Parametric dictionary design for sparse coding

Citation for published version:

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Early version, also known as pre-print

Published In:
IEEE Transactions on Signal Processing

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Parametric Dictionary Design for Sparse Coding

Mehrdad Yaghoobi, Member, IEEE, and Laurent Daudet, Member, IEEE and Mike E. Davies, Member, IEEE

Abstract—This paper introduces a new dictionary design method for sparse coding of a class of signals. It has been shown that one can sparsely approximate some natural signals using an overcomplete set of parametric functions, e.g. [1], [2]. A problem in using these parametric dictionaries is how to choose the parameters. In practice these parameters have been chosen by an expert or through a set of experiments. In the sparse approximation context, it has been shown that an incoherent dictionary is appropriate for the sparse approximation methods. In this paper we first characterize the dictionary design problem, subject to a constraint on the dictionary. Then we briefly explain that equiangular tight frames have minimum coherence. The complexity of the problem does not allow it to be solved exactly. We introduce a practical method to approximately solve it. Some experiments show the advantages one gets by using these dictionaries.


I. INTRODUCTION

PARSE modeling of signals has recently received much attention as it has shown promising results in different applications. It has been used for coding, source separation, feature extraction and compressive sampling. A basic assumption to apply this model is that the given class of signals can be sparsely represented or approximated in an underdetermined generative model. Often, a linear model has been used as the generative model. In this framework, one can use a matrix \( D \in \mathbb{R}^{d \times N} \), called dictionary, to represent the signal approximately using \( y \approx Dx \).

Sparse approximation and sparse representation methods have been studied theoretically and practically [3]. Let \( y \in \mathbb{R}^d \) and \( x \in \mathbb{R}^N \) be the given signal and the coefficient vector respectively. A sparse approximation would be,

\[
\hat{x} = \arg\min_x \|x\|_0 \quad \text{s.t.} \quad \|y - Dx\|^2 \leq \xi,
\]

where \( \|x\|_0 \) is the sparsity measure that counts the number of the non-zero coefficients and \( \xi \) is a small positive constant. This problem in general, like the sparse representation problem \( (\xi = 0) \), is an NP-hard problem [4] and can not be solved in a reasonable time. Numerous algorithms have been proposed to find an approximate solution. These algorithms are classified as greedy methods, like Matching Pursuit (MP) [5] and its derivations [6], and relaxation methods, like Basis Pursuit Denoising (BPDN) [7] and IRLS-type algorithms [8], [9].

The sparsity of the representation can be increased using an appropriate dictionary for the given class of signals. The common methods for dictionary selection are to concatenate orthogonal bases, see for example [10] and [11] for the possible advantages of using such a dictionary in theory and practice, or to use a tight frame [12]. These dictionaries can be improved using dictionary learning methods [13]–[16]. These methods adapt an initial dictionary to a set of training samples. Therefore the aim is to learn a dictionary for which an input signal, taken from a given class of signals, has a sparse approximation.

There is another dictionary selection method, which is called dictionary design. Different methods exist to design a suitable \( D \) for a set of natural signals. One method is based on a generative model of the signals. If these signals are to be received by the human sensory system, a more effective method to design \( D \) is to use a human perception model [1], [2]. In fact, the stimuli responses generate elementary functions which are more related to the analysis dictionary [17]. These elementary functions have also been used for generating the synthesis dictionary \( D \). Here, we assume that the set of elementary functions can be described by using a set of parameters and a parametric function. For example, in the multiscale Gabor functions [5], the parameters are scale, time and frequency shifts and the parametric function is Gaussian.

In general the parameters are in the continuous domain. To generate a dictionary based on these generative functions, we can sample these continuous parameters. The question is then how best to sample the parameters. Several researchers have introduced different methods to optimize the sampling process. In [18], a sampling scheme was introduced which finds an approximately tight frame, using 2D Gabor functions. Gammatone and Gammachirp filter banks have been shown to approximate the human auditory system. [19] presented two types of filters, which approximate the Gammatone filter banks, and allow a possible fast VLSI implementations. Alternatively, some researchers optimized the parameters based on the closeness to what is observed in the perceptual systems [20], [21], [22]. In practice, [23] showed that the optimal parameters, found by fitting to the human auditory system, do not match the parameters estimated from English speech signals.

When we use an approximate or a relaxed method to find a sparse approximation, having an exact generative model does not guarantee that we find the best sparse approximation. An important parameter of a dictionary, for successful sparse recovery, is its coherence \( \mu \) [24]. The coherence is defined as the absolute value of the largest inner-product of two

Copyright (c) 2008 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org.

This research was partially supported by EPSRC grant D000246/1 and EU FP7, FET-Open grant number 225913. M. Yaghoobi and M. Davies are with the Institute for Digital Communication and with the Joint Research Institute for Signal and Image Processing, Edinburgh University, Kings Buildings, Mayfield Road, Edinburgh EH9 3LJ, UK (e-mail: yaghoobi@ieee.org, mike.davies@ed.ac.uk). L. Daudet is with UPMC Univ. Paris 06, LAM / Institut Jean le Rond d’Alembert (UMR 7190), Paris, France (email: daudet@lam.jussieu.fr).
use parametric dictionaries. In [30], [31] some methods are proposed to sparsely approximate signals using continuous parameter parametric dictionaries. The convergence rate of MP algorithm with this setting is also studied in [31]. In contrast the designed dictionary, using parametric dictionary design, is discretized and can be used by the conventional sparse coding methods.

