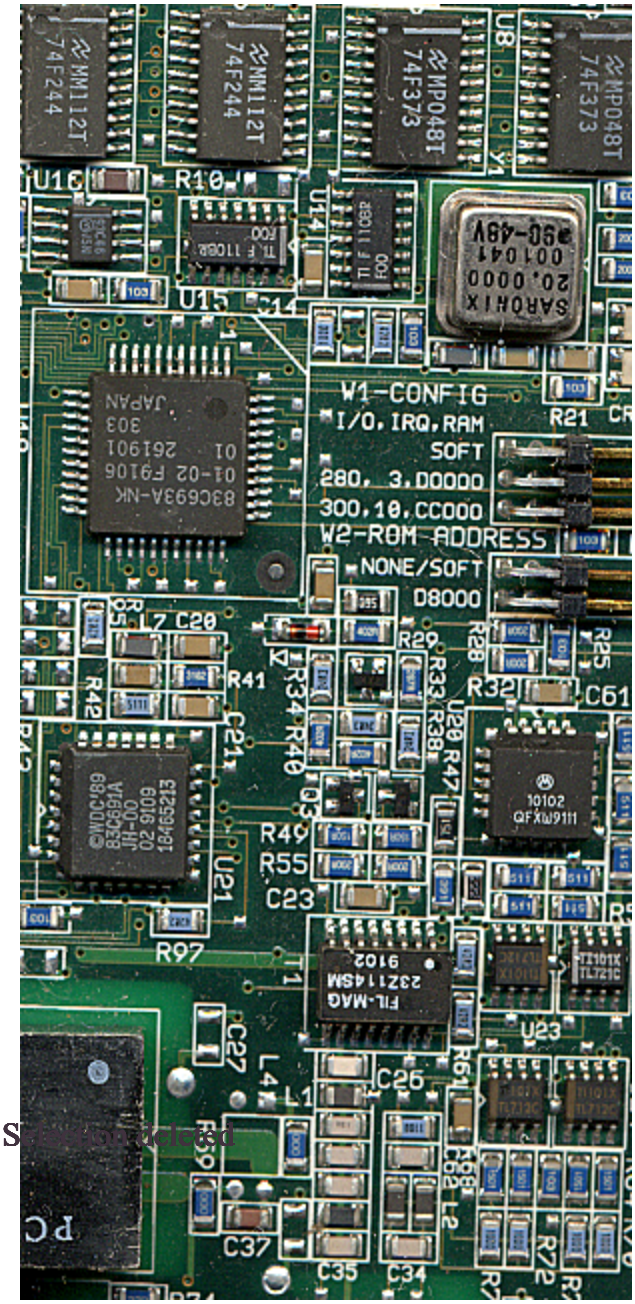
```
# 590 Final Project Matthew Soham

# Need to reconstruct image using less singular values and then determine compression ratio
```

```
using ImageQualityIndexes, Plots,TestImages, LinearAlgebra, ImageView, Images, ImageMagick, FileIO, Wavelets, DSP
```

