ROBUST VOLUME CONTROL PERSONALISATION FROM ON-LINE PREFERENCE FEEDBACK

Alexander Ypma[1], *Bert de Vries*[1,2] *and Job Geurts*[1,2]

[1]GN ReSound Research, GN ReSound A/S, Horsten 1, 5612 AX Eindhoven
[2]Signal Processing Systems group, Dept. Electrical Engineering, TU Eindhoven
{aypma,bdevries}@gnresound.com

ABSTRACT

We describe a Learning Volume Control (LVC) algorithm that learns the volume control operations of a hearing aid user in such a way that the internal settings of the volume control will over time absorb the user's preferences. For practical use, the algorithm should be robust against inconsistent behaviour of the user. We propose two algorithms for LVC (based on LMS learning and Kalman filtering) and demonstrate desirable properties of our LVC in simulations. Further, we provide evidence for the practical relevance of our algorithms in a listening test.

1. INTRODUCTION

Many electronic personal devices contain algorithm parameters that are preset to values that aim at optimally matching the preferences and behaviour of its user. To a certain extent, this can indeed be done in a fitting session, e.g. at a hearing aid dispenser. However, not every individual user preference can be put into the device in this manner: some particularities of the user may be hard to code/represent into the algorithm, his typical sound environments may be mismatched or changing, and also his preference patterns may be changing. Therefore one would like to *personalise* the instrument to the preferences of its user in an on-line fashion, i.e. during usage in-the-field. The main idea of this paper is to absorb user adjustments to the volume control of a hearing aid in the parameters of the volume control algorithm. Ultimately, this strategy should lead to fewer user manipulations. Clearly, the learning algorithm should be robust to inconsistent user behaviour.

Here we extend the work in [1] by introducing an 'enhanced model' for our Kalman filter LVC, exploiting additional assumptions on the user. Further, we show that our approach has practical relevance by experiments with a real-time implementation of the algorithms, one of which is a listening test with eight normal hearing subjects.

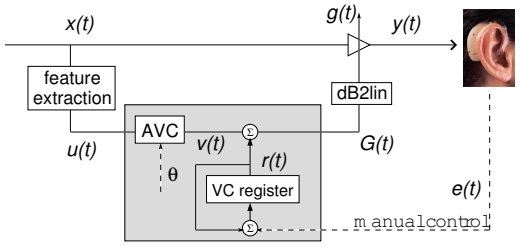

Fig. 1. *Learning Volume Control (LVC) flow diagram*

2. SYSTEM DESCRIPTION

The situation is illustrated in Fig. 1. The amplitude of a signal x_t is adjusted by a gain g_t to yield $y_t = g_t x_t$. In a typical application, the gain g_t is controlled by an *automatic volume control* (AVC) module. The AVC unit takes as input u_t, which holds a vector of relevant features w.r.t. the desired gain for signal x_t. For instance, u_t could hold short-term RMS and SNR estimates of x_t. In a linear AVC, the desired (log-domain) gain G_t is a linear function (with saturation) of the input features, i.e.

$$G_t = u_t^T \theta_t + r_t \tag{1}$$

where the offset r_t is read from a volume-control (VC) register[1]. Sometimes, during operation of the device, the user is not satisfied with the volume of the received signal y_t. He is provided with the opportunity to manipulate the gain of the received signal by changing the contents of the VC register through turning a volume control wheel. We will let e_t represent the accumulated change in the VC register from $t-1$ to t as a result of user manipulation. Our learning goal is to absorb the regular patterns in the VC register into our AVC model parameters θ. We will use an additive learning process,

$$\theta_{t+1} = \theta_t + \mathring{\theta}_t \tag{2}$$

[1]We work in discrete time steps, so subscript $t-1$ actually refers to $t - T_s$, where T_s is the sampling period.

Parameter update factor $\mathring{\theta}_t$ is determined by specific choices for learning algorithms, such as LMS or Kalman filtering.

