Ukládání a komprese obrazu

Pavel Strachota

FJFI ČVUT v Praze

6. listopadu 2022

Obsah

Obsah

Reprezentace obrazu

- **vektorová** - popis geometrických objektů a jejich vlastností (polohy, barvy, překrývání atd.)
  - nezávisí na rozlišení zobrazovacího zařízení
  - lze libovnolně škálovat při zachování kvality
- **rastrová** - popis obrazu pomocí matice pixelů
  - konečné rozlišení
  - vysoká paměťová náročnost $\Longrightarrow$ snaha o **kompresi** informace

Obsah

Reprezentace barvy

- černobílý obraz - 1 bit na pixel
- šedotónový obraz - 8 bitů na pixel, 256 stupňů šedi
- **RGB** reprezentace
  - 5+6+5 bitů (565 High Color mode)
  - 8+8+8 bitů True Color mode
- reprezentace v jiném barevném prostoru (viz dále)
- **indexovaná barva** - ukládá se index do palety
- **paleta**
  - pevně daná (např. standardní VGA paleta)
  - **adaptivní** (vhodně přizpůsobená obrázku, uložená v souboru spolu s obrázkem)
  - typicky 4, 16, 256 barev

Použití pevné palety

- barvy reprezentovány body v 3D prostoru RGB nebo jiném
- často se používá L*a*b* (CIELAB 1976)
- problém: každému pixelu o obecné barvě přiřadit barvu z palety
- paleta n barev o složkách $[R_i, G_i, B_i]$
- vzdálenost barev měříme euklidovskou mírou
- barvu $[R, G, B]$ nahradíme barvou palety s indexem

$$\operatorname*{argmin}_{i \in \mathbb{N}_n} \sqrt{(R - R_i)^2 + (G - G_i)^2 + (B - B_i)^2}$$

- oblasti prostoru RGB nahrazené i-tou barvou palety jsou **Voroného diagramy** (viz *výpočetní geometrie*)

Tvorba adaptivní palety

- kvantizace barev pomocí algoritmů pro **shlukování** (*clustering*) bodů v 3D prostoru
- body v prostoru = množina všech barev obsažených v obrázku

Vektorová kvantizace k-means

Spustit demo

1. náhodně umísti k tzv. **centroidů** do prostoru (budoucí barvy palety)
2. $\forall i \in \{1, 2, \ldots, k\}$ najdi množiny bodů M_i, kterým je nejblíže i-tý centroid
3. $\forall i$ přemísti i-tý centroid do těžiště množiny M_i
4. pokud se alespoň jeden centroid pohnul, jdi na 2

- náhodná inicializace nemusí vždycky zafungovat (centroid zůstane „sám")
- k-means++ : lepší algoritmus pro inicializaci k-means

Tvorba adaptivní palety

- kvantizace barev pomocí algoritmů pro **shlukování** (*clustering*) bodů v 3D prostoru
- body v prostoru = množina všech barev obsažených v obrázku

Vektorová kvantizace k-means Spustit demo

1. náhodně umísti k tzv. **centroidů** do prostoru (budoucí barvy palety)
2. $\forall i \in \{1, 2, \ldots, k\}$ najdi množiny bodů M_i, kterým je nejblíže i-tý centroid
3. $\forall i$ přemísti i-tý centroid do těžiště množiny M_i
4. pokud se alespoň jeden centroid pohnul, jdi na 2

- náhodná inicializace nemusí vždycky zafungovat (centroid zůstane „sám")
- k-means++ : lepší algoritmus pro inicializaci k-means

Tvorba adaptivní palety

- kvantizace barev pomocí algoritmů pro **shlukování** (*clustering*) bodů v 3D prostoru
- body v prostoru = množina všech barev obsažených v obrázku

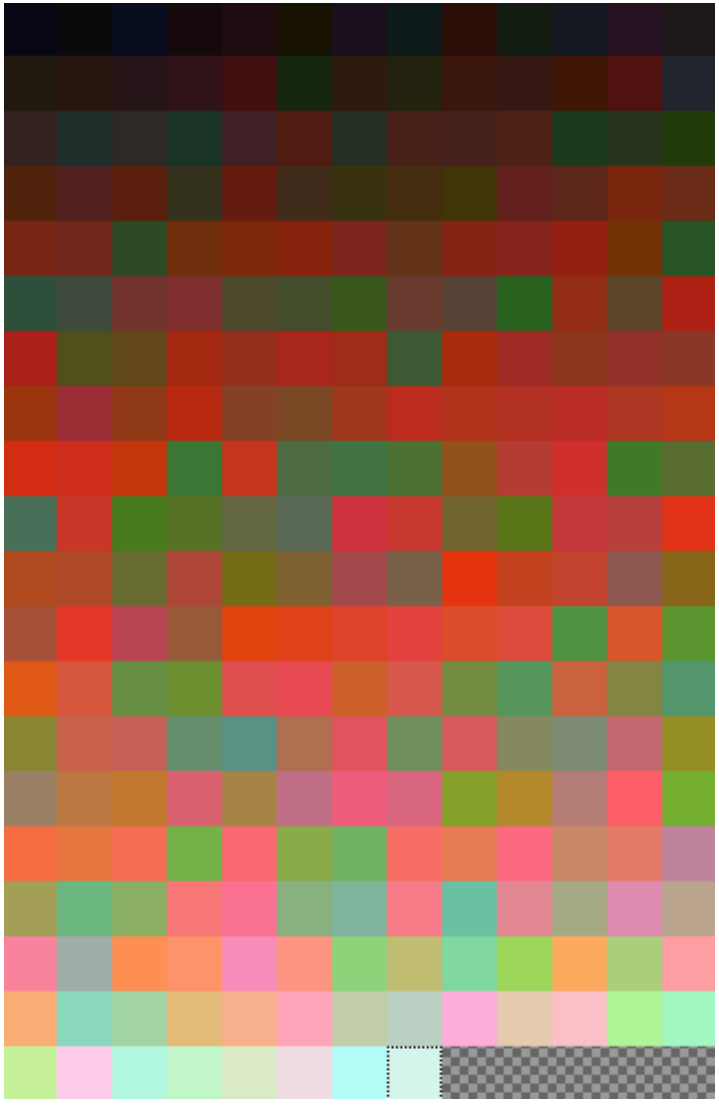
Algoritmus *median cut*

1. začni s jedním kvádrem obsahujícím všechny body
2. rozděl kvádr s absolutně nejdelší stranou podél této strany na 2 části tak, aby v každé ležela **polovina** bodů ($\Longrightarrow$ hledání mediánu)
3. vzniklé poloviny kvádru zmenši co nejvíce, aby se do nich ještě vešly dané body
4. pokud je počet kvádrů menší než velikost palety n, jdi na 2
5. i-tá barva palety se najde např. jako průměr, resp. medián bodů v i-tém kvádru

