A Local Block Coordinate Descent Algorithm for the CSC Model Supplementary Material

Ev Zisselman
ev_zis@campus.technion.ac.il

Jeremias Sulam
jsulam1@jhu.edu

Michael Elad
elad@cs.technion.ac.il

This Supplementary material elaborates on the LoBCoD algorithm and our experiments. In Section A we provide proofs for our derivations of the LoBCoD algorithm and its stochastic version. Section B analyzes the complexity of our approach and compares it to contending batch and online algorithms. Section C details the algorithm for image fusion and in Section D we specify the implementation details and parameters settings of our experiments. Lastly, in Section E we demonstrate the advantages of LoBCoD on two additional applications: multi-exposure image fusion and salt-and-pepper text image denosing.

A. Mathematical proofs

A.1. Transitioning from a high dimensional problem to a low dimensional problem

Transition from minimization problem (7) to (8) in the main paper. We denote p_i as the patch that fully contains the slice $s_i = \mathbf{D}_L\alpha_i$, and define a patch-layer $L_{\tilde{i}}$ as the set of non-overlapping patches taken from the image that contains the patch p_i. We can write the identity matrix as a sum of non-overlapping patch-extraction matrices $\sum_{k\in L_{\tilde{i}}}\mathbf{P}_k^T\mathbf{P}_k = \mathbf{I}$, where $\mathbf{P}_i$ is one of these matrices. By using these definitions and writing the definition of R_i (the residual image without the contribution of the i-th needle) we can write the l_2 fidelity term of Equation (7) as

$$\begin{aligned}
&\arg\min_{\alpha_i}\frac{1}{2}\|\sum_{k\in L_{\tilde{i}}}\mathbf{P}_k^T\mathbf{P}_kR_i - \mathbf{P}_i^T\mathbf{D}_L\alpha_i\|_2^2 = \\
&\arg\min_{\alpha_i}\frac{1}{2}\|\sum_{k\in L_{\tilde{i}}}\mathbf{P}_k^T\mathbf{P}_k(X-\sum_{\substack{j=1\\j\neq i}}^{N}\mathbf{P}_j^T\mathbf{D}_L\alpha_j) - \mathbf{P}_i^T\mathbf{D}_L\alpha_i\|_2^2 = \\
&\arg\min_{\alpha_i}\frac{1}{2}\|\sum_{\substack{k\neq i\\k\in L_{\tilde{i}}}}\mathbf{P}_k^T\mathbf{P}_k(X-\sum_{\substack{j=1\\j\neq i}}^{N}\mathbf{P}_j^T\mathbf{D}_L\alpha_j) + \\
&\qquad\mathbf{P}_i^T(\mathbf{P}_i(X-\sum_{\substack{j=1\\j\neq i}}^{N}\mathbf{P}_j^T\mathbf{D}_L\alpha_j) - \mathbf{D}_L\alpha_i)\|_2^2.
\end{aligned} \tag{A.1}$$

Since the patch-extraction matrix $\mathbf{P}_i^T$ is orthogonal to all the matrices $\mathbf{P}_k^T$ for $k\neq i$, the previous problem (A.1) is equal to

$$\begin{aligned}
\arg\min_{\alpha_i}\frac{1}{2}\|\sum_{\substack{k\in L_{\tilde{i}}\\k\neq i}}\mathbf{P}_k^T\mathbf{P}_k(X-\sum_{\substack{j=1\\j\neq i}}^{N}\mathbf{P}_j^T\mathbf{D}_L\alpha_j)\|_2^2 + \\
\|\mathbf{P}_i^T(\mathbf{P}_i(X-\sum_{\substack{j=1\\j\neq i}}^{N}\mathbf{P}_j^T\mathbf{D}_L\alpha_j) - \mathbf{D}_L\alpha_i)\|_2^2.
\end{aligned} \tag{A.2}$$

Note that the first term of the above objective does not depend on α_i, and thus we can ignore this term in our minimization of the objective:

$$\arg\min_{\alpha_i}\frac{1}{2}\|\mathbf{P}_i^T(\mathbf{P}_i(X-\sum_{\substack{j=1\\j\neq i}}^{N}\mathbf{P}_j^T\mathbf{D}_L\alpha_j) - \mathbf{D}_L\alpha_i)\|_2^2. \tag{A.3}$$