2.1. When to Learn?

We should only execute a parameter update when we actually learn something about the user's preferences. While the VC wheel is not being manipulated during normal operation of the device, we may assume that the user is content with the delivered volume, but we cannot be sure. After all, the user may not be wearing the device. However, when the user starts turning the VC wheel, we will accept that he is not content at that moment. We identify the beginning of a VC manipulation phase with the term *dissent moment*. While the user manipulates the VC wheel, he is likely still searching for a better gain. A next learning moment occurs right after the user has stopped changing the VC wheel position. At this time, we may assume that he has found a satisfying gain; we'll call this the *consent moment*. Dissent and consent moments identify situations for collecting negative and positive teaching data, respectively. In this paper, we will only learn from consent moments, i.e. we will only use positive data. Assume that the kth consent moment is detected at $t = t_k$. Since our updates only take place at times t_k, it is useful to define a new time series (with slight abuse of the subscript notation) as

$$G_k = \sum_t G_t \delta(t - t_k)$$

and similar definitions for converting r_t to r_k etc. The new sequence, indexed by k rather than t, only selects samples at consent moments from the original time series. Note that by considering only instances of explicit consent, there is no need for an internal clock in the system. In order to complete the algorithm, we still need to specify the update factor $\mathring{\theta}_k$. Next, we present two update algorithms.

3. LEARNING BY THE NLMS ALGORITHM

In our nLMS algorithm for Learning Volume Control (LVC), the learning update Eq. (2) should not affect the actual gain G_t and hence we need to compensate by subtracting an amount $u_t^T \mathring{\theta}_t$ from the VC register. The VC register contents are thus described by

$$r_{t+1} = r_t - u_t^T \mathring{\theta}_t + e_{t+1}. \tag{3}$$

Note that r_t always (at all times t) holds a value, but that only at consent times $t = t_k$ discount $u_t^T \mathring{\theta}_t$ is applied. The correction e_k at a consent time t_k is equal to the accumulated corrections $\sum_{t_{k-1}+1}^{t_k} e_t$. We will assume that $u_t^T \theta_t = \left[1, u_t^1, \ldots, u_t^m\right] \left[\theta_t^0, \theta_t^1, \ldots, \theta_t^m\right]^T$, where the superscript i refers to the $i + 1^{st}$ component of the vectors u_t and θ_t. In other words, we provide a facility in θ_t^0 to absorb the preferred mean VC offset. It is now reasonable to assume a cost criterion $E[r_k^2]$ which we try to minimise w.r.t. θ (and $E[\cdot]$ denotes expectation). A normalised LMS-based learning volume control is effectively implemented using the following update equation, [2]

$$\mathring{\theta}_k = \mu_k {u_k}^T r_k = \frac{\mu}{\sigma_k^2 + u_k^T u_k} {u_k}^T r_k \tag{4}$$

where μ and μ_k are initial and estimated learning rates, respectively, and σ_k^2 is an estimate of $E[r_k^2]$. In practice, it is helpful to select a separate learning rate for adaptation of the offset parameter θ_0. We propose to track $E[r_k^2]$ by a leaky integrator,

$$\sigma_k^2 = \sigma_{k-1}^2 + \gamma \times [r_k^2 - \sigma_{k-1}^2] \tag{5}$$

where γ sets the effective window of the integrator. Note that the LMS-based updating implicitly assumes that 'adjustment errors' are Gaussian distributed.

The variable σ_k^2 essentially tracks the user inconsistency. As a consequence, for enduring large values of r_k^2, the parameter update factor will be small, which means that we do not absorb the user's preferences. This is a desired feature of our LVC system. It is possible to replace σ_k^2 in Eq. (4) by alternative measures of user inconsistency. Alternatively, in the next section we introduce the Kalman filter which also improves our dealings with inconsistent user responses.

4. LEARNING WITH A KALMAN FILTER

When a user changes his preferences, he will probably induce noisy corrections to the volume wheel. In the nLMS algorithm, these increased corrections would contribute to the estimated variance σ_k^2, hence lead to a decrease in the estimated learning rate. However, the noise in the correction could also be attributed to a transition to a new 'parameter state'. One would like to *increase* the learning rate with the estimated state noise variance in order to respond quickly to a changed preference pattern.

4.1. An inconsistent user with changing preferences

We assume a user with a preferred gain given by $G_t = u_t^T a_t^{\mathrm{d}}, \forall t$. The 'user preference vector' a_t^{d} may be nonstationary (hence the subscript t) and is supposed to generalise to different auditory scenes. This requires that feature vector u_t contains relevant features that describe the acoustic input well. The user will express his preference for this sound level by adjusting the volume wheel, i.e. by feeding back a correction factor that is ideally noiseless (e_k^{d}) and adding it to the register r_k. In reality, the actual user correction e_k will be noisy, $r_{k+1} = r_k + e_{k+1} = r_k + e_{k+1}^{\mathrm{d}} +$

ε_{k+1}. Here, ε_{k+1} is the accumulated noise from the previous consent moment to the current, and it is supposed to be Gaussian distributed. We assume that the user experiences an 'annoyance threshold' $\bar{e}$ such that $|e_t^{\mathrm{d}}| \leq \bar{e} \rightarrow e_t = 0$. In other words, only if the intended correction exceeds the annoyance threshold, the user will be in explicit dissent and will issue a (noisy) correction.