Adaptivní paleta

Originál

Adaptivní paleta



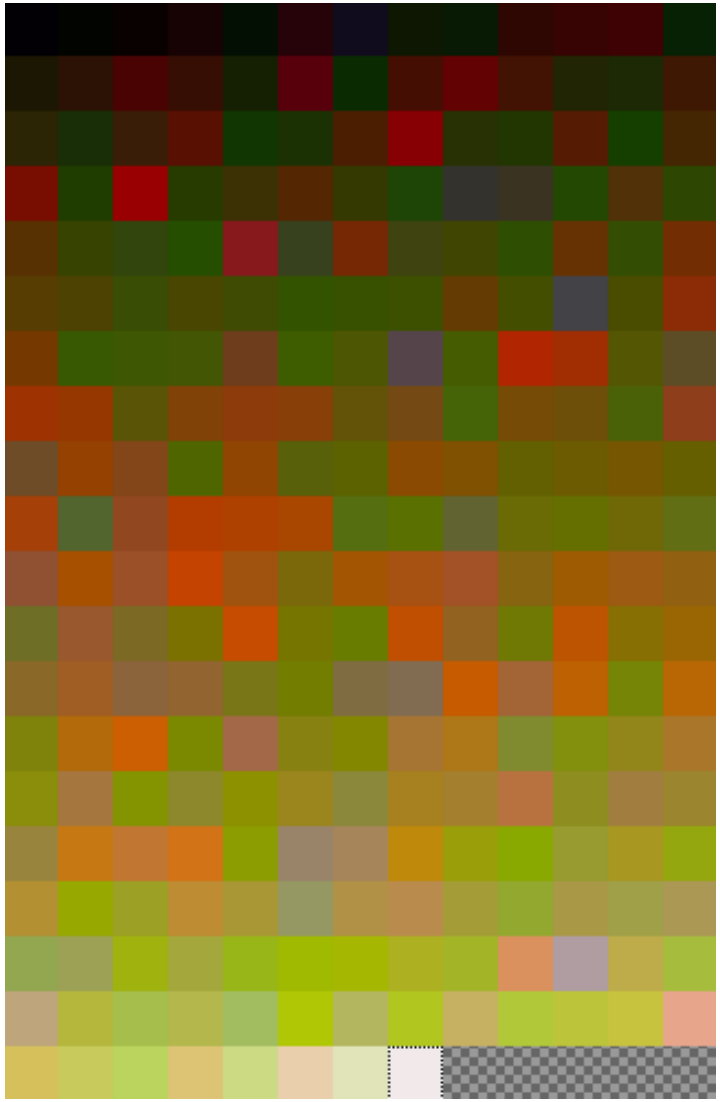
Adaptivní paleta

VGA paleta

Adaptivní paleta

Originál

Adaptivní paleta



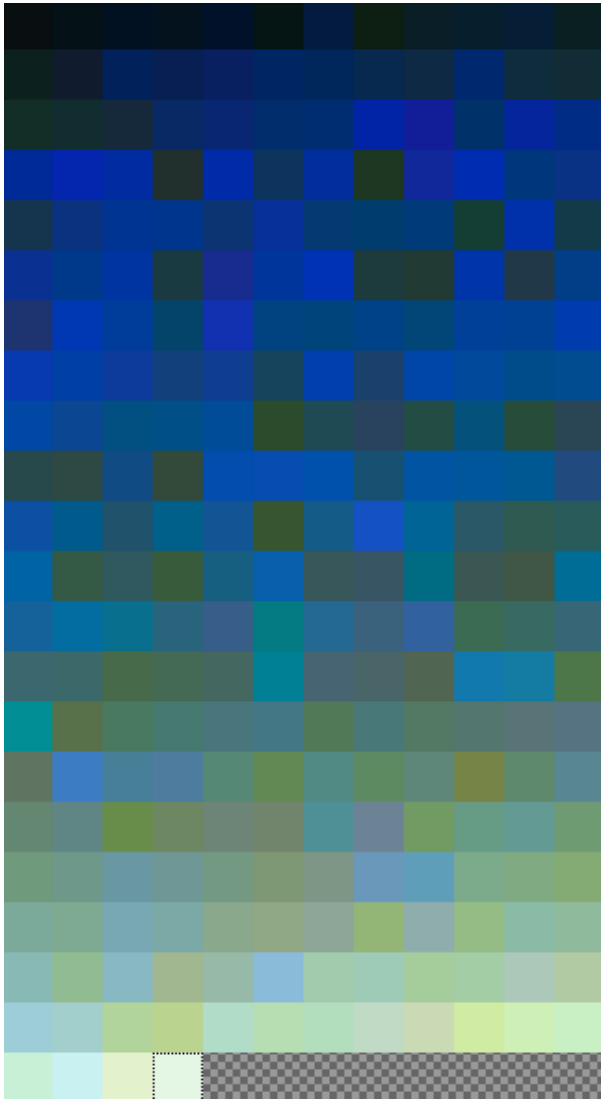
Adaptivní paleta

VGA paleta

Adaptivní paleta

Originál

Adaptivní paleta



Adaptivní paleta

VGA paleta

Obsah

Komprese dat

- snaha najít co nejúspornější zápis pro nějakou zprávu
- sekvence znaků (bytů) zapsány pomocí jiných, pokud možno kratších sekvencí $\Longrightarrow$ kódování
- efektivita komprese závislá na povaze zpracovávaných dat
- **bezztrátová komprese** (*lossless compression*)
  - aplikovatelná na libovolná data
  - možnost přesně rekonstruovat původní zprávu
- **ztrátová komprese** (*lossy compression*)
  - lze použít jen na subjektivně vnímaná data (obraz, video, zvuk)
  - možnost rekonstruovat původní zprávu **jen přibližně**
  - využití znalosti lidského vnímání $\Longrightarrow$ efektivní komprese při zachování malého **subjektivně vnímaného** rozdílu mezi původní a rekonstruovanou zprávou

Kódování RLE

- ***R**un-**L**ength **E**ncoding*
- jednoduchá metoda efektivní pro obrázky s oblastmi vyplněnými stejnou barvou
- posloupnost opakujících se bytů (resp. dvojic, trojic bytů) nahradí jednou hodnotou s uvedením počtu opakování

0	value

zápis 7-bitové hodnoty

1	0000000	value

zápis hodnoty $\geq 10000000_2$

1	counter	value

zápis hodnoty opakující se $(1 + \texttt{counter})$-krát

- pokud se v sousedních pixelech ve směru ukládání (obvykle po řádcích) často neopakují stejné hodnoty
  $\Longrightarrow$ **záporná komprese** (výsledek delší než originál)

Huffmanovo kódování

- statistická metoda
- teoretický zdroj informace generující konečný počet zpráv se **známým rozdělením pravděpodobnosti**
- kódování celých zpráv D-znakovými **kódovými slovy** (obsahujícími znaky $0, 1, 2, \ldots, D-1$)
- nejpravděpodobnější zprávy kódovány nejkratšími kódy a naopak
- Huffmanův kód je **optimální** (minimalizující střední délku kódu) mezi instantními jednoznačně dekódovatelnými kódy
- **instantní** kód = kódové slovo žádné zprávy **není prefixem** kódového slova jiné zprávy
  - postupně procházíme sekvenci znaků
  - jakmile narazíme na řetězec, který je kódovým slovem, lze ihned dekódovat
- nejčastěji binární Huffmanův kód ($D = 2$)

Huffmanovo kódování

Konstrukce kódu

- původně n různých samostatných (jednoduchých) zpráv
- pracujeme se **sdruženými zprávami** = množinami zpráv
- každé jednoduché zprávě odpovídá **kódové slovo**
- přiřadit sdružené zprávě M kódový znak = přidat jej na začátek kódových slov všech jednoduchých zpráv v M

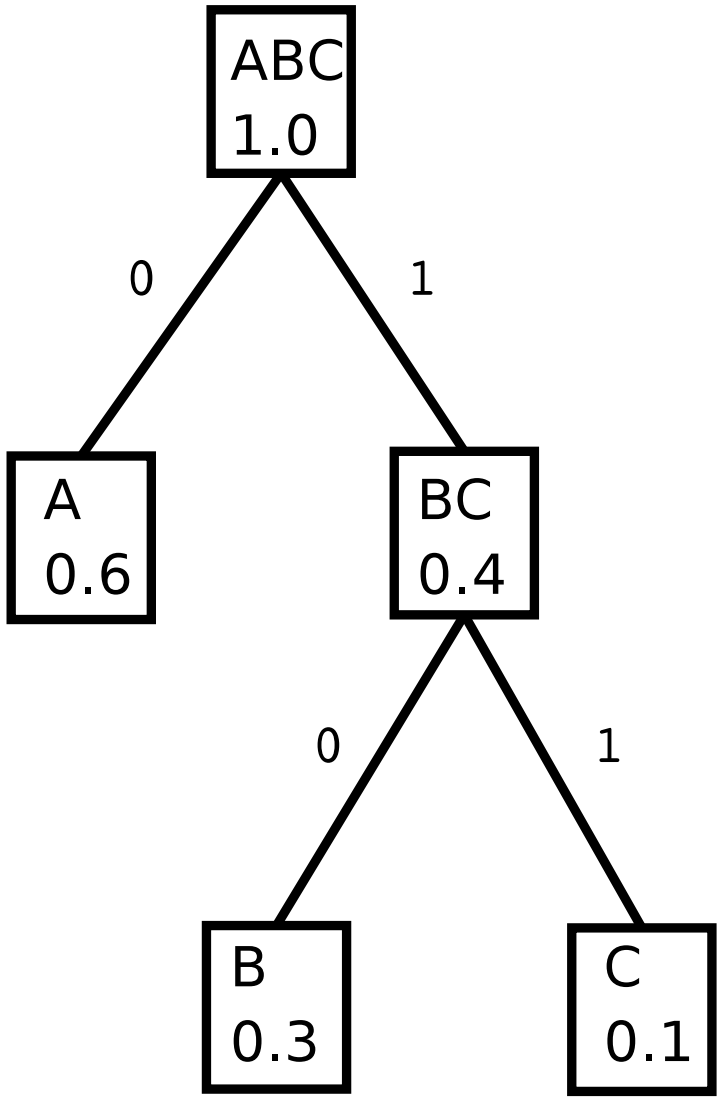
Algoritmus

1. zpočátku n samostatných zpráv
2. seřaď zprávy podle pravděpodobnosti
3. vyber D zpráv s nejnižší pravděpodobností a přiřaď jim kódové znaky $0, 1, 2, \ldots, D-1$
4. tyto zprávy spoj do sdružené zprávy s pravděpodobností danou součtem pravděpodobností původních zpráv
5. dokud existuje více než 1 zpráva, jdi na 2

Huffmanovo kódování

Binární kód a jeho reprezentace binárním stromem

- binární kód $\Longrightarrow$ v každé iteraci spojíme vždy 2 zprávy
- spojování zpráv = vytváření **binárního stromu** „odspodu“ od listů
  - uzel - sdružená zpráva
  - jeho potomci - zprávy, z nichž vznikl
  - levému potomku přiřazen kódový znak 0, pravému potomku znak 1
  - listy - jednoduché zprávy
- **dekódování zprávy** - procházení stromem od kořene doleva, resp. doprava, pokud ze vstupu čteme 0, resp. 1, až dojdeme k listu



