Quaternion K-SVD for Color Image Denoising

Technical Report

Amit Carmeli

Instructor: Mr. Javier Turek

Table of Contents

Introduction:

In this work, we introduce the use of Quaternions within the field of sparse and redundant representations. The Quaternion space is an extension of the complex space, where each element is composed of four parts – a real-part and three imaginary parts. The major difference between Quaternion space $\mathbb{H}$ and the complex space $\mathbb{C}$ is that the Quaternion space has non-commutative multiplication.

We design and implement Quaternion variants of state-of-the-art algorithms OMP and K-SVD. We show various results, previously established only for the real or complex spaces, and use them to devise the Quaternion K-SVD algorithm, nicknamed QK-SVD. Finally, we discuss the use of QK-SVD for the purpose of color image denoising, and compare the achieved results with other work in the field.

1. On Sparse and Redundant Representations:

1.1 What is this All About?

For the past decade, the field of sparse and redundant representations has become the leading model in image processing **[1]**. The model aims to describe a class of signals Γ, assuming the existence of an $n \times m$ matrix D, called **dictionary**, with $n < m$, such that for any signal $y \in \Gamma$ there exists a sparse linear combination of the columns of D that approximates y. Formally, $\exists x \in \mathbb{R}^m$ such that $Dx \approx y$ with $\|x\|_0 \ll n$.

The notation $\|\cdot\|_0$ refers to the $\ell_0 -$ quasi-norm that counts the number of non-zero entries. The sparse nature of the model is $\|x\|_0$, implying that few columns of D, called **atoms**, can approximate any signal in Γ; its redundant nature is the number of atoms, which is typically greater than the signal dimension.

1.2 Orthogonal Matching Pursuit (OMP):

A fundamental problem imposed by the model is the recovery of the sparse representation x of a vector y given a dictionary D. We formulate this problem as follows,

$$\textbf{(1)} \quad \min_{x \in \mathbb{R}^m} \quad \|x\|_0 \quad s.t. \quad \|y - Dx\|_2^2 \leq \varepsilon^2$$

that is, we seek a vector x as sparse as possible that can approximate y well, given D. Unfortunately, the problem is NP-hard, and thus cannot be solved optimally.

A common family of methods to approximate the solution of **(1)** is greedy algorithms. These iterative algorithms choose one or more columns of D at each iteration, aiming to reduce the residual norm $\|y - Dx\|_2^2$ as greatly as possible. One such greedy algorithm is the Orthogonal Matching Pursuit (OMP), depicted in Figure 1.

The OMP chooses a column of D, denoted d_j, that can decrease the norm of the residual $y - Dx$ further than any other column of D. Once the column is chosen, x is re-computed by least-squares, subject to the chosen support, i.e. the non-zero entries of x will correspond only to the chosen columns. The least-squares process assures that no column is chosen twice.

The process then continues by re-computing the residual and choosing an additional column of D, until the residual is sufficiently low, i.e. $\|y - Dx\|_2^2 \leq \varepsilon^2$.

Input: $D \in \mathbb{R}^{n \times m}, y \in \mathbb{R}^n, \varepsilon \geq 0, T > 0$

1. $k \leftarrow 0, x^0 \leftarrow 0, r^0 \leftarrow y, \mathcal{S}^0 = \phi$
2. while $\left\| r^k \right\|_2 > \varepsilon$ and $k < T$, do:
   - 2.1 $k \leftarrow k+1$
   - 2.2 For all $j \notin \mathcal{S}^{k-1}$, compute $\nu(j) = \min_{z_j} \left\| d_j z_j - r^{k-1} \right\|_2^2$.
   - 2.3 $j_0 = \arg\min_{j \notin \mathcal{S}^{k-1}} \nu(j)$
   - 2.4 $\mathcal{S}^k \leftarrow \mathcal{S}^{k-1} \cup \{ j_0 \}$
   - 2.5 $x^k \leftarrow \arg\min_{x \in \mathbb{R}^m} \left\| y - Dx \right\|_2^2$ $\quad s.t. \quad$ $Support\{x\} = \mathcal{S}^k$
   - 2.6 $r^k \leftarrow y - Dx^k$

Figure 1: Orthogonal Matching Pursuit (OMP) (for real numbers)

1.3 K-SVD Dictionary Learning:

To use the OMP on a signal, one must already have a dictionary at hand. While choosing a dictionary is a possible option, in **[2]** Mairal et. al. suggested a method to learn the dictionary D from a given set of sample vectors. This method, known as K-SVD, starts with some initial dictionary, and operates in two stages.

The first stage, called **sparse coding**, computes the sparse representation of every sample vector using the current dictionary via OMP. The second stage, called **dictionary update**, iterates over every atom in the dictionary and fixes it, with respect to the current sparse representations. We shall dwell on this fix later on. These two stages are performed iteratively, until convergence.The algorithm is depicted in Figure 2.

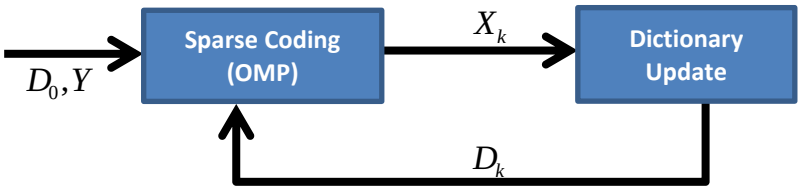


Figure 2: Block-diagram for the K-SVD algorithm. Sparse coding stage is performed via OMP to retrieve sparse representations, then, the dictionary update stage updates the representations and the dictionary.

The K-SVD algorithm is the backbone of a state-of-the-art denoising scheme for grayscale images, suggested in **[3]**, **[4]**. Patches are extracted from the image to serve as sample vectors for the K-SVD algorithm, and are then denoised by the OMP applications. In **[5]**, this basic structure was broadened to color images as well; by extracting 3-dimensional patches and letting the dictionary be of size $3n \times m$, it was shown that using a modified OMP leads to surprising results in color image denoising, as well.

The modified OMP effectively alters the inner-product between two vectors x, y , by introducing a weight matrix W , so that the inner-product is computed as $x^T W y$. The matrix W was chosen specifically for the purpose of color image denoising. For each respective channel, the matrix W adds the average for that channel. This changes the histogram of the patch, so instead of denoising with respect to the histogram formed by all 3 channels, the OMP now denoises the patches with respect to the averages of each channel. Introducing the weight matrix W requires only a matrix-by-matrix multiplication of W by each of the patches and by the dictionary. For a more detailed explanation, see **[5]**.

We follow the work in **[5]**, in an attempt to show better qualities of the K-SVD algorithm from a different approach. The use of 3-dimensional patches is a direct result of interpretation of the color image as three separate grayscale images; instead, we suggest that a color image is a single matrix, whose entries have Quaternion values.

2. Quaternions and Color Images:

The Quaternion space, denoted $\mathbb{H}$, is a ring, with non-commutative multiplication. Each Quaternion can be written as $a_1 + a_2 i + a_3 j + a_4 k$, with $a_i \in \mathbb{R}$ and i, j, k satisfying:

$$\textbf{(2)} \quad \begin{aligned} ij &= k & ji &= -k \\ ik &= -j & ki &= j \\ jk &= i & kj &= -i \end{aligned}$$

In color images, we shall treat each pixel as a Quaternion value, whose real-part a_1 is zero. The imaginary parts will correspond to the R-, G-, and B-channels of the pixel in the color image. To use the K-SVD algorithm in a similar manner to [5] on this new representation, we must implements both the OMP and the dictionary update stages of the algorithm, so that they will apply to Quaternion vectors and matrices as well.

Appendix A and **[7]** contain further information regarding properties that hold in $\mathbb{H}$ and $\mathbb{H}^n$ that are important in order to implement the K-SVD algorithm under the Quaternion space. These properties are intuitive in the complex space $\mathbb{C}$, bur require careful attendance when addressed in the Quaternion space $\mathbb{H}$, due to the non-commutative multiplication.

3. Quaternion OMP (Q-OMP):

3.1 Selection of the Column

In OMP, the column is chosen to minimize the Euclidean norm of the residual $y - Dx$. The same goal remains even in the Quaternion space. In order to implement the OMP algorithm for the Quaternion space, we need to implement steps 2.2 and 2.5 described in Figure 1 specifically for Quaternion matrices; implementation of the remainder of the steps is straightforward and requires no changes with respect to the original OMP.

For step 2.2, we need to solve a one-dimensional Quaternion least-squares problem. Solving such a problem is no harder than solving it within the complex space, and simply requires solution of the normal equations (see Appendix A). Thus, the minimizer of $\left\| d_j z_j - r^{k-1} \right\|_2^2$ is given by:

$$\textbf{(3)} \quad z = \frac{d_j^* r^{k-1}}{d_j^* d_j} = \frac{d_j^* r^{k-1}}{\left\| d_j \right\|_2^2}$$

Setting **(3)** in the expression $\left\| d_j z_j - r^{k-1} \right\|_2^2$ will yield the minimum Euclidean norm of the residual, when the j^{th} atom is chosen. The OMP algorithm will add to the support the atom that best minimizes that Euclidean norm. If the columns of the dictionary are normalized, this can be achieved by choosing the column d_j that maximizes $\left| d_j^* r^{k-1} \right|^2$ (see Appendix B). Thus, steps **2.2** and **2.3** are implemented directly by computing $\left| D^* r^{k-1} \right|$ and choosing the index j of maximum value.

