ABSTRACT
In this work, we propose a time-recursive multi-pitch estimation algorithm, using a sparse reconstruction framework, assuming only a few pitches from a large set of candidates to be active at each time instant. The proposed algorithm utilizes a sparse recursive least squares formulation augmented by an adaptive penalty term specifically designed to enforce a pitch structure on the solution. When evaluated on a set of ten music pieces, the proposed method is shown to outperform state-of-the-art multi-pitch estimators in either accuracy or computational speed.

Index Terms—sparse recursive least squares, group sparsity, multi-pitch estimation, dictionary learning

1. INTRODUCTION
The problem of estimating the fundamental frequency, or pitch, arises in a variety of fields, such as in speech and audio processing, non-destructive testing, and biomedical modeling (see, e.g. [1, 2] and the references therein). In such applications, the measured signal may often result from several partly simultaneous sources, meaning that both the number of pitches, and the number of overtones of each such pitch, may be expected to vary over the signal. Such would be the case, for instance, in most forms of audio signals. The resulting multi-pitch estimation is in general difficult, with one of the most notorious issues being the so-called sub-octave problem, i.e., distinguishing between pitches whose fundamental frequencies are related by powers of two. Both non-parametric, such as methods based on autocorrelation (see, e.g. [3] and references therein), and parametric multi-pitch estimators (see, e.g. [2]) have been suggested, where the latter is often more robust to the sub-octave problem, but rely heavily on accurate a priori model order information of both the number of pitches present and the number of harmonic overtones for each pitch. Regrettably, the need for accurate model order information is a significant drawback, as such information is typically difficult to obtain. In order to alleviate this, several sparse reconstruction algorithms tailored for multi-pitch estimation have recently been proposed, allowing for estimators that do not require explicit knowledge of the number of sources or their harmonics (see, e.g., [4, 5]). These estimators instead form implicit model order decisions based on one or more tuning parameters that dictate the relative weight of various penalties. As shown in the above cited works, the resulting estimators are able to allow for (rapidly) varying model orders, without significant loss of performance. The noted sparse multi-pitch estimators process each data frame separately, treating each as an isolated and stationary measurement, without exploiting the information obtained from earlier data frames when forming the estimates. To allow for such correlation over time, the estimator in [4] was recently extended to exploit the previous pitch estimates, as well as the power distribution of the next frame, when processing the current data frame [6]. In this work, we extend on this effort, but instead propose a fully time-recursive problem formulation using the sparse recursive least-squares (RLS). The resulting estimator does not only allow for more stable pitch estimates as compared to earlier sparse multi-pitch estimators, as more information is used at each time-point, but also decreases the computational burden of each update, as new estimates are formed by updating already available ones. Sparse adaptive filtering is a field attracting steadily increasing attention, with, for instance, the sparse RLS algorithm being explored for adaptive filtering in, e.g., [7, 8]. Other related studies include [9], wherein the authors use a projection approach to solve a recursive LASSO-type problem, and [10], which introduced an online recursive method allowing for an underlying dynamical signal model and the use of sparsity-inducing penalties. Recursive algorithms designed for group-sparse systems have also been introduced, such as the ones presented in [11–13], but to the best of our knowledge, no such technique has so-far been applied to the multi-pitch estimation problem. The remainder of this paper is organized as follows; in the next section, we introduce the multi-pitch signal model and its corresponding dictionary formulation. Then, in section 3, we introduce the group sparse RLS formulation for multi-pitch estimation, followed by a discussion about various algorithmic considerations. Finally, section 4 contains numerical examples illustrating the performance of the proposed estimator on various audio signals.
2. SIGNAL MODEL

Consider a measured signal $^1$, $y(t)$, that is generated according to the model $y(t) = x(t) + e(t)$, where

$$x(t) = \sum_{k=1}^{K(t)} \sum_{\ell=1}^{L_k(t)} \tilde{w}_{k,\ell}(t)e^{i2\pi f_k(t)\ell t}$$

