<table>
<tr><td colspan="6" align="center">ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE<br>FAKULTA STAVEBNÍ<br>KATEDRA GEOMATIKY</td></tr>
<tr><td colspan="6">Název předmětu<br> Geoinformatika </td></tr>
<tr><td>Úloha<br>U1</td><td colspan="5">Název úlohy:<br>JPEG komprese rastru</td></tr>
<tr><td>akademický rok<br>2024/2025</td><td>semestr<br>zimní</td><td>studijní skupina<br>C102</td><td>vypracoval<br>Matyáš Pokorný<br>Tereza Černohousová</td><td>datum<br>22.10.2024</td><td>klasifikace</td></tr>
</table>

# Technická zpráva

## 1 Bonusové úlohy

Z bonusových úloh námi byly zpracovány:

- JPEG komprese a dekomprese
- Resamplování některou z metod
- Konverze pixelů do ZIG-ZAG sekvencí
- Huffmanovo kódování
- Náhrada DCT s využitím diskrétní vlnkové transformace

## 2 Pracovní postup

JPEG je hojně využívaná metoda komprese rastrů. Dosahuje vysokých kompresních poměrů, ale je ztrátová. Metoda má několik částí.

- Rozdělení rastru na RGB složky
- Převod z RGB na $YC_BC_R$

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.1687 & -0.3313 & 0.5 \\ 0.5 & -0.4187 & -0.0813 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}$$

- Rozdělení na submatice 8x8
- Převzorkování
- Diskrétní kosinová transformace
- nebo diskrétní vlnková transformace
- Kvantizace - provádí se dělením po prvcích, již vytvořenou vhodnou kvantizační maticí
- Převod matic na zig-zag sekvence
- Komprese do HUffmanova kódu

Dekomprese

- Dekvantizace - násobení po prvcích stejnou kvantizační maticí
- Inverzní kosinová transformace

- Převod z $YC_BC_R$ na RGB

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.4020 \\ 1 & -0.3441 & -0.7141 \\ 1 & 1.7720 & -0.0001 \end{bmatrix} \begin{bmatrix} Y \\ Cb - 128 \\ Cr - 128 \end{bmatrix}$$

**Diskrétní kosinova transformace s převzorkováním**

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

$$C(u) = \begin{cases} \sqrt{\frac{\sqrt(2)}{2}} & \text{if } u = 0 \\ 1 & \text{if } u \neq 0 \end{cases}$$

$$C(v) = \begin{cases} \sqrt{\frac{\sqrt(2)}{2}} & \text{if } v = 0 \\ 1 & \text{if } v \neq 0 \end{cases}$$

Převzorkování bylo provedeno pomocí průměrování submatic o rozměrech 2x2 pixely

**Funkce `resampling2x2(A)`**

1. Vstup: Matice $A$, kterou chceme převzorkovat.
2. Výstup: Matice $B$ po převzorkování.
3. Postup:
   (a) Zjisti velikost matice $A$ (počet řádků a sloupců).
   (b) Inicializuj matici $B$ se stejnými rozměry jako $A$.
   (c) Pro každý blok $2 \times 2$ v matici $A$:
      i. Vypočítej průměr hodnot v aktuálním bloku.
      ii. Vytvoř dočasnou matici velikosti $2 \times 2$, kde všechny hodnoty jsou průměrné.
      iii. Zapiš tuto matici do odpovídajícího bloku v $B$.
   (d) Vrátí matici $B$.

**Inverzní kosinova transformace**

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

$$C(u) = \begin{cases} \sqrt{\frac{\sqrt(2)}{2}} & \text{if } u = 0 \\ 1 & \text{if } u \neq 0 \end{cases}$$

