Matrix Decompositions: QR & SVD Applied to Image Processing With Python

By John Stulich

Research Supervisor:
Dr. Debananda Chakraborty

New Jersey City University, 2019

Matrix Decompositions

A = QR **R = Q^{T}A** **A = QR**

Transpose: interchange the rows and columns of a matrix or array

A_Transpose = A.T or B = np.transpose(A)

A = Image Matrix

Q = Orthogonal Matrix --> **Q^{T}Q** = I (Identity Matrix)

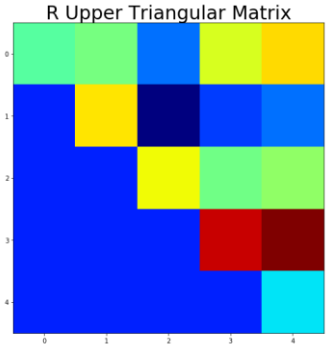
R = Upper Triangular Matrix

R = np.triu(A) or R1 = np.triu(A,0) # 0 is the index

D = Diagonal Matrix

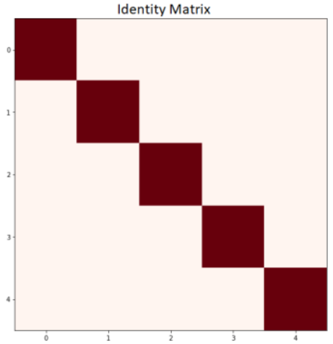
D = np.diag(A)

$$R = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ \mathbf{0} & a_{22} & a_{23} & \cdots & a_{2n} \\ \mathbf{0} & \mathbf{0} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & a_{nn} \end{bmatrix}$$



Upper Triangular Matrix

$$I = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$



Identity Matrix

Gram–Schmidt General Formula

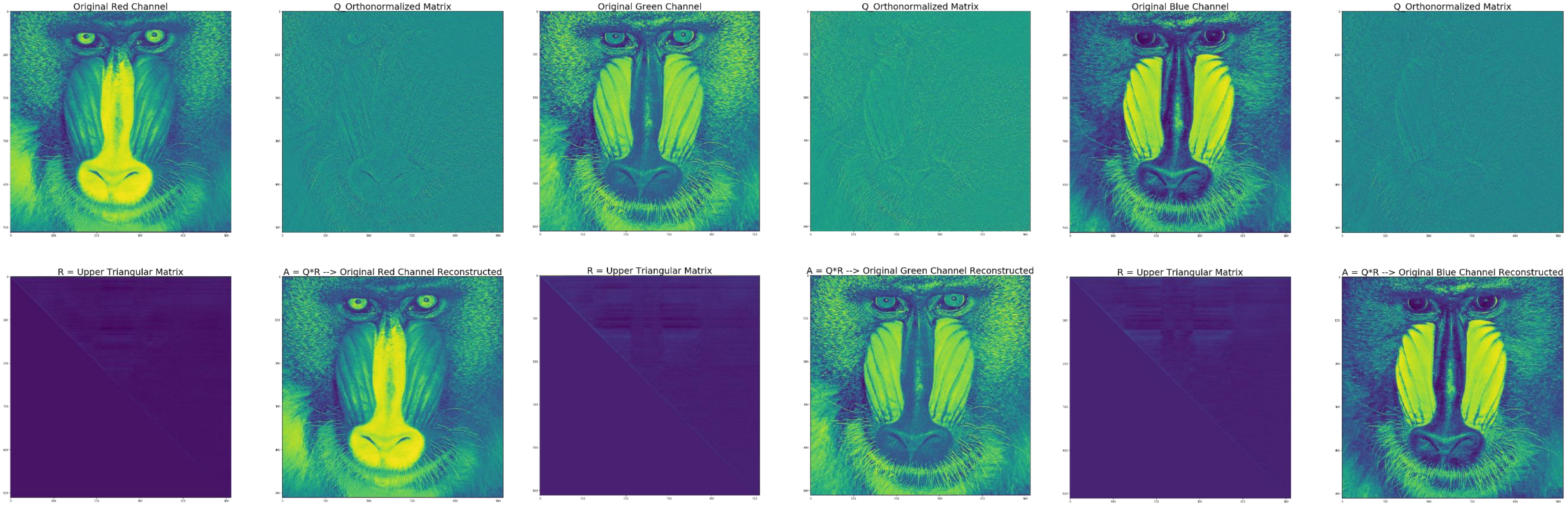
$$\mathbf{u}_{\mathrm{p}} = \mathbf{v}_{\mathrm{p}} - \sum_{i=1}^{p-1} \left(\frac{\mathbf{u}_i \cdot \mathbf{v}_p}{\| \mathbf{u}_i \|^2} \right) \mathbf{u}_i \text{ for p} = 2, 3, \ldots, \mathrm{n}$$