In addition, the matrix $\mathbf{P}_i^T$ translates a patch-size vector to the i-th position in the global vector padded with zeros. Hence, we can ignore all the zero-entries in the resulting vector, and the problem becomes equivalent to solving the following reduced minimization problem:

$$\begin{aligned}
&\arg\min_{\alpha_i}\frac{1}{2}\|\mathbf{P}_i(X-\sum_{\substack{j=1\\j\neq i}}^{N}\mathbf{P}_j^T\mathbf{D}_L\alpha_j) - \mathbf{D}_L\alpha_i\|_2^2 = \\
&\qquad\arg\min_{\alpha_i}\frac{1}{2}\|\mathbf{P}_iR_i - \mathbf{D}_L\alpha_i\|_2^2,
\end{aligned} \tag{A.4}$$

where $R_i = X - \sum_{\substack{j=1\\j\neq i}}^{N}\mathbf{P}_j^T\mathbf{D}_L\alpha_j$.

A.2. Local dictionary update - gradient calculation

The derivation of the gradient, Equation (11) in the main paper. We can rewrite $\mathbf{D}_L$ as a column vector d_L and write problem (10) as:

$$\arg\min_{\mathbf{D}_L}\frac{1}{2}\|X-\sum_{i=1}^{N}\mathbf{P}_i^T(\alpha_i\otimes\mathbf{I}_{n\times n})d_L\|_2^2, \tag{A.5}$$

where we denote $\otimes$ as the Kronecker matrix product and apply property no. (40) from [9].
By defining $\mathbf{A}_i = (\alpha_i \otimes \mathbf{I}_{n\times n})$, the above problem can be rewritten as

$$\arg\min_{\mathbf{D}_L} \frac{1}{2}\|X - \sum_{i=1}^{N} \mathbf{P}_i^T \mathbf{A}_i d_L\|_2^2. \tag{A.6}$$

Now it easy to see that the gradient of problem (A.6) w.r.t. d_L is given by

$$\begin{aligned}\nabla_{d_L} = -(\sum_{i=1}^{N} \mathbf{P}_i^T \mathbf{A}_i)^T (X - \sum_{i=1}^{N} \mathbf{P}_i^T \mathbf{A}_i d_L) = \\ -\sum_{i=1}^{N} \mathbf{A}_i^T \mathbf{P}_i (X - \sum_{i=1}^{N} \mathbf{P}_i^T \mathbf{A}_i d_L) = \\ -\sum_{i=1}^{N} \mathbf{A}_i^T \mathbf{P}_i (X - \widehat{X}).\end{aligned} \tag{A.7}$$

Note that $\mathbf{P}_i(X - \widehat{X})$ is the i-th patch of the residual image, and by substituting back the definition of $\mathbf{A}_i$, and using the same property as before (see [9] property no. (40)) we can write the gradient as

$$\begin{aligned}\nabla_{d_L} = -\sum_{i=1}^{N} (\alpha_i \otimes \mathbf{I}_{n\times n})^T \mathbf{P}_i (X - \widehat{X}) = \\ -\sum_{i=1}^{N} vec(\mathbf{P}_i (X - \widehat{X}) \cdot \alpha_i^T),\end{aligned} \tag{A.8}$$

where $vec(\cdot)$ denotes the vec-operator that stacks the columns of the gradient matrix into a vector, so by reshaping the above expression we get the final expression for the gradient (11)

$$\nabla_{\mathbf{D}_L} = -\sum_{i=1}^{N} \mathbf{P}_i (X - \widehat{X}) \cdot \alpha_i^T. \tag{A.9}$$

B. Complexity Analysis

This section details the computational complexity analysis of our algorithm and compares it to previous methods.

