Supplementary material for the paper: *Learning Multi-level Sparse Representations*

Ferran Diego **Fred A. Hamprecht**
Heidelberg Collaboratory for Image Processing (HCI)
Interdisciplinary Center for Scientific Computing (IWR)
University of Heidelberg, Heidelberg 69115, Germany
{ferran.diego,fred.hamprecht}@iwr.uni-heidelberg.de

1 Optimization

We now consider the optimization in Fig. 1 (Fig. 3(d) in the main paper). The problem is not *jointly* convex, but becomes convex w.r.t. one variable while keeping fixed the others if we assume that the norms Ω_U, Ω_D, and Ω_A are also convex. Hence, the main optimization procedure is based on iteratively optimizing a group of variables while fixing the others. In addition, these norms are not restricted to control only the sparsity of $\bar{\mathbf{U}} = [\mathbf{U}^0, \ldots, \mathbf{U}^L]$, $\mathbf{D}$ and $\bar{\mathbf{A}} = [\mathbf{A}^1, \ldots, \mathbf{A}^L]$ but can also bias towards non–zero patterns of a specific structure, *e.g.* sets of variables forming rectangles or specific shapes.

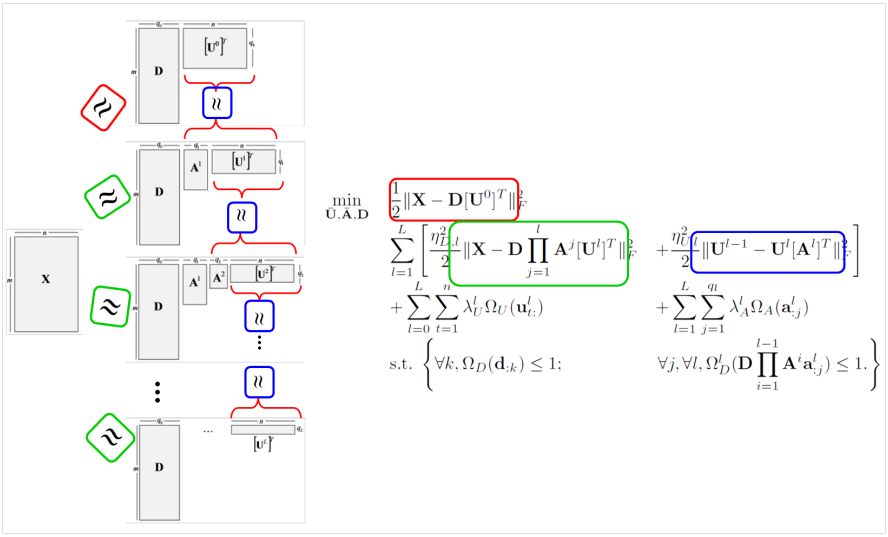


Figure 1: Decomposition of $\mathbf{X}$ into multiple levels. From left to right: Illustration of matrix decomposition and the main equation. Boxes indicates the different matrix factorizations included in the main equation.

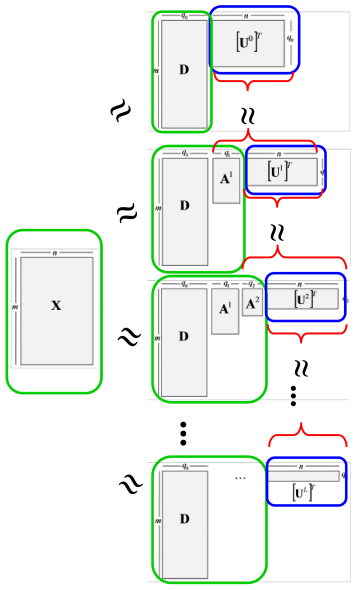
Update of U: The optimization of $\bar{\mathbf{U}}$ can be solved using a bottom–up block coordinate descent (from coarser to higher levels). The update of U^j at level j is a convex problem while keeping fixed the other variables. This can be written as follows:

$$\min_{\mathbf{U}^j} \frac{1}{2}\|\hat{\mathbf{X}} - \hat{\mathbf{D}}[\mathbf{U}^j]^T\|_F^2 + \sum_{t=1}^{n} \lambda_U^j \Omega_U(\mathbf{u}_{t:}^j), \tag{1}$$

