DISCRETE COSINE TRANSFORMATION

-HACKERS DELIGHT-

Michael Eiler

15.5.2013

AGENDA

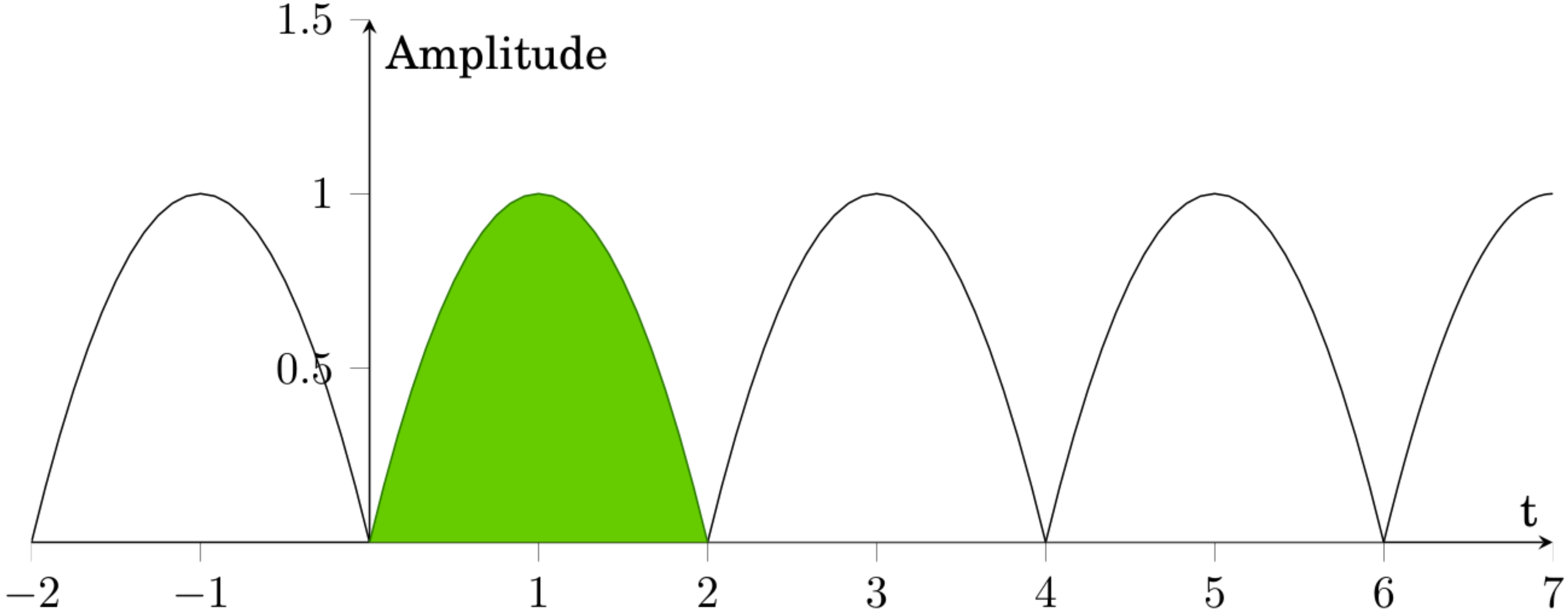
Theory (many formulas)

- Fourier-Series & Fourier-Transformation (3-5)
- Discrete Cosine-Transformation (6-13)
- Multiple Dimensions (14)

Applied Science

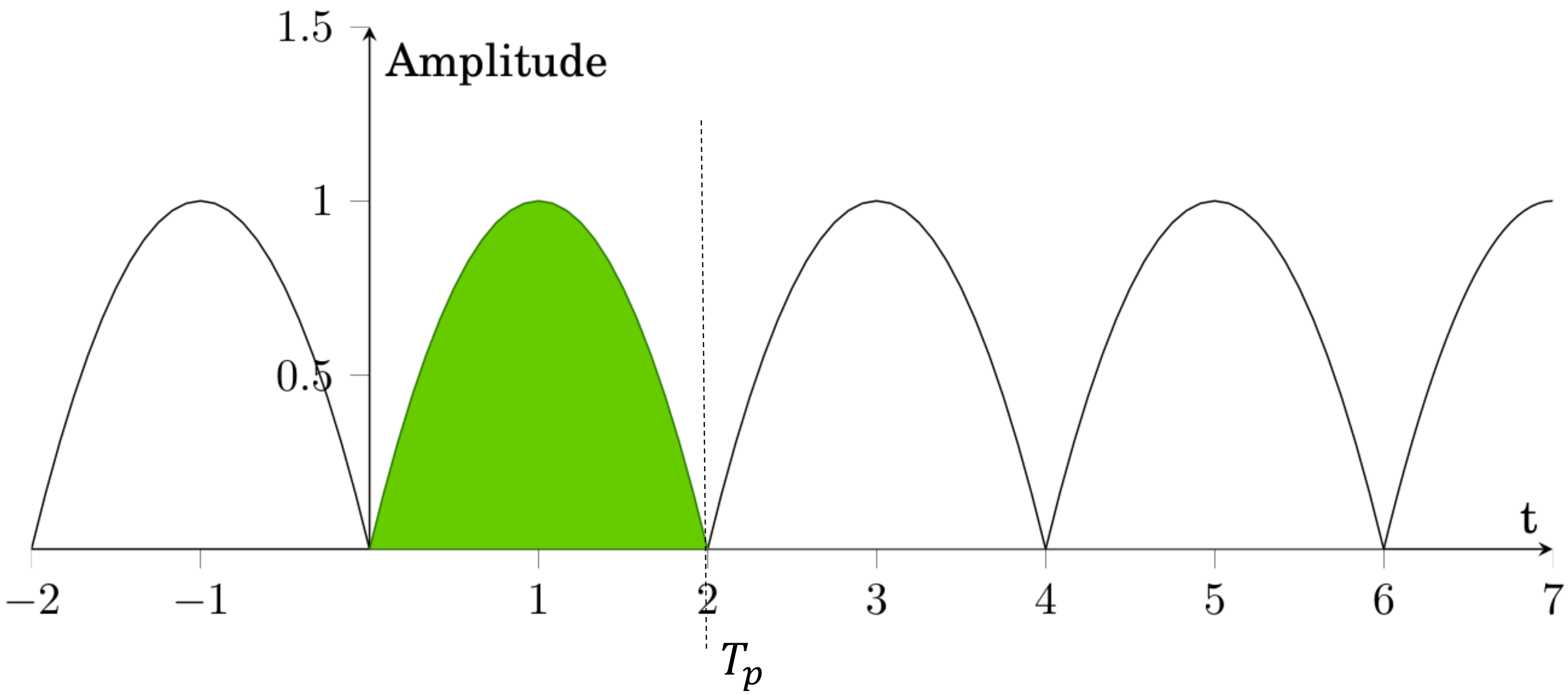
- Image Compression – JPEG Algorithm Example (15-25)
- JPEG Decoder Implementation (25-34)

FOURIER-SERIES (1/2)



$$y(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos(k\omega_0 t) + b_k \sin(k\omega_0 t))$$

