Max Kramer

I affirm that I have adhered to the honor code on this assignment

> *Hello again, scientist! I'll do all my writing in italics, and problems for you will be in* ***bold.*** *Comment your code, and* explain your ideas in plaintext. *As a general rule, I expect you to do at least as much writing as I do. Code should be part of your solution, but I expect variables to be clear and explanation to involve complete sentences. Cite your sources; if you work with someone in the class on a problem, that's an extremely important source. Don't work alone.*

Problem F.06: Compress the image.

> *Okay, first you need an image. The internet has plenty. Choose something that's roughly square. No memes.* ***Import it with imread() and call it A.***

```
A = imread('lowpoly.png');
```

> *Okay, now we're going to clean it and make it 400x400.* ***Uncomment this.***

```
A = imresize(A,[400 400]);
A = rgb2gray(A);
A = double(A);
A = A-min(A(:));
A = A/max(A(:));
figure;
imshow(A);
```

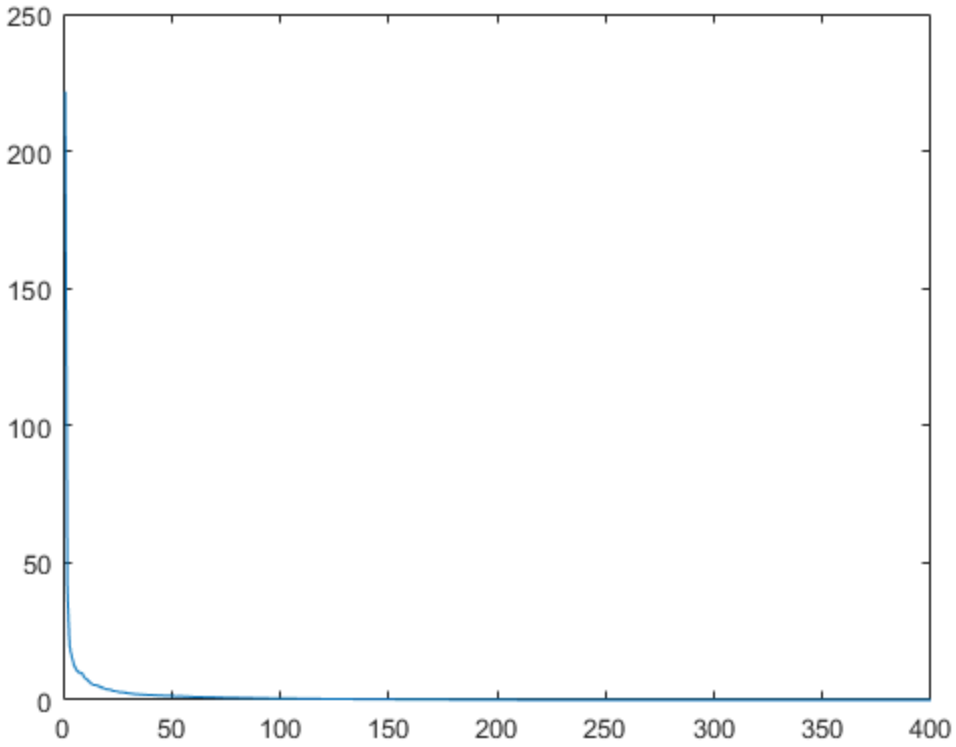
Find the SVD of A. *How many singular values are there?*

```
[U,S,V] = svd(A);
```

There are technically 400 nonzero singular values in S. The values start larger and quickly approach 0, but never reach 0.

Show me a plot of the singular values. *They'll probably drop off pretty quickly; by construction, as much information as possible is retained by the first singular component.*