3.2 Projection onto the Support

Once the column is chosen, x is re-computed by least-squares, as illustrated in step **2.5**. For step **2.5**, we solve least-squares over Quaternion space, but we wish to do so differently, without the aid of normal equations. We first write $D = D_1 + iD_2 + jD_3 + kD_4$ and $y = y_1 + iy_2 + jy_3 + ky_4$, with D_1, D_2, D_3, D_4 being real matrices and y_1, y_2, y_3, y_4 real vectors, and let $x = x_1 + ix_2 + jx_3 + kx_4$ with real vectors x_1, x_2, x_3, x_4. Since the Euclidean norm of a Quaternion vector is the sum of the Euclidean norms of its real part, $i-$ part, $j-$ part and $k-$ part, we have (see Appendix B):

$$\textbf{(4)}\ \left\|y - Dx\right\|_2 = \left\| \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} - \begin{pmatrix} D_1 & -D_2 & -D_3 & -D_4 \\ D_2 & D_1 & -D_4 & D_3 \\ D_3 & D_4 & D_1 & -D_2 \\ D_4 & -D_3 & D_2 & D_1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \right\|_2$$

The expression in **(4)** allows us compute the entries of x by directly solving a real least-squares problem. Since **(4)** is merely a different formulation for the same objective function $\left\|y - Dx\right\|_2^2$, computing the minimum via **(4)** will also yield the minimum for $\left\|y - Dx\right\|_2^2$.

To assure that x is subject to the computed support $\mathcal{S}^k$, instead of using the full D, we use $D_{\mathcal{S}^k}$ - the sub-matrix of D that is composed of all columns that are part of $\mathcal{S}^k$. Thus, we only compute the entries of x that correspond to $\mathcal{S}^k$, and set all other entries to zero, instead.

4. Quaternion K-SVD (QK-SVD):

4.1 Quaternion Dictionary Update

The implementation of Q-OMP also completes the first stage of the K-SVD algorithm for Quaternions. However, in order to implement the second stage, the dictionary update, we must first consult the original K-SVD algorithm.

The dictionary update stage iterates over every atom d_{j_0}, and observes all sample vectors whose sparse representation uses d_{j_0}. Let $\Omega_{j_0} = \left\{ i \mid X_{j_0 i} \neq 0 \right\}$, with X being a matrix whose columns are the sparse representations computed at the sparse coding stage. Then, it computes the residual matrix E_{j_0} as:

$$\textbf{(5)}\quad E_{j_0} = Y - \sum_{\substack{j=1 \\ j \neq i}}^{k} d_j X_j^T$$

with X_j^T being the j^{th} row of X, and Y the matrix of sample vectors. Then, the algorithm removes all columns matrix E_{j_0} that correspond to sample vectors that do not use the atom d_{j_0} in the current representation. Hereafter, $X_{j_0}^T$ refers to the restricted version of $X_{j_0}^T$; this assures that the sparse representations remain sparse, as we do not update representations that do not initially use the atom d_{j_0}.

4.2 Dictionary Update via Exact Minimization

The dictionary update stage then aims to minimize the Frobenius-norm of the error,

$$\textbf{(6)} \quad \left\| E_{j_0} - d_{j_0} X_{j_0}^T \right\|_F^2$$

where the variables are both d_{j_0} and $X_{j_0}^T$. This allows updating both the atom and the sparse representations in accordance. This rank-one approximation of E_{j_0} can be solved either by exact minimization via the computation of the singular-value decomposition (SVD) of E_{j_0}, or by alternating minimization that updates d_{j_0} and $X_{j_0}^T$ in a block-coordinate fashion. Notice that in either case, d_{j_0} must be normalized.

As in the original K-SVD algorithm, we can solve **(6)** by an exact minimization, given by truncated SVD of the matrix E_{j_0}. Since E_{j_0} in our case is Quaternion, we write $E_{j_0} = A + Bj$, with A, B being complex matrices. The SVD of E_{j_0} can be computed by defining the **complex adjoint**,

$$\textbf{(7)} \quad \chi_{E_{j_0}} = \begin{pmatrix} A & B \\ -B^* & A^* \end{pmatrix}$$

and then computing the SVD for $\chi_{E_{j_0}}$, which is a complex matrix. Using Caley's formula **[6]** one can extract the singular values and singular vectors of E_{j_0} directly from those of $\chi_{E_{j_0}}$. Also notice that since only the first singular vectors and singular value are required, we can suffice in computing the first singular vectors and singular value of $\chi_{E_{j_0}}$, as well. This allows for an economical implementation, even when exact minimization is performed.

Once the SVD of E_{j_0} is computed as $E_{j_0} = U \Sigma V^*$, we update $d_{j_0} = u_1$, the first column of the unitary matrix U, which also retains normalization of the atom; and $X_{j_0}^T = \sigma_1 v_1^*$, where σ_1 is the first singular value of E_{j_0} and v_1^* is the first row of V^*.

4.3 Dictionary Update via Alternating Minimization

An alternative method to solve **(6)** would be by alternating minimization performed in a block-coordinate fashion. Since there are two variables in this case, d_{j_0} and $X_{j_0}^T$, we first minimize **(6)** over d_{j_0}, then over $X_{j_0}^T$. Minimization over either variable can be performed exactly as the Frobenius-norm of the matrix can be described as the sum of the Euclidean-norms of each of its columns, and thus, similar analysis can be done as in the case of Q-OMP (see Appendix C). Zeroing the gradient of **(6)** with respect to d_{j_0} results in:

$$\textbf{(8)} \quad d_{j_0} = \frac{E \bar{X}_{j_0}}{\left\| X_{j_0} \right\|_2^2}$$

As mentioned above, d_{j_0} must be normalized, so $d_{j_0} = E \bar{X}_{j_0} / \left\| E \bar{X}_{j_0} \right\|_2$. Zeroing the gradient of **(6)** with respect to $X_{j_0}^T$ results in:

$$\textbf{(9)} \quad X_{j_0}^T = \frac{d_{j_0}{}^* E}{\left\| d_{j_0} \right\|_2} = d_{j_0}{}^* E$$

Thus, **(8)** and **(9)** allow us to perform the dictionary update stage without exact computation of SVD, even under the Quaternion space. The complete QK-SVD algorithm, with exact minimization for the dictionary update stage, is depicted in Figure 3.

5. Q-Lib for Quaternions in MatLab:

Since MatLab does not directly support Quaternions, we implemented the Q-Lib library that allows manipulation and usage of Quaternion matrices and vectors. The library supports a handful of operations, from inner and outer products, to homomorphism onto the real space, to truncated Quaternion SVD.

The library was easily implemented by treating all Quaternion matrices as concatenations of four real matrices, one after the other. This structure allows for some optimizations and an easy implementation of operations like Frobenius norm (Euclidean norm for vectors), matrix by matrix multiplications, and more.

Most function calls are rather intuitive in Q-Lib, and retain the same name from the original MatLab function (if exists), preceded by "Q_". Thus, Q_size will compute the size of a Quaternion matrix, while Q_eye and Q_randn can assist in creating Quaternion identity matrix and Quaternion random matrix with Gaussian entries, respectively.

Input: $Y \in \mathbb{H}^{n\times k}$ Quaternion matrix of sample vectors,
ε **Q-OMP** threshold,
J number of iterations,
$D_0 \in \mathbb{R}^{n\times m}$ initial Quaternion dictionary with normalized columns

1. $D \leftarrow D_0$
2. For $j = 1 \ldots J$ do:
   - 3.1 **sparse coding**
     - 3.1.1 For each column Y_i of Y do:
       - 3.1.1.1 Solve $\min_{x_i \in \mathbb{R}^m} \|x_i\|_0 \quad s.t. \quad \|Y_i - Dx_i\|_2^2 \le \varepsilon$ via **Q-OMP**
   - 3.2 **dictionary update**

     Let $X = [x_1 | \ldots . | x_k]$.

     3.2.1 For each atom d_{j_0} of D do:

     3.2.1.1 $\Omega_{j_0} = \{ i \mid X_{j_0 i} \neq 0 \}$

     3.2.1.2 $E_{j_0} \leftarrow Y - \sum_{\substack{j=1 \\ j \neq j_0}}^{k} d_j X_j^T$

     3.2.1.3 Restrict E_{j0} by choosing only the columns corresponding to Ω_{j_0}

     3.2.1.4 Let $E_{j_0} = U \Sigma V^*$ be the Quaternion singular-value decomposition (SVD) of E_{j_0} . Then: $d_{j_0} \leftarrow u_1$, $X_{j_0, \Omega_{j_0}} \leftarrow \sigma_1 \cdot v_1^*$

Figure 3: Quaternion K-SVD (QK-SVD) with exact minimization for dictionary update stage.