img =



Stack Overflowed

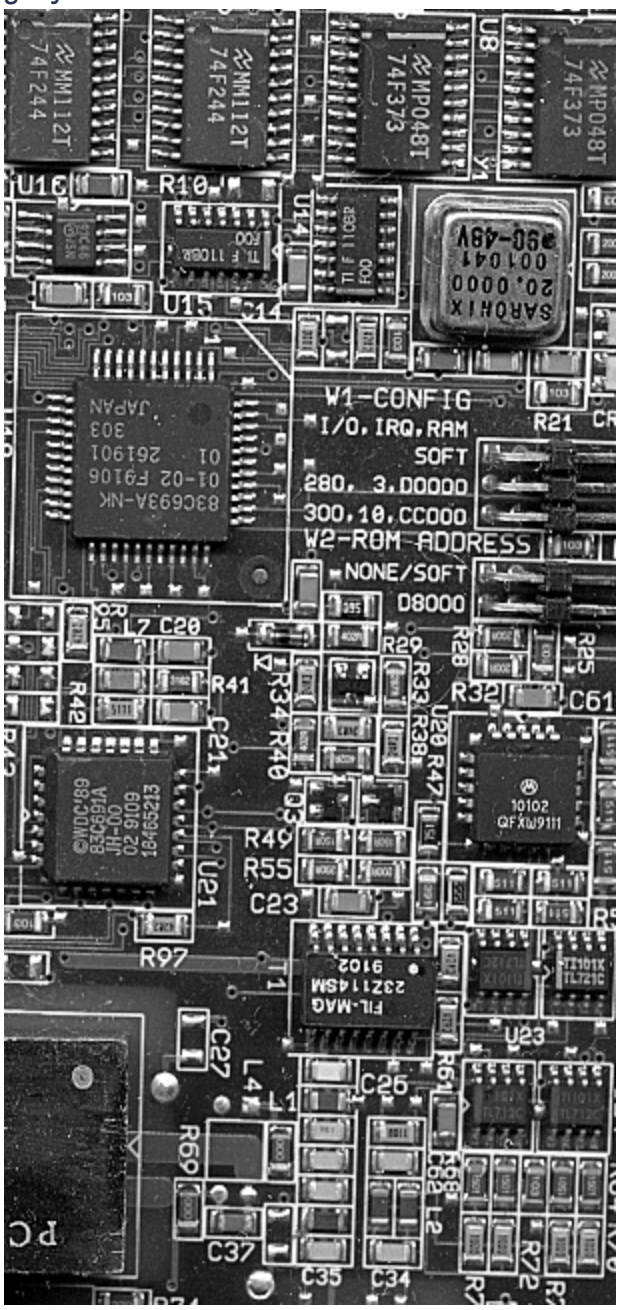
```
img = load("board.tif")
```

```
3×648×306 reinterpret(reshape, N0f8, ::Array{RGB{N0f8},2}) with eltype N0f8:
[:, :, 1] =
 0.055  0.0    0.0    1.0  0.69   0.212  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.306  0.306  0.086  1.0  0.792  0.31      0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.243  0.247  0.0    1.0  0.827  0.314     0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0

[:, :, 2] =
 0.373  0.286  0.141  0.624  0.49   0.333  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.369  0.286  0.133  0.514  0.482  0.333     0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.286  0.306  0.157  0.392  0.404  0.298     0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0

[:, :, 3] =
 0.369  0.325  0.133  0.129  0.075  0.173  …  0.0  0.0  0.0  0.0  0.0    0.0  0.0
 0.337  0.29   0.165  0.122  0.098  0.169     0.0  0.0  0.0  0.0  0.016  0.0  0.0
 0.318  0.255  0.149  0.063  0.047  0.18      0.0  0.0  0.0  0.0  0.0    0.0  0.0

;;; …

[:, :, 304] =
 0.337  0.0  0.396  1.0  1.0  0.357  0.0    …  0.027  0.118  0.016  0.059  0.047
 0.227  0.0  0.384  1.0  1.0  0.541  0.133     0.173  0.216  0.2    0.153  0.239
 0.227  0.0  0.239  1.0  1.0  0.643  0.0       0.188  0.188  0.169  0.129  0.145

[:, :, 305] =
 0.369  0.0  0.337  1.0  1.0  0.424  0.0    …  0.114  0.016  0.0    0.153  0.086
 0.314  0.0  0.325  1.0  1.0  0.678  0.086     0.192  0.196  0.196  0.227  0.165
 0.298  0.0  0.204  1.0  1.0  0.745  0.0       0.125  0.11   0.133  0.176  0.145

[:, :, 306] =
 0.4    0.0  0.275  1.0  1.0  0.439  0.0    …  0.051  0.067  0.11   0.094  0.129
 0.255  0.0  0.306  1.0  1.0  0.659  0.075     0.18   0.208  0.18   0.192  0.263
 0.29   0.0  0.137  1.0  1.0  0.765  0.012     0.176  0.157  0.192  0.176  0.247
```

```
1 channelview(img)
```

Selection deleted

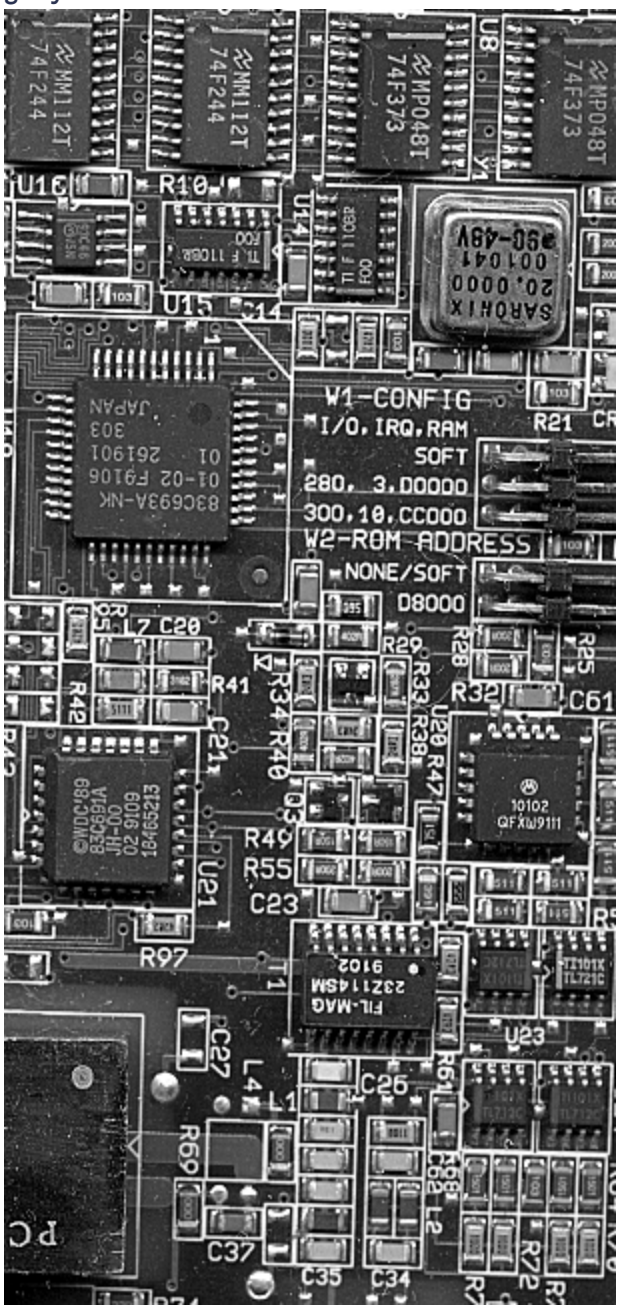
grayPic1 =




```
grayPic1 = Gray.(img)
```