A. Contributions of the paper

In this paper we introduce a new framework for dictionary design. To the authors knowledge, this formulation has not been considered previously. This formulation can be used to design a dictionary when dictionary learning is not possible, or is computationally intractable. We show how we can find an approximate solution using an alternating minimization type method.

The parametric dictionary is represented using a small number of parameters (often less than 5). Therefore we do not need to store the dictionary explicitly. This can save a considerable amount of memory when using sparse approximation algorithms.

Finally we show experimentally that there are sparse approximation benefits in using such a parametric dictionary for audio coding.

B. Organization of the paper

In the next section we formulate the parametric dictionary design problem. We then present a practical algorithm to find an approximate solution. For a case study we present the parametric dictionary formulation and the update formula derivation. Experiments, in the simulation subsection, show the advantages of the proposed dictionary design. The stability of the algorithm is analyzed after its introduction in Section III, while the convergence of the proposed algorithm is shown in Appendix A.

C. Notation

In this paper we use small and capital bold face characters to indicate vectors and matrices respectively. All the parameters have real values, even though we do not state this explicitly each time. The matrix and vector norm spaces that we use in this paper are defined over the real fields with \( \ell_2 \) and \( \| \cdot \|_F \), which is the Frobenius norm, as the corresponding norms respectively.

The tensor product used in this paper is for the multiplication of two three-dimensional arrays. This multiplication uses the first two indices to make a simple matrix-matrix product and the third parameter as the indices of these products. In other words, the third parameter specifies two matrices from the three dimensional tensors and simplifies the tensor product to matrix-matrix multiplications. The number of these multiplications is the size of third index.

The terms “ETF” and “Grassmannian Frame” have been used interchangeably for the same concept [32], [28], [29]. In this paper we prefer to use ETF, which is more comprehensive.

1The optimal dictionary is that by which the given class of signals has the sparsest approximation.
II. PARAMETRIC DICTIONARY DESIGN: FORMULATION

In this section we formulate the problem of optimizing $\mathbf{D}$ to be close to an ETF. Let $\mathbf{D}_F \in \mathcal{D}$ be a parametric dictionary. $\Gamma$ is the parameter matrix, with $\gamma_i$ as its $i^{th}$ column and $\mathcal{D}$ is the set of admissible parametric dictionaries. Each column of $\mathbf{D}_F$, $\mathbf{d}_i$ (with the associated parameters $\gamma_i$), is called an atom. In this paper, by letting $\mathbf{D}_F$ be a matrix, we implicitly assume that the generative model is discrete. This model can be extended to a continuous model, which is out of our scope.

To select a column normalized dictionary $\mathbf{D}$, we need to introduce an objective. In section I we explained that, if and only if, $\mathbf{D}$ is the solution of, $\inf_{\mathbf{G}, \mathbf{D} \in \mathcal{F}} \| \mathbf{D}^T \mathbf{D} - \mathbf{G} \|_F$, where $\mathbf{G}$ is the Gram matrix of the dictionary, the set of admissible parametric dictionaries. Each column of $\mathbf{D}_F$, $\mathbf{d}_i$ (with the associated parameters $\gamma_i$), is called an atom. In this paper, by letting $\mathbf{D}_F$ be a parametric dictionary, we introduce the following formulation, $\inf_{\mathbf{G}, \mathbf{D} \in \mathcal{F}} \| \mathbf{D}^T \mathbf{D} - \mathbf{G} \|_F$, where $\| \cdot \|_F$ is the Frobenius norm. This is a non-convex optimization problem in general. It might have a set of solutions or it may not have any solution (e.g. $\Theta^N_d$ is empty as there do not always exist ETF’s for the arbitrary $N$ and $d$). One can extend $\Theta^N_d$ to a convex set $\Lambda^N$ [28], which is non-empty for any $N$, by $\Lambda^N = \{ \mathbf{G} \in \mathbb{R}^{N \times N} : \mathbf{G} = \mathbf{G}^T, \text{diag} \mathbf{G} = 1, \max_{i \neq j} |g_{i,j}| \leq \mu_G \}$. Relaxing (2), by replacing $\Theta^N_d$ with $\Lambda^N$, gives the following optimization problem.

Let $\Theta^N_d$ be the set of Gram matrices of all $d \times N$ ETFs.

An important difference between (2) and (3) is that the relaxed problem, by using non-empty admissible sets, is guaranteed to have at least one solution. In this work, it is assumed that $\mathcal{T}$ is closed, which allows us to use the “$\inf$” operator instead of “$\min$” in (3). We therefore use the relaxed formulation from now on. We show experimentally that the approximate solutions of (3), even though the Gram matrix of the dictionary might only be close to $\Lambda^N$, show good performances in sparse approximation.

Fig. 1. Different alternating optimization methods: (a) Alternating Projection, (b) Alternating Minimization and (c) Proposed Method.