6. Color Image Denoising:

The resulting Quaternion K-SVD algorithm, nicknamed QK-SVD, allowed us to devise a scheme to the denoising of color images. As mentioned before, a color image can be thought of as a Quaternion matrix, whose pixels have a real-part of zero, and their imaginary parts, i, j, k correspond to the R-, G- and B-channels of the pixel.

In accordance with **[5]**, with extract all fully-overlapped patches of size $\sqrt{n} \times \sqrt{n}$ from the color image to a get a matrix $Y \in \mathbb{H}^{n \times k}$ of sample vectors. Notice that each sample vector has its real-part zero. We then apply the QK-SVD algorithm with some initial dictionary, in order to learn both the dictionary and the sparse representations. As mentioned before, the sparse coding stage that uses OMP (or Q-OMP in this case), is responsible for the denoising of the patches, resulting a better quality of image.

Once the patches are denoised by the QK-SVD, they are stitched back into the image; since the patches overlap one another, we average the values for each pixel according to the number of patches that overlap that pixel.

7. Numerical Experiments:

The above scheme for color image denoising will now be demonstrated. In addition, we use only the exact minimization via Quaternion SVD in the dictionary update stage, and fix the number of iterations of QK-SVD globally to $J = 20$. For the purpose of experiment, we use the images shown in Figure 4.

Figure 4: The original images CASTLE**,** MUSHROOM and HORSES on which we demonstrate the use of the QK-SVD algorithm for color image denoising.

We first aim to select the best patch-size $\sqrt{n} \times \sqrt{n}$ for the QK-SVD algorithm. Table 1 shows the PSNR of the image CASTLE after restoration as function of both the size of patches and the noise standard deviation denoted σ.

σ / PSNR	4 x 4	5 x 5	6 x 6	7 x 7	8 x 8
5 / 34.1521	39.0896	**39.1281**	38.9855	38.8415	38.7551
10 / 28.1236	34.7610	**35.0282**	34.9942	34.8162	34.7833
25 / 20.1834	28.8082	29.5333	**29.8637**	29.7997	29.7188
σ / PSNR	**9 x 9**	**10 x 10**	**11 x 11**	**12 x 12**	**13 x 13**
25 / 20.1834	29.6010	29.4335	29.2985	29.0706	28.9829

Table 1: PSNR of the image CASTLE as function of the noise standard deviation σ and the size of patches extracted. We use a fixed number of iterations $J = 20$ with 256 atoms ($m = 256$). The optimal size of patches for each σ is bolded. For this experiment, noise was added to the real-part of the signal.

The image quality is affected by the size of the patch. This is evident not only by Table 1, but also by Figure 5.

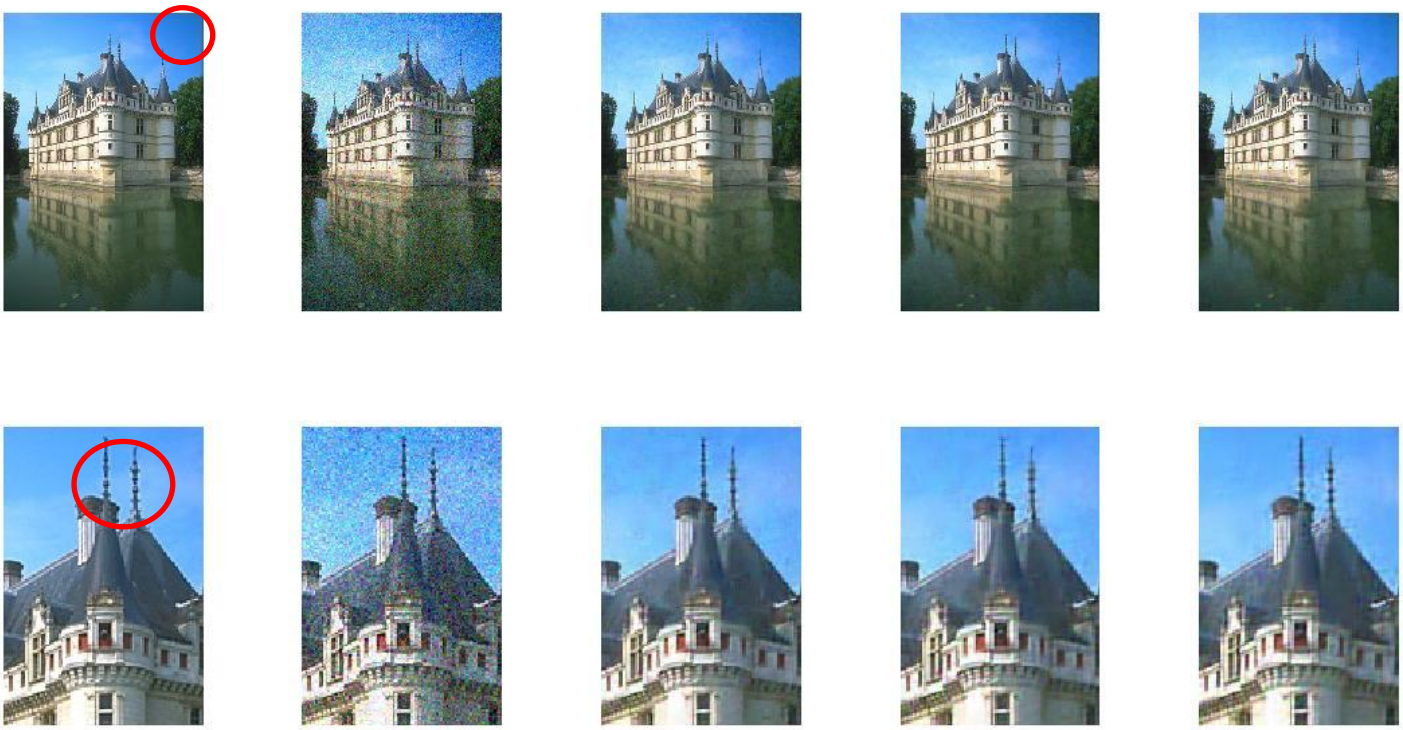


Figure 5: The image CASTLE restored by QK-SVD with varying patch-sizes for $\sigma = 25$. We use a fixed number of iterations $J = 20$ with 256 atoms ($m = 256$). For this experiment, noise was added to the real-part of the signal. Top row, from the left: $n = 6$; $n = 8$; $n = 10$. Bottom is enlargement of the top row. Notice the marked patch of sky and the spreading effect on the pillar.

The spreading effect, as expected, correlated with larger patch sizes. This implies that finer element in the image, such pillars marked above, can be better denoised with smaller patch sizes. However, if one observe patches containing mostly the sky, which are more flattened, then the large patch size leads to a smoother patch, and thus more visually satisfying.

In **[5]**, Mairal et. al. fixed the number of atoms to be $m = 256$. Since we suspect Quaternions better exploit the correlations between the channels, we tested the QK-SVD algorithm with various amounts of atoms to see if this correlations can result in a lower-redundancy for the dictionary; the results are depicted in Figure 6.

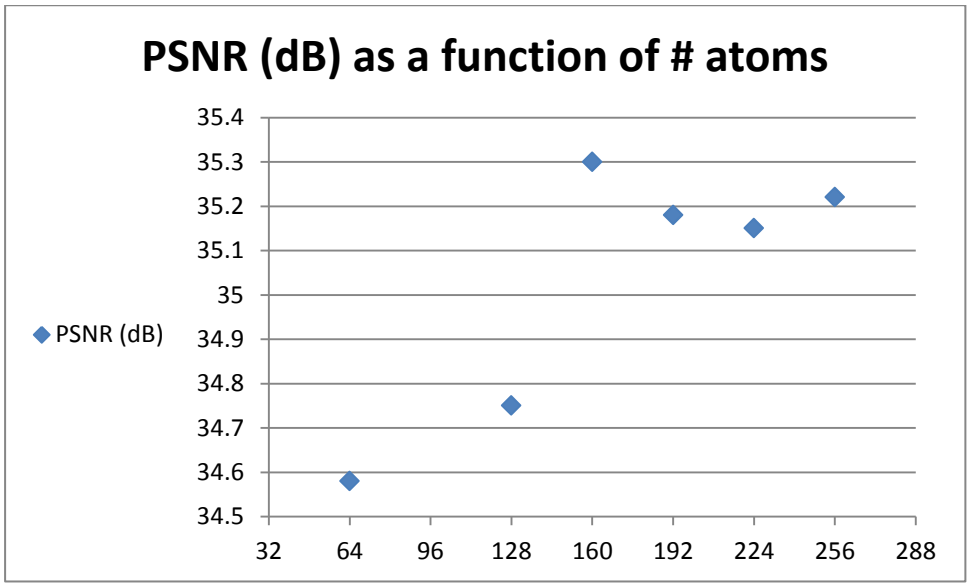


Figure 6: PSNR of the image CASTLE as function of the number of atoms, for $\sigma = 10$. We use patches of size 5×5 with a fixed number of iterations $J = 20$.