Selection deleted

grayPic =




```
grayPic = float32.(grayPic1)
```

Selection deleted

```
mat =
648×306 reinterpret(reshape, Float32, ::Array{Gray{Float32},2}) with eltype Float32:
 0.223529   0.360784  0.345098    0.329412  …  0.258824   0.329412   0.301961
 0.207843   0.290196  0.298039    0.352941     0.0        0.0        0.0
 0.0509804  0.137255  0.152941    0.262745     0.372549   0.313726   0.278431
 1.0        0.533333  0.117647    0.184314     1.0        1.0        1.0
 0.764706   0.47451   0.0862745   0.34902      1.0        1.0        1.0
 0.282353   0.329412  0.172549    0.258824  …  0.498039   0.611765   0.603922
 0.0        0.211765  0.203922    0.337255     0.0784314  0.0509804  0.0431373
 ⋮                                          ⋱                        ⋮
 0.0        0.0       0.0         0.0          0.117647   0.152941   0.164706
 0.0        0.0       0.0         0.0          0.129412   0.160784   0.141176
 0.0        0.0       0.0         0.0          0.184314   0.133333   0.160784
 0.0        0.0       0.00784314  0.0       …  0.141176   0.129412   0.160784
 0.0        0.0       0.0         0.0          0.121569   0.2        0.160784
 0.0        0.0       0.0         0.0          0.172549   0.137255   0.219608
```

```
1 mat = channelview(grayPic) # size of this matrix is 648x306=198288
```

```
 -0.050351   -0.00533432    0.117585        0.0114939    0.0713969  -0.0157048
 -0.0462733  -0.0220276     0.0354032      -0.00711463  -0.0446527   0.00631449
 -0.0397382  -0.0380507    -0.0116156   …  -0.0297316    0.0263422  -0.0208641
 -0.0359115   0.000304513   0.00588178      0.0606513   -0.0104282   0.0124549
  ⋮                                      ⋱                          ⋮
 -0.0255683   0.0257879     0.00622675     -0.0267657    0.0424737  -0.0192875
 -0.0267084   0.036828     -0.0038444      -0.0255819    0.015422   -0.0415904
 -0.0285282   0.0290763    -0.0209823       0.0124278   -0.0273652   0.0332386
 -0.0271371   0.0166118    -0.0188462   …  -0.0493506    0.0420988  -0.0105545
 -0.0279977   0.0136782     0.0153778      -0.0649911    0.0534091  -0.00303688
 -0.0282837   0.00624708    0.0274959      -0.0158944    0.0133213  -0.0277772
singular values:
306-element Vector{Float32}:
 167.28459
  32.449837
  31.012547
  24.602507
  23.108452
  22.032993
  21.819658
   ⋮
   0.82532257
   0.81449
   0.7968628
   0.769987
   0.7526094
   0.70835143
Vt factor:
306×306 Matrix{Float32}:
 -0.0538312   -0.0470396   -0.0443013   …  -0.058469   -0.0536723  -0.0585421
 -0.0729599   -0.0527298   -0.0543032       0.0429203   0.0104032   0.0173915
 -0.0283877   -0.0166323   -0.0266195      -0.0165464   0.0029313   0.0102202
  0.0575245    0.0753941    0.0743636       0.0862158   0.114285    0.119568
  0.0159566    0.0273878    0.0347014      -0.0954446  -0.0952867  -0.0817291
```

