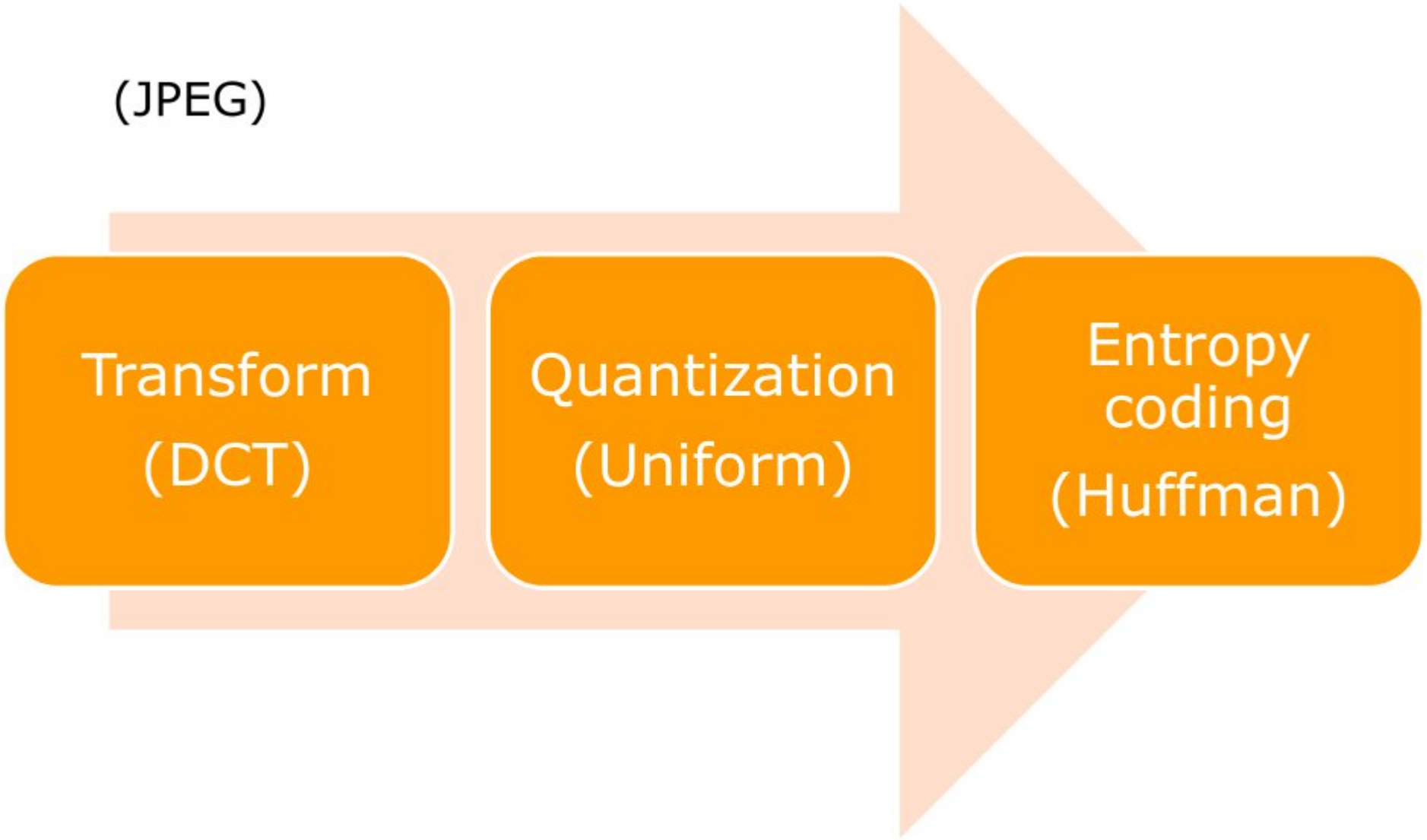
JPEG-kodning

Daniel Brasholt s214675

Formål og beskrivelse

- DCT
- Kvantisering
- Huffman
- PSNR

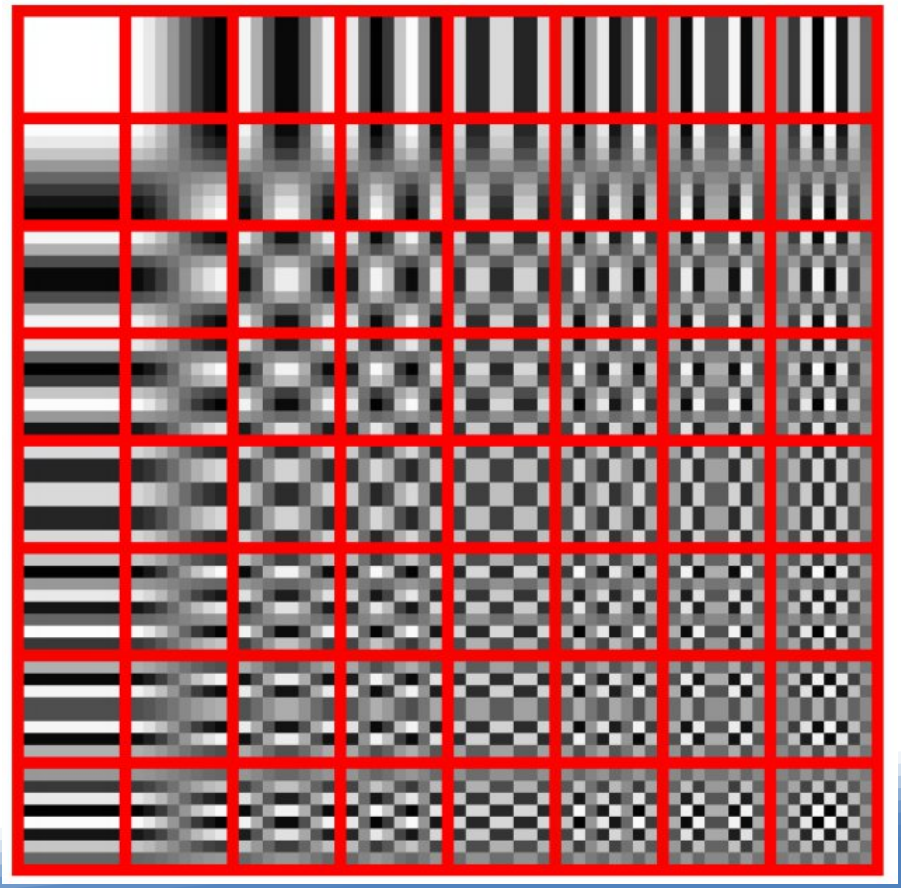
(JPEG)


Transform
(DCT)
Quantization
(Uniform)
Entropy
coding
(Huffman)

DCT – kort fortalt

- Struktur og detaljer
- 8x8 blokke

788	45	44	10	-3	-1	-7	-1
219	29	43	28	3	-1	5	0
11	59	46	4	-2	0	2	2
38	26	19	-25	-22	-6	-3	2
9	3	-16	-6	-19	-9	3	2
-8	6	-10	-14	-6	-5	0	1
-2	-7	2	-9	-6	1	4	5
7	-5	-3	-1	6	1	8	3



Kvantisering

- Forskellige muligheder
- Fjerner detaljer

$$\tilde{x} = \left\lfloor \frac{x}{q} \right\rfloor$$

```
Q = np.matrix([
    [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]
])
```

25	1	1	0	0	0	0	0
7	1	1	1	0	0	0	0
0	2	1	0	0	0	0	0
1	1	1	0	-1	0	0	0
0	0	-1	0	-1	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Huffman

- Differens fra blok til blok
- Kategori
- 2’s komplement
- Fx -5

SSSS	DIFF values
0	0
1	–1,1
2	–3,–2,2,3
3	–7..–4,4..7
4	–15..–8,8..15
5	–31..–16,16..31
6	–63..–32,32..63
7	–127..–64,64..127
8	–255..–128,128..255
9	–511..–256,256..511
10	–1 023..–512,512..1 023
11	–2 047..–1 024,1 024..2 047

