IE531: Algorithms for Data Analytics
Spring, 2020
Mid-Term Programming Assignment: Low-Rank Approximations for Color Images using Image-Blocks
Due Date: March 27, 2020

1 Introduction

We looked at Python code that found low rank approximations of color images using SVD and QR-Factorization. In all these procedures the matrices that represented the RGB-colors of a color image were replaced by lower-rank approximations, which were combined to form a lower-rank approximation of the original color image.

In problem 5 of Homework 3 you showed that if an $n \times d$ data matrix $\mathbf{A}$ can be partitioned as

$$\underbrace{\mathbf{A}}_{n\times d} = \begin{pmatrix} \underbrace{\mathbf{A}_1}_{n_1\times d_1} & \underbrace{\mathbf{A}_2}_{n_1\times d_2} \\ \underbrace{\mathbf{A}_3}_{n_2\times d_1} & \underbrace{\mathbf{A}_4}_{n_2\times d_2} \end{pmatrix},$$

and we replaced the sub-matrices $\{\mathbf{A}_i\}_{i=1}^4$ with their lower-rank approximations $\{\mathbf{B}_i\}_{i=1}^4$ where $\|\mathbf{A}_i - \mathbf{B}_i\|_f \leq \epsilon$ for $i \in \{1,2,3,4\}$, then the matrix

$$\mathbf{B} = \begin{pmatrix} \mathbf{B}_1 & \mathbf{B}_2 \\ \mathbf{B}_3 & \mathbf{B}_4 \end{pmatrix}$$

satisfies the requirement $\|\mathbf{A} - \mathbf{B}\|_F \leq 4\epsilon$.

For this programming assignment I want you to to take an original color image and divide its RGB-color matrices into smaller blocks – like $\mathbf{A}$ was split into 4 sub-matrices $\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3$, and $\mathbf{A}_4$ above. You can split the image-matrices into 4 sub-matrices and find the low-rank approximations of each sub-matrix – which is then used to arrive at the low-rank approximation of the original image. For this programming assignment divide the three RGB-color matrices of the original $n \times d$ color image into: (a) (2×2) sub-matrices, (b) (3×3) sub-matrices and (c) 16 sub-matrices. For each of these cases, I want you to find the low-rank approximation of the sub-matrices, which are then combined to form the low-rank approximation of the original color image.

If you are running your code on command-line (as opposed using Jupyter Notebooks), I am looking at an output along the lines of what is shown in figure 1.

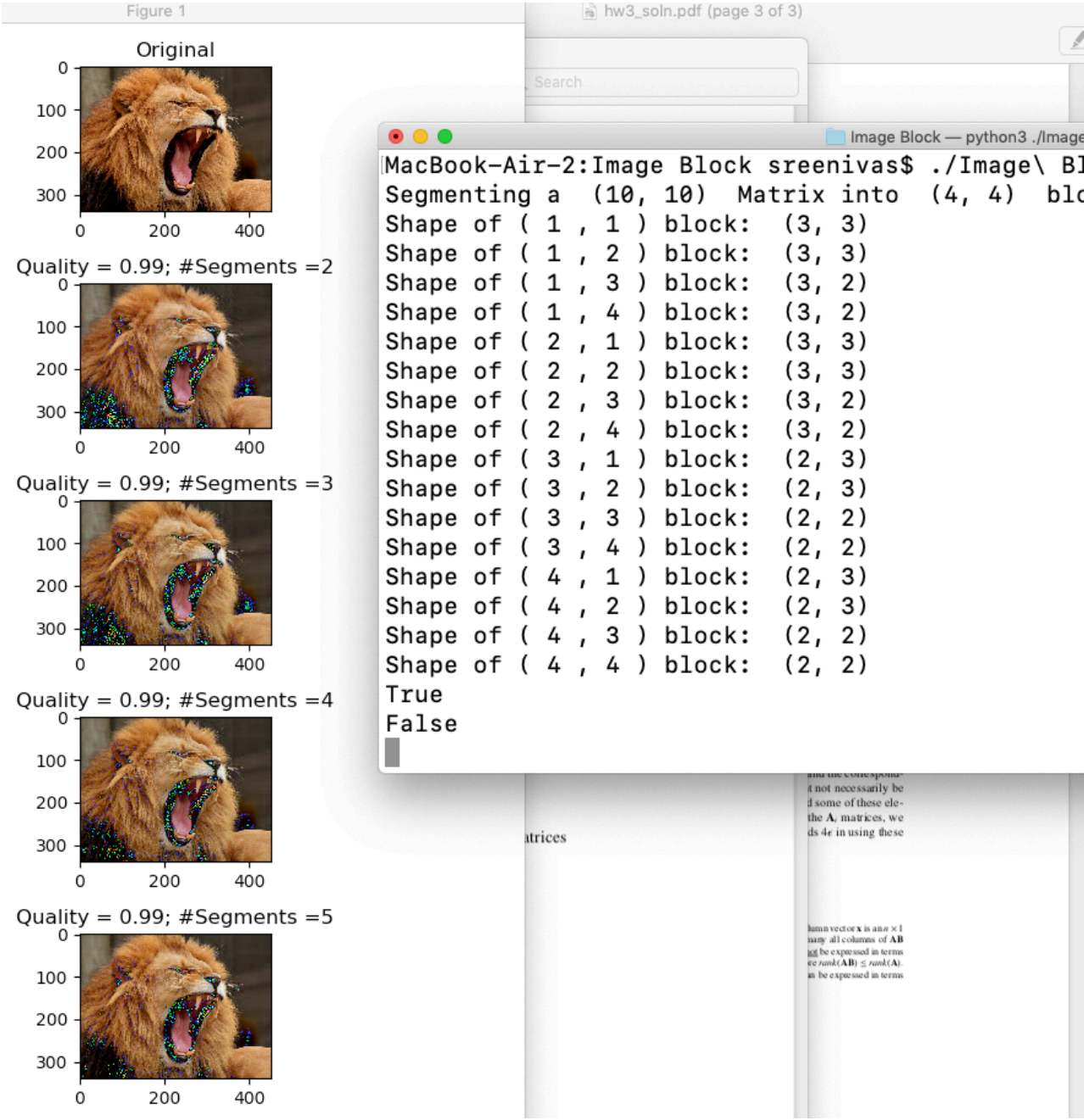

Figure 1: Sample output.