Linear Algebra in File Compression: SVD and DCT

By: Andrew Fraser

How Are Images Stored?

- Images are generally stored and visualized through storing a 2D array of values, called Raster images, which are meant to correspond to the amount of shading each pixel has
- For a colored image, three matrices are used instead to store the Red, Green, and Blue values of the RGB format
- Popular forms of image storage use different methods to compress their data:
- PNG: Raster format with lossless compression
- JPEG: Discrete Cosine Transform (DCT) with lossy conversion. Known to compress to 1/10th of a file’s original size with little visual loss.

Effectiveness of Compression

- Many images can be compressed around to around 1/10th of their original size, while still remaining quite recognizable
- Makes streaming, a service that often loads 60 images per second, into something possible to do without ridiculously fast internet speeds
- Even in cases where high-quality images must be preserved, lossless conversions help to keep image sizes down
- Different methods of bit storage can also help in compression

Singular Value Decomposition

- In Linear Algebra, it turns any matrix A into the form $U\Sigma V^T$
- Based upon the singular values of A, which are found by taking the square root of each eigenvalue of A^TA
- U = Colspace of A and nullspace of A^T, all orthogonalized. mxm
- Σ = Diagonal matrix, with each diagonal containing a singular value of A, going from greatest to least. Same size as A, which is mxn
- V = A matrix with its columnspace comprised of the eigenvectors of A^TA. Also happens to be the rowspace of A and nullspace of A all orthogonalized. nxn
- V^T = Transpose of V

SVD in File Compression

- With larger matrix sizes, many singular values held in the Σ matrix become very small
- By removing many smaller values in the Σ matrix while keeping the larger ones, many rows can be removed from U as well as many columns from V^T, as they would just be multiplied by zeroes anyway
- By keeping the larger values, all three matrices that must be stored become much smaller, but most of the meaningful image values are still kept
- Thus, SVD results in a lossy compression, but it still keeps the image's meaning

Discrete Cosine Transformation

- Involves splitting up the image matrix into many NxN matrices, then multiplying each by the NxN DCT matrix, which is calculated using a complex set of calculations involving cosine, matrix size, and relative column/row sizes
- Then, for each NxN matrix, symbolized by M, calculate the compressed form of that matrix by performing the following matrix multiplies:
- $D = TMT^T$
- D = Compressed coefficients of the image matrix and T = The DCT matrix

