Comparing Lossy Image Compression Algorithms

Kevin Slater

Introduction

Most of us make use of image compression every day, and there is a constant demand to find the best way to store quality images in the smallest amount of space.

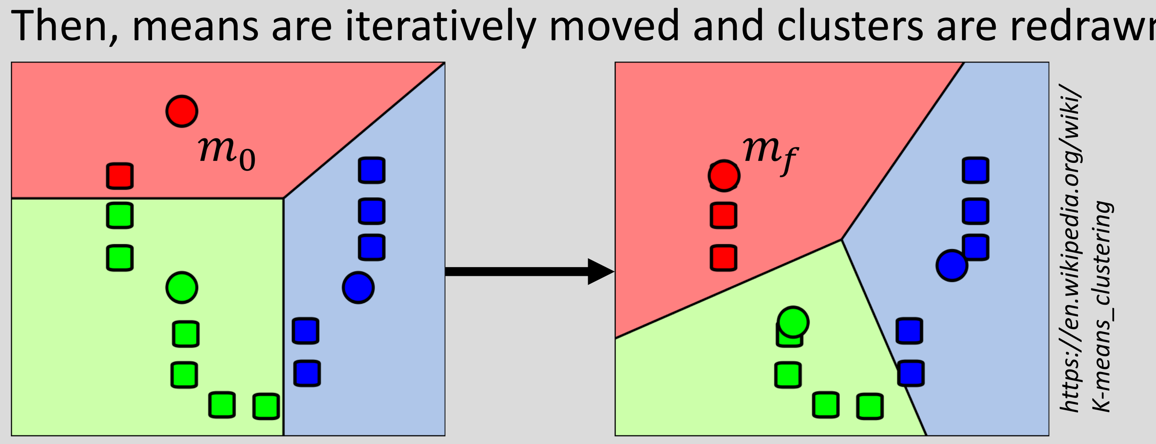
To investigate the strengths and weaknesses of different image compression methods, I implemented them and measured their tradeoff between quality and compression ratio on a selection of images. These results highlight some of the challenges faced in image compression, and suggest optimal use cases for each technique.

Methods

Original images were read as height x width x RGB arrays using the opencv python package, then compressed to varying degrees with:

1. K-means clustering

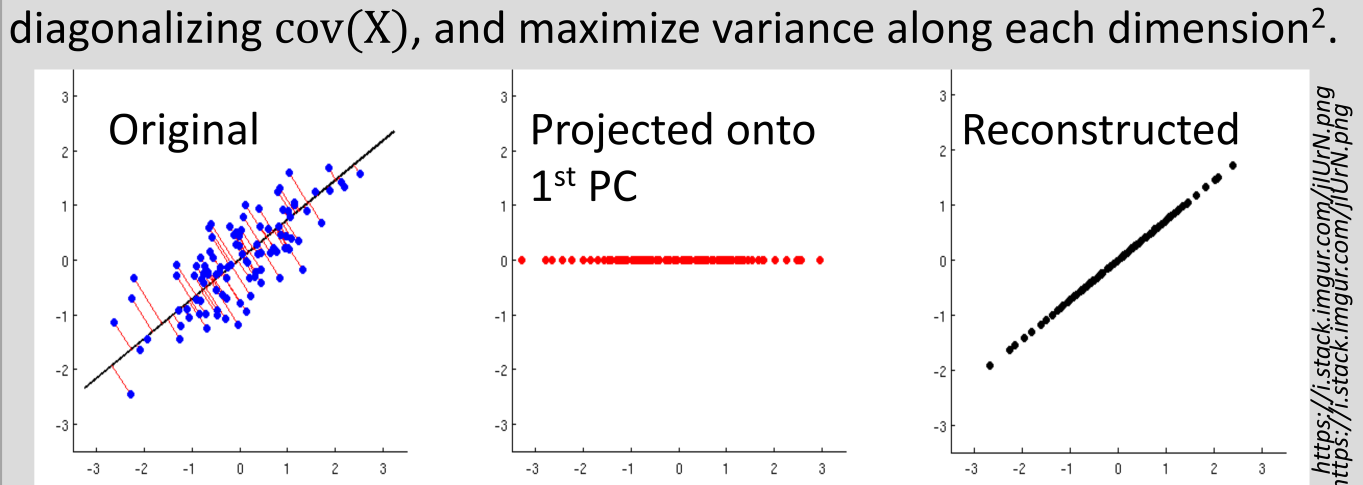
From an initial set of k means, clusters pixels around their nearest mean[1]. Then, means are iteratively moved and clusters are redrawn:



Instead of storing (R,G,B) value for each pixel, we only need to store its cluster index, and the cluster means.

2. Principle Component Analysis

PCA reduces the dimension of a dataset X_{m*n} by projecting it onto its principle components V_{n*k}. The principle components are found by diagonalizing $\mathrm{cov}(X)$, and maximize variance along each dimension[2].

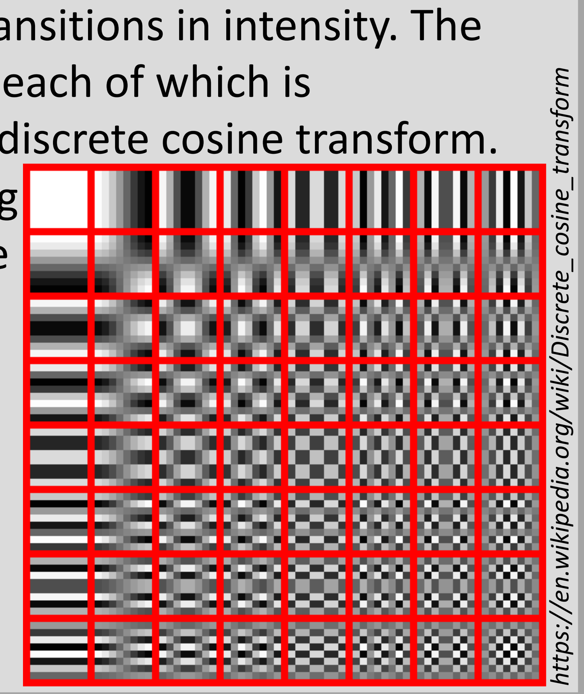


Space is saved by only storing the projected data $Z_{m*k}=XV$ and the projection matrix V.

3. JPEG Compression

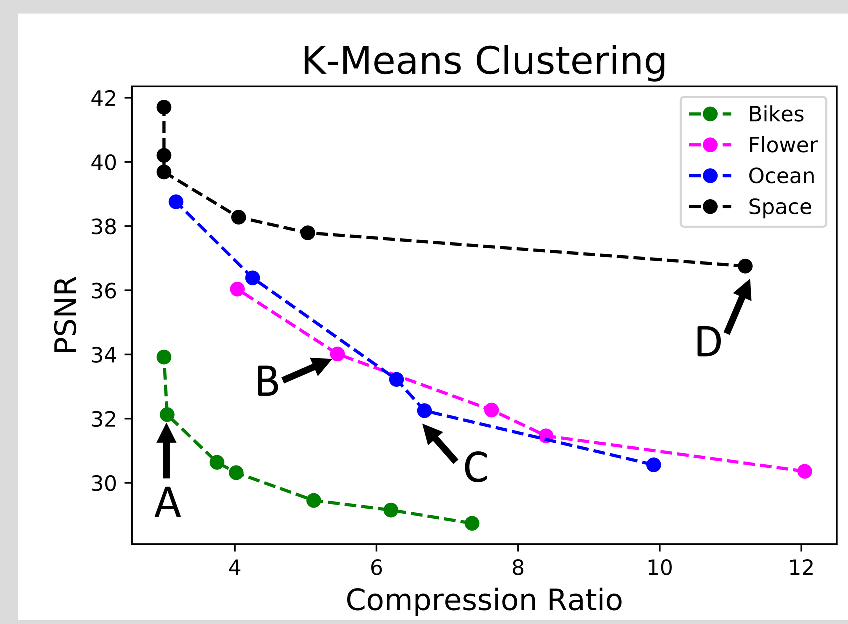
JPEG compression uses the fact that our eyes are not very sensitive to high-frequency information like sharp transitions in intensity. The original image is broken into 8x8 blocks, each of which is transformed to frequency space using a discrete cosine transform. Higher frequencies are set to zero, letting the compressed image be encoded more efficiently[3].