$\mu_G$ are one and $\mu_G$ respectively. A nearness measure of $\mathbf{D} \in \mathbb{R}^{d \times N}$ to the set of ETFs can be defined as the minimum distance between the Gram matrix of $\mathbf{D}$ and $\mathbf{G}_2 \in \Theta^N_d$ [28]. To optimize the distance of a dictionary to an ETF, we can solve, $\min_{\mathbf{G}, \mathbf{D} \in \mathcal{F}} \| \mathbf{D}^T \mathbf{D} - \mathbf{G} \|_F$.

Although the matrix space with $\ell_\infty$ is a well defined Banach space, here, we use $\ell_2$ norm Hilbert space to use easy formulation of the optimization process.
In the next section we introduce a practical method to find an approximate solution to (3). Our approach has similarities with alternating minimization. This method is guaranteed not to increase the objective function in each step. Because the objective is non-negative, the algorithm is stable due to Lyapunov’s second theorem [33]. Also, one can show that the objective value converges. The stability of the algorithm and the convergence of the objective value do not prove the convergence of the algorithm. In Appendix A, it has been shown that the algorithm converges to a set of accumulation points under mild conditions.

III. PARAMETRIC DICTIONARY DESIGN: A PRACTICAL ALGORITHM

A standard method to solve (3) is alternating projection, see for example [34], [28] and references therein. In this method we alternatingly project the current solution onto the admissible sets, see Fig.1.a. When the admissible sets are convex, the algorithm converges to a solution in $\mathcal{D} \cap \Lambda^N$ or a pair of solutions in $\mathcal{D}$ and $\Lambda^N$, when $\mathcal{D} \cap \Lambda^N = \emptyset$, respectively. In the following, we derive a formulation for the projection onto $\Lambda^N$, but there is no easy formulation for the projection onto the set of admissible dictionaries, in general. Therefore we choose a different method which has similarities with alternating minimization [36] (or generalized alternating projection [37]), see Fig.1.b. In the alternating minimization framework, we choose the new solutions in $\mathcal{D}$ and $\Lambda^N$ alternatingly such that the objective does not increase in each update and is thus stable. If the algorithm converges, the fixed point is either in $\mathcal{D} \cap \Lambda^N$, or is a pair of points in $\mathcal{D}$ and $\Lambda^N$ respectively.

Although the proposed algorithm has similarities with alternating minimization, it does not follow its steps exactly. The difference is that in the stage in which we update the current solution with respect to $\Lambda^N$, we choose a point which is somewhere between the current solution and the projection onto $\Lambda^N$. Fig.1.c shows a schematic representation of the proposed method. The reason for this modification is that by projection onto $\Lambda^N$, the structure of the Gram matrix changes significantly so that the selection of a new point in $\mathcal{D}$ in the following step is very difficult. We can gradually select a closer point to the projected point on $\Lambda^N$, when the current $\mathbf{D}_t$ is close to $\Lambda^N$. In the other step, we update $\mathbf{D}$ such that it does not increase the objective in (3).

The parametric dictionary design is summarized in Algorithm 1. In line 4, the algorithm finds the projection onto $\Lambda^N$. In line 6, a point in $\mathcal{D}$ is selected which is closer to $\mathbf{G}_{R_{k+1}}$. In the following we show how we calculate the updates in lines 4 and 6.

A. Projection onto $\Lambda^N$:

In the objective function (3), $\mathbf{G}$ is a Hermitian matrix. By sign change of any related off-diagonal pair of elements, i.e. $g_{i,j}$ and $g_{j,i}$, we get a new $\mathbf{G} \in \Lambda^N$. The closest $\mathbf{G}$ to $\mathbf{D}_t^2 \mathbf{D}_t$, in a Frobenius norm space, is the $\mathbf{G}$ with a similar sign pattern. We know that in a normed space, finding the nearest elements is a small positive value. The parameter $\epsilon$ should be chosen such that the update reduces the objective function in

Algorithm 1   Parametric Dictionary Design

1: initialization: $k = 1$, $\mathbf{D}_{G_1} \in \mathcal{D}$, $\{\alpha_l\}_{1 \leq l \leq K} : 0 < \alpha_l \leq 1$
2: while $k \leq K$ do
3: \hspace{1em} $\mathbf{G}_{R_k} = \mathbf{D}_{G_k}^2 \mathbf{D}_{G_k}$
4: \hspace{1em} $\mathbf{G}_{F_{R_k+1}} = \min_{\mathbf{G} \in \Lambda^N} \| \mathbf{G}_{R_k} - \mathbf{G} \|_F$
5: \hspace{1em} $\mathbf{G}_{R_{k+1}} = (1 - \alpha_k) \mathbf{G}_{R_k} + \alpha_k \mathbf{G}_{F_{R_k+1}}$
6: \hspace{1em} $\mathbf{D}_{R_{k+1}} = \mathbf{D}_{G_k} \cup \{ \mathbf{D} \in \mathcal{D} : \| \mathbf{D}^2 \mathbf{D} - \mathbf{G}_{R_{k+1}} \|_F < \| \mathbf{G}_{R_k} - \mathbf{G}_{R_{k+1}} \|_F \}$
7: \hspace{1em} $k = k + 1$
8: end while

