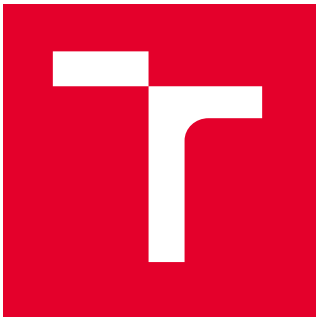


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

WEBOVÉ APLIKACE PRO PODPORU VÝUKY KOMPRIMAČNÍHO ALGORITMU JPEG

WEB APPLICATIONS SUPPORTING EDUCATION OF JPEG COMPRESSION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Valeriia Dziuina**
AUTHOR

VEDOUCÍ PRÁCE — **prof. Mgr. Pavel Rajmic, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Telekomunikační a informační systémy**

Ústav telekomunikací

Studentka: Valeriia Dziuina ***ID:*** 221273

Ročník: 3 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Webové aplikace pro podporu výuky komprimačního algoritmu JPEG

POKYNY PRO VYPRACOVÁNÍ:

V JavaScriptu vytvořte tři webové aplikace, které poslouží jako interaktivní podpora výuky v kurzech zaměřených na zpracování obrazů. Aplikace budou tématicky orientovány na kompresní algoritmus JPEG, konkrétně půjde o:
1) Ilustraci podvzorkování jasové složky a barvonosných složek obrazu,
2) Vizualizace obrazů po dekompresi,
3) Ilustrace efektu změny DCT koeficientů na vzhled obrazu.
Při tvorbě aplikací se zaměřte především na názornou podobu a funkčnost pro potřebu výuky. Ke každé aplikaci vytvořte webovou stránku na které budou teoretické základy, s případnými odkazy na detailnější zdroje.

DOPORUČENÁ LITERATURA:

[1] Beneš, B.; Sochor, J.; Felkel, P.; Žára, J.: Moderní počítačová grafika. Computer Press, Brno, 2005.
[2] Gonzalez, R.C.; Woods, R.E.: Digital Image Processing. Třetí vydání. Pearson; 2007. ISBN 978-0131687288

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 26.5.2023

Vedoucí práce: prof. Mgr. Pavel Rajmic, Ph.D.

prof. Ing. Jiří Mišurec, CSc.
předseda rady studijního programu

ABSTRAKT

Daná bakalářská práce se zabývá problematikou ztrátové komprese obrazových dat realizované algoritmem JPEG. Je zde popsán samotný algoritmus JPEG a také nezbytné pro jeho pochopení podklady, jako například fyzikální a fyziologická podstata vzniku barvy, její základní parametry a barevné modely RGB a YC_BC_R. Výsledkem této práce jsou tři webové aplikace, každá ze kterých by měla názorně demonstrovat průběh určitých kroků daného algoritmu a jejich vliv na výsledný obraz. Tyto aplikace jsou nejprve zaměřené na jednodušší pochopení procesu komprese obrazových dat v rámci algoritmu JPEG.

KLÍČOVÁ SLOVA

JPEG, RGB, YC_BC_R, komprese, kvantovani, podvzorkovani, DCT, huffmanův kód, JavaScript

ABSTRACT

This bachelor thesis deals with the problem of lossy compression of image data implemented by the JPEG algorithm. The JPEG algorithm itself is described, as well as the background necessary for its understanding, such as the physical and physiological nature of colour formation, its basic parameters and RGB and YC_BC_R colour models. The result of this work is three web applications, each of which should demonstrate the flow of certain steps of the algorithm and their influence on the final image. These applications are first aimed at simplifying the understanding of the image data compression process within the JPEG algorithm.

KEYWORDS

JPEG, RGB, YC_BC_R, compression, quantization, subsampling, DCT, huffman code, JavaScript

Vysázeno pomocí balíčku `thesis` verze 4.07; `http://latex.feec.vutbr.cz`

DZIUINA, Valeriia. *Webové aplikace pro podporu výuky komprimačního algoritmu JPEG*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2023, 45 s. Bakalářská práce. Vedoucí práce: prof. Mgr. Pavel Rajmic, Ph.D

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Valeriia Dziuina
VUT ID autora:	221273
Typ práce:	Bakalářská práce
Akademický rok:	2022/23
Téma závěrečné práce:	Webové aplikace pro podporu výuky komprimačního algoritmu JPEG

Prohlašuji, že svou závěrečnou práci jsem vypracovala samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autorka uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušila autorská práva třetích osob, zejména jsem nezasáhla nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědoma následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autorky*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Ráda bych poděkovala vedoucímu bakalářské práce panu prof. Mgr. Pavlu Rajmicovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam výpisů

Úvod

V dnešní době se všude používá obrovské množství obrázků a ilustrací. Problém větší velikosti těchto souborů postihl každého uživatele mobilních zařízení a počítačů, protože jeden plnobarevný obraz může zabrat desítky megabytů. Jako řešení tohoto problému bylo vyvinuto velké množství komprimačních algoritmů. V oblasti komprese obrazu je jedním z nejrozšířenějších algoritmů JPEG.

Hlavním cílem dané práce je vytvoření tří webových aplikací na podporu výuky komprimačního algoritmu JPEG. Applety jsou zaměřené na pochopení jednotlivých kroků algoritmu, odhalení základních principů daného algoritmu a názorné ukázky, co se právě uvnitř něj děje. Aby byly webové aplikace interaktivní, byl použit programovací jazyk JavaScript. V rámci této bakalářské práce je implementována aplikace „Ilustrace podvzorkování jasové složky a barevných složek obrazu“, která popisuje proces transformace a redukce barev v obrazech, aplikace „Ilustrace efektu změny DCT koeficientů na vzhled obrazu“, která se zaměřuje na pochopení procesu změny obrazových dat během DCT a kvantování, a aplikace „Vizualizace obrazů po dekompresi“, která demonstruje vliv zaokrouhlení kvantovaných DCT koeficientů na dekomprimovaný obraz.

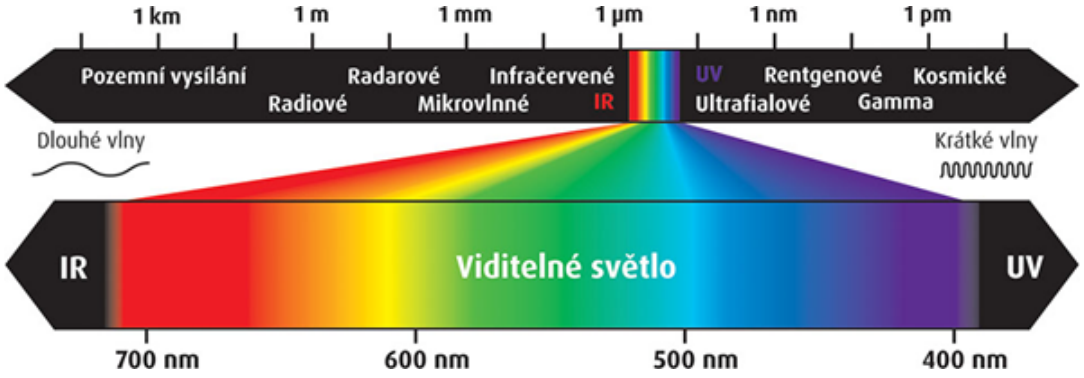
Text této bakalářské práce je rozdělen do dvou částí. V teoretické části je popsáno, jak funguje samotný algoritmus JPEG, jak lidské oko vnímá barvy a jak to souvisí s JPEG. V praktické části jsou popsány použité technologie. Podrobně jsou popsány samotné applety a jaké funkce byly k tomu implementovány.

1 Teoretický úvod

1.1 Barva a její vnímání člověkem

Barva reprezentuje subjektivní kvalitativní vlastnost elektromagnetického záření v rámci optického spektra. Je určená na základě vznikajícího fyziologického zrakového vjemu, jež závisí na řadě fyzických, fyziologických a psychologických faktorů. To znamená, že každý jednotlivý člověk v určitém okamžiku může barvu se stejnými fyzikálními vlastnostmi vnímat jinak [1].

Možnost rozkladu světla na spojité spektrum jednotlivých barev je známá již velmi dlouho. Již v roce 1666 anglický vědec Isaac Newton experimentálně zjistil, že sluneční paprsek procházející skleněným hranolem se na výstupu rozděluje na spektrum barev od červené po fialovou (obr.1.1). Navíc zde není žádné rozdělení na konkrétní barevné rozsahy. Každá z barev plynule přechází do jiné [3].



Obr. 1.1: Spektrum záření [2].

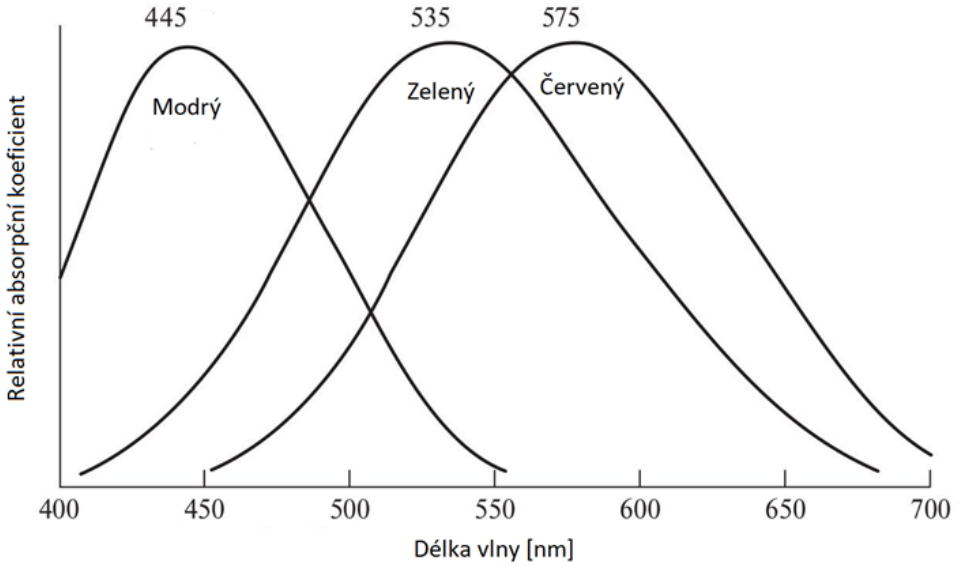
1.1.1 Barevný vjem člověka

Z fyzikálního hlediska je barva vnímaná člověkem jako barva předmětu určena charakterem světla odraženého od předmětu. Člověk je schopen vnímat pouze malou část elektromagnetického spektra a to je právě světlo o vlnové délce 380–750 nm. Objekt, který odráží světlo s maximální intenzitou v celém viditelném spektrálním rozsahu, se pozorovateli jeví jako bílý. Předmět, který odráží světlo o určité vlnové délce viditelného spektra, získá určitou barvu [3].