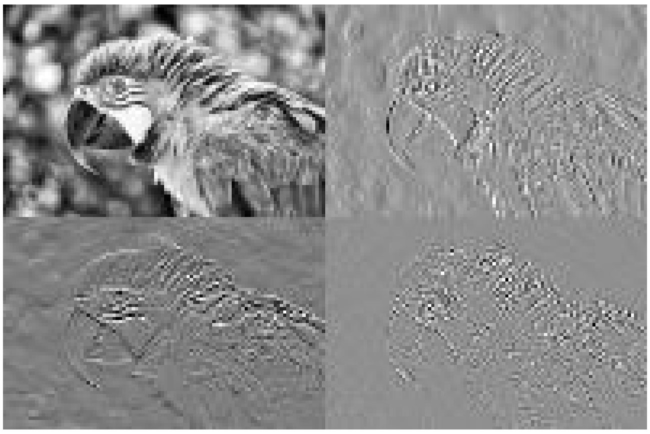
$$C(v) = \begin{cases} \sqrt{\frac{\sqrt(2)}{2}} & \text{if } v = 0 \\ 1 & \text{if } v \neq 0 \end{cases}$$

### 2.1 Diskrétní vlnková transformace

Diskrétní vlnková transformace (DWT) probíhá pomocí nizkofrekvenčních a vysokofrekvenčních filtrů, které vycházejí z použité mateřské vlnky. V této úloze byla použita jako mateřská vlnka Haarova. Nizkofrekvenční filtr $h$ a vysokofrekvenční filtr $g$ mají následující hodnoty:

$$h = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}, \quad g = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

Tyto filtry se nejprve aplikují na řádky a zároveň je zachován každý druhý řádek $\rightarrow$ získáme dvě nové matice o poloviční velikosti. Na tyto dvě matice jsou flitry aplikovány ještě jednou, tentokrát na sloupce, a opět je zachován každý druhý sloupec. Výsledkem jsou čtyři matice $\rightarrow$ LL = 2x aplikovný nízkofrekvenční filtr, LH a HL = kombinace filtrů, HH = 2x aplikovaný vysokofrekvenční filtr. Obrázek po kompresy vypadá nějak takto (normalizované hodnoty):



Obrázek 1: obraz po DWT

Inverzní DWT probíhá tak, že ze čtyř složek LL, LH, HL, HH musí nejprve vzniknout dvě složky L a H a z nich se složí obraz o původní velikosti. Matice L vznikne spojením LL a LH složek. LL je vepsána na liché řádky, LH na sudé řádky. Tím se zvětší velikost matice na poloviční velikost původního obrázku. Matice H vzniká obdobně ze složek HL a HH. Výsledná obraz vzniká pomocí bilineární transformace, která spojuje matice L a H v původní obraz

U této transformace nebyla použita kvantizace. Rozklad na submatice, kvantizace a aplikace DWT na jednotlivé submatice není efektivní způsob z časového hlediska a také bylo obtížné zrekonstruovat zpět obraz. Proto jsme si dovolili práci zjednodušit a DWT aplikovat na celý obraz bez kvantizace.

#### 2.1.1 Popis kódu

**Postup implementace komprese obrazu**

1. **Načtení vstupního obrazu.**

    (a) Čti obrázek ve formátu RGB.

(b) Rozděl obraz na jednotlivé složky: $R$, $G$, $B$.

2. **Transformace RGB do YCbCr barevného prostoru.**

   (a) Vypočítej složky $Y$, $Cb$, $Cr$ podle převodních vzorců:

$$
\begin{aligned}
Y &= 0.299R + 0.587G + 0.114B, \\
Cb &= -0.1687R - 0.3313G + 0.5B + 128, \\
Cr &= 0.5R - 0.4187G - 0.0813B + 128.
\end{aligned}
$$

   (b) Roztáhni intervaly složek na požadovaný rozsah.

3. **Komprese pomocí DWT**

4. **Kvantizace.**

   (a) Kvantizace je v této implementaci vynechána.

5. **Dekompresní dekomprese pomocí DWT**

6. **Transformace zpět z YCbCr do RGB barevného prostoru.**

   (a) Přepočítej složky $R$, $G$, $B$ z $Y$, $Cb$, $Cr$ pomocí převodních vzorců:

$$
\begin{aligned}
R &= Y + 1.402(Cr - 128), \\
G &= Y - 0.344136(Cb - 128) - 0.714136(Cr - 128), \\
B &= Y + 1.772(Cb - 128).
\end{aligned}
$$

7. **Sestavení obrazu zpět, uložení nebo zobrazení**

   (a) Ulož výsledný obraz do souboru nebo jej zobraz.