We assume that the number of the non-zeros in every needle is limited to at most k non-zeros, and we denote by I the number of the training images. We evaluate the complexity of every outer iteration (single epoch) of our algorithm and compare it to the complexity of executing an epoch in the alternative algorithms. Every inner iteration of our algorithm (Algorithm 1 in the main paper) operates on a layer of N/n needles, while the global iteration operates on the whole dataset. The resulting computational cost of the residuals R_j for all the layers is $O(IN(nk+n))$, which is comprised of $O(INnk)$ for computing all the N slices $\mathbf{D}_L\alpha_i$ of all the I images, and $O(INn)$ computations for subtracting them from the global residual. Given the residuals, every iteration applies the LARS algorithm for solving the local sparse pursuit for all the NI needles, requiring $O(k^3 + mk^2 + nm)$ per needle [8], and $O(IN(k^3 + mk^2 + nm) + n^2m^2)$ computations for all the N needles in all the I images. The latter term, n^2m^2, corresponds to the precomputation of the Gram matrix of the dictionary $\mathbf{D}_L$ at every layer, which is usually negligible since it is computed once for all the needle in the layer. Next, we evaluate the complexity of reconstructing the signal. Direct computation requires $O(IN(nk+n+k))$ operations, which is effectively $O(INnk)$, since this is the dominant term. The computation of the global residual requires another $O(IN)$ operations. These last two phases are negligible compared to the sparse pursuit stage, and are therefore omitted from the final expression. Finally, the computation of the gradient is $O(IN(n+nk))$, and the dictionary update stage is $O(INm)$ (updating the dictionary requires $O(mn)$ computations and occurs IN/n times in every epoch). We summarize the above analysis in Table 1 in the main paper, and compare it to the complexity analysis of the SBDL algorithm [10]. In addition, Table 1 presents the complexity of executing an epoch of the two SGD based online algorithms that were introduced in [6], where q corresponds to the number of inner iterations of the sparse pursuit stage (preformed in the Fourier domain).

The most demanding stage, in both the SBDL algorithm and in our approach, is the local sparse pursuit, which is $O(IN(k^3 + mk^2 + nm))$. Assuming that the needles are very sparse $k \ll m$, which is often the case with real-world signals, the complexity of the local sparse pursuit stage is governed by $O(NInm)$ in both algorithms. This implies that our algorithm is of comparable order of complexity[1] as SBDL, which is a batch algorithm. On the other hand, the complexity of the online algorithm in [6] is dominated by the computation of the FFT in the pursuit stage, which is $O(qImNlog(N))$, meaning that their algorithm scales as $O(Nlog(N))$ with the global dimension of the signals, while our algorithm grows linearly.

C. Multi-focus

In this section we address the task of multi-focus image fusion and further detail our implementation.

We denote the set of source images to fuse as $\{Y^k\}_{k=1}^L$, and assume that a pretrained dictionary $\{d_i\}_{i=1}^m$ is provided. Each image Y^k is decomposed into a base component Y_b^k, which is a smooth piece-wise constant image, and an edge

[1] While the complexity of our algorithm is of the same order as the complexity of SBDL, we empirically demonstrated that our method converges faster (see Figure 2 in the main paper).

component Y_e^k that contains the high frequency elements:

$$Y^k = Y_b^k + Y_e^k, \tag{C.1}$$

where the separation is performed by means of applying distinctive priors. The base component is usually extracted by imposing a prior which penalizes the l_2 norm of its gradient. Modeling the edge component, however, is more involved and has been the subject matter of many image-processing algorithms [3, 5, 13, 14, 16, 17]. In this work, we employ the CSC model to describe the edge components, as it has shown promising results in [7].

Using the aforementioned priors, the separation of the image to its components amounts to solving the following optimization problem:

$$\min_{\Gamma_e^k, Y_b^k} \frac{1}{2}\|Y^k - \mathbf{D}_e\Gamma_e^k - Y_b^k\|_2^2 + \lambda\|\Gamma_e^k\|_1 + \mu\frac{1}{2}\|\nabla Y_b^k\|_2^2, \tag{C.2}$$

where Γ_e^k is the sparse representation of Y_e^k, under the given convolutional dictionary $\mathbf{D}_e$, i.e. $Y_e^k = \mathbf{D}_e\Gamma_e^k$, and $\|\nabla Y_b^k\|_2^2$ is given by

$$\|\nabla Y_b^k\|_2^2 = \|g_x * Y_b^k\|_2^2 + \|g_y * Y_b^k\|_2^2, \tag{C.3}$$

where $g_x = [-1 \;\; 1]$ and $g_y = [-1 \;\; 1]^T$ are the horizontal and vertical gradient operators, respectively.