Algorithm 2   Parameters Update

1: initialization: $l = 1$, $\Gamma^{[0]}_l = \Gamma_k$, $\epsilon \in \mathbb{R}^+$, $\phi(\Gamma) = \| \mathbf{D}_t^2 \mathbf{D}_t - \mathbf{G} \|_F^2$
2: for all $l \leq L$ do
3: \hspace{1em} $\Gamma^{[l+1]}_l = \Gamma^{[l]}_l - \epsilon \nabla \phi(\gamma^{[l]}_k)$
4: \hspace{1em} $l = l + 1$
5: end for
6: $\Gamma_{k+1} = \Gamma^{[L]}_L$

4Here stability means boundedness of the algorithm output.
5At least in finite dimensional spaces. There are counter-examples for the lack of convergences in the infinite dimension setting [35].
In this framework, \( \Gamma_{k+1} = \lim_{t \to \infty} \Gamma_{k+1}^{[p]} \). In practice we stop after a given number of iterations or when \( \nabla_{\Gamma} \phi(\Gamma) \) becomes very small. Algorithm 2 summarizes this parameter update algorithm.

Because \( \phi(\Gamma) \) is a continuous function, its epigraph [40], for an initial \( \Gamma_0 \), is closed. By choosing a bounded set of admissible parameters \( \Upsilon \), the epigraph is a compact set in Euclidean space. To show that the algorithm gets as close as possible to the set of limit points, we need to use the Bolzano-Weierstrass theorem.

**Theorem 2:** [41, 3.24] Every bounded infinite subset of \( \mathbb{R}^N \) has at least one limit point in \( \mathbb{R}^N \).

Therefore, when the set of admissible parameters is bounded and \( \epsilon \) is selected such that moving in the gradient direction with this step size reduces the objective, this gradient descent algorithm has at least one limit point in the admissible set.

**Remark 1:** The function \( \phi(\Gamma) \) is a lower bounded function. Hence, if we reduce \( \phi \) in each iteration, due to Lyapunov’s second theorem [33], the algorithm is stable.

**Remark 2:** Algorithm 1 is an iterative algorithm in which we also used another iterative method for the dictionary update in line 6. The stability and the convergence of the updates mentioned above were related to the inner loop in Algorithm 1. We deal with the convergence of Algorithm 1 in Appendix A.

**Remark 3:** We draw the readers attention to the formulation (6). The parameters \( \nabla_{D_{\Gamma}} \phi \) and \( \nabla_{\Gamma} \phi \) are tensors of rank 3, 2 and 2 respectively. If \( \Gamma \in \mathbb{R}^{p \times N} \) and \( D \in \mathbb{R}^{d \times N} \) then \( \nabla_{\Gamma} D_{\Gamma} \in \mathbb{R}^{p \times d \times N}, \nabla_{D_{\Gamma}} \phi \in \mathbb{R}^{d \times 1 \times N} \) and \( \nabla_{D} \phi \in \mathbb{R}^{p \times 1 \times N} \). A graphical presentation of this formulation is presented in Fig. 2. Furthermore, to use this directional update in (7), we need to map \( \nabla_{\Gamma} \phi \in \mathbb{R}^{p \times 1 \times N} \) into the appropriate matrix in \( \mathbb{R}^{p \times N} \). It is easily done by changing the order of indices (1, 2, 3 to 1, 3, 2), following by cancelling the third dimension. Because the rank of \( \nabla_{\Gamma} \phi \) is 2, this mapping is injective.

![Fig. 2. The chain rule (6) in the tensor form.](image)

The problem we formulated in this paper is developed in a general form. To show the advantages of using parametric dictionary design, we choose a case study. In sparse audio processing, an important question is how to choose the dictionary [42], [43]. Different methods have been introduced to adapt the dictionary to better fit a set of training samples [44], [45], [46]. For example, some researchers used a class of parametric dictionaries based on Gammatone filter banks, which have been shown to have similarities with the human auditory system [23], [47]. We now show that the parametric dictionary design improves the performance of audio sparse approximation and exact recovery based around a Gammatone representation.

### A. Gammatone parametric dictionary

The generative function for a Gammatone dictionary is as follows,

\[
g(t) = a t^{n-1} e^{-2\pi B t} \cos(2\pi f_c t),
\]

where \( B = f_c/Q + b_{\text{min}}, f_c \) is the center frequency and \( a \in \mathbb{N}, a, b, Q, b_{\text{min}} \) are some constants. The optimal parameter selection is not easy. One can select the parameters such that the generated atoms match the auditory impulse response. The auditory system has been optimized through evolution and may not be optimized for a practical application. Our goal is to optimally select these parameters so that sparse approximation methods can be used. Another difficulty in using the Gammatone filter banks as a dictionary is its large size. A moderate size dictionary can be designed by the proposed method.

