A Shift Tolerant Dictionary Training Method

B. Vikrham Gowreesunker, Ahmed H. Tewfik

A Shift Tolerant Dictionary Training Method

B. Vikrham Gowreesunker
University of Minnesota
Department of Electrical and Computer Engineering
200 Union Street SE.
Minneapolis, MN 55455
Email: gowr0001@umn.edu

Ahmed H. Tewfik
University of Minnesota
Department of Electrical and Computer Engineering
200 Union Street SE.
Minneapolis, MN 55455
Email: tewfik@umn.edu

***Abstract*—Traditional dictionary learning method work by vectorizing long signals, and training on the frames of the data, thereby restricting the learning to time-localized atoms. We study a shift-tolerant approach to learning dictionaries, whereby the features are learned by training on shifted versions of the signal of interest. We propose an optimized Subspace Clustering learning method to accommodate the larger training set for shift-tolerant training. We illustrate up to 50% improvement in sparsity on training data for the Subspace Clustering method, and the K-SVD method [1] with only a few integer shifts. We demonstrate improvement in sparsity for data outside the training set, and show that the improved sparsity translates into improved source separation of instantaneous audio mixtures.**

I. Introduction

In recent years, there has been a renewed interest in the sparse representation of signals with overcomplete dictionaries because of their potential for emerging applications like blind source separation and compressed sensing. There are two research areas that have been particular active, namely the designing of algorithms for extracting sparse representation from a given overcomplete dictionary and the design of the dictionaries, which is the topic of interest for this paper. There are a number of approaches to designing dictionaries, but the gold standard is the K-SVD algorithm [1], which we have successfully used in the context of blind source separation (BSS) of instantaneous audio mixtures [2], [3]. However, the K-SVD, like most of its dictionary design counterpart typically does not generalized well to data beyond what it was trained on. Furthermore, in the context of speech or audio, where typical frame size vary from 256 to as high as 1024, the K-SVD becomes very computationally expensive. For an application like BSS, where one might have limited reference signal, and potentially large amount of data to process, it is highly desirable to have dictionaries that are generalizable beyond the training set and computationally more tractable than the K-SVD. This is our motivation in seeking a new approach to training dictionaries.

Given a long signal, traditional dictionary training methods will first vectorize the signal by windowing and framing, which makes the learnt dictionary strongly dependent on the time-localized frames. In this paper, we study training the dictionaries on shifted versions of the signal to make the dictionary more shift-tolerant. We find that this method produces sparser representations and better generalizability than traditional training methods, with only a few shifts. Although, this method can be applied to most training methods, it is better suited for those with low complexity requirements. We propose an optimized version of the recently introduced Subspace Clustering dictionary training method [4] which combines the low complexity of the subspace method, and an optimization step to accommodate the large volume of training data that results from multiple shifts. The new Optimized Subspace Method can produce dictionaries of comparable size and sparsity as the K-SVD at a fraction of the computation cost. Overall, we find that using as few as 5 shifts can translate into big improvement in sparsity and source separation performance.

In section II, we describe the shift-tolerant training method. In section III, we first give a summary of the Subspace Clustering method and the Orthogonal Least Square(OLS) sparse decomposition algorithm, and then give a detailed description of the proposed Optimized Subspace Clustering method. In section IV, we give a brief description of the K-SVD, and in section V, we describe how the dictionaries were designed and demonstrate the benefits of shift-tolerant training in terms of sparsity improvements and source separation performance. In section VI and VII, we discuss some aspect of the training that were not explored in this paper and finish with some final remarks.

II. Shift-Tolerant training

The goal of dictionary training methods is to identify and extract inherent structures that make up our signals. Given a signal $y(n)$, it is common practice to window and vectorize it into frames of size N using an overlap factor (which is assumed to be $N/2$ here), before training. However, these structures are not phased locked and due to the linearity of the windowing and framing commonly employed in audio signal processing, some of these patterns could be missed by the training method. We propose training on shifted versions of the training set to improve the generalizability of the dictionary. We define a signal shifted by τ samples as $y(n-\tau)$ and similarly the matrix, $Y(n-\tau)$, as the vectorized frames of signal $y(n-\tau)$. For shifts in the range of $\tau_{min} \leq \tau \leq \tau_{max}$, our training set is defined as $\{Y(n-\tau)\}_{\tau=\tau_{min}}^{\tau_{max}}$.