By taking similar steps to those presented in Section 3.1 in the main paper, we can rewrite the above optimization problem as

$$\begin{aligned}\min_{\{\alpha_j^k\}, Y_b^k} \frac{1}{2}\|Y^k - \sum_{j=1}^N \mathbf{P}_j^T\mathbf{D}_L\alpha_j^k - Y_b^k\|_2^2 \\ +\lambda\sum_{j=1}^N \|\alpha_j^k\|_1 + \mu\frac{1}{2}\|\nabla Y_b^k\|_2^2,\end{aligned} \tag{C.4}$$

where $\{\alpha_j^k\}_{j=1}^N$ are the needles which compose the sparse vector Γ_e^k, and $\mathbf{D}_L$ is the local dictionary of $\mathbf{D}_e$.

This problem can be solved by alternating between minimizing w.r.t. Y_b^k and Y_e^k, where the latter boils down to seeking for the sparse needles $\{\alpha_j^k\}_{j=1}^N$. To that end, the update rule of the $\{\alpha_j^k\}_{j=1}^N$ is the set of local pursuit problems:

$$\min_{\{\alpha_j^k\}} \frac{1}{2}\|(Y^k - Y_b^k) - \sum_{j=1}^N \mathbf{P}_j^T\mathbf{D}_L\alpha_j^k\|_2^2 + \lambda\sum_{j=1}^N \|\alpha_j^k\|_1, \tag{C.5}$$

which can be solved using our proposed algorithm, whereas the update rule of Y_b is the following least square minimization problem:

$$\min_{Y_b^k} \frac{1}{2}\|(Y^k - \mathbf{D}_e\Gamma_e^k) - Y_b^k\|_2^2 + \mu\frac{1}{2}\|\nabla Y_b^k\|_2^2. \tag{C.6}$$

For solving problem C.6, we set its gradient w.r.t. Y_b to zero to obtain the following update rule:

$$Y_b^k = (I + \mu(G_x^T G_x + G_y^T G_y))^{-1}(Y^k - \mathbf{D}_e\Gamma_e^k), \tag{C.7}$$

where G_x and G_y are the matrix representations of the gradient-operators.

Once these problems have been solved for all the input images $\{Y^k\}_{k=1}^L$, we aim to merge each set of feature maps[2] $\{Z_i^k\}_{k=1}^L$ in a way that best captures the focused objects in the resulting images. For each image Y^k, we generate an activity map based on the intensity of the l_1-norm of its feature maps. More specifically, we sum pixel-wise the absolute value of its m feature maps $\{Z_i^k\}_{i=1}^m$ to form an activity map that matches the size of the image N:

$$\tilde{\mathbf{A}}^k(u,v) = \sum_{i=1}^m \|Z_i^k(u,v)\|_1. \tag{C.8}$$

To make this method more robust and less susceptible to misregistration, we convolve the above activity maps with a uniform kernel U_s, of a small support $s \times s$, to produce the final activity maps:

$$\mathbf{A}^k = \tilde{\mathbf{A}}^k * U_s. \tag{C.9}$$

Based on the observation that a significant value in the activity map $\mathbf{A}^k$ indicates a sharp region in the image Y^k, we then reconstruct the all-in-focus edge component by selectively assembling the most prominent regions from the feature maps based on their pixel-wise values in the corresponding activity maps:

$$Z_i^f(u,v) = Z_i^{k^*}(u,v), \;\; k^* = \arg\max_k(\mathbf{A}^k(u,v)), \tag{C.10}$$

where $\{Z_i^f\}_{i=1}^m$ are the feature maps of the fused image.