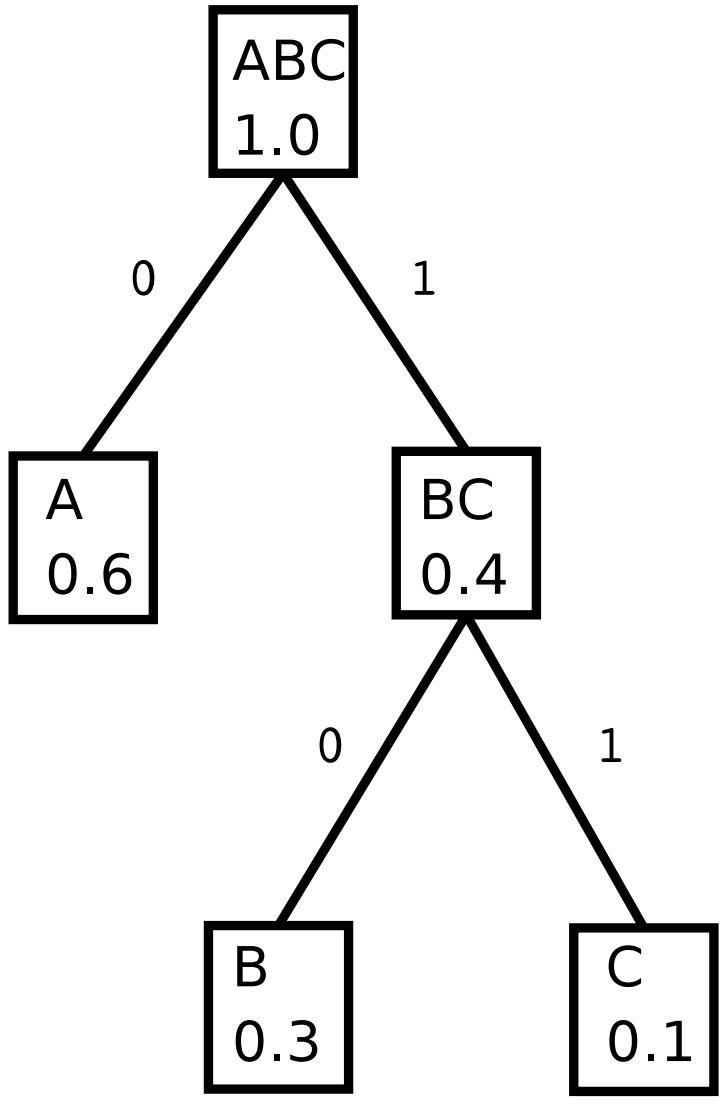
Huffmanovo kódování

Binární kód a jeho reprezentace binárním stromem

- 3 zprávy A,B,C

Zpráva	Pravděpodobnost	Kód
A	0.6	0
B	0.3	10
C	0.1	11

- nelze např. aby zpráva B měla kódové slovo jen „1“, protože výsledný kód by nebyl instantní („1“ je prefixem „11“)



Huffmanovo kódování v praxi 1/3

- **zprávy** - sekvence bitů v obrázku, resp. obecných datech
- jejich pravděpodobnosti odhadnuty (napevno určeny) statistickým zpracováním typických dokumentů
- standardy CCITT (Comité Consultatif International Téléphonique et Télégraphique)
- původně pro **přenos černobílých dokumentů faxem**

Huffmanovo kódování v praxi 2/3

standard **G31D** (*group 3 1-dimensional*):

- zakódování pomocí RLE → hodnoty čítače opakování kódovány dále pomocí Huffmanova kódu
- krátké kódy pro typické úseky bílých, resp. černých pixelů a dále:
  - kód `FILL` (stejná barva až do konce řádku)
  - kód `EOL` (konec řádku) $\Longrightarrow$ synchronizace, odolnost proti chybám
  - kód `RTC` (*return to control*) - konec dat
- možno dekódovat bez vyrovnávací paměti

Huffmanovo kódování v praxi 2/3

standard **G32D** (*group 3 2-dimensional*):

- kódování pozic změny barvy, relativně vůči předchozí pozici
- uvažuje se i změna oproti předchozímu řádku (ve vertikálním směru)
- každý K-tý řádek se kóduje pomocí G31D kvůli spolehlivosti
- mnoho detailů a ošetření chyb při přenosu

standard **G42D** (*group 4 2-dimensional*):

- v podstatě G32D bez mechanismů odstraňování chyb
- použití ve formátu TIFF

Kódování LZW

- Lempel-Ziv-Welch, 1984
- odvozeno od **LZ77** (Jakob Ziv, Abraham Lempel, 1977)
- metoda vhodná pro zcela obecná data
- implementovaná v programu **`compress`** a ve formátu GIF
- snaha efektivně komprimovat data, kde neznáme pravděpodobnost zpráv (např. symbolů atd.)
- princip: **slovníkové kódování** (*dictionary based encoding*)

Kódování LZW

Princip metody 1/3

- slovník mapující bitové kódy na řetězce bytů (slova)
- slovník se tvoří **dynamicky ze vstupních dat**
- kódy mají délku 9-12 bitů
  (max. $2^{12} = 4096$ položek slovníku)
- komprimovaný soubor je posloupnost bitových kódů (přes hranice bytů)
- na počátku je délka kódu 9 bitů, slovník obsahuje 258 položek
  - prvních 256 položek (kódy $0 - 255$) jsou jednobytové „řetězce“ (kód se shoduje s 1-bytovou hodnotou)
  - kód 256 je „*clear code*“ - kód pro vyprázdnění slovníku
  - kód 257 je „*end of data*“

Kódování LZW

Princip metody 2/3

Algoritmus komprese LZW

1. $w = \emptyset$ (prázdné slovo)
2. načti znak K ze vstupu
3. pokud je wK ve slovníku, proveď $wK \rightarrow w$
4. jakmile wK není ve slovníku:
   - vypiš na výstup kód C slova w
   - wK přidej do slovníku pod novým kódem
   - $K \rightarrow w$
5. jdi na 2

- pokud v bodu 4 nestačí délka kódu, zvýší se o 1 bit, max. však na 12 bitů
- když se slovník zaplní, vyprázdní se (až na prvních 258 položek), vypíšeme *clear code* a začneme od začátku

Kódování LZW

Princip metody 3/3

- slovník **není obsažen** v komprimovaných datech, při dekompresi jej lze **rekonstruovat**
- právě při dekompresi je potřeba *clear code*
- při kompresi se kód na výstupu objevil vždy spolu s novým zápisem do slovníku
- **totéž** se bude dít při dekompresi
- délka kódu (kolik bitů číst ze vstupu) se bude automaticky řídit délkou slovníku
  (v následujícím se délkou kódu ani vyprázdněním slovníku nezabýváme)

Kódování LZW

Princip metody 3/3

Myšlenkový postup **de**komprese LZW

1. načti kód C ze vstupu
2. první kód určitě ve slovníku je
3. vypiš na výstup odpovídající slovo w
   - **vím**, že enkodér vypsal C, protože narazil na takový znak K, že wK nebylo ve slovníku (a wK zařadil do slovníku)
   - znak K **ještě neznám**, ale je to první znak slova, které kóduje následující kód
     $\Longrightarrow$ příště zjisti K a slovo wK zařaď do slovníku pod novým kódem.
4. jestliže však příští kód **není** ve slovníku:
   - **vím**, že enkodér musel tento kód právě vygenerovat
     $\Longrightarrow$ je to kód odpovídající slovu wK a tedy K je zároveň **prvním znakem** w
   - přidej wK do slovníku pod tímto kódem, $wK \rightarrow w$ a jdi na 3

Kódování LZW

Princip metody 3/3

Algoritmus **de**komprese LZW

1. načti první kód C_1 ze vstupu a najdi k němu ve slovníku slovo w_1
2. načti kód C_2 ze vstupu
3. vypiš w_1 na výstup
4. pokud C_2 **není** ve slovníku
   - přidej do slovníku slovo $w_1 K_1$, kde K_1 je první znak w_1
5. pokud C_2 **je** ve slovníku
   - najdi slovo w_2 odpovídající C_2
   - přidej do slovníku slovo $w_1 K_2$, kde K_2 je první znak w_2
6. $C_2 \rightarrow C_1$, $w_2 \rightarrow w_1$ a jdi na 2

Kódování LZW - příklad

Předpoklady

- znaky v datech jsou jen velká písmena anglické abecedy
- na začátku je ve slovníku 26 písmen + znak # (ukončení zprávy)
- na začátku používáme 5-bitový kód

```
# = 00000 = 0
A = 00001 = 1
B = 00010 = 2
C = 00011 = 3
⋮
Z = 11010 = 26
```

- kódujeme řetězec

TOBEORNOTTOBEORTOBEORNOT#

Kódování LZW - příklad

Kódování

TOBEORNOTTOBEORTOBEORNOT#

```
Current  Next     Output      Extended
Sequence Char  Code   Bits    Dictionary
-------  ----  ---  ------    ----------
NULL     T
T        O     20 =  10100    27:     TO <-- 27 = first available code after 0 through 26
O        B     15 =  01111    28:     OB
B        E      2 =  00010    29:     BE
E        O      5 =  00101    30:     EO
O        R     15 =  01111    31:     OR
R        N     18 =  10010    32:     RN <-- 32 requires 6 bits, so for next output use 6 bits
N        O     14 = 001110    33:     NO
O        T     15 = 001111    34:     OT
T        T     20 = 010100    35:     TT
TO       B     27 = 011011    36:    TOB
BE       O     29 = 011101    37:    BEO
OR       T     31 = 011111    38:    ORT
TOB      E     36 = 100100    39:   TOBE
EO       R     30 = 011110    40:    EOR
RN       O     32 = 100000    41:    RNO
OT       #     34 = 100100                <--- # stops the algorithm; send the cur seq
                0 = 000000                     and the stop code

Unencoded length = 25*5       = 125 bits
Encoded length   = 6*5 + 11*6 =  96 bits.
```