FOURIER-SERIES (2/2)



$$a_k = \frac{2}{T_p}\int_0^{T_p} x(t)\cos(k\omega_0 t)dt \qquad b_k = \frac{2}{T_p}\int_0^{T_p} x(t)\sin(k\omega_0 t)dt \qquad T_p = \frac{2\pi}{\omega_0}$$

FOURIER-TRANSFORMATION

- Can also be used for non-periodic signals
- Is based on Fourier-Series

- Transformation

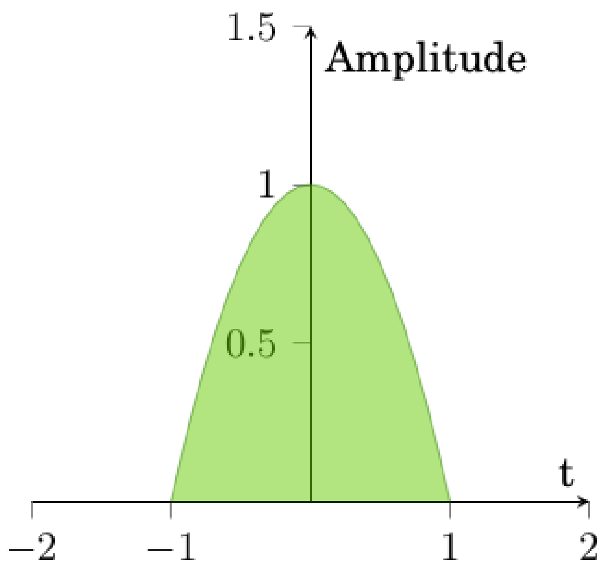
$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi t} dt$$

$$\mathrm{x}(t) : \mathbb{R} \rightarrow \mathbb{C}$$
$$X(f) : \mathbb{R} \rightarrow \mathbb{C}$$

- Inverse Transformation

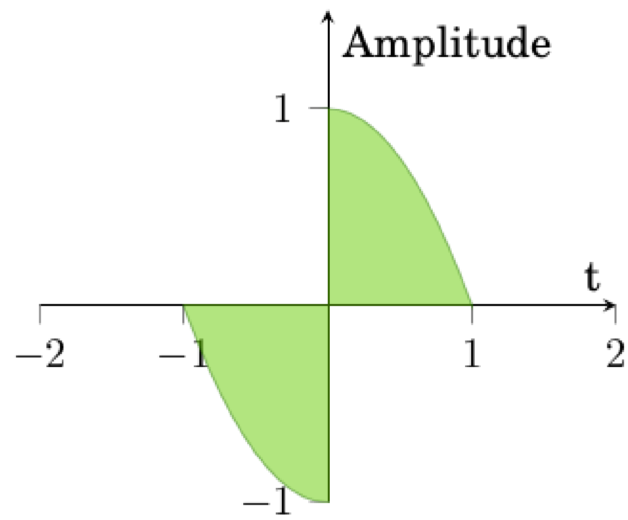
$$x(t) = \int_{-\infty}^{\infty} X(f) e^{j2\pi t} dt$$

PROPERTIES OF FT/FS



axially symmateric impulse

- No Sine components
- $\forall f \epsilon \mathbb{R}$: $X(f) \epsilon \mathbb{R}, X(f) \overset{FT}{\leftrightarrows} x(t)$



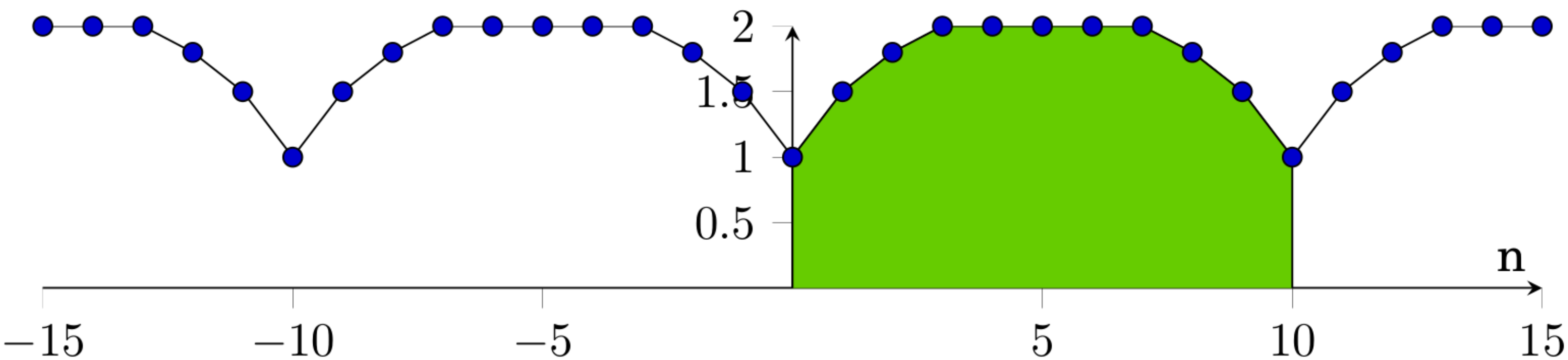
point symmetric impulse

- No Cosine components
- $\forall f \epsilon \mathbb{R}$: $X(f) \epsilon \mathbb{C}, X(f) \overset{FT}{\leftrightarrows} x(t)$

7

DISCRETE COSINE TRANSFORMATION

DCT-I (1/4)

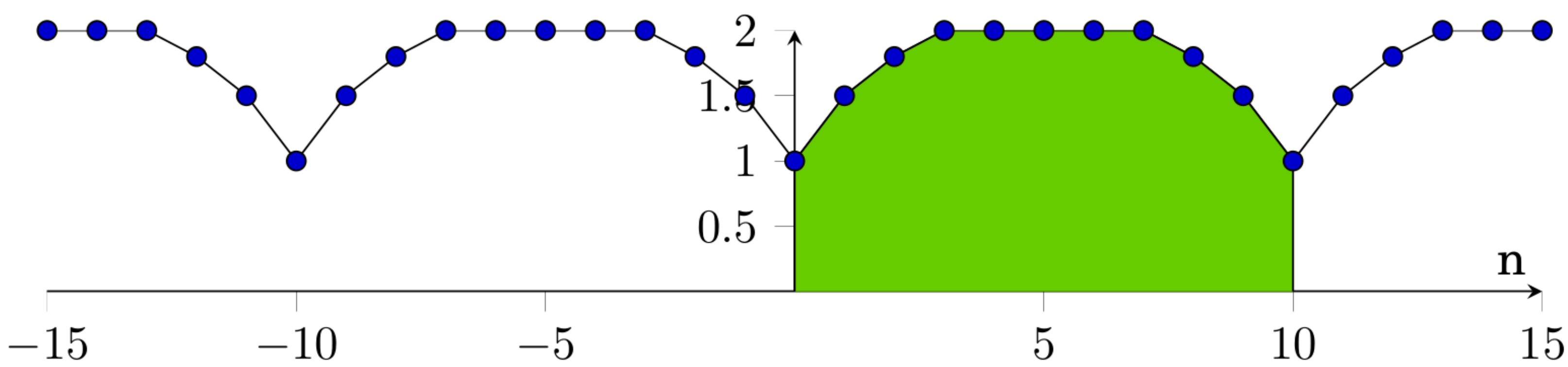


Flip impulse signal: $$\mathrm{x}'(n) = \begin{cases} x(n), & n \geq 0 \\ x(-n), & otherwise \end{cases}$$

