```
% Name: Jack Goldrick
% Date: 7/2/23

%% Housekeeping

clc
clear
close all;

%% Import Images datasets

X = imread('square.jpg');
Y = imread('squareedit100.jpg');

%figure(3)
%mshow(X);

%figure(4)
%imshow(Y);

Xg = rgb2gray(X);
Yg = rgb2gray(Y);

Xgray = double(Xg)/255;
Ygray = double(Yg)/255;





[Xm,Xn] = size(Xgray);
[Ym,Yn] = size(Ygray);


Xg_n120 = Xgray-120;
Yg_n120 = Ygray-120;


%{
figure(5)
imagesc(Xgray);
figure(6)
imagesc(Ygray);
%}

%figure(3)
%imshow(Xgray);
%figure(4)
%imshow(Ygray);

if (Xm ~= Xn && Ym ~= Yn)
```

```
    Xds = transpose(Xgray)*(Xgray);
    Yds = transpose(Ygray)*(Ygray);

    [Xds_v,Xds_e] = eig(X_ds,"matrix");
    [Yds_v,Yds_e] = eig(Y_ds,"matrix");





else

    [Xgray_v,Xgray_e] = eig(Xgray,"matrix");
    [Ygray_v,Ygray_e] = eig(Ygray,"matrix");
    CompXY = abs(Xgray - Ygray);

    %% gram matrix
    Xgram = transpose(Xgray)*(Xgray);
    Ygram = transpose(Ygray)*(Ygray);
    CompGram = transpose(CompXY) * CompXY;
    %% inner product comparison

    XYin = transpose(Xgray)*(Ygray);


    %% Eigen the shit out of gram

    [XYin_v,XYin_e] = eig(XYin,"matrix");
    [Xgram_v,Xgram_e] = eig(Xgram,"matrix");
    [Ygram_v,Ygram_e] = eig(Ygram,"matrix");
    [CompGram_v,CompGram_e] = eig(CompGram,"matrix");

end
%% Values

ResXYX_e = abs(XYin_e - Xgram_e);
ResXYY_e = abs(XYin_e - Ygram_e);
DiffG_e = abs (Xgram_e - Ygram_e);




%% Vectors

ResXYX_v = abs(XYin_v - Xgram_v);
ResXYY_v = abs(XYin_v - Ygram_v);
ResXY_v = abs(Xgram_v - Ygram_v);

%% RMS Average
rms_XG_e = sqrt(trace(Xgram_e* Xgram_e)/rank(Xgram_e));
```

```
rms_YG_e = sqrt(trace(Ygram_e * Ygram_e)/rank(Ygram_e));
rms_XYD_e = sqrt(trace(DiffG_e * DiffG_e)/rank(DiffG_e));
rms_IN_e = sqrt(trace(XYin_e * XYin_e)/rank(XYin_e));
rms_CompGram_e = sqrt(trace(CompGram_e * CompGram_e)/rank(CompGram_e));

% Variance
Xgram_ed = abs(Xgram_e - rms_XG_e);
Ygram_ed = abs(Ygram_e - rms_YG_e);
DiffG_ed = abs(DiffG_e - rms_XYD_e);
XYin_ed = abs(XYin_e - rms_IN_e);
CompGram_ed = abs(CompGram_e - rms_CompGram_e);

rxged = rank(Xgram_ed);
rxge = rank(Xgram_e);
rxgv = rank(Xgram_v);

ryged = rank(Ygram_ed);
ryge = rank(Ygram_e);
rygv = rank(Ygram_v);

rdifed = rank(DiffG_ed);
rdife = rank(DiffG_e);
%rdifv = rank(DiffG_v);

rined = rank(XYin_ed);
rine = rank(XYin_e);
rinv = rank(XYin_v);

TraceGX_e = trace(Xgram_e);
TraceGX_ed = trace(Xgram_ed);

TraceGY_e = trace(Ygram_e);
TraceGY_ed = trace(Ygram_ed);

TraceDiffG_e = trace(DiffG_e);
TraceDiffG_ed = trace(DiffG_ed);

TraceXYin_e = trace(XYin_e);
TraceXYin_ed = trace(XYin_ed);

TraceCompGram_e = trace(CompGram_e);
TraceCompGram_ed = trace(CompGram_ed);



sdev_XG_e = sqrt(trace(Xgram_ed * Xgram_ed)/rank(Xgram_ed));
sdev_YG_e = sqrt(trace(Ygram_ed * Ygram_ed)/rank(Ygram_ed));

sdev_XYD_e = sqrt(trace(DiffG_ed * DiffG_ed)/rank(DiffG_ed));
sdev_XYD_ebar = sqrt(trace(DiffG_ed * conj(DiffG_ed))/rank(DiffG_ed));
```

```
sdev_IN_e = sqrt(trace(XYin_ed * XYin_ed)/rank(XYin_ed));
sdev_IN_ebar = sqrt(trace(XYin_ed * conj(XYin_ed))/rank(XYin_ed));

sdev_CompG_e = sqrt(trace(CompGram_ed * CompGram_ed)/rank(CompGram_ed));

println(ResXYX_e);
println(ResXY_e);

println(ResXYX_v);
println(ResXYY_v);
println(ResXY_v);
pause(100);


%{
A = imread("DSC_0089_200.png");
A = rgb2gray(A);
A = double(A)/255;
[row, col] = size(A);
rows = 1:row;
cols = 1:col;

%%Computing Singular Values:

n = 50; %% number of wanted eigenvectors and eigenvalues (AKA HOW COMPRESSED THE IMAGE↙
IS)
rand_vec = randn(col,1); % random eigenvector guess
u_vec = rand_vec./norm(rand_vec); % make into unit vec
K = A'*A;  "A transpose A" matrix
u_new = (K*u_vec)./norm(K*u_vec); % find closer eigenvector
% itterate until u_new ~= previous u_new
for i = 1:60
    u_new = (K*u_new)./norm(K*u_new);
end

eigvec1 = u_new; % first eigenvector
eigval1 = norm(K*u_new); % first eigenvalue

% Initialize vectors and matrices

V = zeros(col,n); % vector for eigenvectors
Eig = zeros(n,1); % vector for eigenvalues
V(:,1) = eigvec1;
Eig(1) = eigval1;

% Gram-Schmidt Orthogonalization Method

for i = 2:n % for every other eigenvector
    u_new = rand_vec/norm(rand_vec);
    tolerance = 10; % initialize tolerance for while loop

    while tolerance >= 0.001
    u_old = u_new;
    u_star = K*u_old;
    tot_sum = 0;
        for m = 1:i-1
            new_sum = u_star'*V(:,m);
            new_sum = new_sum*V(:,m);
            tot_sum = tot_sum + new_sum;
        end
    u_new = u_star - tot_sum;
    u_new = u_new/norm(u_new);
    tolerance = norm(u_new-u_old);
    end

    % collect eigenvectors and eigenvalues
    V(:,i) = u_new;
    Eig(i) = norm(K*u_new);
end

%% The Incomplete SVD Decomposition

I = eye(n); % 50x50 identity matrix
Sigma = sqrt(Eig).*I; % Sigma matrix
U = A*V/Sigma; % U matrix

A_new = U*Sigma*V'; %
%}

% Plot new image
%%figure(1)
%%imshow(Xgray_e*X_gray_v);
% Plot pre-compressed image
%%figure(2)
%%imshow(Xgray);
```