

1. JPEG komprese a dekomprese rastru

Matěj Klimeš, Tomáš Zbíral
ZS 2024/25, číslo skupiny: 2, datum zpracování: 21.10.2024

1 Zadání

Implementujte algoritmus pro JPEG kompresi/dekompresi rastru v prostředí MATLAB (popř. v programovacím jazyce dle vlastního výběru), zahrnující tyto fáze:

- transformaci do $YC_B C_R$ modelu,
- diskrétní kosinovou transformaci,
- kvantizaci koeficientů,

a to bez využití vestavěných funkcí.

Kompresní algoritmus otestujte na různých typech rastru: rastr v odstínech šedi, barevný rastr (viz tabulka) vhodného rozlišení a velikosti (max 128×128 pixelů) s různými hodnotami faktoru komprese $q = 10, 50, 70$. Pro každou variantu spočítejte střední kvadratickou odchylku σ jednotlivých RGB složek.

$$\sigma = \sqrt{\frac{\sum_{i=0}^{m \cdot n} (z - z')^2}{m \cdot n}}$$

Výsledky umístěte do přehledných tabulek pro jednotlivá q . Na základě výše vypočtených údajů zhodnoťte, ke kterým typům dat je JPEG komprese nejvíce a naopak nejméně vhodná.

1.1 Řešené bonusové úlohy

Nad rámec zadání byly řešeny tyto bonusové úlohy:

- Resamplování rastru některou z metod
- Náhrada DCT s využitím diskrétní Fourierovy transformace

2 Teoretický úvod

JPEG (Joint Photographic Experts Group) je jedním z nejběžněji používaných metod komprese obrazových dat. Jedná se o ztrátovou kompresní metodu, což znamená, že při ukládání obrazových dat dochází k částečné ztrátě původních informací, a to za účelem dosažení podstatného snížení velikosti souboru. Metodu JPEG lze také označit jako transformační, jelikož využívá geometrické transformace obrazových dat. Obraz je při kompresi aproximován určitou funkcí a při dekompresi zpětně matematicky rekonstruován. JPEG komprese využívá faktu, že lidské oko je méně citlivé na malé změny barvy než na malé změny jasu. Při kompresi jsou tedy nevýznamné změny barev odstraněny a naopak změny jasu jsou zachovány.

3 Pracovní postup

Úloha byla zpracovávána pomocí programovacího jazyka MATLAB. V této části budou jednotlivé kroky výpočtu popisovány formálním či matematickým jazykem. Syntaxi jednotlivých kroků v programovacím jazyce MATLAB je možné prohlédnout v Příloze 1.

- Před samotným výpočtem bylo nezbytné načíst vstupní rastr a zvolit faktor komprese q .
- Prvním výpočetním krokem byla separace vstupního obrazu na R , G , B složky a následná transformace těchto složek na jasovou složku Y a barevné složky C_B a C_R dle následující rovnice:

$$\begin{bmatrix} Y \\ C_B \\ C_R \end{bmatrix} = \begin{bmatrix} 0.2990 & 0.5870 & 0.1140 \\ -0.1687 & -0.3313 & 0.5000 \\ 0.5000 & -0.4187 & -0.0813 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}$$

- Dalším krokem bylo převzorkování barevných složek C_B a C_R pomocí metody nejbližšího souseda, nebo pomocí lineární interpolace. Obě metody byly implementovány pro výpočet ze submatic 2×2 a 4×4 . Princip je následující:

1. Nejbližší soused (Nearest Neighbor) - Hodnota cílového pixelu je převzata z nejbližšího pixelu původní matice.

- (a) Downsampling (např. z 4×4 na 2×2):

$$\text{Původní matice: } \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \rightarrow \text{Výsledná matice: } \begin{bmatrix} a & c \\ i & k \end{bmatrix}$$

- (b) Upsampling (např. z 2×2 na 4×4):

$$\text{Původní matice: } \begin{bmatrix} a & b \\ c & d \end{bmatrix} \rightarrow \text{Výsledná matice: } \begin{bmatrix} a & a & b & b \\ a & a & b & b \\ c & c & d & d \\ c & c & d & d \end{bmatrix}$$

2. Lineární interpolace - Hodnota cílového pixelu je vážený průměr okolních pixelů z původní matice.

- (a) Downsampling (např. z 4×4 na 2×2):

$$\text{Původní matice: } \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \rightarrow \text{Výsledná matice: } \begin{bmatrix} \frac{a+b+e+f}{4} & \frac{c+d+g+h}{4} \\ \frac{i+j+m+n}{4} & \frac{k+l+o+p}{4} \end{bmatrix}$$

