrary directions in a 2-D space, it is practically difficult to design the transformation as the perception of formants is often nonlinear. Also, it can be difficult for some users to intentionally and precisely control the movement direction by continuously, and sometimes independently, changing the first and second formants.

The second strategy, which we use in our system, is *soft classi-*

fication. We categorize the vowel space and map each category to a principle direction, as shown in Figure 1. Then, we have a classifier $p_t = g(o_t)$ that produces a posterior probability vector p_t given acoustics o_t as input. Finally a linear transform $v_t = f(p_t)$ maps the posterior probability vector p_t to a unit motion vector v_t. The goal of the Vocal Joystick mouse control can be formulated as follows: assuming a user's *intended* unit motion vector at time t is $\hat{v}_t$, we desire that

$$f(g(o_t)) = \hat{v}_t \tag{1}$$

Several stages in this process, however, can encounter errors or constraints, posing potential challenges in VJ control. The first possible error is due to human imprecision in articulation. As mentioned in the introduction, it is sometimes difficult for a user to precisely make the vowel that will produce her intended motion vector. An analogy is when a beginning violinist plays a note on a violin, it is quite likely to be out of tune. Second, the vowel classifier may not be accurate, leading to system errors in classification. The analogous scenario is that the violin itself may be out of tune. More importantly, there are inherent system constraints in the classification process. Since $g(\cdot)$ is usually a nonlinear transform, some values of $g(o_t)$ will be more likely to occur than others given that o_t is uniformly distributed. Consequently, the mouse will be more likely to move along certain directions. Taking the violin analogy again, imagine we replace the violin with a piano, we will then lose the ability to produce certain pitches and pitch variations because of the constraints imposed by the pitch-quantized equal-tempered keyboard.

Our design goals for the VJ system are that it should maximally reduce these errors and constraints by considering the following factors: (1) *producibility*, the system should use easily-producible vowels, reducing the effect of human imprecision; (2) *discriminability*, the system should use distinct vowels, reducing the chance of system errors; (3) *flexibility*, the system should provide enough degrees of freedom in direction control; (4) *predictability*, the system should work in a relatively intuitive way; and (5) *cognitive load*, the system should try to minimize the user's cognitive load. There certainly exist tradeoffs between these factors — for example, to increase flexibility, we may want to increase the number of vowel categories, as will be seen in the next section, but this may sacrifice producibility, discriminability, and cognitive load. The adaptive filtering algorithm we propose in Section 4 provides a way to balance these tradeoffs.

3. A Natural Control Strategy

A natural strategy associated with the soft classification scheme is to choose a number of vowel categories, *e.g.* four or eight, and map them to directions as in Figure 1. Specifically, we let $g(\cdot)$ be a two-layer MLP classifier, which ideally will output posterior probabilities of the categories if trained using the minimum relative entropy objective [4]. We also apply a supervised speaker adaptation algorithm for MLPs to improve classification accuracies [5, 6] [1]. Furthermore, to obtain the motion vector, $\mathbf{v}_t = (\Delta x, \Delta y)^T$, we define the linear transform function $f(\cdot)$ as follows:

$$\begin{aligned} \Delta x &= f_x(p_t) = \langle p_t, w_x \rangle \\ \Delta y &= f_y(p_t) = \langle p_t, w_y \rangle \end{aligned} \tag{2}$$

where w_x is $w_x^{(i)} = \cos\frac{2\pi i}{n}$, and $w_y^{(i)} = \sin\frac{2\pi i}{n}$; here n is the number of vowel categories, and the vowel categories are indexed counterclockwise with respect to Figure 1, starting from /ɑ/.

We first chose to use four vowel categories at the corners of the vowel triangle, namely /æ/, /ɑ/, /u/ and /i/, to maximize discriminability. As suggested in the previous section, a significant drawback of this system is its lack of flexibility. Due to the nature MLP classifiers, the posterior probability of one category is usually much higher than the others. This results in a system that witnesses mouse movements only along four cardinal directions. We therefore call it a "4-way" system.

To increase flexibility, we developed an "8-way" system using all eight vowel categories, namely /æ/, /a/, /ɑ/, /o/, /u/, /ɨ/, /i/ and /e/. The 8-way system relaxes the constraints imposed by the 4-way system to a great extent. For example, if we want to move the cursor along the up-right direction, in the 4-way system we might have to do it in a zig-zag pattern by saying "/æ/-/ɑ/" repetitively, while in the 8-way system we can simply say "/a/". The 8-way system, however, is obviously less advantageous compared with the 4-way system in terms of producibility and discriminability. In fact, we found that many users have trouble producing certain vowels, such as /a/ and /ɨ/. Even when a user can produce all eight vowels, it is sometimes hard to distinguish them since they are less separated in vowel space. Frame-level vowel classification experiments showed that the 8-way system has a classification error rate of around 8% while that of the first system is only 1%.