MSE & PSNR

$$PSNR = 10 \cdot log_{10}\left(\frac{MAX^2}{MSE}\right)$$

$$MSE = \frac{1}{MN}\sum_{M}\sum_{N}(X_{M,N} - \tilde{X}_{M,N})^2$$

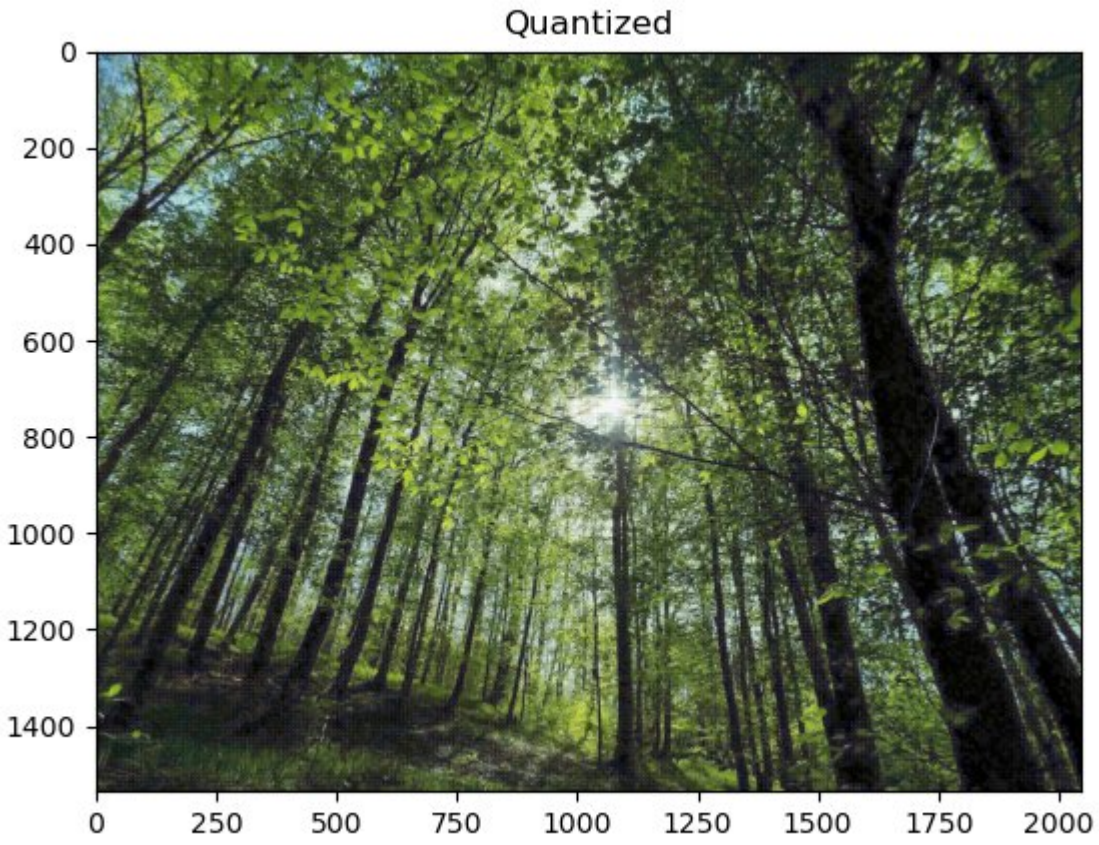
Implementering

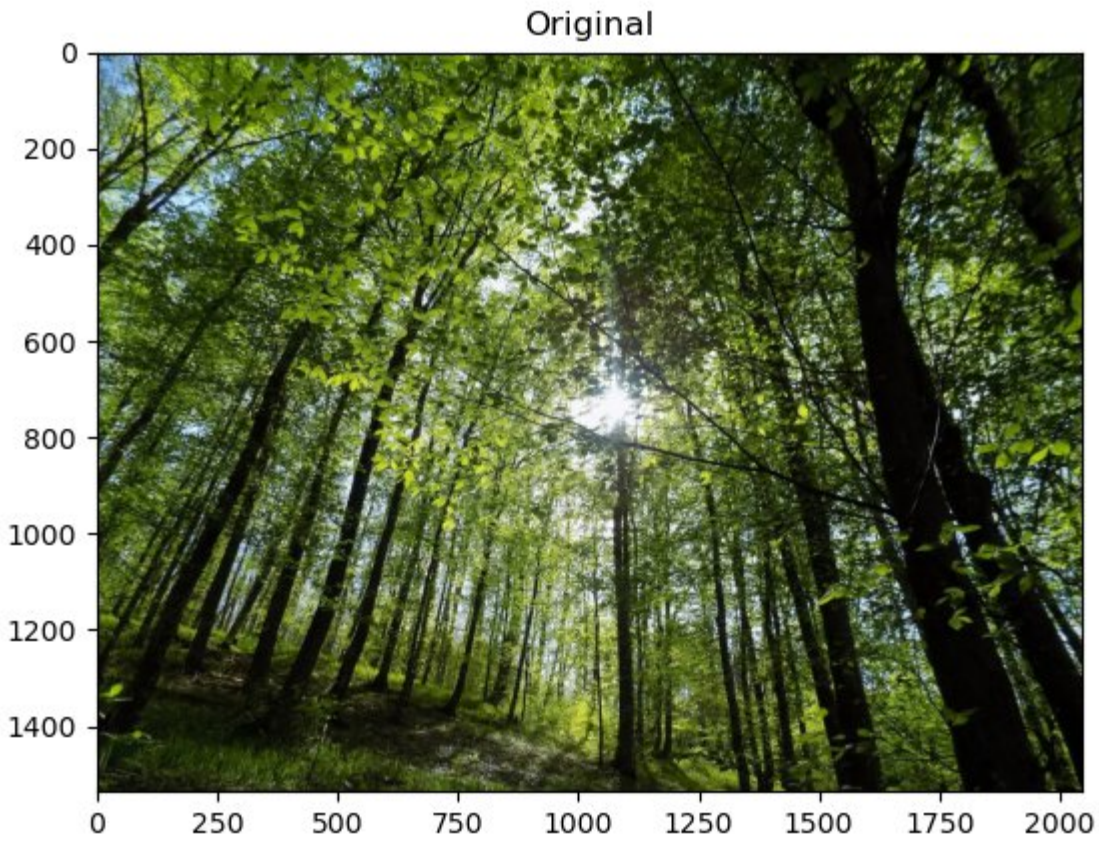
- Python
- Numpy
- Scipy

DCT & kvantisering

```
def get_dct(img_matrix):
    matrix_size = img_matrix.shape
    dct_matrix = np.zeros(matrix_size)
    quantized_dct = np.zeros(matrix_size)

    for i in np.r_[:matrix_size[0]:8]:
        for j in np.r_[:matrix_size[1]:8]:
            dct_matrix[i:(i+8), j:(j+8)] = dct2( img_matrix[i:(i+8), j:(j+8)] )
            quantized_dct[i:(i+8), j:(j+8)] = np.divide(dct_matrix[i:(i+8), j:(j+8)], Q)
    return np.rint(dct_matrix), np.rint(quantized_dct)
```


Quantized
0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750
2000


Original
0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750
2000

Huffman

Table K.3 – Table for luminance DC coefficient differences

Category	Code length	Code word
0	2	00
1	3	010
2	3	011
3	3	100
4	3	101
5	3	110
6	4	1110
7	5	11110
8	6	111110
9	7	1111110
10	8	11111110
11	9	111111110

```
def encode_dc_values(diff_array, category_codes):
    encoded = []
    for diff in diff_array:
        cat = get_diff_category(diff)
        if diff < 0:
            bit_value = bin(twos_complement(-diff+1, cat))
            encoded.append( category_codes[cat] + bit_value[2:].rjust(cat, '0') )
        else:
            bit_value = bin(diff)
            encoded.append( category_codes[cat] + bit_value[2:].rjust(cat, '0') )
    return encoded
```

```
diffs = difference_dc_values(y_quantized.astype('int'))
codes = encode_dc_values(diffs,k3)
codelens = [len(i) for i in codes]
print(sum(codelens))
print(len(diffs)*11)
```

```
367057
540672
```

```
diffs = difference_dc_values(v_quantized.astype('int'))
codes = encode_dc_values(diffs,k4)
codelens = [len(i) for i in codes]
print(sum(codelens))
print(len(diffs)*11)
```

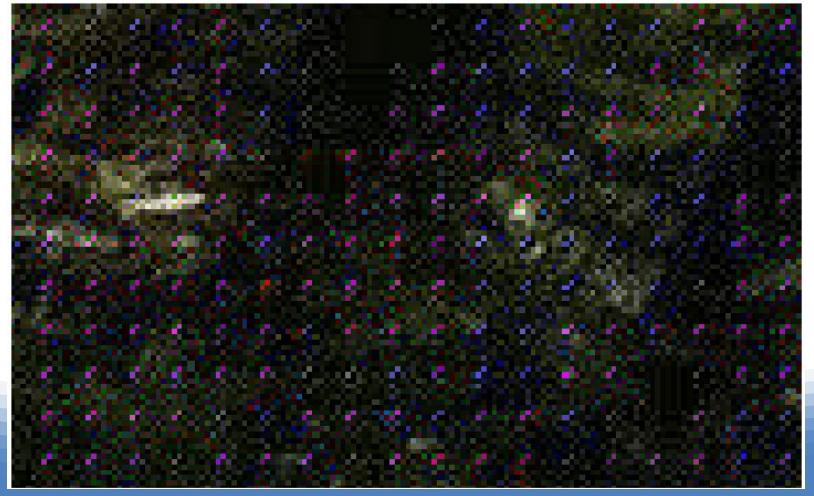
```
160962
540672
```

MSE & PSNR

```python
def MSE(original, changed):
    imgsize = original.shape
    MN = imgsize[0] * imgsize[1]
    diffs = original - changed
    diffs = np.multiply(diffs, diffs)
    diffs = diffs / MN
    summed = diffs.sum()
    return summed
```

```python
ms = MSE(y, y_inv)
PSNR = 10*math.log10((np.max(y)**2) / ms)
print(ms, PSNR)
```

```
0.4859628677368166 51.23064347008631
```



```python
ms = MSE(y, y_q_inv)
PSNR = 10*math.log10((np.max(y)**2) / ms)
print(ms, PSNR)
```

```
1508.6540505091357 16.31077770124197
```