Fast Non-Negative Orthogonal Matching Pursuit

Fast Non-Negative Orthogonal Matching Pursuit

Mehrdad Yaghoobi, Di Wu and Mike E. Davies
Institute for Digital Communications, the University of Edinburgh, EH9 3JL, UK
{m.yaghoobi-vaighan, d.wu and mike.davies}@ed.ac.uk

***Abstract*—One of the important classes of sparse signals is the non-negative signals. Many algorithms have already been proposed to recover such non-negative representations, where greedy and convex relaxed algorithms are among the most popular methods. The greedy techniques have been modified to incorporate the non-negativity of the representations. One such modification has been proposed for Orthogonal Matching Pursuit (OMP), which first chooses positive coefficients and uses a non-negative optimisation technique as a replacement for the orthogonal projection onto the selected support. Beside the extra computational costs of the optimisation program, it does not benefit from the fast implementation techniques of OMP. These fast implementations are based on the matrix factorisations. We here first investigate the problem of positive representation, using pursuit algorithms. We will then describe a new implementation, which can fully incorporate the positivity constraint of the coefficients, throughout the selection stage of the algorithm. As a result, we present a novel fast implementation of the Non-Negative OMP, which is based on the QR decomposition and an iterative coefficients update. We will empirically show that such a modification can easily accelerate the implementation by a factor of ten in a reasonable size problem.**

***Index Terms*—Matching Pursuit, Orthogonal Matching Pursuit, Non-negative Sparse Approximations, QR Matrix Factorisation, Non-negative Least Square and Spectral Decomposition**

I. Introduction

If the signal of interest is $\mathbf{y} \in \mathbb{R}^M$ and a dictionary of elementary functions $\mathbf{\Phi} \in \mathbb{R}^{M,N}$ are given, the linear sparse approximation can be formulated as finding the sparsest $\mathbf{x} \in \mathbb{R}^N$, $M < N$, *i.e.* having the minimum number of non-zero elements, as follows,

$$\mathbf{y} \approx \mathbf{\Phi}\mathbf{x}. \tag{1}$$

The greedy sparse approximation algorithms are generally computationally low cost algorithms, suitable for real-time and large scale sparse approximations. One simple greedy algorithm is called Matching Pursuit (MP) [1], whichaa builds the sparse representation of a signal by iteratively adding the most correlated element of the dictionary, which is called an atom, to the set of selected elements. A disadvantage of MP is that the representation found by the algorithm is not the best representation using selected atoms. It may also reselect already selected atoms in the later iterations, which slows down the convergence of the algorithm. OMP algorithm [2], [3] is proposed to compensate these issues, with some extra computation cost, which is mainly due to the orthogonal projection of the signal $\mathbf{y}$ onto the selected support s. Such a projection can be found by,

$$\tilde{\mathbf{x}}_s := \text{argmin}_{\mathbf{x}_s} \|\mathbf{y} - \mathbf{\Phi}_s\mathbf{x}_s\|_2, \tag{2}$$

This work was supported by EPSRC grants EP/K014277/1 and the MOD University Defence Research Collaboration in Signal Processing.

where $\mathbf{\Phi}_s$ and $\mathbf{x}_s$ are respectively the sub-dictionary and the coefficient vector restricted to the support s. As the residual signal, after pruning out the contribution of current atoms, is orthogonal to the selected atoms, these atoms would not be selected in the future iterations.

There are many applications for which not only are the coefficient vectors sparse, but they are also non-negative. Spectral and multi-spectral unmixing [4], [5] and microarray analysis [6] are only few examples of such applications. Raman spectral deconvolution [7] and multi-touch sensing [8] have been our initial motivation for the current work.

Some modifications have been proposed for the MP and OMP algorithms to incorporate the non-negativity of the coefficients [9]. The only necessary modification for MP is to choose the most "positively" correlated atom, at each iteration of the algorithm. This means that we only select the atoms with the positive coefficients. In OMP, we have a projection step at each iteration, which may find some negative coefficients. A sub-optimal approach, which guarantees the non-negativity of the coefficients, is to use a non-negative least square (NNLS) program to refine the selected coefficients at each iteration [9], as follows,

$$\text{argmin}_{\mathbf{x}_s \geq 0} \|\mathbf{y} - \mathbf{\Phi}_s\mathbf{x}_s\|_2,$$