where

$$\hat{\mathbf{X}} = \begin{bmatrix} \eta_{D,j}\mathbf{X} \\ \eta_{U,j}[\mathbf{U}^{j-1}]^T \\ \eta_{U,j+1}\mathbf{A}^{j+1}[\mathbf{U}^{j+1}]^T \end{bmatrix}, \hat{\mathbf{D}} = \begin{bmatrix} \eta_{D,j}\mathbf{D}\prod_{i=1}^{j}\mathbf{A}^i \\ \eta_{U,j}\mathbf{A}^j \\ \eta_{U,j+1}\mathbf{I}_{q_j} \end{bmatrix},$$

and $\mathbf{I}_{q_j}$ denotes the identity matrix of size q_j. In addition, if we assume that all the coefficient matrices are regularized with the same Ω_u but with different λ_U^j that increases monotonically, *i.e.* $\lambda_U^0 \leq \lambda_U^1 \leq \ldots \leq \lambda_U^L$, or Ω_U that relates all the coefficients, we can optimize *jointly* the multi-level sparse representation $\bar{\mathbf{U}}$ and take advantage of well-developed solvers. This joint optimization is written as shown in Fig. 2.



$$\min_{\tilde{\mathbf{U}}} \frac{1}{2}\|\tilde{\mathbf{X}} - \tilde{\mathbf{D}}\bar{\mathbf{U}}^T\|_F^2 + \sum_{l=0}^{L}\sum_{t=1}^{n} \lambda_U^l \Omega_U(\mathbf{u}_{t:}^l),$$

where $\tilde{\mathbf{X}} = \left[\mathbf{X}^T, \ldots, \eta_{D,L}\mathbf{X}^T, \mathbf{0}, \ldots, \mathbf{0}\right]^T$ and

$$\tilde{\mathbf{D}} = \begin{bmatrix} \mathbf{D} & & & & \mathbf{0} \\ & \eta_{D,2}\mathbf{D}\mathbf{A}^1 & & & \\ & & \ddots & & \\ \mathbf{0} & & & & \eta_{D,L}\mathbf{D}\prod_{i=1}^{L}\mathbf{A}^i \\ -\eta_{U,1}\mathbf{I}_{q_0} & \eta_{U,1}\mathbf{A}^1 & & & \mathbf{0} \\ & -\eta_{U,2}\mathbf{I}_{q_1} & \eta_{U,2}\mathbf{A}^2 & & \\ & & \ddots & \ddots & \\ \mathbf{0} & & & -\eta_{U,L}\mathbf{I}_{q_{L-1}} & \eta_{U,L}\mathbf{A}^L \end{bmatrix}$$

Figure 2: Joint optimization of the multi-level sparse representation $\bar{\mathbf{U}}$. From left to right: Illustration of matrix decomposition such that green boxes indicates fixed variables and blue the variable to estimate, and the main equation.

Update of $\mathbf{A}$: The update of adjacency matrix $\bar{\mathbf{A}}$ requires more attention since it relates successive levels. The updated $\bar{\mathbf{A}}$ is computed iteratively following a bottom–up scheme, from the lowest level $j = 1$ to the highest level $j = L$. The minimization task for $\mathbf{A}^j, j = 1, \ldots, L$, is shown in Fig. 3.

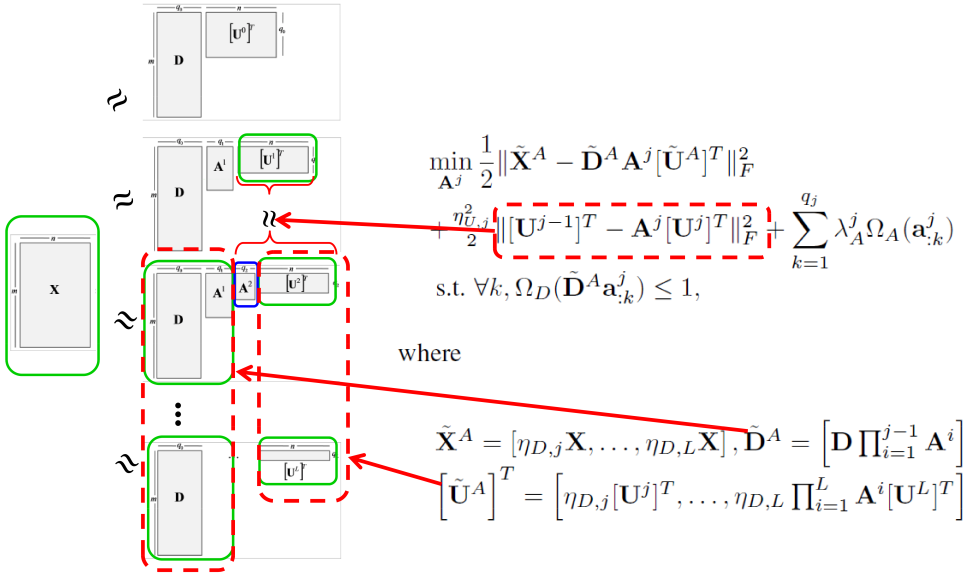


Figure 3: Joint optimization of the adjacency matrix $\mathbf{A}^j$ at level j fixing the lowest levels. From left to right: Illustration of matrix decomposition such that green boxes indicates fixed variables and blue the variable to estimate, and the main equation.