Afterward, we fuse the base components, either by taking their average

$$Y_b^f = \frac{1}{N}\sum_{k=1}^L Y_b^k, \tag{C.11}$$

or by nominating regions of the base components according to the maximum value in the respective activity maps, i.e.

$$Y_b^f(u,v) = Y_b^{k^*}(u,v), \;\; k^* = \arg\max_k(\mathbf{A}^k(u,v)), \tag{C.12}$$

where Y_b^f is the base component of the fused image. Here, we opt for the latter since it produces better results.

Finally, the fusion result Y^f is obtained by gathering its components:

$$Y^f = Y_b^f + \sum_{i=1}^m d_i * Z_i^f. \tag{C.13}$$

[2]This set of feature maps refers to the i-th feature maps of the input images.

D. Experiments

In this section we turn to describe the implantation details of our experiments.

Throughout all our experiments we used a local dictionary composed of $m = 81$ filters, each of size 8×8. In addition, we used the LARS algorithm [1] for solving the local sparse pursuit stage (problem (8) in the main paper).

Run time comparison of the batch methods (Figure 2 in the main paper). We used $\lambda = 1$ and the Fruit dataset [2] for training the dictionaries. The dataset contains 10 images of size 100×100 pixels. As a preprocessing step, we mean-subtracted the images. The mean was computed by convolving each image with an 8×8 uniform kernel and subtracting the result from the original image. Additionally, we used the ADAM algorithm in the initial 30 iterations, with $\eta = 0.02$, and set the ADAM parameters in accordance with the authors' recommendation: $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^8$. Subsequent iterations applied the Momentum algorithm with $\eta = 10^{-7}$ and $\gamma = 0.8$ until convergence[3].

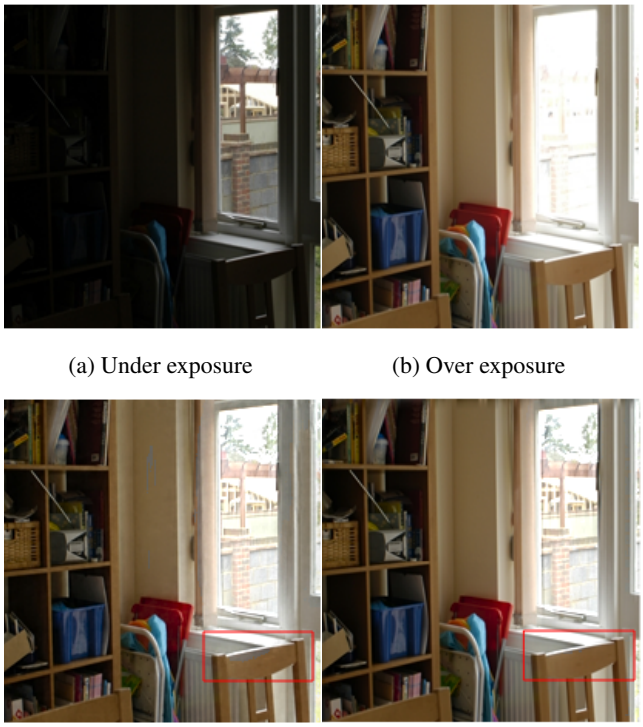
Run time comparison of the online algorithms (Figure 3 in the main paper). For training the dictionary we used the MIRFLICKR-1M dataset [4], where we sampled (without recurrence) 40 images for training and an additional 5 images for testing. The images were cropped to reduce their size from 512×512 to 256×256 pixels in both the training and testing sets to expedite the computation. In addition, we divided the images by 255 and mean-subtracted them as previously described. In this experiment, we used $\lambda = 0.1$ for all methods. For our method we set the learning rate to $\eta = 0.1$, with learning rate decay[4] of $\tilde{\eta} = \eta/(1 + 3/t)$ updated every 5 epochs. Lastly, we setup the momentum parameter $\gamma = 0.8$.

Multi-focus image fusion (Section 7.3 in the main paper). The dictionaries of both methods were pretrained on the Fruit dataset [2]. In addition, we set the coefficients described in Equation (C.2) to $\lambda = 1$ and $\mu = 5$ for the sparse pursuit (C.5) and the base-image extraction (C.6) stages, respectively, and alternate between the stages at each iteration. In practice, convergence was achieved within 2-4 iterations of alternating between these two stages. For reconstructing the image Barbara we used a reconstruction kernel U_s (Equation (C.9)) of size 9×9, and for the image Butterfly we used a reconstruction kernel U_s of size 8×8.