Insert into DFT: $$X(f) = \sum_{n=-\infty}^{\infty} x(n) e^{-j2\pi f n}$$

(DFT … Discrete Fourier Transformation)

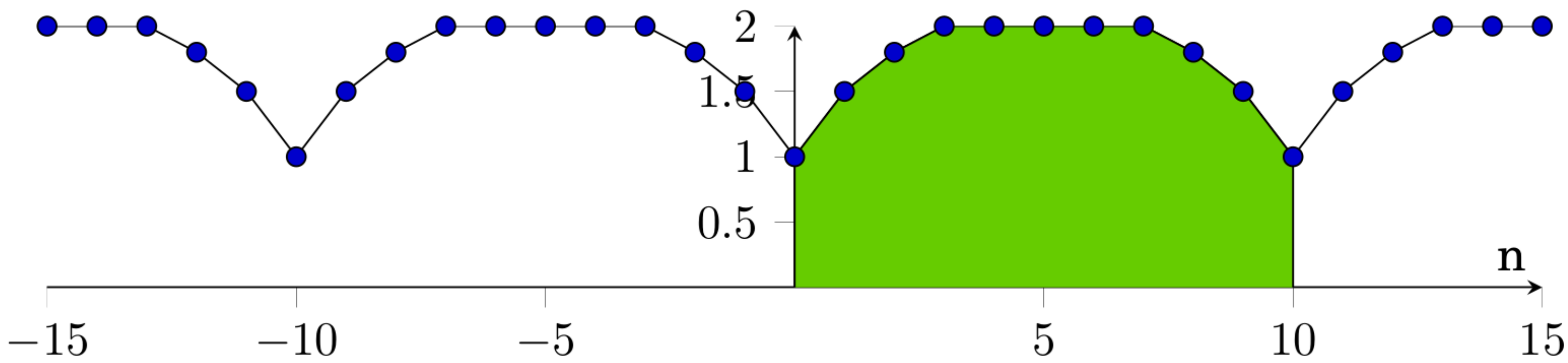
DCT-I (2/4)



$$X(f) = \sum_{n=-N+1}^{N-1} x'(n) e^{-j2\pi \frac{f}{2N-2} n}$$

$$= x(0) + x(N-1) e^{-j2\pi \frac{f(N-1)}{2N-2}} + \sum_{n=1}^{N-2} x(n) (e^{j2\pi \frac{f}{2N-2} n} + e^{-j2\pi \frac{f}{2N-2} n})$$

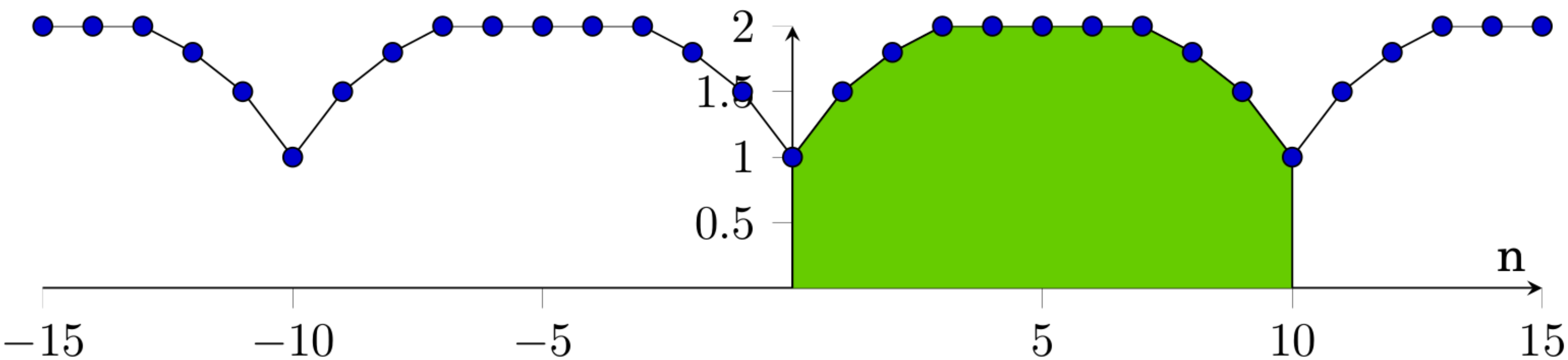
DCT-I (3/4)



$$X(f) = x(0) + x(N-1)e^{-j\pi f} + 2\sum_{n=1}^{N-2} x(n)\cos(\frac{\pi}{N-1}fn)$$

$$= \cancel{2}\left(\frac{1}{2}\Big(x(0) + (-1)^f x(N-1)\Big) + \sum_{n=1}^{N-2} x(n)\cos\Big(\frac{\pi}{N-1}fn\Big)\right)$$

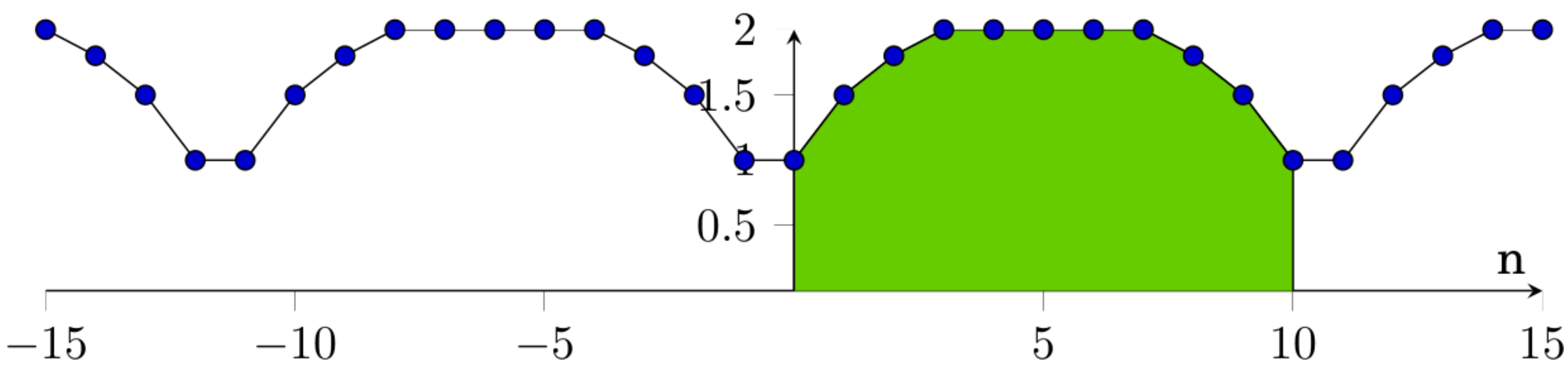
DCT-I (4/4)