- Converts non-orthgonal matrix into an orthogonal matrix
- “Distills” non-”cost-efficient” matrix into a “cost-efficient” matrix
    - “Cost-efficient” = computationally efficient

Gram–Schmidt Process – Python Output

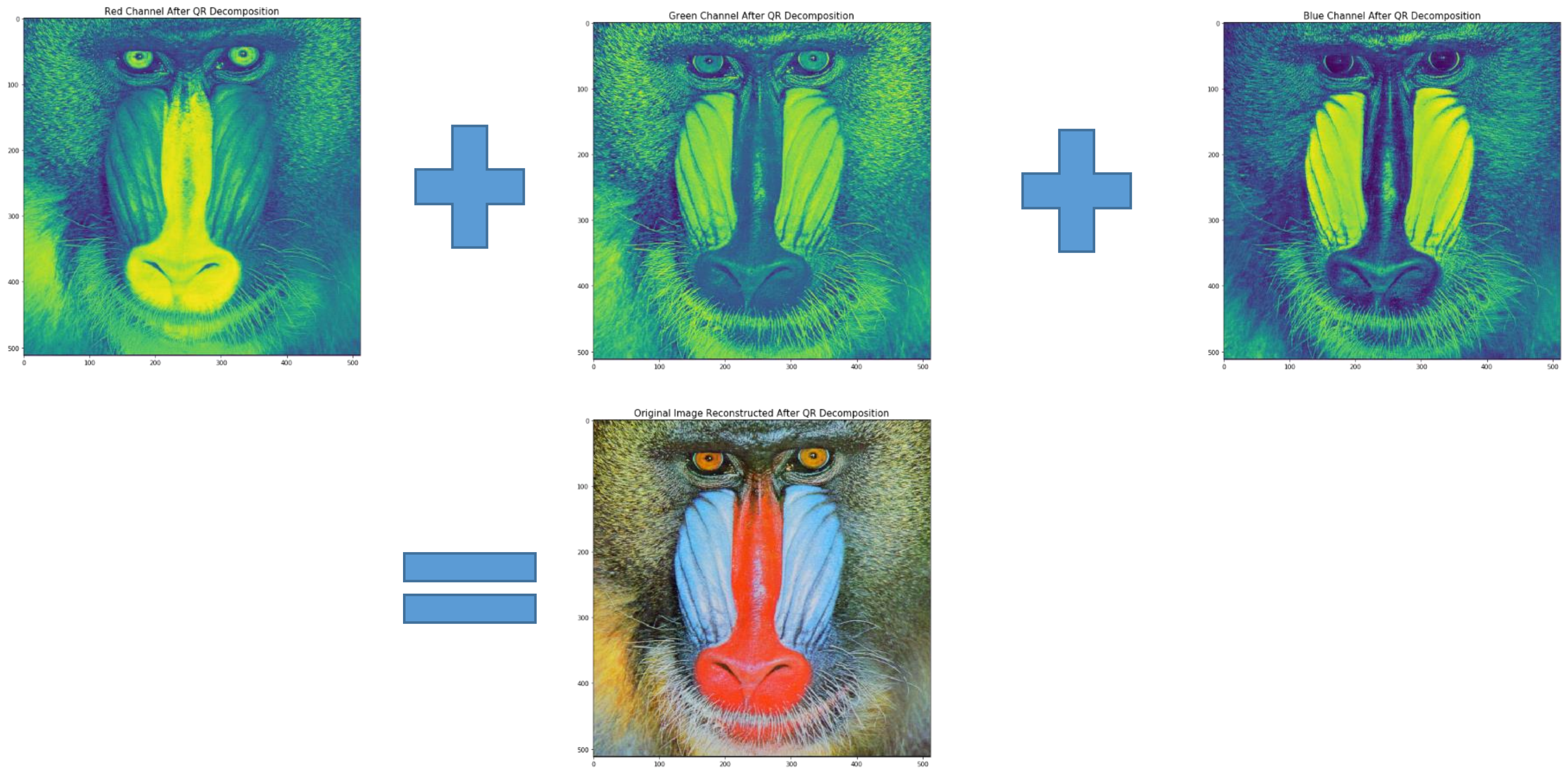
Image Decomposition: $A = QR$ & $R = Q^TA$