Z fyziologického pohledu vzniká pocit barvy v mozku díky receptorům lidské sítnice – čípků. Prostřednictvím velkého počtu experimentů bylo zjištěno, že všechny čípky lidského oka, kterých je celkem několik milionů, se dělí podle jejich citlivosti ke spektrálnímu složení světla do tří hlavních skupin [3]:

- Červená – 65 %
- Zelená – 33 %
- Modrá – 2 %

Tím pádem lidské oko vnímá barvy jako různé kombinace tří základních barev (obr.1.2): červené (R), zelené (G) a modré (B).



Obr. 1.2: Průměrná spektrální charakteristika citlivosti lidských barevných receptorů – čípků [3].

1.1.2 Základní parametry barvy

Jakoukoliv barvu lze popsat určitou sadou parametrů jasu, sytosti a odstínu.

Jas (světlost) – stejně syté odstíny připisované stejné barvě spektra se mohou od sebe lišit stupněm světlosti, která vyjadřuje achromatickou představu o intenzitě dopadajícího světla [3]. Intenzita pak měří stupeň odraženého světla – jak světlá, nebo tmavá barva je. Jas na rozdíl od intenzity je subjektivní vlastností (obr.1.3, kterou téměř nelze změřit, nicméně je jedním z klíčových parametrů pro popis barevného vjemu člověka.

Sytost definuje rozsah od čisté barvy (100 %) po šedou (0 %) při konstantní úrovni jasu (obr.1.4). Čistá barva je plně sytá [4]. Z hlediska vnímání lidským okem ovlivňuje sytost stupeň čistoty nebo jasnosti barvy.

Obr. 1.3: Změna světlosti objektu v závislosti na jeho pozadí. Všechny centrální čtverce mají stejný jas, ale v závislosti na pozadí se jeví světlejší nebo tmavší [3].

Obr. 1.4: Stupně sytosti pro červenou barvu [4].

Odstín (barevny ton) charakterizuje dominantní barvu vnímanou pozorovatelem, přičemž každý vnímaný barevný tón odpovídá určité spektrální barvě [3]. Tak na níže uvedeném obrázku 1.5 jsou barvy azurová, blankytná, safírová a akvamarínová. Každá z nich má své určité parametry, ale současně mají modrý odstín [4].

Obr. 1.5: Různé barvy modrého odstínu [4].

Sloučením sytosti a odstínu vzniká pojem chromatičnost neboli barevnost [3]. Tím pádem lze barvu popsat dvěma parametry světlosti a chromatičnosti. Na daném způsobu definování barvy jsou založeny některé barevné modely. Jedním z nich je model YC_BC_R, který se používá v algoritmu JPEG a o kterém se podrobněji mluví v kapitole 1.2.2.

1.2 Barevné modely

Hlavním účelem barevného modelu je naznačení některého obecně přijatého standardu popsání barev. V podstatě je to prostor souřadnicového systému, v němž je každá barva definována jednotlivým bodem.

1.2.1 Model RGB

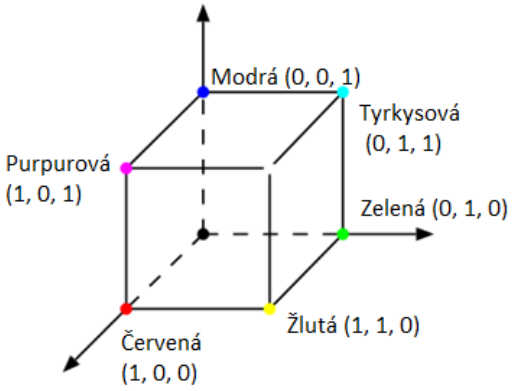
Jedná se o aditivní barevný model, který popisuje způsob kódování barvy pro její další reprodukci pomocí tří základních barev – červené (R), zelené (G) a modré (B) [5]. Volba barev je dána fyziologií lidského oka, receptory jehož sítnice jsou nejvíce citlivé právě na tyto barvy (viz.1.1.1).

RGB model je aditivní v tom smyslu, že je založen na míšení světelných paprsků (obr.1.6). Úplná absence záření určuje černou barvu. Mícháním všech tří složek v jejich maximálním poměru vznikne bílá. Když jedna ze složek má největší intenzitu a ostatní jsou nulové, jde o primární barvu. Sekundární barva je pak tvořena součtem dvou základních barev stejné intenzity – mícháním zelené a modré vzniká tyrkysová, modré a červené – purpurová, červené a zelené – žlutá. Každá sekundární barva je doplňkem jedné primární barvy: tyrkysová doplňuje červenou, purpurová doplňuje zelenou a žlutá doplňuje modrou [6].

Obr. 1.6: Aditivní míchání barev [6].

Model RGB se nejčastěji znázorňuje ve tvaru krychle ležícím v souřadnicovém systému s osami r, g a b a jeho složky nabývají hodnoty z intervalu $\langle 0, 1\rangle$ (obr.1.7). Černá barva leží v počátku [0, 0, 0] krychle, bílá je reprezentována vektorem [1, 1, 1]. Vrcholy krychle, které jsou umístěné na osách, představují primární barvy, zatímco

zbývající vrcholy reprezentují barvy sekundární. Na diagonále od černé barvy k bílé jsou umístěny stupně šedé [5].



Obr. 1.7: Barevný model RGB [6].

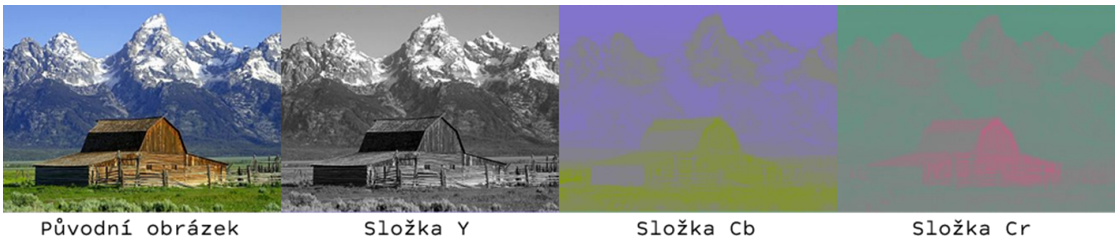
V počítačích je každá ze souřadnic reprezentována jedním oktetem, jehož hodnoty jsou pro usnadnění označeny celými čísly od 0 do 255 včetně, kde 0 je minimální a 255 je maximální intenzita [5]. RGB se považuje za základní barevný model pro většinu grafických aplikací, protože výsledný obrázek nepotřebuje žádnou další konverzi pro zobrazení na obrazovce.

1.2.2 Model YC_BC_R

YC_BC_R je barevným modelem, který se používá u videa a digitální fotografie. Tento model je ještě více ponořen do fyziologie oka člověka. Je rozdělen do tří složek – jasová Y a dvě chromatické C_B a C_R (obr.1.8), stejně jako jsou rozděleny receptory sítnice na tyčinky, které odpovídají za vytvoření celkového obrazu a vnímaní kontrastu, a kuželové čípky, které nám umožnují barevné vidění [7]. Jelikož je lidské oko citlivější na změnu jasu, daný barevný model byl založen na principu vyčlenění jasu do samostatné složky. Proto je nejvíce užitečný při komprese obrazu.

1.2.3 Převod z RGB do YC_BC_R a zpátky

Podle standardu ITU-R BT.601 má hodnota Y být v intervalu [0, 1] a hodnoty C_B, C_R v intervalu $[-0,5, 0{,}5]$ [7]. Ale pro realizaci algoritmu JPEG je potřeba převést tyto hodnoty do digitální podoby, aby byly v 8bitovém intervalu 0 až 255. Na to se používají vzorce uvedené níže [9]:



Obr. 1.8: Převod obrazu do barevného modelu YC_BC_R a vzhled jeho luminancni a chrominančních složek [8].

$$\begin{aligned} Y &= 0 + (0,299 \cdot R) + (0,587 \cdot G) + (0,114 \cdot B) \\ C_B &= 128 - (0,168736 \cdot R) - (0,331264 \cdot G) + (0,5 \cdot B) \\ C_R &= 128 + (0,5 \cdot R) - (0,418688 \cdot G) - (0,081312 \cdot B) \end{aligned} \tag{1.1}$$

$$\begin{aligned} R &= Y + 1,402 \cdot (C_R - 128) \\ G &= Y - 0,34414 \cdot (C_B - 128) - 0,71414 \cdot (C_R - 128) \\ B &= Y + 1,772 \cdot (C_B - 128) \end{aligned} \tag{1.2}$$

Při prací s šedotónovymi obrazy se barevné složky C_B a C_R ignoruji.

1.3 Komprimační algoritmus JPEG

V roce 1992 byl vyvinut organizací Joint Photographic Experts Group první mezinárodní standard komprese pro šedotónové a barevné nepohyblive snímky s hladkým přechodem v tónu. Tento standard byl pojmenován jako akronym od „Joint Photographic Experts Group“ a je známý jako JPEG [10]. O dva roky později mu byl přiřazen ISO standard ISO/IEC 10918 [11]

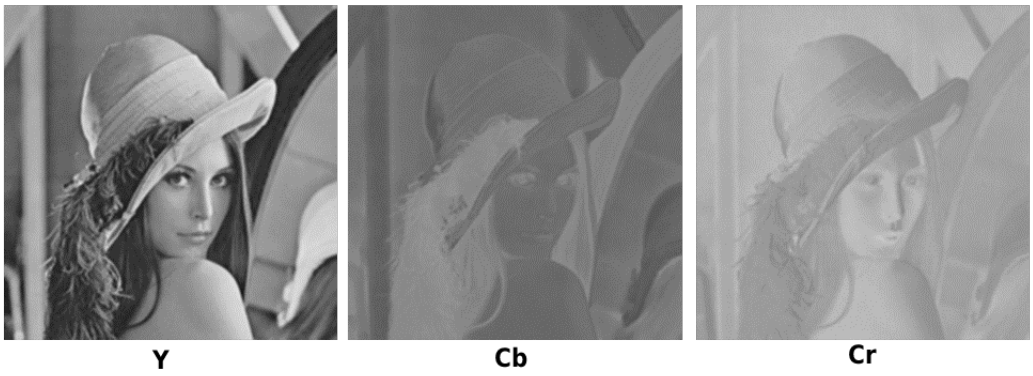
V praxi JPEG umožňuje dosáhnout kompresního poměru 1:25, což znamená, že velikost obrazových dat se sníží, ale zachová se dobrá kvalita obrazu [12]. Tato schopnost JPEG efektivně ukládat obrazová data vede k tomu, že se nejčastěji používá u digitálních obrazů, zejména u fotografií.

Existují dvě zakladni metody komprese dat - ztrátová a bezeztrátová. Algoritmus bezeztrátové komprese se liší od ztrátového tím, že obraz po dekompresi přesně odpovídá původnímu před zahájením procesu komprese, ale kompresní poměr je nizky. Naopak ztrátová metoda komprese umožňuje dosáhnout mnohem většího kompresního poměru, ale na druhou stranu dochází ke ztrátě určitého množství informací.