Inverse Transformation:

$$x(n) = \frac{2}{N-1} * DCTI$$

DCT-II



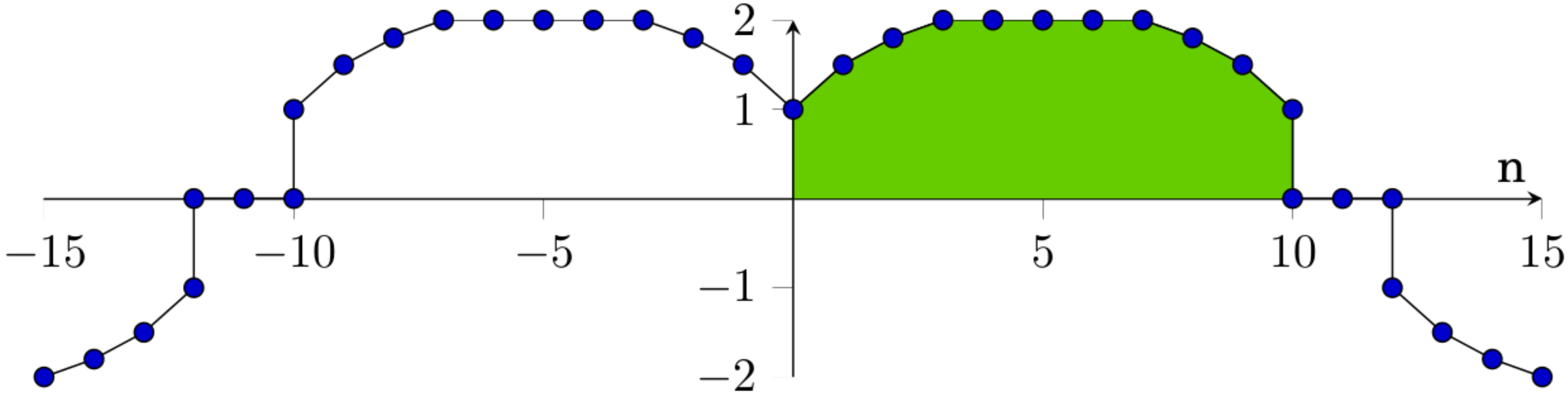
Transformation:

$$x(f) = \sum_{n=0}^{N-1} x(n) \cos\left(\frac{\pi}{N}\left(n + \frac{1}{2}\right) f\right)$$

Inverse Transformation:

$$x(n) = \frac{2}{N} * DCTIII$$

DCT-III



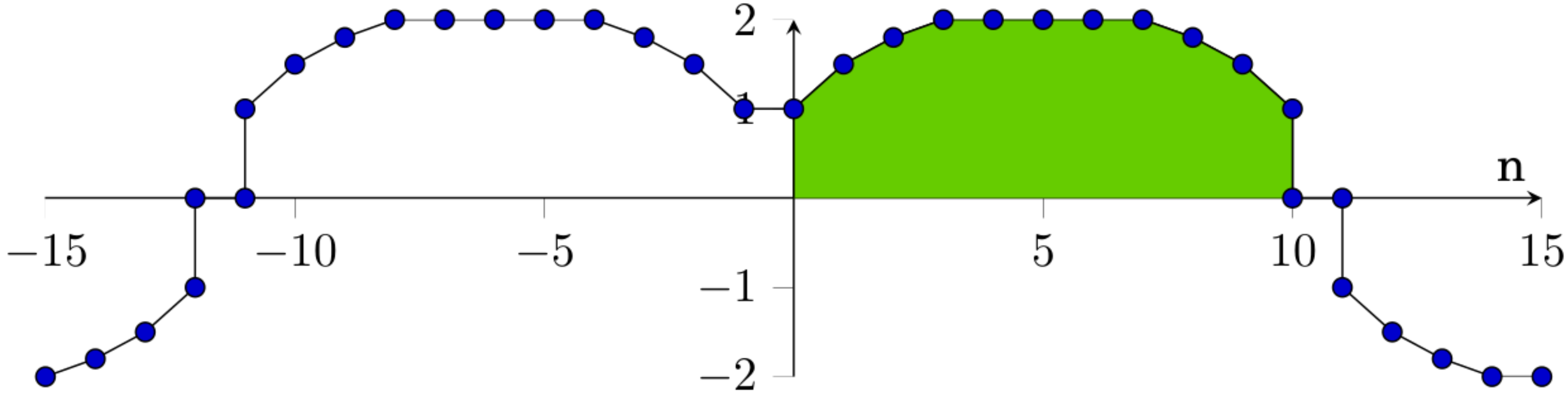
Transformation:

$$x(f) = \frac{1}{2}x(0) + \sum_{n=1}^{N-1} x(n)\cos\left(\frac{\pi}{N}\left(f + \frac{1}{2}\right)n\right)$$

Inverse Transformation:

$$x(n) = \frac{2}{N} * DCTII$$

DCT-IV



Transformation:

$$x(f) = \sum_{n=0}^{N-1} x(n)\cos\left(\frac{\pi}{N}\left(f+\frac{1}{2}\right)\left(n+\frac{1}{2}\right)\right)$$

Inverse Transformation:

$$x(n) = \frac{2}{N} * DCTIV$$

MULTIPLE DIMENSIONS

- Used for jpeg image compression (8x8 matrices)

$$C(k) = \begin{cases} \sqrt{2}^{-1}, & k == 1 \\ 1, & otherwise \end{cases}$$

- Transformation

$$F(x,y) = C(x)C(y)\sum_{m=0}^{7}\sum_{n=0}^{7} f(m,n)\cos\left(\frac{(2m+1)x\pi}{16}\right)\cos\left(\frac{(2n+1)y\pi}{16}\right)$$

- Inverse Transformation

$$f(x,y) = \sum_{m=0}^{7}\sum_{n=0}^{7} C(m)C(n)F(m,n)\cos\left(\frac{(2x+1)m\pi}{16}\right)\cos\left(\frac{(2y+1)n\pi}{16}\right)$$

