Optimizing Codes for Source Separation in Compressed Video Recovery and Color Image Demosaicing

Alankar Kotwal[1] and Ajit V. Rajwade[2]

[1]Department of Electrical Engineering, [2]Department of Computer Science
Indian Institute of Technology Bombay

Abstract

There exist several applications in image processing (eg: video compressed sensing [8] and color image demosaicing) which require separation of constituent images given measurements in the form of a coded superposition of those images. Physically practical code patterns in these applications are non-negative and do not obey the nice coherence properties of other patterns such as Gaussian codes, which can adversely affect reconstruction performance. The contribution of this paper is to design code patterns for video compressed sensing and demosaicing by minimizing the mutual coherence given a fixed dictionary. Our method explicitly takes into account the special structure of those code patterns as required by these applications: (1) non-negativity, (2) block-diagonal nature, and (3) circular shifting. In particular, the last property enables for accurate patchwise reconstruction.

Keywords - video compressed sensing, source separation, sensing matrices, overlapping patchwise reconstruction, coherence, optimization, gradient descent

1 Introduction and Related Work

Compressed sensing has been explored as an alternative (usually, faster) way of sampling continuous-time signals. Its success with still images has inspired efforts to apply it to video. Indeed, [8] achieves compression across time by combining frames into coded snapshots while sensing and separating them with a pre-trained over-complete dictionary. [8] makes a diagonal choice for the sensing matrix. T vectorized input frames $\{X_i\}_{i=1}^T$ are sensed so that the vectorized output Y appears as a coded combination (dictated by the 'sensing matrices' ϕ_i) of the inputs. The sensing framework is

$$Y = \sum_{i=1}^{T} \phi_i X_i \tag{1}$$

The sparsifying basis here is a 3D dictionary learned on video patches. Given this dictionary, called D, any given signal X, and in particular, its frames $\{X_i\}_{i=1}^T$ can be approximately reconstructed as a sum of its projections α_j on the K atoms in D:

$$X_i = \sum_{j=1}^{K} D_{ji}\alpha_j \tag{2}$$

where D_{ji} is the i^{th} frame in the j^{th} 3-D dictionary atom D_{ji}. From the measurements and the dictionary, the input images are recovered solving the following optimization problem:

$$\min_{\alpha} \|\alpha\|_0 \text{ subject to } \left\| Y - \sum_{i=1}^{T} \phi_i \sum_{j=1}^{K} D_{ji}\alpha_j \right\|_2 \leq \epsilon \tag{3}$$

This problem can be approximately solved with sparse recovery techniques like orthogonal matching pursuit [1].

The drawback here, though, is that the 3D dictionary imposes a smoothness assumption on the scene. Since a linear combination of dictionary atoms cannot 'speed' an atom up, the typical speeds of objects moving in the video must be roughly the same as the dictionary. Also, because of the nature of the training data, the dictionary fails to sparsely represent sudden scene changes caused by, say, lighting or occlusion. Other techniques like [14] exploit additional structure within the signal, like periodicity, rigid motion or analytical motion models and cannot be used in the general video sensing case.

We try relaxing these constraints using a source-separation approach [13], where precise error bounds on the recovery of the images have been derived, with possible improvement using the techniques in [2]. Each of the coded snapshots is treated as a mixture of sources, each sparse in some basis. We experimented with basis pursuit recovery with Gaussian-random sensing matrices, getting excellent results with no visible ghosting for both similar and radically different images. Unfortunately, the more realizable positive sensing matrices do not have the nice incoherence properties of Gaussian-random matrices, which are sufficient conditions for near-accurate recovery as derived in [13].

We aim to design such sensing matrices with low mutual coherence, making them ideal for compressed video. Most current approaches to this problem have their limitations: the method in [5], for instance, involves a step that requires a Cholesky-type decomposition of a 'reduced' Gram matrix, and the non-linear reduction process is not guaranteed to keep the Gram matrix positive-semidefinite. Besides, the methods in both [4, 5, 11] optimize objective functions that are some forms of average of normalized dot products of effective dictionary columns, and minimizing averages doesn't guarantee minimizing the maximum (which is coherence in this case) of the quantities forming this average. Other than these, some authors have taken an information-theoretic route to this problem [3, 12, 15]. These papers design sensing matrices Φ such that the mutual information between a set of small patches $\{X_i\}_{i=1}^n$ and their corresponding projections $\{Y_i\}_{i=1}^n$ where $Y_i = \Phi X_i$, is maximized. Computing this mutual information first requires estimation of the probability density function of X and Y using Gaussian mixture models, for instance. This can be expensive and is an iterative process. Moreover these learned GMMs for a class of patches may not be general enough. Besides, the literature cited so far does not account for the special structure of the sensing matrices used for video compressed sensing as in [8] or for demosaicing, a framework which this paper expressly deals with.

There are obvious applications for this in the fields of fast video sensing and the general problem of coded source separation. Besides, this will find applications in improving multi-spectral imaging and image demosaicing, where inputs are coded linear combinations of images sparse in some domain and need to be solved for in a source-separation framework.

2 Method and Analysis

2.1 Our framework

We propose to use a recovery method different from the one used in [8], within the same acquisition framework. Thus, our signals are still acquired according to Eq. 1. However, the choice of the sparsifying basis is different: we use a DCT basis D to model each frame in the input data. The dictionary Ψ sparsifying the entire video sequence, thus, is a block-diagonal matrix with the $n \times n$ sparsifying basis D on the diagonal. Thus,

$$Y = \begin{pmatrix} \phi_1 & \dots & \phi_T \end{pmatrix} \begin{pmatrix} D\alpha_1 & \dots & D\alpha_T \end{pmatrix}^T \tag{4}$$

$$= \begin{pmatrix} \phi_1 D & \dots & \phi_T D \end{pmatrix} \begin{pmatrix} \alpha_1 & \dots & \alpha_T \end{pmatrix}^T \tag{5}$$

Given a measurement Y, we recover the input $\{X_i\}_{i=1}^T$ through the DCT coefficients α by solving the optimization problem

$$\min_{\alpha} \|\alpha\|_1 \text{ subject to } Y = \Phi\Psi\alpha,\ \alpha = \begin{pmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_T \end{pmatrix}^T \tag{6}$$

In our implementation we used the `CVX` [6] solver for solving the convex optimization problem in Eq. 6.

