Sparse Approximation by Matching Pursuit using Shift-Invariant Dictionary

Karl Skretting and Kjersti Engan

University of Stavanger, 4036 Stavanger, Norway,
{karl.skretting,kjersti.engan}@uis.no

Abstract. Sparse approximation of signals using often redundant and learned data dependent dictionaries has been successfully used in many applications in signal and image processing the last couple of decades. Finding the optimal sparse approximation is in general an NP complete problem and many suboptimal solutions have been proposed: greedy methods like Matching Pursuit (MP) and relaxation methods like Lasso. Algorithms developed for special dictionary structures can often greatly improve the speed, and sometimes the quality, of sparse approximation. In this paper, we propose a new variant of MP using a Shift-Invariant Dictionary (SID) where the inherent dictionary structure is maximally exploited. The dictionary representation is simple, yet flexible, and equivalent to a general M channel synthesis FIR filter bank. Adapting the MP algorithm by using the SID structure gives a fast and compact sparse approximation algorithm with computational complexity of order $\mathcal{O}(N \log N)$. In addition, a method to improve the sparse approximation using orthogonal matching pursuit, or any other block-based sparse approximation algorithm, is described. The SID-MP algorithm is tested by implementing it in a compact and fast C code (Matlab mex-file), and excellent performance of the algorithm is demonstrated.

Keywords: Sparse Approximation, Matching Pursuit, Dictionary, Shift-Invariant

1 Introduction

Obtaining a sparse representation of a signal is an important part of many signal processing tasks. A typical way to do this is to use a dictionary of atoms, and to represent the signal as a linear sum of some of the available atoms. A sparse approximation of a signal $\mathbf{x}$ of size $N \times 1$, allowing a (small) error $\mathbf{r}$, can be written as

$$\hat{\mathbf{x}} = D\mathbf{w} = \sum_k w(k)\mathbf{d}_k, \quad \mathbf{x} = \hat{\mathbf{x}} + \mathbf{r} \tag{1}$$

where D is the $N \times K$ dictionary, $\mathbf{d}_k$ of size $N \times 1$ is atom number k, $w(k)$ is coefficient k, and $\mathbf{w}$ is the $K \times 1$ coefficient vector. For a sparse approximation, only a limited number, s, of the coefficients are non-zero.

Common predefined dictionaries are the Discrete Cosine Transform (DCT) basis, wavelet packages and Gabor bases, both orthogonal and overcomplete variants can be used. Advantages of these predefined dictionaries are: They can be implicitly stored, they are, in general, signal independent and thus can successfully be used for most classes of signals, and fast algorithms are available for common operations. The overlapping transforms, i.e. wavelets and filter banks, perform better than the block-based transforms, like DCT, on many common signals. This is because the structure in the overlapping transforms does not need the signal to be divided into (small) blocks and thus they avoid the blocking artifacts.

Learned dictionaries are an alternative to predefined dictionaries. Since they are learned, i.e. adapted to a certain class of signals given by a relevant set of training signals, they will usually give better sparse approximation results for signals belonging to the class. Many methods have been proposed for dictionary learning; K-SVD [2], ODL [14], MOD [8] [9] and RLS-DLA [23]. All these methods are block-based and the blocks should be rather small to limit the number of free variables in the dictionary; as seen from Eq. 1 the number of elements in D is NK.

Structured learned dictionaries try to combine the advantages of predefined dictionaries and general (all elements are free) learned dictionaries [22] [17]. Imposing a structure on the dictionary allows the atoms to be overlapping and the dictionary could be very large and still have a reasonable number of free variables. In addition, the structure can make dictionary learning computationally more tractable. Several variants of structured dictionaries are possible [1], [22], [9] and [3]. An example of a structured dictionary that has been shown to be useful is the Shift-Invariant Dictionary (SID) [18] [12] [20]. In this paper, we propose a flexible representation for the SID structure.

The sparse approximation step is an important part when both using and learning a dictionary. It is often the computationally most expensive part, thus its speed is very important. Popular approaches are the l_1-norm minimization methods like LARS/Lasso [7], and the greedy algorithms directly using the l_0-quasinorm. For the latter approach Matching Pursuit (MP) [15] and its many variants, like orthogonal matching pursuit [19], are much used. MP algorithms typically has computationally complexity of $\mathcal{O}(N^3)$ and this effectively limits the dictionary size. To overcome this restriction FFT-based algorithms tailored for shift-invariant dictionaries have been developed [10].