where $\geq$ operator is the component-wise greater or equal operator. A pseudo-code of Canonical Non-Negative OMP (CNNOMP) is presented in Algorithm 1. Let $\mathbf{r}_k := \mathbf{y} - \mathbf{\Phi}\mathbf{x}_k$ be the kth signal residual. In this algorithm, the sub-optimality of such approach is due to the fact that the selected positive coefficient at an iteration, may force already selected coefficients to become zero to still remain in the admissible set, which reduce the efficiency of the algorithm. In other words, this is caused by dividing the selection and the NNLS as two separate tasks. We show here that there is an alternative approach which combines these two steps. As a result of such combination, we can implement the algorithm in an efficient approach, which has similarities with the canonical fast OMP implementations, *i.e.* OMP with QR factorisation [10]–[13].

To introduce our algorithm, we first need to briefly explain the fast factorization based OMP, which can be found in section II. In section III, we explain how the non-negativity constraint of the coefficients stops us of using the canonical OMP and how we can modify the algorithm to not only have a more intuitive atom selection step, but also have a lower computational complexity. We then show that the proposed method has a much faster implementation in the simulation part of section IV.

Algorithm 1 Canonical Non-Negative Orthogonal Matching Pursuit

initialisation: $s = \emptyset$, $k = 0$, $\mathbf{x} = \mathbf{0}$ and $\mathbf{r}_0 = \mathbf{y}$
while $k < K$ & $\max(\mathbf{\Phi}^T\mathbf{r}_k) > 0$ **do**
  $(\zeta, \iota) \leftarrow \max(\mathbf{\Phi}^T\mathbf{r}_k)$
  $s \leftarrow s \cup \iota$
  $\mathbf{x}_s \leftarrow \operatorname{argmin}_{\boldsymbol{\theta}>0}\|\mathbf{y} - \mathbf{\Phi}_s\boldsymbol{\theta}\|_2$
  $\mathbf{r}_{k+1} \leftarrow \mathbf{y} - \mathbf{\Phi}_s\mathbf{x}_s$
  $k \leftarrow k + 1$
end while
$\mathbf{x}|_s \leftarrow \mathbf{x}_s$

II. Fast Orthogonal Matching Pursuit (FOMP)

In the standard implementation of OMP, we solve (2) at each iteration. It can be solved by calculating the Moore-Penrose pseudo-inverse of $\mathbf{\Phi}_s$, *i.e.* $\mathbf{\Phi}_s^\dagger$ and $\tilde{\mathbf{x}}_s = \mathbf{\Phi}_s^\dagger\mathbf{y}$. At iteration k, $|s| = k$ and calculation of $\mathbf{\Phi}_s^\dagger$ needs a matrix inversion with the size $k \times k$. When k increases, it computationally becomes very expensive, *i.e.* $\mathcal{O}(k^3)$. To combat such a computational burden, incorporation of a matrix decomposition of the selected sub-dictionary has been proposed, where QR factorisation is among the most effective techniques. Let $\mathbf{\Phi}_k = \mathbf{\Psi}_k\mathbf{R}_k$ be the QR factorisation of selected k atoms of dictionary $\mathbf{\Phi}$, where $\mathbf{\Psi}_k$ is column orthonormal and $\mathbf{R}_k$ is upper-triangle with positive diagonal elements on its main diagonal. With some abuse of notation, we assume that in iteration k, the columns of $\mathbf{\Phi}_k$ are sorted based on the iteration number and $\boldsymbol{\phi}_i$, for $1 \le i \le k$, is the ith selected atom. As the column span of $\mathbf{\Phi}_k$ and $\mathbf{\Psi}_k$ are equivalent, we can simply solve, $\tilde{\mathbf{z}}_k := \operatorname{argmin}_{\mathbf{z}_k}\|\mathbf{y} - \mathbf{\Psi}_k\mathbf{z}_k\|_2$, instead of solving (2), with $\mathbf{\Phi}_s \leftarrow \mathbf{\Phi}_k$ and $\tilde{\mathbf{x}}_s \leftarrow \mathbf{x}_k$, and find the solution by $\tilde{\mathbf{x}}_k = \mathbf{R}_k^{-1}\tilde{\mathbf{z}}_k$. As $\mathbf{\Psi}_k$ is orthonormal, $\tilde{\mathbf{z}}_k = \mathbf{\Psi}_k^T\mathbf{y}$, which can be implemented quickly. The algorithm would not be fast, if the calculation of $\mathbf{\Psi}$, $\mathbf{R}$ and $\mathbf{R}^{-1}$ could not be done efficiently. There has been some fast realisations of $\mathbf{\Psi}_k$ and $\mathbf{R}_k$ based on its iterative updates. To derive an update formula for $\mathbf{\Psi}_k$, we recall the Graham-Schmidt (GS) orthogonalisation procedure. As the first k terms of $\mathbf{\Phi}_{k+1}$ has a QR factorisation $\mathbf{\Psi}_k\mathbf{R}_k$, we only need to continue the GS procedure to find the last column of $\mathbf{\Psi}_{k+1}$. In this setting, we have,