Kódování LZW - příklad

Dekódování

TOBEORNOTTOBEORTOBEORNOT#

Input Bits	Code	Output Sequence	New Dictionary Entry Full		Conjecture	
10100 =	20	T			27: T?	
01111 =	15	O	27: TO		28: O?	
00010 =	2	B	28: OB		29: B?	
00101 =	5	E	29: BE		30: E?	
01111 =	15	O	30: EO		31: O?	
10010 =	18	R	31: OR	<---	32: R?	----- created code 31 (last to fit in 5 bits);
001110 =	14	N	32: RN		33: N?	so start using 6 bits
001111 =	15	O	33: NO		34: O?	
010100 =	20	T	34: OT		35: T?	
011011 =	27	TO	35: TT		36: TO?	
011101 =	29	BE	36: TOB	<---	37: BE?	---- 36 = TO + 1st symbol (B) of
011111 =	31	OR	37: BEO		38: OR?	next coded sequence received (BE)
100100 =	36	TOB	38: ORT		49: TOB?	
011110 =	30	EO	39: TOBE		40: EO?	
100000 =	32	RN	40: EOR		41: RN?	
100100 =	34	OT	41: RNO		42: OT?	
000000 =	0	#				

Další varianty a vylepšení LZ77

- **LZ78** - upravený LZ77 s predikcí
- **LZC** - kontroluje efektivitu komprese, při klesající efektivitě smaže slovník
- **DEFLATE** - kombinace LZ77 a Huffmanova kódování (velmi populární: **`zip`**, **`gzip`**, PNG)
- **LZMA** - efektivní algoritmus použitý v programu **`7zip`**
- **LZX** - modifikace používaná v souborech „cabinet“ (**`.cab`**) a „MS compressed HTML“ (**`.chm`**)
- **LZJB** - souborový systém ZFS
- ...

Obsah

Přehled populárních formátů 1/4

- **BMP** (Windows Bitmap) - jednoduchý formát původně pro OS/2 a Windows 3.0
  - nekomprimovaná data v různých barevných hloubkách
  - hlavička + sekvenčně po řádcích uložené hodnoty barev pixelů
- **PNM** (Portable AnyMap) - formát podobný BMP, dobře zdokumentovaný v manuálových stránkách UNIXu
  - hlavička uložená vždy v ASCII začíná „magickým číslem“ - rozlišuje typ
  - typy: monochromatický (PGM), barevný (PPM), oba v binární i ASCII (!!) formě
  - snadno lze naprogramovat vlastní funkce pro čtení / zápis
  - před vykreslením nutné provést **gama korekci**, jinak bude obrázek vypadat tmavší
- **PCX** - jeden z nejstarších formátů, používá kompresi RLE

Přehled populárních formátů 2/4

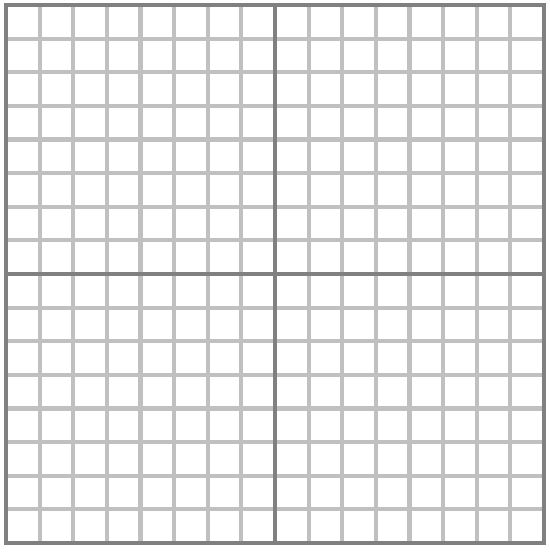
- **GIF** (Graphics Interchange Format)
  - omezení na 256 barev
  - používá LZW kompresi
  - umožňuje ukládat **sekvence snímků** (animace vhodné např. pro webové stránky)
    - typ snímku (rozdílový snímek, plný snímek)
    - časování
  - podporuje průhlednost (jedna barva definována jako průhledná)
  - textové informace

Přehled populárních formátů 3/4

- **TIFF** (Tag(ged) Image File Format)
  - původně pro ukládání černobílých snímků (ze scanneru, faxu) - kódování G42D
  - nyní ve verzi 6.0 vysoce flexibilní kontejnerový formát, standardizován ISO 12234-2, ISO 12639
  - soubor obsahuje adresáře (IFD, *Image File Directory*) a bloky obrazových dat
  - IFD je tabulka **popisků** (*tag*, nověji *field*) určující typ dat a adresu začátku datového bloku
    $\Longrightarrow$ v jednom souboru i **více obrázků** (stránek)
  - každá stránka navíc může být komprimována jinak
    - PackBits, JPEG (!), ...
    - RGB, indexovaná barva
    - ukládání po dlaždicích $\Longrightarrow$ lze zpracovávat jen část obrazu
    - ...
  - nedostatek podpory všech možností TIFFu v (starším) softwaru $\Longrightarrow$ **T**housands of **I**ncompatible **F**ile **F**ormats

Přehled populárních formátů 4/4

- **PNG** (Portable Network Graphics) - relativně nový formát, zaměřen na web
  - standardizován (ISO)
  - **bezztrátová** komprese na bázi LZ77
  - ukládání pixelů po **bezztrátovém** předzpracování (*none*, *Sub* - rozdíl oproti pixelu vlevo, *Up* - rozdíl oproti pixelu na předchozím řádku, *Average* - průměr pixelu a dvou sousedů, *Paeth* - složitější průměrování)
  - umožňuje dvourozměrné **prokládání** (nejprve přenese např. 1 pixel z každé matice 2×2) $\Longrightarrow$ umožňuje náhled obrázku před stažením všech dat z webu
  - ukládá True Color RGBA (včetně **průhlednosti**)



Prokládání Adam7

Formát JPEG

- **JPEG** - Joint Photographic Experts Group, 1991
- standardizován (ISO)
- ztrátová komprese s nastavitelnou úrovní (kvalitou)
- **vhodné** pro fotografie a další složité obrázky, kde skoro žádné dva pixely nemají stejnou barvu
- **nevhodné** pro obrázky s nižším barevným rozlišením
  - rozmazává ostré přechody, nejviditelněji na hranicích souvislých barevných ploch
- praxe: při nastavení kvality na „75%“ není většinou pouhým okem poznat rozdíl od originálu, kompresní poměr je přitom cca 1 : 20!
- kromě obrázku obsahuje **metadata** (EXIF informace)

Formát JPEG

Komprese JPEG 1/4

1. Transformace barev

- převedení barev do prostoru YC_BC_R
- na změny jasové složky Y je člověk citlivější než na změny odstínů
  $\Longrightarrow$ kanály Y a C_BC_R se zpracovávají odděleně, C_BC_R lze více komprimovat
- 1 byte na kanál

2. Redukce barev (ztrátová, ale ztráta nevýznamná)

- průměrování hodnot C_BC_R sousedních pixelů:
  - **2h1v** - průměrování sousedních dvojic v řádku (2 horizontálně, 1 vertikálně)
    $\Longrightarrow$ redukce ze 6 na 4 byty
  - **2h2v** - průměrování čtveřic (2 horizontálně, 2 vertikálně)
    $\Longrightarrow$ redukce ze 12 na 6 bytů

Formát JPEG

Komprese JPEG 2/4

3. DCT:

- data v **každé barevné složce** rozdělena do čtverců 8×8 pixelů
- každý čtverec podroben dopředné **diskrétní kosinové transformaci** (DCT) - podobná diskrétní Fourierově transformaci

$$F(u,v) = \frac{1}{4} C(u) C(v) \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}$$

$$C(t) = \begin{cases} \frac{1}{\sqrt{2}} & \text{pro } t = 0 \\ 1 & \text{jinak} \end{cases}$$

Formát JPEG

Komprese JPEG 2/4

3. DCT:

- data v **každé barevné složce** rozdělena do čtverců 8×8 pixelů
- každý čtverec podroben dopředné **diskrétní kosinové transformaci** (DCT) - podobná diskrétní Fourierově transformaci

- pozn: zpětná (inverzní) transformace:

$$f(x,y) = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C(u)\, C(v)\, F(u,v) \cos \frac{(2x+1)\, u\pi}{16} \cos \frac{(2y+1)\, v\pi}{16}$$

- $F(u,v)$ je koeficient v rozkladu f do kosinových vln

Formát JPEG

Komprese JPEG 3/4

- výsledkem je matice 8×8 **neceločíselných** koeficientů
- vlevo nahoře tzv. DC člen - určuje konstantní (neperiodickou) složku v datech
  - je roven osminásobku průměru celé původní matice
- ostatní členy jsou tzv. AC členy, jejich vliv klesá se vzdáleností od DC členu

4. Kvantizace: (ztrátová)

- matice koeficientů se po prvcích vydělí **tabulkou kvantizačních koeficientů** a výsledek se zaokrouhlí na celá čísla
  - prvky v kvant. tabulce rostou se vzdáleností od levého horního rohu
    $\Longrightarrow$ potlačují se méně významné AC členy, po zaokrouhlení vyjdou rovny nule