2.2 Calculation of coherence for our matrices

Our aim, then, is to optimize the sensing matrices ϕ_i directly for minimum coherence with gradient descent. We now calculate gradients of the coherence with respect to the elements of ϕ_i. As in Eq. 5, with an $n \times n$ dictionary D, we have the effective dictionary

$$\Phi\Psi = \begin{pmatrix} \phi_1 D & \phi_2 D & \dots & \phi_T D \end{pmatrix} \tag{7}$$

The coherence of a general dictionary D, with the i^{th} column defined as d_i, is

$$\mu = \max_{i \neq j} \frac{|\langle d_i, d_j \rangle|}{\sqrt{\langle d_i, d_i \rangle \langle d_j, d_j \rangle}} \tag{8}$$

This expression contains `max` and `abs` functions that a gradient-based scheme cannot handle. Instead, we soften the `max` and convert the `abs` to a square by using, for large enough θ,

$$\max_i \{t_i^2\}_{i=1}^n \approx \frac{1}{\theta} \log \sum_{i=1}^n e^{\theta t_i^2} \tag{9}$$

We need to evaluate the coherence of this dictionary as a function of the elements of Φ. We will call the index varying from 1 to T as μ or ν, and the index varying from 1 to n as α, β or γ. The μ^{th} block of Φ is thus ϕ_μ. Let the β^{th} diagonal element of ϕ_μ be $\phi_{\mu\beta}$. Define the α^{th} column of D^T to be d_α. Then, it can be shown [Appendix A] that the normalized dot product between the β^{th} column of the μ^{th} block and the γ^{th} column of the ν^{th} block is

$$M_{\mu\nu}(\beta\gamma) = \frac{\sum_{\alpha=1}^n \phi_{\mu\alpha}\phi_{\nu\alpha} d_\alpha(\beta) d_\alpha(\gamma)}{\sqrt{\left(\sum_{\alpha=1}^n \phi_{\mu\alpha}^2 d_\alpha^2(\beta)\right)\left(\sum_{\tau=1}^n \phi_{\nu\tau}^2 d_\tau^2(\gamma)\right)}} \tag{10}$$

Finally, using the squared soft-max function [Eq. 9] to deal with the `max` and the `abs` in the coherence expression, we get the squared soft coherence $\mathcal{C}$ to be

$$\mathcal{C} = \frac{1}{\theta} \log \left[\sum_{\mu=1}^{T} \sum_{\nu=1}^{\mu-1} \sum_{\beta=1}^{n} \sum_{\gamma=1}^{n} e^{\theta M_{\mu\nu}^2(\beta\gamma)} + \sum_{\mu=1}^{T} \sum_{\beta=1}^{n} \sum_{\gamma=1}^{\beta-1} e^{\theta M_{\mu\mu}^2(\beta\gamma)} \right] \tag{11}$$

In the above, the first term corresponds to all $(\mu > \nu)$ blocks that are 'below' the block diagonal. Here, we consider all terms in the given block for the maximum. The second term corresponds to $(\mu = \nu)$ blocks on the block diagonal. Here, we consider only consider $(\beta > \gamma)$ below-diagonal elements for the maximum.

2.3 Calculation of coherence derivatives

We note that the $\mathcal{C}$ computed in the section above is a function of Φ. We differentiate $\mathcal{C}$ with respect to $\phi_{\delta\epsilon}$. For this, we define the numerator of the expression for $M_{\mu\nu}(\beta\gamma)$ as $\chi_{\mu\nu}(\beta\gamma)$ and the denominator as $\xi_{\mu\nu}(\beta\gamma)$. The derivative of the objective function can be found in terms of these quantities. Defining $\uparrow_{\mu\delta}$ to be the Kronecker delta function that is 1 only if $\mu = \delta$, it can be shown [Appendix B]

$$\frac{d\chi_{\mu\nu}(\beta\gamma)}{d\phi_{\delta\epsilon}} = d_\epsilon(\beta) d_\epsilon(\gamma) \left(\phi_{\mu\epsilon} \uparrow_{\nu\delta} + \uparrow_{\mu\delta} \phi_{\nu\epsilon} \right) \tag{12}$$

$$\frac{d\xi_{\mu\nu}(\beta\gamma)}{d\phi_{\delta\epsilon}} = \frac{1}{\xi_{\mu\nu}(\beta\gamma)} \left[\phi_{\mu\epsilon} d_\epsilon^2(\beta) \uparrow_{\mu\delta} \sum_{\tau=1}^{n} \phi_{\nu\tau}^2 d_\tau^2(\gamma) + \phi_{\nu\epsilon} d_\epsilon^2(\gamma) \uparrow_{\nu\delta} \sum_{\alpha=1}^{n} \phi_{\mu\alpha}^2 d_\alpha^2(\beta) \right] \tag{13}$$

Using these, we do gradient descent with adaptive step-size and use a multi-start strategy to combat the non-convexity of the problem.

2.4 Time complexity and the need for something more

The calculation of coherence for a matrix requires us to evaluate normalized dot products between columns of the matrix. In our case, the size of the matrix is $n \times nT$, and each dot product needs $\mathcal{O}(n)$ operations, warranting the calculation of $\mathcal{O}(n^3T^2)$ quantities. Optimizing this rapidly becomes intractable as n increases. The performance of gradient descent on this non-convex optimization problem also worsens as the dimensionality of the search-space ($\mathcal{O}(nT)$) increases.

Empirically, we observe that it is intractable to design codes that are more than 20×20 in size in any reasonable time. This points to the fact that we need something more to make designing effective codes possible.

2.5 Circularly-symmetric coherence minimization

The computational intractability of optimizing large codes leads us to designing smaller masks and tiling them to fit the image size we're dealing with. A small coherence for the designed patch guarantees good reconstruction for patches exactly aligned with the code block; however, other patches see a code that is a circular shift of the original code. Fig 1 provides a visual explanation. The big outer square denotes the image. On top of the image we show tiled designed codes. Now, the patch in red clearly multiplies with the exact designed code; however the patch in green multiplies with a code shifted in both the coordinates circularly.

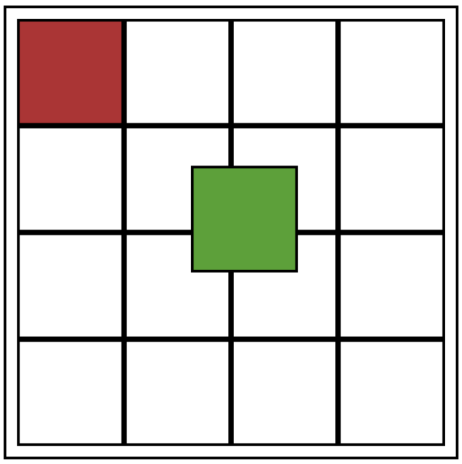

Figure 1: Motivation behind circularly-shifted optimization

This points to designing sensing matrices that have small coherence in all their circular permutations (note that these permutations happen in two dimensions and must be handled as such). To this end, we modify the above objective function to minimize the maximum coherence resulting from all circularly-shifted vectorized versions of Φ. We thus have

$$\mathcal{C} = \frac{1}{\theta} \log \left[\sum_{\zeta \in \text{perm}(\Phi)} \left[\sum_{\mu=1}^{T} \sum_{\nu=1}^{\mu-1} \sum_{\beta=1}^{n} \sum_{\gamma=1}^{n} e^{\theta M_{\mu\nu}^{(\zeta)2}(\beta\gamma)} + \sum_{\mu=1}^{T} \sum_{\beta=1}^{n} \sum_{\gamma=1}^{\beta-1} e^{\theta M_{\mu\mu}^{(\zeta)2}(\beta\gamma)} \right] \right] \tag{14}$$

where $M_{\mu\nu}^{(\zeta)}(\beta\gamma)$ represents the normalized dot product between the β^{th} column of the μ^{th} block and the γ^{th} column of the ν^{th} block, resulting from the instance of the circular permutation ζ of Φ. Derivatives of this expression are found exactly like in Appendix B, except that the μ, ν, β and γ parameters are subjected to the appropriate circular permutation.