(1)

with $K(t)$ denoting the number of pitches at time $t$, with fundamental frequencies $f_k(t)$, having $L_k(t)$ harmonics, and where $e(t)$ denotes a broad-band noise. It should be stressed that the number of pitches, as well as their fundamental frequencies and the number of harmonics for each source, may vary over time. As in [4, 5], the signal sources are approximated using a sparse modeling framework containing $P$ candidate pitches, each allowed to have up to $L_{\max}$ harmonics, such that

$$x(t) \approx \sum_{p=1}^{P} \sum_{\ell=1}^{L_{\max}} w_{p,\ell}(t)e^{i2\pi f_p(t)\ell t}$$

(2)

where the dictionary is selected large enough so that (at least) $K(t)$ candidate pitches, $f_p(t)$, reasonably well approximate the true pitch frequencies, i.e., such that $P \gg \max_t K(t)$ and $L_{\max} \gg \max_{t,k} L_k(t)$. It should be noted that as the signal is assumed to contain relatively few pitches at each time instance, the resulting amplitude dictionary will be sparse, although with an harmonic structure reflecting the overtones of the pitches. Furthermore, it may be noted that the frequency grid-points, $f_p(t)$, are allowed to vary with time, which will be implemented using a dictionary learning scheme. Using this framework, the pitches present in the signal at time $t$ may be implicitly estimated by identifying the non-zero amplitude coefficients, $w_{p,\ell}(t)$.

3. GROUP-SPARSE RLS FOR PITCHES

Exploiting the structure of the signal, we introduce the group-sparse filter $w(t)$, which at time $t$ is divided into $P$ groups according to

$$w(t) = [ w_1(t)^T \ldots w_P(t)^T ]^T$$

(3)

$$w_p(t) = [ w_{p,1}(t) \ldots w_{p,L_{\max}}(t) ]^T$$

(4)

implying that, ideally, only $K(t)$ subvectors $w_{p}(t)$ will be non-zeros at time $t$. In order to achieve this, the filter is formed

$$\hat{w}(t) = \arg \min_w g_t(w) + h_t(w)$$

(5)

where

$$g_t(w) = \frac{1}{2} \sum_{\tau=1}^{t} \lambda^{t-\tau} \| y(\tau) - w^T a(\tau) \|_2^2$$

(6)

with $h_t(w)$ denoting a sparsity inducing penalty function,

$$a(t) = [ a_1(t)^T \ldots a_P(t)^T ]^T$$

(7)

$$a_p(t) = [ e^{i2\pi f_p(t)\ell t} \ldots e^{i2\pi f_p(t)L_{\max\ell t}} ]^T$$

(8)

and $\lambda \in (0, 1)$ being a user-determined forgetting factor. The sparsity inducing function, $h_t(w)$, should be selected to encourage a pitch-structure in the solution; in [5], which considered multi-pitch estimation on isolated time frames, this function was selected as

$$h_t(w) = \gamma_1 \| w \|_1 + \sum_{p=1}^{P} \| Fw_{G_p} \|_2$$

(9)

where $F$ is the first difference matrix and $G_p$ is the index set for candidate pitch $p$. This choice was there shown to induce the sought group sparsity, although requiring some additional refinements. To allow for a fast implementation, we will here instead consider the penalty function

$$h_t(w) = \gamma_1 \| w \|_1 + \sum_{p=1}^{P} \gamma_2 \| w_{G_p} \|_2$$

(10)

where $\gamma_1(t)$ and $\gamma_2,p(t)$ are non-negative regularization parameters. This is reminiscent of the so-called PEBS method introduced in [4]. However, when, as in [4], using fixed penalty parameters $\gamma_1(t)$ and $\gamma_2,p(t)$, the resulting estimate has been shown to be prone to mistaking a pitch for its sub-octave. In order to discourage this type of erroneous solutions, we will herein introduce a way of adaptively choosing the group sparsity parameter $\gamma_2,p(t)$, as further discussed below. Noting that $g_t(w)$, as defined in (6), may be expressed in matrix form as

$$g_t(w) = \left\| A_{1:t}^{1/2} y_{1:t} - A_{1:t}^{1/2} A_{1:t} w \right\|_2^2$$

(11)

where

$$y_{1:t} = [ y(1) \ldots y(t) ]^T$$

(12)

$$A_{1:t} = [ a(1) \ldots a(t) ]^T$$

(13)

and with $A_{1:t} = \text{diag} \left( \lambda^{t-1} \lambda^{t-2} \ldots 1 \right)$, define

$$R(t) \triangleq A_{1:t}^H A_{1:t}$$

(14)

$$r(t) \triangleq A_{1:t}^H A_{1:t} y_{1:t}$$

(15)