Red Channel Green Channel Blue Channel



Gram–Schmidt Process – Python Output

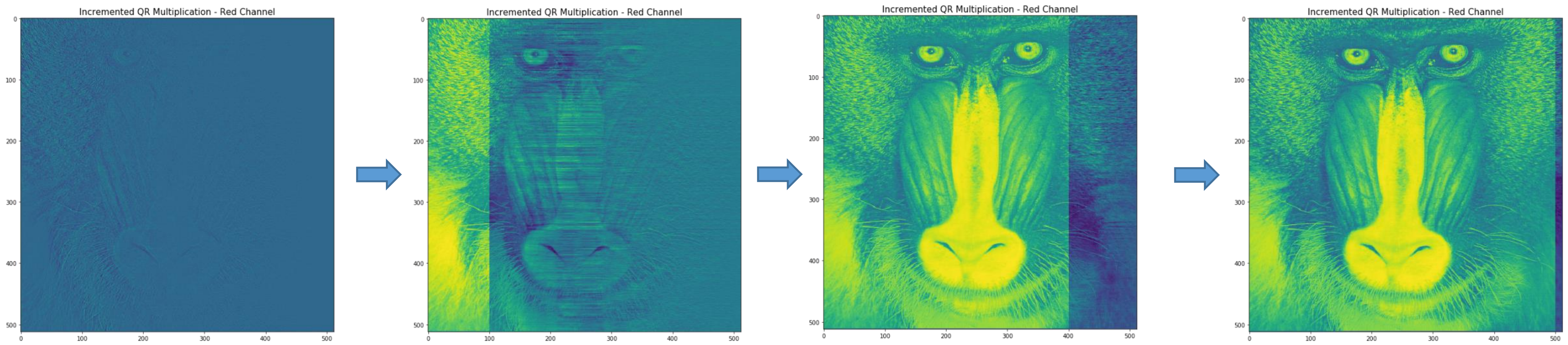
- Image Reconstruction – 3 Channels Combined to Reconstruct Original Image



Gram–Schmidt Process – Python Output

Image Reconstruction Through Incremental

A = QR Multiplication – One Channel



$A = QR$ → $R = Q^TA$ → $A = QR$

- Original channel incrementally reconstructed after successive QR multiplication, (501 iterations), reconstructing approximately, 97.8515625 % per color channel of each original channel.

SVD – Singular Value Decomposition

$$\mathbf{A} = U\Sigma V^{\mathsf{T}}$$

A = Original Matrix

$\sum$ = diagonal matrix (square root of eigenvalues of AA^T & A^TA)

U = Eigenvectors of AA^T (Left singular vectors)

V = Eigenvectors of A^TA (Right singular vectors)

$U^T U = I$ (Identity Matrix)

$V^T V = I$ (Identity Matrix)

U, Sig, V = np.linalg.svd(A)

A_Recon = np.linalg.multi_dot([U,np.diag(Sig),V])

