Table of Contents

JPEG-like Image Compression using DCT

Refactored implementation – same logic, modified structure & naming

```
clc;
clear;
close all;
```

1. Read Input Image

```
imgRGB = imread("input.jpeg");

figure("Name","Input Image");
imshow(imgRGB);
title("Original RGB Image");
```

2. Convert RGB to YCbCr

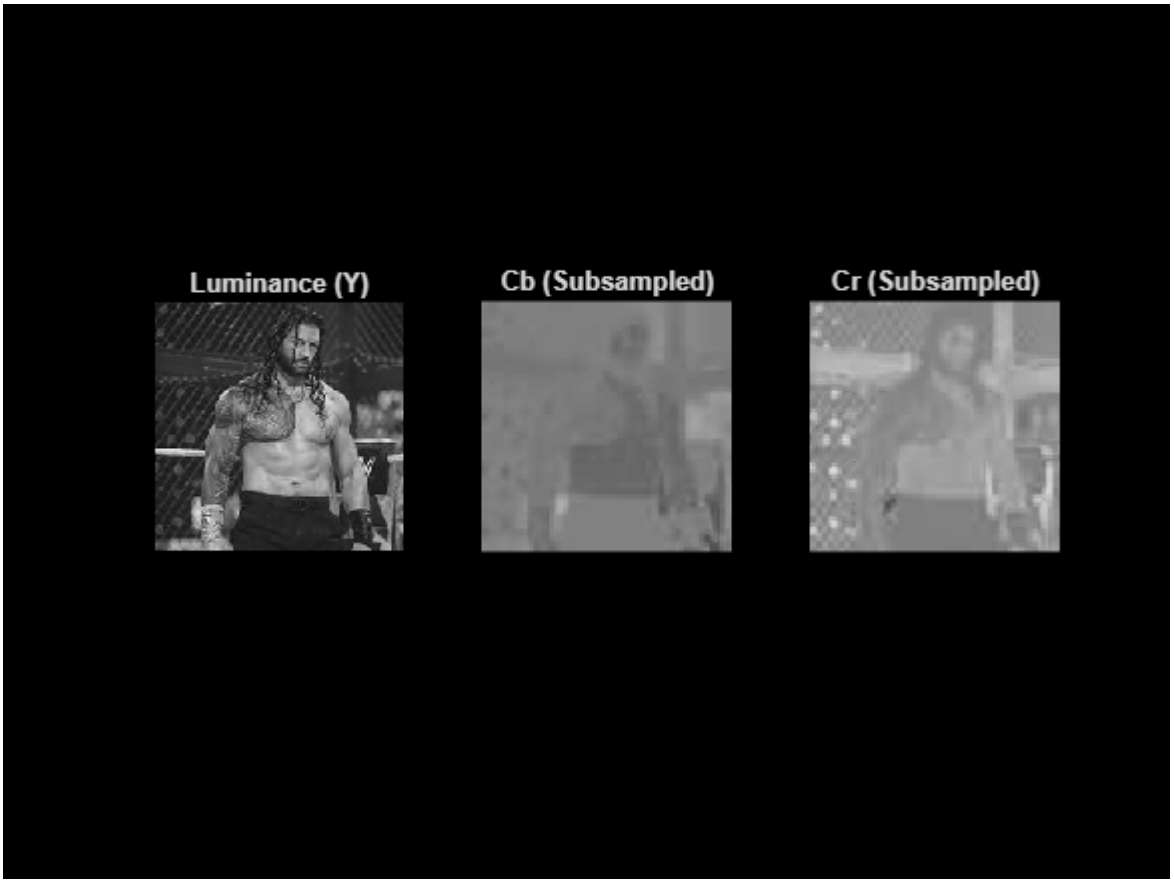
```
imgYCbCr = rgb2ycbcr(imgRGB);

Y_channel  = imgYCbCr(:,:,1);
Cb_channel = imgYCbCr(:,:,2);
Cr_channel = imgYCbCr(:,:,3);

figure("Name","Color Space Transformation");
subplot(1,2,1), imshow(imgRGB), title("RGB");
subplot(1,2,2), imshow(imgYCbCr), title("YCbCr");
```