Although a simple idea, this procedure raises a number of questions. First, we wish to find if there are any benefits to

this method compared to traditional training methods without delays. Second, we are concerned with the practical implications to training on a rather large training set. Finally, we wish to explore the trade-offs between potential performance improvement and the computational cost involved. We will explore into detail the issue of large training set in section III where we introduce an optimized dictionary training method based on subspace clustering which is tailored to accommodate the large volume of data. The relatively low computational complexity of the subspace training makes the shift-tolerant training an attractive procedure. We will also discuss the case of applying the shift-tolerant training to the K-SVD dictionary design method, where computational cost makes it more restrictive. Overall, we find that training on relatively few shifts (as low as 5 shifts) has significant advantages.

III. Dictionary Training Using Subspace Clustering

Given a set of observation vectors, a dictionary learning algorithm attempts to infer the underlying features that constitutes the building block of the observation. Each feature is referred to as a dictionary atom and the collection of features, is known as a dictionary. There exists a number of dictionary learning methods, each of which leverages some assumption regarding how the data was generated. The concept of learning dictionaries by subspace clustering, that was recently introduced in [4], is build upon the assumption that if a set of observation data was generated from the same set of dictionary atoms, then all of these observations should lie on the subspace spanned by those atoms. In sections III-A and III-B, we briefly summarize the essence of the previously introduced algorithm. In section III-C, we describe a pruning and subspace optimization step that is necessary for large data set such as that generated by training on shifted versions of our signals.

A. Subspace Clustering Dictionary Learning Method

Given a set of observation vectors, strip away observations with negligible power, the threshold of which is dependent on the application. Assuming there is a holdover of K observation vectors each of dimension $T \times 1$, we normalize each vector to unit l_2 norm and the resulting data is our training set, $Y = \{y_i\}_{i=1}^{K}$. We initialize the dictionary as D^0 as the data Y. The algorithm has two stages, first we identify a subspace from the training data, and second we find all the training data that lives on this subspace and remove them from Y before looking for the next subspace. The algorithm can be describe as follows:

1) Initialize Algorithm
   $i = 0$, $D^0 = Y$,
2) Sparse coding
   a) $i = i+1$ Choose the i^{th} vector, y_i, from the training set, and remove it from the dictionary
   b) Find a representation of y_i in terms of D^i using a sparse decomposition algorithm, where $D^i = D \otimes y_i$
   c) Let S_i be the set of λ_i vectors that represents y_i
   d) Find the SVD of S_i such that $U\Sigma V^T = S_i$
   e) We define the subspace, A_i, as the first λ_i columns of U
3) Create data clusters
   a) Find the projection of y_j onto the subspace of A_i, where $j \neq i$
   b) Assign all y_j that lies within a certain error to subspace A_i to the same cluster and remove them from training set, Y
4) The learned dictionary is A = the collection of subspaces $\{A_i\}_{i=1}^{M}$, where M is the number of subspaces

The operator $A \otimes B$ returns the vectors that not common between set A and B. The best sparse decomposition algorithm for finding the orthogonal subspace is the Orthogonal Least Square(OLS) method, originally introduced in the context of regression [5]. This algorithm is briefly described next.

B. Orthogonal Least Square

Our sparse decomposition algorithm of choice for sparse coding it the Orthogonal Least Square (OLS) algorithm, which is an iteratively greedy pursuit method, similar to the Matching Pursuit (MP) and Orthogonal Matching Pursuit (OMP) [6], with a key difference in the directional update procedure. Although more computationally demanding than MP or OMP, the OLS produces a much sparser representation and works best for finding the subspaces. Let vector y, be a vector of dimension $N \times 1$, and let D be a known overcomplete dictionary. A sparse decomposition algorithm seeks to approximate y in terms of a minimum number of columns of D. The OLS procedure involves two steps, the first involves finding the most correlated atom from the dictionary, and the second involves a dictionary decorrelation step where the atoms that were not selected are decorrelated from previously chosen atoms. The algorithm is described below:

1) Initialization
   $m = 0, r^0 = y, \hat{y}^0 = 0, \Omega^0 = \{1, 2, 3, ..., K\}, \Gamma^0 = \oslash$, $W = \{d_i\}_{i \in \Omega^0}$
2) While stopping criteria are not met
   a) $m = m + 1$
   b) Pick atom with maximum correlation to residual $i^m = \arg_j \max| < w_j, r^{m-1} > |$, $j \in \Omega^{m-1}$
   c) Remove atom index from set Ω and add to set Γ $\Omega^m = \Omega^{m-1} \otimes i^m$; $\Gamma^m = \Gamma^{m-1} \cup i^m$
   d) Update residual and approximation $r^m = r^{m-1} - < w_{i^m}, r^{m-1} > w_{i^m}$; $\hat{y}^m = \hat{y}^{m-1} + < w_{i^m}, r^{m-1} > w_{i^m}$
   e) Decorrelate remaining dictionary atoms from w_{i^m} such that for all $j \in \Omega^m$, $w_j = w_j - < w_j, w_{i^m} > w_{i^m}$; $w_j = w_j / norm(w_j)$
   f) Check stopping criteria

C. Subspace Optimization

One downside to seeking subspaces is the presence of "spurious" or redundant subspaces, the frequency of which

increases with the signal dimension and the size of the training set. There are a few practical modifications that can be made to the above algorithm to reduce the production of these redundant subspace, but ultimately a pruning procedure is needed for best results. In this section we describe a post-processing method to prune the subspaces such that the most relevant ones are retained and further optimized to better fit our data. This procedure also allows us to keep the dictionary size from growing beyond a practical limit.

Starting with M subspaces $\{A_i\}_{i=1}^{M}$, and a set of data clusters, $\{Y_i\}_{i=1}^{M}$, such that the points in Y_i are closest to subspace A_i than any other subspace, we seek at most M_{max} subspaces that can represent our data. We define the distance metric, $d(y, A_i)$ of a point y to the plane A_i as the l_2 norm of the error between y and its projection onto the subspace A_i, i.e. $d(y, A_i) = \|(I_N - A_i A_i^T)y\|_2$, where I_N is an $N \times N$ identity matrix. Below we outline a iterative optimization method, much like the generalized Lyold algorithm, to find the best fit M_{max} subspaces to our data.

1) Initialization
   a) Iteration, $k = 0$.
   b) Retain M_{max} of the subspaces with the largest cluster size.
   c) Cluster data from discarded subspaces
   d) Calculate total error at k^{th} iteration, error(k) = $\sum_{i=1}^{M_{max}} \sum_{y_j \in Y_i} d(y_j, A_i)$
2) Optimization Step
   While abs(error(k+1) - error(k)) $\geq$ threshold
   a) k = k+1
   b) Let $\{A_i\}_{i=1}^{M_{max}}$ be the new set of subspaces.
   c) Repartition the data into M_{max} clusters such that Y = $\{Y_i\}_{i=1}^{M_{max}}$, and the points in Y_i is closest to subspace A_i
   d) For i = 1 to M_{max}.
      i) Find the SVD of Y_i such that $U\Sigma V^T = Y_i$
      ii) Let λ_i = min(cluster size of Y_i,size of A_i)
      iii) Update subspace, A_i, as the first λ_i columns of U
   e) Compute error(k+1)
3) Return new subspaces and total error

IV. The K-SVD Algorithm

The K-SVD algorithm is a generalization of the K-Means method that is used to design optimal codebooks for vector quantization (VQ). The algorithm tries to solve the following problem:

$$min_{D,X}\{\|Y - DX\|_F^2\} \text{ s.t } \forall i \ \|x_i\|_0 \leq T_0, \qquad (1)$$

where T_0 is a hard constraint on the maximum number of non-zero elements allowed, and D is the learned dictionary. This is solved iteratively in two stages, involving a sparse coding stage using a pursuit method, and a update stage where the dictionary atoms and the coefficients are sequentially updated using SVD operations.

V. Evaluating Performance

There are a number of ways to evaluate the performance of dictionaries used. However, for our purposes, we restrict the evaluation to two specific criteria, namely the sparsity of the signal and its impact on source separation. Given a dictionary D, and a vector y, such that $y = Dx$, the strict sparsity measure of the representation, is the l_0 norm of the coefficient vector x, i.e, the number of non-zero coefficients in x. We have previously demonstrated that dictionaries learned from data using methods like the K-SVD yields sparse representations that translate into improved separation performance when compared to standard dictionaries such as the Cosine Packet [2]. We first describe the types of dictionaries that were designed, followed by experiments to evaluate sparsity and source separation performance.

