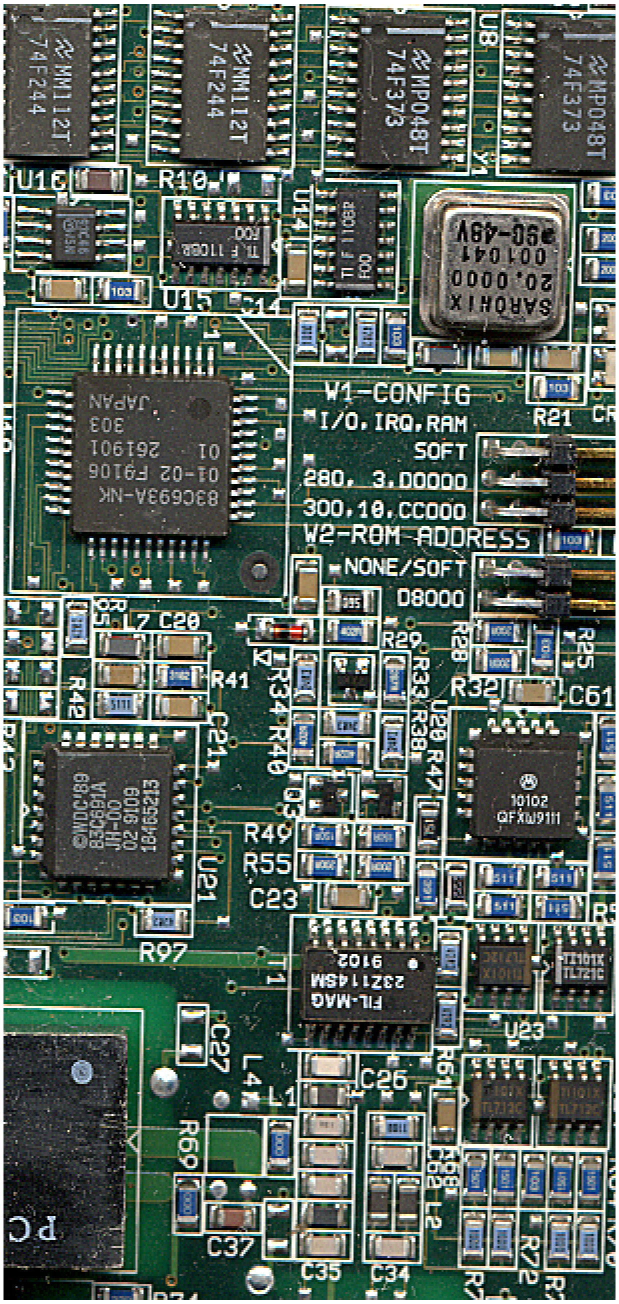
```
# 590 Final Project Matthew Soham
```

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

img =

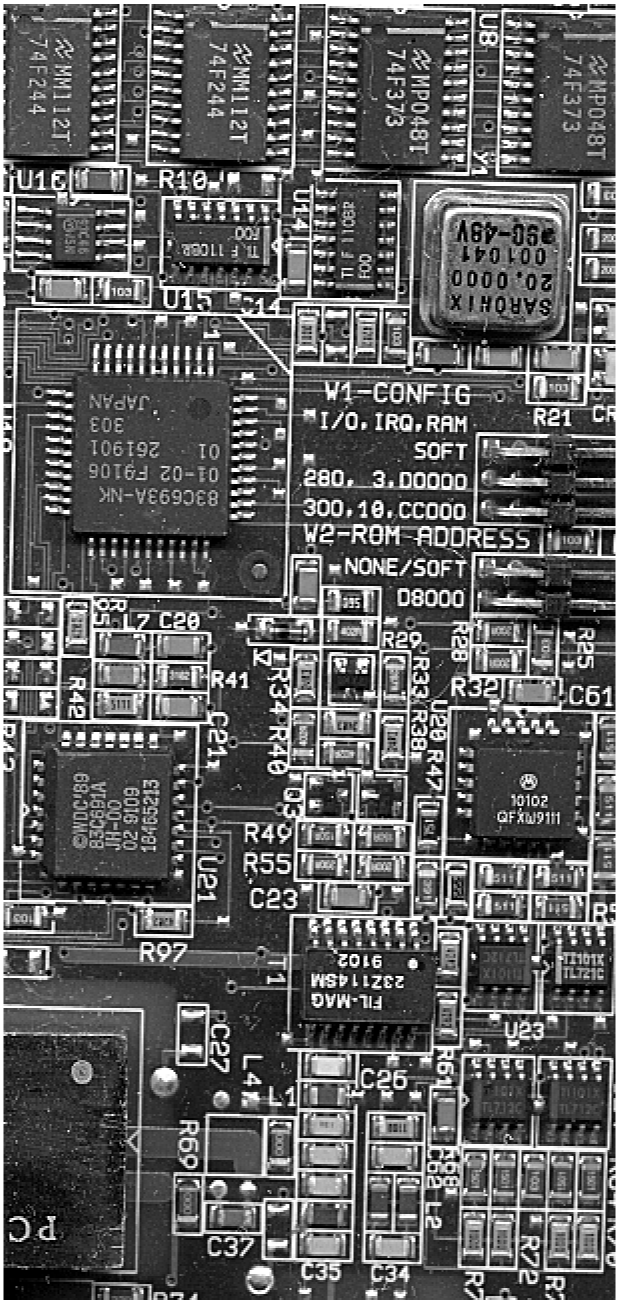


```
img = load("board.tif")
```

```
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
```