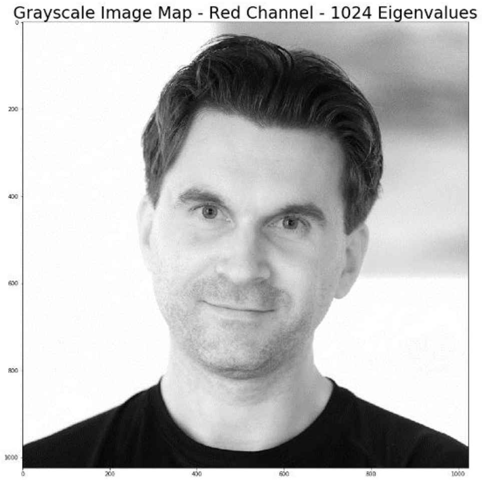
Eigenvectors and eigenvalues store key information / key features of a matrix; and help in time travel

Source: www.inverse.com

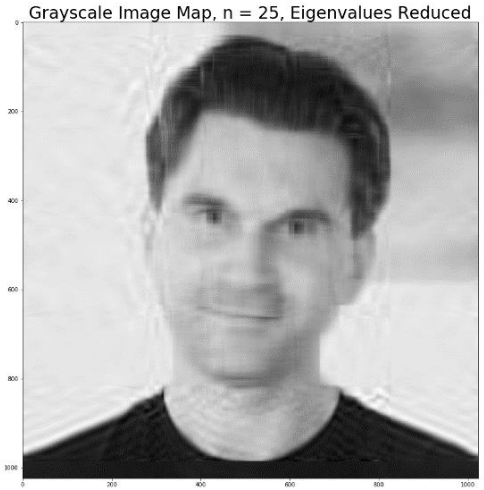
$$\begin{bmatrix} \vdots & \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \hat{u}_1 & \hat{u}_2 & \hat{u}_3 & \cdots & \hat{u}_m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \cdots & \vdots \end{bmatrix} \quad \begin{bmatrix} \sigma_{11} & 0 & 0 & \cdots & 0 \\ 0 & \sigma_{22} & 0 & \cdots & 0 \\ 0 & 0 & \sigma_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_{mn} \end{bmatrix} \quad \begin{bmatrix} \cdots & \cdots & \hat{v}_1 & \cdots & \cdots \\ \cdots & \cdots & \hat{v}_2 & \cdots & \cdots \\ \cdots & \cdots & \hat{v}_3 & \cdots & \cdots \\ \cdots & \cdots & \vdots & \cdots & \cdots \\ \cdots & \cdots & \hat{v}_n & \cdots & \cdots \end{bmatrix}$$

$U_{m \times m}$ $\quad$ $\Sigma_{m \times n}$ $\quad$ $V^T_{n \times n}$

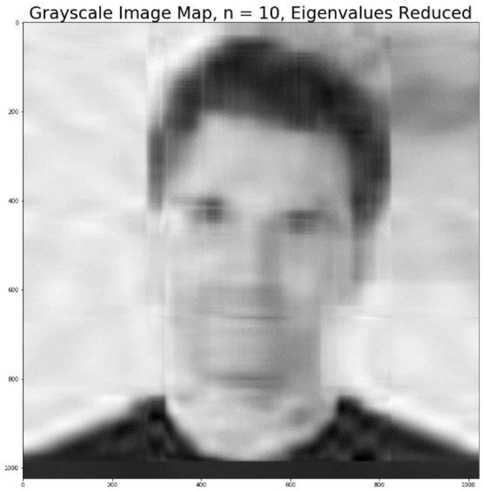
Applications of SVD – Eigenvalue Reduction