Konkrétně JPEG ve svém algoritmu využívá kombinaci obou způsobů. Během výpočtu kvantovaných DCT koeficientů se ztrácí některé množství obrazových dat. Zatímco RLE a Huffmanův algoritmus pouze překódují informaci za účelem snížení bitové rychlosti, přičemž žádná obrazová data se neztrácí. Tyto vlastnosti JPEG umožňují dosáhnout velkého kompresního poměru a zároveň vysoké kvality obrazu.

1.3.1 Transformace barev

Prvním krokem daného algoritmu je transformace barev, při které se obrázek ze svého primárního barevného modelu RGB převádí do modelu YC_BC_R. Na to se používá vztah (1.1). Tento barevný model odděluje informaci o světlosti od informace o barvě, s jehož pomocí je hlavní informační obsah, který je v RGB rozprostřen ve všech třech kanálech, nyní soustředěn v kanálu světlosti. To je dobře vidět z obrázku 1.9, kde kanál světlosti Y nese v sobě mnohem více informací než barevné kanály C_B a C_R. Proto pak budou jasová a dvě chrominanční složky zpracované samostatně [14].



Obr. 1.9: Jašová a barvonosné složky obrazu po transformaci barev [13].

1.3.2 Podvzorkování barvonosných složek

Tyčinky, kterých je priblizne dvacetkrát vice nez cipek, jsou hlavnim nastrojem pro tvoreni obrazu v nasem mozku, z cehoz plyne, ze lidské oko je citlivější na změnu jasu než na změnu barvy. Z toho důvodu byl v komprimačním algoritmu JPEG zvolen model YC_BC_R, aby docházelo k redukci dat pouze v barvonosných složkach C_B a C_R a jasova složka Y byla zachovana v puvodnim stavu.

Podvzorkováním se míní redukce barvových dat obrázku. Existuje několik možností jeho provedení. Nejprve je nutné zmínit, že podvzorkování se provádí pro každou barvonosnou složku zvlášť. Celý obrázek se dělí na určité bloky pixelů. Pak se

vypočítá průměrná hodnota všech pixelů v daném bloku. A nakonec se vypočtená hodnota přiřazuje každému pixelu v bloku [9].

Schémata podvzorkování:

- „4:2:0“ – nejčastěji používané schéma, při němž je obrázek rozdělen na čtverce pixelů 2×2. Dál je těmto superpixelům přiřazena průměrná hodnota vypočtená ze všech jeho čtyř pixelů. Výsledná komprese v tomto případě je 50 % [15].
- „4:2:2“ – při využití daného schématu se obrázek dělí na bloky 2×1. Výsledná komprese je 33 % [15].
- „4:4:4“ – toto schéma se naopak používá, když je potřeba zachovat informaci o barvě, každý pixel se v tomto případě zpracovává samostatně. Tím pádem ke kompresi v daném kroku nedochází [9].

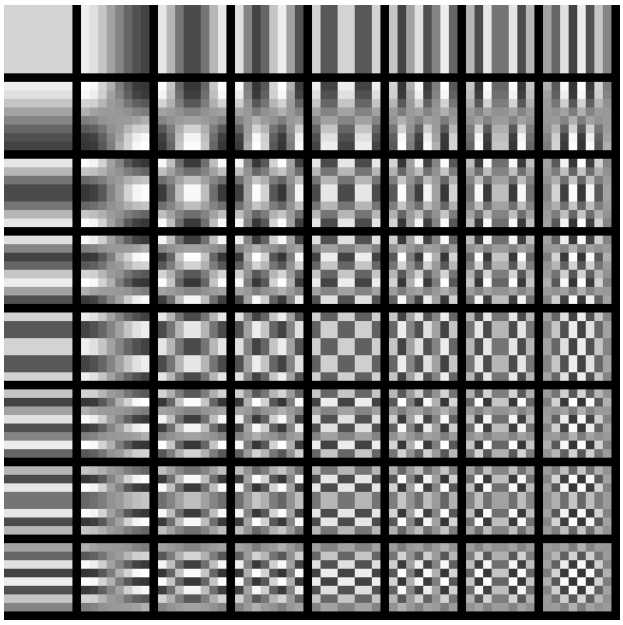
1.3.3 Diskrétní kosinová transformace

Dalším krokem se obrázky jednotlivých složek s redukovanou informací o barvě rozdělí na bloky 8×8 pixelů. Z duvodu přiblížení hodnot výstupních koeficientů k nule, od hodnot jednotlivých složek se odečte 128 [15]. Dál se každý koeficient v bloku projde vzorcem diskrétní kosinové transformace (DCT). Pro příklad DCT a následující kvantizace bude použita níže uvedená matice g, obsahující hodnoty pixelů.

$$g = \begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}.$$

DCT (Diskrétní kosinová transformace) je jednou variantou diskrétní transformace, která je založena na systému kosinových funkcí. Tyto kosinové funkce se nazývají bázové. A pro matici 8×8 dvourozměrné DCT reprezentované vzorcem (1.3) vysledkem je celkem 64 bázových funkcí[17]. Tyto funkce lze následně ilustrovat pomocí obrázku 1.10.

Diskrétní kosinová transformace je podobná diskrétní Fourierově transformaci, ale na rozdíl od ní produkuje pouze reálné hodnoty. Používá se hlavně ve ztrátových komprimačních algoritmech, například MPEG a JPEG. Algoritmus JPEG využívá dvourozměrnou variantu diskrétní kosinové transformace DCT-II (1.4) a zpětnou DCT (IDCT) pro dekódování obrazu (1.5) [17].



Obr. 1.10: Bázové funkce diskrétní kosinové transformace pro matice 8×8 [16].

$$\cos \frac{\pi \left(2x+1\right) u}{16} \cos \frac{\pi \left(2y+1\right) v}{16} \tag{1.3}$$

$$G\left(u,\ v\right) = \frac{1}{4} C\left(u\right) C\left(v\right) \sum_{x=0}^{7} \sum_{y=0}^{7} g\left(x,y\right) \cos \frac{\pi \left(2x+1\right) u}{16} \cos \frac{\pi \left(2y+1\right) v}{16}, \tag{1.4}$$

$$g\left(x,y\right) = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C\left(u\right) C\left(v\right) G(u,v) \cos \frac{\pi(2x+1)u}{16} \cos \frac{\pi(2y+1)v}{16}, \tag{1.5}$$

$$C\left(u\right) C\left(v\right) = \begin{cases} 1/\sqrt{2}\ pro\ u,\ v = 0 \\ 1\ pro\ u, v > 0 \end{cases}$$

Výsledkem je matice DCT koeficientů. Tato metoda umožňuje roztřídit frekvence od nizkych v levém horním rohu po vysoké v pravém dolním rohu. Vysoké frekvence lidské oko rozpoznává nejméně, proto se daný atribut DCT pak využívá při kvantování, kde podstatná část vysokých frekvencí bude vynulována. Jako prvni v matice je umisten stejnosměrný DC koeficient, který schovává větší část informace v rámci daného bloku 8×8. Je možné si také všimnout, že ostatní 63 stridave AC koeficienty mají mnohem menší rozsah nebo dokonce nulovou hodnotu. Timto

zpusobem algoritmus JPEG je schopen soustredit většinu signalu v jednom bodě. Příklad výsledných DCT koeficintů je znázorněn v podobě matice G níže:

$$G = \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix}$$

1.3.4 Kvantování DCT koeficientů

Při kvantizaci se každý prvek DCT matice dělí odpovídajícím prvkem kvantizační matice **Q** a zaokrouhluje se na celá čísla [15].

$$B_{u,v} = \mathbf{round}\left(\frac{G_{u,v}}{Q_{u,v}}\right) \tag{1.6}$$

Velmi užitečnou vlastností algoritmu JPEG je možnost pomocí určitých kvantizačních matic získat různé úrovně komprese a kvality obrazu. To umožňuje uživateli vybrat úrovně kvality v rozsahu od 1 do 100 (kde 1 poskytuje nejhorší kvalitu obrazu a maximální kompresi, 100 – nejlepší kvalitu a minimální kompresi [18]. Tím pádem je možné upravit poměr kvalita/komprese podle různých potřeb.

Kvantizační matice pro kvalitu Q = 50, která je specifikovaná v původním standardu JPEG, vypadá následovně:

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}. \tag{1.7}$$

Ostatní kvantizační matice jsou odvozené z matice Q(50) tímto způsobem [15]:

$$\mathbf{Q}(Q) = \mathbf{round}(\mathbf{Q}(50) \cdot a), kde\ a = \begin{cases} 50/Q\ pro\ Q \in 1,\ \ldots,\ 49 \\ 2 - 2/Q\ pro\ Q \in 51,\ \ldots,\ 99 \end{cases} \tag{1.8}$$

Prvek, který se rovná nule po vypočítání jednotlivé matice, se mění na jedničku, to znamená, že v daném bodě kvantizace neproběhne. Tedy při Q = 100 se celá matice bude skládat ze samých jedniček [15].

Po kvantování matice DCT koeficienty z příkladu vypadají takto:

$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{1.9}$$

1.3.5 Kódování kvantovaných koeficientů

Po kvantizaci se stejnosměrné složky (DC) a střídavé složky (AC) kódují zvlášť.

Kódování DC koeficientů

Pro kódování DC koeficientů JPEG využívá DPCM (Differential Pulse Code Modulation) a Huffmanův kód. DC koeficienty sousedních bloků se v reálných obrazech mění pomalu, proto není výhodné ukládat jejich absolutní hodnoty. DPMC v tomto případě umožní zakódovat ne skutečnou hodnotu DC členů, ale rozdíl mezi stejnosměrnou složkou současného a předešlého bloku. Přitom pro DC koeficient prvního bloku platí, že DC koeficient předchozího bloku je nulový [19].

Pro vstupní pixely o bitové hloubce 8 bitů platí, že dynamický rozsah DCT koeficientů je 11 bitů. Po výpočtu rozdílů stejnosměrných koeficientů sousedních bloků ale dynamický rozsah vzroste do 12 bitů, proto se pro kódování používá Huffmanova tabulka s dvanácti kategoriemi [15].

Před rozdílovou hodnotu se zapisuje číslo kategorie, aby dekodér věděl, kolik bitů má přečíst. Zapisuje se číslo kategorie do souboru pomocí Huffmanových tabulek, které se liší podle typu koeficientu (DC a AČ) a typů složek obrazu (jasová a barvonosná), jsou celkem 4 [19].

Samotná rozdílová hodnota se převádí z decimálního do binárního tvaru, přičemž záporné hodnoty se pak ještě zrcadlí [19]. Například, rozdílová hodnota 14 v binární soustavě vypadá jako 1110, ale hodnota −14 se v tomto případě bude zapisovat jako 0001.