A natural way to update $\mathbf{A}^j$ would be to compute the closed form solution available by applying some properties of the Kronecker product $\otimes$ and matrix vectorization $v(\cdot)$ as follows:

$$\begin{aligned}\min_{\mathbf{A}^j} \frac{1}{2}\left\| \begin{bmatrix} v\left(\tilde{\mathbf{X}}^A\right) \\ v\left(\eta_{U,j}[\mathbf{U}^{j-1}]^T\right) \end{bmatrix} - \begin{bmatrix} \tilde{\mathbf{U}}^A \otimes \tilde{\mathbf{D}}^A \\ \mathbf{U}^j \otimes \eta_{U,j}\mathbf{I}_{q_{j-1}} \end{bmatrix} v(\mathbf{A}^j) \right\|_F^2 \\ + \sum_{k=1}^{q_j} \lambda_A^j \Omega_A(\mathbf{a}_{:k}^j) \text{ s.t. } \forall k, \Omega_D(\tilde{\mathbf{D}}^A \mathbf{a}_{:k}^j) \leq 1.\end{aligned} \tag{2}$$

However, the solution of (2) would require to solve large–scale optimization problem due to the Kronecker product. Instead, we consider the optimization on the rows $\mathbf{a}_{k:}^j$ of $\mathbf{A}^j$ via Block Coordinate Descent (BCD) [2, Section 2.7]. Assuming that the others $\{\mathbf{a}_{i:}^j\}_{i \neq k}$ are kept fixed, some basic algebra and Lemma 1.1 leads to

$$\begin{aligned}\min_{\mathbf{a}_{k:}^j} \frac{1}{2}\left\| \begin{bmatrix} \frac{[\mathbf{E}^A]^T \tilde{\mathbf{d}}_{:k}^A}{\|\tilde{\mathbf{d}}_{:k}^A\|_2^2} \\ \frac{\eta_{U,j}\mathbf{U}^{j-1}}{\|\tilde{\mathbf{d}}_{:k}^A\|_2^2} \end{bmatrix} - \begin{bmatrix} \tilde{\mathbf{U}}^A \\ \frac{\eta_{U,j}\mathbf{U}^j}{\|\tilde{\mathbf{d}}_{:k}^A\|_2^2} \end{bmatrix} [\mathbf{a}_{k:}^j]^T \right\|_F^2 + \frac{\lambda_A^j}{\|\tilde{\mathbf{d}}_{k:}^A\|_2^2}\Omega_A(\mathbf{a}_{k:}^j) \\ \text{s.t. } \forall i, \Omega_D(\tilde{\mathbf{D}}^A \mathbf{a}_{:i}^j) \leq 1,\end{aligned} \tag{3}$$

where

$$\mathbf{E}^A = \tilde{\mathbf{X}}^A - \sum_{q \neq k} \tilde{\mathbf{d}}_{:q}^A \mathbf{a}_{k:}^j [\tilde{\mathbf{U}}^A]^T.$$

Fast algorithms are available for efficiently solving this problem with a subsequent Euclidean projection of $\tilde{\mathbf{D}}^A \mathbf{a}_{:k}^j$ onto the unit ball of Ω_D. Note that the norm Ω_A in Eq. (3) is applied to the row vectors instead of column vectors of $\mathbf{A}$. For regularizations such as l_1-norm, this is equivalent; for many others, it can be reformulated using the variational equality [4, Lemma 3.1] to a weighted quadratic problem. In addition, this formulation allows to impose an inclusion hierarchy between levels by constraining the number of non–zeros elements of $\mathbf{a}_{k:}^j$ to be lower than one.

Lemma 1.1. *Let $V \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times q}$ be two matrices, and let $y \in \mathbb{R}^m$ and $w \in \mathbb{R}^q$ be two vectors of respective sizes. Also assume that $y^T y > 0$. Then the following holds:*

$$\underset{w}{\operatorname{argmin}} \frac{1}{2}\left\|V - yw^T B^T\right\|_F^2 + \lambda\Omega(w) = \underset{w}{\operatorname{argmin}} \frac{1}{2}\left\|\frac{V^T y}{\|y\|_2^2} - Bw\right\|_2^2 + \frac{\lambda}{\|y\|_2^2}\Omega(w).$$

Proof: See the appendix.