3. Block Alignment (8x8)

```
blk = 8;
[rows, cols] = size(Y_channel);

rows8 = floor(rows/blk)*blk;
cols8 = floor(cols/blk)*blk;

Y_channel  = Y_channel(1:rows8, 1:cols8);
Cb_channel = Cb_channel(1:rows8, 1:cols8);
Cr_channel = Cr_channel(1:rows8, 1:cols8);
```

4. Chroma Subsampling (4:2:0)

```
Cb_sub = Cb_channel(1:2:end, 1:2:end);
Cr_sub = Cr_channel(1:2:end, 1:2:end);

figure("Name","Chroma Subsampling");
subplot(1,3,1), imshow(Y_channel), title("Luminance (Y)");
subplot(1,3,2), imshow(Cb_sub), title("Cb (Subsampled)");
subplot(1,3,3), imshow(Cr_sub), title("Cr (Subsampled)");
```



5. Level Shift

```
Y_shifted = double(Y_channel) - 128;
```

6. Construct 8x8 DCT Matrix

```
N = 8;
DCTmat = zeros(N);

for u = 0:N-1
    for x = 0:N-1
        if u == 0
            scale = sqrt(1/N);
        else
            scale = sqrt(2/N);
        end
        DCTmat(u+1, x+1) = scale * cos((2*x+1)*u*pi/(2*N));
    end
end
```

7. Apply DCT Blockwise

```
Y_dctCoeff = zeros(size(Y_shifted));
```

```
for r = 1:blk:rows8
    for c = 1:blk:cols8
        patch = Y_shifted(r:r+7, c:c+7);
        Y_dctCoeff(r:r+7, c:c+7) = DCTmat * patch * DCTmat';
    end
end
```

8. JPEG Standard Quantization Matrix (Luminance)

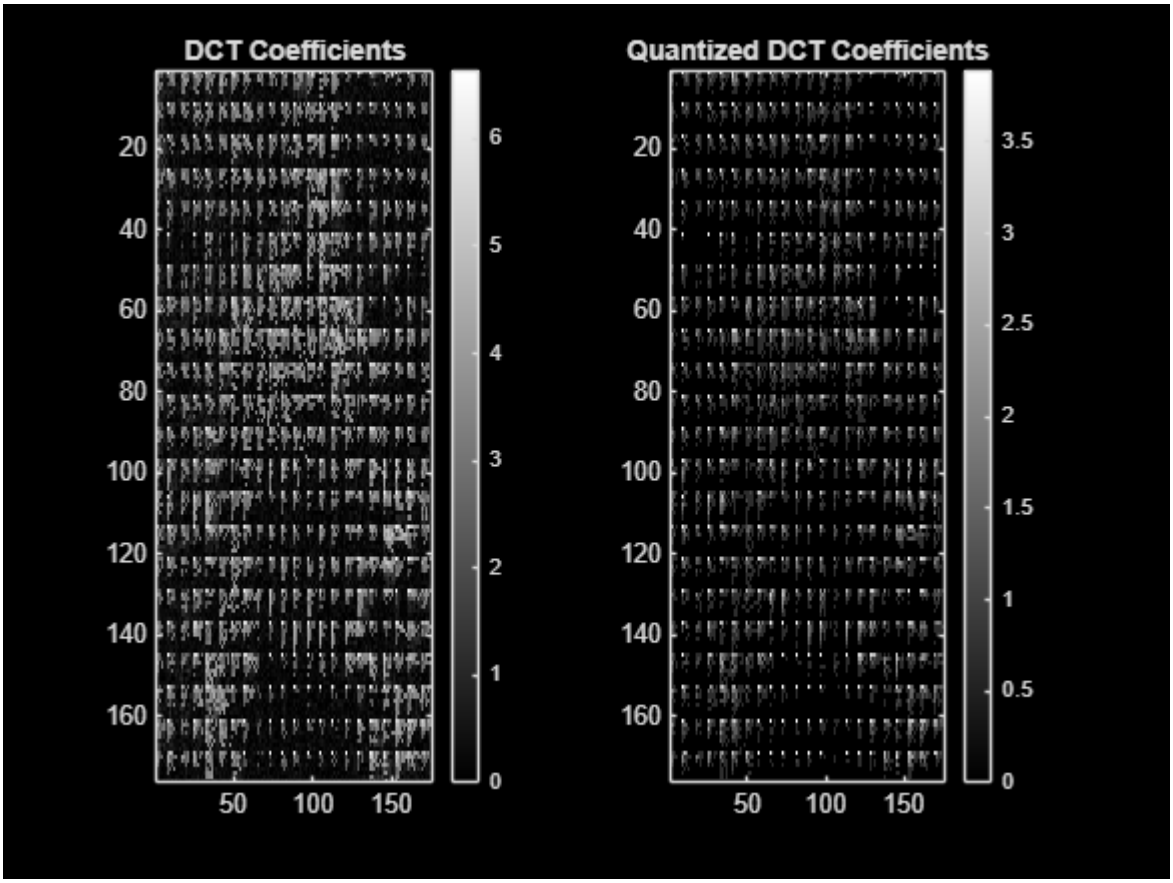
```
Q_L = [ ...
16 11 10 16 24 40 51 61;
12 12 14 19 26 58 60 55;
14 13 16 24 40 57 69 56;
14 17 22 29 51 87 80 62;
18 22 37 56 68 109 103 77;
24 35 55 64 81 104 113 92;
49 64 78 87 103 121 120 101;
72 92 95 98 112 100 103 99];
```