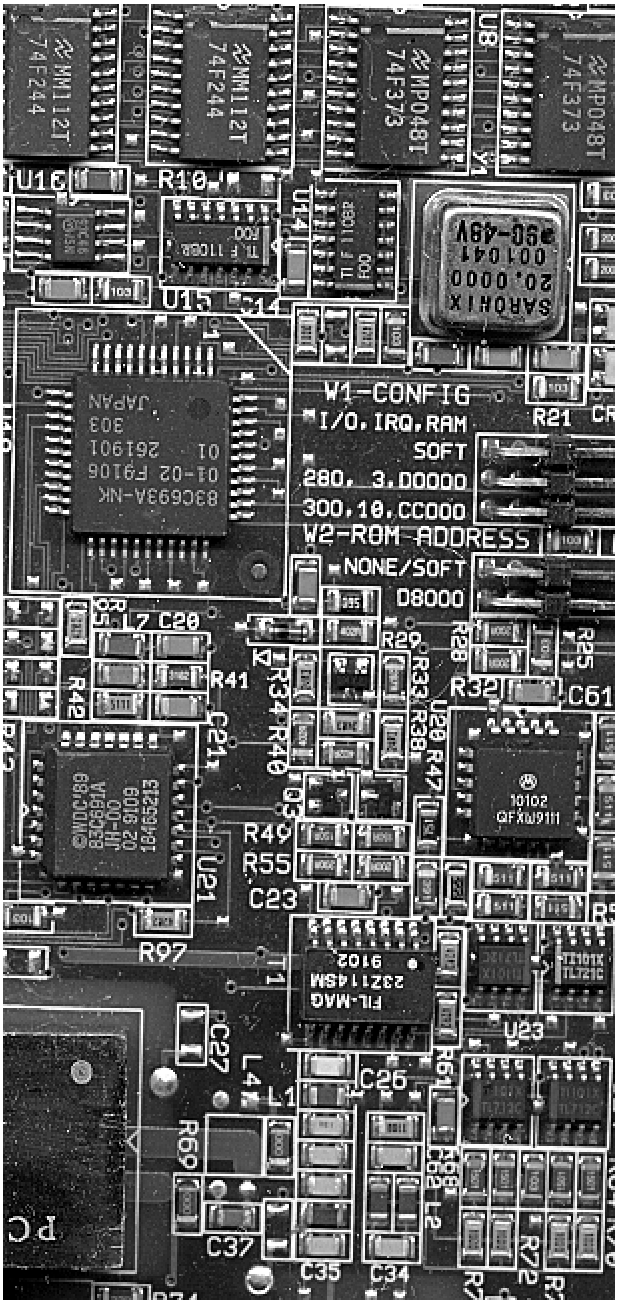
```julia
channelview(img)
```

grayPic1 =



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

```
grayPic =
```



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

```
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
```