4.2. State space formulation

Allowing the parameter vector that is to be estimated to 'drift' with some (state) noise, leads to the following state space formulation of the LVC:

$$\left\{ \begin{array}{rcll} \theta_{k+1} & = & \theta_k + \nu_k & , \quad \nu_k \sim \mathcal{N}(0, \delta^2 I) \\ G_k & = & u_k^T \theta_k + r_k & , \quad r_k \sim \text{nongaussian} \end{array} \right.$$

Besides the gain model (cf. Eq. (1)) we now also have a model for the parameter drift. We can estimate the posterior of θ_k recursively using the corresponding Kalman filter update equations. The resulting LVC algorithm is referred to as *simplified* Kalman filter LVC. It is instructive to compare the estimated learning rates in the nLMS algorithm and the simplified Kalman filter. Both give rise (see [3]) to an effective update rule

$$\hat{\theta}_{k+1} = \hat{\theta}_k + \mathring{\theta}_k = \hat{\theta}_k + \mu_k {u_k}^T r_k \tag{6}$$

for the mean $\hat{\theta}_k$ of the parameter vector (and additionally, the Kalman filter also updates its variance Σ_k). The difference between the algoritms is in the μ_k term, which in the Kalman LVC is

$$\mu_k = \Sigma_{k|k-1}(u_k \Sigma_{k|k-1} {u_k}^T + \sigma_k^2)^{-1} \tag{7}$$

where μ_k is now a learning rate *matrix*. For the Kalman algorithm, the learning rate is dependent on the state noise ν_k, through the predicted covariance of state variable θ_k, $\Sigma_{k|k-1} = \Sigma_{k-1} + \delta^2 I$. The state noise can become high when a transition to a new dynamic regime is experienced. Furthermore, it scales inversely with observation noise σ_k^2, i.e. the uncertainty in the user response. The more consistent the user operates the volume control, the smaller the estimated observation noise, the larger the learning rate. The nLMS learning rate only scales (inversely) with the user uncertainty. On-line estimates of the noise variances δ^2, σ^2 can be made with the Jazwinski method (see again [3]). Further, note that the observation noise is nongaussian in both nLMS and the state space formulation of the LVC. Especially the latter, which is solved with a recursive (Kalman filter) algorithm is sensitive to model mismatch. This can be solved by making an explicit distinction between the 'structural part' e_k^{d} in the correction and the actual noisy adjustment $e_k = e_k^{\mathrm{d}} + \varepsilon_k$ (see next section).

4.3. Additional user assumptions

We take the approach that a user correction can be fully absorbed by the AVC in one update instant, provided that it represents the underlying desired correction (and not the noisy version that is actually issued). We model the *desired correction factor* by $e_k^{\mathrm{d}} = u_k^T \lambda_k$ and incorporate this in θ_k in one update instant. The idea behind this model is that the user deduces from the temporal structure in the past values $v_{t-M} ... v_t$ the mismatch between his desired overall gain vector a^{d} and the currently realised gain vector θ_t, even though he does not know[2] the instantaneous value of the u_t (but only experiences the current $v_t = u_t^T \theta_t$, see figure 1). In this case, his desired correction at the next update would then be the result of an implicit comparison of a^{d} with θ_t, or $e_{k+1}^{\mathrm{d}} = u_{k+1}^T \lambda_{k+1} = u_{k+1}^T (a^{\mathrm{d}} - \theta_k)$. In this model there is no need for a register with memory, since the instantaneous correction is fully absorbed on the next instant. We therefore find the following register value:

$$r_k = e_k = u_k^T \lambda_k + \varepsilon_k, \;\; \text{if } |\lambda_k| \geq \bar{\lambda} \tag{8}$$

where $\varepsilon_k \sim \mathcal{N}(0, \sigma^2)$ and where we now assume an 'annoyance threshold' (vector) $\bar{\lambda}$ on λ_t rather than e_t. We write the gain inference problem as an 'enhanced state space model':