FFT-based algorithms work very well for regular dictionaries, i.e. Gabor, harmonic, chirp or DCT based atoms, but for short arbitrary waveforms time-domain algorithms should be expected to perform better. As an example the execution times for Matlab functions `fftfilt()` and `filter()` can be compared for different signal and filter lengths; the time domain approach is faster for filters shorter than 300 and a signal length of 10000 samples. For many learned shift-invariant dictionaries, the atoms will be arbitrary and of limited (short) length, and sparse approximation should benefit from using a well-designed and well-

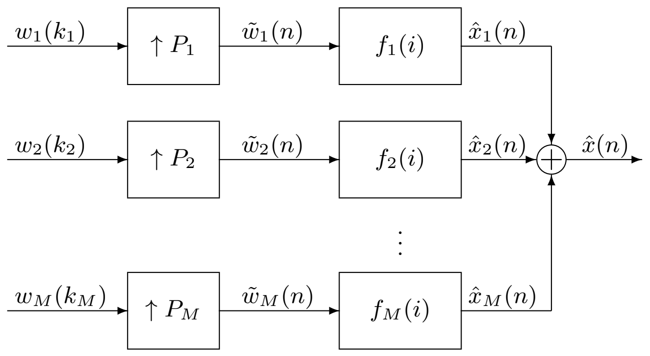

Fig. 1. A general synthesis filter bank of M channels (filters). Filter $\mathbf{f}_m$ has length Q_m and upsampling factor P_m, these values may be different for each filter. $\tilde{w}_m(n)$ is the upsampled coefficients for filter m. If each filter $\mathbf{f}_m$ here is given as the reversed (and shifted) atom $\mathbf{q}_m$ the synthesis equation can be written as in Eq. 5.

implemented time domain based algorithm. Therefore, this is what we present here.

The matching pursuit variant presented in this paper, denoted as SID-MP, is completely implemented in time domain but still has the same low computationally complexity of order $\mathcal{O}(N \log N)$ as the FFT-based algorithms, it uses a simple and flexible representation for the shift-invariant dictionary, and it allows the atoms to be shifted in steps larger than one. For short arbitrary shape atoms, it should be faster than FFT-based algorithms.

Orthogonal Matching Pursuit (OMP) finds better approximations than MP using the same number of non-zeros [4] and fast algorithms are available for small dictionaries. For large shift-invariant dictionaries (true) OMP is slow, but a low complexity approximate OMP variant has been presented by Mailhé et al. [11]. This variant is FFT-based and achieves results close to true OMP with execution times close to MP. We propose another OMP variant, denoted SID-OMP. Its main idea is to divide the long signal into blocks (segments) of moderate size that are processed by OMP or any other block-oriented sparse approximation algorithm. By doing several iterations, or using partial search [24] instead of OMP, better approximation than (true) OMP can be achieved. The SID structure is used in the proposed method, it has computationally complexity of order $\mathcal{O}(N)$ but the constant factor is large and depends on the segment size.

The organization of this paper is: Section 2 presents the representation of the SID structure. The proposed algorithm SID-MP is presented in Section 3 and the SID-OMP method in Section 4. Finally, in Section 5, it is shown that the proposed algorithms are fast and efficient.

2 The Shift-Invariant Dictionary

A Shift-Invariant Dictionary (SID) should be represented independently of the signal size, and it should be easy to extend it to match the (possibly large) signal length N. In this paper, we suggest to use a dictionary structure, denoted as SID structure, equivalent to the general synthesis filter bank in Fig. 1. A matrix representation as in Eq. 1 is equivalent to the M channels filter bank in Fig. 1 when each channel is represented by one submatrix D_m and the dictionary is a concatenation of these:

$$D = [D_1, D_2, \cdots, D_M], \tag{2}$$

One submatrix, D_m, contains one atom (the reversed synthesis filter) and its shifts (below the shift $P_m = 1$ is visualized):

$$D_m = \begin{bmatrix} q_m(1) & & & \\ q_m(2) & q_m(1) & & \\ \vdots & \vdots & \ddots & \\ q_m(Q_m) & q_m(Q_m-1) & \ddots & q_m(1) \\ & q_m(Q_m) & \ddots & q_m(2) \\ & & \ddots & \vdots \\ & & & q_m(Q_m) \end{bmatrix}. \tag{3}$$