Update of D: The update of the dictionary $\mathbf{D}$ is again a convex problem when fixing the other variables. This can be written as shown in Fig. 4. This formulation allows to learn the lowest dictionary elements $\mathbf{D}$ using the information contained in all the sparse representations and the latent heterarchical structure between consecutive levels. Instead of updating $\mathbf{D}$ by computing the closed form solutions available for each row vector, the update of $\mathbf{D}$ follows the technique suggested by Mairal *et al*. [5]. This is again a BCD scheme on the columns of $\mathbf{D}$ and a Euclidean projection onto the unit ball of Ω_D. Hence, it avoids the computation of q_0 non–diagonal matrix inversions. In practice, the stopping criterion relies on the relative decrease (typically 10^{-3}) in the cost function shown in Fig. 1 (Fig. 3(d) in the main paper).

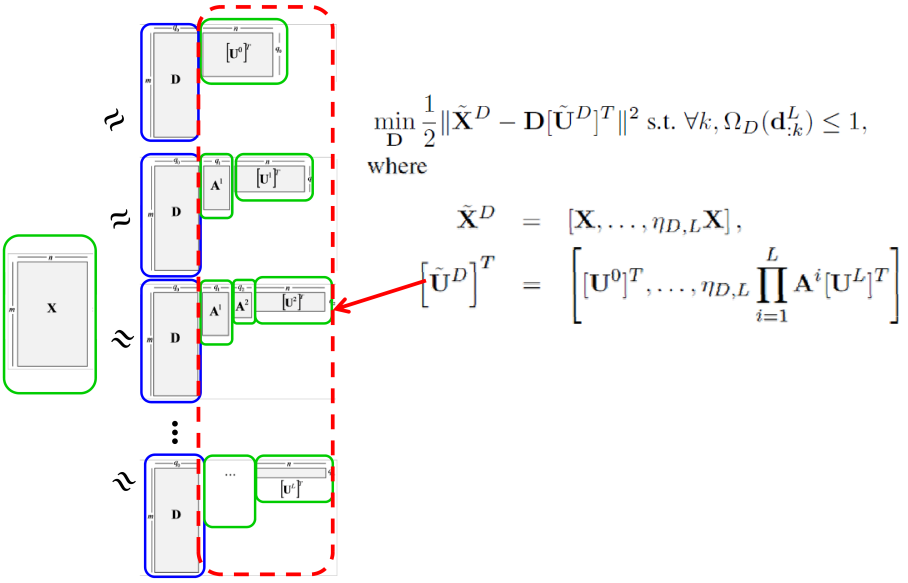


Figure 4: Joint optimization of the dictionary matrix $\mathbf{D}$. From left to right: Illustration of matrix decomposition such that green boxes indicates fixed variables and blue the variable to estimate, and the main equation.

Algorithm 1 Learning Heterarchical Structure

Input: Data matrix $\mathbf{X}$, number of levels L, dictionary sizes q_j, regularization parameters, the norms Ω_U, Ω_A and Ω_D, and η.
Initialization: Initialization of $\mathbf{D}$ and $\bar{\mathbf{U}}$ (possibly random), and $\mathbf{A}^j \leftarrow \mathbf{I} \in \mathbb{R}^{q_{j-1} \times q_j}$.
while (not converged) **do**
    Update $\mathbf{U}$: Compute $\bar{\mathbf{U}}$ that minimizes shown in Fig. 2.
    Update $\mathbf{A}$ by BCD:
    for $j = 1, \ldots, L$ **do**
        for $k = 1, \ldots, q_j$ **do**
            Compute $a^j_{k:}$ that minimizes Eq. (3).
        end for
    end for
    Update $\mathbf{D}$: Compute $\mathbf{D}$ that minimizes shown in Fig. 4.
end while
Output: Decomposition $\mathbf{D}$, $\bar{\mathbf{U}}$ and $\bar{\mathbf{A}}$.

2 Implementation Details

2.1 Parameters

The trade-off parameters are chosen depending on the structure of the problem, and also their confidence. The main parameters are the weights of the data fidelity and temporal consistency. Depending on their values, the algorithm relies only on the data fidelity as KSVDS by setting $\eta_U = 0$, or on the temporal consistency as MNNMF by setting $\eta_D = 0$. Both terms are needed because the temporal consistent indicates when neurons fire and the raw data indicates the intensity of the firing. In our experiments, we set up η_D bigger than η_U because neurons are not firing exactly at the same time.