A. Dictionary Design

Starting with three 10 seconds male speech, from the Signal Separation Evaluation Campaign (SiSEC 2008) database [7], each sampled at 16 KHz, we split each track into a training set and a testing set. The training set consists of the first 5 seconds of each track, and the testing set was the last 5 seconds of the tracks. The dictionaries were learned from the training set and evaluated on both the training set and the testing set. For the subspace method, we have two types of dictionaries. The first is the dictionary obtained with a straight-forward subspace clustering method, which we label as Subspace Method. The second, labeled as Optimized Subspace Method, is the dictionary that has been optimized as described above by trimming the number of subspaces and doing a Lyold-like optimization. The K-SVD dictionary was build using the package [8]. We used frames of size N = 256, and a Hanning window with 50% overlap. For N = 512 or 1024, the K-SVD algorithm becomes prohibitively slow to run and proper comparison were not be made.

We denote the training signal, as $y(n-\tau)$, where τ is the integer delay in samples. After being windowed and vectorized, the training signal is in the matrix form denoted by $Y(n-\tau)$. We present two sets of results, the first where $\tau = 0$, referred to as the "No delay" version, which is the traditional training method. For the second, we used shifts up to a maximum delay of 5 samples, which is denoted as "Max delay = 5" and in this case, the training set becomes $Y = \{Y(n-\tau)\}_{\tau=0}^{\tau=5}$. It should be obvious that the size of the training set increases linearly with the number of delays considered and this has direct cost in computational time for the both the K-SVD and the subspace clustering method. For the K-SVD, the penalty is large because of the iterative column-by-column optimization which requires an SVD operation per dictionary column per iteration. Furthermore this SVD becomes costlier with larger data sets. In the case of the subspace optimization, there are as many SVD operation as subspaces found and the penalty here is far smaller than for the K-SVD. However, the real bottleneck of the subspace method that stems from using the OLS which can be computationally demanding with data sets larger than $10,000$ elements. For the "No delay" case, the

subspace method takes about 3 to 4 minutes for 15 seconds of speech data, while 4 iterations of K-SVD can take up to 60 minutes. For the "Max Delay = 5", the subspace method takes up to 1 hours and the K-SVD up to 7 hours. Clearly, this approach is more restrictive for the K-SVD.

As mentioned previously, one issue specific to the subspace clustering, is the generation of redundant or spurious subspaces that can result in a rather large dictionary. This is tackled using the subspace optimization technique described in III-C, where we retained and optimized M_{max} subspaces. For the Subspace Optimization results below, we used a fixed bound on the maximum subspaces such that $M_{max} = (\tau_{max} + 1) \times \frac{N}{2}$. Although not necessarily the optimal number to pick, we found this to give the best tradeoffs between performance and dictionary size when compared to the K-SVD. There is obviously a computational penalty for this optimization, which is dependent on the number of SVD's per iteration and the number of iteration before convergence. From our experience, the algorithm converges in a few iterations and total cost of this step is at less than the time for the subspace clustering step. We have chosen to develop the subspace clustering and the optimization step independently but if combined, there are a few modifications that can be done to further speed up the process. In table I, we give a summary of the dictionary sizes.

TABLE I
COMPARISON OF THE SIZE OF THE DIFFERENT DICTIONARIES

Method	Max delay	dictionary size
Subspace	0	4053
Optimized Subspace	0	1097
KSVD	0	896
Subspace	5	11434
Optimized Subspace	5	3505
KSVD	5	4480

B. Evaluating Sparsity

To evaluate the sparsity of representation for the different dictionaries, we compare the average number of dictionary atoms it take to encode a frame. For evaluation, we used the same setup discussed above, where the first half of the signal, denoted as the training set, was used for training the dictionary, and the second half of the signal was denoted as the testing set. Given the vectorized matrix of the unshifted data, $Y(n)$, we normalize each column of $Y(n)$ to unit l_2 norm and encode each column within an error norm of 0.1 by using the OLS sparse decomposition method. In table II, we show the average sparsity for both the training set and the testing set. The sparsity is given as $(\mu + / - \sigma)$, where μ is the average number of dictionary atoms needed to code one frame of data, and σ is the standard deviation. We find that shift-tolerant training, even if it is for small shifts, results in compression of up to 50% on the training set, and almost 25% on the testing set for all the methods. It should also be noted, that the improvement is more pronounced for the subspace method.