With these definitions, the minimization in (5) may be formed using proximal gradient iterations, such that the $j$th iteration may be expressed as

$$\hat{w}^{(j+1)}(t) = \arg \min_w \frac{1}{2s(t)} \left\| \nu^{(j)} - w \right\|_2^2 + h_t(w)$$

(16)

where

$$\nu^{(j)} = \hat{w}^{(j)}(t) + s(t) \left( r(t) - R(t)\hat{w}^{(j)}(t) \right)$$

(17)

---

$^1$For notational and computational simplicity, we here consider the discrete-time analytic signal of any real-valued measured signal.
with $s(t)$ denoting the step-size. We note that this update is similar to the one presented in [8], there cast as an EM-algorithm using some further assumptions about the signals properties. The update in (16) may be computed in closed form for each group $p$ separately as

$$
\hat{\nu}^{(j)}_{gp} = S_1 \left( \nu^{(j)}_{gp}, s(t) \gamma_1(t) \right) \\
\hat{w}^{(j+1)}_{gp}(t) = S_2 \left( \hat{\nu}^{(j)}_{gp}, s(t) \gamma_{2,p}(t) \right)
$$

(18) (19)

where $S_1$ and $S_2$ are the soft thresholding operators corresponding to the $\ell_1$- and $\ell_2$-norms, respectively, i.e.,

$$
S_1(z, \alpha) = \frac{\max (|z| - \alpha, 0)}{\max (|z| - \alpha, 0) + \alpha} \odot z
$$

(20)

$$
S_2(z, \alpha) = \frac{\max (\|z\|_2 - \alpha, 0)}{\max (\|z\|_2 - \alpha, 0) + \alpha} - z
$$

(21)

where in (20) the max and absolute value functions operate element-wise on the vector $z$ and $\odot$ denotes element-wise multiplication. Furthermore, $R(t)$ and $r(t)$ can be updated as new samples become available according to

$$
R(t) = \lambda R(t-1) + a(t)a(t)^H
$$

(22)

$$
r(t) = \lambda r(t-1) + y(t)a(t)
$$

(23)

where $\cdot^*$ denotes complex conjugation.

We proceed to examine some implementation aspects of the presented algorithm, first discussing the appropriate choice of the penalty parameters, then possible computational speed-ups, as well as ways of adaptively updating the used pitch dictionary.

In order to discourage solutions containing erroneous sub-octaves, we here propose to update the group penalty parameter, in iteration $j$ of the filter update, as

$$
\gamma_{2,p}(t) = \gamma_2(t) \max \left( 1, \frac{1}{|\hat{w}^{j-1}_{p,1}(t)| + \epsilon} \right)
$$

(24)

where $|\hat{w}^{j-1}_{p,1}(t)|$ is the estimated amplitude of the first harmonic of group $p$, obtained in iteration $j - 1$, with $\epsilon \ll 1$ being a user-specified parameter selected to avoid a division by zero. As sub-octaves will have missing first harmonics, such a choice will encourage shifting power from the sub-octave to the proper pitch. Studies using many different kinds of pitch signals indicate that the overall performance of the algorithm is relatively insensitive to the choice of the parameter $s(t)$, which may typically be selected in the range $s(t) \in [10^{-5}, 10^{-3}]$. Performance is also robust to different choices of the group-penalty parameter, $\gamma_2(t)$, with a good rule of thumb being choosing it in the neighborhood of (25) with $\mu = 1$. Furthermore, the choice of the parameter $\gamma_1(t)$ can be made using inner-products between the dictionary and the signal. Letting $\Delta$ denote the time-lag, we update the $\gamma_1(t)$ at time $t$ to

$$
\gamma_1(t) = \mu \left\| \mathbf{A}^{1/\Delta} \mathbf{A}^{1/\Delta} \mathbf{y}_{t-\Delta} \right\|_\infty
$$

(25)

where $\mu \in (0, 1)$, with $\mu = 0.2$ being a reasonable choice. This will emulate choosing $\gamma_1(t)$ based on the correlation between the signal and the dictionary in a finite window. Here, the window length, $\Delta$, is determined by the forgetting factor, $\lambda$, and by how much correlation one is willing to lose as a result from the truncation. For example, selecting

$$
\Delta = \frac{\log(0.01)}{\log \lambda}
$$

(26)

will yield a window such that the excluded samples will contribute to less than 0.01 of the correlation. It should be noted that for smoothly varying signals, $\gamma_1(t)$ only needs to be updated infrequently.