16

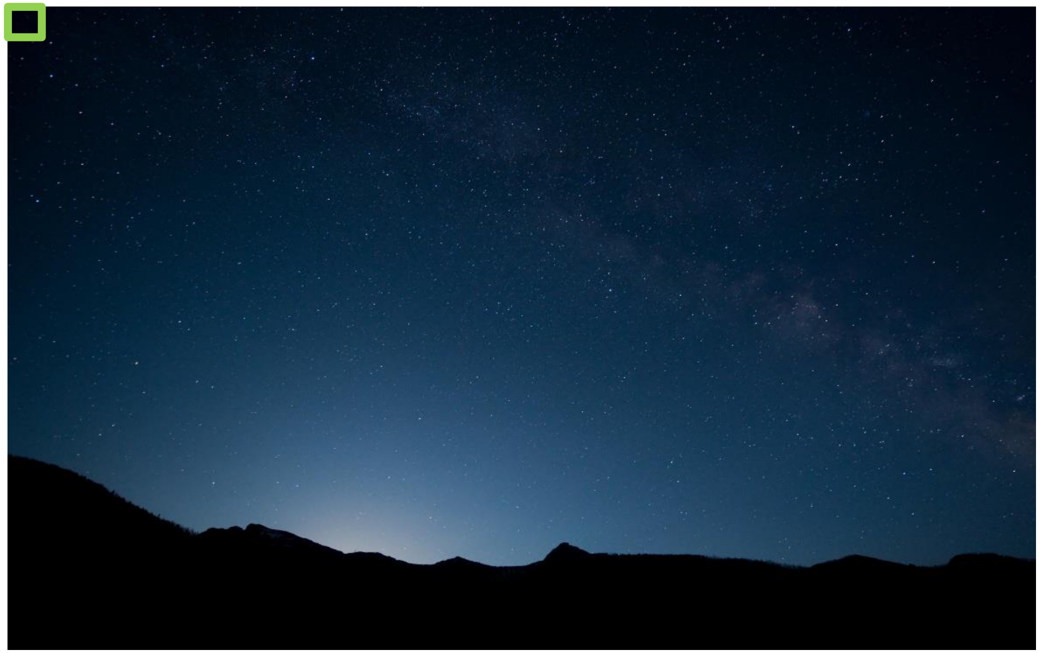
IMAGE COMPRESSION

JPEG ALGORITHM

(Example)

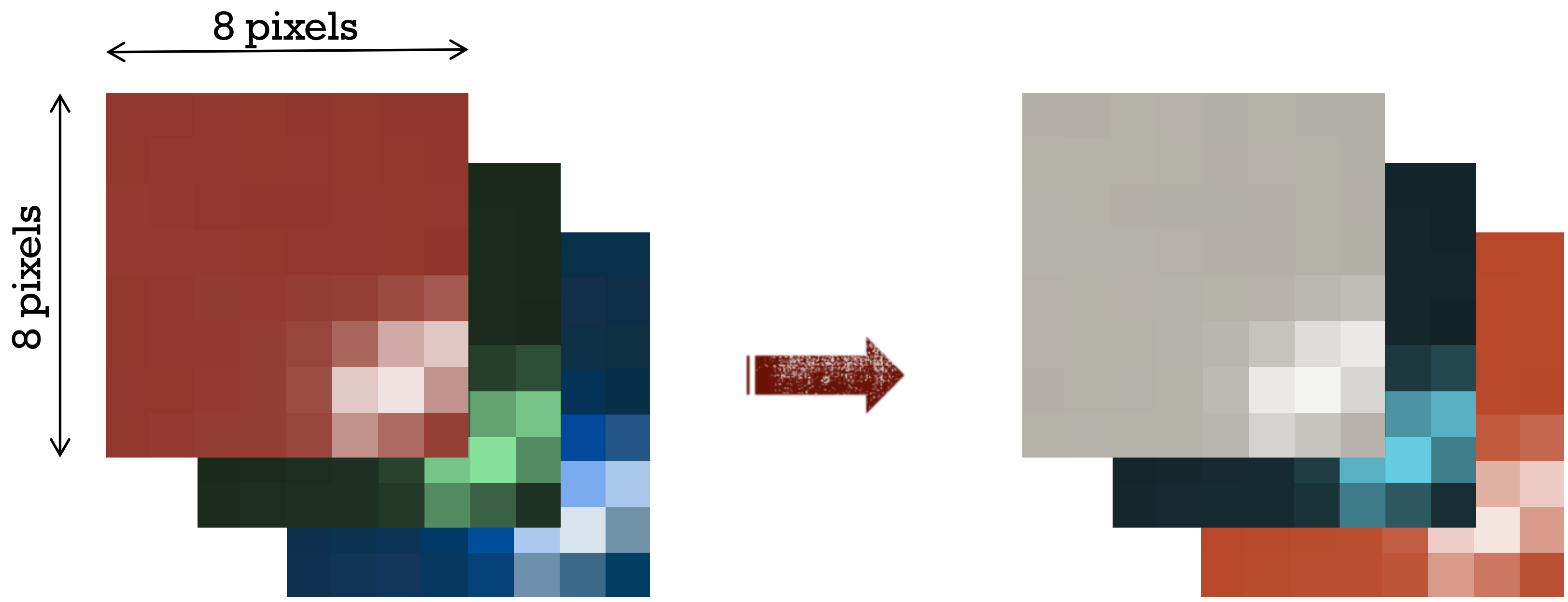
JPEG (1/10)

Selecting one Block



Example Image

JPEG (2/10)



Converting RGB Color Scheme to YCbCr Color Scheme

JPEG (3/10)

139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	156
150	155	160	163	158	156	156	156
159	161	162	160	160	159	159	159
159	160	161	162	162	155	155	155
161	161	161	161	160	157	157	157
162	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

Example values of an 8 by 8 matrix
(which represents one color component of a random image)

JPEG (4/10)

256.6	-1.0	-12.1	-5.2	2.1	-1.7	-2.7	1.3
-22.6	-17.5	-6.2	-3.2	-2.9	-0.1	0.4	-1.2
-10.9	-9.3	-1.6	1.5	0.2	-0.9	-0.6	-0.1
-7.1	-1.9	0.2	1.5	0.9	-0.1	0.0	0.3
-0.6	-0.8	1.5	1.6	-0.1	-0.7	0.6	1.3
1.8	-0.2	1.6	-0.3	-0.8	1.5	1.0	-1.0
-1.3	-0.4	-0.3	-1.5	-0.5	1.7	1.1	-0.8
-2.6	1.6	-3.8	-1.8	1.9	1.2	-0.6	-0.4

DCT applied on the values

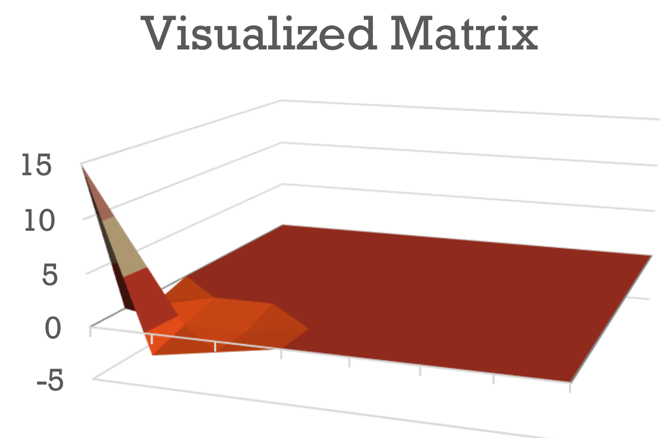
JPEG (5/10)

