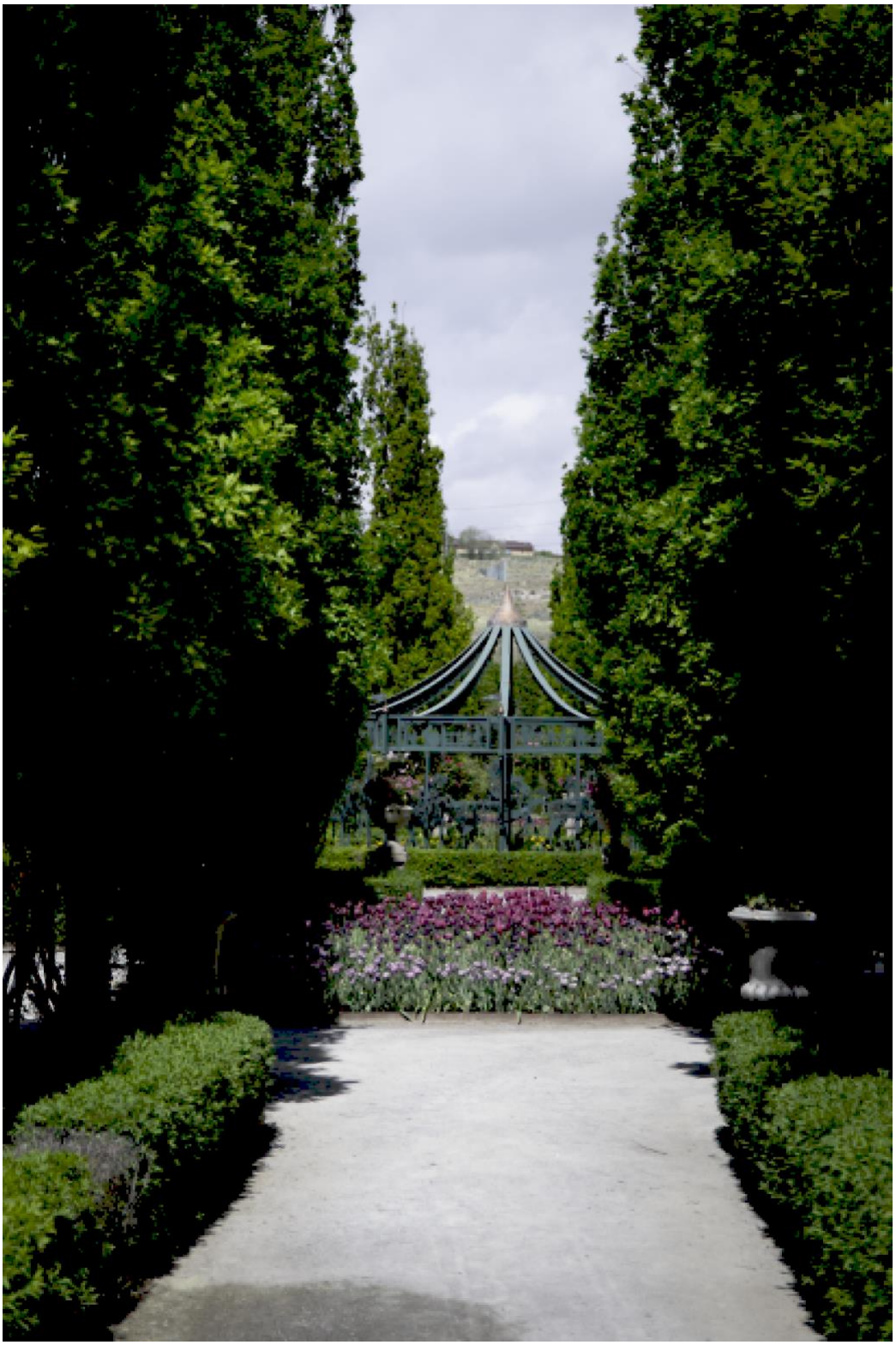
Original Image | size: 48.5 MB

Base 8x8 block at location 1112 block before compression

```
Single 8 x 8 block of red, green, and blue before DCT at position 1112

RED
 [[19 17 18 15 13 16 16 14]
 [19 17 18 17 14 15 16 14]
 [33 26 19 19 19 19 18 17]
 [34 29 27 22 18 15 17 18]
 [42 37 32 29 26 23 20 22]
 [57 56 56 55 59 57 39 26]
 [73 73 77 84 81 64 69 41]
 [81 89 89 86 91 94 75 51]]

GREEN
 [[ 26  24  24  21  18  22  22  20]
 [ 26  23  25  23  20  20  22  20]
 [ 46  36  27  26  26  26  25  25]
 [ 48  40  37  31  24  21  24  26]
 [ 58  51  45  41  38  34  30  32]
 [ 78  76  76  75  80  77  55  38]
 [ 99  97 103 111 109  89  91  56]
 [108 117 119 116 123 125  99  69]]

BLUE
 [[10  9 10  9  7  9  9  8]
 [ 9  8 10  9  8  8  9  8]
 [16 12  9 10 10 10  9  9]
 [15 12 12 10  8  7  8  9]
 [15 13 11  9  9  8  7  9]