8. **Vyhodnocení kvality komprese.**

   (a) Vypočítej střední kvadratickou chybu (MSE) mezi původním a rekonstruovaným obrazem pro složky $R$, $G$ a $B$:

$$
\mathrm{MSE} = \frac{1}{m \cdot n} \sum_{i=1}^{m} \sum_{j=1}^{n} (X_{ij} - Y_{ij})^2,
$$

   kde $X$ a $Y$ jsou původní a rekonstruované obrazy.

   (b) Vypočítej směrodatnou odchylku pro chyby v každé složce.

**Funkce** `DWT(obraz)`

1. Vstup: Matice obrazu.
2. Výstup: Čtyři submatice $LL$, $LH$, $HL$, $HH$ po diskretní vlnkové transformaci.
3. Postup:
   (a) Definuj Haarovy filtry: low-pass a high-pass.
   (b) Aplikuj filtry na řádky:
      i. Použij low-pass filtr na všechny řádky → matice $L$.
      ii. Použij high-pass filtr na všechny řádky → matice $H$.
   (c) Zredukuj počet řádků (vyber každý druhý řádek) v maticích $L$ a $H$.
   (d) Aplikuj filtry na sloupce:
      i. Použij low-pass filtr na $L \rightarrow LL$.
      ii. Použij high-pass filtr na $L \rightarrow LH$.
      iii. Použij low-pass filtr na $H \rightarrow HL$.
      iv. Použij high-pass filtr na $H \rightarrow HH$.
   (e) Zredukuj počet sloupců (vyber každý druhý sloupec) ve všech čtyřech maticích.
   (f) Vrátí submatice $LL$, $LH$, $HL$, $HH$.

**Funkce** `iDWT(LL, LH, HL, HH)`

1. Vstup: Submatice $LL$, $LH$, $HL$, $HH$.
2. Výstup: Rekonstruovaná matice obrazu.
3. Postup:
   (a) Obnov původní počet řádků:
      i. Inicializuj matici $L$ a $H$ s dvojnásobným počtem řádků.
      ii. Zapiš hodnoty $LL$ do lichých řádků $L$.
      iii. Zapiš hodnoty $LH$ do sudých řádků $L$.
      iv. Zapiš hodnoty $HL$ do lichých řádků $H$.
      v. Zapiš hodnoty $HH$ do sudých řádků $H$.
   (b) Obnov původní počet sloupců:
      i. Inicializuj matici $A$ s dvojnásobným počtem sloupců.
      ii. Zapiš hodnoty $L$ do lichých sloupců $A$.
      iii. Zapiš hodnoty $H$ do sudých sloupců $A$.
   (c) Rekonstruuj původní hodnoty:

i. Pro každý blok $2 \times 2$ v matici $A$:

A. Vypočítej původní hodnoty na základě vzorců pro Haarovu vlnku.

ii. Zapiš rekonstruované hodnoty do matice výstup.

(d) Vrátí rekonstruovanou matici.

### 2.2 Převod na zig-zag sekvence

Využívá se pro uspořádání prvků matice, pro následnou efektivní kompresi pomocí Huffmanova kódování. Sekvence globálně probíhá ve směru hlavní diagonály matice, podrobně probíhá ve směru kolmém.

V našem zpracování úlohy provádí převod funkce *zig_zag* a převod ze sekvence zpět na matici funkce *inv_zig_zag*.

Algoritmus začíná v bodě (1,1) v matici a začíná přidáním prvku napravo. Poté se vytvoří pomocná matice, obecně o velikosti (m,m), kde m je maximální hodnota z indexů, kde se aktuálně nacházíme. V závislosti na směru, který je aktuální, se matice přetočí podle osy x nebo y, a vybere se její diagonála. Tato diagonála se přidá do sekvence. Následuje připojení sousedního prvku v matici (souseda koncového prvku sub-diagonály), podle směru je to prvek buď pod nebo v pravo, v první půlce matice, nebo prvek nad nebo vlevo v druhé půlce matice. Směry se mění ve chvíli, kdy dojdeme na nějaký okraj matice. Algoritmus končí přidáním posledního prvku matice.

Inverzní algoritmus je zpracován na základě stejného principu, je však třeba dopředu znát rozměr matice (pokud není vždy čtvercová).

