



## $Q_K R_K$ FACTORIZATION FOR IMAGE COMPRESSION

Erik Eckenberg<sup>1</sup> and Knarik Tunyan<sup>2</sup>

<sup>1</sup> Faculty of Science, New Jersey Institute of Technology  
University Heights, Newark, 07102 New Jersey, USA

<sup>2</sup> Department of Mathematics and Computer Science, Purchase College  
State University of New York, Purchase, 10577 New York, USA

ORCID IDs: Erik Eckenberg  
Knarik Tunyan

 <https://orcid.org/0000-0002-2276-8710>  
 <https://orcid.org/0009-0008-6281-5471>

**Abstract.** We store and exchange more digital images than ever before. Image qualities are often reduced to decrease the cost of storage and transfer of the images. Digital image compression is a technique that reduces the size of an image while preserving its quality to be acceptable for a particular purpose. Matrix factorizations are widely used for this purpose. This paper presents an application of the  $Q_K R_K$  and block  $Q_K R_K$  factorizations of a matrix to image compression. We have conducted a series of experiments using Matlab software. This paper presents the comparative analysis of the compressed images using the  $QR$ ,  $SVD$ ,  $Q_K R_K$ , and block  $Q_K R_K$  factorizations. The similarity between the original and compressed images is measured using the  $L_2$ -norm and structural similarity. It is demonstrated that using the  $Q_K R_K$  factorization for image compression allows the achievement of the desired quality of a fragment of the image compared to the rest of the image and is also computationally efficient.

**Keywords:**  $Q_K R_K$  factorization,  $QR$  factorization,  $SVD$  factorization, block  $Q_K R_K$  factorization.

### 1. Introduction

Nowadays, the number of digital photos taken, stored, and shared on a daily basis is increasing exponentially. Therefore, there is an enormous demand for effective and efficient image compression algorithms that reduce the size of the images while

---

Received August 25, 2021, revised: August 05, 2024, accepted: August 08, 2024

Communicated by Marko Petković

Corresponding Author: Knarik Tunyan. E-mail addresses: [ete2@njit.edu](mailto:ete2@njit.edu) (E. Eckenberg), [knarik.tunyan@purchase.edu](mailto:knarik.tunyan@purchase.edu) (K. Tunyan)

2020 *Mathematics Subject Classification*. Primary 15A23, 68U10, 94A08; Secondary 15A10

© 2024 BY UNIVERSITY OF NIŠ, SERBIA | CREATIVE COMMONS LICENSE: CC BY-NC-ND

preserving their quality or minimizing transmission time. There are two main image compression techniques, lossless and lossy [10]. The lossless compression algorithms reduce the image size, preserving the data, while the lossy compression techniques eliminate redundant data from the images without compromising the image quality. The choice of the image compression algorithm depends on the objective of the application. Many applications utilize image compression methods: facial recognition, blurry and noisy image restoration, removing an object from an image in machine learning, and digital image watermarking [2, 4, 12, 18, 19, 20], just to name a few.

Matrix factorizations, such as  $LU$  and  $QR$  decompositions, Cholesky factorization, and singular value decomposition ( $SVD$ ), are widely used in digital signal and image processing. The idea of using matrix factorization in the lossy image compression is based on constructing a new approximated matrix of a lower rank to approximate the original image matrix. This approach allows the elimination of unnecessary data and extracts a piece of vital information from an image.

The effectiveness and efficiency of various matrix factorizations in image processing have been extensively studied [13, 15, 17, 18, 19]. The theoretical and experimental results demonstrate that using  $SVD$  leads to a better approximation in terms of the quality of the compressed image, as it allows retrieval of the principal information from the data effectively. The  $QR$  decomposition is computationally more efficient than  $SVD$  [16, 17].

Many extensions and variations of the  $SVD$  and  $QR$  based image compression methods have been proposed, analyzed, and compared. For example, in [25], it was shown that  $QR$  with column pivoting, although more computationally expensive, produces better image approximations than  $QR$  without using the pivoting rule. In [1], an image compression method was proposed based on the block  $SVD$  power method. Several quality measurements of the compressed images, such as mean square error, compression ratio, and peak signal-to-noise ratio, have been computed and compared with the results obtained by using MATLAB's  $SVD$  function and other in the state-of-the-art compression techniques. The author demonstrated that the proposed method produces visually similar images but computationally is more efficient and provides a better compression ratio and peak signal-to-noise values.

The current intense area of research in image compression is based on the recent development of the randomized matrix factorizations technique [8, 15]. The idea behind the randomized matrix factorizations is to use random sampling to extract the most significant part of an original matrix and then apply standard factorizations. Theoretical error analysis and computational experiments show that using the randomized  $SVD$  and  $QR$  in image compression is computationally faster, while the image reconstruction errors are similar to the errors produced by the other restoration techniques [14, 25].

The methods described above deal with the overall quality of the image. However, in some applications, such as partial image encryption, facial recognition, blurring, and masking [11], the objectives are to keep only a part of the image at its best possible quality or encrypt only a specific part of the image. The standard approach

to this class of problems involves adding data to the original image resulting in a higher dimensionality or computational inefficiency. Moreover, the methods based on  $SVD$ ,  $QR$ , and other matrix decompositions and their variations are not flexible enough to accommodate these goals. On the other hand, the parametric nature of the  $Q_K R_K$  matrix factorization and its block version [22] allows us to extract a subclass of matrix decompositions that can be used to solve these problems.