Selection deleted

```
svd(mat) # size to store is = (648x306)+306+(306x306) = 198288+306+93636
```

sigma =

```
306×306 Matrix{Float64}:
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 ⋮                        ⋮              ⋱           ⋮                        ⋮
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
```

```
sigma = zeros(length(S), length(S))
```

```
for i in range(1, length(S))
    sigma[i,i] = S[i]
end
```

picMatrix =

```
648×306 Matrix{Float64}:
  0.286133    0.295034  -0.0665196  -0.224569  …   0.409763    0.139262    0.125027
  0.318685    0.553288   0.151429   -0.183031      0.131384   -0.208109    0.93739
  0.764908    1.07512    0.328813   -0.55811       0.0655658  -0.430851    0.34467
  0.379551    0.459715   0.159627   -0.197196      0.463751   -0.0963181  -0.264698
  0.599884    0.925783   0.209197   -0.37249       0.703566   -0.352143   -0.0172605
  0.632335    0.892113   0.14036    -0.584801  …   0.352621    0.325064   -0.411844
  0.381108    0.548157   0.0337274  -0.207802      0.0861296   0.182831   -0.198336
  ⋮                                                                        ⋮
 -0.0561835   0.307855   0.0181286  -0.287963     -0.246682   -0.233064    0.241668
  0.00141553  0.136509   0.0796967  -0.421525     -0.0288187  -0.151497    0.525336
  0.14506     0.28036    0.217302   -0.628872      0.210542   -0.0615812   0.682473
  0.135089    0.386651   0.254256   -0.638633  …   0.322301   -0.208008    0.583466
  0.292491    0.465074   0.169691   -0.405014     -0.0263315  -0.0724359   0.469568
  0.27557     0.276138   0.252472   -0.380087     -0.148075   -0.0524043   0.0908044
```

```
picMatrix = U * sigma * VT
```

errorMatrix =

```
648×306 Matrix{Float64}:
 -0.062604     0.0657504   0.411618   0.553981  …  -0.150939    0.19015    0.176934
 -0.110842    -0.263092    0.146611   0.535972     -0.131384    0.208109  -0.93739
 -0.713927    -0.937861   -0.175872   0.820856      0.306983    0.744576  -0.0662387
  0.620449     0.0736183  -0.0419798  0.38151       0.536249    1.09632    1.2647
  0.164822    -0.451273   -0.122923   0.72151       0.296434    1.35214    1.01726
 -0.349982    -0.562702    0.0321895  0.843624  …   0.145418    0.286701   1.01577
 -0.381108    -0.336393    0.170194   0.545057     -0.0076982  -0.131851   0.241473
  ⋮                                                                        ⋮
  0.0561835   -0.307855   -0.0181286  0.287963      0.364329    0.386005  -0.0769625
 -0.00141553  -0.136509   -0.0796967  0.421525      0.158231    0.312281  -0.384159
 -0.14506     -0.28036    -0.217302   0.628872     -0.0262285   0.194915  -0.521689
 -0.135089    -0.386651   -0.246412   0.638633  …  -0.181125    0.33742   -0.422682
 -0.292491    -0.465074   -0.169691   0.405014      0.1479      0.272436  -0.308784
 -0.27557     -0.276138   -0.252472   0.380087      0.320624    0.189659   0.128803
```

```
# this is supposed to be the zero matrix if the original picture matrix equals the
reconstructed SVD matrix

errorMatrix = mat - picMatrix
```

calculate_svd_memory (generic function with 1 method)

```
function calculate_svd_memory(U, S, VT)
    U_memory = size(U, 1) * size(U, 2) * sizeof(eltype(U))
    S_memory = length(S) * sizeof(eltype(S))
    VT_memory = size(VT, 1) * size(VT, 2) * sizeof(eltype(VT))
    total_memory = U_memory + S_memory + VT_memory
    return total_memory
end
```

load_grayscale_image (generic function with 1 method)

```
function load_grayscale_image(path)
    img = load(path)
    img_gray = Gray.(img)
    return img_gray
end
```

svd_compression (generic function with 1 method)

```
function svd_compression(image, k)
    U, S, V = svd(Float64.(image))
    img_compressed = U[:, 1:k] * Diagonal(S[1:k]) * V[:, 1:k]'
    return img_compressed
end
```

save_compressed_image (generic function with 1 method)