Clearly, fewer atoms are required than in **[5]**. However, in **[5]** the atoms are 3-dimensionals, of sizes $3\times6\times6$ or $3\times7\times7$, meaning that 256 atoms provide a redundancy factor of ~2.37 or ~1.74, respectively. Using 128 atoms for patches of size 5×5 with QK-SVD leads a redundancy factor of 5.12, a much higher value. We thus attempt to test QK-SVD on the same patch sizes (6×6 and 7×7) with the same redundancy factors, i.e. with 85 atoms in both cases.

Another important issue is the initial dictionary used for the algorithms. Table 1 shows results when the QK-SVD is initialized by a local dictionary. This local dictionary is composed of several patches, chosen at random. In **[5]**, Mairal et. al. used a globally-trained dictionary by applying their scheme offline on a database of images.

We now show the results of QK-SVD when initialized by the same globally-trained dictionaries used in **[5]**. We do not use the full dictionary, but instead select some atoms at random from that dictionary, primarily to maintain the same redundancy factor; the results are depicted in Table 2.

σ / PSNR	$6\times6\left(m=85\right)$	$6\times6\left(m=128\right)$	$7\times7\left(m=85\right)$	$7\times7\left(m=128\right)$
5 / 34.1590	38.5789	39.2134	38.4575	38.8542
10 / 28.1351	34.7983	35.1911	34.6087	35.0378
25 / 20.1705	30.0251	30.1970	29.8892	30.0606

Table 2: PSNR of the image CASTLE as function of the noise standard deviation σ. We use patches of sizes 6×6 and 7×7 with a fixed number of iterations $J=20$, with the initial dictionary is built by choosing m patches at random from the globally-trained dictionaries used in **[5]**.

We now move to compare the results of the QK-SVD with the work shown in **[5]**. To do so, we compare QK-SVD with 3 different schemes for color image denoising. We use a fixed number of iterations $J=20$.

The first scheme denoises each channel of the color image separately by the grayscale K-SVD algorithm described in **[3]**, **[4]**; since each channel is denoised separately, we expect the results of the QK-SVD, that exploits the correlations between channels via Quaternions, to outperform that scheme.

The second scheme follows the work in **[5]** and performs K-SVD algorithm by concatenation of the R-, B- and G-channels into a single dictionary of dimensions $3n\times m$. Thus, a single application of K-SVD algorithm denoises all 3 channels together.

Notice that this scheme is a special case of QK-SVD. The dictionary is composed of three parts corresponding to each of the channels, which is identical to using a Quaternion dictionary whose atoms have real-part zero. In general, we do not assume that the atoms have real-part zero, but rather the signals (the extracted patches) do.

Furthermore, in **[5]**, to generate a signal, the dictionary is multiplied by a single sparse representation, so each of the parts corresponding to each of the channels is multiplied by the same vector,

$$\textbf{(14)}\quad \begin{bmatrix} R \\ G \\ B \end{bmatrix} x = \begin{bmatrix} Rx \\ Gx \\ Bx \end{bmatrix}$$

and thus, this is another special case of QK-SVD, where the sparse representations are assumed to be real. However, since the real-part of the atoms is necessarily zero, it means that the standard inner-product has been altered by introducing a weight-matrix that forces a zero-real part.

As mentioned above, Mairal, Elad and Sapiro introduced a weight-matrix W incorporated in the sparse coding stage of the second scheme. W is multiplied by the dictionary and by each of the patches before performing the sparse coding stage. Once the sparse coding stage is completed, we

use the original dictionary and patches for the dictionary update stage. W is a function of some scalar parameter γ; in **[5]**, $\gamma = 5.25$ leads to the best results. $\gamma = 0$ nullifies the weight-matrix W, effectively setting $W = I$, and thus is exactly the second scheme described above.

σ / PSNR	CASTLE		MUSHROOM		HORSES	
5 / 34.15	38.01	40.36	37.58	40.06	37.12	40.17
	40.46	39.47	**39.12**	39.09	**40.26**	39.09
10 / 28.13	33.92	35.74	33.27	35.38	32.56	35.01
	36.14	35.30	**35.47**	34.71	**35.27**	34.34
25 / 20.18	28.96	30.20	28.39	29.31	27.93	29.07
	30.77	**30.24**	29.54	**29.64**	29.18	**28.96**

Table 3: PSNR of the images CASTLE, MUSHROOM and HORSES as function of the noise standard deviation σ. We use patches of size 5×5 with a fixed number of iterations $J = 20$ with 256 atoms ($m = 256$) for all schemes, except QK-SVD. For each given σ and image, the top-left results are obtained by applying the grayscale **K-SVD** image separately on each of the channels; the top-right results are obtained by the scheme in **[5]** with $\gamma = 0$; the bottom-left results are obtained by the scheme in **[5]** with $\gamma = 5.25$; the bottom-right results are obtained by the QK-SVD algorithm with patches of size 6×6 and the globally-trained dictionary from **[5]** with 160 atoms. For each σ and image, the bolded value is the best PSNR achieved amongst all algorithms.

Table 3 shows the results of the comparison for each of the images. Notice that the each of the algorithms, save QK-SVD, is initialized with dictionaries obtained by applying K-SVD on a database of 600 images. These dictionaries are the same used in **[5]** for initialization.

As can be seen from Table 3, the use of Quaternions indeed exploits the correlation between the different channels, leading for far better results than the first scheme that uses K-SVD on each of the channels separately. In terms of PSNR, the PSNR of the image restored by QK-SVD is, by average, 1.4 dB higher than the PSNR of image restored by three K-SVD applications.

Regarding the second scheme, it is clear that use of $\gamma = 5.25$ improves the results comparing to $\gamma = 0$. However, in almost all cases, when comparing the second scheme to QK-SVD, either QK-SVD was definitely better than both variants (for $\sigma = 25$) or definitely worse ($\sigma = 5,10$).

Figure 7 and Figure 8 show the results for the images CASTLE and HORSES respectively, for $\sigma = 25$. For CASTLE, surprisingly enough, the image restored by QK-SVD resembles the one restored by 3 K-SVD on each channel, though leading to a higher PSNR; compare to the image restored by the scheme in **[5]**, QK-SVD was better at retaining complicated textures, as the one at the bottom of the castle and somewhat at the lake, than the algorithm in **[5]**, which tends to flatten such textures. This comes at a cost of keeping some of the noise as well.

For HORSES, QK-SVD leads to a less-spread image. The spread sky for the image restored by **[5]** resembles the sky in the original image, however, some details on the horses themselves look less familiar to the spreading effect. Again, QK-SVD and K-SVD on each channel lead to images with similar effects, though QK-SVD is much sharper.

Figure 8 and Figure 10 compare CASTLE and MUSHROOM with the scheme purposed in **[5]** without the weight-matrix, i.e. $\gamma = 0$. Once more the algorithm in **[5]** loses some textures on the brown leaves behind the mushroom, which were better retained by QK-SVD.

Figure 7: Comparison of CASTLE image with $\sigma = 25$. Top row, from the left: original image; noisy image (PSNR is 28.13 dB); the image restored by applications of K-SVD to each channel separately (PSNR is 28.96 dB) ; the image restored by QK-SVD (PSNR is 30.24 dB); the image restored by the scheme in **[5]** (PSNR is 30.77 dB). Bottom row is an enlargement of the above row. Notice the finer details recovered by QK-SVD in comparison to the spreading effect created by the scheme in **[5]**.

Figure 8: Comparison of HORSES image with $\sigma = 25$. Top row, from the left: original image; noisy image (PSNR is 28.13 dB); the image restored by applications of K-SVD to each channel separately (PSNR is 27.93 dB). Bottom row, from the left: the image restored by QK-SVD (PSNR is 28.96 dB); the image restored by the scheme in **[5]** (PSNR is 29.18 dB).

Figure 9: The learned dictionaries for HORSES with $\sigma = 25$. From left: dictionary learned by the purposed method in **[5]** with $\gamma = 0$; dictionary learned by the purposed method in **[5]** with $\gamma = 5.25$; dictionary learned by QK-SVD. QK-SVD leads to a slightly more colored dictionary in this case.