The dictionary is generated by sampling the parameters of \( g(t - t_c) \), where \( t_c \) is the time-shift. In this paper, \( \gamma = [c_c, f_c, n, b]^T \) are the optimization parameters. The parameters \( t_c \) and \( f_c \) change the center of the atoms in the time-frequency plane. \( n \) and \( b \) control the rise time and the width of the atoms in the time domain, respectively. The parameter \( a \) is chosen to normalize the atom to unit length. Let \( \{\gamma_i\}_{1 \leq i \leq N} \) be a set of the parameters and \( g_{\gamma_i}(t) \) be the atom generated using \( \gamma_i \). The parameter matrix \( \Gamma \) and the parametric dictionary \( D_{\Gamma} \) are generated using \( \gamma_i \) and \( g_{\gamma_i}(|t f_{\text{samp}}|) \) as the columns respectively, where \( f_{\text{samp}} \) is the sampling frequency.

The differentiability of \( D_{\Gamma} \) with respect to \( \Gamma \) makes the parameter update easier. In this paper we assume the parametric dictionary satisfies this constraint. Letting \( n \in \mathbb{R}, (8) \) becomes a generative function over a continuous domain \( \Upsilon \). This function is differentiable with respect to \( \Gamma \). We can choose an upper bound for the magnitude of each parameter to generate a bounded admissible set. By including the boundary values, \( \Upsilon \) is a compact set thus guaranteeing that the algorithm converges to a set of fixed points. A necessary modification in Algorithm 1 is to use a mapping to \( \Upsilon \), when at least one parameter goes out of \( \Upsilon \), and comparing to the previous solution (to make sure that we do not increase the objective by the parameter update). A simple mapping operator is the thresholding operator, where it chooses the closest admissible parameter.
Although the computation of the gradient of a parametric dictionary generated using \( g(t) \) is straightforward, we derive it in the Appendix B for completeness.

\section*{B. Simulations results}

We study the proposed dictionary design method using the Gammatone dictionary discussed in IV. We first investigate the characteristics of the dictionaries throughout the design iterations. The stability of the algorithm is demonstrated by showing the reduction of the objective function. In the second part of this subsection, we compare the performance of the initial and the optimized dictionaries in terms of sparse approximation and exact sparse recovery. Gammatone type dictionaries have been proposed for sparse approximation of audio and we choose our examples accordingly. In all the simulations we choose two times overcomplete dictionaries and window size 1024.

1) Algorithm Evaluation: In this part, we evaluate the given algorithm in three different areas. In the first step we show that the algorithm reduces, (or at least keep the same) the objective \( (3) \) in each iteration. The parameter \( B \), defined after (8), is the bandwidth of the audio filterbank at the center frequency \( f_c \). We use the fixed values \( n = 4, Q = 9.26449, b_{\text{min}} = 24.7 \), as they have been suggested in [48] and [49], and \( b = 0.65 \). To generate the initial dictionary, we sample \( f_c \) and \( t_c \).

We use the method introduced in [50] to generate the filter bank. In this method an extra parameter \( \delta \), called step factor, is introduced to indicate the amount of frequency overlap. In this framework the \( k^{th} \) time center is found using,

\[
t_{lc} = t_p + \sigma (l - 1) t_p .
\]

\( \sigma \) is set to 0.75 in our simulations. We draw the readers attention to the point that \( t_{lc} \) is implicitly a function of \( f_{kc} \). We therefore generate a set of \( \{ f_{kc} \}_{k \in \mathcal{X}} \) and for each generated atom using \( f_{kc} \) and \( t_c = 0 \), we make a set of time-shifted versions using \( \{ t_{lc} \}_{l \in \mathcal{L}} \).

To generate a dictionary of \( g_{\gamma_i}(t) \), we window it to a size equal to the signal length \( d \) and make it periodic such that one period is selected as an atom using the following formula,

\[
d_{\gamma_i,j} = \begin{cases} g_{\gamma_i,j+d) } & 1 \leq j < j_{c_i} \\ g_{\gamma_i,j) } & j_{c_i} \leq j \leq d , \end{cases}
\]

where \( j_{c_i} = \lfloor t_{c_i} f_{\text{samp}} \rfloor \). As the proposed algorithm is a relaxed version of the alternating minimization, the relaxation
parameters \( \{\alpha_k\} \) should be selected. We choose a simple sequence of \( \{\alpha_k\} \) using \( \alpha_k = \alpha \) for all \( k \) and a fixed \( \alpha \) in all simulations. A more complicated sequence might improve the performance of Algorithm 1. However we have not present this here. Here, we intend to show that the designed dictionary is superior to the initial dictionary in practice, even with a simple \( \{\alpha_k\} \). In the first experiment we want to investigate the effect of \( \alpha \). We have plotted the objective function (3) using selected \( \alpha \)'s, in Fig. 3. As we expect, simulations show reduction of the proposed objectives in each iteration. It is also demonstrated that if \( \alpha \) is small, the algorithm converges very slowly. Although using a large \( \alpha \) is desireable for a fast convergence, the solution is not as good as the solution found by using a medium range \( \alpha \). For other simulations we use \( \alpha = 0.5 \) to find a good solution after an acceptable number of iterations.

The proposed algorithm searches for an equiangular tight frame. Therefore one way to show the performance of the proposed algorithm is to compare the singular values (SV) of the designed dictionary and the tight frame. A tight frame in \( \mathbb{R}^{d \times N} \) has \( d \) non-zero SV equal to \( \sqrt{N/d} \). We have plotted the sorted SV's of the dictionaries at selected iterations in Fig. 4. It can be seen that the SV's of the designed dictionary get closer to the SV's of the tight frame at each iteration.