 [18 18 18 18 20 20 13  9]
 [21 21 23 27 24 18 25 13]
 [22 25 24 21 23 26 24 16]]
```

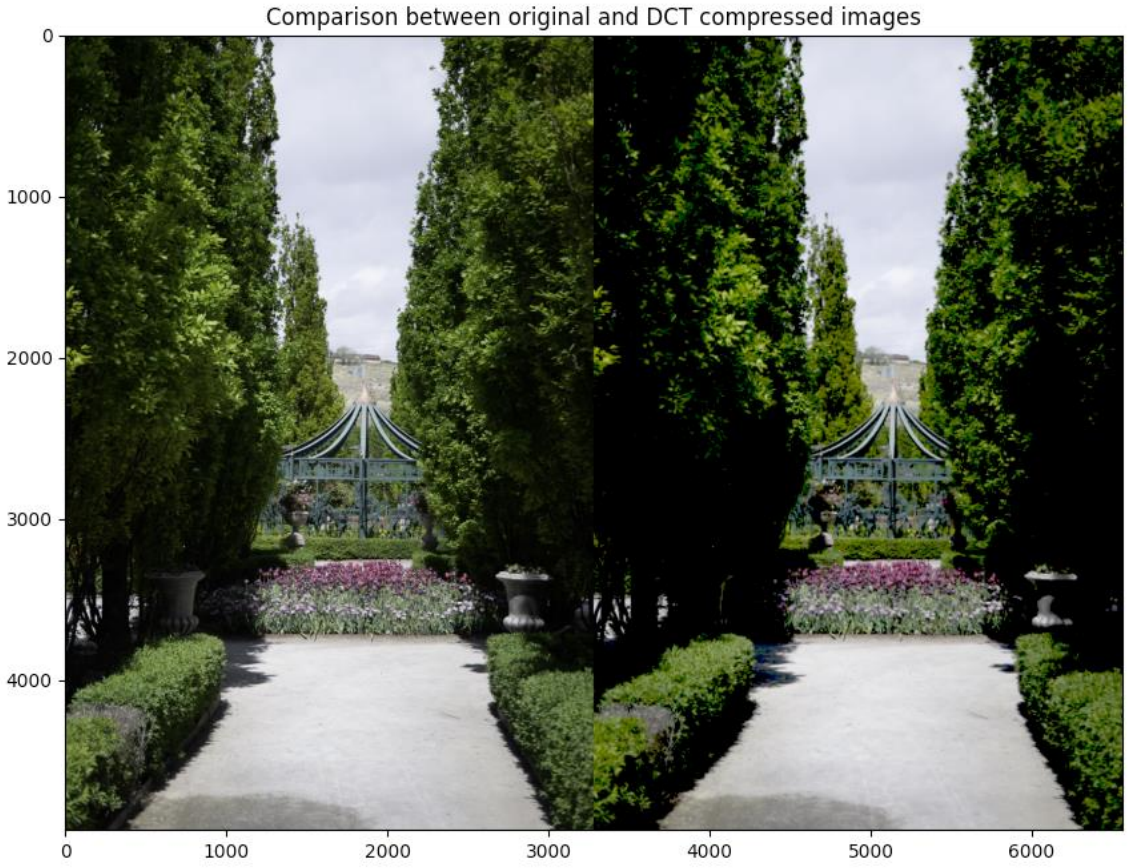
Usable Image | size 488 KB



Compressed 8x8 block at location 1112 with the p-value 19

```
Single 8 x 8 block of red, green, and blue after DCT at position 1112

RED
 [[0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]]

GREEN
 [[52 52 52 52 52 52 52 52]
 [52 52 52 52 52 52 52 52]
 [52 52 52 52 52 52 52 52]
 [52 52 52 52 52 52 52 52]
 [52 52 52 52 52 52 52 52]
 [52 52 52 52 52 52 52 52]
 [52 52 52 52 52 52 52 52]
 [52 52 52 52 52 52 52 52]]

BLUE
 [[0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]]
```

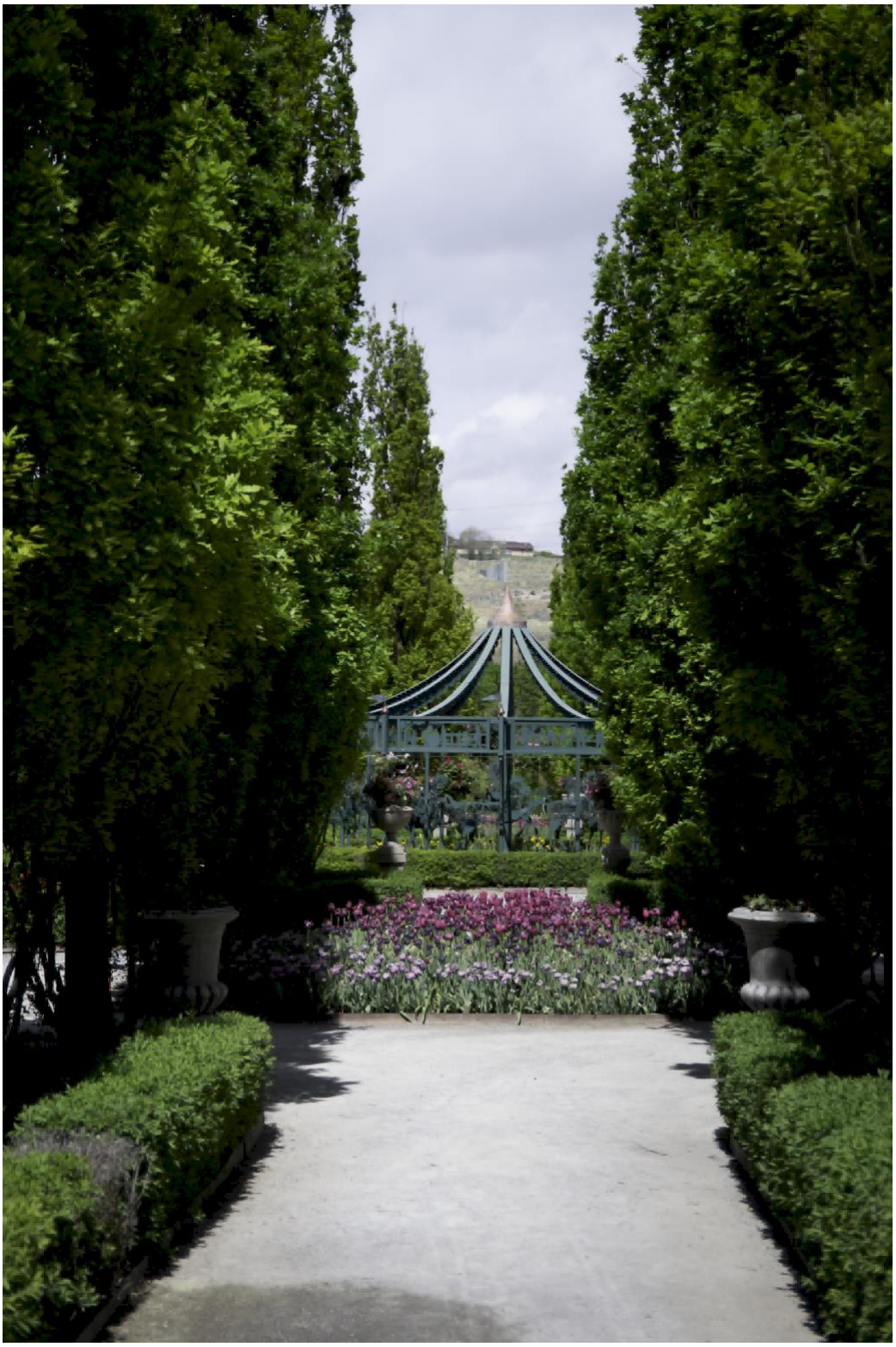
Side-by-side comparison



Full console output

```
Compressing [ ./imgs/DSC_1903a.tif ]

 Base dct matrix
 [[ 0.354  0.49   0.462  0.416  0.354  0.278  0.191  0.098]