The time complexity for determining this maximum coherence among all circular permutations is $\mathcal{O}(n^5T^2)$, out of which a $\mathcal{O}(n^3T^2)$ term arises from the calculation of coherence for each circular permutation, and a $\mathcal{O}(n^2)$ arises from the fact that there are n^2 such permutations. The advantage here, though, is that we don't need to optimize masks having very high values of n; we can do away with keeping n a small constant because the scheme works for any n such that n-sized patches are sparse in the dictionary D. This scheme is, thus, more scalable in terms of the size of the input image. Therefore the effective dimension of the optimization problem in such a scheme is, in terms of the variables that matter, $\mathcal{O}(T^2)$.

It is worth mentioning that this simple idea has been largely ignored in literature concerning sensing matrix optimization. As mentioned in the introduction, previous attempts mostly use an average coherence minimization technique [4, 5, 11] for full-sized sensing matrices, and are not as scalable as ours is for large images because they involve optimization problems in variables whose dimensions are at least of the order of image size. Sensing matrices can be designed at the patch level as well, for instance using information theoretic techniques as in [3, 12, 15], but the methods therein are not designed to account for the issue of overlapping reconstruction. To the best of our knowledge, ours is the first piece of work to handle this important issue in a principled manner.

3 Validation and Results

3.1 Validating our framework

We start with testing the proposed framework visually. In all such results in this paper, we show successive frames top-to-bottom, and different types of reconstruction left-to-right. Here, for the sake of saving time, all reconstructions are done in a non-overlapping way. We first use two synthetic images that are known to have very low sparsity. These are 20×20 images, with only 3 out of the 400 DCT coefficients set to non-zero values. The results, with relative root mean errors of the order of 10^{-5}, for these are shown in Fig. 2. The results are similar for Gaussian sensing matrices and positive random matrices.

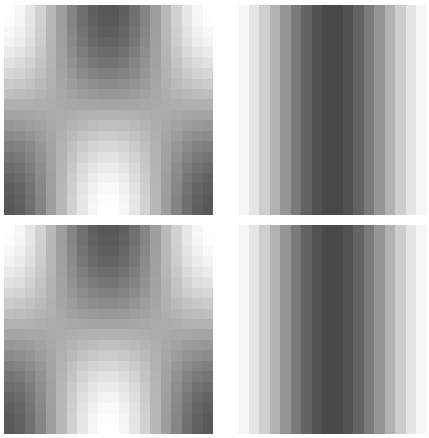

Figure 2: Synthetic image results. Left: input images, Right: reconstructions

Next, we test on two video frames that are very similar, with Gaussian random matrices. The relative root mean square errors are around 0.0019 for each image. The results are shown in Fig. 3.

Figure 3: Real images, Gaussian matrices. Left: input images, right: reconstructions

Next, with positive random diagonal matrices, the relative root mean square errors are around 0.0036 for each image. The results are shown in Fig. 4. Looking at these

Figure 4: Real images, uniform matrices. Left: input images, right: reconstructions

results, one notices that there is very little to no ghosting, that is, appearance of features from one image into the other, in the output images even when the images are very close to each other. This is a very desirable property in any algorithm that separates images from compressed video.

To evaluate how this works for multiple images, we try separating three images with uniform matrices. See Fig. 5. Here, we notice ghosting happening in the third frame. However, with better-designed sensing matrices, one can think of getting rid of this effect. The relative root mean square errors here are worse, around 0.005 for each image.

Figure 5: Separating three images, uniform matrices. Up: input images, down: reconstructions

To simulate sudden changes, we run the optimization with two very different input images. We can separate these well, as is shown in Fig. 6.

We do a numerical comparison between our designed codes and random codes for various values of $s = \|x\|_0/n$ and T. We randomly generate T s-sparse (in 2D DCT) 8×8 signals $\{x_i\}_{i=1}^T$, combine them using random matrices to get y. Average relative

Figure 6: Sudden change, uniform matrices. Left: input images, right: reconstructions

root mean square errors on recovering the input signals from y as a function of s and T are shown in Figs. 7 and 14. Errors are near-zero in the region where both T and s are small, and one can expect reasonable quality reconstructions till $T = 4$ from random matrices. To increase T further, we would need to optimize our sensing matrix appropriately, as is shown further in this paper.

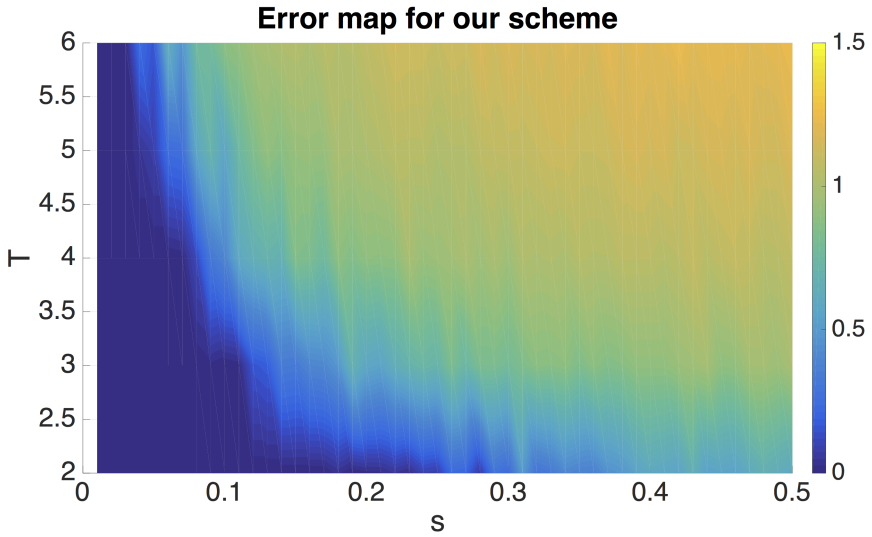

Figure 7: Average relative root mean square errors in our scheme as a function of s and T with positive random matrices

3.2 Demosaicing

To demonstrate the utility of this scheme, we show results on demosaicing RGB images. The general demosaicing problem involves addressing the difficulty that on a camera sensor, a single pixel can sense only one of the three R, G and B channels. Therefore, raw

camera data needs to be interpolated to recover all the three channels. Traditional approaches to demosaicing involve the use of the Bayer pattern, which tiles a fixed [B, G; G, R] pattern over the image and use variants of algorithms like edge-directed interpolation which are tuned to the Bayer pattern. The Matlab `demosaic` function, for instance, uses [10], which takes a gradient-corrected bilinear interpolated approach. However, recently a case has been made for panchromatic demosaicing [7], where we sense a linear combination of the three channels and use techniques from compressive recovery to reconstruct. However, it turns out that the Bayer pattern has very high mutual coherence, so it is unsuitable for compressive recovery. Here, we propose to design the mosaic patterns by minimizing coherence.