$$\mathbf{\Psi}_{k+1} = [\mathbf{\Psi}_k \ \boldsymbol{\psi}_{k+1}],$$

where $\boldsymbol{\psi}_{k+1} = \mathbf{q}_{k+1}/\|\mathbf{q}_{k+1}\|_2$ and $\mathbf{q}_{k+1} = (\mathbf{I} - \mathbf{\Psi}_k\mathbf{\Psi}_k^T)\boldsymbol{\phi}_{k+1}$. $\mathbf{\Psi}_k\mathbf{\Psi}_k^T$ operator projects the operand to the span of $\mathbf{\Psi}_k$ and $(I - \mathbf{\Psi}_k\mathbf{\Psi}_k^T)$ finds the orthogonal element to the span of $\mathbf{\Psi}_k$. We normalise such an orthogonal element to find $\mathbf{\Psi}_{k+1}$. A similar approach can be used to calculate $\mathbf{R}_{k+1}$ using $\mathbf{R}_k$ as follows:

$$\mathbf{R}_{k+1} = \begin{bmatrix} \mathbf{R}_k & \nu \\ \mathbf{0} & \mu \end{bmatrix}, \tag{3}$$

where $\nu = \mathbf{\Psi}_k^T\boldsymbol{\phi}_{k+1}$ and $\mu = \|\mathbf{q}_{k+1}\|_2$. If we calculate $\mathbf{R}^{-1}$ in the same way, we have the following update formula,

$$\mathbf{R}_{k+1}^{-1} = \begin{bmatrix} \mathbf{R}_k^{-1} & -\frac{\mathbf{R}_k^{-1}\nu}{\mu} \\ \mathbf{0} & \frac{1}{\mu} \end{bmatrix}. \tag{4}$$

The OMP algorithm with this implementation is faster than standard pseudo-inverse implementation, for medium to large

Algorithm 2 Fast Non-Negative Orthogonal Matching Pursuit

initialisation: $s = z_0 = \emptyset$, $k = 0$ and $\mathbf{r}_0 = \mathbf{y}$
while $k < K$ & $\max(\mathbf{\Phi}^T\mathbf{r}_k) > 0$ **do**
  $(\boldsymbol{\zeta}, \boldsymbol{\iota}) \leftarrow \operatorname{sort}_\downarrow(\mathbf{\Phi}^T\mathbf{r}_k)$
  $p \leftarrow 1$
  $p^c \leftarrow 1$
  $z^c \leftarrow 0$
  while $\sim$ Terminate & $p < N$ **do**
    z^t from (5)
    $z \leftarrow \boldsymbol{\psi}_{\boldsymbol{\iota}(p)}^T\mathbf{r}_k : \boldsymbol{\psi}_{\boldsymbol{\iota}(p)} = \mathbf{q}/\|\mathbf{q}\|_2,\ \mathbf{q} = (\mathbf{I} - \mathbf{\Psi}_k\mathbf{\Psi}_k^T)\boldsymbol{\phi}_{\boldsymbol{\iota}(p)}$
    Update based on Table I
  end while
  $s \leftarrow s \cup \boldsymbol{\iota}(p)$
  Update $\mathbf{\Psi}$ and $\mathbf{R}^{-1}$
  $\mathbf{z}_{k+1} \leftarrow [\mathbf{z}_k; z_{k+1}]$
  $\mathbf{r}_{k+1} \leftarrow \mathbf{r}_k - z_{k+1}\boldsymbol{\psi}_{k+1}$
  $k \leftarrow k + 1$
end while
output: $\mathbf{x}|_s \leftarrow \mathbf{R}^{-1}\mathbf{z}_K$