Atom m is denoted as $\mathbf{q}_m$ with elements $q_m(i)$ for $i = 1, 2, \ldots, Q_m$. It has length Q_m and is shifted P_m positions down for each new column in D_m. Here it is assumed that each atom has unit norm and is real. Note that the support for all shifts of an atom is completely within the dictionary column (length N). There is neither circular extension nor mirroring at the ends. This means that the number of columns in submatrix D_m is given as:

$$K_m = \lceil (N - Q_m + 1)/P_m \rceil. \tag{4}$$

$\lceil \cdot \rceil$ denotes the ceiling function. The dictionary D has size $N \times K$ where $K = \sum_{m=1}^{M} K_m$. The overcomplete factor, i.e. the ratio K/N (as $N \to \infty$), is $\sum_{m=1}^{M} 1/P_m$. The SID structure is given by the number of atoms M, the atom lengths $\{Q_m\}_{m=1}^{M}$, their shifts $\{P_m\}_{m=1}^{M}$, and their $Q = \sum_{m=1}^{M} Q_m$ values which are collected into one vector $\mathbf{q}^{\mathbf{T}} = [\mathbf{q}_1^T, \mathbf{q}_2^T, \ldots \mathbf{q}_M^T]$. This structure for a shift-invariant dictionary is actually quite flexible, and many dictionaries fit quite well into this structure. It is a subclass of the flexible structure dictionary introduced in [3].

The sparse approximation equation can be divided into blocks as

$$\begin{aligned} \hat{\mathbf{x}} = D\mathbf{w} &= [D_1, \ldots, D_M] \begin{bmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_M \end{bmatrix} \\ &= D_1\mathbf{w}_1 + \cdots + D_M\mathbf{w}_M = \hat{\mathbf{x}}_1 + \cdots + \hat{\mathbf{x}}_M, \end{aligned} \tag{5}$$

```
Matching pursuit algorithm:

1    $\mathbf{w} = \mathrm{MP}(D, \mathbf{x}, s)$
2      $\mathbf{w} := \mathbf{0}$, $\mathbf{r} := \mathbf{x}$
3      while not Finished
4          $\mathbf{c} := D^T \cdot \mathbf{r}$
5          find $k : |c(k)| = \max_j |c(j)|$
6          $w(k) := w(k) + c(k)$
7          $\mathbf{r} := \mathbf{r} - c(k) \cdot \mathbf{d}_k$
8a         Finished := $(\|\mathbf{r}\| < \text{some Limit})$
8b         Finished := ($s$ non-zero entries in $\mathbf{w}$)
9      end
10     return
```

Fig. 2. The matching pursuit algorithm. The columns (atoms) of D should have unit norm. Input s is the target number of non-zero coefficients to select.

where the approximation is split into M terms, each corresponding to a submatrix of the dictionary. It is easily seen that this is an alternative representation of a general synthesis FIR filter bank [5] of M filters as in Fig. 1.

3 Matching Pursuit using SID

Matching Pursuit (MP), presented and analyzed in [15] and [6], is the simplest of the matching pursuit algorithms. An overview of the algorithm is given in Fig. 2. The MP algorithm is quite simple, but it is not very efficient. The loop may be executed many times ($\geq s$), sometimes selecting the same dictionary atom multiple times, and the calculation of the inner products (line 4) is demanding, where approximately NK multiplications and additions are done. As both K and the number of non-zero coefficients to select, s, are (usually) of the order $\mathcal{O}(N)$ this gives the computational complexity of MP as $\mathcal{O}(N^3)$. Using FFT and taking advantage of the dictionary structure, and all shift steps equal to one ($P_m = 1$), shift-invariant MP can be implemented by an algorithm of the order $\mathcal{O}(N \log N)$ [10]. Below, we present a simple time domain algorithm that exploits the SID structure, and allows different shift steps $P_m \geq 1$. It follows the steps of MP as seen in Fig. 2, but has the complexity order of $\mathcal{O}(N \log N)$ or $\mathcal{O}(MQN \log N)$ if the (constant) SID size is included.