### 2.3 Huffmanovo kódování

Algoritmus přiřazuje binární kódy hodnotám obrazu, na základě jejich četností. Nejčastěji se vyskytujícím se hodnotám přiřadí krátké kódy a delší kódy hodnotám, které se vyskytují méně často. Děje se tak na základě tvorby binárního stromu od spodu, ten je pak třeba při dekompresi, pro správné přidělení hodnot původního obrazu binárním kódům.

V rámci našeho zpracování úlohy je Huffmanovo kódování provedeno pomocí funkce *huffman* a jeho dekódování pomocí funkce *huffman_decode*.

V našem zpracování, které proběhlo v Matlabu, byla pro tvoření stromu použita proměnná typu *cell*. Ta umožňuje do jisté míry spojovat pole, které nemají stejnou velikost. Pro tuto úlohu se však více hodí datové struktury jako je například *dictionary*. Kódování tedy není zcela optimální, do binární podoby však správně hodnoty převádí a inverzní funkce spravně převadí binární hodnoty zpět na původní.

# 3 Výsledky

## 3.1 Obrázky před kompresí

Obrázek 2: barevný obrázek

Obrázek 3: obrázek ve stupních šedi

## 3.2 Převzorkování rastru a diskrétní kosinova transformace

**q = 10**

Obrázek 4: Aplikace DCT na barevný obrázek, q = 10

Obrázek 5: Aplikace DCT na obrázek ve stupních šedi, q = 10

**q = 50**

Obrázek 6: Aplikace DCT na barevný obrázek, q = 50

Obrázek 7: Aplikace DCT na obrázek ve stupních šedi, q = 50

**q = 70**

Obrázek 8: Aplikace DCT na barevný obrázek, q = 70

Obrázek 9: Aplikace DCT na obrázek ve stupních šedi, q = 70

### 3.3 Diskrétní vlnková transformace

Obrázek 10: Aplikace DWT na barevný obrázek

Obrázek 11: Aplikace DWT na obrázek ve stupních šedi

### 3.4 Střední kvadratická odchylka $\sigma$

| metoda | $\sigma_R$ | $\sigma_G$ | $\sigma_B$ |
|---|---|---|---|
| DCT, q = 10 | 23.688 | 23.587 | 24.311 |
| DCT, q = 50 | 19.676 | 19.893 | 19.793 |
| DCT, q = 70 | 19.462 | 19.719 | 19.691 |
| DWT | 26.923 | 27.298 | 26.623 |

Tabulka 1: $\sigma_{R,G,B}$ pro barevný obrázek

| metoda | $\sigma_R$ | $\sigma_G$ | $\sigma_B$ |
|---|---|---|---|
| DCT, q = 10 | 24.417 | 24.108 | 24.352 |
| DCT, q = 50 | 20.678 | 20.665 | 20.708 |
| DCT, q = 70 | 20.749 | 20.749 | 20.779 |
| DWT | 28.438 | 28.435 | 28.486 |

Tabulka 2: $\sigma_{R,G,B}$ pro černobílý obrázek

## 4 Závěr

Byly vpracovány funkce a skripty pro výpočet diskrétní kosinovy transformace s převzorkováním rastru. Dále byla zpracována konverze do zig–zag sekvence a Huffanomo kódování. Diskrétní vlnková transformace byla vypracována taktéž. Ke všem kompresím byly vypracovány i inverzní funkce.

Komprese pomocí DCT poskytuje lepší výsledky pro vyšší hodnotou faktoru komprese $q$. Se zmenšující se hodnotou faktoru komprese, kvalita zrekonstruovaného obrazu rychle klesá, viz Tabulka 1. Použití JPEG komprese je pak vhodné a praktické zejména pro členité, přirozené obrázky (jako testovací obrázek) a zejména pro obrázky, které nejsou určeny k další (geometrické) analýze.

Diskrétní vlnková transformace vrací více zrnitý obraz než DCT. Porovnání pomocí střední kvadratické odchylky s DCT není možné, protože každý výpočet proběhl jinak. Obraz nebyl rozkládán na submatice a neproběhla kvantizace.

Konverze do zig-zag sekvence a z ní zpět do matice a Huffmanovo kódování jsou bezztrátové kroky, není tedy možné je numericky hodnotit.

**V Praze dne: 22.10. 2024**

**T. Černohousová**
**M. Pokorný**