If	Then
$0 < z \le z^t,\ z > z^c$	$z_{k+1} \leftarrow z$, Terminate
$0 < z \le z^t,\ z \le z^c$	$z_{k+1} \leftarrow z^c, p \leftarrow p^c$, Terminate
$z > z^c \ge z^t$	$p \leftarrow p + 1$
$z^c \ge z > z^t$	$z_{k+1} \leftarrow z^c, p \leftarrow p^c$, Terminate
$z > z^t > z^c$	$z^c \leftarrow z^t, p^c \leftarrow p,\ p \leftarrow p + 1$
$z < 0$	Terminate

TABLE I
Decision rules to guarantee positivity of the coefficients.

k's. After a close look at this implementation, we realise that we do not need to keep track of $\mathbf{x}_k$'s in the intermediate iterations! We only need to calculate $\mathbf{z}_k$ at each iteration and find $\mathbf{x}$ after the last iteration K, *i.e.* $\mathbf{x} = \mathbf{R}_K^{-1}\mathbf{z}_K$, where $\mathbf{z}_K$ is $\mathbf{z}$ at the Kth iteration. It is also worth mentioning that we also do not keep track of $\mathbf{R}_k$, only updating $\mathbf{\Phi}_k$ and $\mathbf{R}_k^{-1}$ are necessary.

When we choose positively correlated atoms, then finding $\mathbf{z}_k$, we may get negative elements in the corresponding $\mathbf{x}_k$. This fact does not allow us to directly use the FOMP technique in a non-negative setting.

III. Fast Non-Negative Orthogonal Matching Pursuit (FNNOMP)

Canonical NNOMP chooses the atom which maximises $\mathbf{\Phi}^T\mathbf{r}_k$ in the kth iteration. In the first iteration, we do not need any orthogonalisation and we have $\boldsymbol{\phi}_1 = \boldsymbol{\psi}_1$ and $\mathbf{R} = [1]$. In the kth iteration, let the best approximation of $\mathbf{y}$, with the non-negative coefficients and using $\mathbf{\Phi}_k$, be $\sum_{i=1}^k x_i\boldsymbol{\phi}_i = \sum_{i=1}^k z_i\boldsymbol{\psi}_i$. In the $k+1$ iteration, we have,

$$\begin{aligned}\sum_{i=1}^{k+1} z_i\boldsymbol{\psi}_i &= \sum_{i=1}^{k} z_i\boldsymbol{\psi}_i + z_{k+1}\boldsymbol{\psi}_{k+1} \\ &= \sum_{i=1}^{k} x_i\boldsymbol{\phi}_i + z_{k+1}\boldsymbol{\psi}_{k+1}\end{aligned}$$

As $\boldsymbol{\psi}_{k+1}$ lives in the span of the non-redundant set $\{\boldsymbol{\phi}_j\}_{j\in[1,k+1]}$, $\boldsymbol{\psi}_{k+1} = \sum_{j=1}^{k+1}\gamma_j\boldsymbol{\phi}_j$ for some unique γ_j. We

then have,

$$\sum_{i=1}^{k+1} z_i \boldsymbol{\psi}_i = \sum_{i=1}^{k} x_i \boldsymbol{\phi}_i + \sum_{j=1}^{k+1} z_{k+1}\gamma_j \boldsymbol{\phi}_j$$
$$= \sum_{i=1}^{k} (x_i + z_{k+1}\gamma_i)\boldsymbol{\phi}_i + z_{k+1}\gamma_{k+1}\boldsymbol{\phi}_{k+1}.$$

As $z_{k+1}\gamma_{k+1}$ is always positive, we only need to assure that $x_i + z_{k+1}\gamma_i \geq 0$ or

$$z_{k+1} \leq z^t := \begin{cases} \min_{i,\gamma_i<0} \left|\frac{x_i}{\gamma_i}\right| & \exists i, \gamma_i < 0 \\ +\infty & \text{Otherwise}. \end{cases} \quad (5)$$

To assure that x_i's are all non-negative, z_i's should comply with the condition of (5). We then choose the atom that the corresponding z_{k+1}, or its shrunk by upper-bound of (5), *i.e.* z^t, if it is larger than the upper-bound, has the largest value. We therefore need to track the record of the best possible solution, if the most positively correlated atom has a z_k that does not comply with (5). If we call such a possible solution $\mathbf{z}^c$, $(\boldsymbol{\zeta}, \boldsymbol{\iota}) = \text{sort}_\downarrow(\boldsymbol{\Phi}^T \mathbf{r}_k)$, where $\text{sort}_\downarrow$ is the sorting operator in a descent order, and z is the current candidate, starting with $z = \zeta(p),\ p = 1$, in an internal loop in the kth iteration, we make the decision based on the rules of Table I.