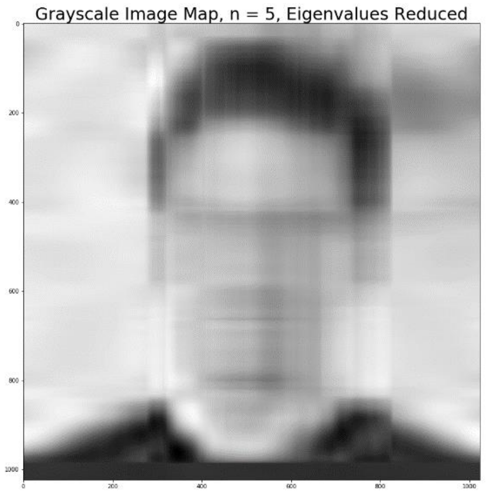
1024
Eigenvalues



25
Eigenvalues



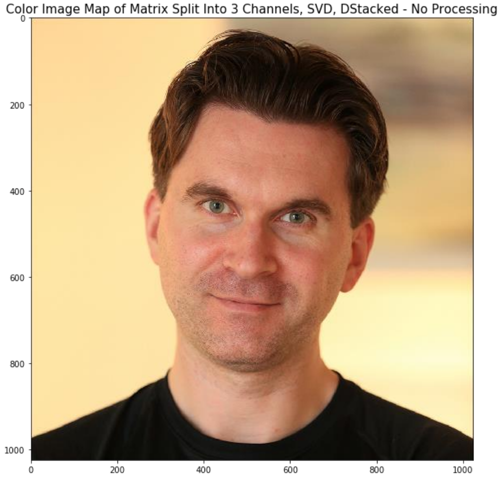
10
Eigenvalues



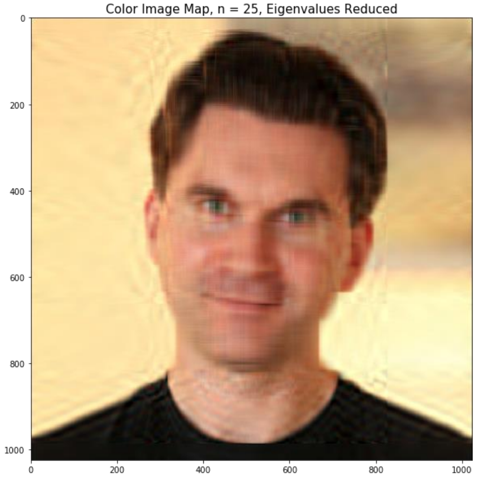
5
Eigenvalues

- Eigenvectors U and V^T remain unchanged, while eigenvalues ∑ are reduced
- Key facial features (principal components) are still recognizable

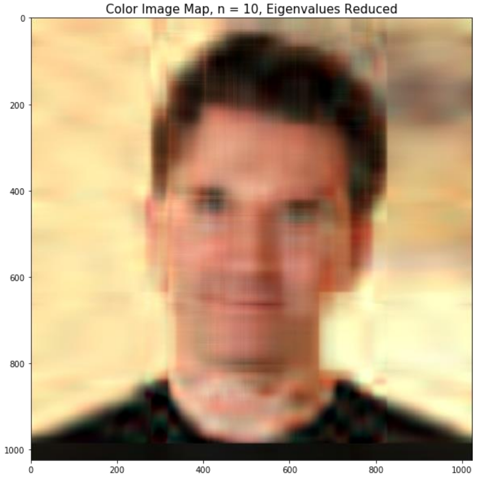
Applications of SVD – Eigenvalue Reduction