 [ 0.354  0.416  0.191 -0.098 -0.354 -0.49  -0.462 -0.278]
 [ 0.354  0.278 -0.191 -0.49  -0.354  0.098  0.462  0.416]
 [ 0.354  0.098 -0.462 -0.278  0.354  0.416 -0.191 -0.49 ]
 [ 0.354 -0.098 -0.462  0.278  0.354 -0.416 -0.191  0.49 ]
 [ 0.354 -0.278 -0.191  0.49  -0.354 -0.098  0.462 -0.416]
 [ 0.354 -0.416  0.191  0.098 -0.354  0.49  -0.462  0.278]
 [ 0.354 -0.49   0.462 -0.416  0.354 -0.278  0.191 -0.098]]

 Base linear quantization matrix
 [[ 152.  304.  456.  608.  760.  912. 1064. 1216.]
 [ 304.  456.  608.  760.  912. 1064. 1216. 1368.]
 [ 456.  608.  760.  912. 1064. 1216. 1368. 1520.]
 [ 608.  760.  912. 1064. 1216. 1368. 1520. 1672.]
 [ 760.  912. 1064. 1216. 1368. 1520. 1672. 1824.]
 [ 912. 1064. 1216. 1368. 1520. 1672. 1824. 1976.]
 [1064. 1216. 1368. 1520. 1672. 1824. 1976. 2128.]
 [1216. 1368. 1520. 1672. 1824. 1976. 2128. 2280.]]