Regarding the sparsity parameters, λ_U should increase monotonically while layers increase since the number of active signals will be higher at lower levels, and λ_A should be a lower value (0.0001) to prevent A from shrinking to zero at the first iterations.

2.2 Computational Complexity

The problem in Fig. 1 is not jointly convex as the simplest case of dictionary learning or non-negative matrix factorization. Due to this non-convex formulation, the simplest cases are only able to prove the convergence to a stable point (or local minima) as indicated in [6] and the convergence rate depends highly on the initialization of the matrices as pointed in [1]. Empirically, the algorithm converges in 80 iterations on average. Despite estimating all parameters, the computational time is similar to KSVDS and MNNMF that only rely on estimating the upper layer. All experiments were conducted on the same conditions and run it in MATLAB.

The complexity of only the first layer depends on the update of D and U. The former based on [10] is $\mathcal{O}(m \cdot q_o \cdot \max(m^{0.5}, n) + m^2 + \max(n, m) \cdot q_0^2)$ and the latter depends on the solver. For instance, orthogonal matching pursuit is $\mathcal{O}(n \cdot q_0 \cdot (m + L^2))$, where L is the number of non-zero elements. For multiple layers this will increase the complexity approx. by a multiplicative factor of l^2 assuming that all q_l are equal to q_0.

2.3 Initialization

Bilevel SHMF can also be initialized randomly achieving similar performance to the other methods, but slightly worse (approx. 3%) than when the algorithm is initialized with the results obtained by ADINA [3]. For a fair comparison, the algorithms were compared using the same conditions.

3 Results

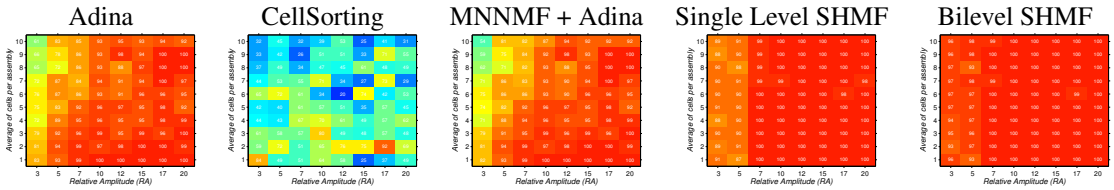
3.1 Artificial Sequences

Fig. 6 shows the ground truth at the neuron level for the synthetic data with 5 cells on average per assembly and a RA equal to 3. In addition, our bilevel SHMF is able to identify and monitor neuronal activity at single cell and assembly level as shown in Fig. 7. It can also infer the assignment matrix behind the neuronal activity, and is able to distinguish highly correlated cells. It extracts smooth and regularly shaped cells. In addition, our approach is able to learn the adjacency matrix relating neurons and neuronal assemblies.

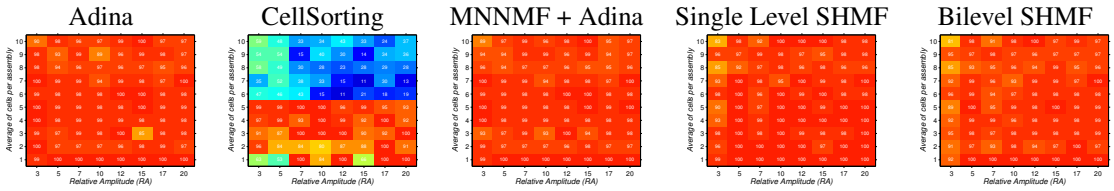
Fig. 5 shows the performance for different noise levels and the average number of cells per assembly. This figure complements the overall performance among the algorithms shown in Fig. 5 in the main manuscript.

Identification of assemblies
The adjacency matrices are binarized as follows to give the best performance:

- KSVDS: entries bigger than 5% of the maximum value in $\mathbf{A}$ are set to 1,
- MNNMF: entries bigger than 0.001 are set to 1, and
- bilevel SHMF: entries bigger than 0 are set to 1 since λ_A is set to 0.001.



(a) Sensitiviy (%)



(b) Precision (%)

Figure 5: Sensitivity and precision of transient detection for individual neuron. Methods that estimate both assemblies and neuron transients perform at least as well as their simpler counterparts that focus on the latter.