The proposed algorithm is done in a straightforward way in time domain, but exploiting the SID structure maximally and thus making the inner loop as fast as possible. An outline of the algorithm is given in Fig. 3 and the similarity to MP in Fig. 2 is obvious. The input is the SID which is given by the number of atoms M, the atom lengths $\{Q_m\}_{m=1}^{M}$, their shifts $\{P_m\}_{m=1}^{M}$, and their Q values, the signal $\mathbf{x}$ (length N) and the target number of non-zeros s. In the analysis below the SID is assumed to be constant and independent of signal length N.

```
Matching pursuit using a shift-invariant dictionary:

1   $\mathbf{w}$ = SID-MP(sid,$\mathbf{x}$, $s$)
2     $\mathbf{w} := \mathbf{0}$,    $\mathbf{r} := \mathbf{x}$
3     $\mathbf{c} := D^T \cdot \mathbf{r}$,    InitHeap
4     while not Finished
5         $k$ =: topHeap,
6         $w(k) := w(k) + c(k)$
7         $\mathbf{r} := \mathbf{r} - c(k) \cdot \mathbf{d}_k$
8         Update $\mathbf{c}$ and Heap
9         Check if Finished
10    end
11    return
```

Fig. 3. The Matching pursuit algorithm using shift-invariant dictionary. Input s is the target number of non-zero coefficients to select. The inner products are stored in a heap, initialized by the InitHeap-function, top extracted by topHeap-function, and in line 8 the inner products and the heap are updated, but only for atoms where the inner product has changed, i.e. the atoms with support overlapping the support of the selected atom.

As for MP, initialization sets $\mathbf{w}$ to zeros and the residual $\mathbf{r}$ to $\mathbf{x}$, but here the calculation of the inner products, $\mathbf{c}$, is done before the main loop. Since the atoms has fixed support the complexity is $\mathcal{O}(N)$. Line 5 in Fig. 2 has the computational complexity of $\mathcal{O}(N)$ which is not wanted inside a loop. To avoid this a *max-heap* data structure is used, to build the heap outside the loop is $\mathcal{O}(N)$. Inside the loop max is found in $\mathcal{O}(1)$ and the heap is updated in $\mathcal{O}(\log N)$.

The most demanding task inside the loop is to update the involved inner products and their place in the heap. But, since all atoms have local support the number of inner products to update is limited, depending only of the SID structure and independent of signal length N. The update step (line 8) is thus $\mathcal{O}(\log N)$ which also becomes the order of the loop. The loop is done at least $s = \mathcal{O}(N)$ times. This gives the computational cost for the SID-MP algorithm in the order of $\mathcal{O}(N \log N)$.

An implementation of this algorithm could be both compact and fast and it will be available on the web: `http://www.ux.uis.no/~karlsk/dle/`.

4 Orthogonal MP using SID

Orthogonal Matching Pursuit (OMP), somewhere denoted as Order Recursive Matching Pursuit (ORMP), finds better approximations than MP using the same number of non-zeros [4]. A throughout description of OMP/ORMP can be found in [4] or [24] and is not included here. For small (or moderately sized) dictionaries OMP is efficient, and fast implementations using QR-factorization are available: SPAMS from Mairal et al. [13] and OMPbox from Rubinstein et al. [21]. Nevertheless, these effective implementations have time complexity of order $\mathcal{O}((K+s^2)N) = \mathcal{O}(N^3)$ where N and K refer to the size of the dictionary for the signal block.

To do OMP on a large signal using a shift-invariant dictionary is more complicated than to do MP; since orthogonalization is done when a new atom is selected the effect on the residual, and thus the updated inner products, goes beyond the support of the selected atom. But the effect decreases as the distance from the selected atom increases. Using this fact, an approximate OMP algorithm can be developed in a way similar to the MP algorithm [11]. Another approach is used in the proposed method. It utilizes the local support of the atoms to divide the large OMP problem into many moderately sized OMP problems. The segment size is fixed and independent of N giving the computational complexity as $\mathcal{O}(N)$, but the constant factor is large $\mathcal{O}(N_{seg}^3)$ where N_{seg} is the signal segment length used in the OMP problems.