As the signal is assumed to have a sparse representation in the dictionary $\mathbf{a}(t)$, one may expect updates of the coefficients of many groups, here indexed by $q$, to result in zero amplitude estimates. As such groups do not contribute to the pitch estimates, these groups would preferably be excluded from the updates in (16)-(17). If assuming the support of $\mathbf{w}(t)$ to be constant for all $t$, one could thus sequentially discard such groups from the updating step, and thereby decrease computation time. However, as generally pitches may disappear and then re-appear, as well as drift in frequency over time, we will here only exclude the groups $q$ from the updating steps temporarily. That is, if at time $\tau$, we have $\|\mathbf{w}_q\|_2 < \check{\epsilon}$, where $\check{\epsilon} \ll 1$, the group $q$ is considered not to be present in the signal and is therefore excluded from the updating steps for a waiting period. After that period, it is again included in the
updates, allowing it to again appear in the signal. Defining the set $\mathcal{U}$, indexing the groups that are considered active, the group $q$ is adaptively included and excluded from $\mathcal{U}$ depending on the size of $\left\| \hat{w}_q \right\|_2$.

In general, a signal’s pitch frequencies may vary over time, for instance, due to vibrato. Applying the filter updating scheme using fixed grid-points will therefore result in rapidly changing support of the filter or energy leakage between adjacent blocks of the filter, here indexed by $p$. In order to overcome this problem, and to allow for smooth tracking of pitches over time, we propose a scheme for adaptively updating the dictionary of candidate pitches. This adaptive adjustment scheme also allows for the use of a grid with coarser resolution than would otherwise be possible. Let $T = \{ \tau_k \}_k$ be the set of time points in which the dictionary is updated. As only groups $\hat{w}_{\nu_{p}}(\tau_k)$ with non-zero power are considered to be present in the signal, one only has to adjust the fundamental frequencies of these. Assuming that the current estimate of such a candidate pitch frequency is $f_p(\tau_{k-1})$, one only needs to consider adjusting it on the interval $f_p(\tau_{k-1}) \pm \frac{1}{2} \delta_f$, where $\delta_f$ denotes the current grid-point spacing. The update can be formed using the approximate non-linear least-squares method in [2] where, instead of $L_{\text{max}}$, one uses the harmonic order corresponding to the non-zero components of $\hat{w}_{\nu_{p}}(\tau_k)$. The adjusted frequency $f_p(\tau_k)$ is then used to update the dictionary on the time interval $[\tau_k, \tau_{k+1})$.

Together with the discussed algorithmic considerations, the presented time-recursive multi-pitch estimator is detailed in Algorithm 1. The algorithms is termed the Pitch Estimation using dictionary-Adaptive Recursive Least Squares (PEARLS) method.

### 4. NUMERICAL RESULTS

In this section, we evaluate the performance of the proposed PEARLS algorithm using real audio recordings. As a first illustration, Figure 1 shows a plot of the estimated fundamental frequencies when applying the PEARLS algorithm to a signal consisting of three trumpets playing the tones A4, B4, and D♭5, corresponding to pitches with fundamental frequencies 440, 493.88, and 554.37 Hz respectively. The pitch amplitudes are illustrated by the color of each pitch track. As can be seen from the figure, the trumpets are playing with a vibrato, making the fundamental frequencies oscillate in a sinusoidal-like pattern around the center frequency. This illustrates the usefulness of the dictionary update scheme: with a fixed grid, one would not be able to track each pitch as smoothly. The trumpets are sustaining their respective pitches from the start of the recording and release them after 4.2 seconds. The examined signal was sampled at 11 kHz, and $L_{\text{max}} = 6$ was used as the maximum harmonic order for the algorithm. The forgetting factor was $\lambda = 0.985$, and the dictionary was updated every 10 ms using 22 ms of signal samples. Using the same algorithm settings, we proceed to evaluate the performance of PEARLS on the Bach10 dataset [14]. This dataset consists of ten excerpts from chorals composed by J. S. Bach, and have been arranged to be performed by an ensemble consisting of a violin, a clarinet, a saxophone, and a bassoon, with each excerpt being 25-42 seconds long. The PEARLS estimates were compared to ground truth values with a time-resolution of one reference point every 30ms. The ground truth fundamental frequencies were obtained by applying the single-pitch estimator YIN [15] to each separate channel with manual correction of obvious errors. The results are presented in Table 1, presenting values of the performance measures Accuracy, Precision, and Recall, as defined in [16]. As in [16], an estimated fundamental frequency is associated with a ground truth fundamental frequency if it lies within a quarter-tone, or 3%, of the ground truth fundamental frequency. For comparison, Table 1 also includes corresponding performance measures for the algorithm PEBSI-Lite [5] and ESACF [3]. The values for PEBSI-Lite and ESACF were originally presented in [5], and the settings for these algorithms are the same as is presented there. As can be seen, PEARLS clearly outperforms ESACF although not performing as well as PEBSI-Lite when considering these measures. It should also be said that PEARLS is considerably more computationally efficient than PEBSI-lite. Using a MATLAB implementation of PEARLS.