TABLE II
SPARSITY COMPARISON. THE CONVENTION μ +/- σ MEANS THAT μ IS THE MEAN AND σ IS THE STANDARD DEVIATION PER FRAME

Method	Delay	Train Sparsity	Test Spasity
Subspace	0	10.8 +/- 10.5	32.5 +/- 31.2
Subspace	5	4.2 +/- 7.4	26.7 +/- 26.0
Optimized Subspace	0	10.3 +/- 9.2	41.3 +/- 37.2
Optimized Subspace	5	5.4 +/- 5.2	29.9 +/- 28.1
KSVD	0	10.4 +/- 24.0	41.8 +/- 41.3
KSVD	5	6.9 +/- 16.1	32.0 +/- 32.5

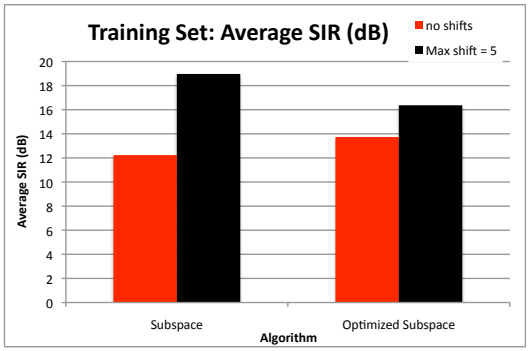

Fig. 1. Signal-to-Interference ratio for training data with frame size = 256

C. Evaluating impact on source separation

We also evaluated the dictionaries in the context of blind source separation (BSS) of instantaneous audio mixtures. The details of the algorithm are beyond the scope of this paper, and the readers are referred to [3] for the particular source separation method and [2] for a discussion of how sparsity of representation in a particular dictionary can improve separation performance. For our purposes, we compare the dictionaries using the same algorithm and the differences is directly correlated to the sparsity of the dictionaries. We used two performance criteria, the Signal-to-Interference Ratio (SIR) which is a measure of interference rejection, and the Signal-to-Artifact Ratio (SAR), which is a measure of the quality of the estimated signals. Both the SIR and the SAR were computed with the widely used BSS Eval toolbox as described in [9].

We consider two 10 seconds mixtures of the 3 male speech signals that were described in section V-A. The average input