The proposed method, denoted as SID-OMP, has almost the same input as the SID-MP algorithm in Fig. 3: the shif-invariant dictionary represented as a SID structure, the long signal $\mathbf{x}$ of length N, and the target number of non-zero coefficients s *or* an initial coefficient vector $\mathbf{w}$ having the target number s non-zero elements. The initial coefficient vector $\mathbf{w}$ is found by the SID-MP algorithm in previous section if s is given as input, or $\mathbf{w}$ could be found by another preferred initialization method and given directly as input. The algorithm then does some few iterations, and in each iteration improved coefficients are found but the *number* of non-zero coefficients is unchanged. The coefficient vector $\mathbf{w}$ is returned. A detailed description of one iteration follows:

1. The signal is divided into L consecutive segments, the segment length is N_{seg} and the signal may be longer than the total length of the L segments, $N = N_b + LN_{seg} + N_e$. The excess portions are N_b samples at the beginning and N_e samples at the end of the signal, $0 \leq N_b, N_e < N_{seg}$.
2. The coefficients for atoms with support completely within one segment are set to zero, for each segment the number of coefficients set to zero is stored: s_i for $i = 1, 2, \dots, L$. The other coefficients are not changed. The original coefficients (and segment residuals) are also stored, and may be restored for some of the segments (in step 5 below).
3. The residual is calculated using the remaining coefficients, i.e. only the non-zero coefficients belonging to atoms with support that span two segments.
4. The residual segments are used as input for the block-oriented sparse approximation algorithm. If the segment size is chosen correctly, depending on the SID (P_m values), the dictionary D_{seg} will be the same for all segments and all iterations. For each segment the target number of non-zeros is s_i as found in step 2.
5. The coefficients are updated. If the sparse representation for a segment is better than what it was the new coefficients are used, if not the coefficients are set to the values they had when the iteration started. This way each iteration will improve (or keep) the sparse approximation.
6. Go back and do next iteration using a new segmentation of the original signal, i.e. new values for N_b and N_e. A few (4-8) iterations are sufficient.

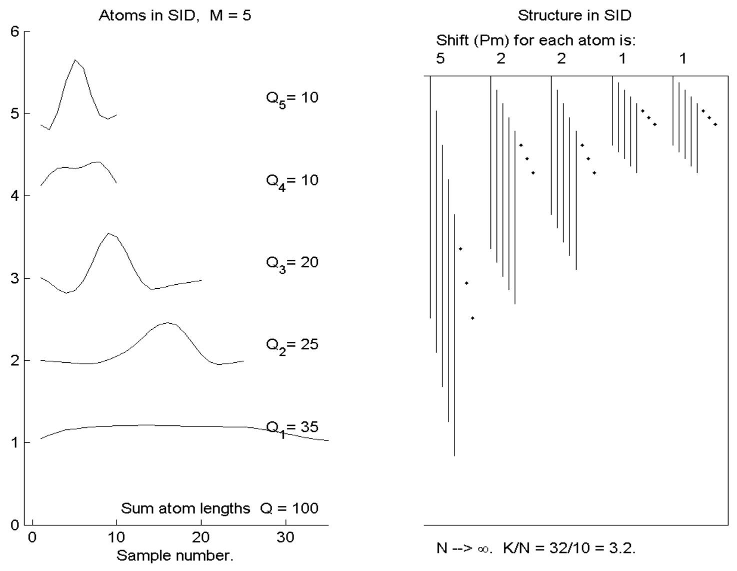

Fig. 4. Example of a shift-invariant dictionary. This structure is used in Table 1 and first row in Table 2.

One remark should be made to the algorithm as it is described above. The locations of the non-zeros coefficients will only change slowly from one iteration to the next, and it will take many iterations to move one non-zero coefficient a long distance. This problem with the algorithm can be avoided by selecting a good initial distribution, as SID-MP will give, or by modifying the algorithm: For all segments ($i = 1, 2, \dots, L$) the sparse approximation in step 4 could find solutions for $s_i - 1$, s_i and $s_i + 1$ non-zero coefficients and the respective residual norms, $r_{i,-1}, r_{i,0}, r_{i,+1}$, can be calculated. A non-zero coefficient should be moved from segment i to segment j if $(r_{j,0} - r_{j,+1}) > (r_{i,-1} - r_{i,0})$. Repeating this will move non-zeros from one segment to another as long as this operation will reduce the total error. The locations of the non-zeros will now move more quickly between signal parts. We should also note that SID-OMP does not need to use OMP as the sparse approximation algorithm, any block-oriented algorithm will do.