Given that the algorithm is based on distances in the Gram matrix domain, another way to evaluate the algorithm is to show the Gram matrix of the dictionary. We have plotted the \( \ell_2 \) norm of each row of the Gram matrix, for the window size 1024, in Fig. 5. The Gram matrix of the original dictionary and the designed dictionary, after 100 iterations, are shown in the left and right windows respectively. We have shown the \( \ell_2 \) norm of a possible ETF with a dashed line as reference. It can be seen that the Gram matrix of the designed dictionary is closer to the desired Gram matrix. Another observation in Fig. 5 is that the atoms with high total cross-correlations, indicated by the peaks, are adapted.

This parametric dictionary is attempting to tile the time-frequency plane. An ETF is a frame having the minimum total correlation between atoms but it may not be localized in the time-frequency plane. A dictionary which is simultaneously ETF, or close to being an ETF, and localized in time and frequency, tiles time-frequency plane more uniformly. To demonstrate this, we choose the Wigner-Ville (WV) time-frequency representation of the atoms. We show the contour plot of the atoms in the time-frequency plane using a similar method used in [51]. Fig. 6 and 7 show the time-frequency planes found for the original and designed atoms, respectively. Although the algorithm attempts to minimize \( \mu \) by changing

![Fig. 6. Wigner-Ville contour plots of the original Gammatone atoms. The WV contour of each atom is calculated at 0.7 times its peak.](image)

![Fig. 7. Wigner-Ville contour plots of the learned Gammatone atoms. The contours are calculated similar to Fig. 6.](image)
the structure of the dictionary, the locations which are not covered by the high energy part of any atom demonstrate its local minimum convergence. It also shows a potential for a more efficient update operator than the gradient descent in Algorithm 2. There exists a shift-invariance structure, with different step size for each frequency band, in the initial parametric dictionary, which disappears in the designed dictionary. If the time-shift is one of the parameters in the parametric dictionary, such a structure can then be preserved. Such a parametric dictionary is not column separable. Designing a structured parametric dictionary, e.g. shift-invariant dictionary, is left for a future work.

The parameter set $\gamma$ is selected intuitively in this experiment. To show the contribution of each parameter in the dictionary design, we show the initial and the final values of the parameters in Fig.8. The scatter-plots of $t_c$ and $f_c$ are shown in part (a) and (b). Note $f_c$ has not changed significantly by the dictionary design, so it could be kept fixed to reduce the computational cost. This simulation is also initialized with some fixed values for $n$ and $b$. The final values of these two parameters are shown in Fig.8.c and Fig.8.d respectively. These plots show significant changes in the values of $n$ and $b$, which demonstrates the importance of correct selection of $n$ and $b$ in each Gammatone atom.

2) Exact sparse recovery and sparse approximation: In this part we demonstrate the advantages of the parametric dictionary design in terms of exact sparse recovery [25] and sparse approximation. The exact recovery condition (ERC) [25] is studied in a worst-case setting. In this setting when a dictionary satisfies ERC, any k-sparse representation can exactly be recovered using (O)MP or BP. In practical applications, an average case analysis is more relevant [52], especially when the probability of the failure is very low. Here, by an experiment, it has been shown that the proposed algorithm improves the average exact recovery. We synthetically generate the sparse coefficient vectors, with different sparsity, and plot the percentages of the exact recovery for those sparse vectors. The location of the non-zero coefficients are selected uniformly at random and the PDF of the magnitudes are selected to be Gaussian with zero mean. The matching pursuit algorithm is used to find the sparse approximation. The rate of exact support recovery is calculated by the ratio of the number of correctly found non-zero coefficient index sets to the number of cases in which at least one location of the zero coefficient is set to non-zero. We run the simulations 1000 times. We have shown this ratio as the percentage of exact recovery in Fig. 9. It is clear that the design method has improved the exact recovery ratio.

For sparse approximation applications, we are more interested to have a dictionary that, if it fails to satisfy exact recovery [25], still gives a sparse approximation for a given class of signals. Therefore as the second experiment, we compare the decay rates of the residual error when the MP is used for sparse approximation [27]. We use an audio signal taken from more than 8 hours recorded from BBC Radio 3, which mostly plays classical music. We first down-sample by a factor of 4 and sum the stereo channels to make a mono signal with 12K samples per second. We use the original Gammatone and the parametric designed dictionaries to approximate 100 randomly selected blocks, each with the length of 1024 samples. The average decay rate of the residual errors, in logarithmic scales, are shown in Fig. 10. This rate directly influences the performance of sparse approximation methods. That is, we can better approximate the signal with fewer coefficients using a high residual error decay rate dictionary. In Fig. 10, although the curves start with the same slope, after a few iterations, here 10, the designed dictionary shows a clear advantage.