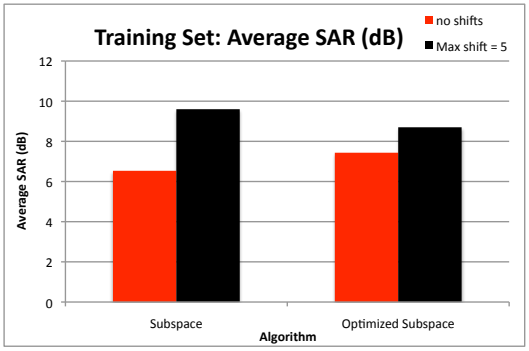

Fig. 2. Signal-to-Artifact ratio for training data with frame size = 256

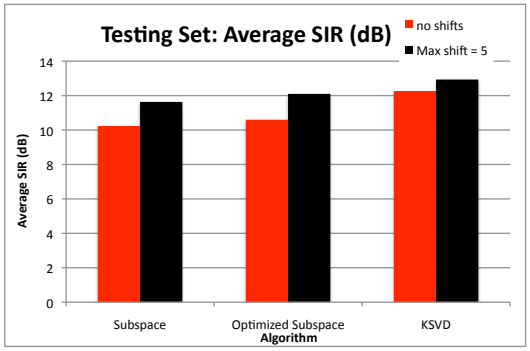

Fig. 3. Signal-to-Interference ratio for test data with frame size = 256

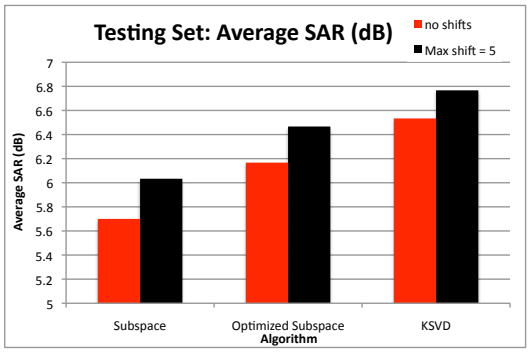

Fig. 4. Signal-to-Artifact ratio for test data with frame size = 256

SIR was -3.4dB. In the same spirit as before, we evaluate the BSS performance on the training set (first 5 seconds) and the testing set (the last 5 seconds). In figures 1 and 2, we compare the SIR and SAR,averaged for the 3 sources, for the Subspace and Optimized Subspace method in the cases of "No delay", and "Max Delay = 5". For both subspace methods, there is a significant improvement in interference rejection and quality of the estimated signals. There was very little improvement for the K-SVD on the training set. In figures 3 and 4, we show the average SIR and SAR for the testing set, which is more interesting as it pertains to the generalizability of the dictionary beyond the data that it was trained upon. Here, we see improvements in SIR and SAR for the Subspace Method, Optimized Subspace Method, and the KSVD. The improvements are smaller than for the training set but notable nevertheless since we trained on only a few shifts.

VI. Discussion

We presented results for a small number of shift so that we can compare the approach with the Subspace method and the K-SVD. From practical standpoint it is clear that shift-tolerant training is better suited for a relatively low complexity methods like the subspace approach and as such we focused more on the subspace method and how to customize it to fit large datasets. We found that some of the subspaces produced for different delays have some overlap, which would leave the door open for further leveraging this in customizing the algorithm for shift-tolerance. It is also worth noting the recent work of [10] who have formulated the a shift-invariant version of the K-SVD algorithm. Their method decompose the full signal, without framing, into patterns that can be shifted the length of the signal, and uses a K-SVD like iterative optimization to update the dictionary. The shift invariant method can be computationally more demanding than the K-SVD but could prove more generalizable. We had a limited time to experiment with this method, which looks like a promising upgrade to the K-SVD but unfortunately we did not have enough time to do a fair comparison with our method.

VII. Conclusion

In this paper, we introduced an Optimized Subspace Clustering method for dictionary design, that prunes and optimized the subspaces from the original Subspace Clustering method, achieving dictionary size comparable to the K-SVD and with improved sparsity for training and testing data. The main contribution of the paper was to demonstrate that significant performance improvements can be achieved when dictionaries are trained using the proposed shift-tolerance idea, which can make dictionary more generalizable. We demonstrated , improved sparsity and source separation performance with as few as 5 integer shifts. Although the proposed method works with most learning algorithm, it should be noted that it is best suited for lower complexity method such as the proposed Optimized Subspace method.

References

[1] M. Aharon, M. Elad, and A.M. Bruckstein, "The k-svd: An algorithm for designing of overcomplete dictionaries for sparse representation," *IEEE Trans. on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, November. 2006.

[2] B. Vikrham Gowreesunker and Ahmed H. Tewfik, "Two improved sparse decomposition methods for blind source separation," in *Independent Component Analysis and Signal Separation (ICA)*, London, UK, September 2007, vol. 4666, pp. 365–372.

[3] B. Vikrham Gowreesunker and Ahmed H. Tewfik, "Blind source separation using monochannel overcomplete dictionaries," in *IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, May 2008.

[4] B. Vikrham Gowreesunker and Ahmed H. Tewfik, "A novel subspace clustering method for dictionary design," in *Independent Component Analysis and Signal Separation (ICA)*, Paraty, Brazil, March 2009, vol. 5441.

[5] S. Chen, C.F. Cowan, and P.M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE transactions on neural networks*, vol. 2, no. 2, pp. 302–309, 1991.

[6] Stephan Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, 2nd edition edition, 1999.

[7] "Signal separation evaluation campaign," http://sisec.wiki.irisa.fr/.

[8] "K-svd matlab toolbox," http://www.cs.technion.ac.il/elad/software/.

[9] Vincent E., Fevotte C., and Gribonval R, "Performance measurement in blind audio source separation," *IEEE Trans. Audio, Speech and Language Processing*, vol. 14, no. 4, pp. 1462–1469, 2006.

[10] B. Mailhe, S. Lesage, R. Gribonval, and F. Bimbot, "Shift-invariant dictionary learning for sparse representations: extending k-svd," in *in Proc. 16th European Signal Processing Conference (EUSIPCO-2008)*, Lausanne, Switzerland, August 25-29 2008.