5 Experiments

In this section we present three simple experiments to illustrate how the proposed methods work. All experiments are conducted on an ECG signal from the MIT arrhythmia database [16]. The used signal, MIT100 where a short segment is shown in Fig. 5, is a normal heart rhythm, and the reason for using such an ECG signal is that it will obviously benefit from a SID structure. This signal

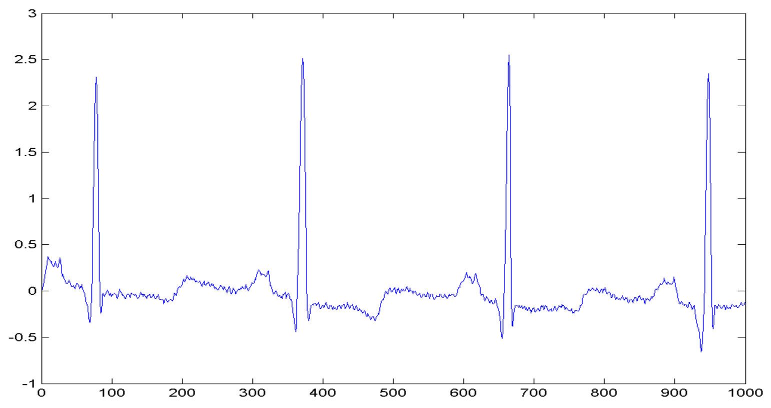

Fig. 5. First part of the used MIT100 normal rhythm ECG signal.

was also used for dictionary learning. As the purpose of these experiments is to test sparse approximation only, not a particular application, the actual signal or dictionaries used are not that important.

Experiment 1 is done to show how well, i.e. how fast, the SID-MP and SID-OMP algorithms run as the signal size increases. A SID with structure $M = 5$, $Q_m = \{35, 25, 20, 10, 10\}$ and $P_m = \{5, 2, 2, 1, 1\}$ as illustrated in Fig. 4, is used to make a sparse approximation of the MIT100 signal. The sparsity factor throughout the experiment is kept at 0.08, i.e. the number of non-zero coefficients is $0.08\,N$, where N is the signal length. The MIT100 signal is 250000 samples long, and longer test signals are made simply by repeating it. Fig. 6 shows the running time for SID-MP as the signal length increases, it scales (as expected) close to linear order and it is fast. For the longest signal, $N = 5$ million samples and the total number of dictionary atoms is $K = 15.6$ million, the number of non-zero coefficients is 0.4 million, and to find these takes less than 8 seconds on an Intel Core i5 3.2 GHz CPU PC. Another advantage of the proposed algorithm is that it is small (the compiled mex-file is 23 kB) and it does not depend on any special pre-installed libraries. The running times for one iteration for SID-OMP are also shown in Fig. 6, this line shows the linear order for computational times.

In **Experiment 2** SID-OMP is tested using different segment sizes N_{seg} and different number of iterations. For one segment size OMP is replaced by the computationally more demanding partial search algorithm [24] in the sparse approximation step. All tests use 250000 samples of the MIT100 signal and the sparsity factor is 0.08. Doing eight iterations the Signal to Noise Ratio (SNR) improves for each iteration as shown in Table 1. Most of the improvements is achieved already after two iterations, and using smaller segment sizes keeps on improving SNR for more iterations. After eight iterations, using $N_{seg} = 150$ is the better option. Also, using many small segments is faster that using fewer

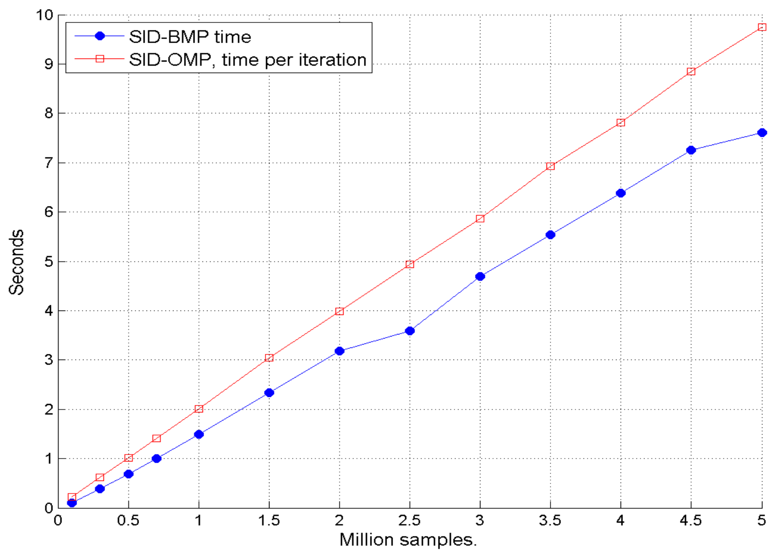

Fig. 6. Running times on an Intel Core i5 3.2 GHz CPU PC for the SID-MP algorithm shown on line marked by disks. The line marked by boxes is running time for one iteration of the SID-OMP algorithm. The segment dictionary D_{seg} has size 360×1170. Note that SID-OMP should do 4-8 iterations.

and larger segments. Thus it is better to use smaller segments than larger, but note that N_{seg} should always be larger than $\max_m Q_m$.

