PCA applied to image compression

High Performance Computing for Science and Engineering I
Data set

Gray-scale image with $m \times n$ pixels

Set of observations:

$x_k \in \mathbb{R}^m \quad k = 1, 2, \ldots, n$

Represent the data by the matrix $A \in \mathbb{R}^{m \times n}$
Algorithm

We want to maximize (under the constraint $u^T u = 1$):

$$I(u) = u^T C u = u^T \lambda u = \lambda \implies C u = \lambda u$$

Steps:
1. Normalize data $\bar{A}$ (zero-mean and unit std.)
2. Build covariance matrix $C \in \mathbb{R}^{n \times n}$
3. Solve eigenvalue problem using LAPACK
4. Compress the image by representing the uncompressed data $A$ in a subspace spanned by the basis $u_k, k = 1, 2, \ldots, p$ where $p < n$. 
You can solve an eigenvalue problem by using the \texttt{dsyev} routine from the LAPACK library.

Use \textit{workspace} for best performance:

- You can set \texttt{LWORK} = -1
- Call \texttt{dsyev} to determine optimal size of the workspace (eigenvalue problem is not solved in this case)
- Allocate the workspace and call \texttt{dsyev} a second time passing the pointer to the workspace array (workspaces are used in LAPACK for blocking)
Image compression

Image can be compressed by only taking into account $p < n$ (principal) components.

$$B = [u_1, u_2, \ldots, u_p] \in \mathbb{R}^{n \times p} \quad (\lambda_1 > \lambda_2 > \cdots > \lambda_p)$$

Compute the $p$ principal components $P \in \mathbb{R}^{m \times p}$

$$P = \bar{A}B$$

where the matrix $\bar{A}$ is the normalized image data.
Image decompression

To decompress the image (reconstruction), we simply invert the process. We compute

$$PB^T = \bar{A}BB^T = \bar{A}$$

The last step is then to denormalize $\bar{A}$ to get back to an approximation of $A$ (lossy compression).