Kódování AC koeficientů

Před zakódováním střídavé složky AC se nejprve seřadí podle tzv. „zig-zag“ schématu (obr.1.11) [15]. Tím pádem vyčítání jde postupně od nižších po nejvyšší frekvence. Kvantované koeficienty vyšších frekvencí jsou často nulové a díky danému

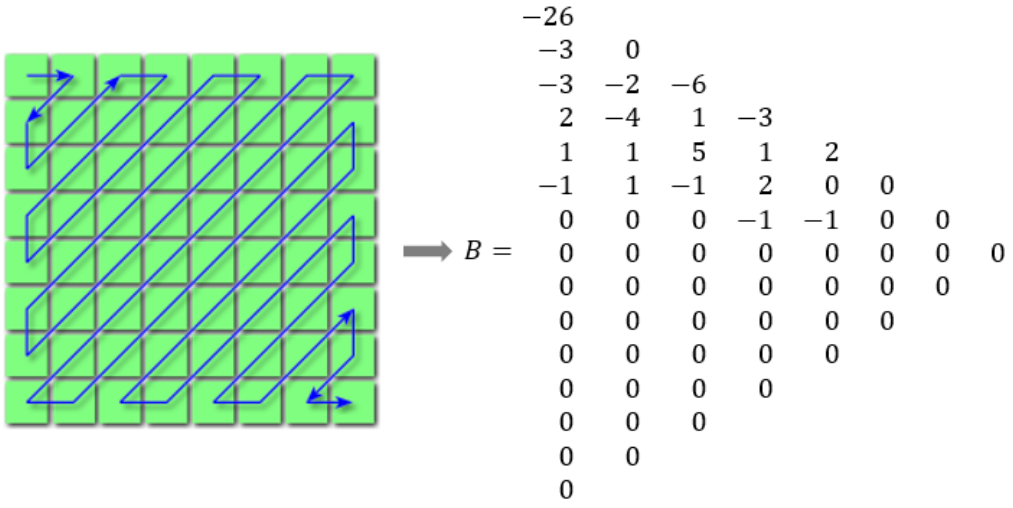
schématu je možné zvětšit úseky nulových hodnot.

Vzniklá sekvence 63 AC koeficientů se zakóduje pomocí Huffmanovy tabulky a RLE (Run-length encoding), který přispívá k bezeztrátové kompresi dat tím, že zakóduje úseky stejných po sobě jdoucích hodnot, v daném případě nul tímto způsobem [15]:

Každý nenulový AC koeficient se vyjádří dvojicí (R|S), kde

- R (0-run) je 4bitový čítač nulových koeficientů, do něj lze zapsat maximálně 15 nul.
- S (size) je 4bitové vyjádření kategorie nenulového AC koeficientu, kterému předcházel úsek nulových hodnot.
- Pro případ patnácti po sobě jdoucích nul (ZRL, Zero Run Length) se používá zápis „(15|0)“
- „(0|0)“ – pro zbývající posloupnost nul (EOB, End Of Block), bitově se značí jako 1010.
- Pro bitový zápis dvojice (R|S) se používá Huffmanova tabulka, kde jsou uvedeny všechny kombinace R a S. Daná tabulka se také vkládá do souboru.

Na konci se zapíše samotný AC koeficient v bitové podobě.



Obr. 1.11: Seřazení kvantovaných koeficientů podle cik-cak schématu.

1.3.6 Dekódování

Dekódování nutné před zobrazením obrázku probíhá opačným směrem. Nejprve proběhne bezeztrátová dekomprese z Huffmanových tabulek a z RLE. Bloky 8×8 se vynásobí použitými kvantovanými maticemi. Pak proběhne zpětná DCT (IDCT)

v každém bloku každé složky obrazu a nadvzorkování barvonosných složek. A nakonec se hodnoty pixelů převedou z YC_BC_R do jejich primárního barevného modelu RGB[15].

Přitom ani při použití nejvyšší kvality (odpovídající kvantizační matici skládající se z jedniček a absenci podvzorkování barvonosných složek) se dekomprimovaný obraz nebude přesně shodovat s původním, což je hlavně dáno nutností zaokrouhlovat DCT koeficienty a hodnoty Y, C_B, C_R na nejbližší celá čísla.

2 Praktická část

2.1 Použité technologie

Pro implementaci webových aplikací je potřeba vědět, jaké technologie se k tomu používají a jak to celkově funguje. Nedílnou součástí každé moderní webové stránky jsou: obsah stránky, stylizace prvků a interaktivita prostředí.

2.1.1 HTML

HTML (Hypertext Markup Language) – je hypertextový značkovací jazyk, který se používá k definování struktury webové stránky v prohlížeči. Pomocí něj se vytváří kostra web-kontentu. HTML definuje, jak má být zobrazen obsah webové stránky, jako například odstavec, seznam, nadpis, obrázek, formulář nebo jeden z mnoha dalších prvků. HTML je hlavním jazykem pro vytváření stránek v systému World Wide Web [20].

V rámci projektu jsou nejčastějšími komponenty pro tvorbu prostředí bloky `<div>`, obrázky vložené pomocí tagu `<img>` a prvky `<canvas>` pro zobrazení obrazových dat.

2.1.2 CSS

CSS (Cascading Style Sheets) – jazyk kaskádových stylů, který je zodpovědný za vzhled obsahu webové stránky. Jeho přítomnost zajistí, že obsah bude vypadat podle návrhu autora a bude mít jednotný styl. Pomocí CSS vlastností lze nastavit barvy prvků, písma a další aspekty vzhledu. Kromě toho CSS umožňuje oddělit prezentaci obsahu od jeho struktury do samostatného souboru, což výrazně usnadňuje úpravu vzhledu webové stránky. [20].

Důležitou částí každého appletu je ovládací oblast. K jejímu vytvoření byly použity především vlastnosti CSS, jako jsou `display: flex`, které zajišťují správné uspořádání bloků appletu, anebo `margin` pro nastavení vzdálenosti mezi rámečkem a okolím prvku.

2.1.3 JavaScript

JavaScript – je objektově orientovaný, interpretovaný programovací jazyk. Používá se jako skriptovací jazyk, pomocí kterého mohou uživatelé ovládat jednotlivé prvky webových stránek. Přítomnost JavaScriptu umožňuje reagovat na akce uživatele a spouštět skripty na základě konkrétních událostí. JavaScript poskytuje přístup

k obsahu stránek a umožňuje jej upravovat. Tudíž se vyuzívá pro vytváření různých efektů a animací na stránkách [20].

Z hlediska čitelnosti a jednoduchosti správy kódu bylo rozhodnuto oddělit některé funkce do jednotlivých tříd podle oblasti použití. Třídy SubsamplingAppletController, DCTAppletController a VisualizationAppletController jsou určeny pro obecnou správu appletů. V rámci těchto tříd se inicializují ovládací prvky stránky a prvotně se nastavují plátna. V rámci třídy Canvas se provádějí základní operace pro aktualizaci obrazových dat na vybraném plátně. Výsledky zpracování jednotlivými kroky appletu jsou uloženy v objektu PixelsData, který obsahuje pole objektů Pixel a hodnotu jejich barevného modelu.

2.2 Společné nástroje appletů

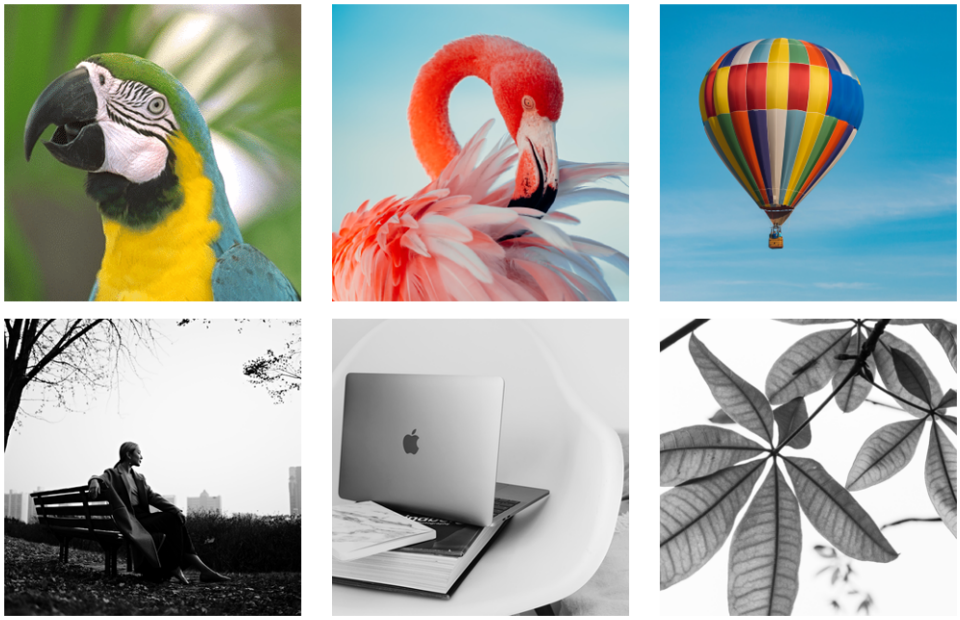
Všechny tři webové aplikace jsou vypracovány v jednotném stylu a sdílí společné funkce, které jsou realizovány v každém appletu.

2.2.1 Galerie obrázků

Každý applet obsahuje galerii tří buď barevných nebo černobílých obrázků (obr.2.1) (v závislosti na funkcionalitě jednotlivého appletu), která se nachází v levé části okna. Díky tomu má uživatel možnost vybrat si z několika variant. Tato funkce umožňuje uživateli zobrazit výsledek komprese různých obrázků. Například v aplikaci „Ilustrace podvzorkování jasové složky a barvonosných složek obrazu“ jsou prezentovány obrázky s různými kombinacemi barev, což může uživateli demonstrovat, jak různé barevné složení obrazu ovlivňuje stav barevných kanálů po transformaci do barevného modelu YC_BC_R. V dalších dvou aplikacích jsou vybírány obrázky s plynulými a ostrými přechody, s více a méně detaily, což také umožňuje lépe porozumět diskrétní kosinové transformaci a kvantizaci.

2.2.2 Lupa

Také ve všech aplikacích byl implementován mechanismus pro zvětšení obrazu. Tento modul se aktivuje při najetí myší na obrázek, pro zvětšení všech fotografií najednou je nutné stiskntout klávesu Ctrl (obr.2.2). Díky této funkci může uživatel podrobněji prohlédnout obrázky a porovnat je. Z hlediska kódu byla tato funkce implementována pomocí upravené verze externí knihovny canvas.magnifier.js [21], ze které byly odstraněny nepotřebné funkce a opraveny některé chyby, jako například rozmazání hranic pixelů. Zároveň se změnil vzhled „lupy“ a její poloha vůči kurzoru.



Obr. 2.1: Galerie obrázků, které se zpracovávají v appletech.