Single 8 x 8 block of red, green, and blue before DCT at position 1112

RED
 [[19 17 18 15 13 16 16 14]
 [19 17 18 17 14 15 16 14]
 [33 26 19 19 19 19 18 17]
 [34 29 27 22 18 15 17 18]
 [42 37 32 29 26 23 20 22]
 [57 56 56 55 59 57 39 26]
 [73 73 77 84 81 64 69 41]
 [81 89 89 86 91 94 75 51]]

GREEN
 [[ 26  24  24  21  18  22  22  20]
 [ 26  23  25  23  20  20  22  20]
 [ 46  36  27  26  26  26  25  25]
 [ 48  40  37  31  24  21  24  26]
 [ 58  51  45  41  38  34  30  32]
 [ 78  76  76  75  80  77  55  38]
 [ 99  97 103 111 109  89  91  56]
 [108 117 119 116 123 125  99  69]]

BLUE
 [[10  9 10  9  7  9  9  8]
 [ 9  8 10  9  8  8  9  8]
 [16 12  9 10 10 10  9  9]
 [15 12 12 10  8  7  8  9]
 [15 13 11  9  9  8  7  9]
 [18 18 18 18 20 20 13  9]
 [21 21 23 27 24 18 25 13]
 [22 25 24 21 23 26 24 16]]

Compressing image..
Keeping only 2.154507% of the DCT coefficients..