We design 8×8 codes for linearly combining the three channels using our method and visually compare overlapping reconstructions. As Figs. 8 and 9 show, results from

Figure 8: Demosaicing. Left: inputs, middle, right: reconstructions with {random, non-circularly designed} matrices

the designed case are more faithful to the ground-truth than the random reconstructions are. The random reconstructions show (more) color artifacts, especially in areas where the input image varies a lot (car headlights in the top image, around parrot eyes in the bottom). Our designed codes do not show as many color artifacts. The relative root mean square errors don't differ much for these two cases, but subtle details of color are better preserved by our matrices. In Fig. 9, notice in the first case the green artifacts near car headlights and the leftmost cyclist in the random reconstruction that is, while that ares is better-reconstructed with our matrices. The car headlight area on the car at the right is also better-reconstructed by our matrices. In the bottom, notice less color artifacts in the densely-varying area near the eye and on the bottom part of the beak.

3.3 Coherence minimization

The coherence of a uniform random matrix of the type we're interested in has a typical value around 0.8 for 8×8 codes. The distribution of these values is shown in the boxplot in Fig 10. The typical profile of descent on coherence from a random initialization is shown in Fig. 11.

The minimum coherence we have been able to achieve in this scheme has been around 0.27 (for $T = 2$). It is interesting to note that all initialization instances lead to coherences

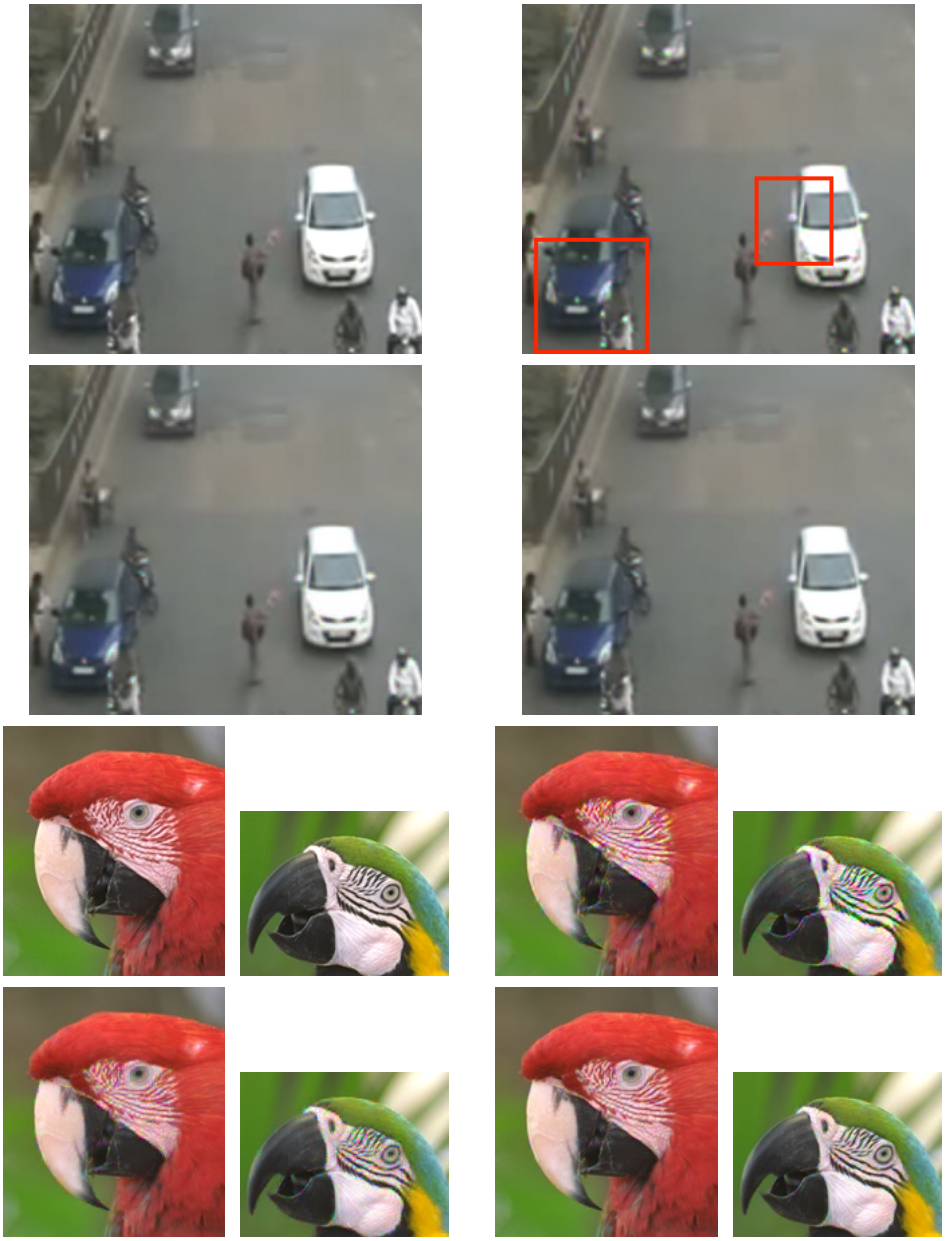

Figure 9: Demosaicing close-ups, examples {1, 2, 3}. Clockwise: inputs, reconstructions with {random, {circularly, non-circularly} designed} matrices

(for $T = 2$) of at the most 0.35, and hence empirically yield nearly as good matrices.

We first visually validate that our matrices perform better than positive random matrices. We design 8×8 codes and tile them, reconstructing patchwise with overlapping patches. An example of this running on six not necessarily close frames in a video is shown in Fig. 12 (Fig. 13 shows an example for $T = 2$). Ghosting artifacts marked out in Fig. 12 in white boxes in the random matrix reconstructions are absent or lower in the designed matrix reconstructions. These outputs show that on the large scale, we do as well as random matrices for low T and better for high T. For a small scale comparison, see Figs. 17 and 18.

Finally, we do a numerical comparison similar to the one in Fig. 7. The resulting error map is shown in Fig. 14. On an average, we see that we perform better than the random case (note the colorbar scale changes). To characterize this, we compute the difference

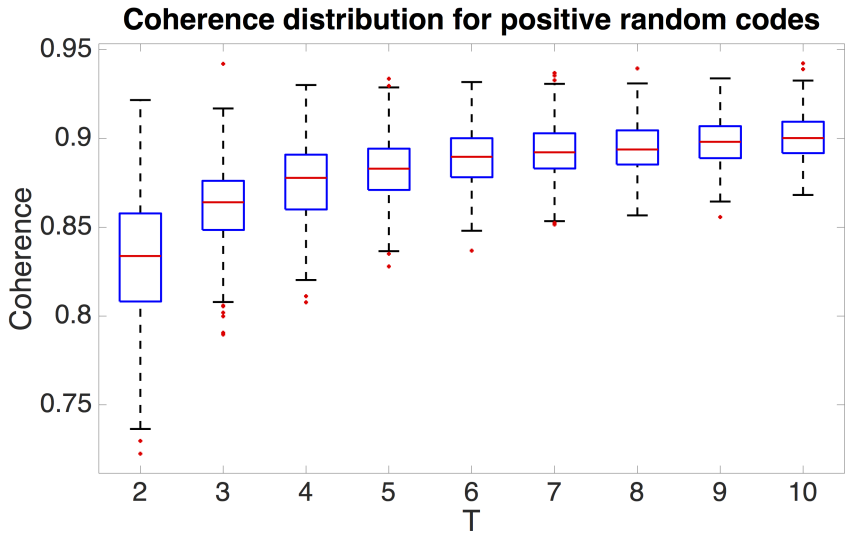

Figure 10: Distribution of coherences for 8×8 random positive codes as a function of T