$$\left\{ \begin{array}{rcll} \theta_k & = & \theta_{k-1} + \lambda_{k-1} + \nu_k & , \quad \nu_k \sim \mathcal{N}(0, \delta^2 I) \\ \lambda_k & = & a^{\mathrm{d}} - \theta_{k-1} + w_k & , \quad w_k \sim \mathcal{N}(0, \delta^2 I) \\ G_k & = & u_k^T \theta_k + \varepsilon_k & , \quad \varepsilon_k \sim \mathcal{N}(0, \sigma^2) \end{array} \right. \tag{9}$$

where $\delta^2 I$ is the covariance matrix of state noises ν_k, w_k and observation noise ε_k represents the user inconsistency. Note that the 'discount formula' for e_k in Eq. (3) now shows up in the form $\lambda_k = a^{\mathrm{d}} - \theta_{k-1}$, since incorporation of previous corrections in θ will diminish future λ_k. We introduce an auxiliary state variable a_k to represent the unknown value of a^{d}. The linear dynamical system (LDS) formulation of Eq. (9) can be rephrased as

$$\left\{ \begin{array}{rcl} \begin{bmatrix} \theta_k \\ \lambda_k \\ a_k \end{bmatrix} & = & \begin{bmatrix} I & I & \mathbf{0} \\ -I & \mathbf{0} & I \\ \mathbf{0} & \mathbf{0} & I \end{bmatrix} \begin{bmatrix} \theta_{k-1} \\ \lambda_{k-1} \\ a_{k-1} \end{bmatrix} + \xi_k \\ G_k & = & [u_k^T \, \vec{0} \, \vec{0}] \begin{bmatrix} \theta_k \\ \lambda_k \\ a_k \end{bmatrix} + \varepsilon_k \end{array} \right. \tag{10}$$

where $\xi_k \sim \mathcal{N}(0, \delta^2 I)$ represents the combined state noise and $\mathbf{0}, \vec{0}$ are a matrix and a vector of zeros of appropriate dimension, respectively. Relabeling state vector and coefficients as F_k, H_k and x_k, we recognise the familiar form for a time-varying LDS:

[2] The user will perceive some aspects of the sound features, though.

$$\begin{cases} x_k &= H_k x_{k-1} + \xi_k \quad , \quad \xi_k \sim \mathcal{N}(0, \delta^2 I) \\ G_k &= F_k x_k + \varepsilon_k \quad , \quad \varepsilon_k \sim \mathcal{N}(0, \sigma^2) \end{cases} \tag{11}$$

The Kalman filter update equations for this model are [4]:

$$\begin{aligned} \hat{x}_{k|k-1} &= H_k \hat{x}_{k-1} \\ \Sigma_{k|k-1} &= H_k \Sigma_{k-1} {H_k}^T + \delta^2 I \\ K_k &= \Sigma_{k|k-1} {F_k}^T (F_k \Sigma_{k|k-1} {F_k}^T + \sigma^2)^{-1} \\ \hat{x}_k &= \hat{x}_{k|k-1} + K_k (G_k - F_k \hat{x}_{k|k-1}) \\ \Sigma_k &= (I - K_k F_k) \Sigma_{k|k-1} \end{aligned}$$

The update formula for $\hat{x}_k$ implies e.g. the update:

$$\hat{\theta}_k = \hat{\theta}_{k-1} + \hat{\lambda}_{k-1} + {K_k}^{(1)} \varepsilon_k \tag{12}$$

where ${K_k}^{(i)}$ is the i-th component (row) of K_k and $\varepsilon_k = G_k - \tilde{G}_k = G_k - F_k H_k \hat{x}_{k-1}$.

5. EXPERIMENTS

5.1. Evaluation of Kalman filter LVC

We performed a Matlab simulation of the Kalman filter LVC to study its behaviour with inconsistent users with changing preferences. As input we used a music excerpt that was preprocessed to give one-dimensional log-RMS feature vectors. This was fed to a simulated user who had a preference vector a_t^{d} and noisy corrections based on the model of section 4.3 were fed back to the LVC.