Single 8 x 8 block of red, green, and blue after DCT at position 1112

RED
 [[0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]]

GREEN
 [[52 52 52 52 52 52 52 52]
 [52 52 52 52 52 52 52 52]
 [52 52 52 52 52 52 52 52]
 [52 52 52 52 52 52 52 52]
 [52 52 52 52 52 52 52 52]
 [52 52 52 52 52 52 52 52]
 [52 52 52 52 52 52 52 52]
 [52 52 52 52 52 52 52 52]]

BLUE
 [[0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]]


Writing file [ ./output/DSC_1903a-usable.jpg ]
Done.
```

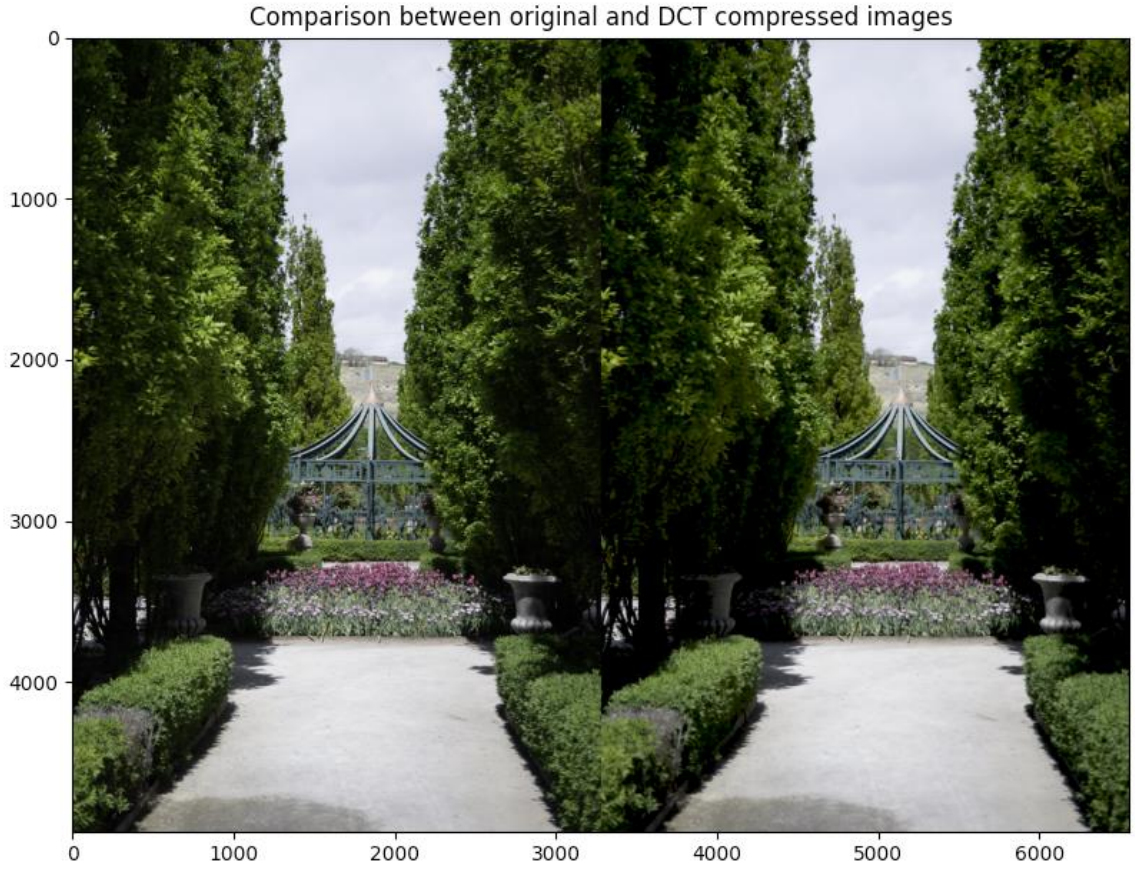
Good Image | size 618 KB



Compressed 8x8 block at location 1112 with the p-value 7

```
Single 8 x 8 block of red, green, and blue after DCT at position 1112

RED
 [[ 7  7  7  7  7  7  7  7]
 [12 12 12 12 12 12 12 12]
 [20 20 20 20 20 20 20 20]
 [32 32 32 32 32 32 32 32]
 [44 44 44 44 44 44 44 44]
 [56 56 56 56 56 56 56 56]
 [64 64 64 64 64 64 64 64]
 [69 69 69 69 69 69 69 69]]

GREEN
 [[10 10 10 10 10 10 10 10]
 [17 17 17 17 17 17 17 17]
 [28 28 28 28 28 28 28 28]
 [44 44 44 44 44 44 44 44]
 [60 60 60 60 60 60 60 60]
 [76 76 76 76 76 76 76 76]
 [87 87 87 87 87 87 87 87]
 [94 94 94 94 94 94 94 94]]

BLUE
 [[0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]]
```

Side-by-side comparison



Full console output

```
Compressing [ ./imgs/DSC_1903a.tif ]

 Base dct matrix
 [[ 0.354  0.49   0.462  0.416  0.354  0.278  0.191  0.098]
 [ 0.354  0.416  0.191 -0.098 -0.354 -0.49  -0.462 -0.278]
 [ 0.354  0.278 -0.191 -0.49  -0.354  0.098  0.462  0.416]
 [ 0.354  0.098 -0.462 -0.278  0.354  0.416 -0.191 -0.49 ]
 [ 0.354 -0.098 -0.462  0.278  0.354 -0.416 -0.191  0.49 ]
 [ 0.354 -0.278 -0.191  0.49  -0.354 -0.098  0.462 -0.416]
 [ 0.354 -0.416  0.191  0.098 -0.354  0.49  -0.462  0.278]
 [ 0.354 -0.49   0.462 -0.416  0.354 -0.278  0.191 -0.098]]

 Base linear quantization matrix