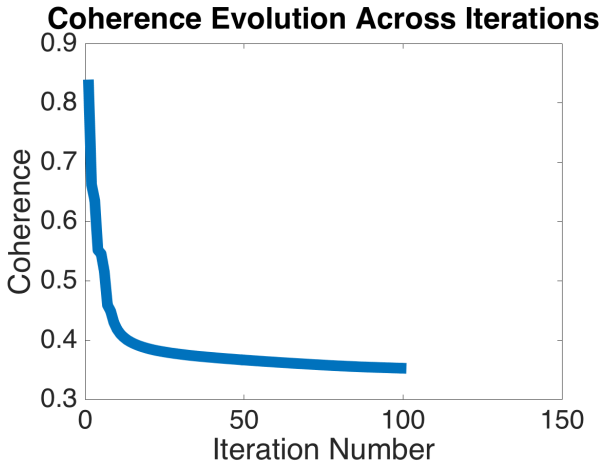

Figure 11: Typical coherence decrease profile

between these two error maps (random minus optimized). The differences add up to a positive quantity (4.5119 in this case), and thus on an average, here, we're better by an relative root mean square factor of 0.018. This is not very significant, though it does produce significant changes in subtle texture as seen in Figs. 17 and 18.

3.4 Circularly-symmetric coherence minimization

Again, we design 8×8 codes for $T = 2$. To show coherence improvement between positive random codes, and codes designed with and without circular permutations, we plot the distribution of coherences of $\Phi^{(\zeta)}D$ in Fig. 15 for all circular permutations ζ. Note that even though the coherences of non-circularly designed matrices are much lower than positive random matrices, the maximum coherence among all permutations is quite large. The circularly-designed matrices, however, have permuted coherences clustered around a low value. We then expect good reconstruction with all circular permutations, yielding good expected reconstructions for images.

Similar to the above section, we validate our matrices visually. Following the same conventions, here is an output for the $T = 2$ case [Fig. 16].

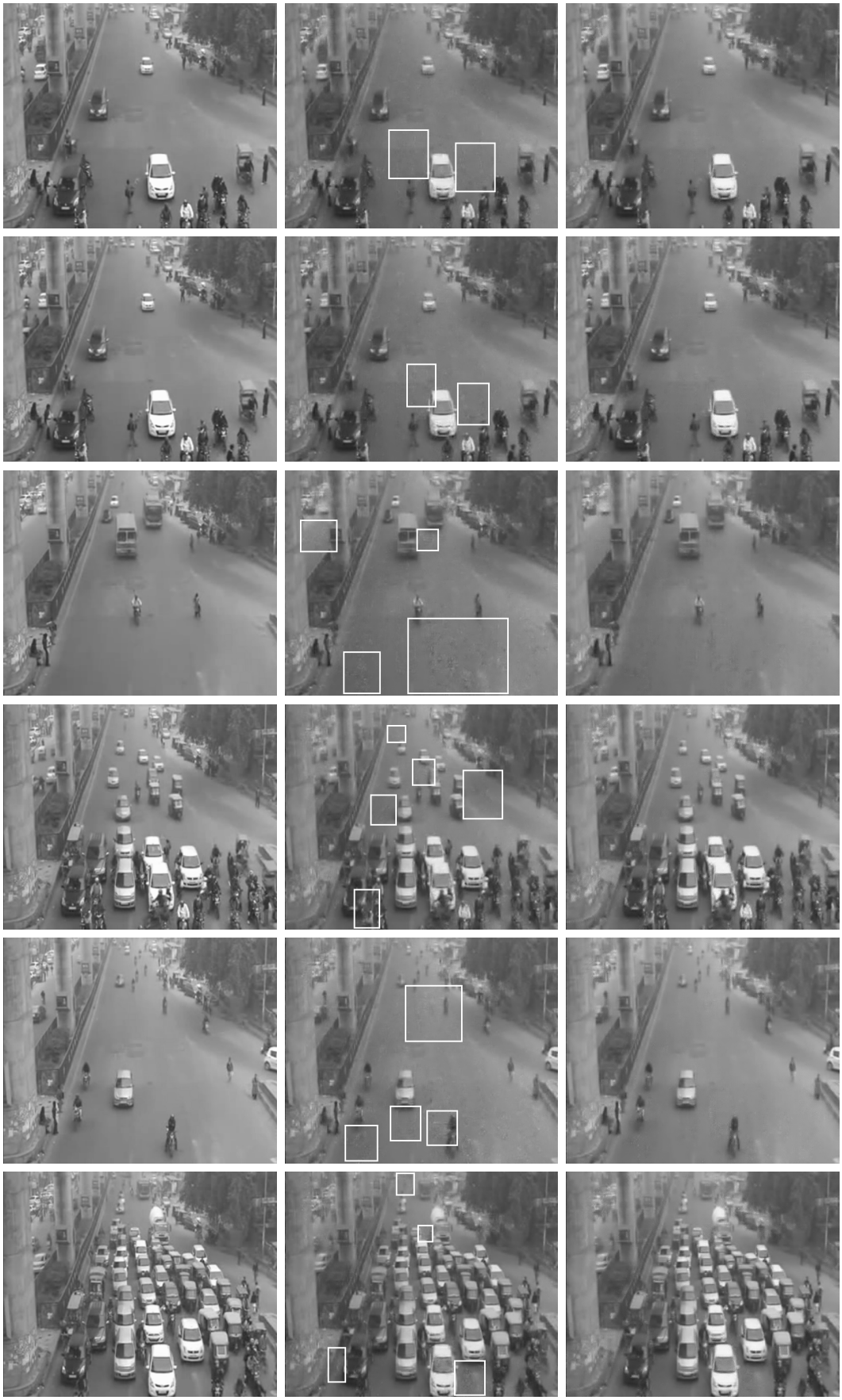

Figure 12: Optimized output from combining six not necessarily close images. Left: inputs, middle, right: reconstructions with {random, non-circularly optimized} matrices

Figure 13: Optimized output from combining two close images. Left: inputs, middle, right: reconstructions with {random, non-circularly optimized} matrices