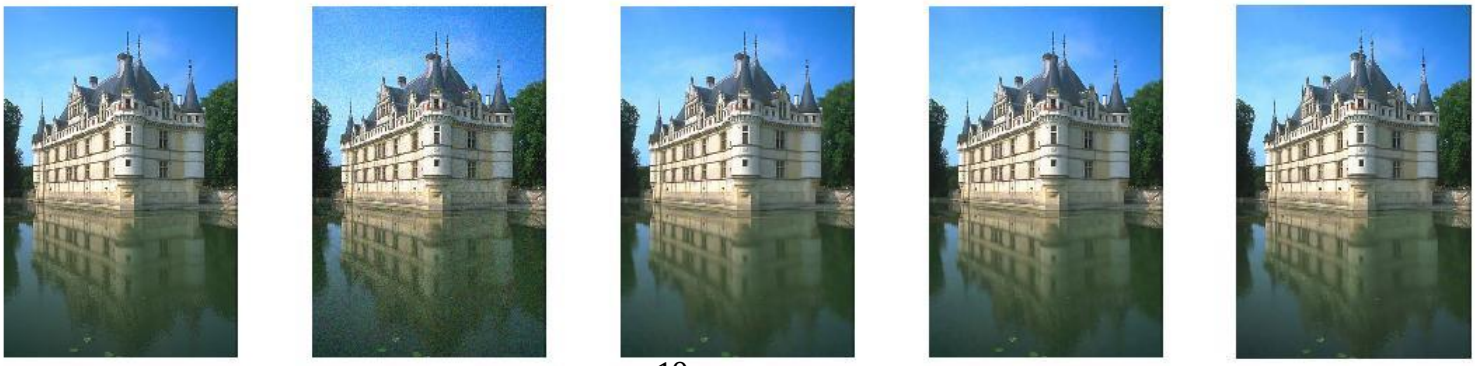


Figure 10: Comparison of CASTLE image with $\sigma = 10$. From the left: original image; noisy image (PSNR is 28.13 dB); the image restored by applications of K-SVD to each channel separately (PSNR is 33.92 dB) ; the image restored by QK-SVD (PSNR is 35.30 dB); the image restored by the scheme in **[5]** with $\gamma = 0$ (PSNR is 35.74 dB).

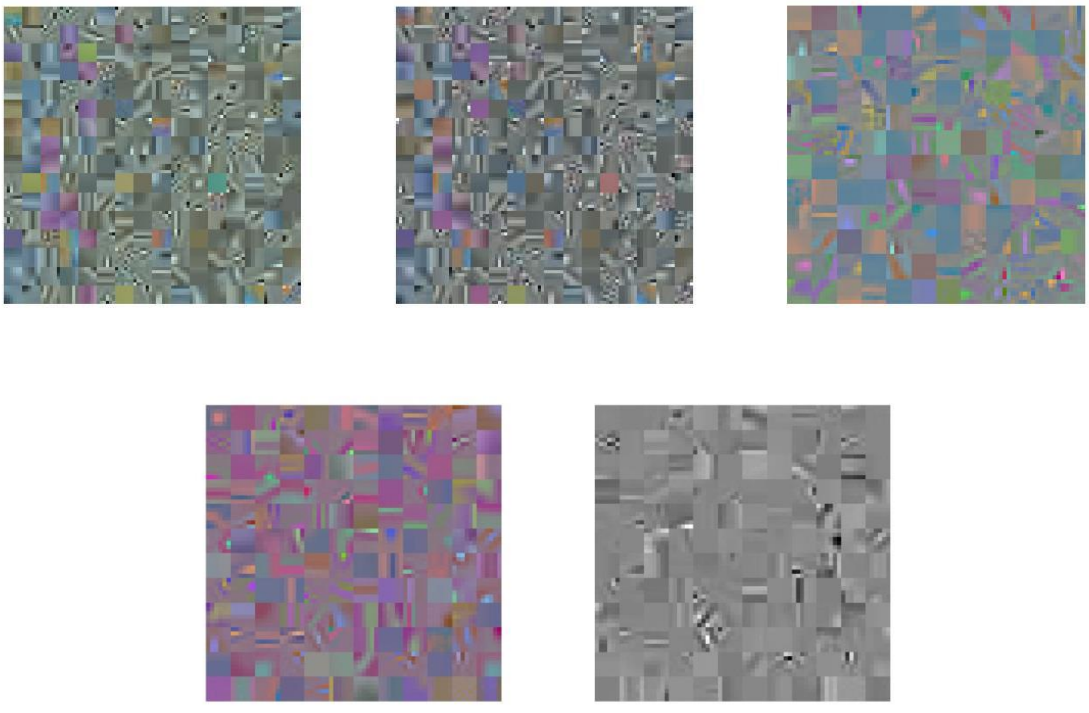
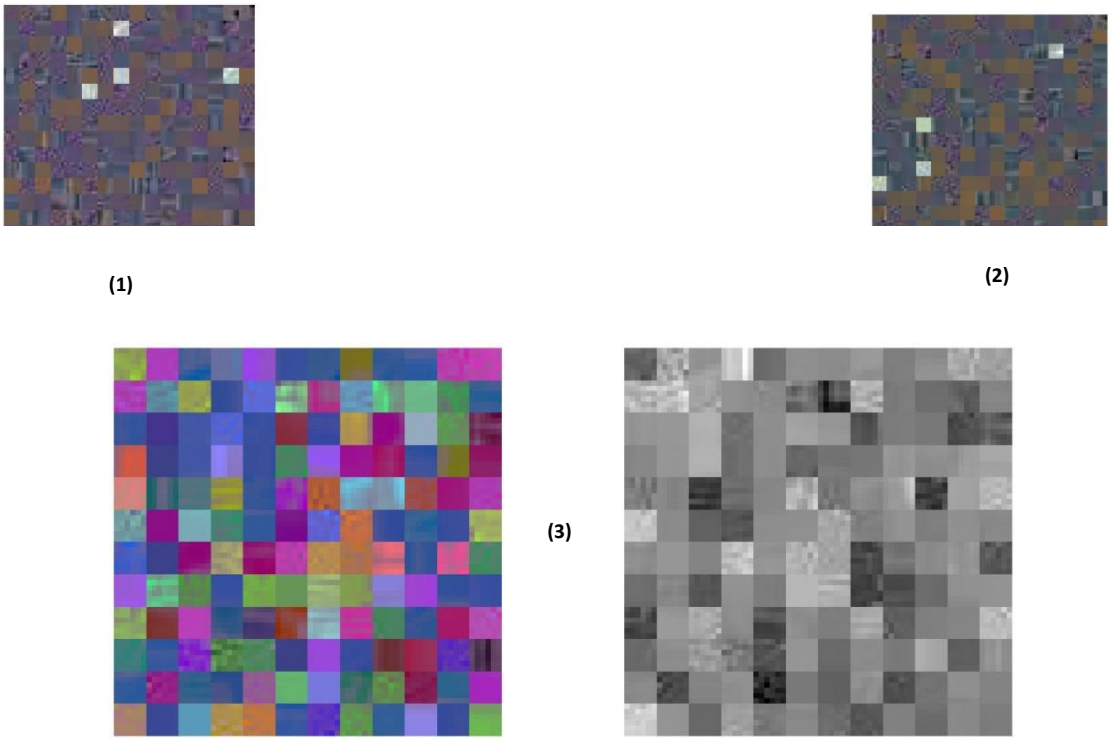


Figure 11: The learned dictionaries for CASTLE with $\sigma = 10$ .Top, from left: dictionary learned by the purposed method in **[5]** with $\gamma = 0$; dictionary learned by the purposed method in **[5]** with $\gamma = 5.25$; dictionary learned by QK-SVD. Bottom, left: The imaginary part of the dictionary learned by QK-SVD; right: The real-part of the dictionary learned by QK-SVD. QK-SVD has a little-less grayish look, with fewer gray atoms.

Figure 12: Comparison of MUSHROOM image with $\sigma = 10$. From the left: original image; noisy image (PSNR is 28.13 dB); the image restored by applications of K-SVD to each channel separately (PSNR is 33.27 dB); the image restored by QK-SVD (PSNR is 34.71 dB); the image restored by the scheme in **[5]** with $\gamma = 0$ (PSNR is 35.38 dB).

As can be seen from Figure 9 and Figure 11, the dictionaries learned by the scheme in **[5]** have a lot of gray atoms, as well as colored atoms. The dictionaries learned by the QK-SVD algorithm have less gray atoms, although they are not as colorful as one would expect. The atoms are mostly blue- or purple-colored, with a gray-shading. Notice that since the dictionary is Quaternion, we present only its imaginary parts as a color image.

Comparing the results of the dictionaries learned by using a local dictionary instead of a globally-trained one reveals that the dictionary learned by QK-SVD is by far more colorful than the one learned by the scheme in **[5]**, as shown in Figure 13.



(1) **(2)** **(3)**

Figure 13: The learned dictionaries for CASTLE with $\sigma = 10$. **(1):** dictionary learned by the purposed method in **[5]** with $\gamma = 0$; **(2):**dictionary learned by the purposed method in **[5]** with $\gamma = 5.25$; **(3):** dictionary learned by QK-SVD, left is the imaginary parts and right is the real parts. QK-SVD is the most colorful dictionary.

8. Conclusions and Further Research:

As one can see from the experiments above, the QK-SVD algorithm indeed serves as a denoising algorithm for color images. QK-SVD is better at retaining certain textures in the image, whereas the method purposed in **[5]** tends to flatten them. The dictionaries learned by QK-SVD have fewer gray atoms that those learned by the methods in **[5]**, though color is more distinguishable in those dictionaries. The most important conclusion, perhaps, is that Quaternions allow a natural extension of established grayscale image processing algorithms for color images, as well.