V. CONCLUSION

The sparse approximation methods successfully approximate a class of signals with a set of sparse coefficient vectors,
A. Definition of a surrogate optimization problem

The objective function in (3) depends on two variables, which makes the convergence analysis more difficult, if we want to use the continuity of the objective in the analysis. Here we define a surrogate objective for (3), which has a single variable, to show the convergence of Algorithm 1 to a set of fixed points. Let $\Gamma^* \in \Upsilon$ and $G^* \in \Lambda^N$ be a solution pair of (3) and $G^*_t = D^*_F G^*_r$. Then $G^* = \mathcal{P}_{\Lambda^N} G^*_r$, which suggests the optimization problem (3) can be replaced by the following problem based on $\Gamma$, as the only parameter,

$$\min_{\Gamma \in \Upsilon} \Phi_S(\Gamma),$$

$$\Phi_S(\Gamma) = \| G_T - \mathcal{P}_{\Lambda^N} G_T \|^F$$

$$= \left( \sum_{i \neq j} (|g_{ij} - \mu_c|)^2 + \sum_{i=j} (g_{ij} - 1)^2 \right)^{1/2},$$

(11)

where $|g_{ij}|$ is the absolute value of the $(i, j)$ element of $G_T = D^*_F D_F$. The problems (3) and (11) share common solutions. Therefore one can optimize (11) to find the solution(s) of (3). Although the surrogate objective is a continuous function of $\Gamma$ ($\Phi_S \in \text{class } \mathcal{C}^0$), a difficulty with the optimization of the surrogate objective directly is that it is non-differentiable. We only use the surrogate optimization problem to show the convergence of the proposed algorithm.

B. Convergence analysis of Algorithm 1 using the surrogate optimization problem

We now show that the proposed algorithm reduces the surrogate objective at each parameter update, using the following proposition.

Proposition 1: Let $G_{R_{k}} = D^*_F D_F$ be the Gram matrix of the dictionary at $k^{th}$ iteration. The Algorithm 1 reduces, or keeps the same, $\| G_{R_{k}} - \mathcal{P}_{\Lambda^N} G_{R_{k}} \|^F$ in each update of the parameters ($\Gamma_k \rightarrow \Gamma_{k+1}$), where $\mathcal{P}_{\Lambda^N}$ is the operator of orthogonal projection onto $\Lambda^N$.

Proof: Let $G_{P_{k+1}}$ be an abbreviation for $\mathcal{P}_{\Lambda^N} G_{R_{k}}$, which is found by using (4). Using the parameter update step (line 6) and the fact that $G_{R_{k+1}} = \alpha_k G_{P_{k+1}} + (1 - \alpha_k) G_{R_{k}}$, we have

$$\alpha_k \| G_{R_{k}} - G_{P_{k+1}} \|^F = \| G_{R_{k}} - G_{R_{k+1}} \|^F \geq \| G_{R_{k+1}} - G_{R_{k}} \|^F$$

$$= \| G_{R_{k}} - G_{R_{k+1}} \|^F \geq \| G_{R_{k+1}} - G_{P_{k+1}} - (1 - \alpha_k) G_{R_{k}} \|^F$$

$$= \| (G_{R_{k+1}} - G_{P_{k+1}}) - (1 - \alpha_k) (G_{R_{k}} - G_{P_{k+1}}) \|^F$$

$$\geq \| G_{R_{k+1}} - G_{P_{k+1}} \|^F - (1 - \alpha_k) \| G_{R_{k}} - G_{P_{k+1}} \|^F,$$

where we used the triangular inequality to derive the last inequality. This provides us the following inequalities,

$$\| G_{R_{k}} - \mathcal{P}_{\Lambda^N} G_{R_{k}} \|^F \geq \| G_{R_{k+1}} - \mathcal{P}_{\Lambda^N} G_{R_{k}} \|^F$$

$$\geq \| G_{R_{k+1}} - \mathcal{P}_{\Lambda^N} G_{R_{k+1}} \|^F,$$

(12)

where the last inequality is easily derived by using the definition of the projection in Hilbert space.

Prop. 1, with the facts that the objective is lower bounded by zero and there exists no continuum of fixed points, guarantees stability of Algorithm 1, due to Lyapunov's second theorem.

Let class $\mathcal{C}^1$ consist of all continuously differentiable functions. The following two Lemmata are needed to show the convergence of Algorithm 1 to a set of fixed points.
Lemma 1: Let $D_r : \mathcal{Y} \rightarrow \mathbb{R}^{d \times N} \in \text{class } C^1$ and $\mathcal{Y}$ be compact. The epigraph of the objective (11) at an admissible $\Gamma_0$ is compact.

Proof: When the parametric dictionary $D_r$ is differentiable on $\mathcal{Y}$, the objective function in (11) is continuous. The continuity of the surrogate objective function and the compactness of $\mathcal{Y}$ prove the compactness of $\text{epi } \Phi_S$ at an admissible point $\Gamma_0$.[40]

Due to the Bolzano-Weierstrass theorem, Algorithm 1 has a non-empty set of accumulation points. We now reformulate Lemma 1 in [46] for a more general (including asymptotically non-regular) sequence. Although the proof is the same, the set of accumulation points can be dis-connected, when the sequence is not asymptotically regular.