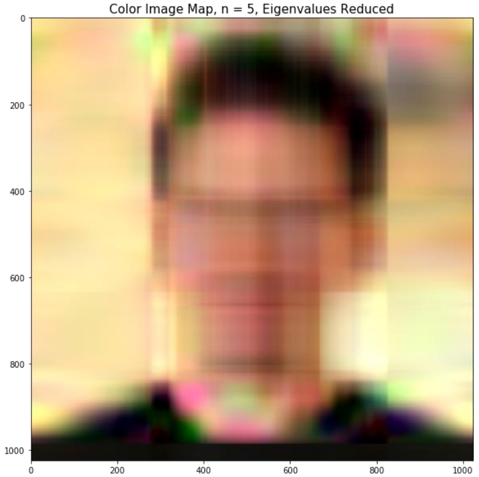
1024
Eigenvalues



25
Eigenvalues



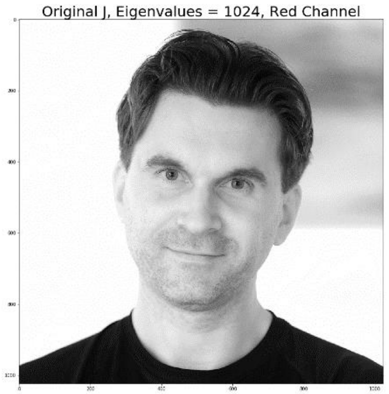
10
Eigenvalues

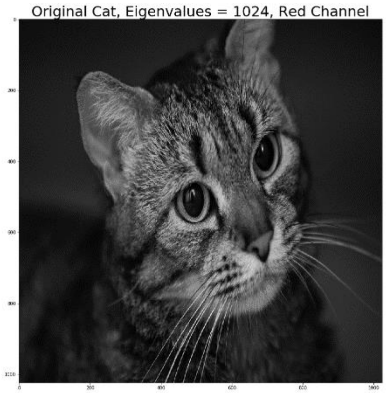


5
Eigenvalues

- Eigenvectors U and V^T remain unchanged, while eigenvalues ∑ are reduced
- Key facial features (principal components) are still recognizable
- SVD performed per channel - RGB

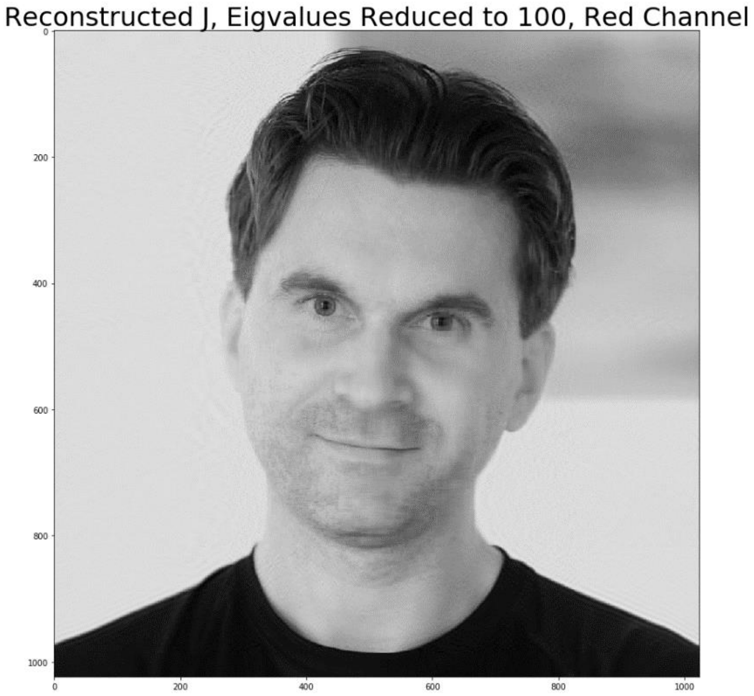
SVD – Mixing Eigenvectors and Eigenvalues

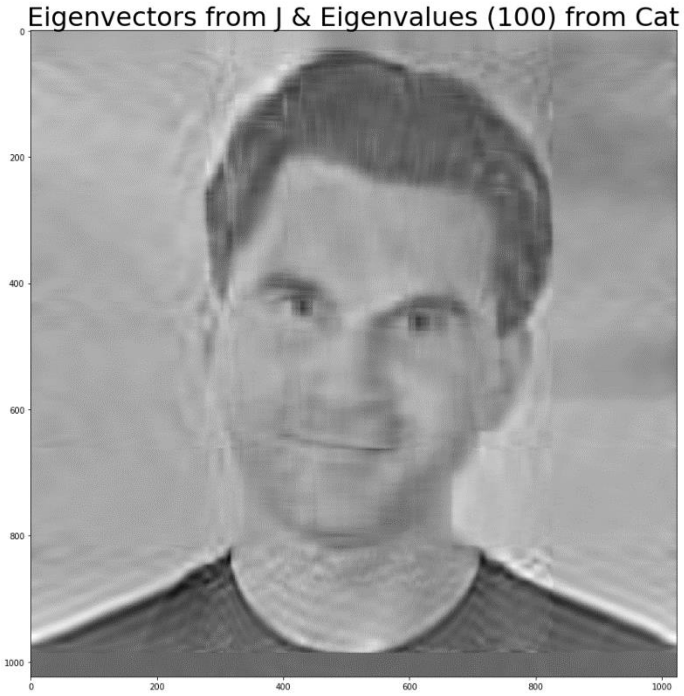




U & V^T

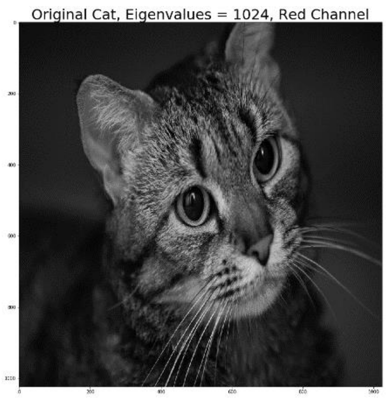
Σ

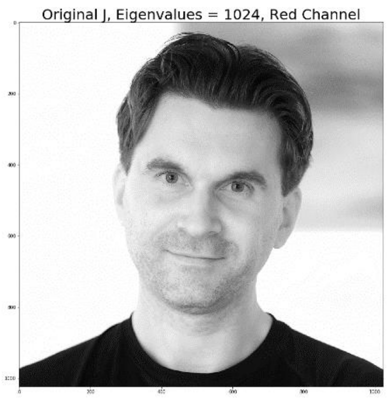




- Eigenvectors from J and Eigenvalues (reduced to 100) from Cat
- Recombined through SVD = U Σ V^T
- Eigenvalues are affecting the luminance (lights and darks) values of the image

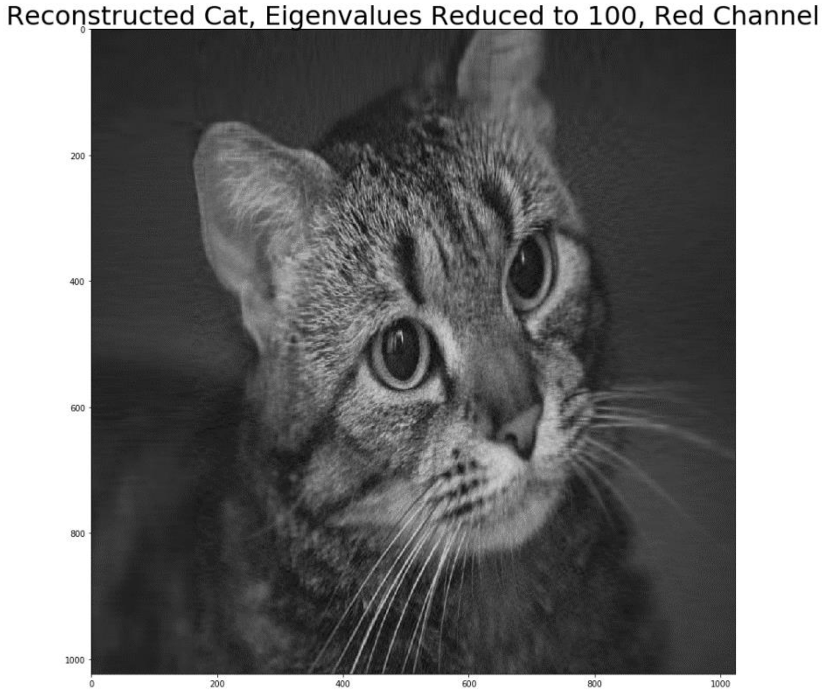
SVD – Mixing Eigenvectors and Eigenvalues