 [[ 56. 112. 168. 224. 280. 336. 392. 448.]
 [112. 168. 224. 280. 336. 392. 448. 504.]
 [168. 224. 280. 336. 392. 448. 504. 560.]
 [224. 280. 336. 392. 448. 504. 560. 616.]
 [280. 336. 392. 448. 504. 560. 616. 672.]
 [336. 392. 448. 504. 560. 616. 672. 728.]
 [392. 448. 504. 560. 616. 672. 728. 784.]
 [448. 504. 560. 616. 672. 728. 784. 840.]]

Single 8 x 8 block of red, green, and blue before DCT at position 1112

RED
 [[19 17 18 15 13 16 16 14]
 [19 17 18 17 14 15 16 14]
 [33 26 19 19 19 19 18 17]
 [34 29 27 22 18 15 17 18]
 [42 37 32 29 26 23 20 22]
 [57 56 56 55 59 57 39 26]
 [73 73 77 84 81 64 69 41]
 [81 89 89 86 91 94 75 51]]

GREEN
 [[ 26  24  24  21  18  22  22  20]
 [ 26  23  25  23  20  20  22  20]
 [ 46  36  27  26  26  26  25  25]
 [ 48  40  37  31  24  21  24  26]
 [ 58  51  45  41  38  34  30  32]
 [ 78  76  76  75  80  77  55  38]
 [ 99  97 103 111 109  89  91  56]
 [108 117 119 116 123 125  99  69]]

BLUE
 [[10  9 10  9  7  9  9  8]
 [ 9  8 10  9  8  8  9  8]
 [16 12  9 10 10 10  9  9]
 [15 12 12 10  8  7  8  9]
 [15 13 11  9  9  8  7  9]
 [18 18 18 18 20 20 13  9]
 [21 21 23 27 24 18 25 13]
 [22 25 24 21 23 26 24 16]]

Compressing image..
Keeping only 3.734806% of the DCT coefficients..

Single 8 x 8 block of red, green, and blue after DCT at position 1112

RED
 [[ 7  7  7  7  7  7  7  7]
 [12 12 12 12 12 12 12 12]
 [20 20 20 20 20 20 20 20]
 [32 32 32 32 32 32 32 32]
 [44 44 44 44 44 44 44 44]
 [56 56 56 56 56 56 56 56]
 [64 64 64 64 64 64 64 64]
 [69 69 69 69 69 69 69 69]]

GREEN
 [[10 10 10 10 10 10 10 10]
 [17 17 17 17 17 17 17 17]
 [28 28 28 28 28 28 28 28]
 [44 44 44 44 44 44 44 44]
 [60 60 60 60 60 60 60 60]
 [76 76 76 76 76 76 76 76]
 [87 87 87 87 87 87 87 87]
 [94 94 94 94 94 94 94 94]]

BLUE
 [[0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]]


Writing file [ ./output/DSC_1903a-good.jpg ]
Done.