First, we assumed a user with a fixed preferred a^{d} of three (not shown in the figures). We also assumed that the user was always in 'explicit dissent' mode, implying $\bar{\lambda} = 0$. We learn continuously from explicit consent, i.e. each correction was used for updating. The user inconsistency changed throughout the simulation (see figure 2, middle graph), where higher values of the inconsistency in a certain time segment denote more 'adjustment noise' in turning the virtual volume control. In figure 2, bottom 'alpha(t)' graph, one can observe the roughly inverse scaling behaviour of implied learning rate μ_k (sometimes referred to in the figures as α_t) with user inconsistency, which is the desired robust behaviour.

We studied the behaviour with a user who now has changing amplification preferences[3] and who experiences a threshold on his annoyance before he we will do the adjustment, i.e. $\bar{\lambda} > 0$. When adjustments are absent (i.e. when the AVC value comes close to the desired amplification level value a^{d}), the noise is also absent (see figures 3 and 4, bottom 'user-applied (noisy) volume control actions' graphs).

[3]See figure 2, the 'User mode: amplification' graph shows that the a_t^d values are 3,-2,0,1 in the four modes. Since the feature values u_t are negative, a *negative* value of a_t^d in fact leads to an effective amplification, and vice versa for positive a_t^d.

The results indicate a better tracking of user preference and much smaller sensitivity to user inconsistencies when the Kalman-filter LVC is used compared to 'no learning'. This can be seen e.g. by comparing the top rows of figures 3 (without learning) and 4 (LVC): the LVC output signal y_t (in log-RMS values) is much smoother than the 'no learning' output, indicating less sensitivity to user inconsistency. Furthermore, we can see in the bottom row of figure 4 that *using the LVC results in less adjustments made by the user*, another desirable feature of the LVC algorithm.

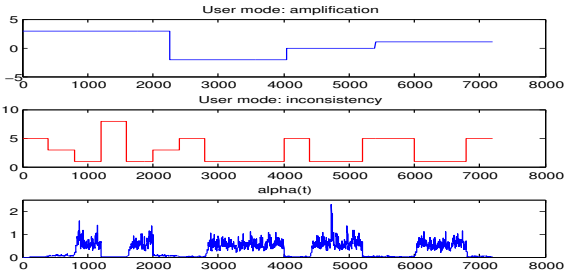

Fig. 2. Top: User amplification preference, Middle: user inconsistency, Bottom: inferred learning rate.

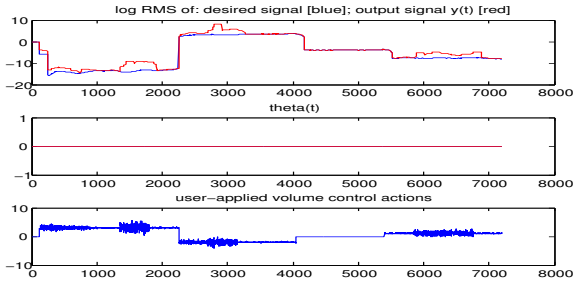

Fig. 3. Without learning. Top: realised output signal y_t and desired output signal (both in log-RMS). Middle: θ_k. Bottom: volume adjustments applied by the virtual user.

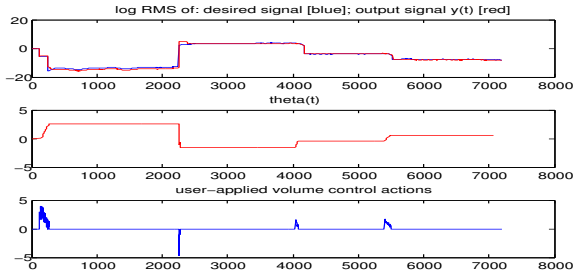

Fig. 4. Learning Volume Control. Graphs as in figure 3.

5.2. Real-time simulation

We implemented our LVC algorithms on a real-time platform, where subjects are allowed to interact with the algo-

rithm in real-time, in order to study the behaviour of the algorithms and the user. We started with a simulated user, i.e. the adjustment sequence was predetermined and we studied the behaviour of the algorithms.

5.2.1. nLMS

In the top graphs of figure 5 we plot the predetermined sequence of noisy user corrections (i.e. $\{e_k\}$). In the left graphs the results with a slowly responding LVC are shown, and we can see that the estimated learning rate ("mu") scales roughly inversely with the noisy adjustments (sample number running from approx. 0.5E4 to 3E4). However, the two 'informative' adjustments around sample number 3E4 to 3.5E4 are considered noise, and lead to a sudden decrease of the learning rate, which is undesirable. This effect is also present in a fast responding LVC (right graphs), although the 'recovery' of this undesirable drop is faster. The algorithm's response to the noisy adjustment episodes is also quite noisy (fast changes in learning rate due to noisy actions). Note that nLMS may easily 'see' a short sequence of informative adjustments as noise, increasing the estimate of σ_k and decreasing the learning rate, which is undesirable.