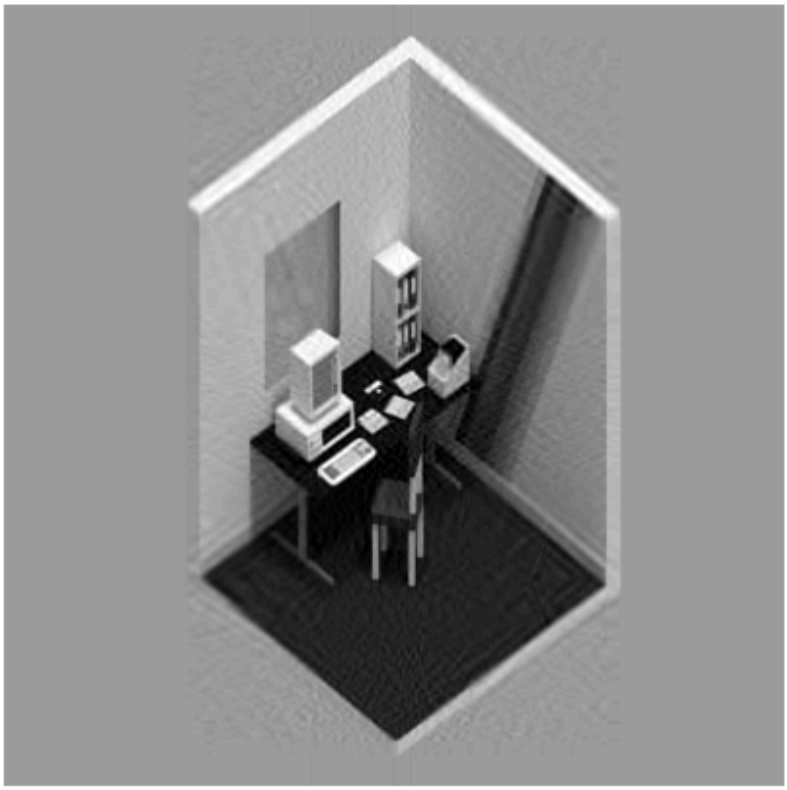
```
L = [1:1:400];
figure;
plot(L,diag(S))
```



The diag() command creates a vector of the entries on the main diagonal of the supplied matrix. Plotted against an axis of 1 to 400 are the singular values taken using diag(S).

Okay, now ***set all but the first 50 singular values equal to 0 and show me the compressed image.*** *The preserve() function will be helpful.*

```
Snew = preserve(S,50);
Acomp = U * Snew * V';
figure;
imshow(Acomp)
```



The preserve(S,k) helper function sets every diagonal value of S after K to be equal to 0. After using preserve, only the first 50 singular values remain in S.

What percentage of the total variance does this preserve? ***Uncomment this and explain what it's computing.***

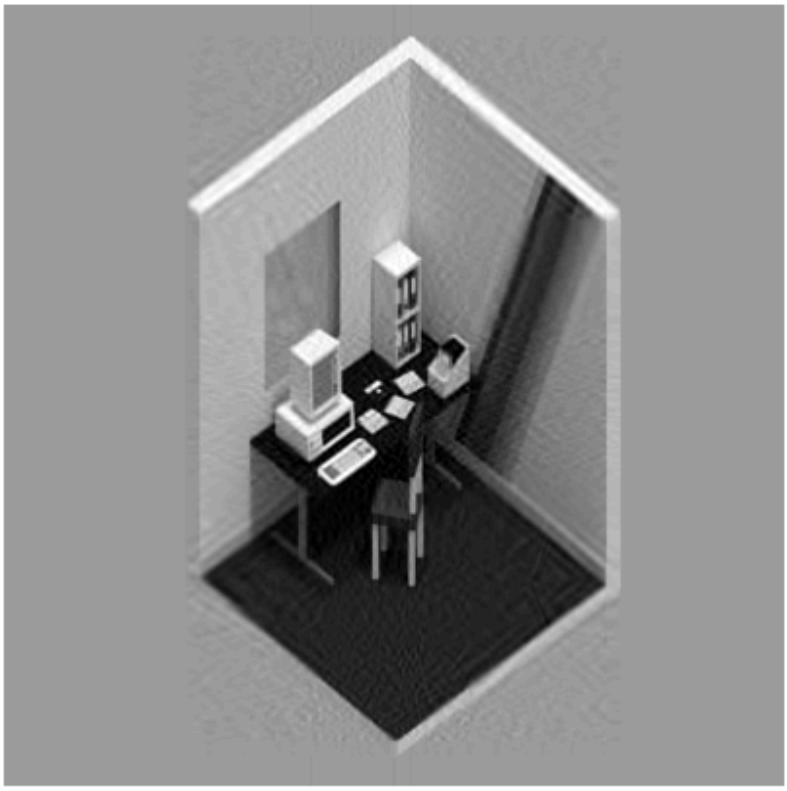
```
svals = diag(S);
vpa(sum(svals(1:50).^2)/sum(svals.^2),6)
```