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

Quantization Table

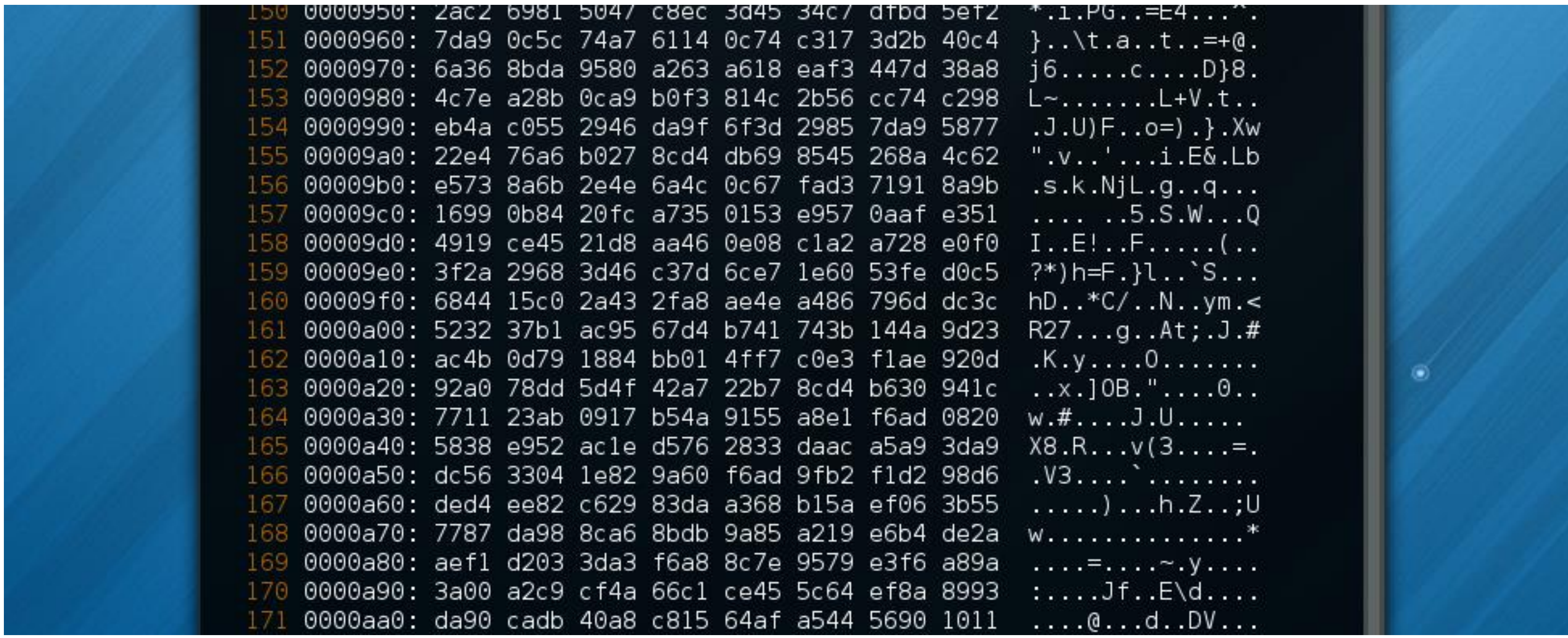
JPEG (6/10)

15	0	-1	0	0	0	0	0
-2	-1	0	0	0	0	0	0
-1	-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	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0



Values after Quantization

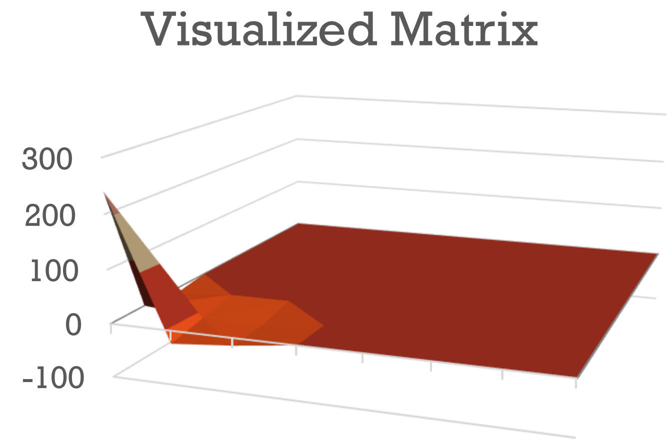
JPEG (7/10)



Further Compression (Huffman-Coding, Run-Length-Encoding)
Storing the Data in a JFIF-Container
(Details will follow in Implementation-Chapter)

JPEG (8/10)

240	0	-10	0	0	0	0	0
-24	-12	0	0	0	0	0	0
-14	-13	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0



Multiply values with Quantizationvalues

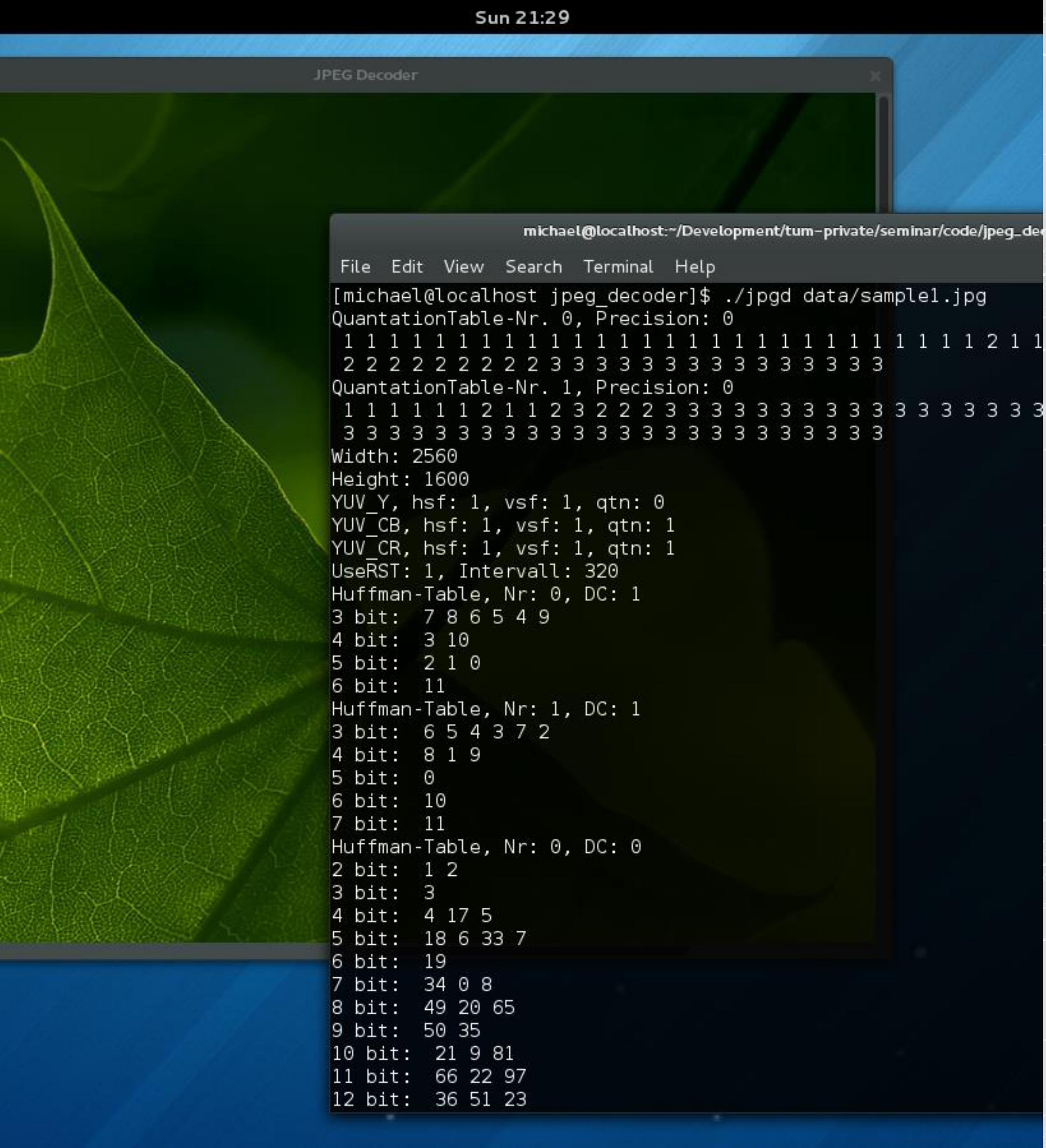
JPEG (9/10)