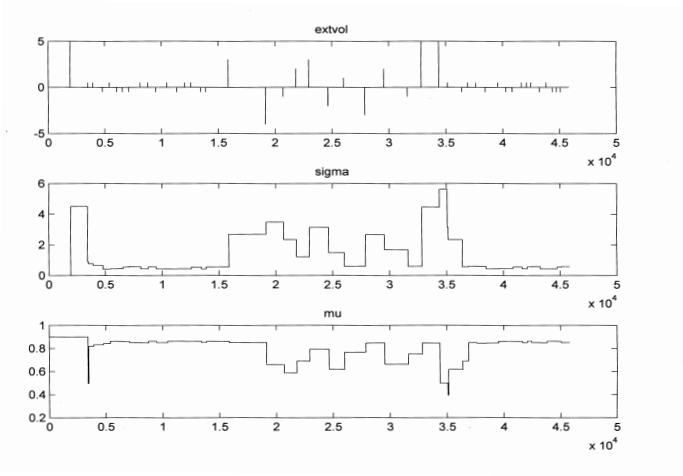

Fig. 5. nLMS learning volume control. Left graphs: slow response ($\mu = 0.9, \gamma = 0.1$). Right graphs: fast response ($\mu = 0.9, \gamma = 0.9$). Top: noisy correction sequence $\{e_k\}$, Middle: inconsistency σ_k , Bottom: learning rate μ_k.

5.2.2. Kalman filter

In figures 6 and 7, the behaviour of the enhanced and the simplified Kalman filter LVC are compared in a setting with *relative* volume control usage, i.e. with adjustment sequences $\{\mathrm{extvol}_k\} = \{e_k\}$. We notice that the enhanced Kalman filter LVC estimated the noise in the adjustments rather nicely (in the observation noise variable σ_k). With the simplified Kalman LVC, we now observe the desired behaviour with the adjustment sequence that was used earlier in the nLMS experiments. Although the observation noise seems to be 'pulled up' along with the state noise (which could be a result of our suboptimal estimation of state noise and observation noise), the learning rate *alpha* is high at the two transition points (informative adjustments around 0.25E4 and 3E4) and mainly low at the noisy adjustments. The relatively high learning rate at the end of the sequence appears an artifact of the overestimation of the observation noise. A better way to estimate state and observation noise (e.g. with recursive EM) may overcome this.

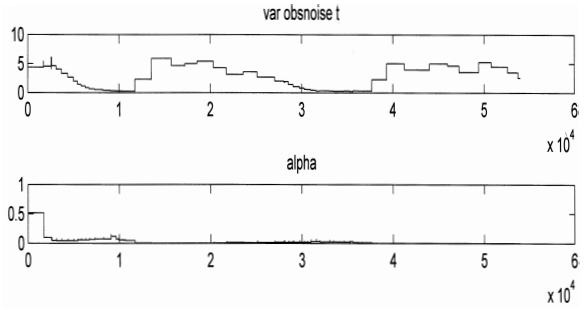

Fig. 6. Enhanced Kalman filter LVC. Top: user inconsistency σ_k^2. Bottom: learning rate μ_k (denoted with *alpha*).

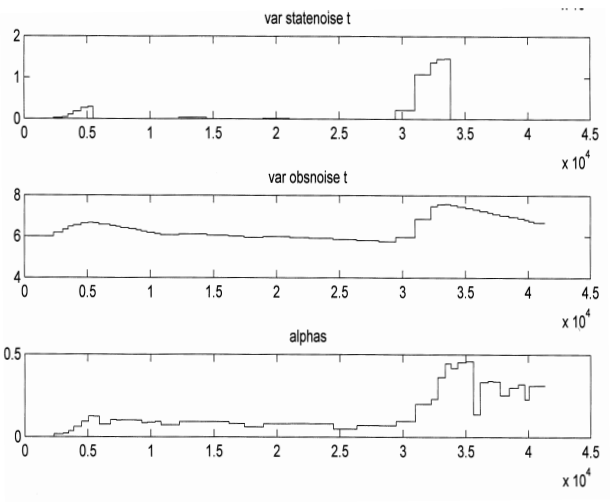

Fig. 7. Simplified Kalman filter LVC. Top: estimated state noise variance δ_k^2. Middle: estimated user inconsistency σ_k^2. Bottom: estimated learning rate μ_k (denoted *alpha*).