2.2.3 Popisy jednotlivých aplikací

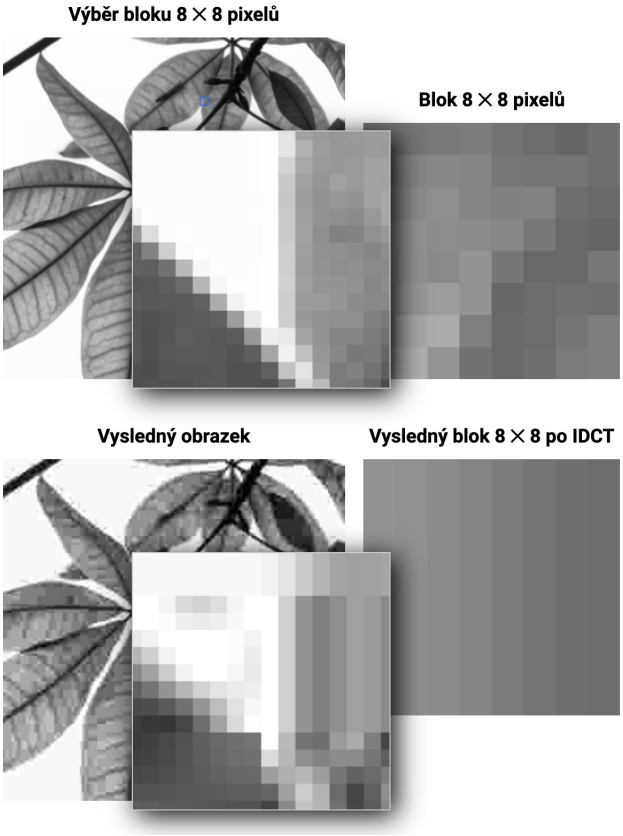
Každý applet obsahuje stručný teoretický úvod do tématu, které je realizováno v určitém appletu. Tento úvod by měl pomoci uživateli, který se s daným tématem setkává poprvé, lépe porozumět tématu, kterým se applet zabývá. Pro rychlejší orientaci v funkcionalitě je do webové aplikaci také přidán návod na jeho používání.

2.3 Applet „Ilustrace podvzorkování jasové složky a barvonosných složek obrazu“

Daný applet slouží ke znázornění prvních dvou kroků algoritmu JPEG – transformace barev do barevného modelu YC_BC_R a následující podvzorkování složek pixelu.

Veškeré ovládání této webové aplikace je realizováno pomocí tzv. přepínačů (radio button). Daný ovládací prvek umístěný v levém horním rohu každého obrázkového elementu. Po jejich rozkliknutí se zobrazí jednotlivé složky obrazu neboli samotný plnobarevný obraz. Také tento typ ovládání je zařazen do bloku „Schéma podvzorkování“. Pomocí přepínačů v daném bloku má uživatel možnost si vybrat typ podvzorkování, o který má zájem.

V aplikaci jsou rozmístěny čtyři bloky (obr.2.3). Každý blok odpovídá jedné z fází tohoto procesu a je propojen šipkami, které ukazují správné pořadí. V levé části okna jsou umístěny dva bloky s obrázky označené jako „Původní obrázek“



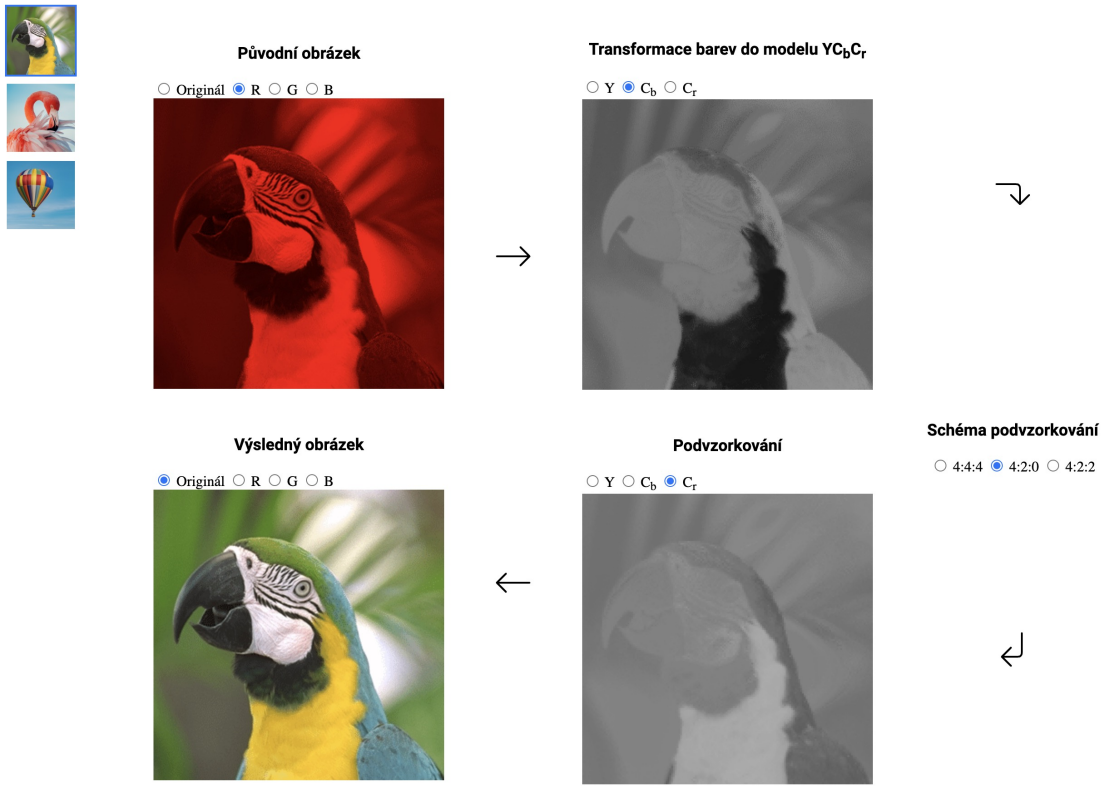
Obr. 2.2: Ukázka zvětšení původního a výsledného obrázku ve stejenem bodě pomocí klávesy Ctrl.

a „Vysledný obrázek“. Každý z těchto obrázků lze rozložit do tří základních barev RGB. Pro zvolení jedné ze složek stačí jenom stisknout odpovídající tlačítko v horní části obrázku.

V prave casti okna jsou znazornene procesy barevne transformace a redukce barev, ktere se pak promitnou ve vyslednem obrazu. V bloku, který se nazývá „Transformace barev“, lze pozorovat, jak se původní obrázek převádí ze svého primárního barevného modelu RGB do modelu YC_BC_R. Zde je umístěn obrázkový element, v němž se také lze pohybovat mezi jednotlivými složkami obrázku pomocí přepínačů.

Samotná transformace jednotlivých složek obrazu z RGB modelu na barevný model YC_BC_R se provádí pomocí následujícího JavaScript kódu (A.1), kde na vstup funkce se postupně přenáší objekty pixelů zvoleného obrázku, obsahující jednotlivé složky RGB, které se následně přepočítávají dle vzorce (1.1) na hodnoty YC_BC_R. Výstupy dané funkce jsou objekty pixelů v barevném modelu YC_BC_R.

Níže se nachází blok „Podvzorkovani“. Cílem daného bloku je ukázat, jak vypa-



Obr. 2.3: Ukázka appletu „Ilustrace podvzorkování jasové složky a barvonosných složek obrazu“.

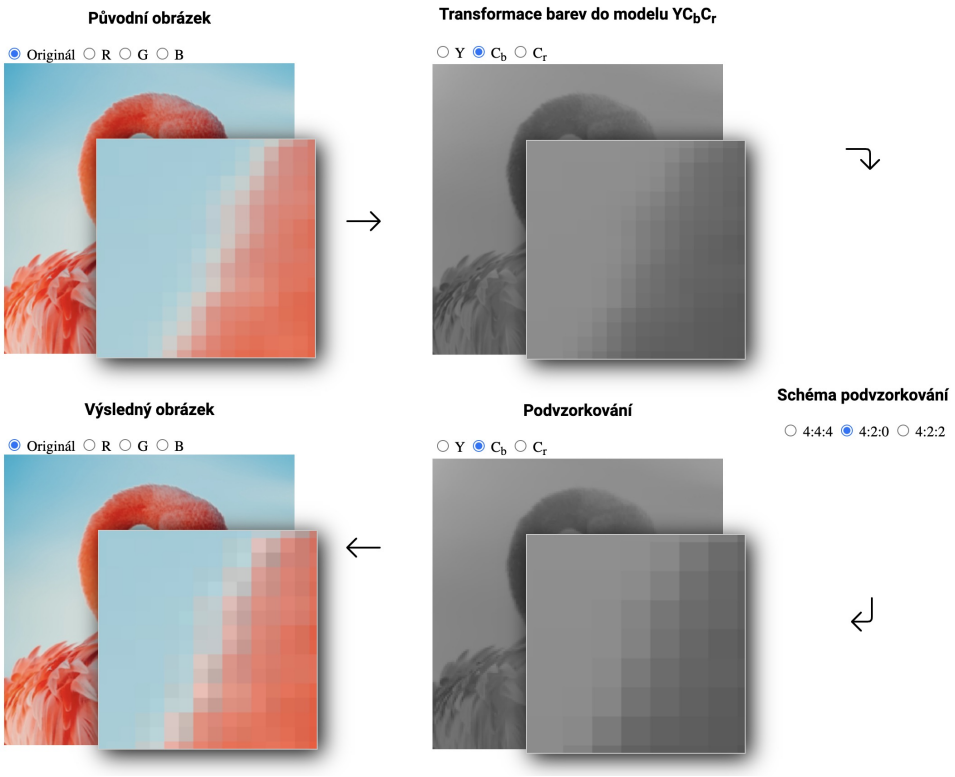
dají po podvzorkování jasová a barvonosné složky obrazu. Vpravo v bloku „Schéma podvzorkování“jsou umístěné přepínače, s jejichž pomocí lze zvolit jedno ze tří schémat redukce chrominančních složek. Po výběru odpovídajícího typu redukce se v bloku „Výsledný obrázek“ zobrazí změněný obraz (obr.2.4).

V závislosti na zvoleném schématu se podvzorkováné pixely rozdělují na jednotlivé bloky 2×2, 2×1 a 1×1 pro schémata 4:2:0, 4:2:2 a 4:4:4. Probíhá to pouze uvnitř chrominančních složek (C_B, C_R) obrazu. K tomu slouží funkce `getChunks()` (A.2), na jejímž vstupu je seznam pixelů obrázku a požadovaná délka bloku. Funkce vrací seznam bloků žádané délky.

Např. při length = 2: [1, 2, 3, 4, 5, 6, 7, 8] => [[1, 2], [3, 4], [5,6], [7, 8]]

Schéma 4:2:0 je kvůli složitějšímu tvaru bloků implementované pomocí další funkce `getSquares()` (A.3), která na začátku zavoláním funkce `getChunks()` generuje bloky, jejichž délka se rovná šířce obrázku. Tím je momentálně vytvořen seznam řádků pixelů. Potom se pixely postupně rozdělují do bloků 2×2. Tyto bloky se následně přenášejí do seznamu.