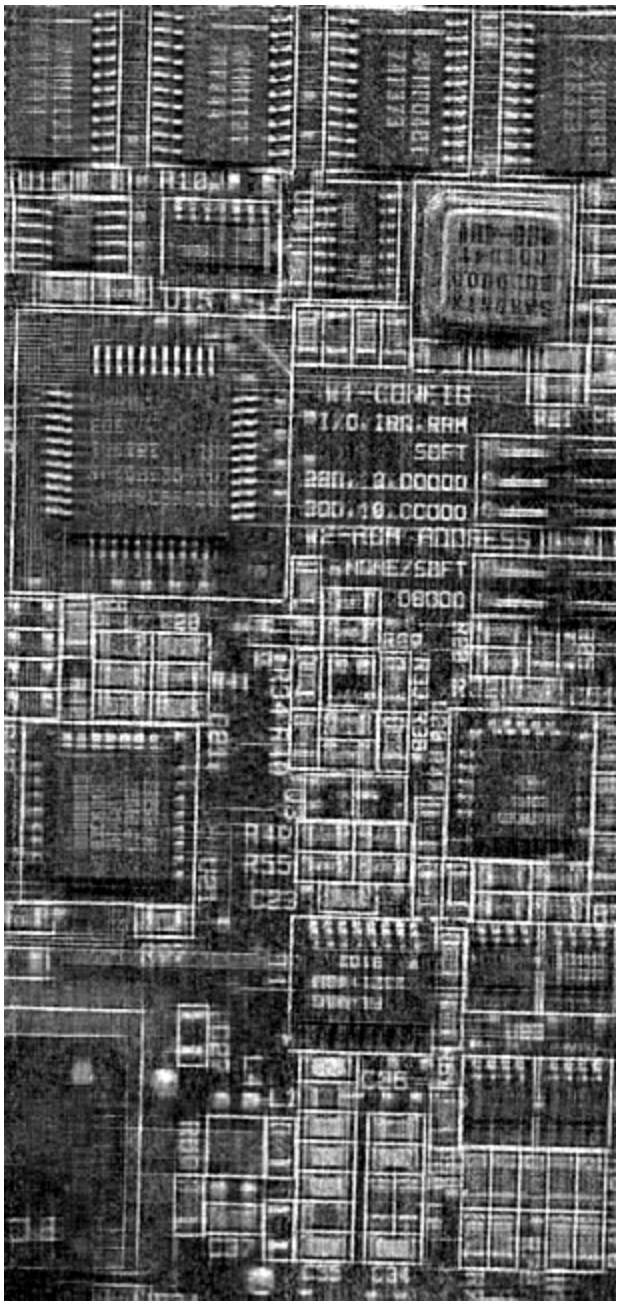
```
function save_compressed_image(compressed_img, path)
    println("Data type and size of the image being saved: ", typeof(compressed_img), " ", size(compressed_img))
    save(path, Gray.(compressed_img))
end
```

compress_image (generic function with 1 method)

```
function compress_image(image_path, k, save_path)
    img_gray = load_grayscale_image(image_path)
    compressed_img = svd_compression(img_gray, k)
    save_compressed_image(compressed_img, save_path)
    println("Image compression completed. Compressed image saved to: ", save_path)
end
```

```
compress_image("board.tif", 50, "board2.tif")
```

```
Data type and size of the image being saved: Matrix{Float64} (648, 306)
Image compression completed. Compressed image saved to: board2.tif
```




```
load("board2.tif") # (648x50) + 50 + (50x306) =
```

chroma_subsampling_420 (generic function with 1 method)

```
 #might continue this later for the final writeup
function chroma_subsampling_420(image_path::String)
    img = load(image_path)
    img_ycbcr = convert.(YCbCr{Float32}, float.(img))

    Y = Array{Float32}(undef, size(img_ycbcr))
    Cb = Array{Float32}(undef, size(img_ycbcr))
    Cr = Array{Float32}(undef, size(img_ycbcr))

    for j in 1:size(img_ycbcr, 2)
        for i in 1:size(img_ycbcr, 1)
            Y[i, j] = img_ycbcr[i, j].y
            Cb[i, j] = img_ycbcr[i, j].cb
            Cr[i, j] = img_ycbcr[i, j].cr
        end
    end

    Cb_subsampled = Cb[1:2:end, 1:2:end]
    Cr_subsampled = Cr[1:2:end, 1:2:end]

    return Y, Cb_subsampled, Cr_subsampled
end
```

```
, 324×153 Matrix{Float32}:                                                , 3
2058   130.459   124.618   126.249   123.591   119.238     124.344   131.794   122.521   124.168
```

```
Y, Cb, Cr = chroma_subsampling_420("board.tif")
```

plot_singular_values (generic function with 1 method)