Formát JPEG

Komprese JPEG 4/4

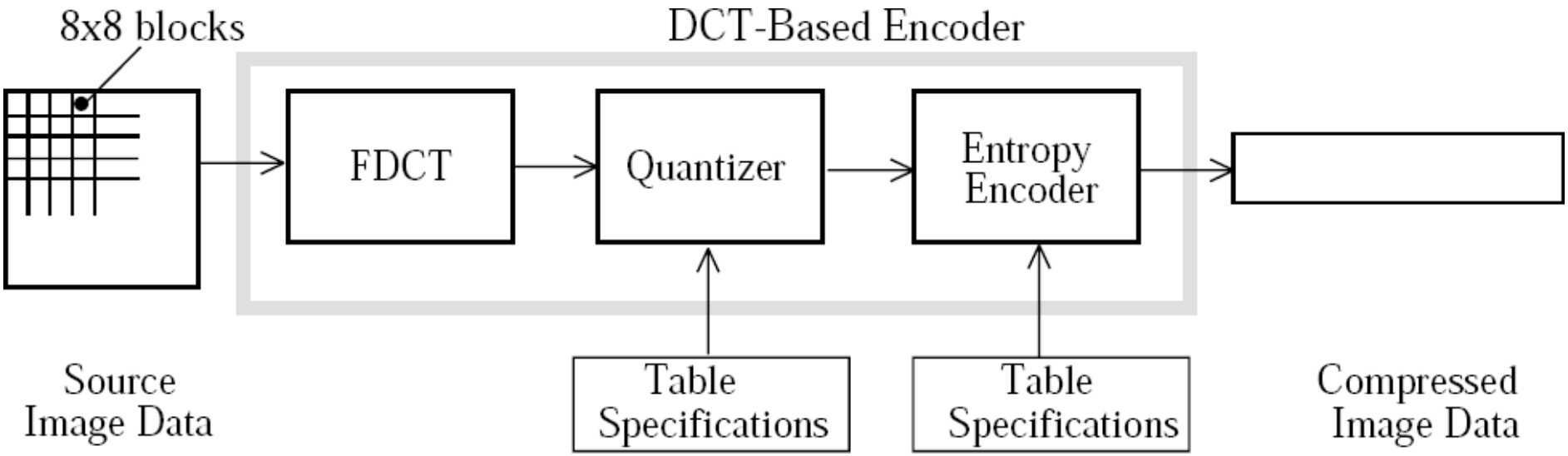
- **standardní** kvant. tabulka určena pro kvalitu cca 75%
- uživatel určí kvalitu Q v rozsahu od 0 do 100%
- skutečně použitá kvantizační tabulka vznikne přepočítáním standardní tabulky nepřímo úměrně ke Q
- použitá tabulka se uloží do souboru spolu s obrázkem (nutné pro dekompresi)
- čím nižší kvalita, tím více AC členů je po kvantizaci rovno 0

5. Kódování (bezztrátové)

- matice 8×8 se zakóduje Huffmanovým kódováním s využitím faktu, že méně významné AC členy jsou obvykle rovny nule
- DC členy jsou zapisovány zvlášť (ukládány relativně - jen rozdíly oproti DC v předchozím bloku 8×8)
- samostatný zápis DC $\Longrightarrow$ umožňuje rychle generovat náhled obrázku s $\frac{1}{8}$-rozlišením v každém směru

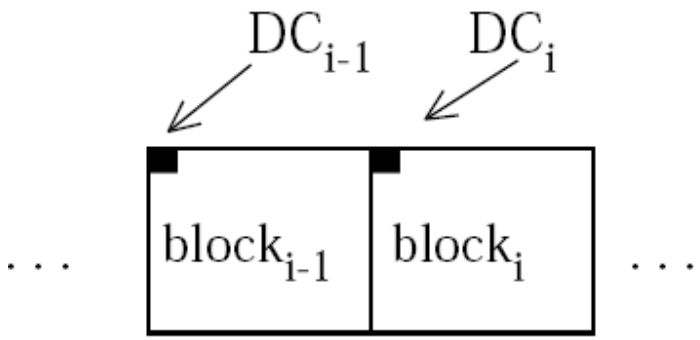
Formát JPEG

Schéma komprese



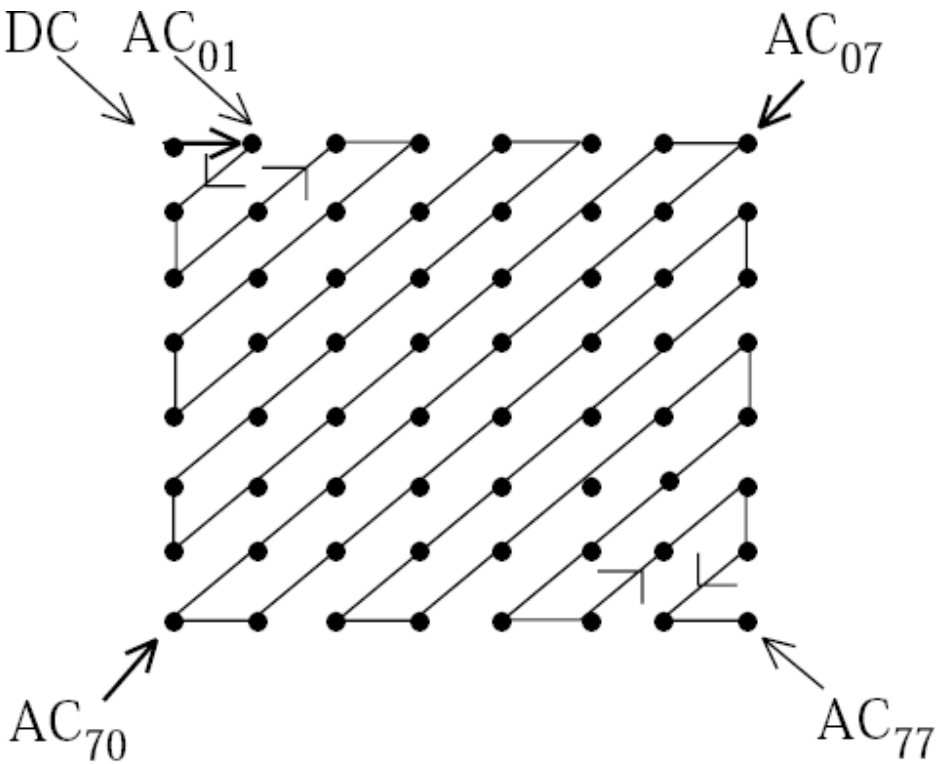
Formát JPEG

Schéma zápisu DC členů a řazení AC členů v matici před kódováním



$$DIFF = DC_i - DC_{i-1}$$

Differential DC encoding



Zig–zag sequence

Formát JPEG

Příklad: DCT, kvantování, rekonstrukce

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

(a) source image samples

235.6	-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

(b) forward DCT coefficients

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

(c) quantization table

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

(d) normalized quantized coefficients

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

(e) denormalized quantized coefficients

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	159	158

(f) reconstructed image samples

Formát JPEG

Příklad 1 - fotografie

Originál JPEG 98%, 25 MPix, cca 14 MB (vs. BMP cca 75 MB)

Formát JPEG

Příklad 1 - fotografie

Výřez 2,2 MPix, JPEG 98% 1149 kB (PNG 2805 kB)