9. Quantize DCT Coefficients

```
Y_quant = zeros(size(Y_dctCoeff));

for r = 1:blk:rows8
    for c = 1:blk:cols8
        patch = Y_dctCoeff(r:r+7, c:c+7);
        Y_quant(r:r+7, c:c+7) = round(patch ./ Q_L);
    end
end
```

10. DCT Coefficient Visualization

```
figure("Name","DCT Compression Effect");
subplot(1,2,1);
imagesc(log(abs(Y_dctCoeff)+1)), colormap gray, colorbar;
title("DCT Coefficients");

subplot(1,2,2);
imagesc(log(abs(Y_quant)+1)), colormap gray, colorbar;
title("Quantized DCT Coefficients");
```



11. De-Quantization

```
Y_dequant = zeros(size(Y_quant));

for r = 1:blk:rows8
    for c = 1:blk:cols8
        patch = Y_quant(r:r+7, c:c+7);
        Y_dequant(r:r+7, c:c+7) = patch .* Q_L;
    end
end
```

12. Inverse DCT

```
Y_reconstructed = zeros(size(Y_dequant));

for r = 1:blk:rows8
    for c = 1:blk:cols8
        patch = Y_dequant(r:r+7, c:c+7);
        Y_reconstructed(r:r+7, c:c+7) = DCTmat' * patch * DCTmat;
    end
end
```