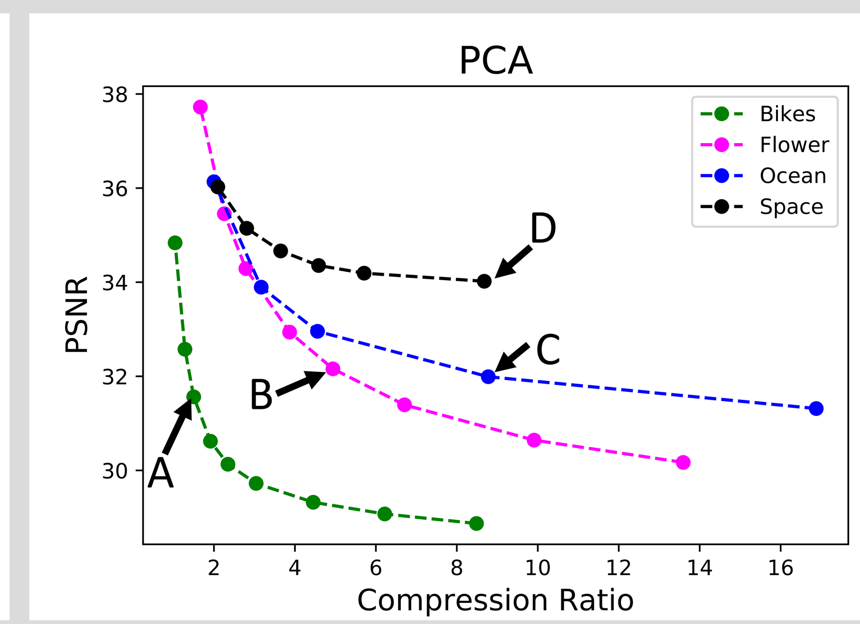
DCT is preferred to FFT, since it implies an even (instead of periodic) extensionof the signal. This results in fewer boundary discontinuities, and convergence in fewer terms[4].