3.2 Real Sequences

As shown in Fig. 8, our approach is able to accurately reconstruct the raw data at both levels of representations, and to identify the underlying neurons and assemblies.

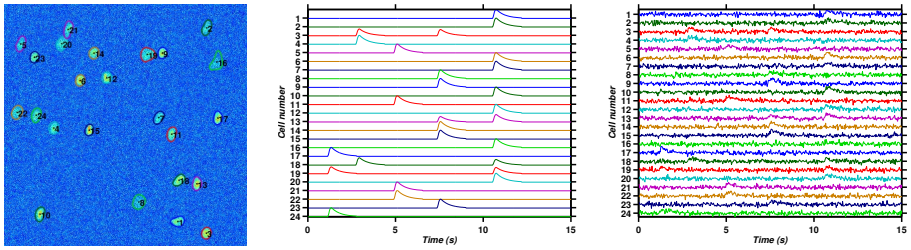


Figure 6: Example of ground truth at neuron level for the synthetic data. From left to right: outline of all neurons used to generate the artificial sequence superimposed on a maximum intensity projection across the noisy image sequence, the induced temporal activation patterns of each individual neuron before and after introducing random Gaussian noise.

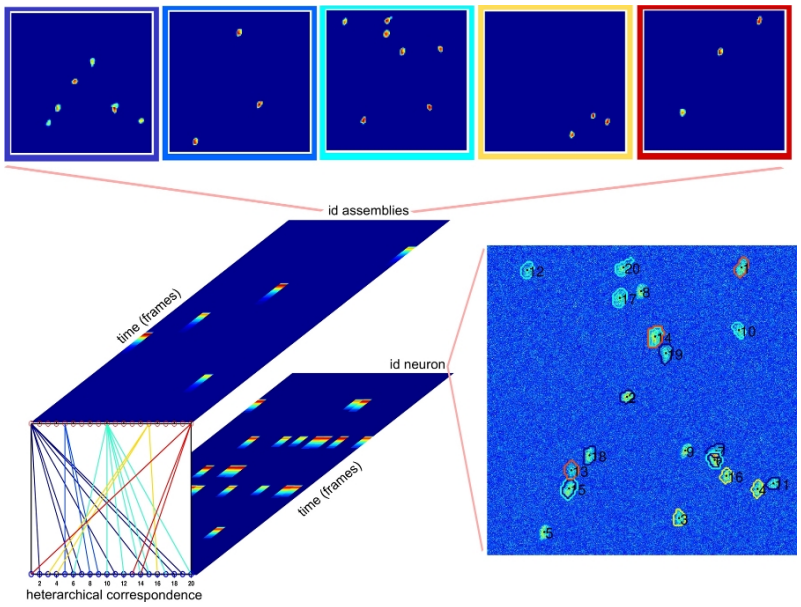


Figure 7: Bottom left: Shown are the temporal activation patterns of individual neurons (lower layer), and assemblies of neurons (upper layer). Neurons and assemblies are related by a bipartite graph the estimation of which is a central goal of this work. The signature of five neuronal assemblies in the spatial domain is shown at the top. The bottom right shows the outline of all found neurons superimposed on a maximum intensity projection across the image sequence. All results shown in this figure issue from computations on a synthetic sequence, with known ground truth.