For reconstructing the colored image Bird in the Lab color space, we built the activity maps A_k based on their L channel only, with kernel U_s of size of 14×14. Then, we reconstructed each channel from the Lab color space by selecting regions based on the maximum pixel-wise value of the activity maps. Finally for fusing the image-pair Clocks we used a kernel U_s of size 9×9.

[3] All our notations are in accordance with those presented in [12]

[4] t here denotes the numbers of the epochs.



(a) Under exposure (b) Over exposure

(c) Liu et al. [7] (d) Proposed

Figure 1: Multi-exposure fusion comparison of real images. The red boxes indicate artifact suppression characteristics of the proposed method.

E. Additional Applications

We turn to demonstrate the benefits of LoBCoD on two additional applications: Multi-exposure image fusion and salt-and-pepper noise removal from text images.

Multi-exposure image fusion. To showcase the versatility of the multi-focus LoBCoD application, we adapt it to the task of multi-exposure image-fusion. The fusion is preformed in the same manner as in the case of multi-focus fusion (Section C), with a slight modification of handling the base components. Here, we fuse the base components by taking their weighted sum:

$$Y_b^f = \sum_{k=1}^{L} a^k Y_b^k. \tag{E.1}$$

Method	PSNR [dB]
Noisy test set	13.37
Wohlberg [15] trained on clean training set	19.83
Plaut et al. [11] trained on clean training set	21.60
Proposed trained on clean training set	20.40
Wohlberg [15] trained on noisy test set	19.62
Plaut et al. [11] trained on noisy test set	22.30
Proposed trained on noisy test set	23.64

Table 1: Average PSNR comparison between our method and the algorithms presented in [11] and [15]. The noisy images are corrupted by converting 10% of the image-pixels.

All other implementation details, including the parameters λ and μ remain the same as in the multi-focus case. For fusing the image Window (Figure 1) we used weights of 0.9 and 0.1 for the over and under exposed images, respectively, as it yields a brighter combined image. For U_s we chose a reconstruction kernel of size of 6×6.

Figure 1 compares our results with the results of an adapted version of Liu et al. [7], demonstrating LoBCoD's superior performance.

Salt-and-pepper text image denoising. Salt-and-pepper noise, also known as impulse noise, is characterized by sparsely occurring white or black (maximum or minimum value) pixels in an image. Here, unlike the inpainting task, we do not assume prior knowledge of the noise mask and we rely solely on the sparse prior to reconstruct the corrupted image. Thus, for a given test image, we aim to solve the following minimization problem:

$$\min_{Z_i} \sum_{i=0}^{m} \|Z_i\|_1 \quad s.t \quad \|X - \sum_{i=0}^{m} d_i * Z_i\|_2^2 < \epsilon. \tag{E.2}$$

In problem E.2 we adopt the approach presented in [11, 15] and augment the dictionary $\{d_i\}_{i=1}^m$ with a single impulse filter $d_0 = [\,1\,0\,0\,...\,0\,]^T$ to represent the impulse noise. For solving problem (E.2), we use the Lagrangian formulation as described in the main paper but with a small modification of assigning a different λ for image reconstruction and noise representation. This is because the sparsity ratio of an image presented by the dictionary $\{d_i\}_{i=1}^m$ depends on the complexity of the image, while the sparsity ratio of the noise depends on the noise level (the amount of corrupted pixels). Thus, the objective for our minimization can be written as follows:

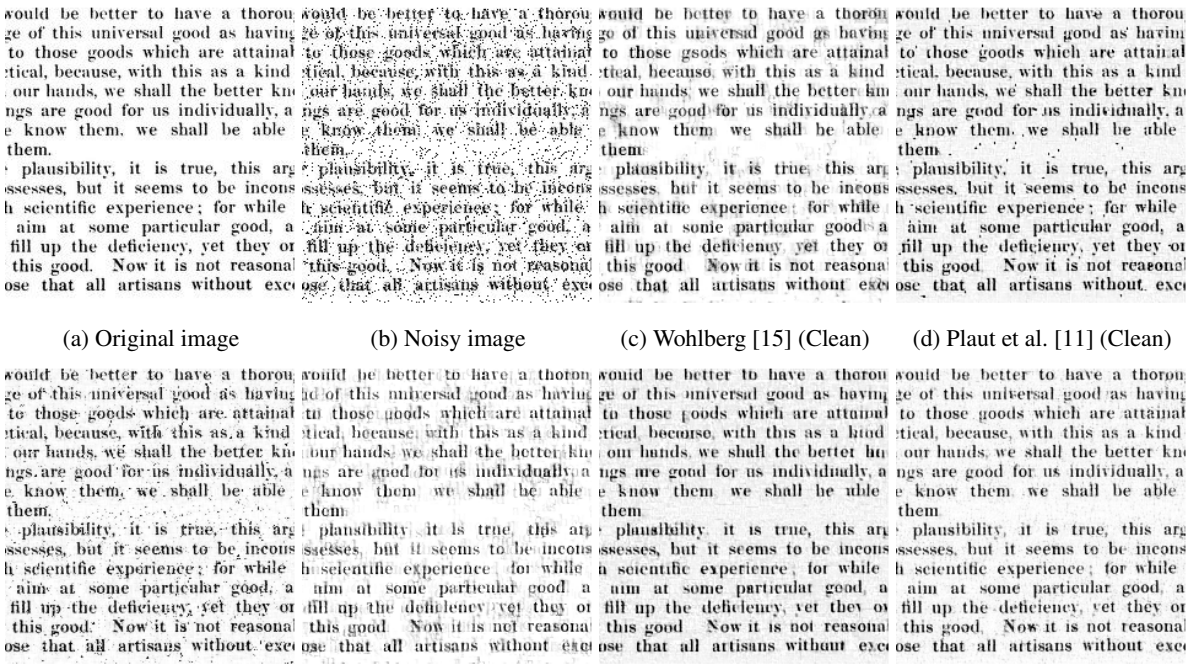
$$\begin{aligned} \min_{Z_i} \frac{1}{2}\|X - \sum_{i=1}^{m} d_i * Z_i - d_0 * Z_0\|_2^2 \\ +\lambda_1 \sum_{i=1}^{m} \|Z_i\|_1 + \lambda_0 \|Z_0\|_1. \end{aligned} \tag{E.3}$$

We minimize the objective (E.3) by splitting it into two minimization sub-problems, one minimizes over Z_0 leaving $\{Z_i\}_{i=1}^m$ fixed, and the other minimizes over $\{Z_i\}_{i=1}^m$ with Z_0 fixed. Then we alternate between these two minimization sub-problems for several iterations. In practice, 2-4 iterations were sufficient for convergence. For reconstructing the restored image, we use only the dictionary $\{d_i\}_{i=1}^m$ with its coefficients and zero the coefficient of the impulse atom.

We evaluate our proposed algorithm using text images taken from the dataset in [11], which is a subset of scanned pages taken from the book Aristotle's Nicomachean Ethics. Each training and test set are composed of 16 pages from the book, each of size 493×383 pixels. The images are gray scale images, normalized to the range $[0, 1]$. Since the the dataset assigns a value of zero to text characters and one to the image background, we employ a pre-processing step of inverting the images as described in [11], and uninvert the results to obtain black-over-white images at the end.

The images in the test set were corrupted by randomly inverting 10% of the pixels in every image, setting 5% of the pixels to black and 5% of the pixels to white. Figure 2(a) shows a portion of an original image taken from the test set and Figure 2(b) shows its corrupted version. We apply our algorithm to reconstruct the test set both in the case of training the dictionary on the clean training set and training on the noisy test images, and compare our results to the methods presented in [11] and [15] using their published code. In all evaluated methods, the dictionary was comprised of 100 atoms, each atom of size 11×11 pixels. Additionally, when training each method on the noisy test set, we include a pruning procedure of noisy atoms, as presented in [11].

Table 1 presents the resulting average PSNR of the different algorithms on the test set. The comparison illustrates that while [11] performs better when training on clean images, the proposed method achieves better results when training on the corrupted test images. Figure 2 compares the various algorithms when applied to a single test example. Here, the advantage of our approach is reiterated in the case of training using noisy test images.



(a) Original image (b) Noisy image (c) Wohlberg [15] (Clean) (d) Plaut et al. [11] (Clean)

(e) Proposed (Clean) (f) Wohlberg [15] (Noisy) (g) Plaut et al. [11] (Noisy) (h) Proposed (Noisy)

Figure 2: Denoising performance comparison on a text image. (a) the original image; (b) the noisy image. PSNR: 13.53; (c) denoising method [15], dictionary trained on clean images. PSNR: 19.23; (d) denoising method [11], dictionary trained on clean images. PSNR: 20.68; (e) denoising using the proposed algorithm, dictionary trained on clean images. PSNR: 20.25; (f) denoising method [15], dictionary trained on noisy images. PSNR: 18.51; (g) denoising method [11], dictionary trained on noisy images. PSNR: 19.91; (f) denoising using the proposed algorithm, dictionary trained on noisy images. PSNR: 22.81.

References

[1] B. Efron, T. iHastie, I. Johnstone, R. Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[2] R. Fergus, M. D. Zeiler, G. W. Taylor, and D. Krishnan. Deconvolutional networks. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition(CVPR)*, volume 00, pages 2528–2535, 2010.

[3] R. Gao and S. A. Vorobyov. Multi-focus image fusion via coupled sparse representation and dictionary learning. *arXiv preprint arXiv:1705.10574*, 2017.

[4] M. J. Huiskes, B. Thomee, and M. S. Lew. New trends and ideas in visual concept detection: the mir flickr retrieval evaluation initiative. In *Proceedings of the international conference on Multimedia information retrieval*, pages 527–536. ACM, 2010.

[5] H. Li, B. Manjunath, and S. K. Mitra. Multisensor image fusion using the wavelet transform. *Graphical models and image processing*, 57(3):235–245, 1995.

[6] J. Liu, C. Garcia-Cardona, B. Wohlberg, and W. Yin. First-and second-order methods for online convolutional dictionary learning. *SIAM Journal on Imaging Sciences*, 11(2):1589–1628, 2018.

[7] Y. Liu, X. Chen, R. K. Ward, and Z. J. Wang. Image fusion with convolutional sparse representation. *IEEE signal processing letters*, 23(12):1882–1886, 2016.

[8] J. Mairal, F. Bach, J. Ponce, et al. Sparse modeling for image and vision processing. *Foundations and Trends® in Computer Graphics and Vision*, 8(2-3):85–283, 2014.

[9] T. P. Minka. Old and new matrix algebra useful for statistics. *See www. stat. cmu. edu/minka/papers/matrix. html*, 2000.

[10] V. Papyan, Y. Romano, J. Sulam, and M. Elad. Convolutional dictionary learning via local processing. In *ICCV*, pages 5306–5314, 2017.

[11] E. Plaut and R. Giryes. A greedy approach to 0,-based convolutional sparse coding. *SIAM Journal on Imaging Sciences*, 12(1):186–210, 2019.

[12] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[13] S. Savić. Multifocus image fusion based on empirical mode decomposition. In *Twentieth International Electro technical and Computer Science Conference*, 2011.

[14] W. Wang and F. Chang. A multi-focus image fusion method based on laplacian pyramid. *JCP*, 6(12):2559–2566, 2011.

[15] B. Wohlberg. Convolutional sparse representations as an image model for impulse noise restoration. In *2016 IEEE 12th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*, pages 1–5. IEEE, 2016.

[16] B. Yang and S. Li. Multifocus image fusion and restoration with sparse representation. *IEEE Transactions on Instrumentation and Measurement*, 59(4):884–892, 2010.

[17] B. Yang and S. Li. Pixel-level image fusion with simultaneous orthogonal matching pursuit. *Information fusion*, 13(1):10–19, 2012.