In this paper, we propose to use the  $Q_K R_K$  matrix factorization and its block version to preserve the quality of a desirable portion of the image, being less concerned with the quality of the rest of the image. The implementation of both algorithms in Matlab is described, and stability issues are discussed. Several experiments using different factorizations have been carried out, and the comparative analysis of the quality of the compressed image is presented.

This paper is organized as follows. In Sections 2. and 3., background information is provided. Section 4. describes the implementation of the proposed algorithms. Section 5. presents the results of numerical experiments. Concluding remarks are presented in Section 6.

## 2. $SVD$ and $QR$ factorizations

In this section, we briefly describe  $SVD$  and  $QR$  matrix factorizations and explain how they are used for low rank approximation of the original matrix.

The  $SVD$  of a real matrix  $A^{m \times n}$  with a rank  $r \leq \min(m, n)$  is a decomposition  $A = U \Sigma V^T$ , where  $U^{m \times r}$  and  $V^{n \times r}$  are matrices with orthonormal columns,  $\Sigma^{r \times r}$  is a diagonal matrix. Calculating  $SVD$  requires knowledge of the eigenvalues and eigenvectors of  $AA^T$  and  $A^T A$ . The eigenvectors are the columns of  $U$  and  $V$ , correspondingly. Square roots of the eigenvalues, called singular values, are the diagonal elements in  $\Sigma$ , arranged in descending order.  $SVD$  is proved to be a very efficient algorithm for non-singular matrices [17]. If a matrix  $A$  has a large condition number, then the condition number of  $A^T A$  is squared. Therefore, the calculated  $SVD$  may not be accurate as needed [6].

On the other hand, the  $QR$  decomposition of a matrix provides computational stability [6, 7]. A matrix  $A$  is decomposed into a product of two matrices  $A = QR$ , where  $Q^{m \times n}$  is a matrix with orthogonal columns and  $R^{n \times n}$  is an upper triangular matrix.  $QR$  factorization is computed using the Gram-Schmidt orthogonalization process, Householder transformations, or Givens rotations. Each of these algorithms has its advantages and drawbacks. Householder transformation or Givens rotations are known to be more robust than Gram-Schmidt orthogonalization. However, the Givens rotations are known for their difficulty in implementation, and Householder transformations cannot be effectively parallelized.

Both  $SVD$  and  $QR$  matrix factorizations are used to construct a new approximated matrix of a lower rank to approximate the original image matrix  $A$ . For example, if  $A$  has rank  $r$ , then the matrix of the compressed image with the rank  $s$ ,  $s < r$  can be obtained by neglecting (setting equal to zero) the last  $r - s$  non-zero

singular values in  $\Sigma$  for the *SVD* or the lowest  $r - s$  non-zero rows in  $R$  for the *QR* factorizations. To achieve the desirable error threshold, the analysis should be conducted [3, 5]. If the image quality should be improved, then the rank  $s$  is increased by including more singular values in *SVD* or adding more rows in  $R$  in *QR* factorization.

### 3. $Q_K R_K$ factorization

$Q_K R_K$  factorization of a matrix  $A$  is obtained using a partial orthogonalization process, the base for which is a parametric linear transformation [22]. Both algorithms are described below.

#### 3.1. Notations and definitions.

In this paper, we will be using the following notations and definitions.

Let  $\mathbf{a} = (a_1, a_2, \dots, a_n)$  denote a row vector in  $\mathbb{R}^n$ . Note that bold-faced variables denote vectors, and variables in italic denote vector elements.

Let  $K = \{1, 2, \dots, k\}$ ,  $1 \leq k \leq n$ . The  $K$ th piece of the vector  $\mathbf{a}$ , or a subvector, is denoted by  $\mathbf{a}(K) = (a_1, a_2, \dots, a_k)$ .

A system of  $m$  row vectors is represented as  $\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_m\}$ , where  $\mathbf{a}_i = (a_{i1}, a_{i2}, \dots, a_{in})$ ,  $i = 1, \dots, m$ . The corresponding  $K$ th pieces of these vectors are denoted by  $\mathbf{a}_i(K) = (a_{i1}, a_{i2}, \dots, a_{ik})$ .

Vectors  $\mathbf{a} = (a_1, a_2, \dots, a_n)$  and  $\mathbf{b} = (b_1, b_2, \dots, b_n)$  are called *partially orthogonal* if their  $K$ th pieces are orthogonal, that is, the dot product  $\mathbf{a}(K)\mathbf{b}(K) = 0$ . Vectors  $\mathbf{a}$  and  $\mathbf{b}$  are called *partially orthonormal*, if they are partially orthogonal and their norms  $\|\mathbf{a}(K)\| = \|\mathbf{b}(K)\| = 1$ .

The partial orthogonalization method [21] is a particular type of oblique projection that enables us to obtain a partially orthogonal/orthonormal system of vectors from a given system of vectors. The foundation for partial orthogonalization is a parametric linear transformation, which is a generalization of the Gaussian and Gram-Schmidt transformation [21]. The idea behind the parametric transformation is to choose  $k$  elements simultaneously (called the pivot vector of length  $k$ ) instead of the pivot element in the Gaussian transformation.

#### 3.2. Parametric linear transformation.

Let a system of linearly independent row vectors  $\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_m\}$  be given, where  $\mathbf{a}_i = (a_{i1}, a_{i2}, \dots, a_{in})$ ,  $i = 1, \dots, m$ . We require  $m \leq k \leq n$ .

Without loss of generality, we assume that the Euclidean norm  $\|\mathbf{a}_1(K)\| \neq 0$ . Otherwise, we can switch the rows to satisfy this condition.