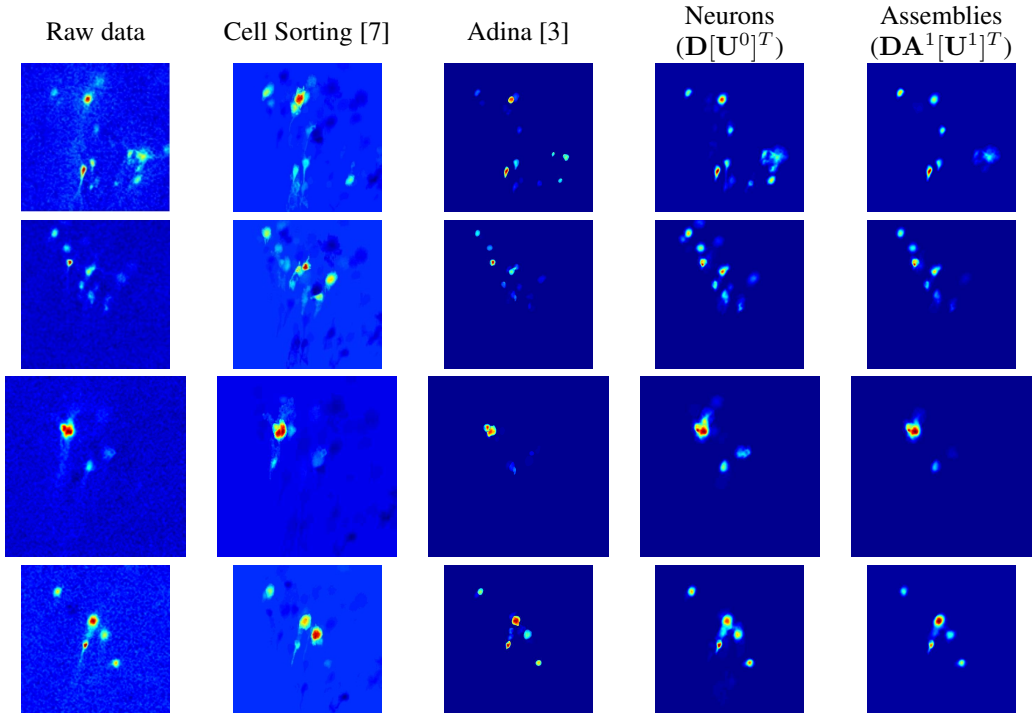


Figure 8: Raw data and reconstructed images at times indicated in Fig. 2(b) of the main paper.

References

[1] R. Albright, J. Cox, D. Duling, A. N. Langville, and C. D. Meyer. Algorithms, initializations, and convergence for the nonnegative matrix factorization. *NCSU Technical Report Math.*, 2006.

[2] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[3] F. Diego, S. Reichinnek, M. Both, and F. A. Hamprecht. Automated identification of neuronal activity from calcium imaging by sparse dictionary learning. In *International Symposium on Biomedical Imaging, in press*, 2013.

[4] R. Jenatton, G. Obozinski, and F. Bach. Structured sparse principal component analysis. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

[5] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009.

[6] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online Learning for Matrix Factorization and Sparse Coding. *Journal of Machine Learning Research*, 2010.

[7] E. A. Mukamel, A. Nimmerjahn, and M. J. Schnitzer. Automated analysis of cellular signals from large-scale calcium imaging data. *Neuron*, 2009.

4 Appendix

Lemma 4.1. *Let $\mathbf{V} \in \mathbb{R}^{m\times n}$ and $\mathbf{B} \in \mathbb{R}^{n\times q}$ be two matrices, and let $\mathbf{y} \in \mathbb{R}^m$ and $\mathbf{w} \in \mathbb{R}^q$ be two vectors of respective sizes. Also assume that $\mathbf{y}^T\mathbf{y} > 0$. Then the following holds:*

$$\operatorname*{argmin}_{\mathbf{w}} \frac{1}{2}\left\|\mathbf{V} - \mathbf{y}\mathbf{w}^T\mathbf{B}^T\right\|_F^2 + \lambda\Omega(\mathbf{w}) = \operatorname*{argmin}_{\mathbf{w}} \frac{1}{2}\left\|\frac{\mathbf{V}^T\mathbf{y}}{\|\mathbf{y}\|_2^2} - \mathbf{B}\mathbf{w}\right\|_F^2 + \frac{\lambda}{\|\mathbf{y}\|_2^2}\Omega(\mathbf{w}).$$

Proof. The equality follows from elementary properties of the trace function:

$$\begin{aligned}
& \operatorname*{argmin}_{\mathbf{w}} \frac{1}{2}\left\|\mathbf{V} - \mathbf{y}\mathbf{w}^T\mathbf{B}^T\right\|_F^2 + \lambda\Omega(\mathbf{w}) = \\
= \; & \operatorname*{argmin}_{\mathbf{w}} \frac{1}{2} Tr\left((\mathbf{V} - \mathbf{y}\mathbf{w}^T\mathbf{B}^T)^T(\mathbf{V} - \mathbf{y}\mathbf{w}^T\mathbf{B}^T)\right) + \lambda\Omega(\mathbf{w}) \\
= \; & \operatorname*{argmin}_{\mathbf{w}} \frac{1}{2}\left[Tr(\mathbf{V}^T\mathbf{V}) - 2Tr(\mathbf{V}^T\mathbf{y}\mathbf{w}^T\mathbf{B}^T) + Tr(\mathbf{B}\mathbf{w}\mathbf{y}^T\mathbf{y}\mathbf{w}^T\mathbf{B}^T)\right] + \lambda\Omega(\mathbf{w}) \\
= \; & \operatorname*{argmin}_{\mathbf{w}} \frac{1}{2}\left[Tr(\mathbf{V}^T\mathbf{V}) - 2Tr(\mathbf{V}^T\mathbf{y}\mathbf{w}^T\mathbf{B}^T) + Tr(\mathbf{B}\mathbf{w}\|\mathbf{y}\|_2^2\mathbf{w}^T\mathbf{B}^T)\right] + \lambda\Omega(\mathbf{w}) \\
= \; & \operatorname*{argmin}_{\mathbf{w}} \frac{1}{2}\left[\frac{1}{\|\mathbf{y}\|_2^2}Tr(\mathbf{V}^T\mathbf{V}) - 2Tr(\frac{\mathbf{V}^T\mathbf{y}}{\|\mathbf{y}\|_2^2}\mathbf{w}^T\mathbf{B}^T) + Tr(\mathbf{B}\mathbf{w}\mathbf{w}^T\mathbf{B}^T)\right] + \frac{\lambda}{\|\mathbf{y}\|_2^2}\Omega(\mathbf{w}) \\
= \; & \operatorname*{argmin}_{\mathbf{w}} \frac{1}{2}\left[\frac{1}{\|\mathbf{y}\|_2^2}Tr(\mathbf{V}^T\mathbf{V}) - 2Tr(\frac{\mathbf{V}^T\mathbf{y}}{\|\mathbf{y}\|_2^2}\mathbf{w}^T\mathbf{B}^T) + Tr(\mathbf{B}\mathbf{w}\mathbf{w}^T\mathbf{B}^T) + Tr(\frac{\mathbf{V}^T\mathbf{y}}{\|\mathbf{y}\|_2^2}\frac{\mathbf{y}^T\mathbf{V}}{\|\mathbf{y}\|_2^2})\right. \\
& \left. -Tr(\frac{\mathbf{V}^T\mathbf{y}}{\|\mathbf{y}\|_2^2}\frac{\mathbf{y}^T\mathbf{V}}{\|\mathbf{y}\|_2^2})\right] + \frac{\lambda}{\|\mathbf{y}\|_2^2}\Omega(\mathbf{w}) \\
= \; & \operatorname*{argmin}_{\mathbf{w}} \frac{1}{2}\left[Tr(\frac{\mathbf{V}^T\mathbf{y}}{\|\mathbf{y}\|_2^2}\frac{\mathbf{y}^T\mathbf{V}}{\|\mathbf{y}\|_2^2}) - 2Tr(\frac{\mathbf{V}^T\mathbf{y}}{\|\mathbf{y}\|_2^2}\mathbf{w}^T\mathbf{B}^T) + Tr(\mathbf{B}\mathbf{w}\mathbf{w}^T\mathbf{B}^T)\right] + \frac{\lambda}{\|\mathbf{y}\|_2^2}\Omega(\mathbf{w}) \\
& + \frac{1}{\|\mathbf{y}\|_2^2}Tr(\mathbf{V}^T\mathbf{V}) - Tr(\frac{\mathbf{V}^T\mathbf{y}}{\|\mathbf{y}\|_2^2}\frac{\mathbf{y}^T\mathbf{V}}{\|\mathbf{y}\|_2^2}) \\
= \; & \operatorname*{argmin}_{\mathbf{w}} \frac{1}{2}\left[Tr(\frac{\mathbf{V}^T\mathbf{y}}{\|\mathbf{y}\|_2^2}\frac{\mathbf{y}^T\mathbf{V}}{\|\mathbf{y}\|_2^2}) - 2Tr(\frac{\mathbf{V}^T\mathbf{y}}{\|\mathbf{y}\|_2^2}\mathbf{w}^T\mathbf{B}^T) + Tr(\mathbf{B}\mathbf{w}\mathbf{w}^T\mathbf{B}^T)\right] + \frac{\lambda}{\|\mathbf{y}\|_2^2}\Omega(\mathbf{w}) \\
= \; & \operatorname*{argmin}_{\mathbf{w}} \frac{1}{2}\left\|\frac{\mathbf{y}^T\mathbf{V}}{\|\mathbf{y}\|_2^2} - \mathbf{w}^T\mathbf{B}^T\right\|_F^2 + \frac{\lambda}{\|\mathbf{y}\|_2^2}\Omega(\mathbf{w}) = \operatorname*{argmin}_{\mathbf{w}} \frac{1}{2}\left\|\frac{\mathbf{V}^T\mathbf{y}}{\|\mathbf{y}\|_2^2} - \mathbf{B}\mathbf{w}\right\|_F^2 + \frac{\lambda}{\|\mathbf{y}\|_2^2}\Omega(\mathbf{w}).
\end{aligned}$$

□