144	146	149	152	154	156	156	156
148	150	152	154	156	156	156	156
155	156	157	158	158	157	156	155
160	161	161	162	161	159	157	155
163	163	164	163	162	160	158	156
163	164	164	164	162	160	158	157
160	161	162	162	162	161	159	158
158	159	161	161	162	161	149	158

Decoded Values after IDCT

JPEG (10/10)

Decoded Image without visual recognisable differences
(For the human eye, actual differences exist)



JPEG-DECODER

IMPLEMENTATION

JPEG-DECODER (1/2)

- C++11, Clang/llvm, Gtkmm 3, Cairomm

Features:

- Decodes all Baseline-DCT JPEG Files (f.e. from your camera)
- Supports Super-Sampling
- Uses only the Integer-Core, doesn‘t require FPU

Stats:

- ~200ms to decode a picture with 2560x1600 pixel
- 1100 Lines of Code (without comments and blank lines)

JPEG-DECODER (2/2)

The most interesting parts:

- Parser for Header-Information
- Tree used for Huffman-Decoding
- Bit-Stream
- Value Parsing

JFIF-PARSER (1/2)

- JPEG Encoded Data stored in JFIF-Container
- Copy of File in Memory
- Stream-Length Validation
- Structure:

JFIF-PARSER (2/2)

Marker 2 Bytes	Length 2 Bytes	|Length|-2 Bytes Payload

2 Bytes 2 Bytes Bytes Payload

- **A Few Common Markers:**

  0xFFD8 … Start of Image

  0xFFC0 … Baseline DCT

  0xFFDA … Start of Scan

  0xFFC4 … Huffman-Tables

  0xFFDB … Quantization-Tables

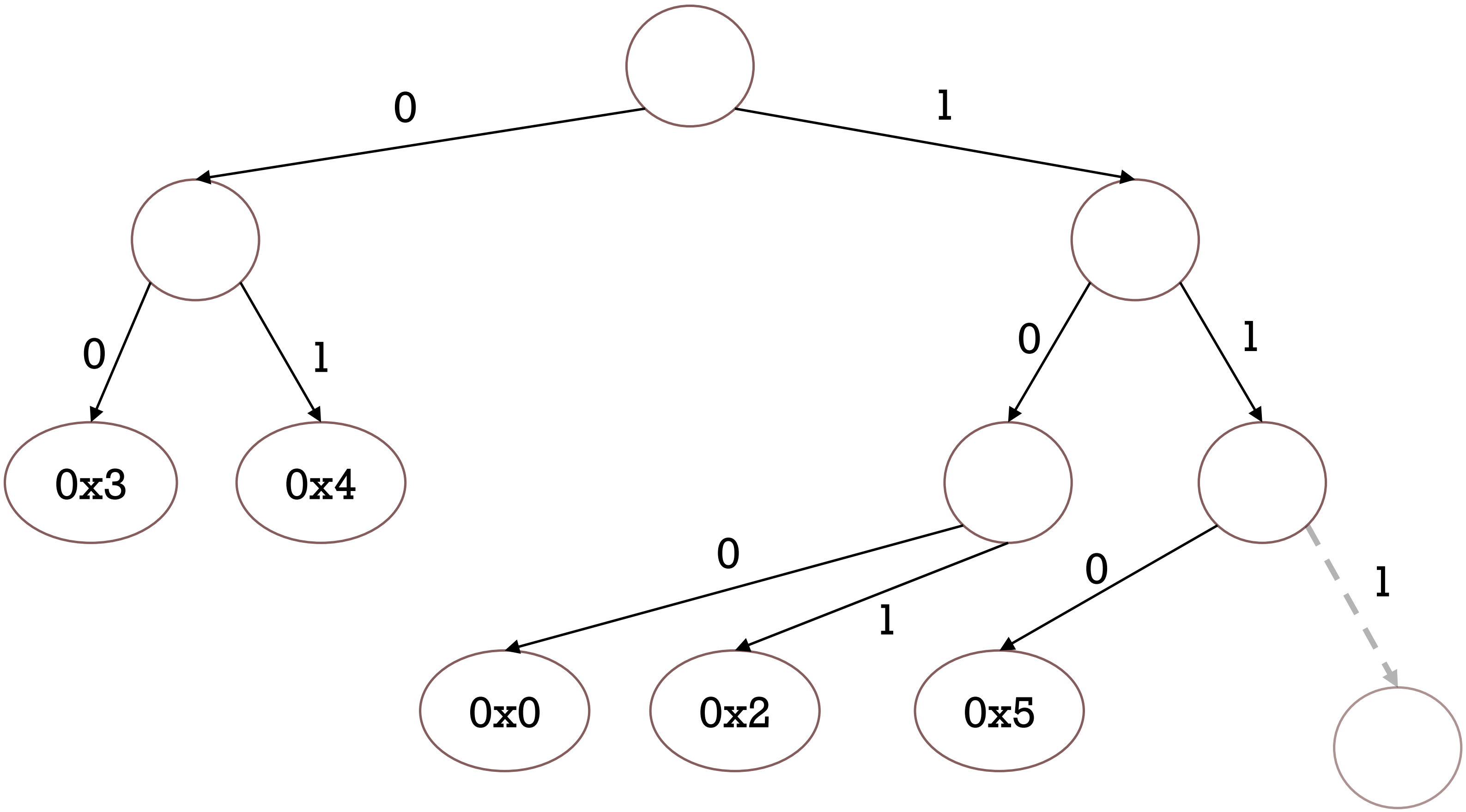
  0xFFD9 … End Of Image

VALUE PARSING

- MCU … Minimum Coded Unit contains at least one 8x8 pixel matrix including for three color components

- Every value in the bitstream contains the following information
  - Path to a value in the Huffman-Tree
    This value contains the amount of preceding zeros in the matrix (upper 4 bits) and the length of the actual value (lower 4 bits)
  - Actual Value (Signed and Unsigned values supported, doesn‘t use two‘s complement)