Např. při squareSize = 2: [1, 2, 3, 4, 5, 6, 7, 8] => [[1, 2, 5, 6], [3, 4, 7, 8]]



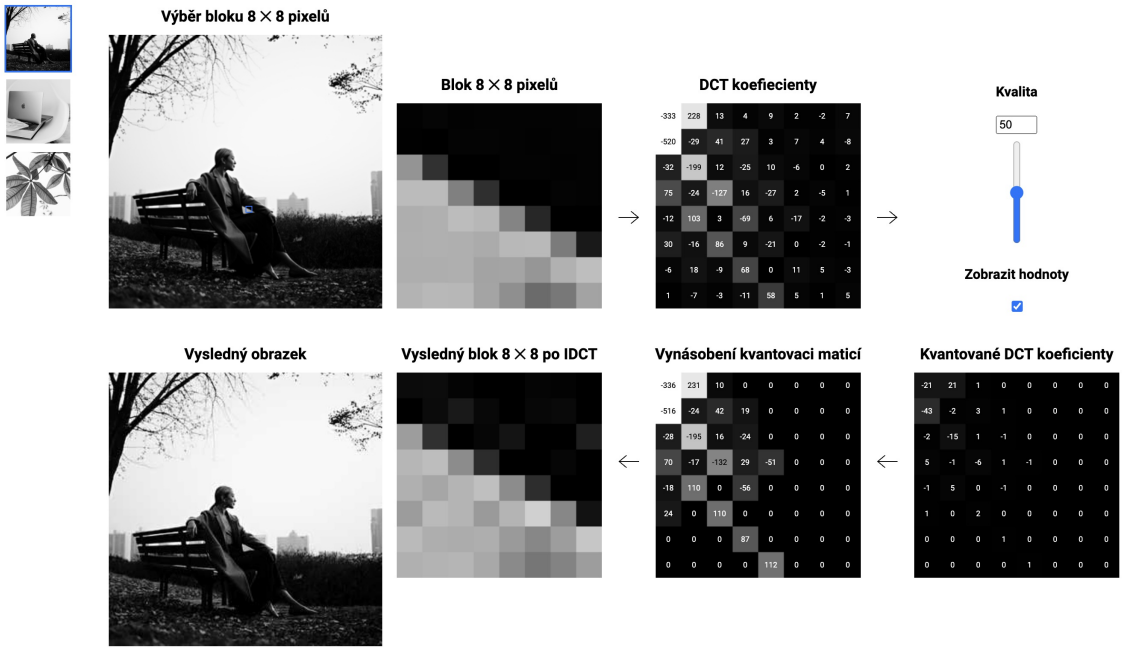
Obr. 2.4: Mezivýsledky každého kroku podvzorkování pro schéma „4:2:0“.

Samotná redukce barev probíhá pomocí funkce `getAverageOfComponent()` (A.4), kde se každému pixelu v bloku přiřazuje průměrná hodnota všech pixelů v bloku.

2.4 Applet „Ilustrace efektu změny DCT koeficientů na vzhled obrazu“

Daný applet slouží k znázornění průběhu komprese obrazových dat prostřednictvím Diskrétní Kosinové Transformace a následného kvanování DCT koeficientů v rámci algoritmu JPEG. Uživatel v rámci tohoto appletu může prozkoumat, co stojí za každým z výše uvedených procesů a jak se mění hodnoty pixelů po každém z kroků.

Okno appletu je smyslově rozděleno na dvě části (obr.2.5). Vlevo jsou umístěny vstupní a výstupní obrázky, přičemž blok se vstupním obrazem se nazývá „Výběr bloku 8×8 pixelů“. Zde uživatel může vybrat libovolný blok 8×8 pixelů, který by chtěl rozebrat detailněji a podívat se na něj zblízka. Po kliknutí na daný blok se vybraných 64 pixelů zobrazí na plátnu „Blok 8×8 pixelů“. Na plátnu „Výsledný obrázek“ se pak nachází výsledný obrázek po zpracování, který je vytvořen na základě vstupního obrázku po provedení všech kroků komprimace.



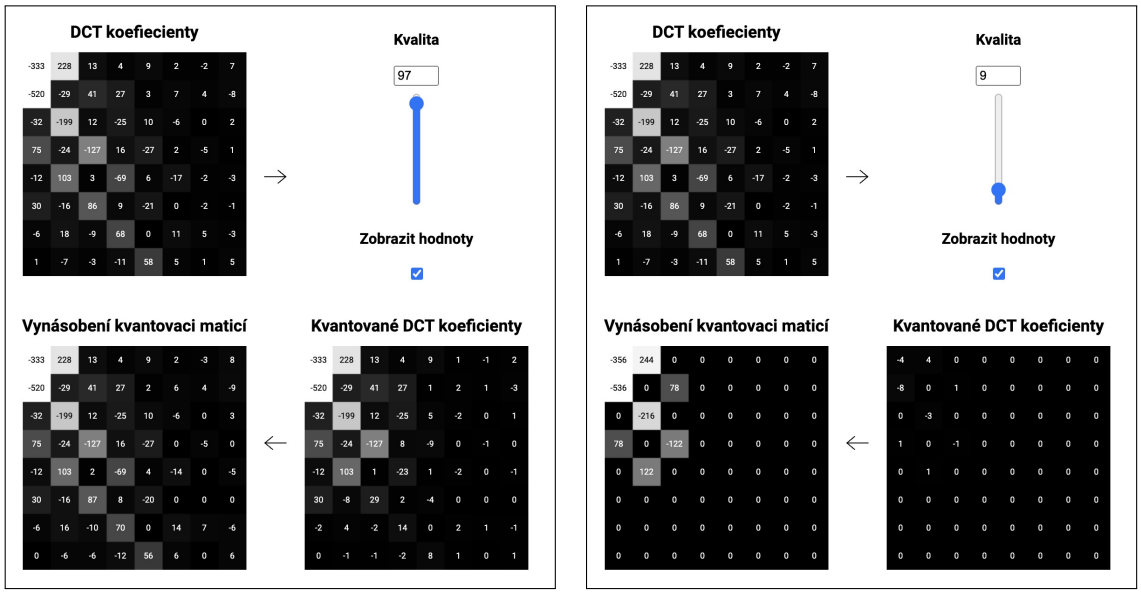
Obr. 2.5: Ukázka appletu „Ilustrace efektu změny DCT koeficientů na vzhled obrazu“.

V pravé části okna appletu je pak znázorněn proces přepočtu hodnot pixelů do kvantovaných DCT koeficientů a zpět do výsledných hodnot pixelů. Pro větší přehlednost a pochopitelnost je pořadí každého kroku označeno šipkou, což umožňuje uživateli snadno sledovat, jak se jednotlivé kroky navzájem ovlivňují a jaký vliv mají na výsledný obrázek. Celkem je zde 5 čtvercových bloků, které reprezentují jednotlivé kroky procesu komprese a dekomprese, a blok, kde uživatel je schopen vybrat kvalitu obrazu v rozmezí 1 až 100 od nejhorší po nejlepší. Díky tomuto nastavení může uživatel snadno experimentovat s různými stupni komprese obrazu a sledovat, jak to ovlivnuje výsledný obrázek.

Prvním krokem je zobrazení zvoleného uživatelem bloku 8×8 pixelů. K tomu slouží funkce `getCroppedPixelsData()`, kde na základě zvolené oblasti se vybere pouze určitý seznam hodnot pixelů. Dále se tyto hodnoty dostávají do funkce `getDCTPixelsData()`, kde probíhá na začátku transformace RGB hodnot do YC_BC_R pomocí kódu, který již byl popsán v kapitole 2.3 Výstupní objekty pixelů v barevném modelu YC_BC_R se přenášejí do funkce `applyDCT()` (A.5), která byla implementována podle vzorce Diskrétní cosinove transformace (1.4).

V bloku Kvantované DCT koefficienty je použita funkce `getQuantizationPixelsData()`, na vstup které přecházejí získané v předchozím kroku DCT koefficienty a zvolená uživatelem kvalita obrazu. Dana poslopnost DCT

koeficientů ve funkci `applyQuantization()` (A.6) se rozdělí posloupností kvantovacich hodnot, která byla vygenerována na základě kvality obrazu pomocí funkce `generateQuantizationMatrix()`. Poté každá kvantována hodnota se zaokrouhlí do nejbližšího celého čísla. Závislost výsledných DCT koeficientů na zvolené kvantovaci matici je dobře vidět na obrázku 2.6.



a) b)

Obr. 2.6: Stav určitého bloku hodnot kvantovaných DCT koeficientů v závislosti na vybrané stupni komprese (kvality): a) DCT koeficienty při kvalitě 97, b)DCT koeficienty při kvalitě 9.

V posledních dvou bloků probíhá zpětný proces, kde pomocí funkce `getDequantizationPixelsData()` hodnoty z predhoziho bloku se zpět vynásobí kvantovaci posloupností a zobrazí se v plátnu „Vynásobení kvantovaci matici“. Nakonec ve funkci `getIDCTPixelsData()` proběhne zpětná diskrétní kosinová transformace realizována funkcí `applyIDCT()`.Tento postup se aplikuje na blok 8×8 hodnot buď pixelů nebo již koeficientů.

Pro kompresi celého obrazu, který se výsledně zobrazí v „Výsledném obrázku“, se používá funkce `getAllProcessingStepsPixelsData()`. Tato funkce obsahuje všechny zmíněné kroky komprese a dekomprese, které zpracovávají hodnoty blok po bloku a nakonec vygenerují celý zkomprimovaný obraz.

V daném appletu je také implementováno zobrazení hodnot pomocí tlačítka „Zobrazit hodnoty“. To umožňuje uživateli prostudovat každou fázi komprese obrazových dat, což může být užitečné pro pochopení procesu a jeho výsledků. Díky této funkci lze nahlédnout do toho, jakymi změnami procházejí hodnoty pixelů – jak DCT ukládá většinu signálu do levého horního rohu bloku a jak poté kvantování

tvoří velký počet nul, které se snadno zakódují.

2.5 Applet „Vizualizace obrazů po dekompresi“

Tato webová aplikace se zabývá problematikou hlavního ztrátového procesu v algoritmu JPEG - kvantování DCT koeficientů. Jak již bylo zmíněno v teoretické části této bakalářské práce, většina nepodstatných obrazových dat se nevratně ztrácí zaokrouhlováním.