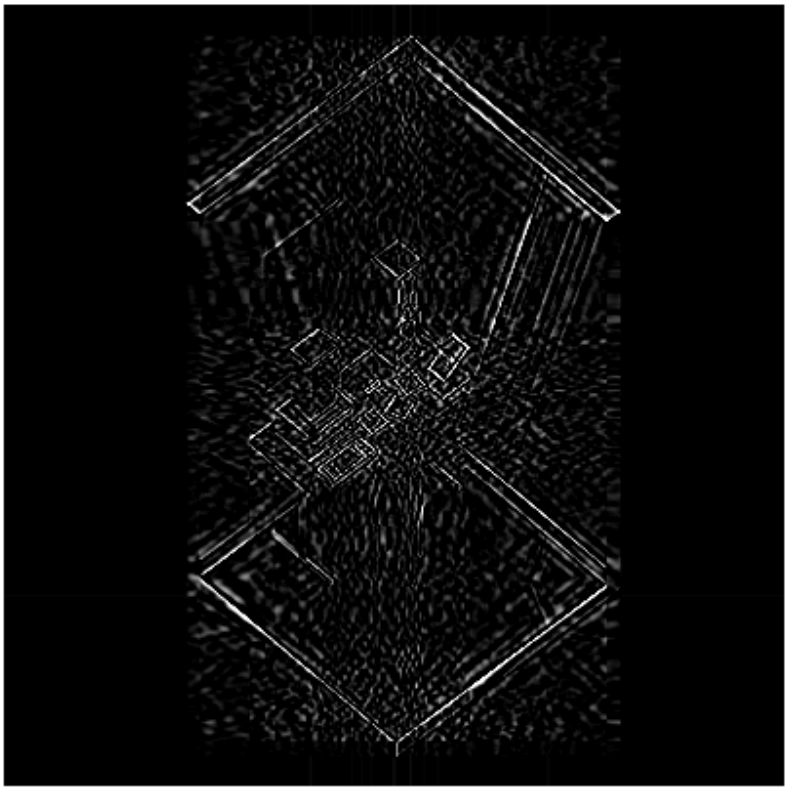
```
ans =

0.999011
```

Svals contains every singular value of the matrix S. The variance of a matrix is the sum of the squared entries on its main diagonal. The variable precision arithmetic command is calculating the proportion of variance out of the total that is accounted for by the first 50 entries of svals. The percentage of the total variance preserved by the first 50 singular values is 99.9011%

Show me the "residual" image, which what you get if you preserve not the first 50 singular values but the last 350. *(Hint: S - preserve(S,50). (Further hint: if the residual image is almost solid black, you can "turn it up" by multiplying through by 10 or so.)*

```
Sres = S - preserve(S,50);
Ares = U * Sres * V';
figure;
imshow(Ares * 10)
```





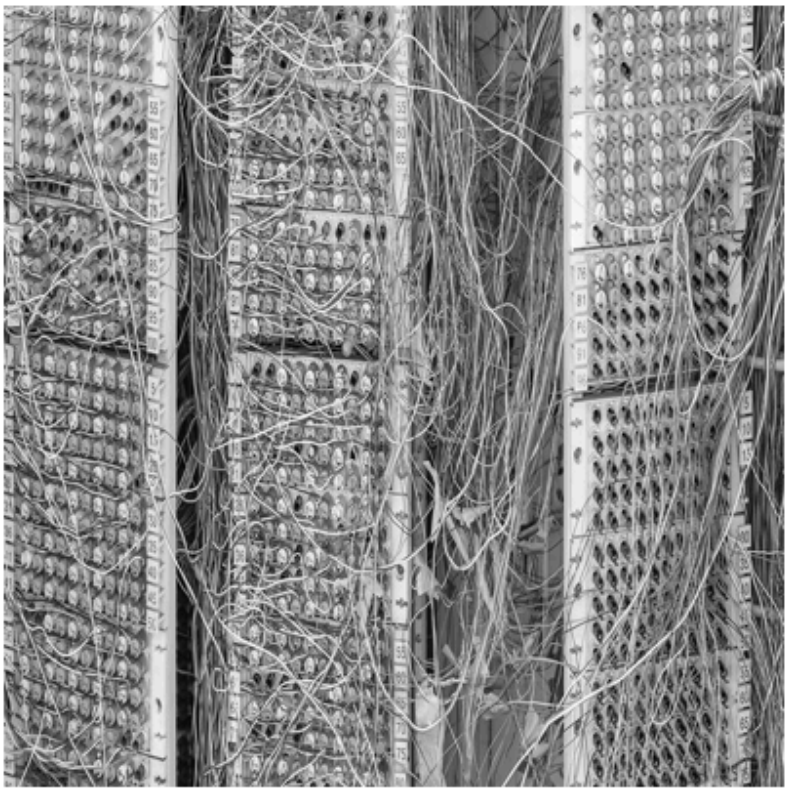
Sres is calculated by subtracting the preserved 50 singular values from the full list, saving only the last 350 values. The SVD is then used with the new Sres to calculate Anew, the residual image. The image was almost entirely solid black, so the shown image is Anew * 10.