- (b) Upsampling (např. z 2×2 na 4×4):

$$\text{Původní matice: } \begin{bmatrix} a & b \\ c & d \end{bmatrix} \rightarrow \text{Výsledná matice: } \begin{bmatrix} a & \frac{a+b}{2} & b & \frac{b+d}{2} \\ \frac{a+c}{2} & \dots & \dots & \dots \\ c & \frac{c+d}{2} & d & \frac{d+b}{2} \\ \dots & \dots & \dots & d \end{bmatrix}$$

- Byly definovány kvantizační matice pro jasovou složku (Q_y) a barevné složky (Q_c) a upraveny dle zvoleného faktoru komprese q .

$$Q_y^{50} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 87 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 26 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \quad Q_c^{50} = \begin{bmatrix} 17 & 18 & 24 & 47 & 66 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 69 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix}$$

Upravené kvantizační matice jsou dány vztahem:

$$Q_y = \frac{50 \cdot Q_y}{q} \quad Q_c = \frac{50 \cdot Q_c}{q}$$

- Dále byly jednotlivé složky rozděleny na submatice 8x8 pixelů.
- Následná komprese byla provedena po jednotlivých submaticích. První operací prováděnou na submatici je transformace. Využívá se diskretní kosínová transformace, jejíž vztah je následující:

Předpis DCT:

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

$$C(u) = \begin{cases} \frac{\sqrt{2}}{2}, & u = 0, \\ 1, & u \neq 0. \end{cases} \quad C(v) = \begin{cases} \frac{\sqrt{2}}{2}, & v = 0, \\ 1, & v \neq 0. \end{cases}$$

- Kromě diskretní kosínové transformace (DCT), která je matematickým základem JPEG komprese, byla implementována i diskretní Fourierova transformace (DFT) pomocí Cooley-Tukey algoritmu.

Předpis DFT:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-j \frac{2\pi kn}{N}}, \quad k = 0, 1, \dots, N-1$$

Postup algoritmu v 1D

1. Rozdělení vektoru x na sudé a liché indexy.
2. Rekursivní výpočet DFT pro sudé a liché části.
3. Kombinace výsledků pomocí:

$$X_k = X_{\text{even}}(k) + W_N^k X_{\text{odd}}(k), \quad X_{k+N/2} = X_{\text{even}}(k) - W_N^k X_{\text{odd}}(k)$$

$$\text{kde } W_N^k = e^{-2\pi i \frac{k}{N}}.$$

Postup algoritmu ve 2D

1. Aplikace 1D FFT na řádky matice.
2. Aplikace 1D FFT na sloupce výsledné matice.

Matematicky:

$$X_{k,l} = \sum_{m=0}^{N-1} \sum_{n=0}^{M-1} x_{m,n} e^{-2\pi i \frac{km}{N}} e^{-2\pi i \frac{ln}{M}}$$

- Po transformaci byla na každé submatici provedena kvantizace koeficientů transformace. V praxi se jedná o dělení submatice kvantizační maticí a zaokrouhlení výsledných hodnot. Cílem je vypuštění koeficientů s malou hodnotou.
- Na komprimovaných hodnotách byla následně prováděna dekomprese opět po submaticích 8x8. Nejprve proběhla dekvantizace, jakožto inverzní krok kvantizace, který již není ztrátový.
- Dále byla provedena inverzní diskrétní kosínová transformace či inverzní diskrétní Fourierova transformace. Principiálně je algoritmus Cooley-Tukey pro IDFT velmi podobný jako pro DFT.

Předpis IDCT:

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

$$C(u) = \begin{cases} \frac{\sqrt{2}}{2}, & u = 0, \\ 1, & u \neq 0. \end{cases} \quad C(v) = \begin{cases} \frac{\sqrt{2}}{2}, & v = 0, \\ 1, & v \neq 0. \end{cases}$$

Předpis IDFT:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot e^{j\frac{2\pi kn}{N}}, \quad n = 0, 1, \dots, N-1$$

- Následně bylo nutné převzorkovat barevné složky na původní rozměry. Opět byly implementovány možnosti využití metody nejbližšího souseda a lineární interpolace pro submatice 2x2 a 4x4 pixelu.
- Dále byly složky Y , C_B , C_R převedeny zpět na R , G , B dle následujícího vztahu:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.0000 & 0.0000 & 1.4020 \\ 1.0000 & -0.3441 & -0.7141 \\ 1.0000 & 1.7720 & -0.0001 \end{bmatrix} \begin{bmatrix} Y \\ C_B - 128 \\ C_R - 128 \end{bmatrix}$$




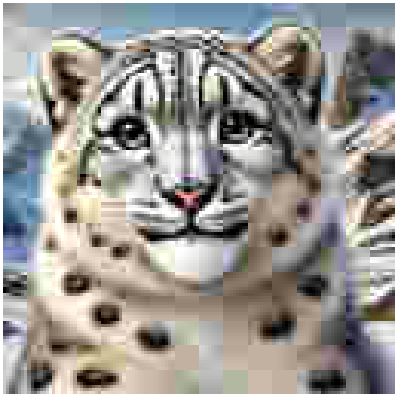








- Složky R , G , B byly opět zkombinovány do výstupního rastru a byly vypočteny směrodatné odchylky původního a komprimovaného rastru.