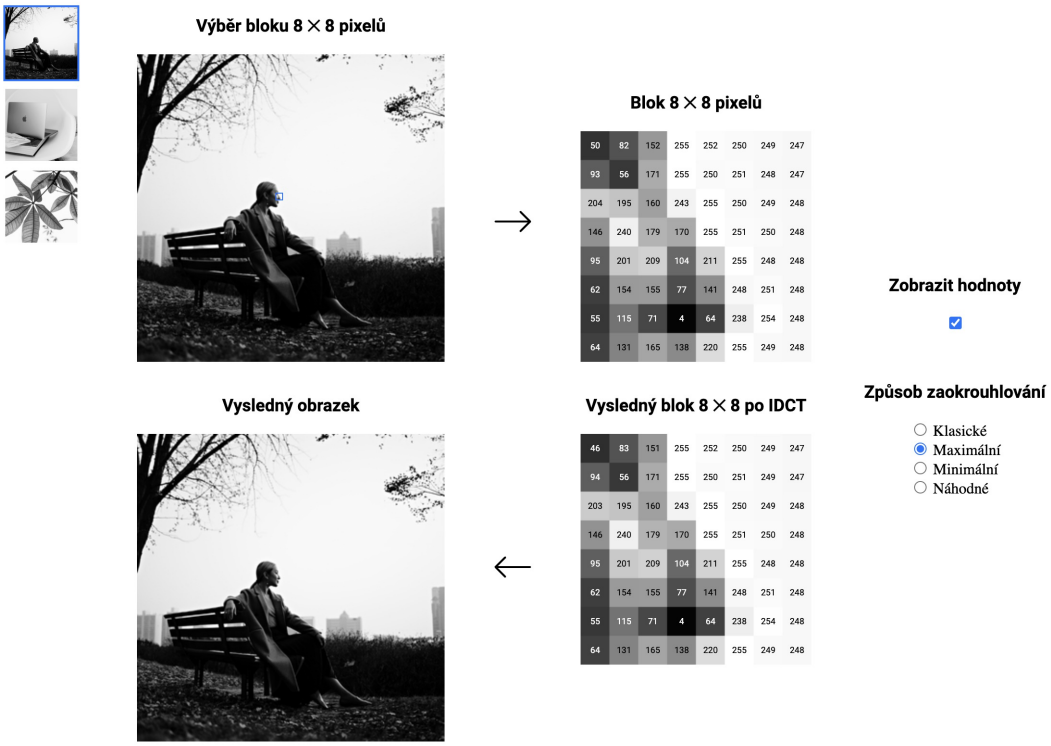
Z teorie je známo, že výsledkem aritmetického zaokrouhlení je nejbližší celé číslo k tomu, co bylo zaokrouhleno. Například $a = 7,4$ a $b = 6,5$. Po aplikaci funkce `round()` jsou oba tato čísla zaokrouhlena na 7. Stejně jakékoliv číslo v rozmezí $[6,5; 7,5)$ bude funkcí `round()` zaokrouhleno na 7.

Na základě této znalosti applet má za cíl ukázat, jak by mohly vypadat obrázky, které by po zaokrouhlování měly stejné hodnoty. Pro rekonstrukci takových obrázků byl navržen následující postup. Na začátku se získané z hodnot pixelů DCT koeficienty zaoukrohli do nejbližšího celého čísla. K zaokrouhleným koeficientům se přičte $+0,499$ nebo $-0,499$, anebo v případě náhodného módu jakékoliv číslo z intervalu $[-0,499; 0,499]$. Tyto nové DCT koeficienty se dekomprimují zpet do hodnot pixelů a zobrazí se v blocích „Výběr bloku 8×8 pixelů“, „Blok 8×8 pixelů“. Nakonec rekonstruované hodnoty pixelů projdou DCT a aritmetickým zaokrouhlováním a zobrazí se ve dvou spodních blocích.

Aplikace vypadá podobně jako dvě ostatní, které byli popsané předtím. Tento applet se skada ze dvou bloků s obrázky - vstupní a výstupní, a ze dvou bloků 8×8 pixelů - vstupní, který byl vybrán uživatelem v sekci „Výběr bloku 8×8 pixelů“, a stejný blok pixelů, ale po dekomprese 2.7. V části „Způsob zaokrouhlování“ uživatel je schopen vybrat z čtyř možností tzv. „zaokrouhlení“:

- „klasické“ – v něm probíhá klasické aritmetické zaokrouhlení pomocí `Math.round()`. Je výchozím nastavením.
- „maximální“ – k zaokrouhleným DCT koeficientům se přičte $0,499$. Pak tyto hodnoty projdou rekonstrukcí do vstupního obrázku a znovu se zkomprimují a dekomprimují.
- „minimální“ – z zaokrouhlených DCT koeficientů se odečte $0,499$. Pak proběhne stejný proces jak v předchozím módu (maximální).
- „náhodné“ – k zaokrouhlenému DCT koeficientů se přičte náhodná hodnota z intervalu $[-0,499; 0,499]$. Pak proběhne stejný proces jak při výběru módu „maximalni“.

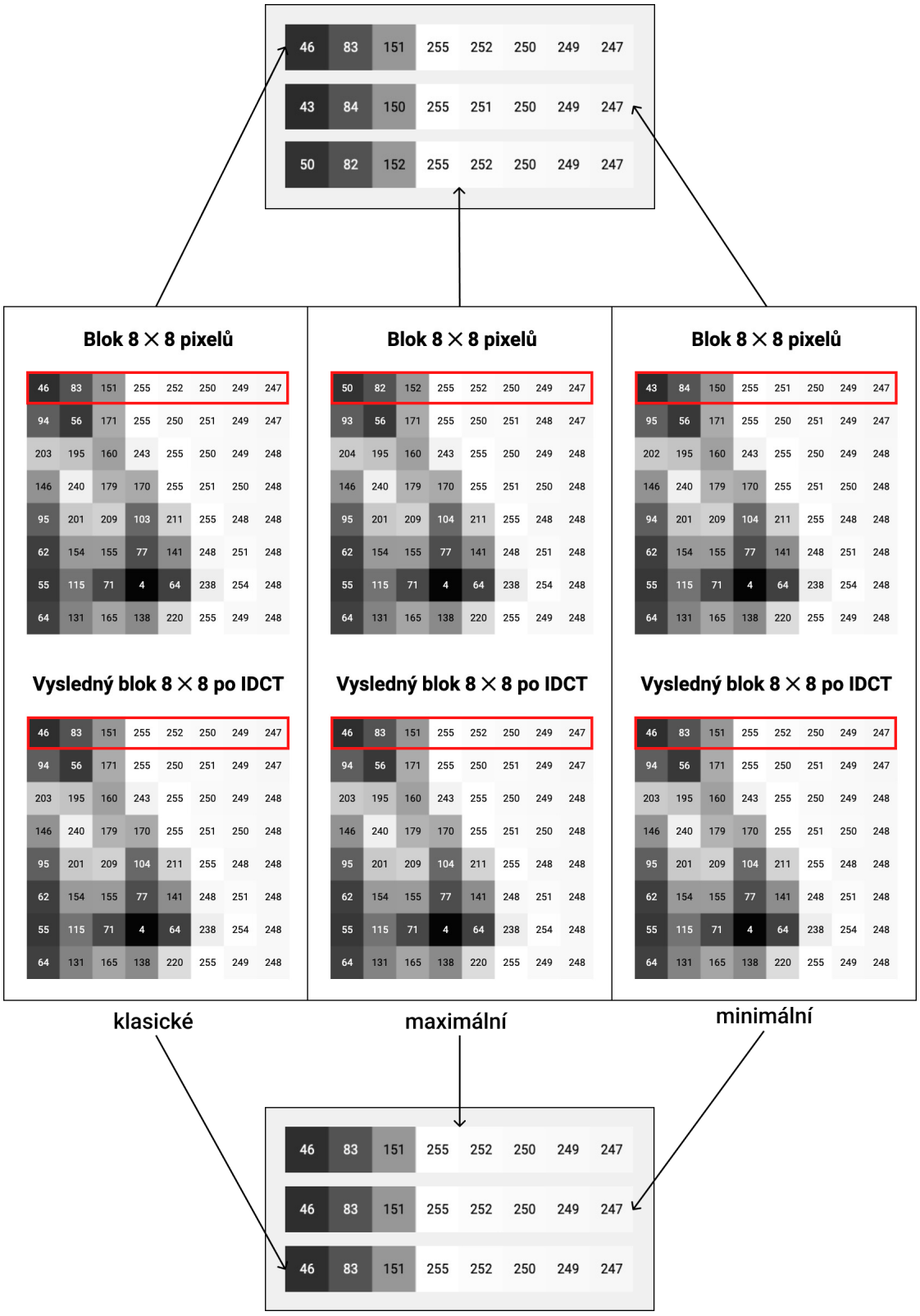
Při zvolení jakéhokoliv z módů zaokrouhlení v blocích „Vysledný obrazek“ a „Vysledný blok 8×8 po IDCT“ se nebude nic měnit, protože jak původní, tak i vytvořené na jeho základě obrázky mají takové hodnoty pixelů, které ve výsledku



Obr. 2.7: Ukázka appletu „Vizualizace obrazů po dekompresi“.

se zaokrouhlí na stejná čísla, což je dobře vidět na obrázku 2.8, kde v blocích „Blok 8×8 pixelů“ jsou zobrazeny hodnoty původního obrazku (klasické) a hodnoty přepočítaných obrazků (maximální a minimální). V blocích „Výsledný blok 8×8 po IDCT“ jsou zobrazeny hodnoty z bloků „Blok 8×8 pixelů“ po provedení komprese. Pro lepší přehlednost hodnoty prvního řádku, které jsou zvýrazněny červeně, byly umístěny do obdélníku nahoře a dole. Je tady vidět, že různé vstupní hodnoty se zaokrouhlují na stejná čísla. V této fázi aplikace funguje pouze v kvalitě 100 (bez dělení a násobení kvantovací maticí).

Fungování celého appletu je postaveno na funkci `getRecalculatedPixelsData()`, která má za účel vytvořit objekt `PixelsData` s rekonstruovanými hodnotami původního obrázku pro další výpočty. Zvolený způsob zaokrouhlení se aplikuje na seznam hodnot pomocí funkce `applySpecificRoundingMode()`. Následně funkce `getIDCTRecalculatedPixelsData()` a `getResultRecalculatedPixelsData()` na základě těchto rekonstruovaných hodnot původního obrazku vrátí objekty `PixelsData` pro zobrazení výsledků.



Obr. 2.8: Porovnání vstupních a výstupních hodnot při výběru různých typů zaokrouhlení – klasického, maximálního a minimálního.

Závěr

Daná bakalářská práce se věnuje problematice ztrátové komprese obrazových dat realizované algoritmem JPEG. V první části práce byla prozkoumána teorie týkající se objektivních a subjektivních charakteristik barvy a jejího vnímání lidským okem. Popisují se dva různé barevné modely (RGB a YC_BC_R), jejich rozdíly a oboustranný převod jednotlivých složek pixelů. Podrobně jsou popsány jednotlivé kroky komprese a dekomprese algoritmu JPEG.

Cílem této bakalářské práce bylo vytvoření tří appletů. První z nich názorně demonstruje průběh transformace barev a závislost vzhledu obrazu na zvoleném schématu podvzorkování, druhý ukazuje, co se děje s hodnotami pixelů během DCT a jaký vliv má volba určité kvantovací matice na vzhled dekomprimovaného obrazu, a třetí prezentuje následky zaokrouhlování kvantovaných DCT koeficientů, které je hlavním ztrátovým procesem v rámci JPEG. Vytvořené webové aplikace jsou interaktivní, což znamená, že uživatel je schopen nějakým způsobem ovlivnit chod jednotlivých kroků, které se zobrazují na obrazovce. Tato vlastnost také přispívá k lepšímu pochopení probíraného materiálu.