The next question is, can we combine the advantages of both systems? In other words, we desire a system that allows mouse movements in more directions but using only four explicit vowel categories. To this end, it is helpful to infer the user's intended motion vector by incorporating context information. For example, when the user says "/æ/-/ɑ/-/æ/-/ɑ/-..." in a target acquisition task, it is likely that he wants to move the mouse diagonally.

4. Online Adaptive Filtering

There has been research on plan recognition which aims to infer the plans of an intelligent agent from observations of the agent's actions [7]. The recent trend in approaching the plan recognition problem is to first construct a dynamic Bayesian network (DBN) for plan execution and then apply inference on this model [8, 9]. To model the plan hierarchy, [9] adopts a model of abstract Markov policies that enables an abstract policy to invoke more refined policies. In such techniques, however, user intent is usually modeled in a finite state space, and inference is often achieved via sampling. This poses problems to the situation where user intent is best expressed as a continuous variable.

We introduce an adaptive filter approach to infer intended values $\hat{p}_t$ from noisy estimates p_t, and then we replace p_t with $\hat{p}_t$ in Equation (2) in an attempt to obtain the intended motion vector $\hat{v}_t$. The model we use is essentially a hierarchical DBN. The idea is to predict the current $\hat{p}_t$ by a "plan" variable, a continuous version of the "abstract policy" used in [9], and then update $\hat{p}_t$ based on the current measurement. The system dynamics can be modeled in such a way that the standard Kalman filter algorithm [10] is directly applicable, so that $\hat{p}_t$ can be exactly inferred in the maximum likelihood sense. This dynamic model is "adaptive" in the sense that the plan variable is updated on the fly.

Specifically, as shown in Figure 2, we model the dynamics of the system using two variables in parallel. Variable $\hat{p}_t$ represents a user's intent, and variable g_t represents the plan, or the long-term trend of $\hat{p}_t$. In other words, $\hat{p}_t$ can be considered a local goal and g_t a global goal, both of which are not directly observable. We

[1] Compared with the results in [5, 6], the classification error rates have been significantly reduced in a recent development.

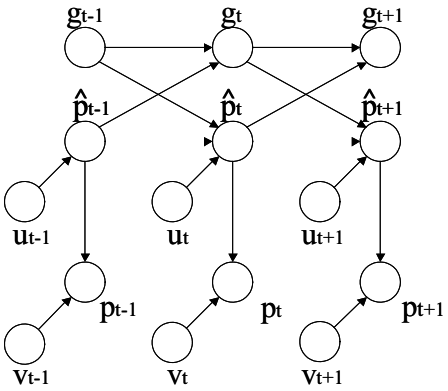


Figure 2: Bayesian network framework for user intent tracking

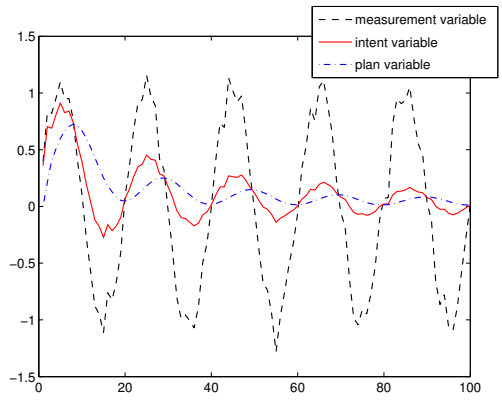


Figure 4: Adaptive filtering simulation

assume that $\hat{p}_t$ can be predicted by g_{t-1} with certain deviation, *i.e.* $\hat{p}_t = g_{t-1} + u_t$, where u_t is a multi-variate Gaussian with covariance matrix $Q(t) = \mathrm{E}[u_t u_t^T]$. This deviation can be caused by plan changes, human imprecision, system constraints (e.g. only allowing mouse movements in certain directions) or the user's intentional adjustments to compensate for previous errors. On the other hand, g_t can be determined by applying a low-pass filter on $\hat{p}_t$. The dynamics of $\hat{p}_t$ and g_t are thereby modeled linearly as follows:

$$\begin{bmatrix} g_{t+1} \\ \hat{p}_{t+1} \end{bmatrix} = \begin{bmatrix} a(t) & (1-a(t)) \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} g_t \\ \hat{p}_t \end{bmatrix} + \begin{bmatrix} 0 \\ u_t \end{bmatrix} \quad (3)$$

Furthermore, p_t is a measurement variable, representing the noisy estimate of $\hat{p}_t$. Specifically $p_t = \hat{p}_t + v_t$, where deviation v_t is due to system errors or environmental noise. For simplicity, we assume that v_t is generated from a multi-variate Gaussian distribution with covariance matrix $R(t) = \mathrm{E}[v_t v_t^T]$,