Formát JPEG

Příklad 1 - fotografie

JPEG 5%, 64 kB

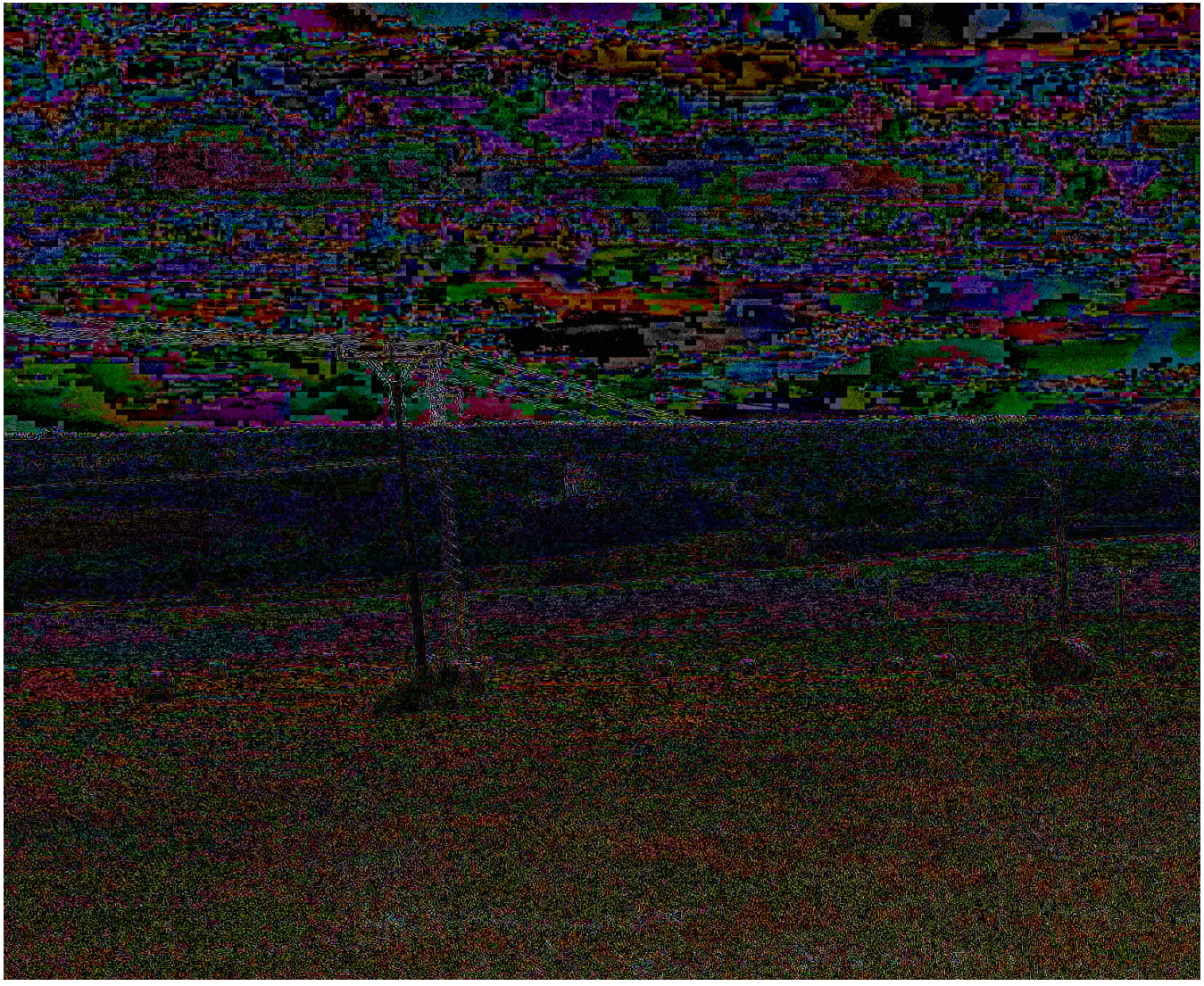
Formát JPEG

Příklad 1 - fotografie

(zvýrazněný) rozdíl od originálu, JPEG 5%

Formát JPEG

Příklad 1 - fotografie



rozdíl od originálu, JPEG 25%

Formát JPEG

Příklad 1 - fotografie

rozdíl od originálu, JPEG 50%

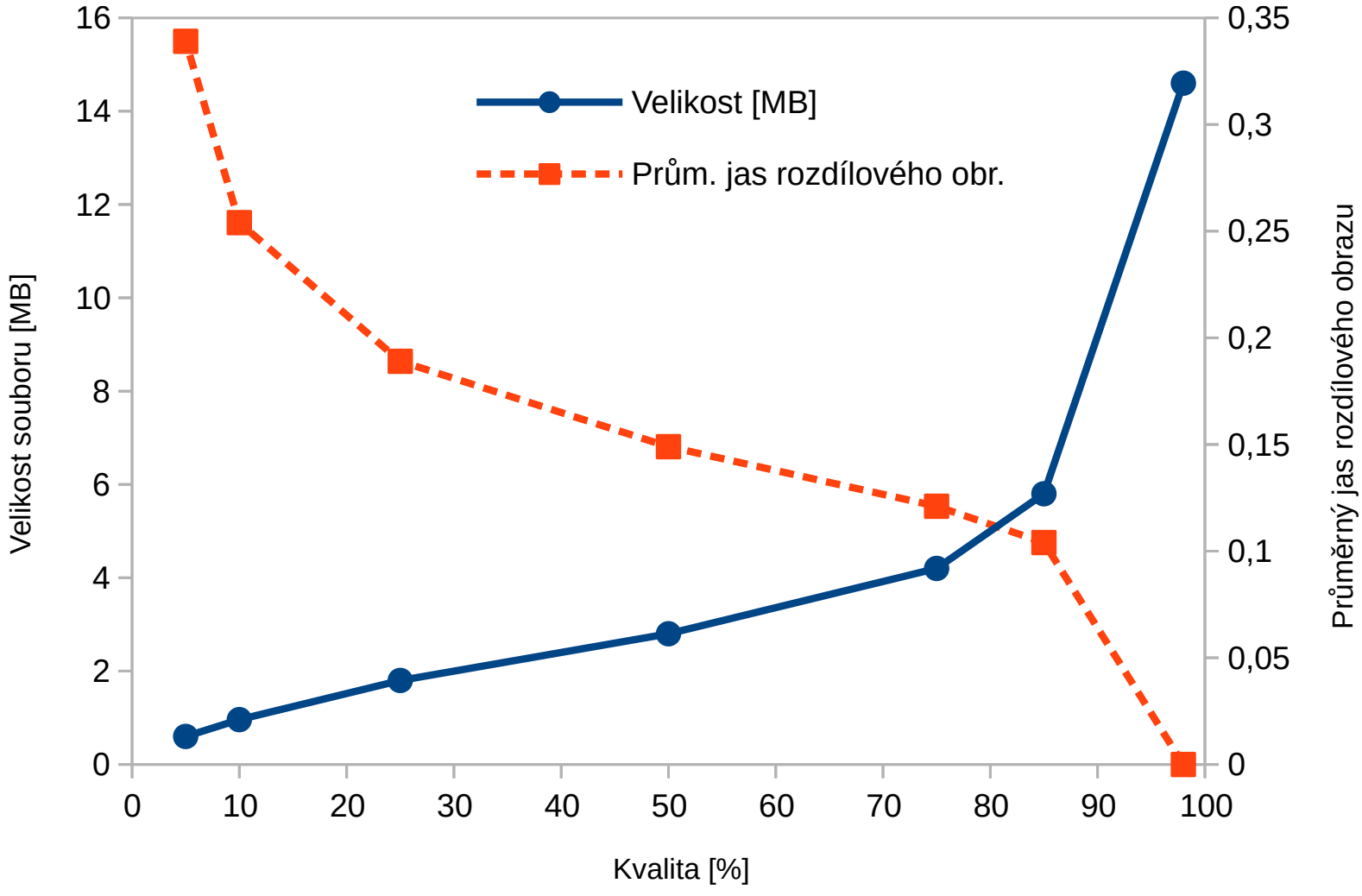
Formát JPEG

Příklad 1 - fotografie

rozdíl od originálu, JPEG 85%

Formát JPEG

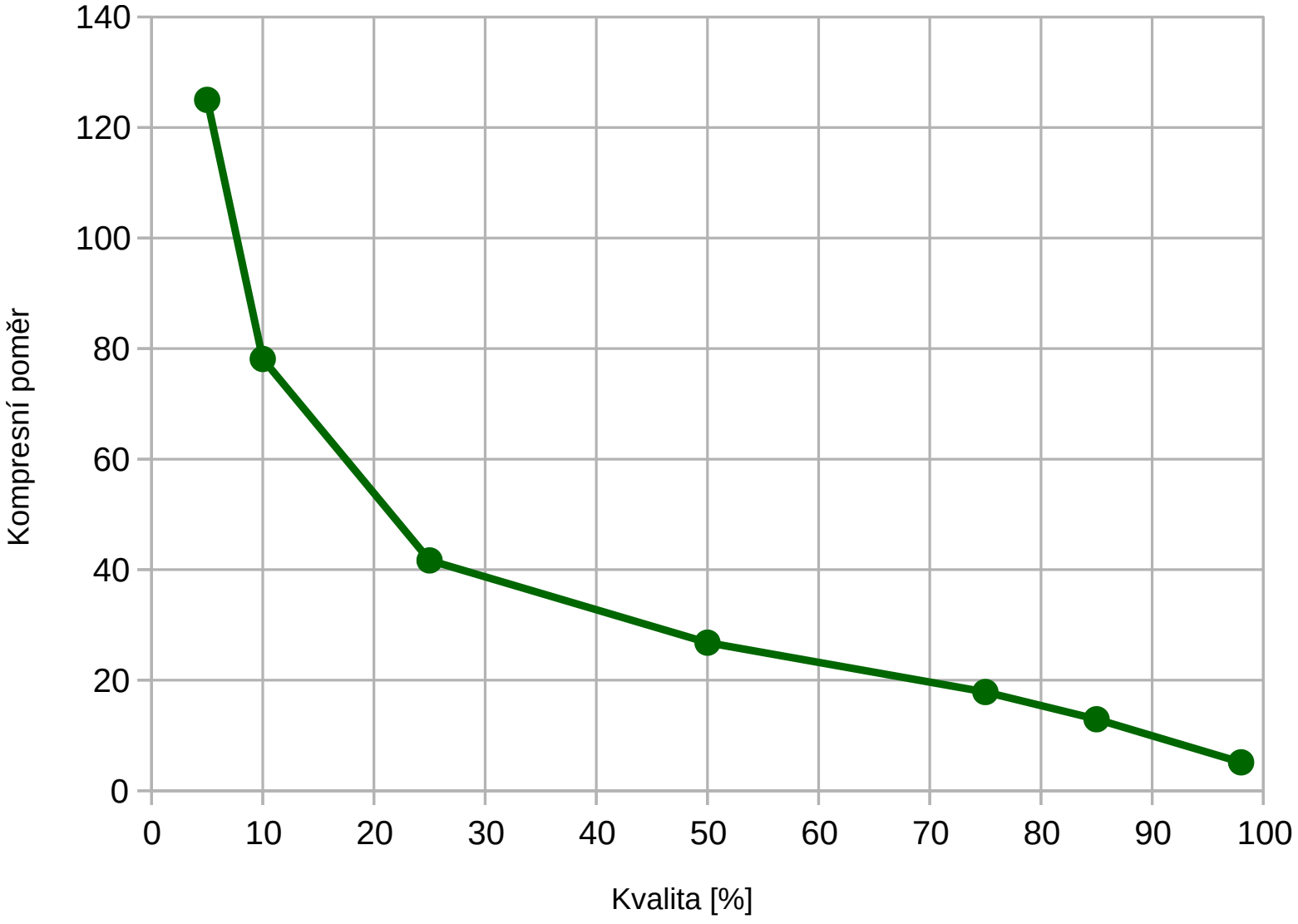
Příklad 1 - fotografie



Velikost a intenzita artefaktů vs. kvalita

Formát JPEG

Příklad 1 - fotografie



Kompresní poměr vs. kvalita

Formát JPEG

Příklad 2 - kresba

Originál PNG 65kB

Formát JPEG

Příklad 2 - kresba

JPEG 98%, 173 kB

Formát JPEG

Příklad 2 - kresba

JPEG 85%, 60 kB

Formát JPEG

Příklad 2 - kresba

JPEG 75%, 48kB

Formát JPEG

Příklad 2 - kresba

JPEG 50%, 34 kB

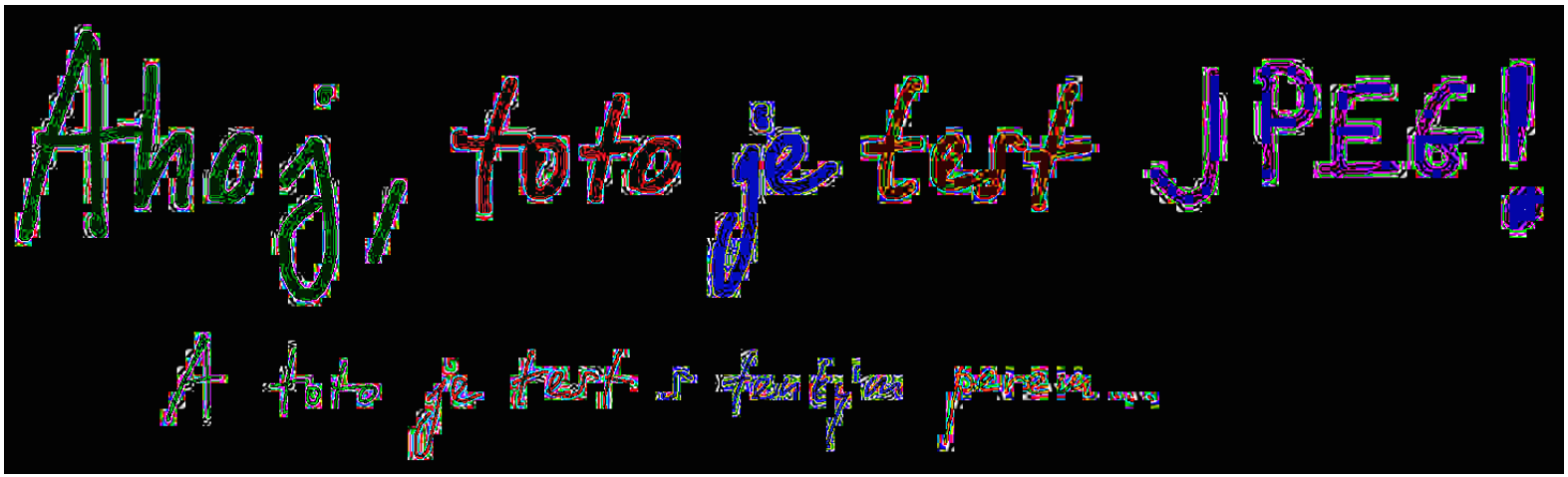
Formát JPEG

Příklad 2 - kresba

JPEG 10%, 14 kB

Formát JPEG