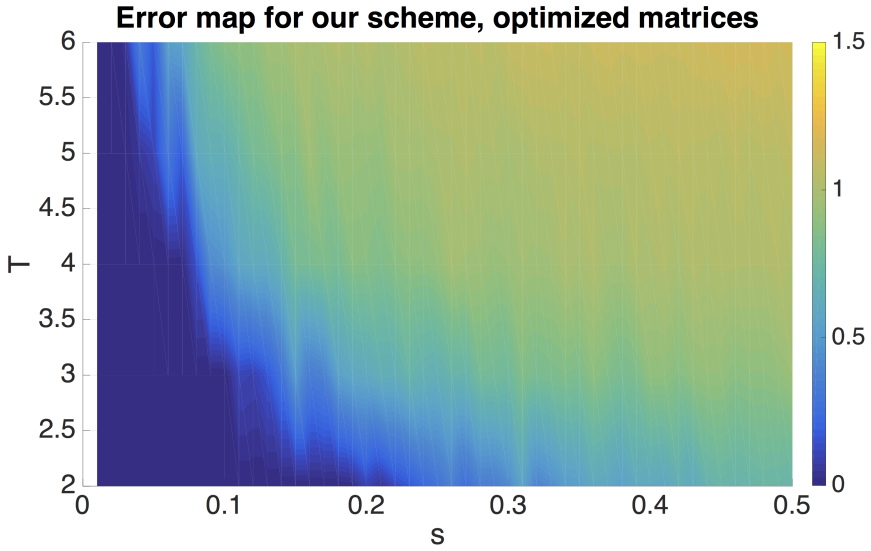

Figure 14: Error map for optimized codes as a function of s and T

We now look at reconstructions from random and both classes of our designed matrices on a small scale. As a first example, we show a close-up from the car video sequence shown earlier [Fig. 17]. Note, to start off, that the reconstruction of the numberplate and headlight area is much clearer in our case than the random matrix case. Further, notice the presence of major ghosting in the random case, especially near the rear-view mirrors, bonnet (marked by arrows) and headlights (marked by boxes), while our reconstructions remain free of these artifacts. Adding circular optimization to the picture further improves image quality especially in the bonnet area, where the non-circular reconstruction is slightly splotchy. Next, in Fig. 18, which is a smaller part of the same image, the superiority of our reconstruction is clearer, with the circular optimization smoothing out blotchier parts of the bonnet.

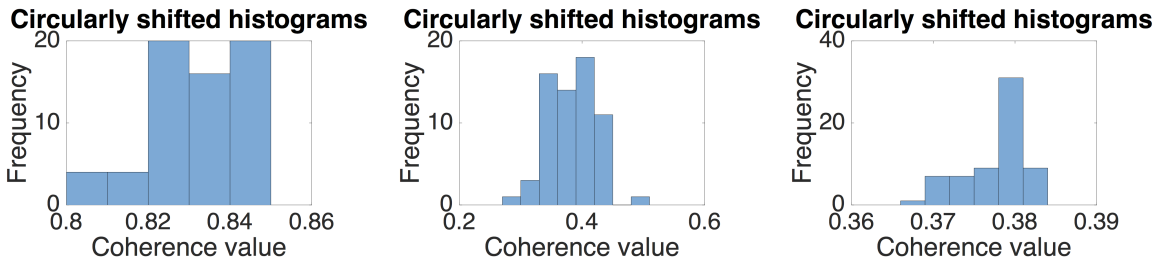

Figure 15: Left to right: Circularly-shifted coherence histograms for {random, non-circularly optimized, circularly optimized} matrices

Figure 16: Circularly optimized output from combining two close images. Left to right: reconstructions with {random, circularly optimized} matrices

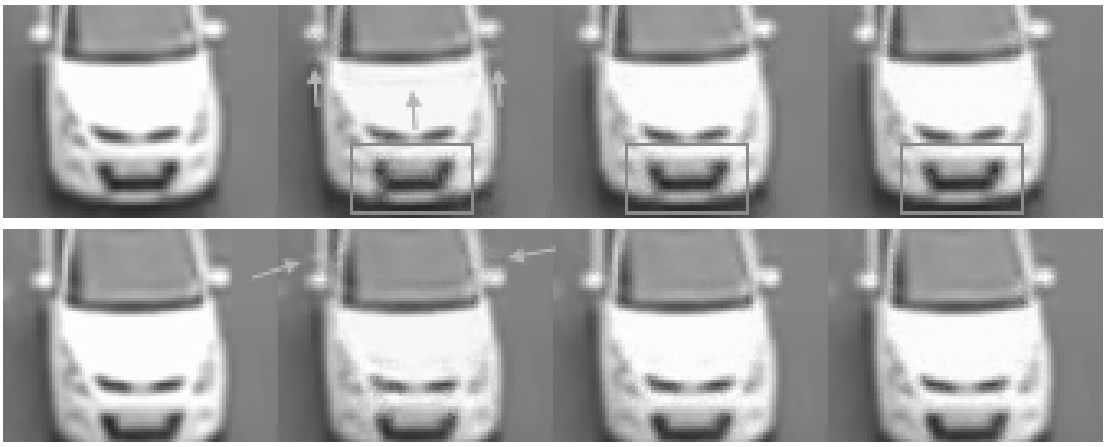

Figure 17: Close-ups showing subtle texture preservation with optimized matrices, example 1. Left to right: inputs, reconstructions with {random, non-circularly optimized, circularly optimized} matrices

4 Conclusion

We cast the video compressed sensing problem as one of separation of coded linear combinations of signals sparse in a given basis. We evaluated this scheme and found it works well for low sparsity levels, and yields reasonably visually good reconstructions. How-

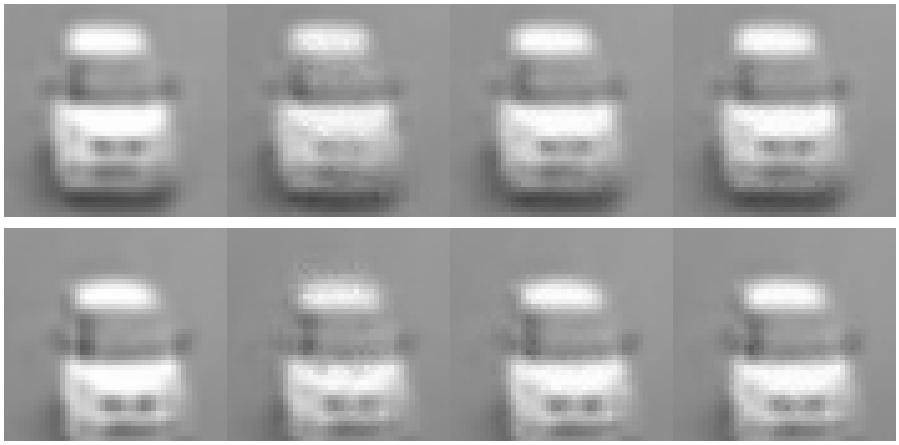

Figure 18: Close-ups showing subtle texture preservation with optimized matrices, example 2. Left to right: inputs, reconstructions with {random, non-circularly optimized, circularly optimized} matrices

ever, especially at high T, we found that random matrices aren't good enough; we need something more.

We then provided an analytical expression for the coherence of the sensing matrix in the coded source separation scheme and optimized for coherence using these. Results showed better quality and less ghosting visually, and less error numerically. However as image size increased, the optimization problem became rapidly intractable, so we settled for optimizing small masks such that they have small coherence in all circular permutations, so they can be tiled for overlapping patchwise reconstruction.