If we define $z_t = [g_t, \hat{p}_t]^T$ and $w_t = [0, u_t]^T$, we get the standard state-space model. The dynamic and observation models hence become

$$z_{t+1} = A(t) z_t + w_t \quad (4)$$

$$p_t = C z_t + v_t \quad (5)$$

where

$$A(t) = \begin{bmatrix} a(t)I & (1-a(t))I \\ I & 0 \end{bmatrix} \quad (6)$$

$$C = \begin{bmatrix} 0 & I \end{bmatrix} \quad (7)$$

with $a(t) \in [0, 1]$ being a scalar. In this way, $\hat{p}_t$ can be obtained in the maximum likelihood sense using the standard Kalman filter algorithm [10]. It is worth mentioning that we infer $\hat{p}_t$ instead of g_t because $\hat{p}_t$ does not have phase delay with respect to p_t whereas g_t does, as will be illustrated by a simulation in the next section.

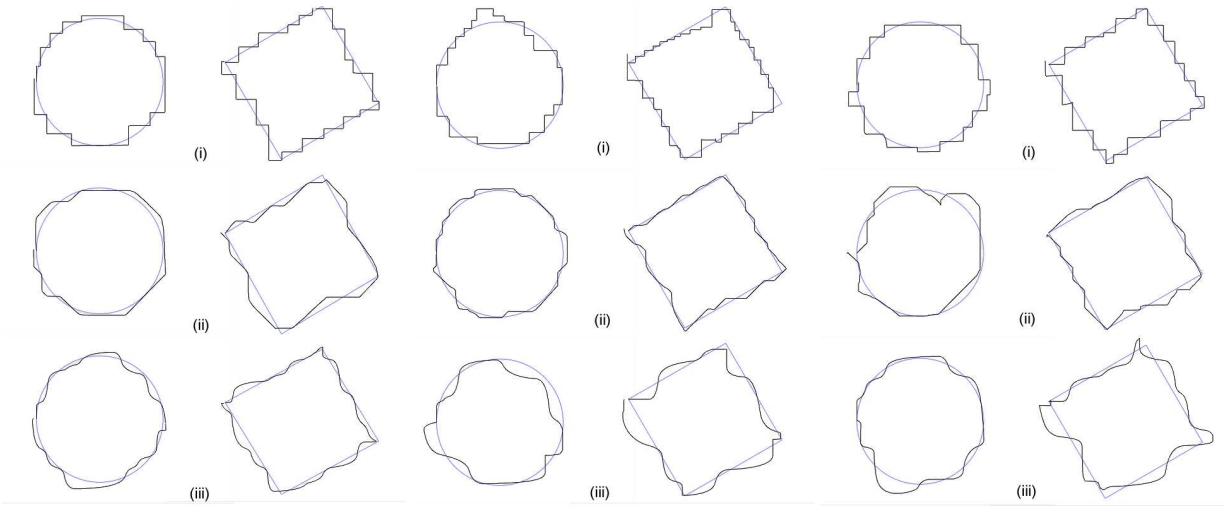
The choice of the model parameters is rather application-specific. First, matrix $A(t)$ decides how stable the plan variable g_t is. In the extreme case where $a(t)$ = 1, $g(t)$ becomes a constant. In target acquisition tasks, the plan for each utterance is to move the cursor along a certain fixed direction. In this case, we can use a function $a(t)$ monotonically increasing over time, *e.g.* $a(t) = 1 - c/(t + c)$, so that g_t will converge to a constant as time goes. Note that $A(t)$ is always reset to its initial value once a pause or a stop in articulation is detected. For steering tasks, however, the plan can change constantly. We thus let $a(t) = \alpha$ where $\alpha \in [0, 1]$ is empirically chosen to determine the smoothness of the plan.

The covariance matrix $Q(t)$ adjusts the tradeoff between the smoothness of the estimate trajectory and the loyalty to the measurement trajectory. If the variance of u_t is small, $\hat{p}_t$ will converge to its long-term trend g_t and the trajectory of the estimates becomes smooth; otherwise $\hat{p}_t$ will be more loyal to the measurements p_t. This parameter also adjusts the tradeoff between the system's automation degree and the system's predictability to humans. Increased automation management is not always desirable since it yields increased unpredictability to humans, which explains why "even perfect intent inference might not be good enough" [11]. Finally, the covariance matrix $R(t)$ depends on the classifier confusion matrix and the environmental noise condition, both of which can be estimated using principled approaches.