To obtain a new system of vectors  $\{\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_m\}$ , set

$$(3.1) \quad \mathbf{b}_i = \begin{cases} \frac{\mathbf{a}_i}{\|\mathbf{a}_1(K)\|}, & i = 1 \\ \mathbf{a}_i + \alpha_i \mathbf{b}_1, & \text{otherwise,} \end{cases}$$

where  $\alpha_i = -\mathbf{a}_i(K)\mathbf{b}_1(K)$ ,  $i \neq 1$ .

Note that  $\mathbf{b}_1(K)\mathbf{b}_1(K) = 1$  for all  $i = 1, 2, \dots, m$ , because

$$\mathbf{b}_1(K)\mathbf{b}_1(K) = \frac{\mathbf{a}_1(K)}{\|\mathbf{a}_1(K)\|} \frac{\mathbf{a}_1(K)}{\|\mathbf{a}_1(K)\|} = 1,$$

and for  $i = 2, \dots, m$ ,

$$\begin{aligned} \mathbf{b}_1(K)\mathbf{b}_i(K) &= \mathbf{b}_1(K)(\mathbf{a}_i(K) + \alpha_i \mathbf{b}_1(K)) \\ &= \mathbf{b}_1(K)\mathbf{a}_i(K) - \mathbf{b}_1(K)\mathbf{a}_i(K)\mathbf{b}_1(K)\mathbf{b}_1(K) \\ &= \mathbf{b}_1(K)\mathbf{a}_i(K) - \mathbf{b}_1(K)\mathbf{a}_i(K)\|\mathbf{b}_1(K)\mathbf{b}_1(K)\| \\ &= \mathbf{b}_1(K)\mathbf{a}_i(K) - \mathbf{b}_1(K)\mathbf{a}_i(K) = 1. \end{aligned}$$

Thus, the parametric linear transformation (3.1) allows us to obtain a new system of vectors, where the first row and any other row are partially orthonormal.

### 3.3. Partial orthogonalization.

To reduce a given linearly-independent system of vectors to a partially orthogonal/orthonormal one, parametric linear transformation (3.1) is used at each step of the process, as described below.

Let a system of linearly independent row vectors  $\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_m\}$  be given. A partially orthonormal system  $\{\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_m\}$  is constructed as follows.

Let

$$(3.2) \quad \mathbf{b}_1 = \frac{\mathbf{a}_1}{\|\mathbf{a}_1(K)\|}.$$

Successively, for  $s = 2, \dots, m$ , set

$$(3.3) \quad \mathbf{b}_s = \frac{\mathbf{b}_s}{\|\mathbf{b}_s(K)\|},$$

where

$$\begin{aligned} \mathbf{b}_s &= \mathbf{a}_s + \alpha_{s1}\mathbf{b}_1 + \dots + \alpha_{s,s-1}\mathbf{b}_{s-1}, \\ \alpha_{si} &= -\mathbf{a}_i(K)\mathbf{b}_1(K), i = 1, \dots, s-1. \end{aligned}$$

The resulting system of vectors  $\{\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_m\}$  is partially orthonormal, that is,

$$\mathbf{b}_i(K)\mathbf{b}_s(K) = \begin{cases} 1, & i = s \\ 0, & \text{otherwise.} \end{cases}$$

Thus, the partial orthogonalization process (3.2)-(3.3) depends on the choice of the parameter  $k$ . In two extreme cases, when  $k = 1$  and  $k = n$ , it is the Gaussian transformation and Gram-Schmidt orthogonalization, respectively.

### 3.4. Stability issues

Stability issues of both Gaussian elimination and Gram-Schmidt orthogonalization have been discussed extensively. It is well known that both processes are unstable. However, more accurate results can be obtained if the computations are organized properly. Gaussian elimination is stable when the partial pivoting rule is used. Namely, the rows are interchanged at each elimination step so that the pivot element has the largest absolute value. Gram-Schmidt orthogonalization provides stable computations when the modified version is used, when at each step, all the rows below the pivot row are transformed [6, 7, 9, 24].

To ensure the stability and accuracy of the results of our experiments, the modified partial orthogonalization is used. The pivoting rule for this process combines both strategies described above. At each  $s$ th step, the row whose  $K$ th piece has the maximum norm is a pivot row. The transformations are performed on all the rows below the pivot one.

### 3.5. $Q_K R_K$ factorization.

Let  $A^{m \times n}$ ,  $m \leq n$ , be a full row rank matrix.

Applying the partial orthogonalization process (3.2)-(3.3) to the rows of a matrix  $A$ , an  $R_K Q_K$  factorization is obtained. Namely,  $A = R_K Q_K$ , where  $R_K^{m \times m}$  is a nonsingular lower triangular matrix constructed from the coefficients  $\alpha$ , and  $Q_K^{m \times n}$  is a matrix with partially orthonormal rows. A matrix with partially orthonormal rows is called a partially orthonormal matrix.

Visually, the  $R_K Q_K$  decomposition can be represented as follows

$$A = \underbrace{\begin{bmatrix} 1 & 0 & \dots & 0 \\ -\alpha_{21} & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -\alpha_{m1} & -\alpha_{m2} & \dots & 1 \end{bmatrix}}_{R_K} \times \underbrace{\left[ \begin{array}{c|c} Q_1 & Q_2 \\ \hline \hline \hline \end{array} \right]}_{Q_K}$$

where  $Q_1$  is an  $m \times k$  submatrix with orthogonal/orthonormal rows.