Literatura

[1] *Color* [online]. poslední aktualizace 9. 12. 2021 14:42. [cit. 9. 12. 2021]. Wikipedie. Dostupné z URL: <https://en.wikipedia.org/wiki/Color>

[2] *Zelené a UV laserys*online. poslední aktualizace 23. 3. 2012. [cit. 9. 12. 2021]. Dostupné z URL: <https://www.mega-blog.cz/lasery/zelene-a-uv-lasery/>

[3] GONZALEZ, Rafael C.; WOODS, Richard E. *Digital Image Processing.* 3. vyd. Upper Saddle River: Pearson Education, 2008, ISBN 978-0-13-168728-8.

[4] *The Basic Properties of Color* [online]. 2016 [cit. 23. 4. 2023]. Dostupné z URL: <https://www.premiumbeat.com/blog/basic-properties-color/>

[5] NOOR A. Ibraheem, MOKHTAR M. Hasan, RAFIQUL Z. Khan, PRAMOD K. Mishra. *Understanding Color Models: A Review* [online]. ARPN Journal of science and technology, 2012. ISSN 2225-7217. Dostupné z URL: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.679.8051&rep=rep1&type=pdf>

[6] *RGB color model* [online]. poslední aktualizace 29. 11. 2021 05:26. [cit. 2. 12. 2021]. Wikipedie. Dostupné z URL: <https://en.wikipedia.org/wiki/RGB_color_model>

[7] ČAPEK, Jan a FABIÁN, Peter. *Komprimace dat: principy a praxe.* Praha: Computer Press, 2000. Internet. ISBN 80-7226-231-9.

[8] *YCbCr* [online]. poslední aktualizace 5. 10. 2021 14:53. [cit. 2. 12. 2021]. Wikipedie. Dostupné z URL: <https://en.wikipedia.org/wiki/YCbCr>

[9] POKORNY, P. *Lossy Compression in the Chroma Subsampling Process* [online]. Zlín: Tomas Bata University in Zlín, 2016. Dostupné z URL: <https://www.wseas.org/multimedia/journals/computers/2016/a105805-094.pdf>

[10] G.K. Wallace. *The JPEG still picture compression standard.* IEEE Transactions on Consumer Electronics, Vol. 38, No. 1, FEBRUARY 1992 ISSN 0098-3063.

[11] *ISO/IEC 10918-1:1994* [online]. [cit. 3. 5. 2023]. Dostupné z URL: <https://www.iso.org/standard/18902.html>

[12] BURGER Wilhelm; BURGE Mark J. *Digital Image Processing.* Springer Nature Switzerland AG, 2022. ISBN 978-3-031-05744-1

[13] *AIC - Color Conversion* [online]. [cit. 3. 12. 2021]. Dostupné z URL: <http://www.bilsen.com/aic/colorconversion.shtml>

[14] ŽÁRA Jiří, Bedřich BENEŠ a Petr FELKEL. *Moderní počítačová grafika.* Praha: Computer Press, 1998. ISBN 80-7226-049-9.

[15] RAJMIC, P.; SCHIMMEL, J. *Moderní počítačová grafika.* Brno: Vysoké učení technické v Brně, 2013. ISBN: 978-80-214-4906-0.

[16] *The JPEG standar (ISO/IEC 10918-1)* [online]. 2014 [cit. 4. 5. 2023]. Dostupné z URL: <https://w3.ual.es/~vruiz/Docencia/Apuntes/Coding/Image/03-JPEG/index.html>

[17] LI, Ze-Nian; Mark S. DREW; Jiangchuan LIU. *Fundamentals of Multimedia.* Springer Cham Heidelberg New York Dordrecht London, 2014. ISBN : 978-3-319-05289-2.

[18] CABEEN, K.; GENT, P. *Image Compression and the Discrete Cosine Transform* [online]. [cit. 5. 3. 2023]. Dostupné z URL: <https://www.math.cuhk.edu.hk/~lmlui/dct.pdf>

[19] *Programujeme JPEG: Huffmanovo kódování kvantovaných DCT složek* [online]. 2007 [cit. 12. 4. 2023]. Dostupné z URL: <https://www.root.cz/clanky/programujeme-jpeg-huffmanovo-kodovani-kvantovanych-dct-slozek/>

[20] *What is JavaScript?* [online]. poslední aktualizace 10. 5. 2023. [cit. 11. 5. 2023]. Dostupné z URL: <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript>

[21] *canvas.magmofier.js* [online]. poslední aktualizace 12. 4. 2016 [cit. 7. 12. 2021]. Dostupné z URL: <https://github.com/jy1989/canvas.magnifier.js>

Seznam symbolů a zkratek

JPEG	Joint Photographic Experts Group
RGB	červená Zelená Modrá - Red Green Blue
DCT	Diskrétní kosinová transformace - Discrete Cosine Transform
IDCT	Inverzní diskrétní kosinová transformace - Inverse Discrete Cosine Transform
DPCM	Differential Pulse Code Modulation
RLE	Run-length encoding
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets

Seznam příloh

A Ukázka některých částí kódu

A.1 Funkce convertRGBtoYCC()

Výpis A.1: Ukázka implementace transformace barev.

```
function convertRGBtoYCC(pixel) {
    // zápis jednotlivých komponent pixelu do proměnných
    const r = pixel.r;
    const g = pixel.g;
    const b = pixel.b;
    // přepočet hodnot RGB na YCC
    const y = 0.299 * r + 0.587 * g + 0.114 * b;
    const cb = -0.168736 * r - 0.331264 * g + 0.5 * b + 128;
    const cr = 0.5 * r - 0.418688 * g - 0.081312 * b + 128;
    // vrácení nového objektu
    return { y, cb, cr }
```

A.2 Funkce getChunks()

Výpis A.2: Ukázka implementace rozdělení obrázku na bloky.

```
function getChunks(array, length) {
    // postupně prochází všechny pixely v seznamu
   return array.reduce(function (result, value, index, array){
        // pokud jsou index pixelu
        // a délka vybraného bloku stejné
        if (index % length === 0)
            // do výsledného seznamu se přidá nový blok
            result.push(array.slice(index, index + length));
        return result;
    }, []);
}
```

A.3 Funkce getSquares()

Výpis A.3: Ukázka implementace rozdělení obrázků na bloky čtverců.

```
function getSquares(array, squareSize, imageWidth) {
    // vytvoření seznamu řádků pixelů
    const rows = getChunks(array, imageWidth);
    let row = 0;
    // postupně prochází každý řádek
    const result2D = rows.reduce((result, rowEl, rowIndex) =>{
        // pokud jsou index řádku a velikost čtverce stejné
        if (rowIndex && rowIndex % squareSize === 0)
            // vezmeme další řádek
            row++;
        let col = 0;
        // postupně prochází pixely v řádku
        rowEl.forEach((colEl, colIndex) => {
            // pokud jsou index pixelu v řádku
            // a velikost ctverce stejné
            if (colIndex && colIndex % squareSize === 0)
                // vezmeme další pixel
                col++;
            // vytvoření prostoru pro přidávání pixelů
            if (!result[row]) result[row] = [];
            if (!result[row][col]) result[row][col] = [];
            // přidání pixelu do bloku
            result[row][col].push(colEl)
        })
        return result;
    }, []);
    // vracíme seznam bloků v správném formátu
    return result2D.flat()
```

A.4 Funkce getAverageOfComponent()

Výpis A.4: Ukázka implementace samotného procesu podvzorkování.

```
function getAverageOfComponent(pixels, component) {
    // vytvoření seznamu hodnot pixelu
    const componentValues = pixels.map(x => x[component])
    // výpočet součtu
    const sum = componentValues.reduce((a, b) => a + b, 0);
    // vrácí se výsledek dělení součtu délkou bloku
    return sum / pixels.length
}
```

A.5 Funkce applyDCT()

Výpis A.5: Ukázka implementace vzorce diskrétní kosinové transformace.

```
function applyDCT (block) {
  // vytvoření výsledného pole
  const result = new Array(64);
  // iterace přes všechny hodnoty výsledného pole
  for (let i = 0; i < 8; i++) {
    for (let j = 0; j < 8; j++) {
      let sum = 0;
      for (let x = 0; x < 8; x++) {
        for (let y = 0; y < 8; y++) {
          // výpočet sumy podle vzorce DCT
          sum +=
            block[x][y] *
            Math.cos(((2 * x + 1) * i * Math.PI) / 16) *
            Math.cos(((2 * y + 1) * j * Math.PI) / 16);
        }
      }
      // výpočet hodnoty výsledného pole
      sum *=
        ((i === 0 ? 1 / Math.sqrt(2) : 1) * (j === 0 ? 1 / Math.
           sqrt(2) : 1)) /
        4;
      // uložení hodnoty do výsledného pole
      result[i * 8 + j] = sum;
    }
  }
  return result;
};
```

A.6 Funkce applyQuantization()

Výpis A.6: Ukázka implementace procesu kvantování.

```
function applyQuantization (block, quality) {
  // generace kvantizační matice
  const matrix = generateQuantizationMatrix(quality);
  // vytvoření výsledného pole
  const result = new Array(64);
  // iterace přes všechny hodnoty výsledného pole
  for (let i = 0; i < 64; i++) {
    // výpočet hodnoty výsledného pole
    result[i] = block[i] / matrix[i];
  }
  return result;
}
```

B Obsah elektronické přílohy

Přiložený ZIP soubor obsahuje soubory formátu .js, .html a .css potřebné pro spuštění appletu. je zde také umístěn obrázek, se krerym pracuje applet. Pro spuštění je potřeba daný applet umístit na web. Applet byl testován v prohlížeči Google Chrome verze 113.0.5672.93.

- `/` — kořenový adresář přiloženého archivu
  - `assets` — složka obsahující zdrojové obrázky
    - `1.png`
    - `2.png`
    - `3.png`
    - `bw_1.png`
    - `bw_2.png`
    - `bw_3.png`
  - `scripts` — složka obsahující JavaScript soubory projektu
    - `canvas.js` — definice třídy `Canvas`
    - `constants.js` — seznam výchozích sdílených hodnot projektu
    - `dct.js` — vnitřní logika appletu „DCT“
    - `functions.js` — definice matematických funkcí projektu
    - `magnifier.js` — doplňkový script pro podporu lupy
    - `subsampling.js` — vnitřní logika appletu „Podvzorkovaní“
    - `utils.js` — definice pomocných funkcí projektu
    - `visualization.js` — vnitřní logika appletu „Vizualizace“
  - `styles` — složka obsahující kaskádové styly jednotlivých appletů
    - `dct.css`
    - `shared.css` — seznam sdílených kaskádových stylů
    - `subsampling.css`
    - `visualization.css`
  - `templates` — složka obsahující soubory HTML jednotlivých appletů
    - `dct.html`
    - `subsampling.html`
    - `visualization.html`