5. Experiments and Discussions

To illustrate the behavior of the adaptive filter, we ran a simulation for a uni-variate random process. The measurement $p_t = sin\frac{\pi}{10}t + v_t$, where $v_t \sim \mathcal{N}(0, 0.01)$. The black trajectory in Figure 4 represents this noisy sinusoid function for $t = 1 : 100$. Assume that the values +1 and -1 represent two principle directions, and that the oscillating pattern implies the user's effort to find a direction in between, represented by the value 0. We hope that our model can aid the user to approach and stabilize in this desired direction. Here we let $a(t) = 1 - 1/t$, $\mathrm{E}[u_t u_t^T] = 0.1/t$ and $\mathrm{E}[u_t u_t^T] = 0.01$. The estimated plan variable g_t is depicted as the blue trajectory, and $\hat{p}_t$ is depicted as the red trajectory in Figure 4. The $\hat{p}_t$ variable (the red), which is inferred by our algorithm, is loyal to the p_t (the black) at the beginning and approaches g_t (the blue) as time goes. This plot also illustrates that $\hat{p}_t$ is synchronized with p_t, while g_t is not.

As a test using real-world applications, the authors used the 4-way system, the 8-way system, and the 4-way system enhanced by adaptive filtering to browse a website. The VJ engine uses a two-layer MLP classifier with 50 hidden units and 7 frames of MFCC features as input. As can be seen in the video demonstration accompanying this paper `http://ssli.ee.washington.edu/vj/icslp-submission/icslp.mov`, the adaptive filter manifested more control flexibility while using



(a) User I (b) User II (c) User III

Figure 3: Steering Snapshots: (i) 4-way system; (ii) 8-way system; (iii) 4-way system with adaptive filtering

only four vowels. Using this system, the user achieved fairly stable movements along directions other than the four cardinal ones by oscillating between two vowels. The video also shows the case where the cursor path became a smooth curve when the user transitioned from one vowel to another.

The curve-drawing capability of the adaptive filter is more pronounced in a steering task. This involves using the VJ mouse to steer along two different shapes, a circle and a square tilted by an angle, shown as the blue paths in Figure 5. The circle had a radius of 300 pixels on a 800×600 screen, and the square was rotated 30 degrees counterclockwise with each side approximately 532 pixels. The cursor always started at the leftmost part of each shape, and its movement was constrained to be within a "tube", with a radius of 30 pixels, centered at the reference shape. The session would fail once the cursor hit the wall of that tube. The users (again the authors), though having experience with the early VJ system, were relatively novice users of the adaptive filter. As shown in the Figure 5, the 8-way system and the system with adaptive filtering produced much smoother paths compared with the 4-way system, but the adaptive filter approach achieved this by using only four vowels. Furthermore, the task completion times were very similar across all three systems.

We found in the steering tasks, however, that the adaptive filter enhanced flexibility at the cost of predictability. In other words, the way the system works is not as intuitive as those using the natural control strategy; the smoothness of the curve is sometimes hard to control. However, we believe the predictability can be significantly increased if given more time to learn this system — analogous once again to learning to play a violin, individuals with motor impairments, moreover, are often quite motivated to learn novel user interface technologies. Given the experience we had using all three systems, we are encouraged by the prospect of beginning a large-scale user study to thoroughly evaluate user preferences and learnability of these control strategies. In addition, we will consider using a different vocal parameter, such as pitch, to determine the smoothness parameters. The authors would like to thank Kelley Kilanski for her work on the VJ vowel database.

6. References

[1] J. Bilmes and et.al., "The Vocal Joystick," in *(to appear) ICASSP*, May 2006.

[2] G. Faconti and M. Massink, "Continuity in human computer interaction," Tech. Rep., CHI Workshop, 2000.

[3] J. Malkin, X. Li, and J. Bilmes, "Energy and loudness for speed control in the Vocal Joystick," in *ASRU Workshop*, Nov 2005.

[4] C. Bishop, *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford, 1995.

[5] X. Li, J. Bilmes, and J. Malkin, "Maximum margin learning and adaptation of MLP classifers," in *Eurospeech'05*, September 2005.

[6] X. Li and J. Bilmes, "Regularized adaptation of discriminative classifiers," in *(to appear) ICASSP*, May 2006.

[7] H. Kautz and J. F. Allen, "Generalized plan recognition," in *AAAI*, 1986, pp. 32–38.

[8] D. Pynadath and M. Wellman, "Accounting for context in plan recognition, with application to traffic monitoring," in *Proc. Conf. on Uncertainty in Artificial Intelligence*, 1995.

[9] H. H. Bui, S. Venkatesh S, and G. West, "The recognition of abstract Markov policies," in *AAAI*, 2000, pp. 524–530.

[10] M. S. Grewal and A. P. Andrews, *Kalman Filtering: Theory and Practice*, Prentice Hall, 1993.

[11] H. B. Funk and C. A. Miller, "User acceptance and plan recognition: Why even perfect intent inferencing might not be good enough," in *AAAI Fall Symposium*, 2001.