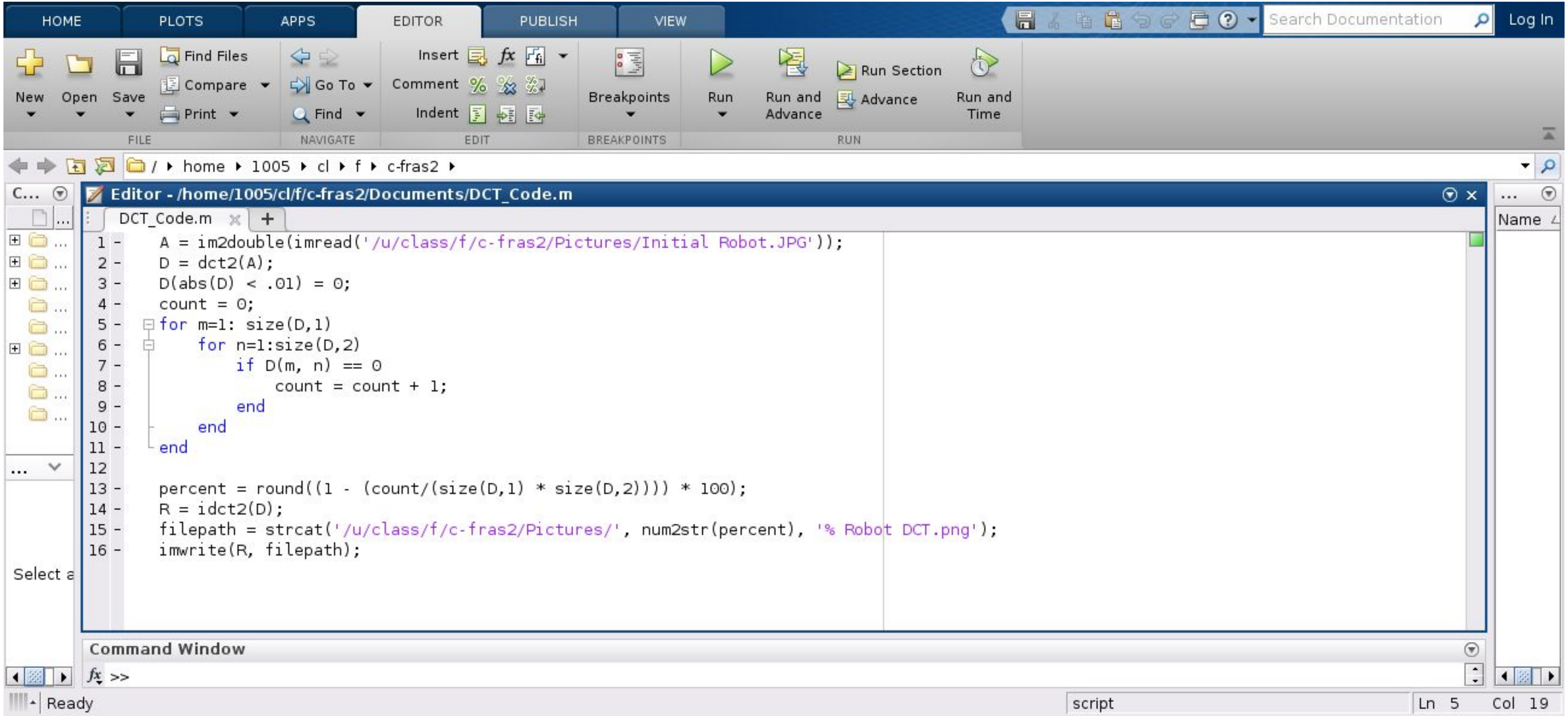
Discrete Cosine Transformation (contd.)

- Then, each matrix D derived from the previous formula is multiplied by a matrix Q_X, which is a set constant matrix based upon how high quality the user wants the image to be on a scale of 100. For example, multiplying by Q_{10} results in a very low quality image with a very high compression ratio, whereas multiplying by Q_{90} produces a higher quality image that is not compressed as effectively.
- Matrices are ordered by sensitivity to human eye, top left = most sensitive, bottom right = least sensitive
- Many values that aren't in the top left end up being nearly zero, allowing for many to be brought to zero and lots of space to be saved
- Undoing this entire process resulting in decompressing the image

```
> with(LinearAlgebra):
> with(ImageTools): # Necessary to read images as matrices and manipulate them
> img:= ToGrayscale(Read("/u/class/f/c-fras2/Downloads/Robot.jpg")): # Reads the black and white image
> Write("/u/class/f/c-fras2/Pictures/Initial Robot.JPG", img):

> U:= LinearAlgebra[SingularValues](img, 'output = U'):
> S:= LinearAlgebra[SingularValues](img, 'output = S'):
> Vt:= LinearAlgebra[SingularValues](img, 'output = Vt'):
  C := 5/100:
> for i from (round((RowDimension(S) * C) + 1)) to RowDimension(S) do
  S[i] := 0:
  end do:

> DiagS:= DiagonalMatrix(S, RowDimension(img), ColumnDimension(img)):
> CompressedImage:= U.DiagS.Vt:
> Write((cat("/u/class/f/c-fras2/Pictures/", convert(round(C * 100), string), "% Robot SVD.png")), CompressedImage)
  :
>
```


HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Log In
New
Open
Save
Find Files
Compare
Print
FILE
Go To
Find
NAVIGATE
Insert
Comment
Indent
EDIT
Breakpoints
BREAKPOINTS
Run
Run and Advance
Run Section
Advance
Run and Time
RUN
/ home 1005 cl f c-fras2
Editor - /home/1005/cl/f/c-fras2/Documents/DCT_Code.m
DCT_Code.m
A = im2double(imread('/u/class/f/c-fras2/Pictures/Initial Robot.JPG'));
D = dct2(A);
D(abs(D) < .01) = 0;
count = 0;
for m=1: size(D,1)
for n=1:size(D,2)
if D(m, n) == 0
count = count + 1;
end
end
end
percent = round((1 - (count/(size(D,1) * size(D,2)))) * 100);
R = idct2(D);
filepath = strcat('/u/class/f/c-fras2/Pictures/', num2str(percent), '% Robot DCT.png');
imwrite(R, filepath);
Command Window
>>
Name
Select a
Ready
script
Ln 5
Col 19


INFO RADIO
93.9 FM

69% DCT

75% SVD

47% DCT

50% SVD

35% DCT

37% SVD

20% DCT

20% SVD

12% DCT

10% SVD

2% DCT

1% SVD

76% DCT

75% SVD

57% DCT

50% SVD

34% DCT

37% SVD

22% DCT

20% SVD

11% DCT

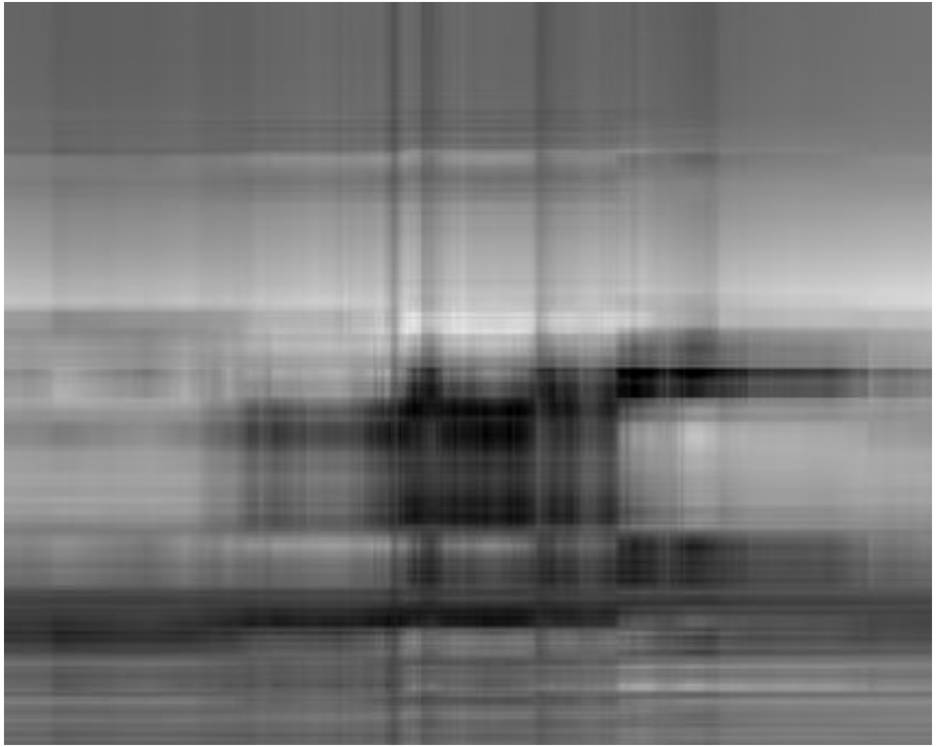
10% SVD

4% DCT

5% SVD

1% DCT

1% SVD



Citations

https://www.math.cuhk.edu.hk/~lmlui/dct.pdf
http://videocodecs.blogspot.com/2007/05/image-coding-fundamentals_08.html
http://www.mvnet.fi/index.php?osio=Tutkielmat&luokka=Yliopisto&sivu=Image_compression
https://ntrs.nasa.gov/search.jsp?R=19920024689
https://www.sitepoint.com/gif-png-jpg-which-one-to-use/