5.3. Evaluation with a listening test

We studied the user's volume control behaviour by setting up a listening test. We selected the simplified Kalman LVC, that was implemented on the real-time platform and used two acoustic features and a bias term. Then several speech

and noise snapshots were picked from a database (typically in the order of 10 seconds) and these were combined in several ratios and appended. This led to 4 streams of signal/noise episodes with different types of signal and noise in different ratios. Eight normal hearing volunteers were asked to listen to these four streams twice in a row, adjusting the volume when desired (referred to as one experiment with two runs). Two volunteers were assigned to the no-learning situation, three were assigned to the learning situation and three were assigned to both. The volunteers were not told whether learning took place in their experiment or not. In the no-learning case, the algorithmic behaviour in the first run of four streams and the second run of four streams are identical (i.e. no learning takes place, so the settings of the automatic volume control remain at their initial values). In the learning case, user corrections are incorporated in the internal volume control throughout the experiment.

5.3.1. Results

We found that in 9 out of 11 experiments the total number of adjustments in the second run of four streams decreased compared to the first run. This can probably be explained by a certain 'getting used to' or accommodation effect (perhaps a 'tiredness of adjusting the volume'). This effect typically gives rise to a reduction to around 80 % adjustments[4]. This figure was obtained by averaging the second-run percentages of the five control experiments. In the six *learning experiments*, an average second-run percentage around 80 % was found as well, but we also found a large variance in the 'turning behaviour' (two out of six had second-run percentages larger than 100, three out of six had second-run percentages around 50). However, when only considering the three subjects who experienced *both* LVC and no-learning, the *total number of adjustments in both runs of an experiment appeared to decrease when the LVC was present*. When the number of adjustments in an experiment for no-learning is set to 100 %, LVC led to some 80 % adjustments, on average. Four out of six 'learning subjects' reported 'a pleasant effect of the LVC'. One of these preferred the LVC run since "no noticeable deteriorations were present, and some of the sharp and annoying transitions were smoothed out".

6. DISCUSSION

We described a new approach to learning volume control of hearing instruments. We sketched two algorithms for implementing this approach, one of which was given in two flavours (simplified or enhanced) depending on the assumption about the user behaviour. Making the right assumptions about hearing aid usage behaviour is clearly a topic of further research; we made an initial effort in this paper. We also mention that the results of the listening test should be interpreted with some caution, since we did not use a randomised experimental design and the number of subjects was small. The *subjective* evaluation of LVC in this blind experiment pointed in the right direction (volunteers subjected to LVC liked it, without knowing that it was LVC). Further, we think that our approach to learning volume control can be adapted to other protocols and interfaces. Our aim is to embed the problem in the general framework for optimal Bayesian incremental fitting that is described in [5].

[4]The percentages refer to the number of adjustments in the second run as a percentage of the number of adjustments in the first run.

7. ACKNOWLEDGMENT

The authors like to thank Rob de Vries for useful comments on an initial version of this paper.

8. CONCLUSION

We proposed a Learning Volume Control that is robust to user inconsistency. In experiments with simulated users, the Kalman filter LVC shows improved tracking of changing user preferences, less sensitivity to user inconsistency and less adjustments to be made by the user. Furthermore, from a listening experiment we gained evidence that the LVC approach is beneficial to the user.

9. REFERENCES

[1] A. Ypma, B. de Vries, and J. Geurts, "A learning volume control that is robust to user inconsistency," in *Proc. 2nd Annual IEEE Benelux/ DSP Valley Signal Processing Symposium*, March 2006, pp. 103–106.

[2] R. S. Sutton, "Gain adaptation beats least squares?," in *Proc. 7th Yale Workshop on adaptive and learning systems*, 1992, pp. 161–166.

[3] W. D. Penny, "Signal processing course," Tech. Rep., University College London, 2000.

[4] T. Minka, "From hidden markov models to linear dynamical systems," Tech. Rep. 531, Dept. of Electrical Engineering and Computer Science, MIT, 1999.

[5] T. Heskes and B. de Vries, "Incremental utility elicitation for adaptive personalization," in *Proc. 17th Belgium-Netherlands Conference on Artificial Intelligence*, October 17-18, 2005, pp. 127–134.