Therefore,  $A^T = Q_K^T R_K^T = [Q_1^T \quad Q_2^T] R_K^T$  is in the following form where  $Q_1^T$  is an  $k \times m$  matrix with orthogonal columns and  $R_K^T$  is an  $m \times m$  nonsingular upper triangular matrix. This decomposition is called a partially orthogonal decomposition.

$$A^T = \begin{matrix} Q_1 \\ \vdots \\ Q_2 \end{matrix} \left[ \begin{array}{c|c} \vdots & \\ \hline \vdots & \end{array} \right] \times \begin{matrix} R_K^T \\ \vdots \\ R_K^T \end{matrix} \begin{bmatrix} 1 & -\alpha_{21} & \dots & -\alpha_{m1} \\ 0 & 1 & \dots & -\alpha_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix},$$

Let us emphasize that in the case  $k = n$ , the modified partial orthogonalization coincides with the modified Gram-Schmidt orthogonalization. Therefore, we obtain the standard  $QR$  factorization of  $A$ .

**3.6. Block  $Q_K R_K$  factorization.**

The block  $Q_K R_K$  factorization of  $A$  allows us to receive a class of decompositions in the form  $A = R_K Q_K$ , where  $R_K$  is a lower triangular matrix, and  $Q_K$  is in the following form

	$K_1$	$K_2$	$\dots$	$K_\tau$	$N_\tau$
$M_1$	$Q_1$		$\dots$		
$M_2$		$Q_2$	$\dots$		
$\vdots$	$0$	$0$	$\dots$		
$\vdots$			$\dots$		
$M_\tau$			$\dots$	$Q_\tau$	
	$Q_K$				

with

$$K_s \subseteq N, s = 1, \dots, \tau, \tau \leq m, \quad K_i \cap K_j = \emptyset, i \neq j \text{ for all } i, j, \quad M = \bigcup_{s=1}^{\tau} M_s,$$

the diagonal submatrices  $Q_s$  are matrices with orthogonal rows,  $0$  is a zero submatrix. Note that depending on the selection of the sets  $K_s$ , the set  $N_\tau$  can be empty. If

$m \geq n$ , then the operations are performed on the transpose of  $A$ .

#### 4. Implementation of $Q_K R_K$ decomposition

The pseudo-code below describes the implementation of the  $Q_K R_K$  algorithm. The algorithm starts by copying the input matrix  $A$  to a matrix  $V$ . While iterating through the rows of  $V$ , we keep track of a *pivotRow*, indicating the pivot row number in the current iteration. If the  $K$ th subvector in the *pivotRow* is a zero vector, then we swap this row with the closest row with a non-zero  $K$ th piece and update the permutation matrix according to the swap. Otherwise, we perform the row reduction on all rows below the *pivotRow*. When all vectors below the pivot row are zero vectors, we move on to the next stage of the process. If we orthonormalize  $V$ , the *alpha* coefficient matrix should be updated accordingly.

---

##### Algorithm 1 $Q_K R_K$ factorization

---

```

function  $Q_K R_K(A, K)$ :
  initialize empty alpha and permutation matrices  $V = A$ 
  for all rows:
    if current  $K$ -entries of current row are 0:
      swap with first nonzero row
      update permutation matrix accordingly
    if all rows are zero rows, move on to next vector in  $K$ 
    perform row reduction on rows below our current pivot row as follows:
    for targetRow in range(pivotRow+1, lastRow):
       $V(i, K)$  represents the columns in  $K$  of the  $i$ -th row in  $V$ 
       $V(i, :)$  represents the  $i$ -th row of  $V$  using all column entries.
       $\alpha(\text{targetRow}, \text{pivotRow}) = \frac{V(\text{pivotRow}, K) * V(\text{targetRow}, K)}{V(\text{pivotRow}, K) * V(\text{pivotRow}, K)}$ 
      edit  $V$  using  $\alpha$ :
       $V(\text{targetRow}, :) = V(\text{targetRow}, :) - \alpha(\text{targetRow}, \text{pivotRow}) * V(\text{pivotRow}, :)$ 

  the below step is optional, depending on if  $V$  is to be orthonormalized
  normalize corresponding  $k$ -entries in  $V$  and edit alpha matrix accordingly
  calculate error =  $\text{norm}(A - \alpha * \text{permutation} * V, 2)$ 
  return  $V, \alpha$  and permutation matrices

```

---

#### 5. Experiments and results

For our experiments, we used a picture of the puppy Anoushig in a jpeg format, see Figure 5.1a). First, the image is translated into black and white, with pixel values ranging from 0 to 255. To capture the details of the image, such as the puppy's eye and crate, we select a portion of the image to run tests against Figure 5.1b). The original image is reduced from  $2016 \times 1512$  to a  $500 \times 900$  full rank matrix.

All experiments have been performed using MATLAB 2020b software. Experiments involving the  $Q_K R_K$  factorization and its block version were performed





FIG. 5.1: Original image that was used for the experiments.

using the code outline described above. Experiments involving the  $QR$  and  $SVD$  factorizations were conducted using MATLAB built-in functions. The command  $[Q, R, P] = qr(A)$  calculates  $QR$  factorization of  $A$ , such that  $A = QRP^{-1}$ , where  $Q$  is an orthogonal matrix,  $R$  is an upper triangular matrix, and  $P$  is a permutation matrix. The command  $[U, E, V] = svd(A)$  calculates  $SVD$  of  $A$ , such that  $A = UEV$ , where  $U$  and  $V$  are orthogonal matrices,  $E$  is a diagonal matrix.

### 5.1. Experiment 1

We examined the performance of  $Q_K R_K$  factorization and its block version for low rank approximation of an image on Figure 5.1b). The approximation of the original 500 rank matrix is made by reducing it to the  $r$  rank approximated matrix, where  $r = 10, 20, 100, 200, 500$ . The comparison analysis has been performed using relative error, or  $L_2$ -norm, and the structural similarity methods.

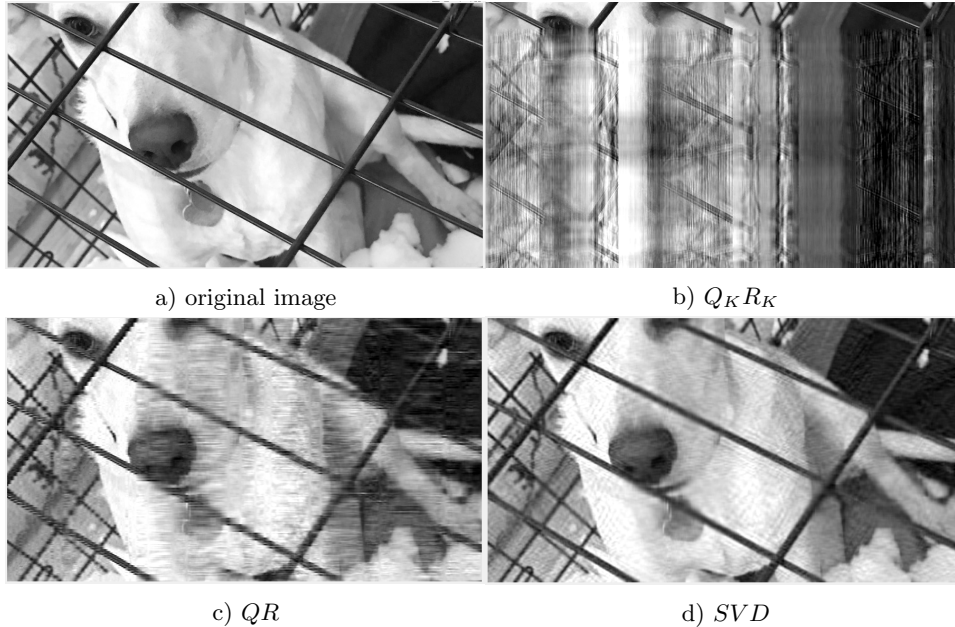
Relative error results for all four algorithms are presented in Table 5.1. The  $Q_K R_K$  algorithm was performed with  $K = \{1, \dots, 500\}$ , and for the block  $Q_K R_K$  decomposition, the partition is  $K = \{K_1, K_2, K_3\}$ , where  $K_1 = \{1, \dots, 200\}$ ,  $K_2 = \{201, \dots, 350\}$ ,  $K_3 = \{351, \dots, 500\}$ .

Our numerical results confirm that, as expected, low rank approximation using  $SVD$  produces the best results with the minimum error for all ranks, followed by the performance of  $QR$  factorization. As the rank of the approximated matrix increases, the error for all algorithms, except for the  $Q_K R_K$  blockwise approach, decreases.

Table 5.1: Relative errors.  $QR$ ,  $SVD$ ,  $Q_K R_K$  and block  $Q_K R_K$  factorizations

Rank	$QR$	$SVD$	$Q_K R_K$	$Q_K R_K$ block
10	0.1	0.06	0.43	0.53
20	0.067	0.035	0.39	0.53
50	0.036	0.013	0.31	0.4
100	0.019	$6.9 * 10^{-3}$	0.31	0.35
200	$6.6 * 10^{-3}$	$2.8 * 10^{-3}$	0.30	0.9
350	$2.0 * 10^{-3}$	$1.0 * 10^{-3}$	0.115	1.066
500	$2.1 * 10^{-16}$	$8.2 * 10^{-16}$	$9.6 * 10^{-17}$	$1.2 * 10^{-16}$

Images of rank 50 reconstructions using the  $QR$ ,  $SVD$ ,  $Q_K R_K$  algorithm are presented in Figure 5.2b)-d). The original image is presented in Figure 5.2a) for visual comparison purposes. From the images, we can see that for the rank  $r = 50$ , the overall image appearance is better when using  $SVD$  or  $QR$ , but the details, such as the eye or crate wire, appear sharper when using the  $Q_K R_K$  while the rest of the image being blurry.

FIG. 5.2: Reconstructed images with  $r = 50$ .

Images on the Figure 5.3 show a comparison between  $Q_K R_K$  with  $K = \{1, \dots, 500\}$  and block  $Q_K R_K$  with  $K = \{K_1, K_2, K_3\}$ , where  $K_1 = \{1, \dots, 200\}$ ,  $K_2 = \{201, \dots, 350\}$ ,  $K_3 = \{351, \dots, 500\}$ .

Table 5.2: Relative errors calculated for a fragment of an image,  $l = 10$ 

Rank	$Q_K R_K$	QR	SVD
10	$1.23 * 10^{-18}$	$2 * 10^{-3}$	$5.9 * 10^{-3}$
50	$1.23 * 10^{-18}$	$7 * 10^{-4}$	$9.3 * 10^{-4}$
100	$1.23 * 10^{-18}$	$2.6 * 10^{-4}$	$5.8 * 10^{-4}$
200	$1.23 * 10^{-18}$	$2.5 * 10^{-4}$	$2.2 * 10^{-4}$
500	$1.23 * 10^{-18}$	$1.4 * 10^{-17}$	$4.1 * 10^{-17}$

The image is reconstructed using 50, 100, 200, and 350 ranks.

From these examples, it is visually evident that after reconstructing the image, the lower right side of the images by  $Q_K R_K$  suffers the most. Similar results are obtained when using its block version implementation. This distortion is caused by the fact that  $Q_K R_K$  is checking the first 500 columns and effectively neglecting the rest. As such, this region's numerical errors are significantly heightened compared to the more conventional  $QR$  and  $SVD$  algorithms. With the block approach, since the lower right corner of the image is reconstructed poorly, the relative error increases beyond 100%. The block lies in the right portion of the bottom  $m - r$  rows. With increasing the  $r$  value, this block emerges, and the remainder of the image is reconstructed reasonably well.

## 5.2. Experiment 2

For this experiment, we test the accuracy of the  $Q_K R_K$  factorization with  $K = \{1, \dots, 500\}$ , and compare it with the  $QR$  and  $SVD$  decompositions on a selected portion of the original image. We reconstruct the entire image but only evaluate its accuracy on an  $l \times l$  submatrix in the upper left corner.

The images of rank 50 reconstructions of the  $100 \times 100$  submatrix using the specified algorithms are in Figures 5.4b)-d) and 5.5. The original image, Figure 5.4 a) is included for visual comparison purposes.

As we can see from Figures 5.4 and 5.5, both  $Q_K R_K$  and its block variant create a perfectly reconstructed upper portion of the sub-image, and a poorly reconstructed bottom part. It happens because the current implementation of the  $Q_K R_K$  algorithm does not utilize permutation to reconstruct the most critical rows first. On the other hand,  $SVD$  and  $QR$  do, leading to although poor, but more consistent quality in the image.

Tables 5.2 and 5.3 present the results of the experiments for  $l = 10$  and  $l = 100$  with the  $r = 10, 50, 100, 200, 500$  rank approximations.

Our experiments demonstrate that the accuracy of the  $l \times l$  portion of the reconstructed image using  $Q_K R_K$  with  $K = \{1, 2, \dots, k\}$  is almost perfect at  $l = k$ . It is because the partial orthogonalization process produces orthogonal  $K$ th  $l \times l$  subvectors. On the contrary,  $QR$  and  $SVD$  work on the entire matrix, resulting in

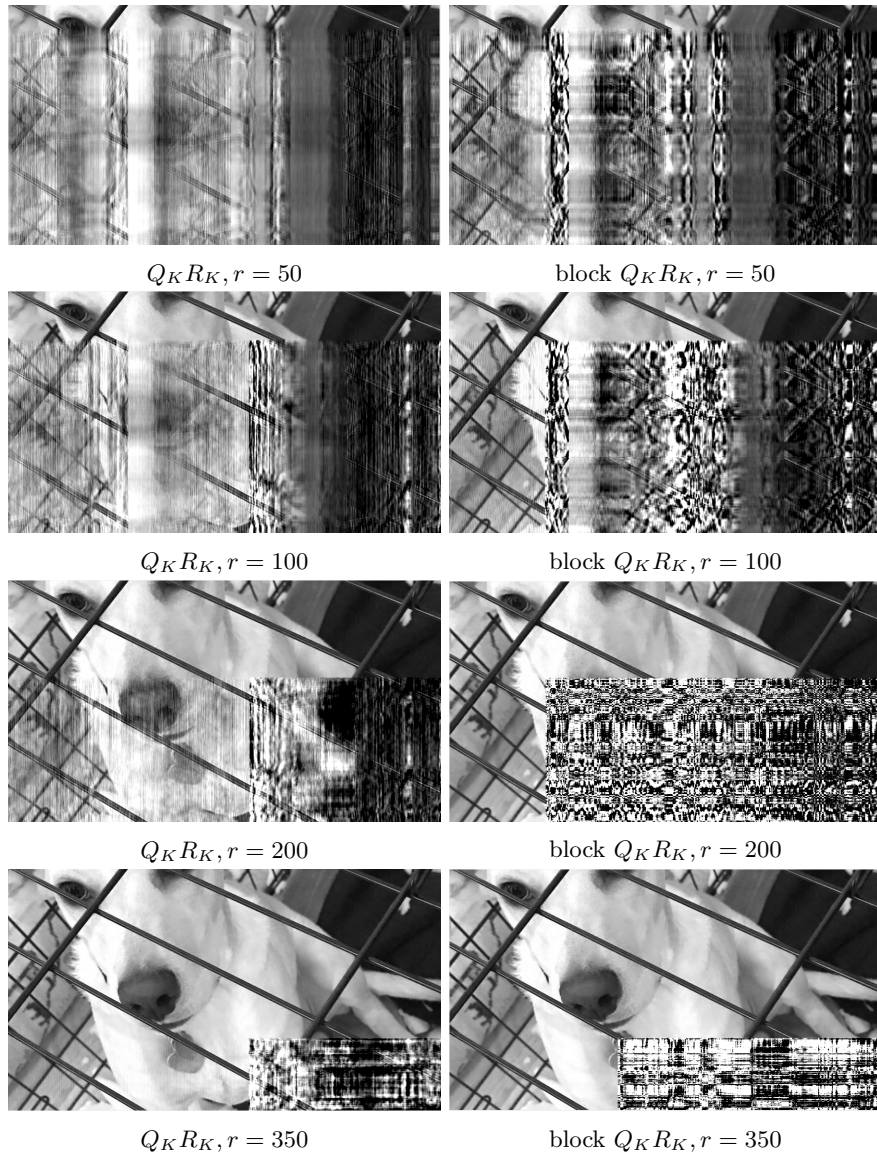


FIG. 5.3: Reconstructed images using  $Q_K R_K$  and block  $Q_K R_K$ .

an approximated image with a better overall visual appearance.

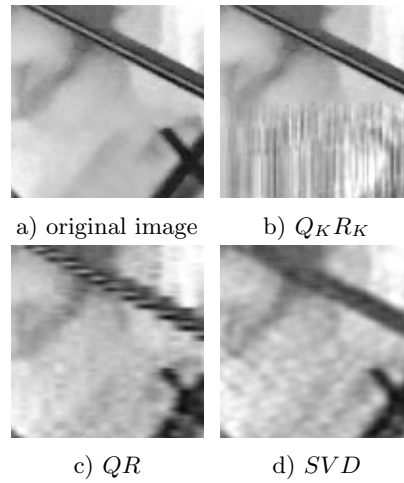
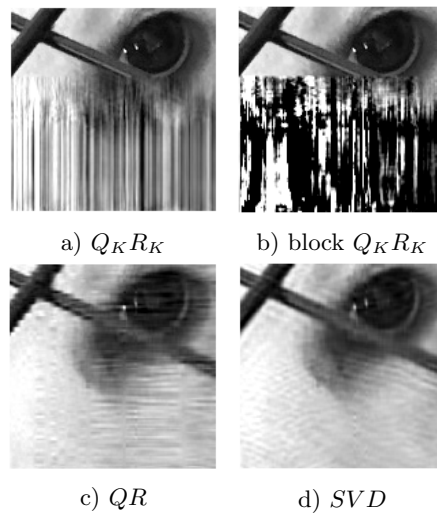
FIG. 5.4: Reconstructed subimages,  $l = 100, r = 50$ .

FIG. 5.5: Reconstructed subimages.

### 5.3. Experiment 3. Structural similarity, $Q_K R_K$ vs $QR$ and $SVD$

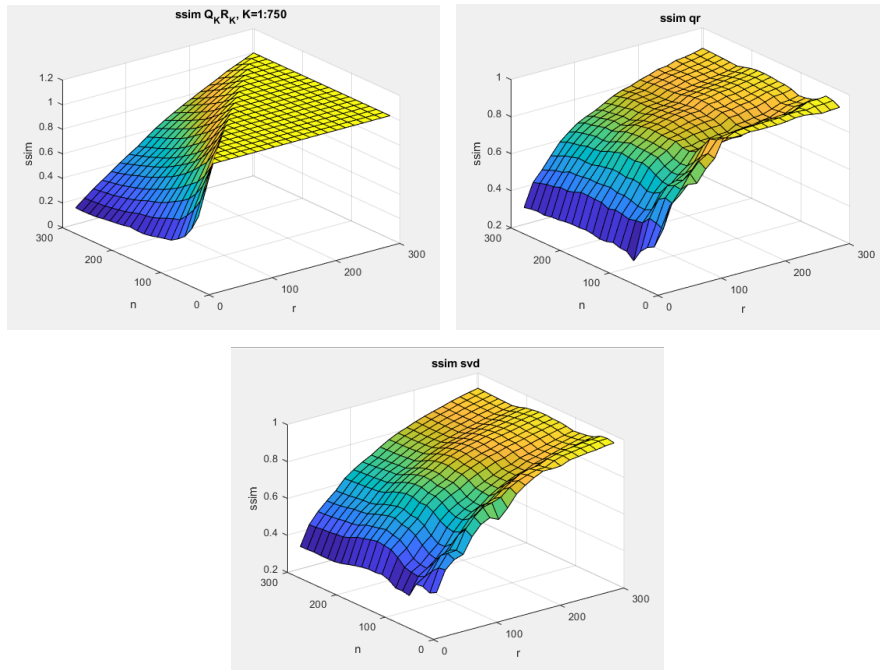
In this experiment, we compare the performance of  $Q_K R_K$ ,  $QR$ , and  $SVD$  using the structural similarity method [23]. For this purpose, we use the MATLAB *ssim* built-in function.

The graphics in Figure 5.6 demonstrate the structural similarity of reconstructions by  $Q_K R_K$ ,  $QR$ , and  $SVD$  up to the rank  $r = 300$ . The structural similarity is calculated for  $n \times n$  square sub-matrices, where  $n$  varies from 20 to 300. Interestingly,

Table 5.3: Relative errors calculated for a fragment of an image,  $l = 100$ 

Rank	$Q_K R_K$	QR	SVD
10	0.071	0.03	0.019
50	0.032	$6.9 * 10^{-3}$	$4.3 * 10^{-3}$
100	$1.27 * 10^{-17}$	$4.1 * 10^{-3}$	$2.6 * 10^{-3}$
200	$1.27 * 10^{-17}$	$1.8 * 10^{-3}$	$9.5 * 10^{-4}$
500	$1.27 * 10^{-17}$	$5.4 * 10^{-17}$	$1.6 * 10^{-16}$

both  $QR$  and  $SVD$  have a concave-down tendency to produce better results already at lower rank approximations. On the contrary,  $Q_K R_K$  shows a concave-up tendency for the lower rank reductions but outperforms both  $QR$  and  $SVD$  as the approximation rank  $r$  increases.

FIG. 5.6: Structural similarity,  $Q_K R_K$ ,  $QR$ ,  $SVD$ .

## 6. Conclusion

In this paper, we propose to use a parametric  $Q_K R_K$  matrix factorization and its block version for low rank approximation in image compression. With our experiments, we have demonstrated that when reducing the original image to rank  $r$ ,

the  $Q_K R_K$  factorization effectively reconstructs the  $K$ th portion of the image, while the rest of the image is reconstructed with less accuracy. In addition,  $Q_K R_K$  produces better approximations as the matrix size increases.  $QR$  attempts to reconstruct the entire image, while  $Q_K R_K$  reconstructs the first  $r$  ranks almost perfectly. Our numerical results suggest that  $Q_K R_K$  can be effectively utilized in applications, such as face recognition and forensics, where the interest is to reconstruct the overall picture while placing emphasis on a certain portion of the image. The parametric nature of  $Q_K R_K$  allows us to focus on any part of the image. Our experiments have also confirmed theoretical results on the computational efficiency of  $Q_K R_K$  and the block  $Q_K R_K$  compared with  $QR$  and  $SVD$  [6, 22]. Furthermore, the block  $Q_K R_K$  algorithm is proven to be computationally less expensive than  $Q_K R_K$ .

The proposed application of  $Q_K R_K$  factorization opens future work and possibilities. Our next focus is to create strategies and algorithms for the optimal choice of the partition  $K$  in the  $Q_K R_K$  and block  $Q_K R_K$  factorizations and investigate them in various image processing applications.

## REFERENCES

1. K. E. ASNAOUI: *Image compression based on block SVD power method*. Journal of Intelligent Systems, **29**(1) (2020), 1345–1359.
2. S. CHOUNTASIS, D. PAPPAS and V. KATSIKIS: *Image restoration via fast computing of the Moore-Penrose inverse matrix*. 16th International Conference on Systems, Signals and Image Processing, Chalkida, Greece (2009). doi:10.1109/IWSSIP.2009.5367731
3. M. T. CHU, R. E. FUNDERLIC and R. J. PLEMMONS: *Structured low rank approximation*. Linear Algebra and Its Applications, **366** (2003), 157–172.
4. H. DADKHAHI, A. GOTCHEV and K. EGIAZARIAN: *Inverse polynomial reconstruction method in DCT domain*. EURASIP Journal on Advances in Signal Processing, **2012**(133) (2012).
5. C. ECKART and G. YOUNG: *The approximation of one matrix by another of lower rank*. Psychometrika, **1** (1936), 211–218.
6. G. E. FORSYTHE and C. B. MOLER: *Computer Solution of Linear Algebraic Systems*. Prentice Hall, Englewood Cliffs, New Jersey, (1967).
7. G. H. GOLUB and C. F. VAN LOAN: *Matrix Computations*. The Johns Hopkins University Press, Baltimore, Maryland, 4th edition, (2013).
8. N. HALKO, P. G. MARTINSSON and J. A. TROPP: *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*. SIAM Review, **53**(2) (2011), 217–288.
9. N. J. HIGHAM: *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, PA, (1996).
10. A. J. HUSSAIN, A. AL-FAYADH and N. RADI: *Image compression techniques: A survey in lossless and lossy algorithms*. Neurocomputing, **300** (2018), 44–69.
11. W. JANG and S-Y. LEE: *Partial image encryption using format-preserving encryption in image processing systems for Internet of things environment*. International Journal on Distributed Sensor Networks, **16**(3) (2020).

12. S. KRIVENKO, V. LUKIN, O. KRYLOVA and K. EGIAZARIAN: *A fast method of visually lossless compression of dental images*. Applied sciences, **11**(1) (2021).
13. R. KUMAR, U. PATBHAJE and A. KUMAR: *An efficient technique for image compression and quality retrieval using matrix completion*. Computer and Information Sciences, **34**(4) (2022), 1231–1239.
14. K. LI and G. WU: *A randomized generalized low rank approximations of matrices algorithm for high dimensionality reduction and image compression*. Numerical Linear Algebra with Applications, **28**(1) (2021).
15. P. G. MARTINSSON and S. VORONIN: *A randomized blocked algorithm for efficiently computing rank-revealing factorizations of matrices*. SIAM Journal on Scientific Computing, **38**(5) (2016), S485–S507.
16. W. S. NG and W. W. TAN: *Some properties of various types of matrix factorization*. The 16th International Conference on Mathematics, Statistics, and Their Applications, ITM Web Conf., **36** (2021). doi:10.1051/itmconf/20213603003
17. A. PANDEY and S. SHROTRIYA: *Comparing the effect of matrix factorization techniques in reducing the time complexity for traversing the big data of recommendation systems*. International Journal of Computer and Communication Engineering, **2**(2) (2013), 170–173.
18. M. A. RAHMAN, M. HAMADA and J. SHIN: *The impact of state-of-the-art techniques for lossless still image compression*. Electronics, **3**(10(360)) (2021), 1–40.
19. A. ROWAYDA: *SVD based image processing applications: state of the art, contributions and research challenges*. International Journal of Advanced Computer Science and Applications, **7**(3) (2012), 26–34.
20. Q. SU, Y. NIU, G. WANG, S. JIA and J. YUE: *Color image blind watermarking scheme based on QR decomposition*. Signal Processing, **94** (2014), 219–235.
21. A. D. TUNIEV: *Pivot vector method and its applications*. Cybernetics and Systems Analysis, **28**(1) (1992), 99–109.
22. K. TUNYAN, K. EGIAZARIAN, A. TUNIEV and J. ASTOLA: *An efficient approach to the linear least squares problem*. SIAM J. Matrix Anal. Appl., **26**(2) (2005), 583–598.
23. Z. WANG, A. C. BOVIK, H. R. SHEIKH and E. P. SIMONCELLI: *Image quality assessment: from error visibility to structural similarity*. IEEE Transactions on Image Processing, **13**(4) (2004), 600–612.
24. J. H. WILKINSON: *Rounding Errors in Algebraic Processes*. Prentice Hall, Englewood Cliffs, New Jersey, (1963).
25. J. XIAO, M. GU and J. LANGOU: *Fast parallel randomized QR with column pivoting algorithms for reliable low-rank matrix approximations*. IEEE 24th International Conference on High Performance Computing (HiPC), (2017), 233–242. doi:10.1109/HiPC.2017.00035