After the termination of the inner-loop, we add $\iota(p)$ to the support s and update $\boldsymbol{\Psi}_k$ and $\mathbf{R}_k^{-1}$. The overall steps of the algorithm are presented in Algorithm 2. The algorithm consists of two loops. The external loop terminates in a finite number of iterations as K is finite. We only need to check the termination of the inner-loop which makes the fast NNOMP different from canonical NNOMP.

***Lemma* 1.** *The inner loop of Algorithm 2 converges in a finite number of iterations.*

Proof: The updating conditions of Table I have a "Terminate" as an output or it increases p. As the dictionary has a finite dimension N, the inner loop terminates when a "Terminate" signal occurs or $p = N$. ■

While there might be worst cases for which the inner-loop has to check all elements before termination, our observation is that this loop terminates after only a few iterations.

For a fast implementation, we have to efficiently calculate $\boldsymbol{\gamma}$. By a careful investigation, we realise that it can be calculated using $\mathbf{R}_k^{-1}$, which is already kept in the memory. In this setting, we can easily check that $\boldsymbol{\gamma}$ is the last column of $\mathbf{R}_{k+1}^{-1}$, if $\boldsymbol{\phi}_{k+1}$ is the selected atom, *i.e.*,

$$\boldsymbol{\gamma} = [-\frac{\mathbf{R}_k^{-1}\nu}{\mu}; \frac{1}{\mu}], \quad (6)$$

where ν and μ are the same as what were defined after (3). It is worth mentioning that we do not update $\mathbf{R}_k^{-1}$ as we are not sure at this stage that it is the most appropriate atom.

Based on (5), we only need to check this condition if some γ_i's are negative. A question may be, is this really happening? The answer to this question is important, as otherwise, the proposed algorithm would be the same as FOMP. We have demonstrated a simple case to show that this case actually happens, in Figure 1. In this figure, we assume that $\boldsymbol{\phi}_1$ is already selected as the first atom and the next selection is

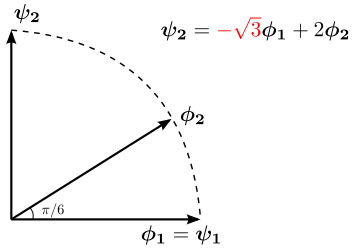

Fig. 1. A simple example in which gamma has negative elements for γ_2 .

$\boldsymbol{\phi}_2$. In this case, it is easy to check that $\boldsymbol{\psi}_2 = -\sqrt{3}\boldsymbol{\phi}_1 + 2\boldsymbol{\phi}_2$. This simple example justifies the use of FNNOMP, which guarantees the positivity of the coefficients, at each iteration of algorithm.

A. Computational Complexity

Having a structured dictionary may help us having a fast dictionary-coefficient multiplication. However, the analysis here is based on a dictionary without such a fast multiplication, as it applies to many applications with non-negative sparsity models. An extension to the fast dictionary is also possible, if we accurately know the complexity of dictionary multiplication.

The new implementation has a significantly different computational complexity to CNNOMP. The CNNOMP of [9] has an internal non-negative least square optimisation step, which has an asymptotic computational complexity of $\mathcal{O}(LMk^2)$, where k is the iteration number and L is the number of internal iterations [14]. L is normally in the order of k. This optimisation step repeats at each iteration, which makes the total cost of $\mathcal{O}(LMK^3)$.

In the fast NNOMP, the inner-loop has some comparison operations from Table I, which have negligible computational cost. The other steps are the calculation of z^t and z, which respectively cost $\mathcal{O}(M(k+1) + (k+1)^2 + 1)$ and $\mathcal{O}(2M(k+1)+1)$. This inner-loop repeats a few times, *i.e.* P. Our observation is that P does not proportionally scale with the size of problem. The total cost of repeating the inner-loop will be $\mathcal{O}(P(3M(K+1)+K^2))$. Another extra computational cost of FNNOMP is in the sorting of the coefficients, which is $\mathcal{O}(N\log(P))$ in the worst case for finding sorted P largest coefficients. As we need to sort the coefficients in each iteration, the total cost will be $\mathcal{O}(KN\log(P))$. The inversion of matrix $\mathbf{R}$, which is a necessary to find $\mathbf{x}$ at the end of algorithm [10], can be avoided using an iterative update of $\mathbf{R}^{-1}$.