Most code used in generating results and optimizing sensing matrices in this paper lives in the Bitbucket repository at `alankarkotwal/coded-sourcesep` [9]. Gradient descent lives in the `src/descent` folder, circularly-symmetric gradient descent in `src/descent-circular` and reconstruction code in `src/circular`.

A Derivation of coherence expression

Recalling our definitions, we call the index varying from 1 to T as μ or ν, and the index varying from 1 to n as α, β or γ. The μ^{th} block of Φ is thus ϕ_μ. Let the β^{th} diagonal element of ϕ_μ be $\phi_{\mu\beta}$. Define the α^{th} column of D^T to be d_α. Thus, the Gram matrix

$\tilde{M} = \Psi^T \Phi^T \Phi \Psi$ has the block structure

$$\begin{aligned}
\tilde{M}_{\mu\nu} &= D^T \phi_\mu^T \phi_\nu D \\
&= D^T \phi_\mu \phi_\nu D \\
&= \begin{pmatrix} d_1 & d_2 & \dots & d_n \end{pmatrix} \begin{pmatrix} \phi_{\mu 1}\phi_{\nu 1} & 0 & \dots & 0 \\ 0 & \phi_{\mu 2}\phi_{\nu 2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \phi_{\mu n}\phi_{\nu n} \end{pmatrix} \begin{pmatrix} d_1^T \\ d_2^T \\ \vdots \\ d_n^T \end{pmatrix} \\
&= \begin{pmatrix} d_1 & d_2 & \dots & d_n \end{pmatrix} \begin{pmatrix} \phi_{\mu 1}\phi_{\nu 1} d_1^T \\ \phi_{\mu 2}\phi_{\nu 2} d_2^T \\ \vdots \\ \phi_{\mu n}\phi_{\nu n} d_n^T \end{pmatrix} \\
&= \sum_{\alpha=1}^{n} \phi_{\mu\alpha}\phi_{\nu\alpha} d_\alpha d_\alpha^T
\end{aligned}$$

The $\beta\gamma^{\text{th}}$ element of $\tilde{M}_{\mu\nu}$, thus, is

$$\tilde{M}_{\mu\nu}(\beta\gamma) = \sum_{\alpha=1}^{n} \phi_{\mu\alpha}\phi_{\nu\alpha} d_\alpha(\beta) d_\alpha(\gamma) \tag{15}$$

Now we need to normalize the columns of $\Phi\Psi$. Squared column norms are diagonal elements of $\tilde{M}_{\mu\nu}$. So the product of the squared norms of the β^{th} column of the μ^{th} block and the γ^{th} column of the ν^{th} block is (call this $\xi^2_{\mu\nu}(\beta\gamma)$)

$$\xi^2_{\mu\nu}(\beta\gamma) = \left(\sum_{\alpha=1}^{n} \phi^2_{\mu\alpha} d^2_\alpha(\beta) \right) \left(\sum_{\tau=1}^{n} \phi^2_{\nu\tau} d^2_\tau(\gamma) \right) \tag{16}$$

Let the normalized Gram matrix be M. Thus, following the same conventions as above (define the numerator of the expression to be $\chi_{\mu\nu}(\beta\gamma)$),

$$M_{\mu\nu}(\beta\gamma) = \frac{\sum_{\alpha=1}^{n} \phi_{\mu\alpha}\phi_{\nu\alpha} d_\alpha(\beta) d_\alpha(\gamma)}{\sqrt{\left(\sum_{\alpha=1}^{n} \phi^2_{\mu\alpha} d^2_\alpha(\beta)\right) \left(\sum_{\tau=1}^{n} \phi^2_{\nu\tau} d^2_\tau(\gamma)\right)}} = \frac{\chi_{\mu\nu}(\beta\gamma)}{\xi_{\mu\nu}(\beta\gamma)} \tag{17}$$

Finally, using the square soft-max function to deal with the `max` in the coherence expression, we get the squared soft coherence $\mathcal{C}$ to be

$$\mathcal{C} = \frac{1}{\theta} \log \left[\sum_{\mu=1}^{T} \sum_{\nu=1}^{\mu-1} \sum_{\beta=1}^{n} \sum_{\gamma=1}^{n} e^{\theta M^2_{\mu\nu}(\beta\gamma)} + \sum_{\mu=1}^{T} \sum_{\beta=1}^{n} \sum_{\gamma=1}^{\beta-1} e^{\theta M^2_{\mu\mu}(\beta\gamma)} \right] \tag{18}$$

In the above, the first term corresponds to all $(\mu > \nu)$ blocks that are 'below' the block diagonal. Here, we consider all terms in the given block for the maximum. The second term corresponds to $(\mu = \nu)$ blocks on the block diagonal. Here, we consider only consider $(\beta > \gamma)$ below-diagonal elements for the maximum.

B Derivation of coherence derivatives

Differentiating the expression for the squared soft coherence above, we get

$$\frac{d\mathcal{C}(\Phi)}{d\phi_{\delta\epsilon}} = \frac{1}{\theta e^{\theta\mathcal{C}(\Phi)}} \left[\sum_{\mu=1}^{T} \sum_{\nu=1}^{\mu-1} \sum_{\beta=1}^{n} \sum_{\gamma=1}^{n} 2\theta e^{\theta M_{\mu\nu}^2(\beta\gamma)} M_{\mu\nu}(\beta\gamma) \frac{dM_{\mu\nu}(\beta\gamma)}{d\phi_{\delta\epsilon}} + \sum_{\mu=1}^{T} \sum_{\beta=1}^{n} \sum_{\gamma=1}^{\beta-1} 2\theta e^{\theta M_{\mu\mu}^2(\beta\gamma)} M_{\mu\mu}(\beta\gamma) \frac{\theta M_{\mu\mu}(\beta\gamma)}{d\phi_{\delta\epsilon}} \right] \tag{19}$$

Next, we calculate the derivatives in the above equation, $dM_{\mu\nu}(\beta\gamma)/d\phi_{\delta\epsilon}$. Define the numerator of the expression for $M_{\mu\nu}(\beta\gamma)$ as $\chi_{\mu\nu}(\beta\gamma)$, and thus, $M_{\mu\nu}(\beta\gamma) = \chi_{\mu\nu}(\beta\gamma)/\xi_{\mu\nu}(\beta\gamma)$. Clearly,

$$\frac{dM_{\mu\nu}(\beta\gamma)}{d\phi_{\delta\epsilon}} = \frac{\xi_{\mu\nu}(\beta\gamma)\frac{d\chi_{\mu\nu}(\beta\gamma)}{d\phi_{\delta\epsilon}} - \chi_{\mu\nu}(\beta\gamma)\frac{d\xi_{\mu\nu}(\beta\gamma)}{d\phi_{\delta\epsilon}}}{\xi_{\mu\nu}(\beta\gamma)^2} \tag{20}$$

Next,

$$\begin{aligned} \frac{d\chi_{\mu\nu}(\beta\gamma)}{d\phi_{\delta\epsilon}} &= \frac{d}{d\phi_{\delta\epsilon}} \sum_{\alpha=1}^{n} \phi_{\mu\alpha}\phi_{\nu\alpha} d_\alpha(\beta) d_\alpha(\gamma) \\ &= \sum_{\alpha=1}^{n} d_\alpha(\beta) d_\alpha(\gamma) \frac{d}{d\phi_{\delta\epsilon}} \left(\phi_{\mu\alpha}\phi_{\nu\alpha}\right) \end{aligned}$$