```

dct_compression.py (script)

```
import numpy as np
from numpy import r_
from skimage import io
from scipy.linalg import hilbert
from scipy.fft import dct
from scipy.fft import idct
import matplotlib.pyplot as plt

IMAGE_PATH = "./imgs/DSC_1903a.tif"
IMAGE_NAME = IMAGE_PATH.split('/')[2].split('.')[0]
QUAL = "good" # good | usable

print(f'\nCompressing [ {IMAGE_PATH} ]')

p = 7 # 4 = "great", 7 = "good", 19 = "usable"
N = 8
Q = (p*8)/(hilbert(8))  # linear quantization matrix

dct_matrix = dct(np.eye(N), axis=1, norm='ortho')

np.set_printoptions(precision=3)
print('\n', 'Base dct matrix\n', dct_matrix)
print('\n', 'Base linear quantization matrix\n', Q)

image_raw = io.imread(IMAGE_PATH).astype(float)
image = np.array(image_raw, dtype=np.uint8)  # uint8 is an 8 bit integer

# 8 x 8 blocks for red, green, and blue before DCT
print('\nSingle 8 x 8 block of red, green, and blue before DCT at position 1112\n')
print('RED\n', image[1112:1120, 1112:1120, 0], '\n') # R
print('GREEN\n', image[1112:1120, 1112:1120, 1], '\n') # G
print('BLUE\n', image[1112:1120, 1112:1120, 2], '\n') # B
```

```
image_size = image.shape

h, w, channels = image_size
height = round(h/N-1)
width = round(w/N-1)
new_image_size = (height, width, channels)
dct_zeros = np.zeros(image_size)


def dct2d(block):
    """Get the DCT of a 2 dimensional array"""
    return dct(dct(block, axis=0, norm='ortho'), axis=1, norm='ortho')


def idct2d(block):
    """Get the IDCT of a 2 dimensional array"""
    return idct(idct(block, axis=0, norm='ortho'), axis=1, norm='ortho')


print('Compressing image..')

# 8x8 DCT on image (in-place)
for i in r_[:image_size[0]:N]:
    for j in r_[:image_size[1]:N]:
        dct_zeros[i:(i+N), j:(j+N)] = dct2d(image[i:(i+N), j:(j+N)])

# p Loss Threshold
p_threshold = p/100
dct_threshold = dct_zeros * (abs(dct_zeros) > (p_threshold*np.max(dct_zeros)))

nonzeros_percent = np.sum(dct_threshold != 0.0) / \
    (image_size[0]*image_size[1]*1.0)
```

```
print("Keeping only %f%% of the DCT coefficients.." % (nonzeros_percent*100.0))

img_dct = np.zeros(image_size)

for i in r_[:image_size[0]:N]:
    for j in r_[:image_size[1]:N]:
        img_dct[i:(i+N), j:(j+N)] = idct2d(dct_threshold[i:(i+N), j:(j+N)])

# create a new image
img = np.array(img_dct, dtype=np.float64)
img = img.astype(np.uint8)

# 8 x 8 blocks for red, green, and blue after DCT
print('\nSingle 8 x 8 block of red, green, and blue after DCT at position 1112\n')
print('RED\n', img[1112:1120, 1112:1120, 0], '\n') # R
print('GREEN\n', img[1112:1120, 1112:1120, 1], '\n') # G
print('BLUE\n', img[1112:1120, 1112:1120, 2], '\n') # B

output_name = f'./output/{IMAGE_NAME}-{QUAL}.jpg'
print(f'\nWriting file [ {output_name} ]')

io.imsave(output_name, img)

plt.figure()
plt.imshow(np.hstack((image, img)), cmap='gray')
plt.title("Comparison between original and DCT compressed images")

plt.show()

print('Done.\n')
```