If we calculate the computational cost of each step of the two algorithms, we can derive Table II. The total cost of two algorithms, after ignoring some low-complexity terms, are presented in the last row of this table. As it can be seen, the complexity of the CNNOMP is of order five when K is large and comparable with M. In comparison, FNNOMP has a term which depends on P. While P is small, the computational complexity of FNNOMP is of order three. This shows that FNNOMP is more favourable for the large scale problems with medium to large K's.

The conclusion of this part relies on the fact that L is scaling with the order of K and P is not scaling directly with the

CNNOMP	FNNOMP
2: $MN + N$	2,3: $MN + N + N\log(N)$
5: LMk^2	8: $M(k+1) + k^2 + 2k + 2$
6: Mk	18: K^2
Total: $MNK + LMK^3$	Total: $KN(M+1) + KN\log(P) + PMK(K+1) + PK^3 + K^2$

TABLE II
COMPUTATIONAL COMPLEXITY OF DIFFERENT IMPLEMENTATIONS OF NNOMP. THE BEGINNING NUMBERS ARE THE CORRESPONDING LINES IN ALGORITHMS 1 AND 2

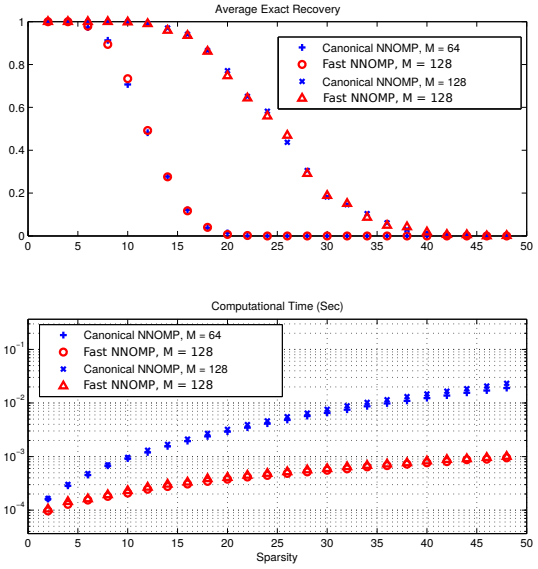

Fig. 2. Exact recovery (top panel) and computation time (botom panel). $N = 256$ and $M = 64$ & 128 are fixed while sparsity K is changing.

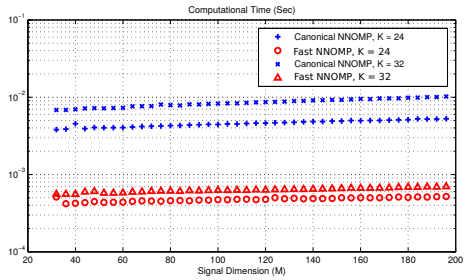

Fig. 3. Computation time for the fixed $N = 256$ and $K = 24$ & 32.

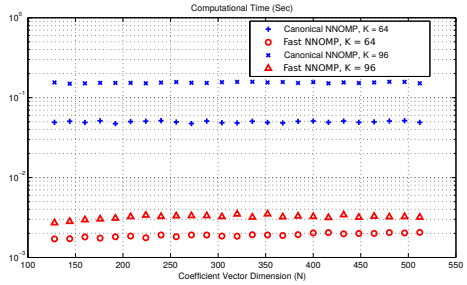

Fig. 4. Computation time for the fixed $M = 128$ and $K = 64$ & 96.

dimension of the problem. Although this is a hypothesis, we next show that it seems to be true in practice, based on our observations.