Furthermore, since the globally-trained dictionaries were designed for the scheme introduced in **[5]**, it is possible that using a globally-trained Quaternion dictionary will lead to better results in the denoising of color images, similarly to how those dictionaries improved much the results for the work in **[5]**.

We have demonstrated the use of Quaternions within the field sparse and redundant representations by introducing the QK-SVD algorithm. This work shows an application of QK-SVD in the form of color image denoising; however, we can also use the same algorithm to solve other image processing problems, such as color image demosaicing and color image inpainting, and even to color image denoising with non-uniform noise. These problems can be also be addressed via the model of sparse and redundant representations, and the addition of Quaternions for color images and the use of QK-SVD seems too natural to ignore.

While the use Quaternions in the QK-SVD algorithm is rather straightforward, it is possible to consider their use in other methods dealing with sparse representations. Specifically, since only the representation of the image is changed and not the conceptual algorithm, we can implement various grayscale algorithms, such as the Non-Local Means (NLM), to operate on Quaternion matrices instead of real ones, thus adapting them to work for color images as well. We can also explore the usage of the real-part of the Quaternion.

Another idea is to employ the joint sparsity model; this model is a generalization of the sparsity model introduced in this work, where signals $y_1,...,y_p \in \Gamma$ are composed of a *common* sparse representation x_0 and a *unique* one $x_1,...,x_p$. Thus, each signals can be described as $y_i = D(x_0 + x_i)$, where D is the dictionary. Joint sparsity can be used in images by attempting to extract sparse codings of similar patches, and incorporate Quaternions to do so even for color images.

In addition, a major hinder to the use of Quaternions it performance. Further research can aim to implement a batched Q-OMP to quicken the applications of sparse coding, as well performing additional optimizations for the implementation of the dictionary update stage. Specifically, implement accessing to indices and restriction of the residual matrices can be implemented significantly faster via mex, instead of MatLab.

Appendix A: Basic Quaternion Theory

The Quaternion space, denoted $\mathbb{H}$, can be thought of as an extension of the complex space $\mathbb{C}$, and is in fact, referred to as the hyper-complex space in some references. Each Quaternion element is composed of four real values x, y, z, w, and the Quaternion itself is written as $x+iy+jz+kw$, with i, j and k representing directions in the Quaternion space, similarly to how i is a direction orthogonal to the real-space in $\mathbb{C}$.

The connections between i, j and k are well-defined, and are described algebraically, in **(1)**.

$$\textbf{(1)}\quad \begin{aligned} ij &= k & ji &= -k \\ ik &= -j & ki &= j \\ jk &= i & kj &= -i \end{aligned}$$

Notice that **(1)** shows that the Quaternion space has non-commutative multiplication. This means that $\mathbb{H}$ is not a field, but a ring.

1. Quaternion Conjugate:

For every Quaternion $v \in \mathbb{H}, v = x+iy+jz+kw$, we define the **Quaternion conjugate** of v, denoted $\overline{v}$, as $\overline{v} = x-iy-jz-kw$. The definition resembles the one that exists in complex space, and in fact, we can write a theorem similar to the one that holds in the complex space.

Theorem 1: Quaternion Conjugate Properties

The following properties hold in $\mathbb{H}$, $\forall u, v \in \mathbb{H}$:

- $\overline{(u+v)} = \overline{u} + \overline{v}$
- $\overline{(u \cdot v)} = \overline{v} \cdot \overline{u}$
- $|u|^2 \overset{\Delta}{=} \overline{u} \cdot u = u \cdot \overline{u} = x^2 + y^2 + z^2 + w^2$

Theorem 1 can be easily verified by the use of **(1)**. Observe that $|u| \geq 0$ and $|u| = 0$ if and only if $x = y = z = w = 0$, i.e. $v = 0 \in \mathbb{H}$. This suggests that we can define an inner-product over vectors in $\mathbb{H}^n$ using the notation of the Quaternion conjugate.

2. Quaternion Inner-Product:

Theorem 2: Quaternion Inner-Product

Let $\langle x, y \rangle : \mathbb{H}^n \times \mathbb{H}^n \to \mathbb{H}$ be a function, defined as $\langle x, y \rangle = \sum_{i=1}^{n} x_i \overline{y}_i$. Then $\langle x, y \rangle$ is a proper inner-product over $\mathbb{H}^n$.

Proof:

For $\langle x, y \rangle$ to be a proper inner-product, the following properties must hold:

- Homogeneity: $\forall \alpha \in \mathbb{H}, \forall x, y \in \mathbb{H}^n : \langle \alpha x, y \rangle = \sum_{i=1}^{n} \alpha x_i \overline{y}_i = \alpha \sum_{i=1}^{n} x_i \overline{y}_i = \alpha \langle x, y \rangle$
- Linearity: $\forall x, y, z \in \mathbb{H}^n : \langle x, y + z \rangle = \sum_{i=1}^{n} x_i \overline{(y + z)}_i = \sum_{i=1}^{n} x_i \overline{(y_i + z_i)} =$

  $\sum_{i=1}^{n} x_i (\overline{y}_i + \overline{z}_i) = \sum_{i=1}^{n} x_i \overline{y}_i + \sum_{i=1}^{n} x_i \overline{z}_i = \langle x, y \rangle + \langle x, z \rangle$
- Conjugate symmetry: $\forall x, y \in \mathbb{H}^n : \langle x, y \rangle = \sum_{i=1}^{n} x_i \overline{y}_i = \sum_{i=1}^{n} \overline{(y_i \overline{x}_i)} =$

  $\overline{\sum_{i=1}^{n} (y_i \overline{x}_i)} = \overline{\langle y, x \rangle}$, courtesy of Theorem 1.
- Norm non-negativity: $\forall x \in \mathbb{H}^n : \langle x, x \rangle = \sum_{i=1}^{n} x_i \overline{x}_i = \sum_{i=1}^{n} |x_i| \geq 0$. In addition,

  $\langle x, x \rangle = 0 \Leftrightarrow \forall i = 1, \ldots, n : x_i = 0 \Leftrightarrow x = 0 \in \mathbb{H}^n$.

Thus, $\langle x, y \rangle$ is a proper inner-product over $\mathbb{H}^n$, as expected, and we can define $\|x\|_2 = \sqrt{\langle x, x \rangle}$ as the Euclidean norm over the Quaternion space. Armed with the new definitions, we can continue exploring the Quaternion space.

Theorem 3: Quaternion Pythagorean Theorem:

Let $x, y \in \mathbb{H}^n$. Then $\|x + y\|_2^2 = \|x\|_2^2 + 2\,\mathrm{Re}\langle x, y \rangle + \|y\|_2^2$.

Proof:

Let $x = u_1 + iv_1 + jw_1 + kz_1, y = u_2 + iv_2 + jw_2 + kz_2$, for some real vectors $u_j, v_j, w_j, z_j \in \mathbb{R}^n, j = 1, 2$. Then, by definition of the Euclidean norm:

$$\|x + y\|_2^2 = \sum_{i=1}^{n} |x_i + y_i|^2 = \sum_{i=1}^{n} |(u_{1i} + u_{2i}) + i(v_{1i} + v_{2i}) + j(w_{1i} + w_{2i}) + k(z_{1i} + z_{2i})|^2 =$$

$$= \sum_{i=1}^{n} (u_{1i} + u_{2i})^2 + (v_{1i} + v_{2i})^2 + (w_{1i} + w_{2i})^2 + (z_{1i} + z_{2i})^2 =$$

$$= \underbrace{\sum_{j=1}^{2} \sum_{i=1}^{n} (u_{ji}{}^2 + v_{ji}{}^2 + w_{ji}{}^2 + z_{ji}{}^2)}_{= \|x\|_2^2 + \|y\|_2^2} + 2 \sum_{i=1}^{n} \underbrace{u_{1i}u_{2i} + v_{1i}v_{2i} + w_{1i}w_{2i} + z_{1i}z_{2i}}_{\mathrm{Re}(\overline{x}_i y_i)} =$$

$$= \|x\|_2^2 + \|y\|_2^2 + 2\,\mathrm{Re}\langle x, y \rangle$$

3. Quaternion Matrices:

Similarly to the definition of Quaternion conjugate, we can define **Quaternion hermitian-conjugate** of a Quaternion matrix $A \in \mathbb{H}^{m\times n}$. The Quaternion hermitian-conjugate of A, denoted A^*, satisfies $A^* \in \mathbb{H}^{n\times m}$ and $(A)_{ij} = \overline{(A)_{ji}}$.

Theorem 4:

Let $A \in \mathbb{H}^{m\times n}, B \in \mathbb{H}^{n\times k}$. Then $(AB)^* = B^* A^*$.

The proof of Theorem 4 is left to the reader, and follows immediately from Theorem 3 and properties of matrix multiplication.