4 Výsledky

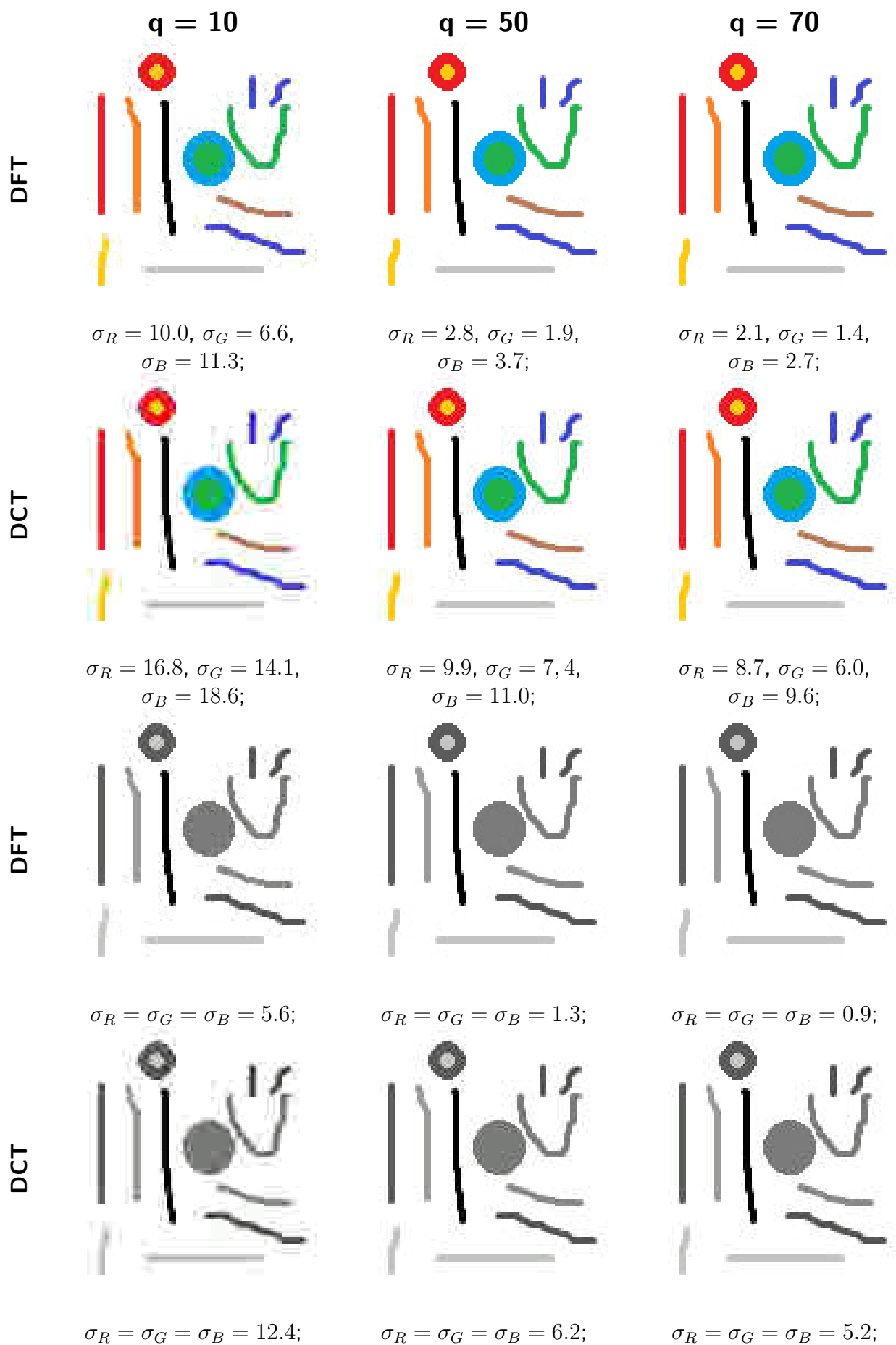
V této sekci budou prezentovány dosažené výsledky kompresního algoritmu. Budou porovnány výsledky pro barevný a šedotónový rastr, pro oba typy transformací (DCT, DFT) bez využití převzorkování. Výsledky kompresního algoritmu byly zkoumány na dvou barevných a dvou šedotónových rastroch.