IV. SIMULATIONS

In this section, we investigate the computational costs and the algorithm outputs of the CNNOMP and FNNOMP on a single core of an Intel core 2.6 GHz processor. In the first experiment, we randomly generated $\mathbf{\Phi}$ with 256 atoms and 64 or 128 rows using *i.i.d.* Gaussian random variables. $\mathbf{y}$ was generated using a Gaussian-Bernoulli model, *i.e.* uniformly random support and Normal distribution for the non-zero coefficients. With different sparsity levels and using the proposed CNNOMP and FNNOMP, we repeated the simulations 1000 times. The exact recovery, *i.e.* correct recovery of the support, and computational time are shown in Figure 2. While the exact recovery is very similar, the proposed method is significantly faster for large K's. Different increasing rates of computational cost is due to different dependencies on K, which is presented in Table II. The dominant term of the complexity of CNNOMP is LMK^3, which changes like K^4, when $L \sim K$. On the other hand, the dominant term of FNNOMP behaves like $\mathcal{O}(K^3)$, for small P's. The other important observation in Figure 2 is the significant difference between CNNOMP and FNNOMP in the computational time, for a fixed K, in favour of the proposed technique.

In the second experiment, we used the same method to generate the dictionary, but we fixed $N = 256$, to investigate the computation time as a function of M. Here $K = 24$ or 32 and M is between 32 and 196. The computational time is plotted in Figure 3. The computational time is slowly increasing with M, as its order in the total complexity is one, see Table II. However, the rate of increase is higher for the CNNOMP.

In the last experiment, we fixed $M = 128$ and changed N from 128 to 512 and repeated the simulations as before for 1000 times. The sparsity was set to be 64 or 96, and the results are shown in Figure 4. The computational cost is increasing very slowly by increasing N in FNNOMP, while it is almost constant for CNNOMP. This difference seems to come from the fact that FNNOMP has the term $N\log(P)$, in contrast with N in CNNOMP. Since the computational time of CNNOMP is much higher than FNNOMP, CNNOMP will be competitive only for very large N's.

V. CONCLUSION

We presented a new greedy technique based on OMP, suitable for non-negative sparse representation, which is much faster than the state of the art algorithm. The new algorithm has a slightly different atom selection procedure, which guarantees the non-negativity of the signal approximations. Although the selection step is more involved, the overall algorithm has a much faster implementation. The reason is that with the new selection procedure, we can use fast QR implementation of the OMP. The computational complexity of two NNOMP implementations were derived and the differences were demonstrated.

REFERENCES

[1] S. Mallat and Z. Zhang, "Matching pursuits with time frequency dictionaries," *IEEE Trans. on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[2] Y.C. Pati, R. Rezaiifar, and P.S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in *Asilomar Conference on Signals, Systems and Computers*, 1993, pp. 40–44.

[3] B.K. Natarajan, "Sparse approximate solutions to linear systems," *SIAM Journal of Comput*, vol. 24, no. 2, pp. 227–234, 1995.

[4] MD Iordache, JM Bioucas-Dias, and A Plaza, "Sparse unmixing of hyperspectral data," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 49, no. 6, pp. 2014–2039, 2011.

[5] Y Qian, S Jia, J Zhou, and A Robles-Kelly, "Hyperspectral unmixing via sparsity-constrained nonnegative matrix factorization," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 49, no. 11, pp. 4282–4297, 2011.

[6] H Kim and H Park, "Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis," *Bioinformatics*, vol. 23, no. 12, pp. 1495–1502, 2007.

[7] D Wu, M Yaghoobi, SI Kelly, ME Davies, and R Clewes, "A sparse regularized model for raman spectral analysis," in *Sensor Signal Processing for Defence*, Edinburgh, 2014.

[8] M Yaghoobi, S. McLaughlin, and M.E. Davies, "Super-resolution sparse projected capacitive multitouch sensing," in *Intelligent Signal Processing (ISP)*, 2013.

[9] AM Bruckstein, M Elad, and M Zibulevsky, "Sparse non-negative solution of a linear system of equations is unique," in *3rd International Symposium on Communications, Control and Signal Processing, ISCCSP*, 2008, pp. 762–767.

[10] T Blumensath and ME Davies, "On the difference between orthogonal matching pursuit and orthogonal least squares," Tech. Rep., 2007.

[11] B Sturm and MG Christensen, "Comparison of orthogonal matching pursuit implementations," in *Proc. Eusipco*, 2012, pp. 220–224.

[12] M Gharavi-Alkhansari and TS Huang, "A fast orthogonal matching pursuit algorithm," in *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*. 1998, vol. 3, pp. 1389–1392, IEEE.

[13] R. Rubinstein, M. Zibulevsky, and M. Elad, "Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit," Tech. Rep., Technion, Apr. 2008.

[14] Y. Luo and R. Duraiswami, "Efficient parallel nonnegative least squares on multicore architectures," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2848–2863, 2011.