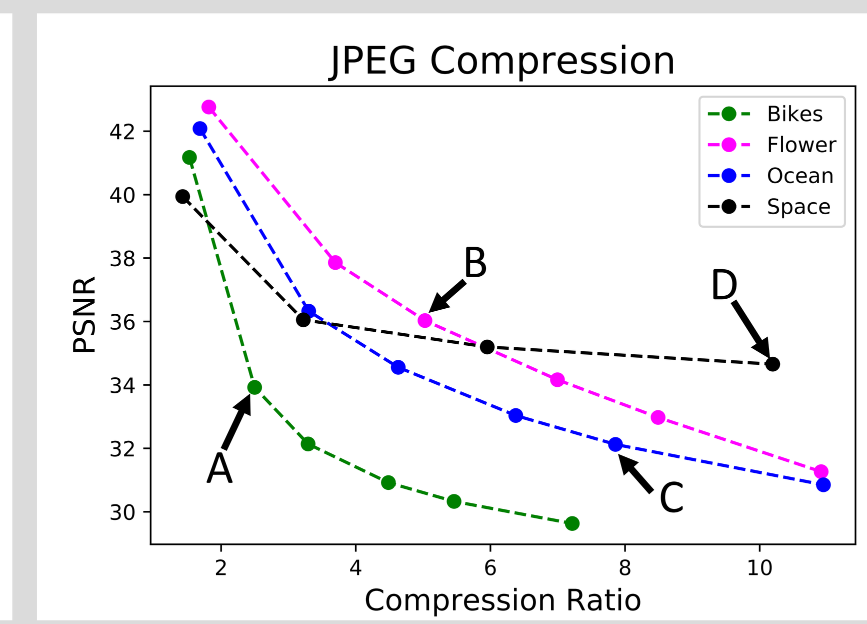


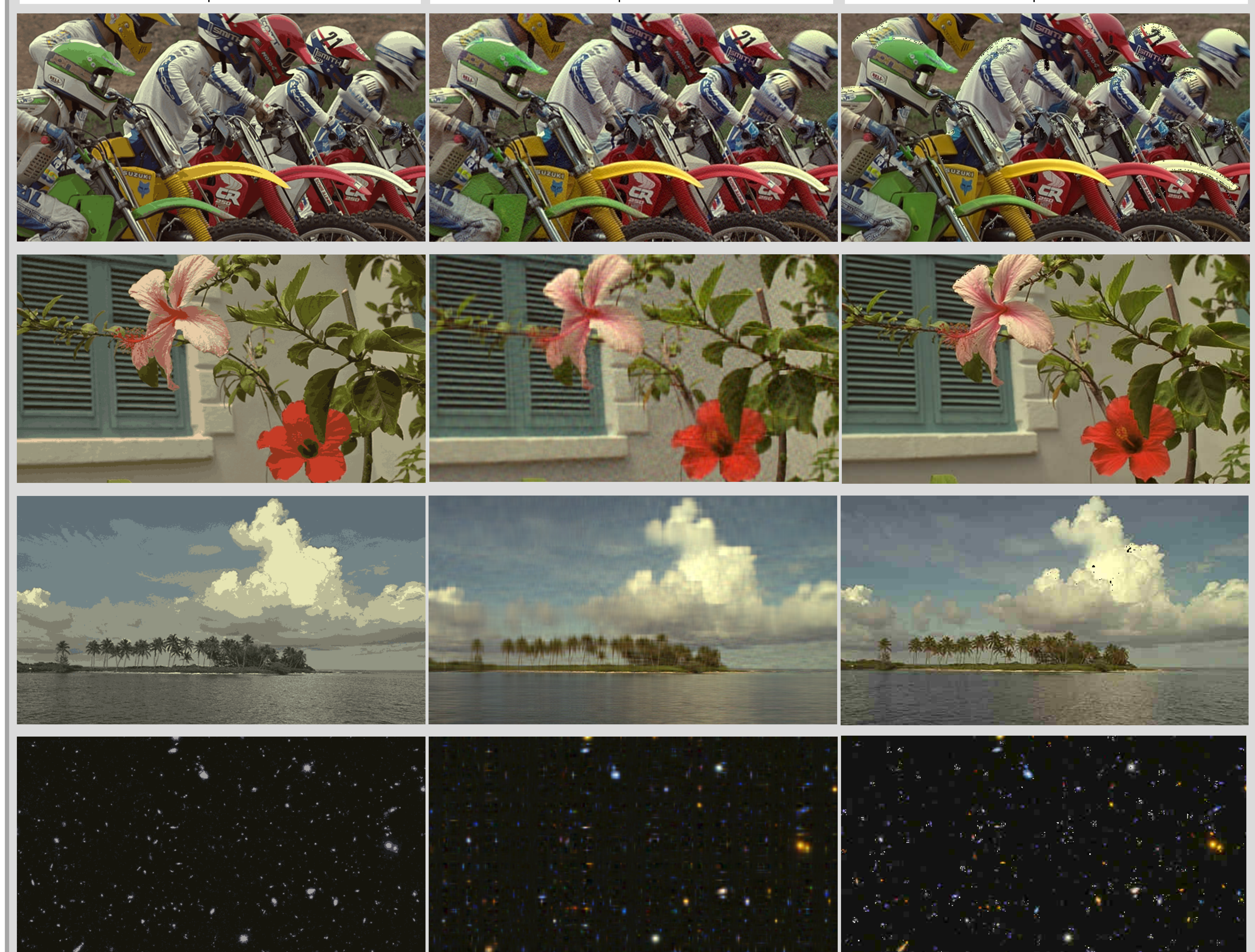
Results

Compressed image quality was measured by the peak signal to noise ratio: $PSNR = 20\log(\frac{255}{\sqrt{MSE}})$. A larger PSNR represents less noise added in compression, and thus better image quality[5].