13. Inverse Level Shift

```
Y_reconstructed = uint8(min(max(Y_reconstructed + 128, 0), 255));
```

14. Chroma Upsampling

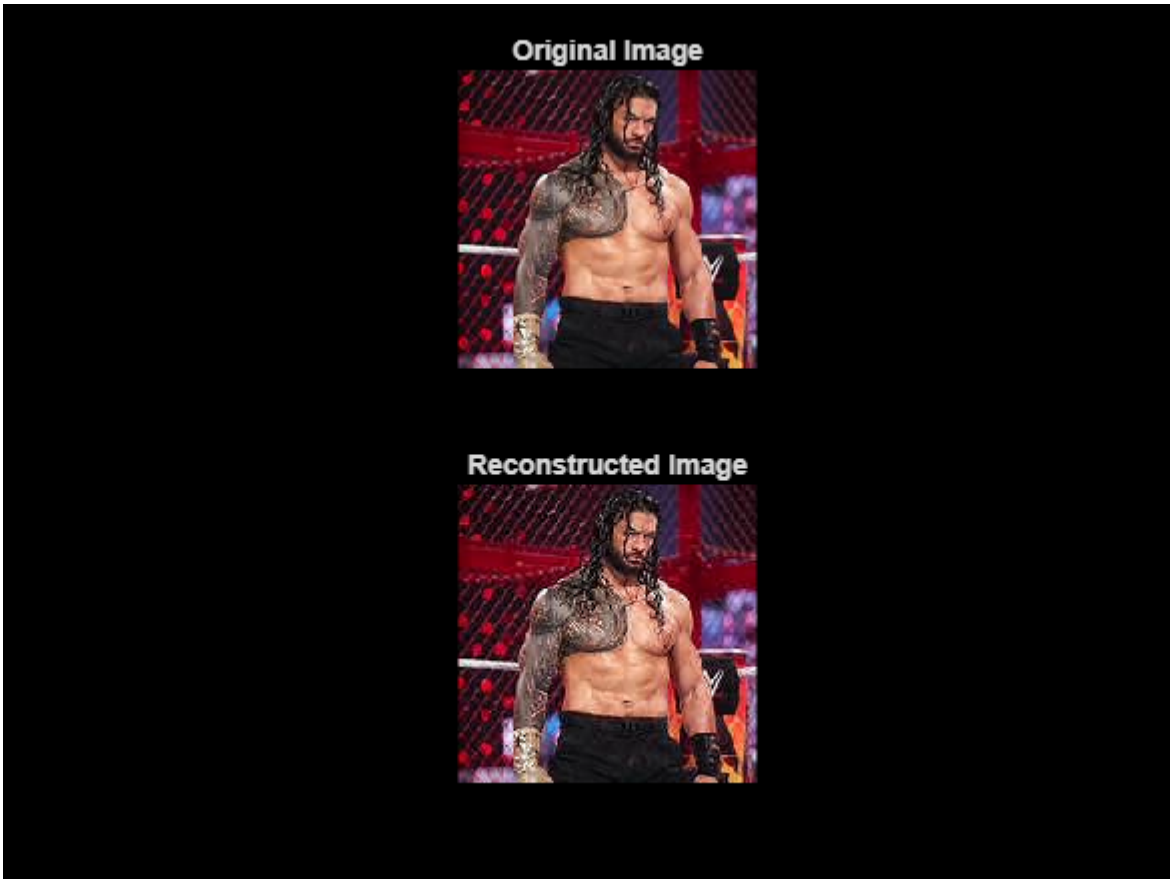
```
Cb_up = imresize(Cb_sub, 2, 'bilinear');
Cr_up = imresize(Cr_sub, 2, 'bilinear');

Cb_up = uint8(Cb_up(1:rows8,1:cols8));
Cr_up = uint8(Cr_up(1:rows8,1:cols8));
```

15. Merge Channels & Convert Back to RGB

```
imgYCbCr_rec = cat(3, Y_reconstructed, Cb_up, Cr_up);
imgRGB_rec = ycbcr2rgb(imgYCbCr_rec);
```

16. Visual Comparison

```
figure("Name","JPEG-like Compression Result");
subplot(2,1,1);
imshow(imgRGB(1:rows8,1:cols8,:));
title("Original Image");

subplot(2,1,2);
imshow(imgRGB_rec);
title("Reconstructed Image");
```



17. Quality Evaluation

```
mse_val = mean((double(imgRGB(1:rows8,1:cols8,:)) - double(imgRGB_rec)).^2,
'all');
psnr_val = 10 * log10(255^2 / mse_val);

fprintf("MSE  = %.4f\n", mse_val);
fprintf("PSNR = %.2f dB\n", psnr_val);
```

```
MSE  = 77.5526
PSNR = 29.23 dB
```

Published with MATLAB® R2025b