Experiment 3 tests SID-MP and SID-OMP using five different dictionaries, i.e. SID structures. The first three dictionaries are learned for the MIT100 signal, the first is as in Fig. 4. The fourth dictionary is a SID structure with atoms set as low-pass or band-pass filters, $Q_m \in \{128, 64, 32, 16\}$ and $P_m \in \{8, 4, 2, 1\}$, and the fifth dictionary has synthesis atoms from the modified DCT transform, $Q_m = 64$ and $P_m = 1$. The results are shown in Table 2.

6 Conclusion

The presented SID-MP is a variant of the matching pursuit algorithm that can be used for a shift-invariant dictionary structure. This structure is flexible as it encompasses any system that can be expressed as a general synthesis FIR filter bank of M filters, including the shift-invariant dictionary. The algorithm is fast and scales well to large signals. We also presented the SID-OMP method that allows us to use any (fast) block-oriented MP algorithm to further improve the sparse approximation.

SA in SID-OMP	D_{seg}	SNR after iteration number (SID-MP: 23.63) 1	2	3	4	5	6	7	8	time [s]
OMP	100 × 275	24.56	25.02	25.17	25.26	25.35	25.39	25.41	25.43	1.92
OMP	150 × 435	24.76	25.15	25.25	25.32	**25.39**	**25.42**	**25.44**	**25.44**	2.29
OMP	200 × 595	24.88	25.20	25.27	25.34	25.38	25.40	25.42	25.43	2.73
OMP	250 × 755	24.95	25.23	25.31	**25.35**	25.38	25.40	25.42	25.43	3.32
OMP	300 × 915	25.02	25.25	25.30	25.34	25.37	25.39	25.40	25.41	3.77
OMP	400 × 1235	25.09	25.27	25.31	25.34	25.36	25.37	25.38	25.39	5.23
OMP	500 × 1555	**25.10**	**25.28**	**25.32**	25.34	25.36	25.37	25.38	25.39	7.09
PS	150 × 435	25.36	25.88	26.03	26.12	26.15	26.19	26.21	26.23	67.5

Table 1. This table shows that SID-OMP in Section 4 improves the sparse approximation (SA) after SID-MP, SNR after SID-MP was 23.63. The table shows the achieved SNR after each of the iterations in SID-OMP. Time is shown for 8 iterations. The SA step in each iteration was performed by the OMP implementation in SPAMS except for the last line where the computationally more demanding Partial Search (PS) was used for comparison. There are 20000 non-zero coefficients and signal length is 250000. The segment size N_{seg}, and thus the segment dictionary D_{seg}, varies for the different rows.

SID structure				SID-MP		SID-OMP	
No.	M	Q	K/N	SNR	time [s]	SNR	time [s]
1	5	100	3.2	23.61	0.3	25.44	2.3
2	6	200	3.3	24.15	0.4	25.74	4.0
3	12	622	7.0	26.20	1.5	27.84	5.9
4	16	960	7.5	21.92	1.8	23.76	5.9
5	32	2048	32.0	18.11	8.8	19.65	16.3

Table 2. The achieved SNR for 250000 samples of the MIT100 signal using different SID structures: dictionary 1-3 are learned, dictionary 4 is a predefined filter bank, and 5 is a modified DCT. For each row, achieved SNR and the computational time for the two methods are shown.