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

```
SVD{Float32, Float32, Matrix{Float32}, Vector{Float32}}
U factor:
648×306 Matrix{Float32}:
 -0.0500886   0.0592246    -0.0788847    …  -0.0815515   -0.0108612   0.0262747
 -0.0427833   0.0557872    -0.105956        -0.0386432    0.0367617   0.0221914
 -0.0438362   0.00237994   -0.000519294      0.0357935   -0.0174905  -0.0229565
 -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
```

```
U, S, VT = 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
```

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

```julia
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
```

```julia
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
```

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

errorMatrix = mat - picMatrix
```

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

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

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

```julia
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)
```

```julia
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)
```

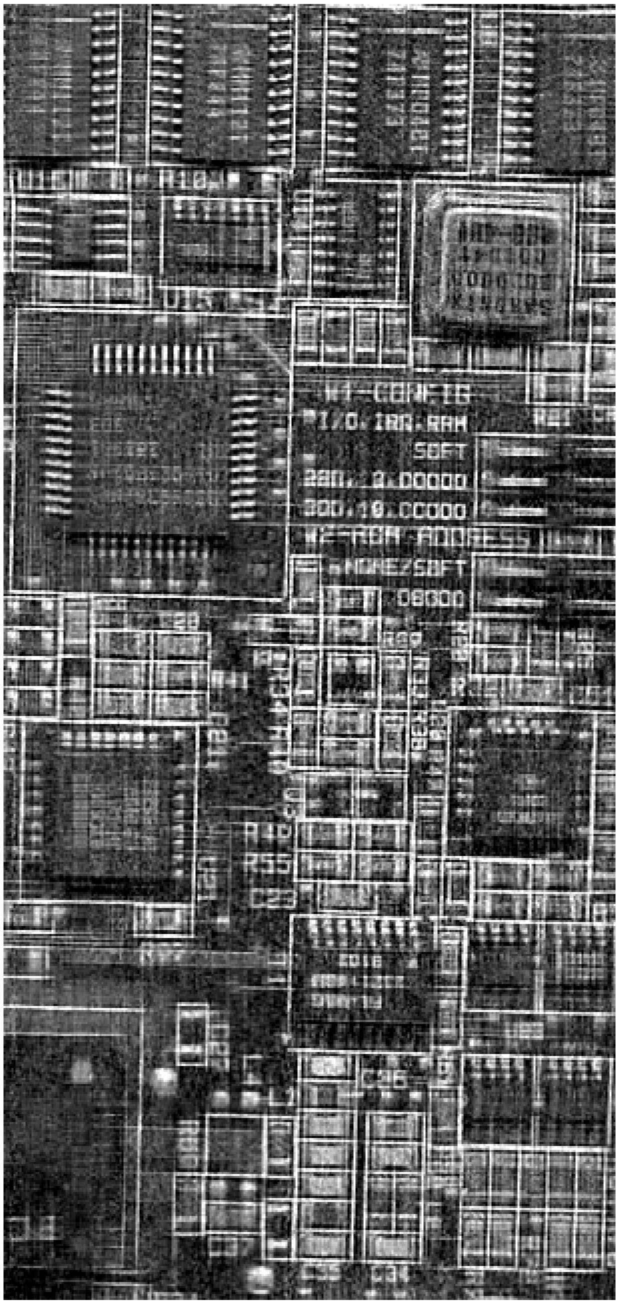
```julia
function compress_image(img_gray, k, save_path)
    #img_gray = load_grayscale_image(image_path)
    compressed_img = svd_compression(img_gray, k)
    save_compressed_image(compressed_img, save_path)
    println("Image compression completed. Compressed image saved to: ", save_path)
end
```

```
compress_image2 (generic function with 1 method)
```

```julia
function compress_image2(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
```

```julia
compress_image2("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
```



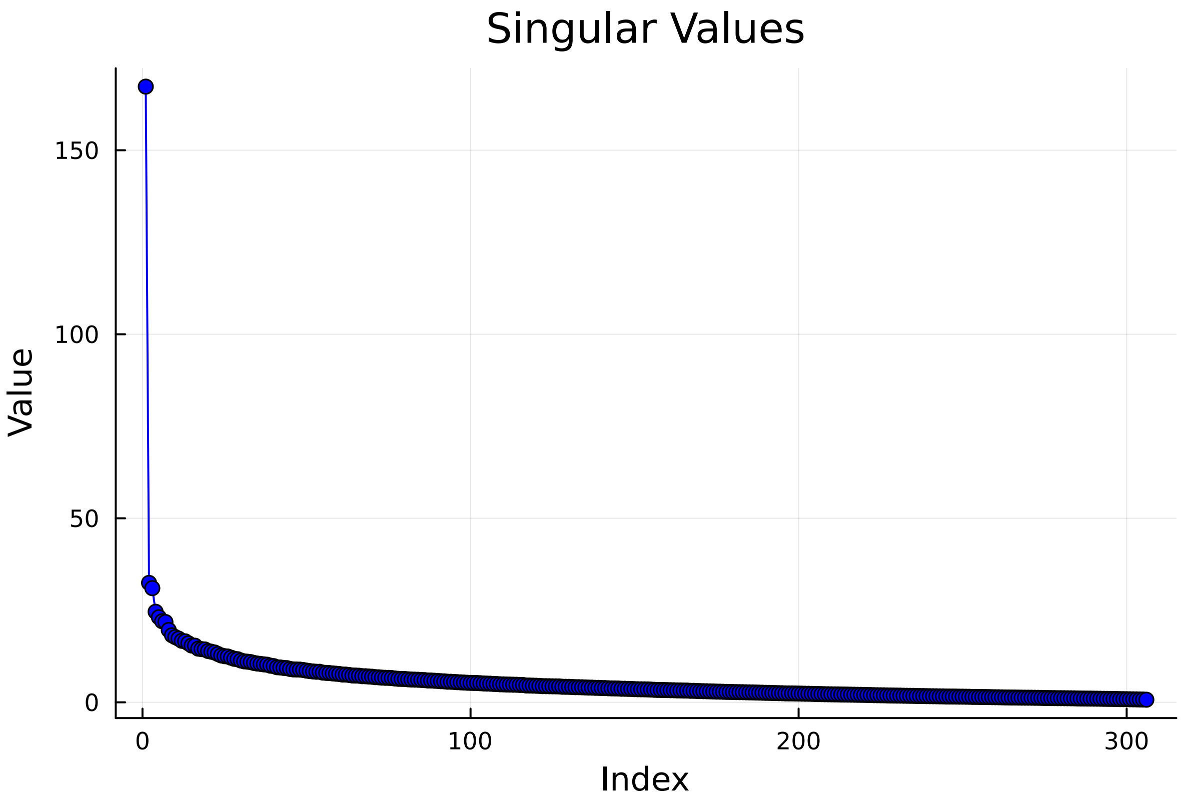
```julia
load("board2.tif") # (648x50) + 50 + (50x306) =
```

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

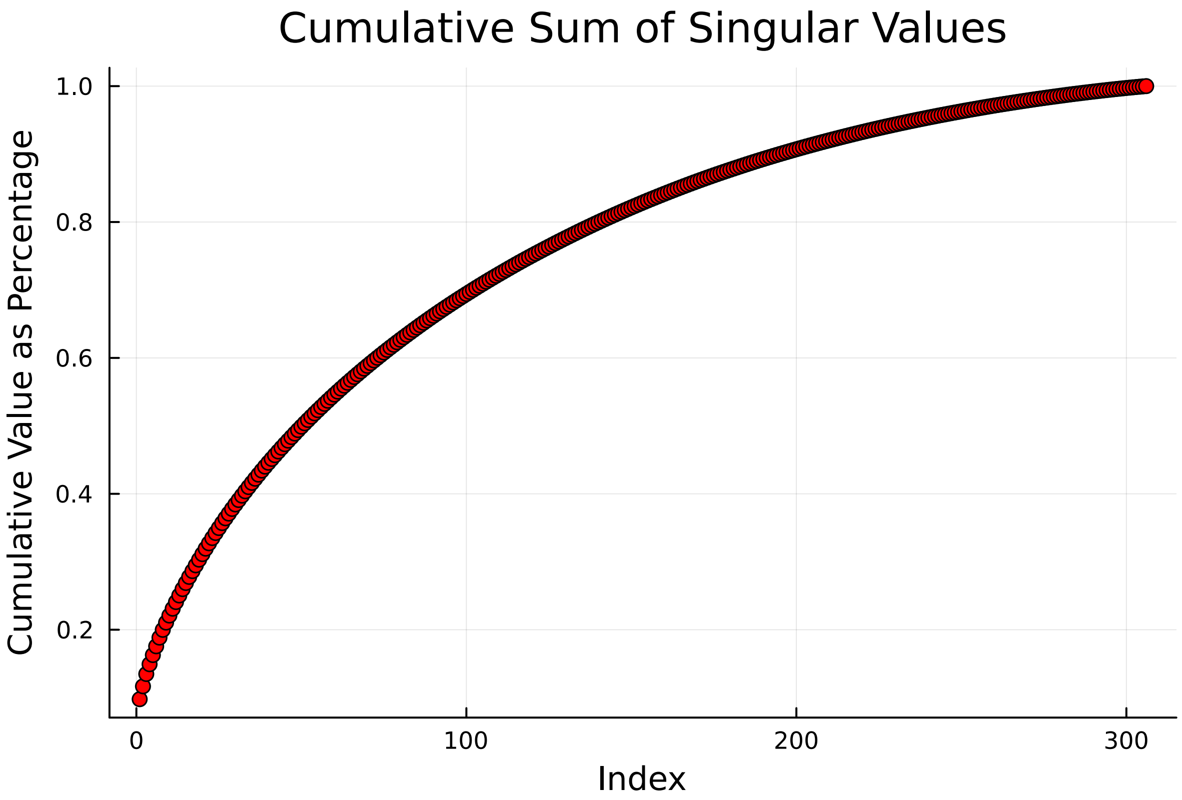
```julia
function plot_singular_values(S)
    plot(S, title="Singular Values", xlabel="Index", ylabel="Value",
         legend=false, markershape=:circle, color=:blue)
end
```

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

```julia
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_singular_values(S)
```



```
plot_cumulative_singular_values(S)
```

```
# https://gregorygundersen.com/blog/2019/01/17/randomized-svd/#phillips1998feret
```

```
pm = 512×512 Matrix{Float64}:
      0.65098     0.494118  0.529412  0.560784  …  0.647059  0.517647  0.576471  0.686275
      0.639216    0.501961  0.454902  0.556863     0.662745  0.47451   0.619608  0.694118
      0.556863    0.580392  0.466667  0.494118     0.670588  0.466667  0.513725  0.568627
      0.568627    0.423529  0.501961  0.494118     0.619608  0.54902   0.560784  0.596078
      0.584314    0.533333  0.627451  0.6          0.545098  0.411765  0.529412  0.596078
      0.607843    0.545098  0.45098   0.388235  …  0.486275  0.501961  0.533333  0.596078
      0.623529    0.482353  0.458824  0.356863     0.545098  0.588235  0.603922  0.639216
      ⋮                                         ⋱                      ⋮
      0.0980392   0.603922  0.560784  0.564706     0.713725  0.690196  0.603922  0.27451
      0.0862745   0.619608  0.545098  0.615686     0.662745  0.647059  0.54902   0.317647
      0.0901961   0.627451  0.552941  0.701961     0.698039  0.694118  0.572549  0.333333
      0.105882    0.635294  0.498039  0.670588     0.741176  0.709804  0.568627  0.333333
      0.0         0.0       0.0       0.0       …  0.0       0.0       0.0       0.0
      0.0         0.0       0.0       0.0          0.0       0.0       0.0       0.0
```

```
pm = convert(Array{Float64}, testimage("bark_512"))
```

```
testimage("bark_512")
```

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

```julia
function rsvd(X, k)
    m, n = size(X)
    Omega = randn(n, k)
    Y = X * Omega
    Q, R = qr(Y)
    Qm = Matrix(Q)
    B = Qm' * X
    Uhat, S, Vt = svd(B)
    Uk = Uhat[:, 1:k]
    Vk = Vt'[1:k, :]
    U = Qm * Uk
    return U, Diagonal(S), Vk
end
```

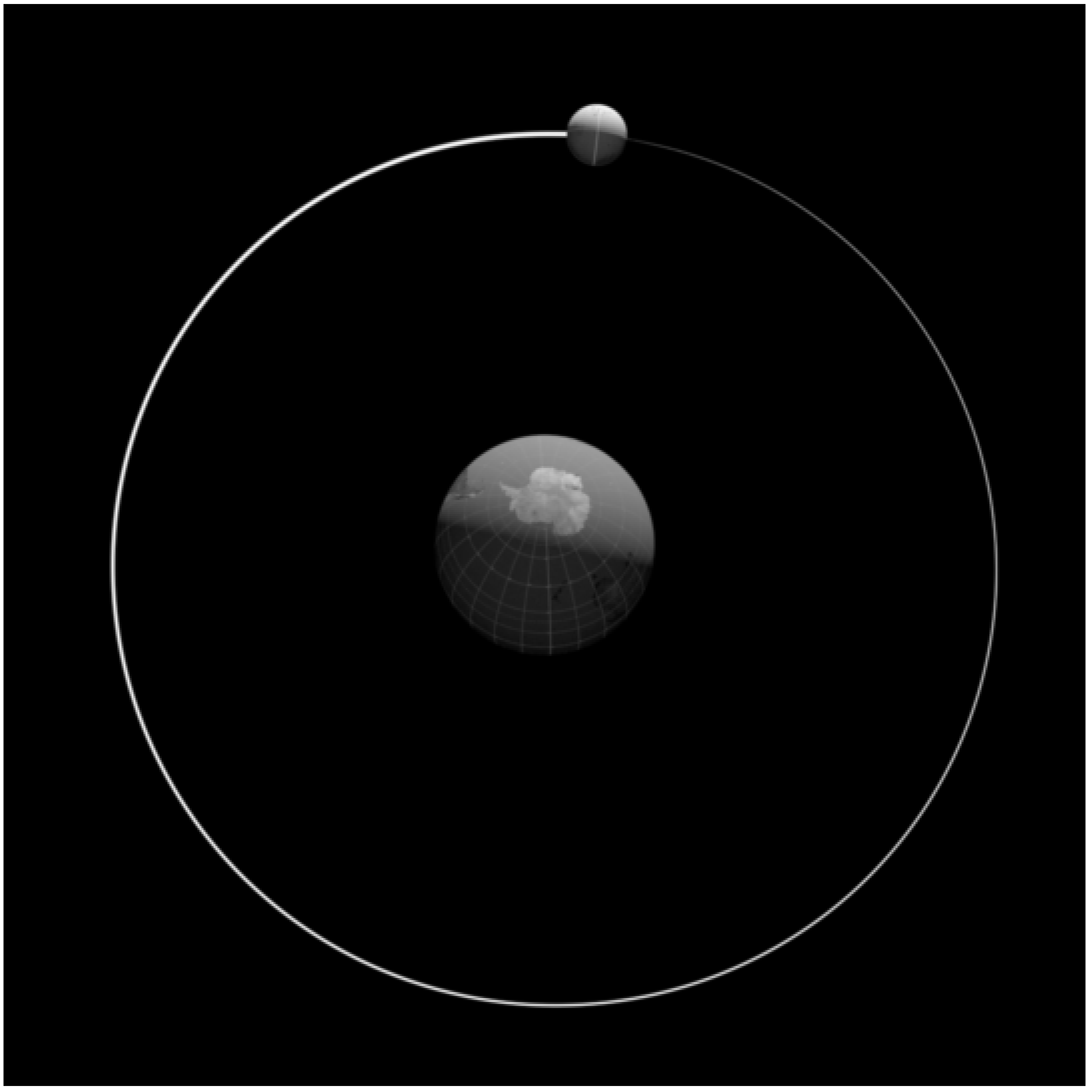
```
reconstruct_image (generic function with 1 method)
```

```julia
function reconstruct_image(P, R, N)
    return P * Diagonal(R) * N
end
```

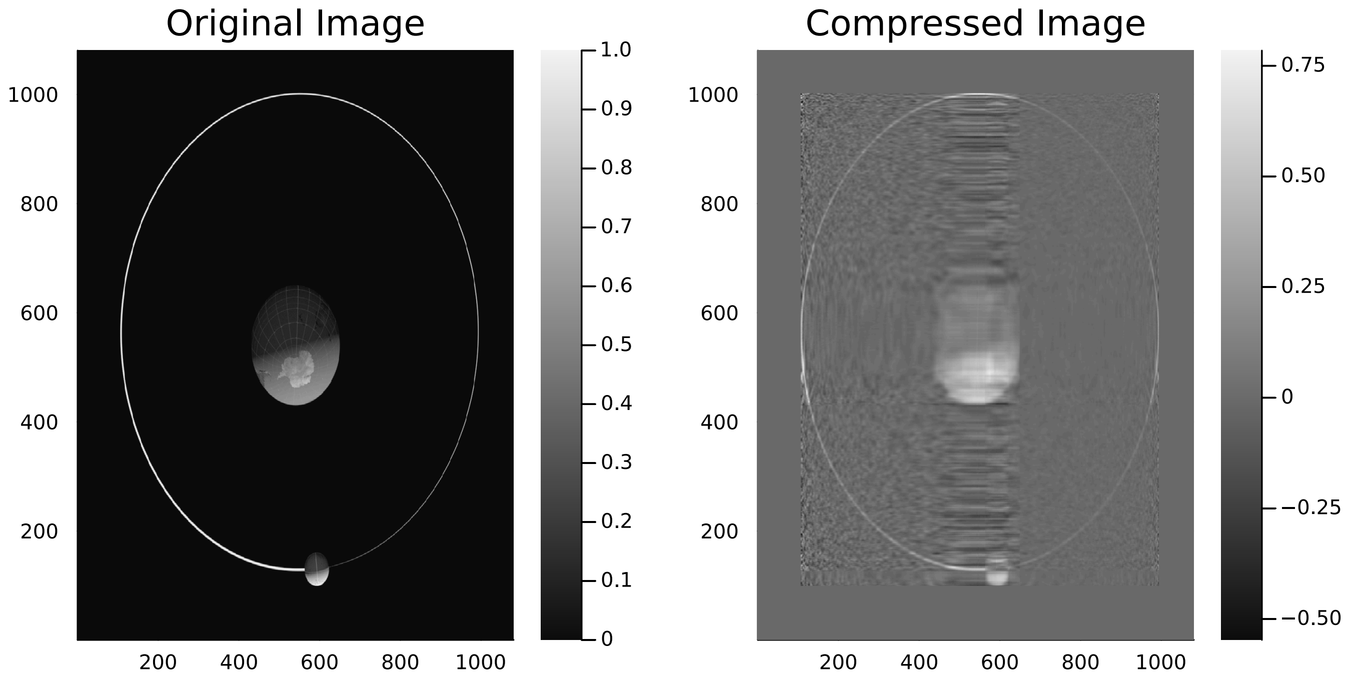
```
display_images (generic function with 1 method)
```

```julia
function display_images(original, compressed)
    p1 = heatmap(original, c=:grays, title="Original Image")
    p2 = heatmap(compressed, c=:grays, title="Compressed Image")
    plot(p1, p2, layout=(1, 2), size=(800, 400))
end
```

```
randImg =
```



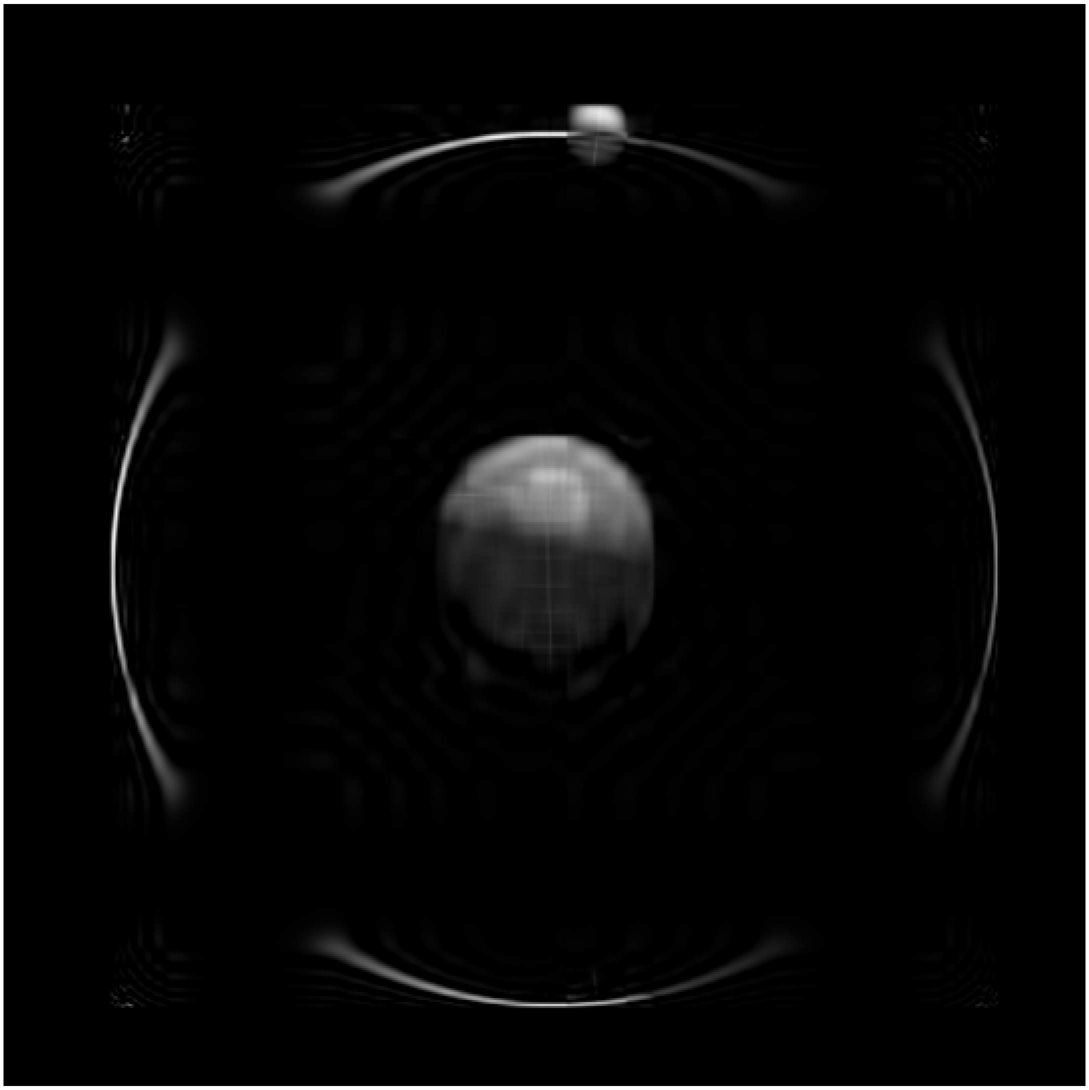
```
randImg = Gray.(load("orbit.0036.tif"))
```



```julia
begin
    # Load and preprocess the image
    img_matrix = convert(Array{Float64}, randImg)

    # Compute the randomized SVD with k ranks
    U3, S3, Vt3 = rsvd(img_matrix, 20)  # Using k=50 for example

    # Reconstruct the image using the top k components
    compressed_img = reconstruct_image(U3, S3, Vt3)
    # Display both the original and compressed image
    display_images(img_matrix, compressed_img)
end
```

```julia
compress_image(Gray.(load("orbit.0036.tif")), 20, "orbit2.tif")
```

```
Data type and size of the image being saved: Matrix{Float64} (1080, 1080)
Image compression completed. Compressed image saved to: orbit2.tif
```



```
load("orbit2.tif")
```

```
0.5398879790260523
```

```
assess_ssim(compressed_img, img_matrix)
```

```
0.8265620437367288
```

```
assess_ssim(load("orbit2.tif"), img_matrix)
```

```
e2mat =
1080×1080 Matrix{Float64}:
 -5.46406e-31   6.56944e-31  -1.1655e-31   …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
  8.23456e-31   2.80159e-32  -2.37463e-32     0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
  1.90144e-30  -7.33601e-31  -2.25658e-32     0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 -5.17413e-31  -5.21238e-31   5.16483e-31     0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
  1.18613e-30  -3.31584e-31  -7.50225e-31     0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
  3.93353e-31   2.89172e-31   2.52745e-31  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 -7.44743e-32   8.74675e-31  -1.06164e-31     0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
  ⋮                                        ⋱                 ⋮
  0.0           0.0           0.0             0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
  0.0           0.0           0.0          …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
  0.0           0.0           0.0             0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
  0.0           0.0           0.0             0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
  0.0           0.0           0.0             0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
  0.0           0.0           0.0             0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
```

```
e2mat = img_matrix - compressed_img
```

```
e3mat =
1080×1080 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
```

```
e3mat = img_matrix - convert(Array{Float64}, Gray.(load("orbit2.tif")))
```