Příklad 2 - kresba



rozdíl od originálu, JPEG 10%

Formát JPEG

Příklad 2 - kresba

rozdíl od originálu, JPEG 50%

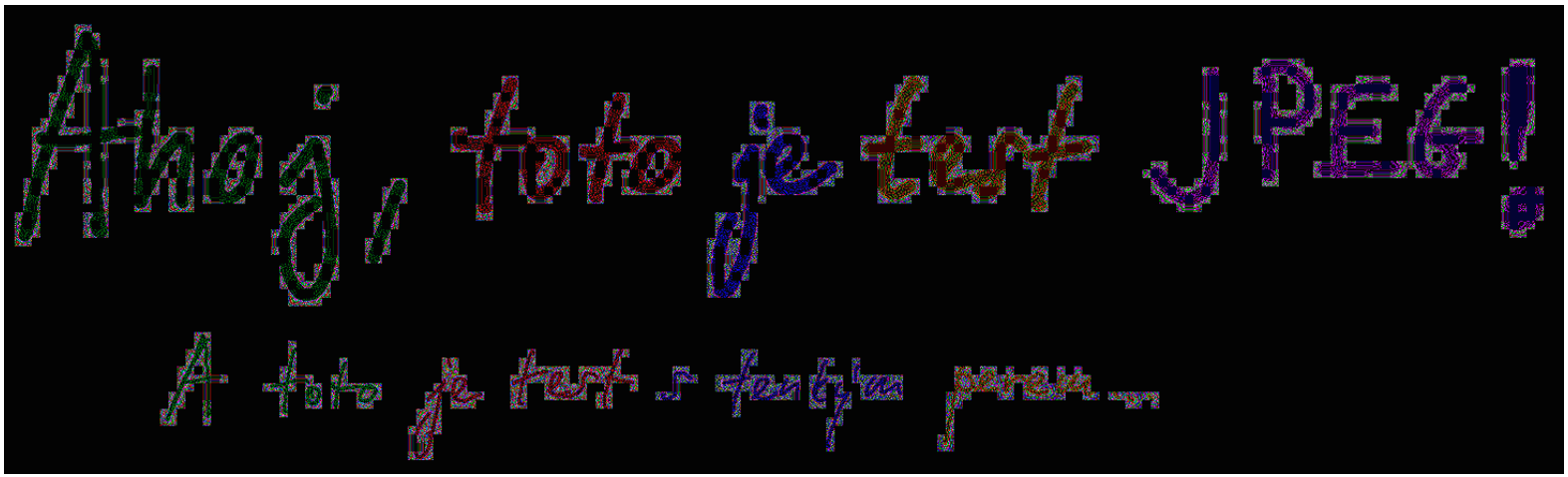
Formát JPEG

Příklad 2 - kresba

rozdíl od originálu, JPEG 75%

Formát JPEG

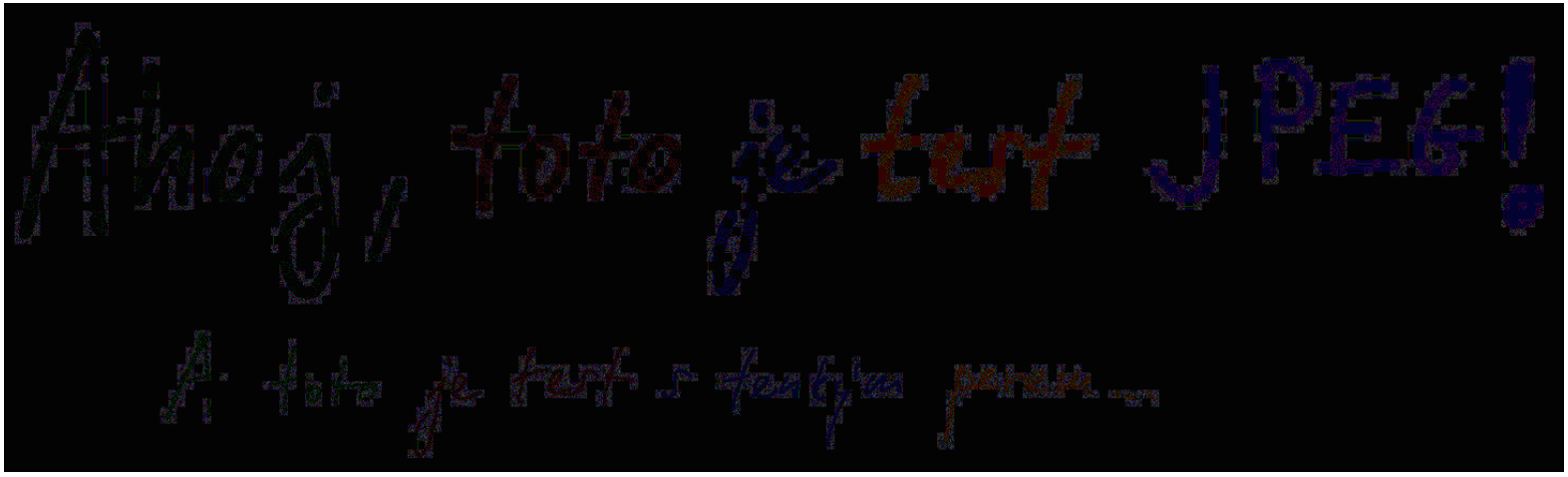
Příklad 2 - kresba



rozdíl od originálu, JPEG 85%

Formát JPEG

Příklad 2 - kresba



rozdíl od originálu, JPEG 98%

Modernější formáty (2000-2021)

- **JPEG 2000**
  - waveletová transformace místo DCT
- **JPEG XT**
  - vyšší bitové hloubky
  - zpětně kompatibilní s JPEG
- **JPEG XR** (2006)
  - původně Microsoft HD Photo
  - standard pro ukládání obrázků v OpenXML Paper Specification
  - umožňuje i bezztrátovou kompresi
- **HEIF**
  - High Efficiency Image Format
  - založen na HEVC (viz dále)
- **WebP** (`.webp`, 2010)
- **AVIF** (2019)
  - AV1 komprese (viz dále)
- **JPEG** XL (`.jxl`, 2020)
  - cílem je nahradit JPEG („L“ = *long-term*)
  - podpora animací, HDR
  - variable-size DCT