References

1. Aharon, M., Elad, M.: Sparse and redundant modeling of image content using an image-signature-dictionary. SIAM Journal on Imaging Sciences 1(3), 228–247 (2008)
2. Aharon, M., Elad, M., Bruckstein, A.: K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. Signal Processing, IEEE Transactions on 54(11), 4311–4322 (2006), `http://dx.doi.org/10.1109/TSP.2006.881199`
3. Barzideh, F., Skretting, K., Engan, K.: The flexible signature dictionary. In: Proc. 23rd European Signal Processing Conf., EUSIPCO-2015. Nice, France (Sep 2015)
4. Cotter, S.F., Adler, J., Rao, B.D., Kreutz-Delgado, K.: Forward sequential algorithms for best basis selection. IEE Proc. Vis. Image Signal Process 146(5), 235–244 (Oct 1999)
5. Cvetković, Z., Vetterli, M.: Oversampled filter banks. IEEE Trans. Signal Processing 46(5), 1245–1255 (May 1998)

6. Davis, G.: Adaptive Nonlinear Approximations. Ph.D. thesis, New York University (Sep 1994)
7. Efron, B., Hastie, T., Johnstone, I., Tibshirani, R.: Least angle regression. Annals of Statistics 32, 407–499 (2004)
8. Engan, K., Aase, S.O., Husøy, J.H.: Method of optimal directions for frame design. In: Proc. ICASSP '99. pp. 2443–2446. Phoenix, USA (Mar 1999)
9. Engan, K., Skretting, K., Husøy, J.H.: A family of iterative LS-based dictionary learning algorithms, ILS-DLA, for sparse signal representation. Digital Signal Processing 17, 32–49 (Jan 2007)
10. Krstulovic, S., Gribonval, R.: MPTK: Matching pursuit made tractable. In: Proceedings ICASSP 2006. vol. 3, pp. 496–499. Toulouse, France (May 2006)
11. Mailhé, B., Gribonval, R., Bimbot, F., Vandergheynst, P.: A low complexity orthogonal matching pursuit for sparse signal approximation with shift-invariant dictionaries. In: Proceedings ICASSP 2009. pp. 3445–3448. Taipei, Taiwan (Apr 2009)
12. Mailhé, B., Lesage, S., Gribonval, R., Bimbot, F.: Shift-invariant dictionary learning for sparse representations: Extending K-SVD. In: Proceedings of the 16th European Signal Processing Conference (EUSIPCO-2008). Lausanne, Switzerland (aug 2008)
13. Mairal, J., Bach, F., Ponce, J.: Sparse modeling for image and vision processing. Foundations and Trends in Computer Graphics and Vision 8(2-3), 85–283 (2014)
14. Mairal, J., Bach, F., Ponce, J., Sapiro, G.: Online dictionary learning for sparse coding. In: ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning. pp. 689–696. ACM, New York, NY, USA (jun 2009)
15. Mallat, S.G., Zhang, Z.: Matching pursuit with time-frequency dictionaries. IEEE Trans. Signal Processing 41(12), 3397–3415 (Dec 1993)
16. Massachusetts Institute of Technology: The MIT-BIH Arrhythmia Database CD-ROM. MIT, 2 edn. (1992)
17. Muramatsu, S.: Structured dictionary learning with 2-D non-separable oversampled lapped transform. In: Proceedings ICASSP 2014. pp. 2643–2647. Florence, Italy (May 2014)
18. O'Hanlon, K., Plumbley, M.D.: Structure-aware dictionary learning with harmonic atoms. In: Proceedings of the 19th European Signal Processing Conference (EUSIPCO-2011). Barcelona, Spain (aug 2011)
19. Pati, Y.C., Rezaiifar, R., Krishnaprasad, P.S.: Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In: Proc. of Asilomar Conference on Signals Systems and Computers (Nov 1993)
20. Pope, G., Aubel, C., Studer, C.: Learning phase-invariant dictionaries. In: Proceedings ICASSP 2013. Vancouver, Canada (May 2013)
21. Rubinstein, R., Zibulevsky, M., Elad, M.: Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit. Tech. rep., CS Technion, Haifa, Israel (Apr 2008)
22. Rubinstein, R., Zibulevsky, M., Elad, M.: Double sparsity: Learning sparse dictionaries for sparse signal approximation. IEEE Transactions on Signal Processing 58(3), 1553–1564 (Apr 2010)
23. Skretting, K., Engan, K.: Recursive least squares dictionary learning algorithm. IEEE Transactions on Signal Processing 58, 2121–2130 (Apr 2010), digital object identifier: 10.1109/TSP.2010.2040671
24. Skretting, K., Husøy, J.H.: Partial search vector selection for sparse signal representation. In: NORSIG-03. Bergen, Norway (Oct 2003), available at `http://www.ux.uis.no/~karlsk/`