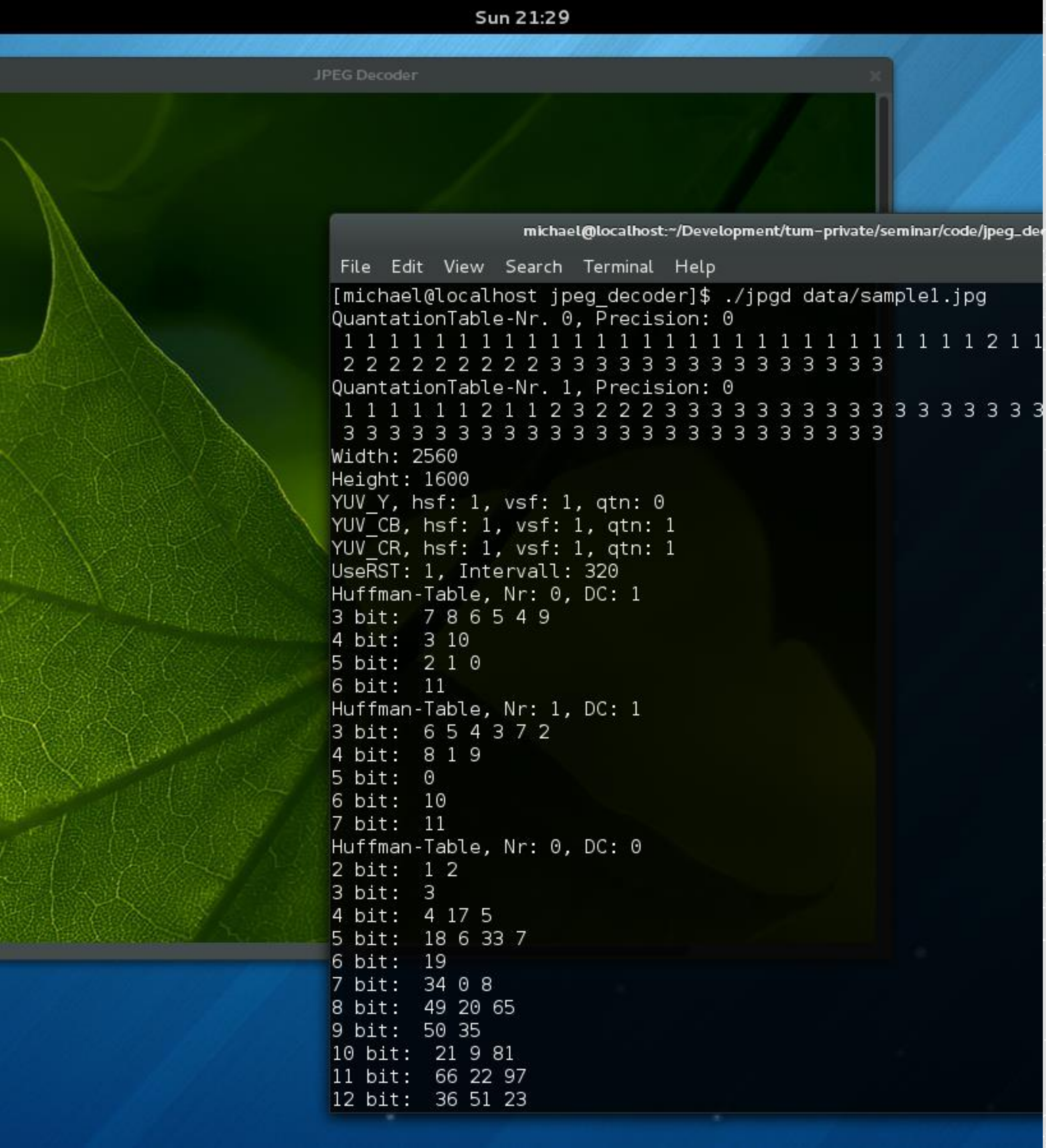
DECODING HUFFMAN-TREE



BIT-STREAM

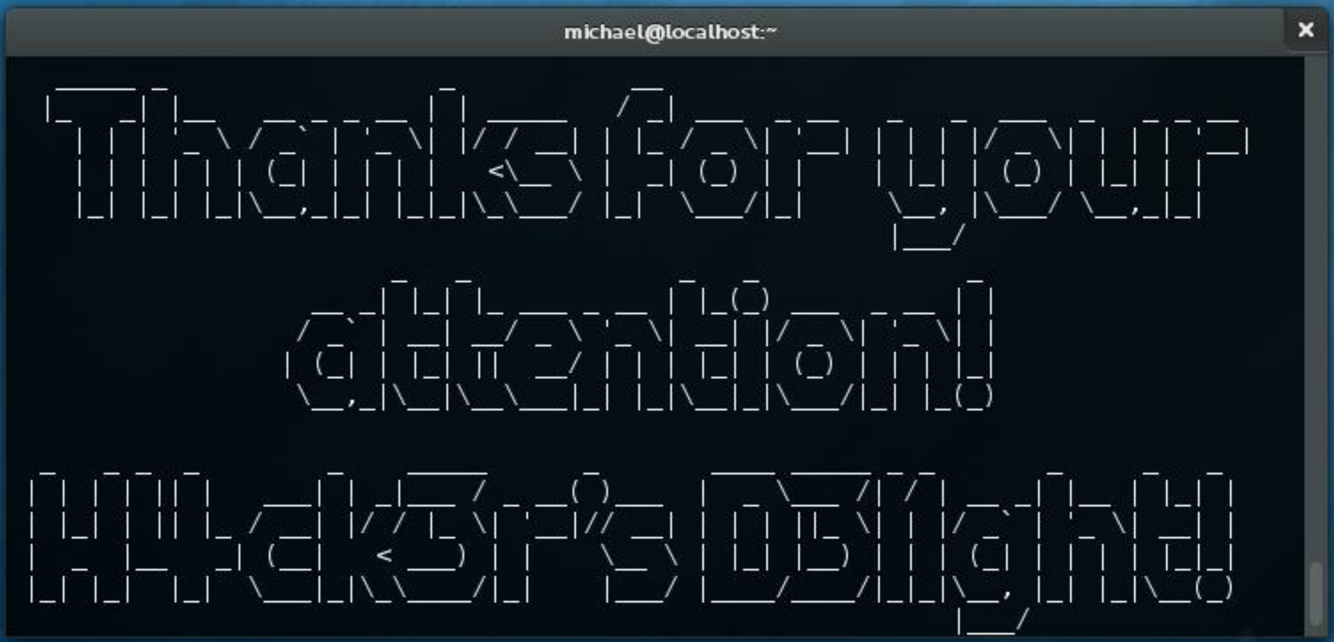
Bit-Stream:

- Skips Stuffing-Bytes
- Handles Bit-Position
- Transparently used in Main-Parsing-Logic and in Huffman-Tree



JPEG-DECODER

DEMO



https://github.com/MichaelE1000/in0013
or http://goo.gl/A4pW6