Obrázek 1: Vstupní dvojice rastrů

	$q = 10$	$q = 50$	$q = 70$
DFT	 $\sigma_R = 9.8, \sigma_G = 9.2,$ $\sigma_B = 10.3;$	 $\sigma_R = 3.2, \sigma_G = 2.6,$ $\sigma_B = 3.9;$	 $\sigma_R = 2.6, \sigma_G = 1.9,$ $\sigma_B = 3.2;$
DCT	 $\sigma_R = 18.9, \sigma_G = 18.0,$ $\sigma_B = 19.7;$	 $\sigma_R = 9.6, \sigma_G = 9.2,$ $\sigma_B = 10.0;$	 $\sigma_R = 8.1, \sigma_G = 7.7,$ $\sigma_B = 8.4;$
DFT	 $\sigma_R = \sigma_G = \sigma_B = 9.1;$	 $\sigma_R = \sigma_G = \sigma_B = 2.4;$	 $\sigma_R = \sigma_G = \sigma_B = 1.7;$
DCT	 $\sigma_R = \sigma_G = \sigma_B = 17.7;$	 $\sigma_R = \sigma_G = \sigma_B = 9.1;$	 $\sigma_R = \sigma_G = \sigma_B = 7.6;$

Obrázek 2: Výsledky kompresního algoritmu na první barevné a šedotónové dvojici rastrů



Obrázek 3: Výsledky kompresního algoritmu na druhé barevné a šedotónové dvojici rastrů

Kvalita (q)	Typ transformace	Barevný rastr			Šedotónový rastr		
		σ_R	σ_G	σ_B	σ_R	σ_G	σ_B
q = 10	DFT	9.8	9.2	10.3	9.1	9.1	9.1
q = 50	DFT	3.2	2.6	3.9	2.4	2.4	2.4
q = 70	DFT	2.6	1.9	3.2	1.7	1.7	1.7
q = 10	DCT	18.9	18.0	19.7	17.7	17.7	17.7
q = 50	DCT	9.6	9.2	10.0	9.1	9.1	9.1
q = 70	DCT	8.1	7.7	8.4	7.6	7.6	7.6

Tabulka 1: Výsledky kompresního algoritmu na první barevné a šedotónové dvojici rastrů

Kvalita (q)	Typ transformace	Barevný rastr			Šedotónový rastr		
		σ_R	σ_G	σ_B	σ_R	σ_G	σ_B
q = 10	DFT	10.0	6.6	11.3	5.6	5.6	5.6
q = 50	DFT	2.8	1.9	3.7	1.3	1.3	1.3
q = 70	DFT	2.1	1.4	2.7	0.9	0.9	0.9
q = 10	DCT	16.8	14.1	18.6	12.4	12.4	12.4
q = 50	DCT	9.9	7.4	11.0	6.2	6.2	6.2
q = 70	DCT	8.7	6.0	9.6	5.2	5.2	5.2

Tabulka 2: Výsledky kompresního algoritmu na druhé barevné a šedotónové dvojici rastrů

5 Závěr

- Byl implementován algoritmus pro JPEG kompresi a dekompresi rastru, který byl v závěru testován na barevných a šedotónových rastrech
- Z výsledků je dobře zřetelné, že algoritmus je vhodný pro rastry s postupnými barevnými přechody (obrázek 2). U těchto rastrů nedochází k výrazné ztrátě kvality obrazu. Naopak algoritmus není vhodný pro rastry, ve kterých se vyskytují ostré hrany (viz obrázek 3). V okolí těchto hran vznikají při zadání nižšího faktoru komprese viditelné artefakty, které významně deformují výsledný obraz.
- Při volbě nižšího faktoru komprese jsou, především při využití DCT, také viditelná rozhraní mezi jednotlivými výpočetními submaticemi.
- Z výsledků je také patrné, že diskrétní Fourierova transformace dosahuje menších směrodatných odchylek jednotlivých barevných složek než diskrétní kosínová transformace.
- Pro šedotónové rastry jsou vypočtené směrodatné odchylky pro jednotlivé kanály téměř shodné, jelikož šedá je v RGB modelu dána stejnou intenzitou všech tří složek R, G, B.
- Návrhy na vylepšení: Použití dalších interpolačních metod; Vylepšení jednotlivých funkcí, omezení dat na vstupu

6 Přílohy

Příloha 1 - MATLAB skripty

V Praze dne 21.10.2024

Matěj Klimeš, Tomáš Zbíral