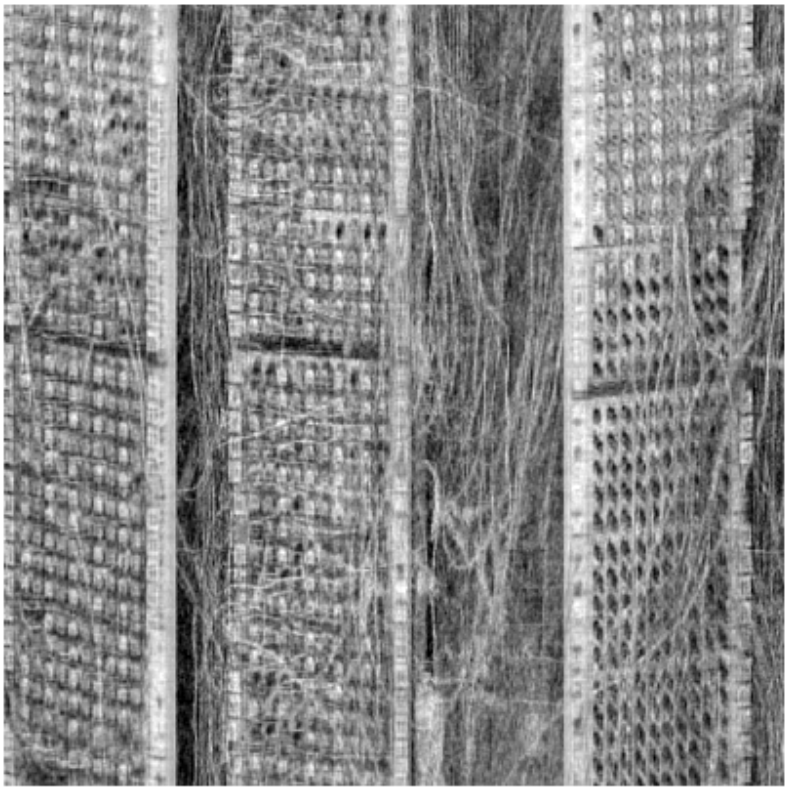
Okay, now let's play. Your goal is to ***find photographs B and C for which preserving the first 50/400 singular values gives amazing and not-so-amazing results. Show me the images, the compressed images, and the percent of total variance preserved for each.*** *Based on your exploration, what seems to make an image "complex?"*

```
B = imread('switchboard.jpg');
B = imresize(B,[400 400]);
B = rgb2gray(B);
B = double(B);
B = B-min(B(:));
B = B/max(B(:));
figure;
imshow(B);
[U2,S2,V2] = svd(B);
Snew2 = preserve(S2,50);
Acomp2 = U2 * Snew2 * V2';
svals2 = diag(S2);
var2 = vpa(sum(svals2(1:50).^2)/sum(svals2.^2),6)
figure;
```

```
imshow(Acomp2)
```

```
var2 =

0.974508
```





The switchboard seen here is a good example of the first 50 singular values not sufficing to capture detail in the image. The complexity of this image seems to be the patch cables, which are both very close together and very different colors. It could be that the number of edges and gradients in this image is what results in the subpar compression results. The preserved variance was still quite high at 97.4508%

```
C = imread('simstroke.jpg');
C = imresize(C,[400 400]);
C = rgb2gray(C);
C = double(C);
C = C-min(C(:));
C = C/max(C(:));
figure;
imshow(C);
[U3,S3,V3] = svd(C);
Snew3 = preserve(S3,50);
Acomp3 = U3 * Snew3 * V3';
svals3 = diag(S3);
var3 = vpa(sum(svals3(1:50).^2)/sum(svals3.^2),6)
figure;
imshow(Acomp3)
```

var3 =

```
0.995587
```

While I selected this image as it is known for being unidentifiable in humans, I was mostly curious to see if the nonsensical shapes would be retained. In fact, they were. The way we think of complexity in an aesthetic sense may not translate to this context. The preserved variance was 99.5587%

Published with MATLAB® R2019b