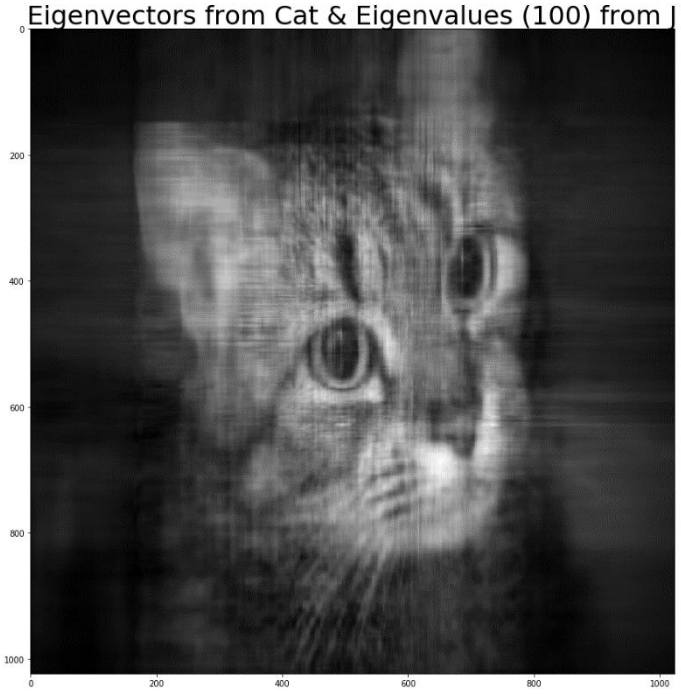




U & V^T

Σ





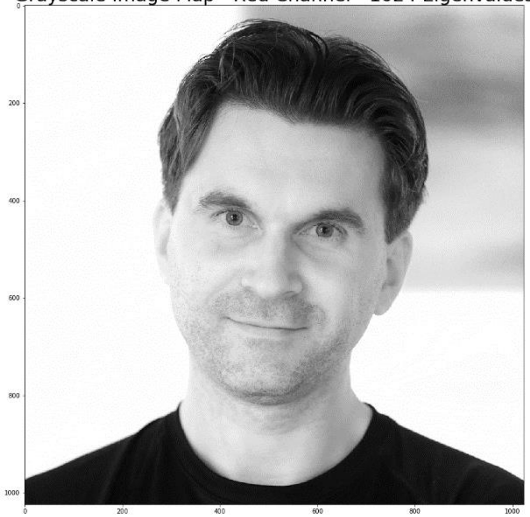
- Eigenvectors from Cat and Eigenvalues (reduced to 100) from J
- Recombined through SVD = $U \Sigma V^T$
- Eigenvalues are affecting the luminance (lights and darks) values of the image

Links

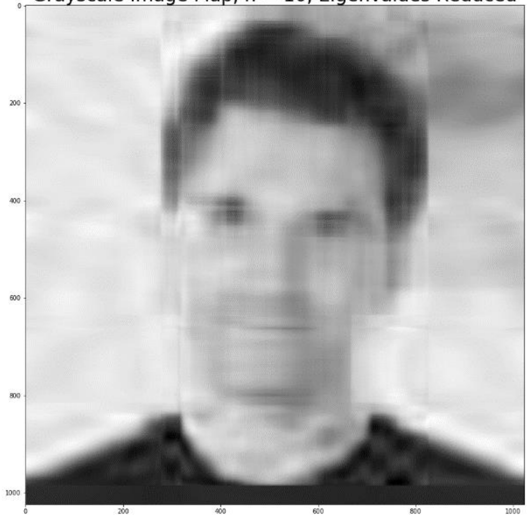
- My GitHub Page:
    - https://github.com/Johnstul
    - https://github.com/Johnstul/QR-Decomposition-Images
    - https://github.com/Johnstul/SVD-Image-Processing
    - Jstulich07@gmail.com
- Jupyter Online Notebooks:
    - https://jupyter.org/try

Thanks

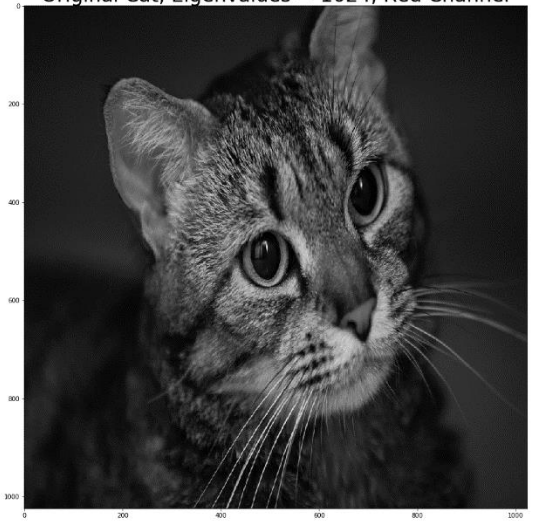
Grayscale Image Map - Red Channel - 1024 Eigenvalues



Grayscale Image Map, n = 10, Eigenvalues Reduced



Original Cat, Eigenvalues = 1024, Red Channel



ReconCat1, Eigenvalues = 10, Red Channel