JPEG vs. JPEG 2000

Obsah

Komprese videa

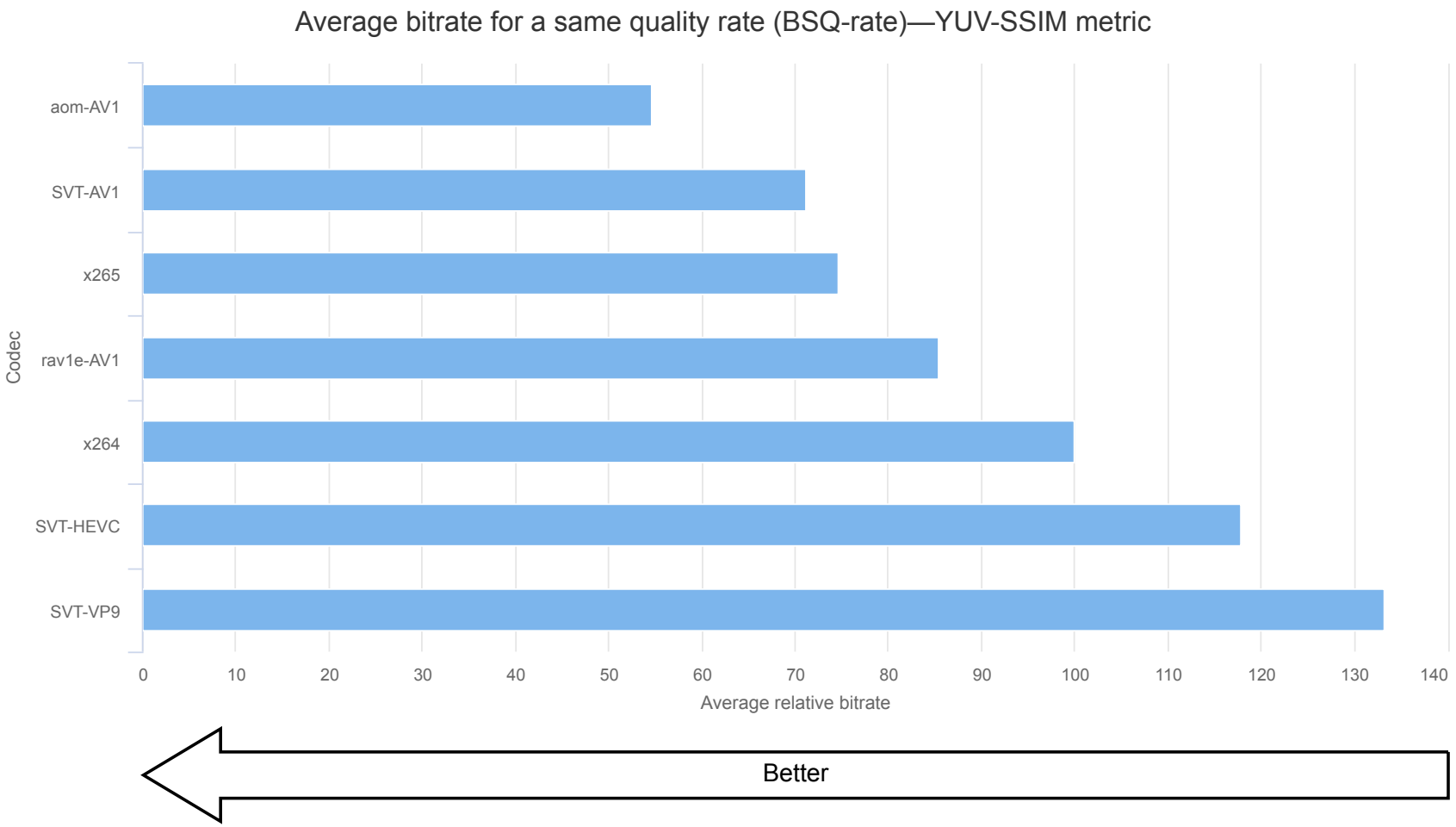
- MPEG-1, MPEG-2 (Motion Picture Experts Group)
  - sekvence klíčových (úplných) a rozdílových snímků
- ztrátová komprese pro obraz i zvuk (MP3, AAC)
- formát **MPEG-4**
  - pokročilá komprese videa
  - definice titulků
  - pohyb 3D objektů, pohyb kamery
  - animace lidského obličeje a popis jeho výrazu (*talking head*)
  - obvykle jen komprese videa (DivX, XviD, **H.264**)
- další formáty: ASF, QuickTime MOV, Flash FLV, MXF, Theora OGV

Moderní kodeky

- 2012: **HEVC** (High Efficiency Video Coding) a.k.a. **H.265**
  - až 64×64 CTUs (coding tree units) $\sim$ MCU blocks, macroblocks
  - DVBT2 ve FullHD
- 2013: Google **VP9**
- 2015: AOMedia **AV1**
  - evoluce Theora OGV
  - bez licenčních poplatků (royalty free)
  - testy potvrzují velmi dobrý poměr kvality vs. bitrate
  - ale je pomalý
- 2020: ve vývoji
  - Daala **OGV**
  - MPEG-5 **EVC** (Essential Video Coding)

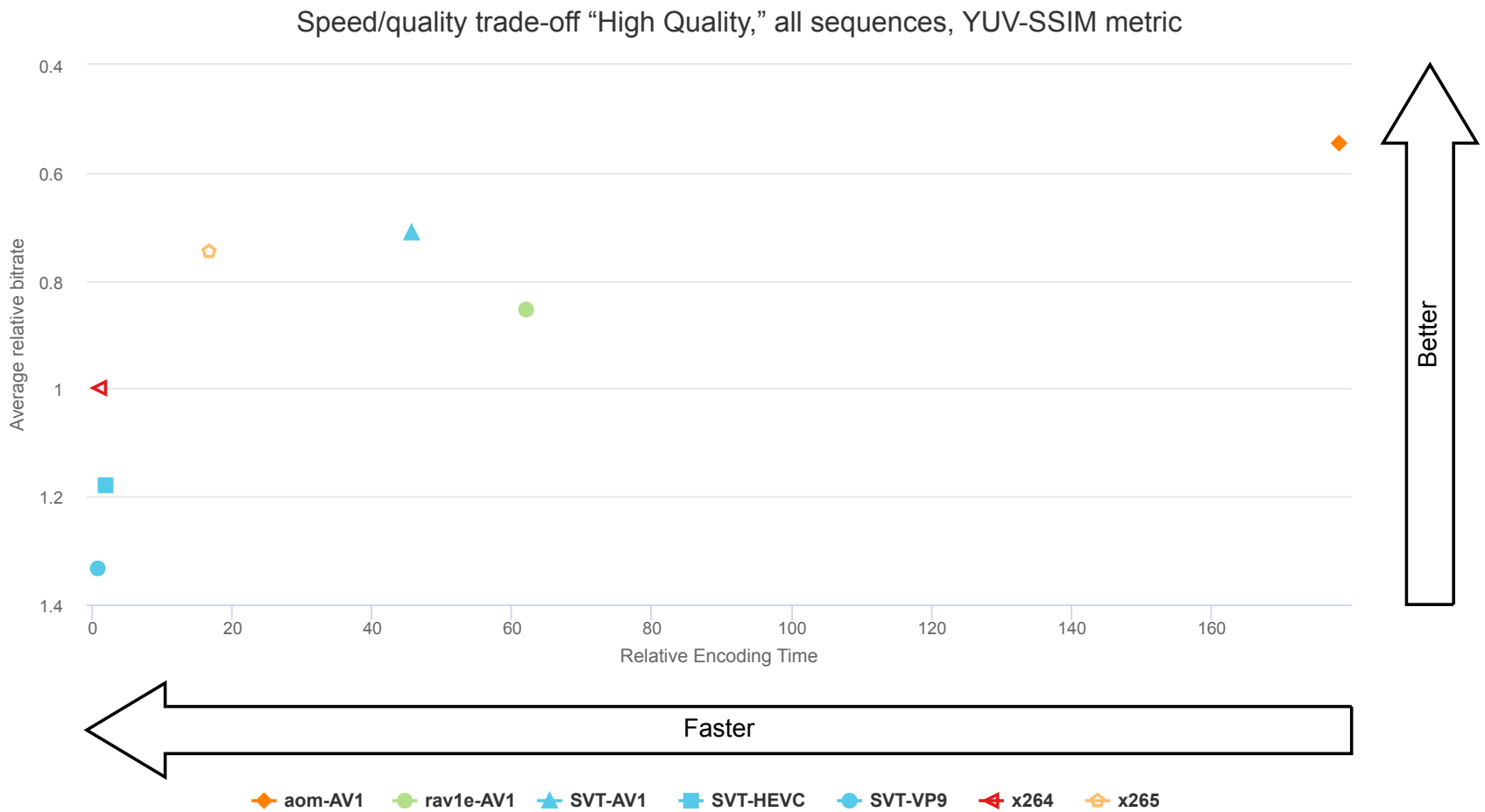
Srovnání kodeků 2019

www.compression.ru



Srovnání kodeků 2019

www.compression.ru



Literatura


I. Vajda: Teorie informace. Vydavatelství ČVUT, Praha, 2004. ISBN: 80-01-02986-7


Žára, Beneš, Sochor, Felkel: *Moderní počítačová grafika*. Computer Press, 2005. ISBN: 80-251-0454-0