Lemma 2: Let $\{\Gamma_n\}_{n \in \mathbb{N}}$ be an infinite sequence in a compact set $\Sigma$ and $T$ be the set of its accumulation points then, $\forall \epsilon > 0, \exists N \in \mathbb{N}$ such that for all $n > N, \exists \Gamma^\dagger \in T, \|\Gamma_n - \Gamma^\dagger\|_F < \epsilon$.

Proof: Let $S$ be an $\epsilon$-neighborhood of $T$ and $S_c$ be its complement in $\Sigma, \Sigma$ is compact, thus $S_c$ is also compact. Because $S$ is a neighborhood of $T$, there is no accumulation point $\Gamma$ in $S_c$. If $\Gamma_n \rightarrow \Gamma$ in $S_c$ has infinite many points in $S_c$, then it has a converging subsequence and at least one accumulation point in $S_c$. This contradicts the fact that there is no accumulation point in $S_c$. Therefore $\exists N : \Gamma_n \in S, \forall n > N$. On the other hand the fact that $S$ being an $\epsilon$-neighborhood implies that for all $n > N, \exists \Gamma^\dagger \in T, \|\Gamma_n - \Gamma^\dagger\|_F < \epsilon$.[1]

Theorem 3: Let $D_r : \mathcal{Y} \rightarrow \mathbb{R}^{d \times N} \in \text{class } C^1$. The Algorithm 1 converges to a set of fixed points by starting from $\Gamma_0 \in \mathcal{Y}$, where $\mathcal{Y}$ is a compact set.

Proof: Due to Lemma 1 the epigraph of the surrogate objective at $\Gamma_0$ (epi $\Phi_S(\Gamma_0)$) is compact. The Proposition 1 shows that the sequence $\{\Gamma_n\}_{n \in \mathbb{N}}$ is in epi $\Phi_S(\Gamma_0)$. The convergence of the algorithm to a non-empty set of accumulation points is guaranteed using Lemma 2. Line 6 of Algorithm 1 prevents the existence of a continuum of accumulation points. Therefore the accumulation points are fixed points.

APPENDIX B
GRADIENT OF THE GAMMATONE DICTIONARY

We calculate the gradient of the parametric Gammatone dictionary with the generative function (8) in this appendix. Let $D_r \in \mathbb{R}^{d \times N}$ and $\Gamma \in \mathbb{R}^{4 \times N}$. The $i$th column of $D_r$ is a function of the $i$th column of $\Gamma$, $d_{ri}$. The rank of $\nabla_c D_r$ is 4 and we represent it by a tensor in $\mathbb{R}^{4 \times d \times N}$. Each sub-matrix of this tensor (fixing the third index) is the gradient of the corresponding atom in $D_r$. Therefore we only need to calculate the gradient of $d_{ri}$ based on $\gamma_i$. Because $d_{ri}$ is calculated using (10), we only need to derive a formula for the gradients of $g_r(t)$ based on $t_c, f_c, n$ and $b$, followed by sampling $t$.

$$
\frac{\partial g_r}{\partial t} = -a((n - 1)t^{n-2} + 2n f t^n + 2n b f^{n-1} + 2n b f^{n-1} - 2n f^{n-1})
$$

where $a = t_s \ln(t_s) - 2 \pi n r(t_s) c^{2n} b f^{n-1}$.

A sequence $\{a_k\}_{k \in \mathbb{N}}$ in a normed space is called asymptotically regular when $\lim_{k \to \infty} \|a_k - a_{k-1}\| = 0$.

REFERENCES


Mehrdad Yaghoobi (S’98-M’09) received the B.Sc. and M.Sc. in electrical and biomedical engineering in 1999 and 2002 from the University of Tehran and Sharif University of Technology, respectively. He started his PhD at Queen Mary University of London in December 2005, before he moved to the University of Edinburgh to accompany his supervisor in April 2006. He is now pursuing the PhD degree in the Institute for Digital Communications (IDCom) at the University of Edinburgh. His current research interests include sparse approximation, dictionary selection, compressed sensing and audio modelling/coding.

Laurent Daudet (M’04) received the M.S. degree in statistical and nonlinear physics from the cole Normale Superieure, Paris, France, in 1997 and the Ph.D. degree in mathematical modeling from the Universite de Provence, Marseille, France, in audio signal representations, in 2000. In 2001 and 2002, he was a Marie Curie Post-doctoral Fellow with the Centre for Digital Music at Queen Mary, University of London, London, U.K. Since 2002, he has been working as an Assistant Professor at the UPMC Univ. Paris 06, Paris, France, where he joined the Laboratoire d’Accoustique Musicale, now part of Institut Jean Le Rond d’Alembert. His research interests include audio coding, time-frequency and time-scale transforms, and sparse representations for audio.

Mike E. Davies (M’00) received the B.A. (Hons.) degree in engineering from Cambridge University, Cambridge, U.K., in 1989 and the Ph.D. degree in nonlinear dynamics and signal processing from University College London, London (UCL), U.K., in 1993. Mike Davies was awarded a Royal Society Research Fellowship in 1993 and was an Associate Editor for IEEE Transactions in Speech, Language and Audio Processing, 2003-2007.

He currently holds the Jeffrey Collins SHEFC funded chair in Signal and Image Processing at the University of Edinburgh. His current research interests include: sparse approximation, compressed sensing and their applications.