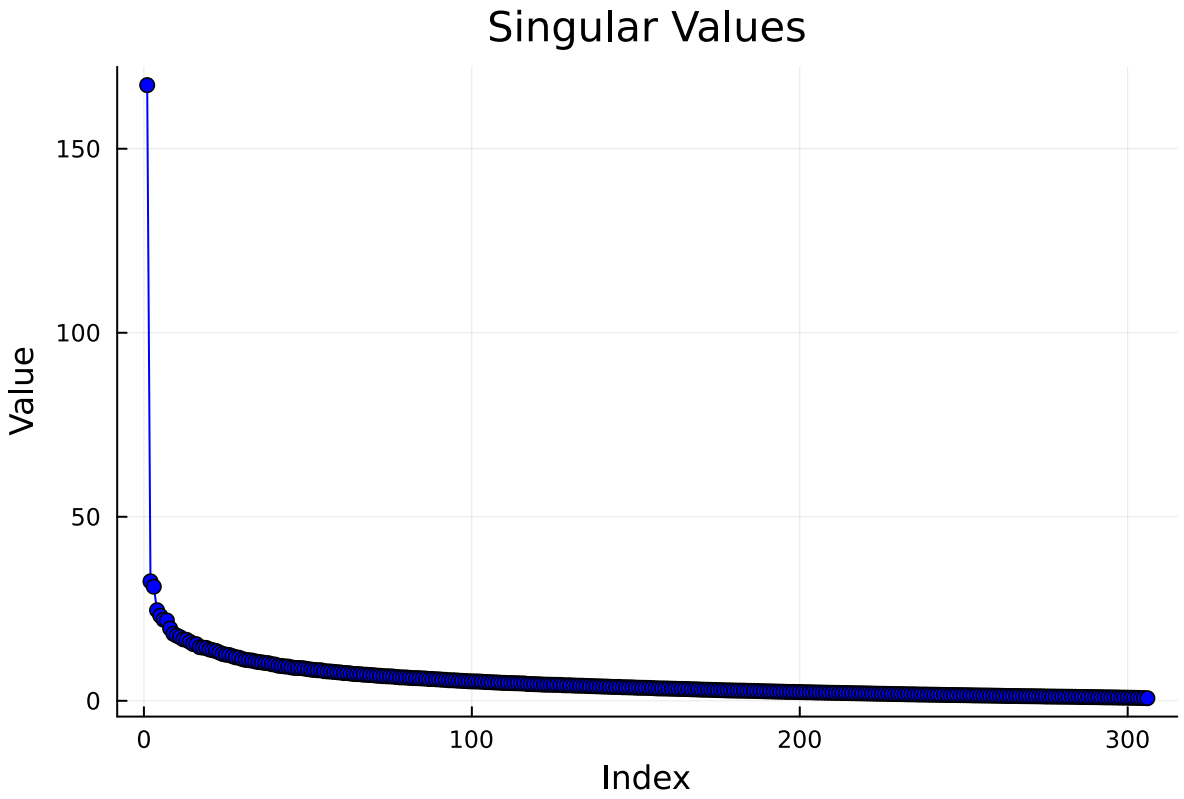
```
function plot_singular_values(S)
    plot(S, title="Singular Values", xlabel="Index", ylabel="Value",
         legend=false, markershape=:circle, color=:blue)
end
```

plot_cumulative_singular_values (generic function with 1 method)

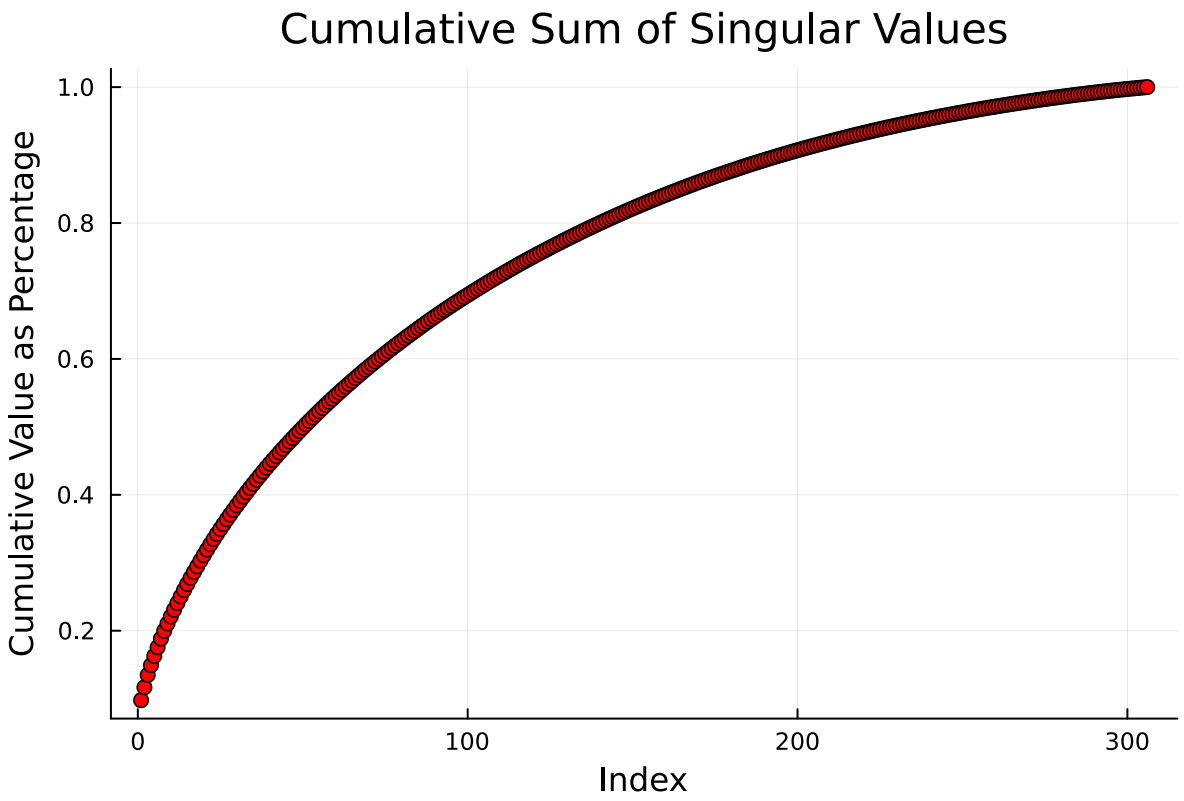
```
function plot_cumulative_singular_values(S)
    cumulative_sum = cumsum(S)
    plot(cumulative_sum/cumsum(S)[length(S)], title="Cumulative Sum of Singular Values", xlabel="Index", ylabel="Cumulative Value as Percentage",
         legend=false, markershape=:circle, color=:red)
end
```

```
plot_ssim_vs_singular_values (generic function with 1 method)
```

```
# Want to visualize structural similarity based on # of singular values to see the relationship
# Can't get this to work
function plot_ssim_vs_singular_values(original_path, compressed_path)
    original_matrix = channelview(original_path)
    compressed_matrix = channelview(compressed_path)

    U, S, Vt = svd(compressed_matrix)

    ssim_values = Float64[]
    max_singular_values = length(S)
    step_size = 10

    for k in 1:step_size:max_singular_values
        reconstructed_matrix = U[:, 1:k] * Diagonal(S[1:k]) * Vt[1:k, :]
        reconstructed_img = clamp.(colorview(Gray, reconstructed_matrix), 0, 1)
        ssim_score = assess_ssim(original_path, compressed_path)
        push!(ssim_values, ssim_score)
    end

    plot(1:step_size:max_singular_values, ssim_values, title="SSIM vs. Number of Singular Values",
         xlabel="Number of Singular Values", ylabel="SSIM", ylim=(0, 1), legend=false,
         markershape=:circle, markersize=8, linewidth=2, linecolor=:blue, grid=true)

    display(plot)
end
```




```
plot_singular_values(S)
```




```
1 plot_cumulative_singular_values(S)
```

```
1 plot_ssim_vs_singular_values(Gray.(load("board.tif")), Gray.(load("board2.tif")))
```

```
plot (generic function with 4 methods)
```