4. Least-Squares over the Quaternion Space:

The least-squares problem, known as data fitting, aims to find a vector x that minimizes the Euclidean norm of the expression

$$\textbf{(2)} \quad \|Ax-b\|_2^2$$

Over the complex space $\mathbb{C}$, it is a well-known fact that in order to minimize **(2)**, one must solve the so-called normal equations:

$$\textbf{(3)} \quad A^*Ax = A^*b$$

We aim to show that **(3)** also holds for the minimizer of **(2)**, even if the matrices and vectors are taken over the Quaternion space, i.e. we show that the solution for

$$(LS_{\mathbb{H}}) \quad \min_{x\in\mathbb{H}^m} \|Ax-b\|_2^2$$

must also satisfy the normal equations in **(3)**.

Theorem 5:

Let $A \in \mathbb{H}^{m\times n}$. Define $\mathcal{N}(A) = \{x \in \mathbb{H}^n \mid Ax = 0\}$. Then $\mathcal{N}(A) = \mathcal{N}(A^*A)$.

Proof:

Let $x \in \mathcal{N}(A)$. Then $A^*Ax = A^*(Ax) = A^*0 = 0$, so $x \in \mathcal{N}(A^*A)$, which completes the first part of the proof. To complete the second part, let $x \in \mathcal{N}(A^*A)$. Then $0 = x^*A^*Ax = \langle Ax, Ax\rangle$. But $\langle Ax, Ax\rangle = 0 \Leftrightarrow Ax = 0$, so $x \in \mathcal{N}(A)$.

Theorem 6: Solution of Least-Squares over Quaternion Space

Let $A \in \mathbb{H}^{m\times n}, b \in \mathbb{H}^n$. Then the problem $(LS_{\mathbb{H}})$ has at least one solution $\hat{x} \in \mathbb{H}^m$, which satisfies the so-called normal equations, $A^*Ax = A^*b$. In addition, if $\tilde{x}$ is another solution of $(LS_{\mathbb{H}})$, then $\hat{x} - \tilde{x} \in \mathcal{N}(A)$.

Proof:

A solution to $\left(LS_{\mathbb{H}}\right)$ must exist; let $\alpha \in \mathbb{H}^n$ be some arbitrary vector, and denote $f_\alpha = \left\|A\alpha - b\right\|_2^2$. A minimizer $\hat{x}$ of $\left\|Ax-b\right\|_2^2$ must satisfy

$$\textbf{(4)}\ \left\|A\hat{x}-b\right\|_2^2 \le f_\alpha$$

and thus, we can add **(4)** as a constraint to $\left(LS_{\mathbb{H}}\right)$ without changing the set of optimal solutions to $\left(LS_{\mathbb{H}}\right)$. Now, and the set of feasible solution is bounded, and since the Euclidean norm is a continuous function, a minimum must exist to $\left(LS_{\mathbb{H}}\right)$.

Denote the solution as $\hat{x} \in \mathbb{H}^m$. Let $x \in \mathbb{H}^m$. Then: $Ax-b = A\left(x-\hat{x}\right)+\left(A\hat{x}-b\right)$.

Denote $x-\hat{x} = y, A\hat{x}-b = r$. Then we have:

$$Ax-b = Ay+r$$

$$\left\|Ax-b\right\|_2^2 = \left\|Ay+r\right\|_2^2 = \left\|Ay\right\|_2^2 + \left\|r\right\|_2^2 + 2\,\mathrm{Re}\left\langle Ay, r\right\rangle$$

Since $\hat{x}$ is the solution, for any $x \in \mathbb{H}^n$, $\left\|r\right\|_2^2 \le \left\|Ax-b\right\|_2^2$. Thus,

$\left\|Ay\right\|_2^2 + 2\,\mathrm{Re}\left\langle Ay, r\right\rangle \ge 0$. Obviously, $\left\langle Ay, r\right\rangle = \left(Ay\right)^* r = y^* A^* r$.

If $A^* r = 0$, this is equivalent to $A^*\left(A\hat{x}-b\right) = 0$, which is the desired result. Otherwise, let $x \in \mathbb{H}^n$ be a vector, such that there exists an $\varepsilon \ge 0, \varepsilon \in \mathbb{R}$, with $y = \varepsilon A^* r$. Setting this expression for y in the above inequality yields:

$$0 \le \left\|Ay\right\|_2^2 + 2\,\mathrm{Re}\left\langle Ay, r\right\rangle =$$

$$\textbf{(5)}\ \left\|\varepsilon A\left(A^* r\right)\right\|_2^2 + 2\,\mathrm{Re}\left(\varepsilon A^* r\right)^*\left(A^* r\right) =$$

$$\varepsilon^2\left\|AA^* r\right\|_2^2 + 2\varepsilon\left\|A^* r\right\|_2^2$$

This is a convex quadratic function of ε, with the minimum value achieved at $\varepsilon = \dfrac{-\left\|A^* r\right\|_2^2}{\left\|AA^* r\right\|_2^2}$ where the value of this quadratic function is:

$$\textbf{(6)}\ \frac{\left\|A^* r\right\|_2^4}{\left\|AA^* r\right\|_2^2} - 2\frac{\left\|A^* r\right\|_2^4}{\left\|AA^* r\right\|_2^2} = -\frac{\left\|A^* r\right\|_2^4}{\left\|AA^* r\right\|_2^2} < 0.$$

Thus, for a choice of ε small enough, the inequality does not hold, which mean $A^* r = 0$ **must hold**.

The second part of this theorem is immediate; If $\tilde{x}$ is another solution of $\left(LS_{\mathbb{H}}\right)$, then $A^{*}\left(A\tilde{x}-b\right)=0, A^{*}\left(A\hat{x}-b\right)=0$. Thus, $A^{*}\left(A\left(\hat{x}-\tilde{x}\right)\right)=A^{*}A\left(\hat{x}-\tilde{x}\right)=0$, which implies $\hat{x}-\tilde{x}\in\mathcal{N}\left(A\right)$.

Appendix B: Quaternion OMP

The Q-OMP algorithm is a pursuit algorithm that aims to approximate the solution to the problem:

$$\textbf{(1)} \qquad \min_{x\in\mathbb{R}^m} \ \|x\|_0 \quad s.t. \quad \|y-Dx\|_2^2 \le \varepsilon^2$$

The Q-OMP algorithm is shown below;

Input: $D\in\mathbb{R}^{n\times m}, y\in\mathbb{R}^n, \varepsilon\ge 0, T>0$

3. $k\leftarrow 0, x^0\leftarrow 0, r^0\leftarrow y, \mathcal{S}^0=\phi$
4. while $\|r^k\|_2>\varepsilon$ and $k<T$, do:
   - 2.1 $k\leftarrow k+1$
   - 2.2 For all $j\notin\mathcal{S}^{k-1}$, compute $v(j)=\min_{z_j}\|d_j z_j - r^{k-1}\|_2^2$.
   - 2.3 $j_0=\arg\min_{j\notin\mathcal{S}^{k-1}} v(j)$
   - 2.4 $\mathcal{S}^k\leftarrow\mathcal{S}^{k-1}\cup\{j_0\}$
   - 2.5 $x^k\leftarrow\arg\min_{x\in\mathbb{R}^m}\|y-Dx\|_2^2 \quad s.t. \quad Support\{x\}=\mathcal{S}^k$
   - 2.6 $r^k\leftarrow y-Dx^k$

Implementation of steps **2.2** and **2.5** requires additional care, due to the use of Quaternion matrices and vectors.

For step **2.2**, we need to solve a one-dimensional Quaternion least-squares problem. Solving such a problem is no harder than solving it within the complex space, and simply requires solution of the normal equations (see Appendix A). Thus, the minimizer of $\|d_j z_j - r^{k-1}\|_2^2$ is given by:

$$\textbf{(2)} \quad z=\frac{d_j^* r^{k-1}}{d_j^* d_j}=\frac{d_j^* r^{k-1}}{\|d_j\|_2^2}$$