<table>
<thead>
<tr>
<th></th>
<th>PEARLS</th>
<th>PEBSI-Lite</th>
<th>ESACF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.393</td>
<td>0.449</td>
<td>0.269</td>
</tr>
<tr>
<td>Precision</td>
<td>0.596</td>
<td>0.631</td>
<td>0.471</td>
</tr>
<tr>
<td>Recall</td>
<td>0.535</td>
<td>0.609</td>
<td>0.386</td>
</tr>
</tbody>
</table>

Table 1. Performance measures for the PEARLS, PEBSI-Lite and ESACF algorithms, when evaluated on the Bach10 dataset.

---

**Algorithm 1** The PEARLS algorithm

1. initialize $\hat{w}(0) \leftarrow 0$ and $R(0) \leftarrow 0$, $r(0) \leftarrow 0$, $t \leftarrow 1$
2. repeat \{Recursive update scheme\}
   3. $R(t) \leftarrow \lambda R(t-1) + a(t)a(t)^H$
   4. $r(t) \leftarrow \lambda r(t-1) + y(t)a(t)$
   5. $j \leftarrow 0$
   6. $\hat{w}(j)(t) \leftarrow \hat{w}(t-1)$
   7. repeat \{Proximal gradient update\}
   8. $\nu(j) \leftarrow \hat{w}(j)(t) + s(t) \{r(t) - R(t)\hat{w}(j)(t)\}$
   9. $\hat{w}(j+1) \leftarrow \arg \min_w \frac{1}{2s(t)} \|\nu(j) - w\|^2 + h_t(w)$
   10. $j \leftarrow j + 1$
   11. until convergence
   12. $\hat{w}(t) \leftarrow \hat{w}(j)(t)$
   13. Update active set $\mathcal{U}$
   14. if $t \in T$ then
   15. Update dictionary
   16. end if
   17. $t \leftarrow t + 1$
18. until end of signal

---

We note that the current implementation has not exploited that the filter updating step (16) can be done for all $P$ candidate pitches in parallel. Similarly, the computations for PEBSI-Lite can also be parallelized, as each time frame can be processed in isolation.
run on a 2.68 GHz PC, the average running time for the Bach pieces was 16 minutes, which were on average 33 seconds long. For PEBSI-Lite, the average running time was 54 minutes, when run on the same computer. For PEBSI-Lite, the signal had been divided into non-overlapping frames of length 30 ms. As an illustration of the performance of PEARLS on the Bach10 dataset, Figures 2 and 3 present plots of the estimated fundamental frequencies obtained using ESACF and PEARLS, respectively, for the piece *Ach, Gott und Herr*, as compared to the ground truth for each instrument. Here, in order to make a fair comparison of the computational complexities of the estimators, the ESACF estimate was computed on windows of length 30 ms, where two consecutive windows overlapped in all but one sample. Although ESACF can arguably be applied to windows with smaller overlap, this setup meant that ESACF would produce pitch tracks with the same time resolution as PEARLS. This resulted in an average running time of 11 minutes per music piece, i.e., comparable to that of PEARLS. As can be seen from the figures, PEARLS is considerably better at tracking the instruments than ESACF.

5. REFERENCES