image =

```
# Wavelet Transform

image = testimage("cameraman")
```

gray_image =

```
gray_image = Gray.(image)

# Select a Daubechies wavelet. The argument "4" denotes the number of vanishing moments (db4)
```

cameraman =

512×512 reinterpret(reshape, N0f8, ::Array{Gray{N0f8},2}) with eltype N0f8:

0.612	0.616	0.627	0.624	0.62	0.612	…	0.588	0.592	0.596	0.596	0.596
0.612	0.616	0.624	0.62	0.62	0.612		0.588	0.592	0.596	0.596	0.596
0.62	0.616	0.612	0.612	0.616	0.616		0.596	0.6	0.596	0.596	0.596
0.627	0.616	0.604	0.604	0.612	0.616		0.604	0.608	0.6	0.596	0.596
0.62	0.616	0.612	0.612	0.616	0.612		0.596	0.6	0.6	0.596	0.596
0.612	0.616	0.624	0.624	0.624	0.612	…	0.592	0.596	0.6	0.6	0.6
0.62	0.616	0.612	0.612	0.616	0.612		0.596	0.6	0.6	0.6	0.6
⋮					⋮	⋱				⋮	
0.475	0.482	0.49	0.49	0.482	0.451		0.525	0.518	0.486	0.459	0.451
0.447	0.486	0.522	0.51	0.478	0.431		0.533	0.541	0.49	0.443	0.431
0.455	0.482	0.51	0.502	0.494	0.506		0.525	0.525	0.482	0.439	0.427
0.475	0.482	0.494	0.494	0.514	0.588		0.522	0.51	0.475	0.443	0.435
0.475	0.482	0.494	0.494	0.514	0.596	…	0.522	0.51	0.475	0.443	0.435
0.475	0.482	0.494	0.494	0.51	0.588		0.522	0.51	0.475	0.443	0.435

```
cameraman = channelview(gray_image)
```

signal =

[0.612, 0.612, 0.62, 0.627, 0.62, 0.612, 0.62, 0.627, 0.62, 0.612, 0.608, 0.608, 0.608, 0.6

```
signal = cameraman[:, 1]
```

[-0.482963, 0.836516, -0.224144, -0.12941]

```
begin
    # Daubechies D4 wavelet coefficients
    h0 = (1 + sqrt(3)) / (4 * sqrt(2))
    h1 = (3 + sqrt(3)) / (4 * sqrt(2))
    h2 = (3 - sqrt(3)) / (4 * sqrt(2))
    h3 = (1 - sqrt(3)) / (4 * sqrt(2))

    # Decomposition low-pass and high-pass filters
    dec_lo = [h0, h1, h2, h3]
    dec_hi = [h3, -h2, h1, -h0]

    rec_lo = reverse(dec_lo)
    rec_hi = [-h0, h1, -h2, h3]
end
```