Setting **(3)** in the expression $\|d_j z_j - r^{k-1}\|_2^2$ will yield the minimum Euclidean norm of the residual the j^{th} atom:

$$\left\|d_j\frac{d_j{}^* r^{k-1}}{d_j{}^* d_j}-r^{k-1}\right\|_2^2=\left\|d_j\frac{d_j{}^* r^{k-1}}{\|d_j\|_2^2}-r^{k-1}\right\|_2^2=\left\|d_j\frac{d_j{}^* r^{k-1}}{\|d_j\|_2^2}\right\|_2^2+\left\|-r^{k-1}\right\|_2^2+2\,\mathrm{Re}\left\langle d_j\frac{d_j{}^* r^{k-1}}{\|d_j\|_2^2},-r^{k-1}\right\rangle=$$

$$=\frac{1}{\|d_j\|_2^2}\left\|d_j d_j{}^* r^{k-1}\right\|_2^2+\left\|r^{k-1}\right\|_2^2-\frac{2}{\|d_j\|_2^2}\mathrm{Re}\left\langle d_j d_j{}^* r^{k-1}, r^{k-1}\right\rangle=$$

$$=\frac{\left\|d_j d_j{}^* r^{k-1}\right\|_2^2}{\|d_j\|_2^2}+\left\|r^{k-1}\right\|_2^2-\frac{2}{\|d_j\|_2^2}\mathrm{Re}\left(\left(d_j d_j{}^* r^{k-1}\right)^* r^{k-1}\right)=$$

$$= \frac{\left\| d_j d_j^{\ *} r^{k-1} \right\|_2^2}{\left\| d_j \right\|_2^2} + \left\| r^{k-1} \right\|_2^2 - \frac{2}{\left\| d_j \right\|_2^2} \mathrm{Re}\left(\left(r^{k-1} \right)^* d_j d_j^{\ *} r^{k-1} \right) =$$

$$= \frac{\left\| d_j d_j^{\ *} r^{k-1} \right\|_2^2}{\left\| d_j \right\|_2^2} + \left\| r^{k-1} \right\|_2^2 - \frac{2}{\left\| d_j \right\|_2^2} \mathrm{Re}\left(\left(d_j^{\ *} r^{k-1} \right)^* d_j^{\ *} r^{k-1} \right) = \frac{\left\| d_j d_j^{\ *} r^{k-1} \right\|_2^2}{\left\| d_j \right\|_2^2} + \left\| r^{k-1} \right\|_2^2 - \frac{2}{\left\| d_j \right\|_2^2} \left| d_j^{\ *} r^{k-1} \right|^2 =$$

$$= \frac{1}{\left\| d_j \right\|_2^2} \left(\left\| d_j d_j^{\ *} r^{k-1} \right\|_2^2 - 2 \left| d_j^{\ *} r^{k-1} \right|^2 \right) + \left\| r^{k-1} \right\|_2^2 = \left\| r^{k-1} \right\|_2^2 + \left(1 - \frac{2}{\left\| d_j \right\|_2^2} \right) \left| d_j^{\ *} r^{k-1} \right|^2$$

Thus, the Q-OMP algorithm chooses the atom the minimizes

$$\textbf{(3)} \quad \left\| r^{k-1} \right\|_2^2 + \left(1 - \frac{2}{\left\| d_j \right\|_2^2} \right) \left| d_j^{\ *} r^{k-1} \right|^2$$

Since $\left\| r^{k-1} \right\|_2^2$ is fixed at each iteration, it suffices to maximize $\left(1 - \frac{2}{\left\| d_j \right\|_2^2} \right) \left| d_j^{\ *} r^{k-1} \right|^2$. However, this expression reduces to $\left| d_j^{\ *} r^{k-1} \right|^2$, when the columns of D are normalized, i.e. $\left\| d_j \right\|_2^2 = 1$.

For step **2.5**, we solve least-squares over Quaternion space, but we wish to do so differently, without the aid of normal equations. We first write $D = D_1 + iD_2 + jD_3 + kD_4$ and $y = y_1 + iy_2 + jy_3 + ky_4$, with D_1, D_2, D_3, D_4 being real matrices and y_1, y_2, y_3, y_4 real vectors, and let $x = x_1 + ix_2 + jx_3 + kx_4$ real vectors x_1, x_2, x_3, x_4. Then, we can write $y - Dx$ as:

$$\begin{aligned} y - Dx = y - \left(D_1 + iD_2 + jD_3 + kD_4 \right) \left(x_1 + ix_2 + jx_3 + kx_4 \right) = \\ y_1 - \left(D_1 x_1 - D_2 x_2 - D_3 x_3 - D_4 x_4 \right) + \\ y_2 i - \left(D_2 x_1 + D_1 x_2 - D_4 x_3 + D_3 x_4 \right) i + \\ y_3 j - \left(D_3 x_1 + D_4 x_2 + D_1 x_3 - D_2 x_4 \right) j + \\ y_4 k - \left(D_4 x_1 - D_3 x_2 + D_2 x_3 + D_1 x_4 \right) k \end{aligned} \quad \textbf{(4)}$$

The Euclidean norm of a Quaternion vector is the sum of the Euclidean norms of its real part, $i-$ part, $j-$ part and $k-$ part. Thus, we have:

$$\textbf{(5)} \quad \left\| y - Dx \right\|_2 = \left\| \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} - \begin{pmatrix} D_1 & -D_2 & -D_3 & -D_4 \\ D_2 & D_1 & -D_4 & D_3 \\ D_3 & D_4 & D_1 & -D_2 \\ D_4 & -D_3 & D_2 & D_1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \right\|_2$$

The expression in **(5)** allows us compute the entries of x by directly solving a real least-squares problem. Since **(5)** is merely a different formulation for the same objective function $\left\| y - Dx \right\|_2^2$, computing the minimum via **(5)** will also result in the minimum for $\left\| y - Dx \right\|_2^2$.

To implement step **2.5**, we also need to assure that x is subject to the computed support $\mathcal{S}^k$. To do so, instead of using the full D, we use $D_{\mathcal{S}^k}$ - the sub-matrix of D that is composed of all columns that are part of $\mathcal{S}^k$. Thus, we only compute the entries of x that correspond to $\mathcal{S}^k$, setting all other entries to zero, instead.

Appendix C: Quaternion Rank-One Approximation

Let $E \in \mathbb{H}^{m \times n}$ be a hyper-complex matrix, and we wish to approximate it by rank-one approximation, i.e. by a multiplication of the form xy^T, with $x \in \mathbb{H}^m, y \in \mathbb{H}^n$, whereas the target is minimization of the Frobenius norm:

$$\min_{x \in \mathbb{H}^m, y \in \mathbb{H}^n} \left\| E - xy^T \right\|_F^2$$

It is well known that such an optimization problem is solvable by the singular-value decomposition (SVD). The solution requires to zero all singular values of E, except for the largest one, and thus, a rank one approximation is achieved. However, we wish to approximate the solution of this optimization problem by alternating minimization, without the use of SVD.

Denote $f(x,y) = \left\| E - xy^T \right\|_F^2$. The function is convex, so minimization is equivalent to zeroing the gradient. Since we use alternating minimization, we need to zero its gradient with respect to one of the variables.

Assume $E = \left[E_1 \left| \ldots \right| E_n \right]$, with E_1 being a column of E. Then:

$f(x,y) = \left\| E - xy^T \right\|_F^2 = \left\| \left[E_1 \left| \ldots \right| E_n \right] - xy^T \right\|_F^2 = \sum_{i=1}^{n} \left\| E_i - xy_i \right\|_2^2$. Let $g_i : \mathbb{H}^m \times \mathbb{H} \to \mathbb{R}$ be the function defined by $g_i(x,y) = \left\| E_i - xy \right\|_2^2$, then, obviously, $f(x,y) = \sum_{i=1}^{n} g_i(x, y_i)$. To minimize $f(x,y)$ over y, we can exploit the fact the function is separable and minimize $g_i(x,y)$ over y.

We have shown that some of the traits of complex linear least-squares are retained even when the matrices and vectors have hyper-complex elements. Thus, we know that in order to minimize $g_i(x,y)$, it is sufficient and necessary that $x^* \left(E_i - xy \right) = 0$, so, $y_i = \dfrac{x^* E_i}{\left\| x \right\|_2^2}$.

Thus, $y^T = \dfrac{1}{\left\| x \right\|_2^2} \begin{pmatrix} x^* E_1 & \cdots & x^* E_n \end{pmatrix}$, or, $y^T = \dfrac{x^* E}{\left\| x \right\|_2^2}$.

How to deal with minimization over x? Observe that $f(x,y) = \left\| E^T - yx^T \right\|_F^2$. Using the same analysis as before leads to a similar expression for x^T, in $x^T = \dfrac{y^* E^T}{\left\| y \right\|_2^2}$, or, $x = \dfrac{E\overline{y}}{\left\| y \right\|_2^2}$.

References:

[1] M. Elad, "Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing", Springer, 2010

[2] M. Aharon, M. Elad, and A. M. Bruckstein, “The K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representations”, IEEE Trans. Image Process., vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[3] M. Elad and M. Aharon, “Image denoising via learned dictionaries and sparse representation", presented at the IEEE Computer Vision and Pattern Recognition, New York, Jun. 2006.

[4] M. Elad and M. Aharon, “Image denoising via sparse and redundant representations over learned dictionaries", IEEE Trans. Image Process., vol. 15, no. 12, pp. 3736–3745, Dec. 2006.

[5] J. Mairal, M. Elad and G. Sapiro, "Sparse representation for color image restoration", IEEE Trans. Image Process., vol .17, no. 1, pp. 53-69, Jan. 2008.

[6] N. Bihan, J. Mars, "Singular value decomposition of quaternion matrices: a new tool for vector-sensor signal processing", Signal Processing, vol. 84, no. 7, July 2004, pp. 1177-1199

[7] "Quaternion", from Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Quaternion