Notice that a term in the above summation can be non-zero only if $\alpha = \epsilon$. Thus,

$$\begin{aligned} \frac{d\chi_{\mu\nu}(\beta\gamma)}{d\phi_{\delta\epsilon}} &= d_\epsilon(\beta) d_\epsilon(\gamma) \frac{d}{d\phi_{\delta\epsilon}} \left(\phi_{\mu\epsilon}\phi_{\nu\epsilon}\right) \\ &= d_\epsilon(\beta) d_\epsilon(\gamma) \left(\phi_{\mu\epsilon} \frac{d\phi_{\nu\epsilon}}{d\phi_{\delta\epsilon}} + \frac{d\phi_{\mu\epsilon}}{d\phi_{\delta\epsilon}} \phi_{\nu\epsilon} \right) \end{aligned}$$

Now, notice that $d\phi_{\mu\epsilon}/d\phi_{\delta\epsilon}$ is non-zero only if $\mu = \epsilon$. Denote by $\uparrow_{\mu\epsilon}$ the Kronecker delta function, which is 1 only if $\mu = \epsilon$, 0 otherwise. Then,

$$\frac{d\chi_{\mu\nu}(\beta\gamma)}{d\phi_{\delta\epsilon}} = d_\epsilon(\beta) d_\epsilon(\gamma) \left(\phi_{\mu\epsilon} \uparrow_{\nu\delta} + \uparrow_{\mu\delta} \phi_{\nu\epsilon}\right) \tag{21}$$

Next,

$$\begin{aligned} \frac{d\xi_{\mu\nu}(\beta\gamma)}{d\phi_{\delta\epsilon}} &= \frac{d}{d\phi_{\delta\epsilon}} \sqrt{\left(\sum_{\alpha=1}^{n} \phi_{\mu\alpha}^2 d_\alpha^2(\beta)\right)\left(\sum_{\tau=1}^{n} \phi_{\nu\tau}^2 d_\tau^2(\gamma)\right)} \\ &= \frac{1}{2\xi_{\mu\nu}(\beta\gamma)} \frac{d}{d\phi_{\delta\epsilon}} \left(\sum_{\alpha=1}^{n} \phi_{\mu\alpha}^2 d_\alpha^2(\beta) \sum_{\tau=1}^{n} \phi_{\nu\tau}^2 d_\tau^2(\gamma)\right) \\ &= \frac{1}{2\xi_{\mu\nu}(\beta\gamma)} \left[\sum_{\alpha=1}^{n} \phi_{\mu\alpha}^2 d_\alpha^2(\beta) \frac{d}{d\phi_{\delta\epsilon}} \left(\sum_{\tau=1}^{n} \phi_{\nu\tau}^2 d_\tau^2(\gamma)\right) + \sum_{\tau=1}^{n} \phi_{\nu\tau}^2 d_\tau^2(\gamma) \frac{d}{d\phi_{\delta\epsilon}} \left(\sum_{\alpha=1}^{n} \phi_{\mu\alpha}^2 d_\alpha^2(\beta)\right) \right] \end{aligned}$$

Again, a term in one of the above summations is non-zero only if α or τ is the same as ϵ. Thus,

$$\frac{d}{d\phi_{\delta\epsilon}}\left(\sum_{\alpha=1}^{n}\phi_{\mu\alpha}^2 d_\alpha^2(\beta)\right) = 2\phi_{\mu\epsilon}d_\epsilon^2(\beta)\uparrow_{\mu\delta}$$

Thus,

$$\frac{d\xi_{\mu\nu}(\beta\gamma)}{d\phi_{\delta\epsilon}} = \frac{1}{\xi_{\mu\nu}(\beta\gamma)}\left[\phi_{\mu\epsilon}d_\epsilon^2(\beta)\uparrow_{\mu\delta}\sum_{\tau=1}^{n}\phi_{\nu\tau}^2 d_\tau^2(\gamma) + \phi_{\nu\epsilon}d_\epsilon^2(\gamma)\uparrow_{\nu\delta}\sum_{\alpha=1}^{n}\phi_{\mu\alpha}^2 d_\alpha^2(\beta)\right] \quad (22)$$

This completes the calculation of derivatives.

References

[1] T. T. Cai and L. Wang. Orthogonal matching pursuit for sparse signal recovery with noise. *IEEE Transactions on Information Theory*, 57(7):4680–4688, July 2011.

[2] T. T. Cai, L. Wang, and G. Xu. New bounds for restricted isometry constants. *IEEE Transactions on Information Theory*, 56(9):4388–4394, Sept 2010.

[3] William R. Carson, Minhua Chen, Miguel R. D. Rodrigues, Robert Calderbank, and Lawrence Carin. Communications-inspired projection design with application to compressive sensing. *SIAM Journal on Imaging Sciences*, 5(4):1185–1212, 2012.

[4] J. M. Duarte-Carvajalino and G. Sapiro. Learning to sense sparse signals: Simultaneous sensing matrix and sparsifying dictionary optimization. *IEEE Transactions on Image Processing*, 18(7):1395–1408, July 2009.

[5] M. Elad. Optimized projections for compressed sensing. *IEEE Transactions on Signal Processing*, 55(12):5695–5702, Dec 2007.

[6] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, March 2014.

[7] K. Hirakawa and P. J. Wolfe. Spatio-spectral color filter array design for optimal image recovery. *IEEE Transactions on Image Processing*, 17(10):1876–1890, Oct 2008.

[8] Y. Hitomi, J. Gu, M. Gupta, T. Mitsunaga, and S. K. Nayar. Video from a single coded exposure photograph using a learned over-complete dictionary. In *2011 International Conference on Computer Vision*, pages 287–294, Nov 2011.

[9] Alankar Kotwal. Implementation. `http://bitbucket.org/alankarkotwal/coded-sourcesep/`.

[10] Rico Malvar, Li wei He, and Ross Cutler. High-Quality Linear Interpolation for Demosaicing of Bayer-Patterned Color Images. In *International Conference of Acoustic, Speech and Signal Processing*, May 2004.

[11] M. Mordechay and Y. Y. Schechner. Matrix optimization for poisson compressed sensing. In *Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on*, pages 684–688, Dec 2014.

[12] F. Renna, M. R. D. Rodrigues, M. Chen, R. Calderbank, and L. Carin. Compressive sensing for incoherent imaging systems with optical constraints. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5484–5488, May 2013.

[13] Christoph Studer and Richard G. Baraniuk. Stable restoration and separation of approximately sparse signals. *Applied and Computational Harmonic Analysis*, 37(1):12 – 35, 2014.

[14] A. Veeraraghavan, D. Reddy, and R. Raskar. Coded strobing photography: Compressive sensing of high speed periodic videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4):671–686, April 2011.

[15] Yair Weiss, Hyun Sung Chang, and William T. Freeman. Learning Compressed Sensing. In *Allerton Conference on Communication, Control, and Computing*, 2007.