wavelet_transform (generic function with 1 method)

```
function wavelet_transform(signal, filter_lo, filter_hi)
    # Apply convolution with the low-pass and high-pass filters
    low_passed = conv(signal, filter_lo)[2:end-1]  # Convolution and remove padding
    high_passed = conv(signal, filter_hi)[2:end-1] # Same for high-pass

    # Downsampling
    downsampled_low = low_passed[1:2:end]
    downsampled_high = high_passed[1:2:end]

    return downsampled_low, downsampled_high
end
```

```
begin
    # Apply the wavelet transform
    approx_coeffs, detail_coeffs = wavelet_transform(cameraman, dec_lo, dec_hi)

    println("Approximation Coefficients: ", approx_coeffs)
    println("Detail Coefficients: ", detail_coeffs)
end
```

```
Approximation Coefficients: [0.8072108455674795, 0.8793027769429989, 0.874
2278940081812, 0.8782878003560355, 0.8742278940081812, 0.8589765721857563, 0.
8615140136531652, 0.8746358372434013, 0.8776443309841468, 0.8640150191004291,
0.8487636972780042, 0.8452477152437767, 0.8631724596120172, 0.847241232397558
8, 0.8524520964107833, 0.871862869639324, 0.8584690838922746, 0.8413237899743
292, 0.8407167566225859, 0.8443687197352198, 0.8467701801242221, 0.8569830550
319743, 0.8408163016808475, 0.8275949330323497, 0.8413602259944742, 0.8514371
198238196, 0.8847491670928916, 0.8853197644244902, 0.880045791373149, 0.88545
57455028969, 0.8578256145203861, 0.8432177620698497, 0.8467701801242221, 0.85
69830550319743, 0.8742643300283262, 0.8686819588000267, 0.8604990370662017,
0.8482562089845224, 0.8481566639262608, 0.8823477067038896, 0.880281317509817
5, 0.8645225073939109, 0.8618855208682402, 0.8849846932295602, 0.865637029039
1361, 0.8592485343425698, 0.8801817724515557, 0.8964480708609442, 0.878622871
5509654, 0.8744998561649946, 0.9011150105605419, 0.8903582113391629, 0.879402
3220012606, 0.9043954664581008, 0.9024650583424355, 0.9060174763968079, 0.901
9575700489537, 0.8867062482265288, 0.8770008616122584, 0.8852202193662285, 0.
8958046014890558, 0.8867062482265287, 0.878015838199222, 0.8823112706837445,
0.8818037823902628, 0.8785233264927038, 0.8912736428678648, 0.906524964690289
8, 0.9048665187314376, 0.8936386672367219, 0.8947896249020922, 0.893774648315
1285, 0.8966835969976125, 0.9139648719939646, 0.9093974773526285, 0.900199579
03184, 0.9063889836118829, 0.8926236906497584, 0.8958046014890557, 0.90689647
19053647, 0.9029725466359172, 0.8979340997212445, 0.8880927320285674, 0.89631
20897825374, 0.9120708998984441, 0.9014500817554719, 0.8815318202334493, 0.88
36977544857831, 0.8826827778988195, 0.8959405825674623, 0.8979340997212445,
0.9032445087927307, 0.916601858519635, 0.9028365655575107, 0.878622871550965
4, 0.8664164794894312, 0.8731133723629559, 0.8838337355641898, 0.870439949817
1403, 0.8806892607450375, 0.8925241455914966, 0.8745994012232562, 0.884341223
8576717, 0.8764933733187766, 0.886235195953192, 0.8947896249020922, 0.8877212
248134924, 0.8821752896053378, 0.8766293543971833, 0.8729773912845492, 0.8857
277076597102, 0.9009790294821352, 0.8903582113391629, 0.8581971217354613, 0.8
908025905945279, 0.9091619512159601, 0.8922521834346834, 0.9070324529837714,
```

inverse_wavelet_transform (generic function with 1 method)

```
function inverse_wavelet_transform(approx_coeffs, detail_coeffs, rec_lo, rec_hi)

    upsampled_low = zeros(2 * length(approx_coeffs))
    upsampled_low[1:2:end] .= approx_coeffs

    upsampled_high = zeros(2 * length(detail_coeffs))
    upsampled_high[1:2:end] .= detail_coeffs

    recon_low = conv(upsampled_low, rec_lo)[3:end-2]
    recon_high = conv(upsampled_high, rec_hi)[3:end-2]

    reconstructed_signal = recon_low + recon_high

    return reconstructed_signal
end
```

```
Float64[
    1:  0.611765
    2:  0.611765
    3:  0.619608
    4:  0.627451
    5:  0.619608
    6:  0.611765
    7:  0.619608
    8:  0.627451
    9:  0.619608
    10:  0.611765
    11:  0.607843
    12:  0.607843
    13:  0.607843
    14:  0.611765
    15:  0.615686
    16:  0.623529
    17:  0.619608
    18:  0.619608
    19:  0.611765
    20:  0.607843
      more
263668:  0.403922
263669:  0.411765
263670:  0.439216
263671:  0.466667
263672:  0.45098
263673:  0.431373
263674:  0.427451
263675:  0.435294
263676:  0.435294
263677:  0.326471
]
```

```
1 inverse_wavelet_transform(approx_coeffs, detail_coeffs, rec_lo, rec_hi)
```