Conclusions

PSNR is not a perfect gauge of compressed image quality, but it gives some objective measurements:

1. K-means clustering

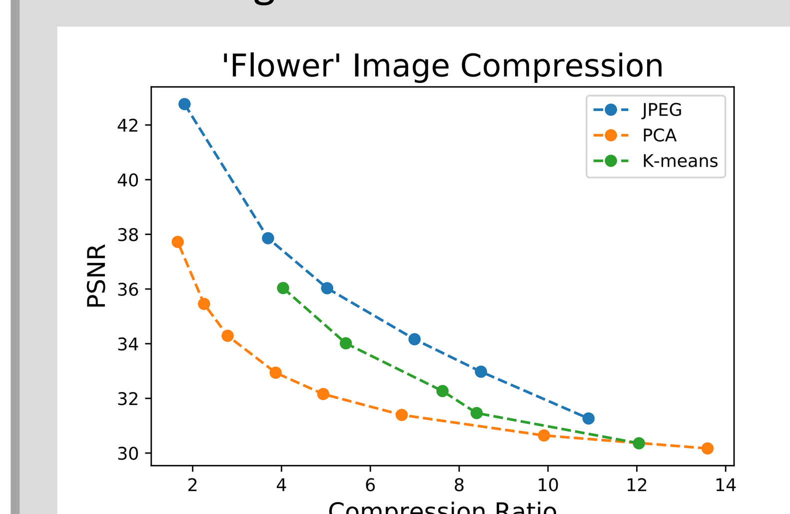
In general, k-means clustering works best for images with a small number of distinct colors. For images with lots of distinct or rapidly varying colors, it takes longer to converge and the compressed image takes more space to store.

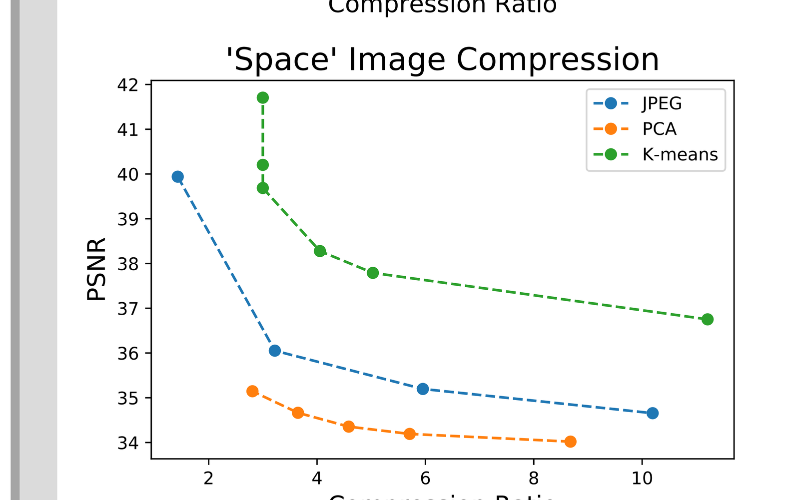
2. Principle Component Analysis

This method generally performed the worst as measured by PSNR – however, we can expect that images with less variance and fewer principle components will be compressed more efficiently.

3. JPEG Compression

This was the best compression method overall, but suffered from artifacts at the borders of sharply contrasting colors. Therefore this method appears to work best on images with less contrast.





Ultimately, the "best" compression method depends on the image and properties you are willing to compromise for compression.

Sources

https://github.com/kevinslater/PHYS250_Final

1. *https://en.wikipedia.org/wiki/K-means_clustering*
2. *http://www.dsc.ufcg.edu.br/~hmg/disciplinas/posgraduacao/rn-copin-2014.3/material/SignalProcPCA.pdf*
3. *http://www.maths.usyd.edu.au/u/olver/teaching/Computation/ExampleProject.pdf*
4. *https://en.wikipedia.org/wiki/Discrete_cosine_transform*
5. *https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio*