VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

KÓDOVÁNÍ KOEFICIENTŮ DISKRÉTNÍ VLNKOVÉ TRANSFORMACE

DISCRETE WAVELET TRANSFORM CODING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JIŘÍ ANTOŠ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. DAVID BAŘINA, Ph.D.**
SUPERVISOR

BRNO 2017

Abstrakt

Cílem práce je popsat a porovnat způsoby kódování obrazových dat získané pomocí diskrétní vlnkové transformace, který využívá standard JPEG 2000. Zejména se zabývá algoritmy kódování koeficientů z této transformace a srovnání několika postupů jejich komprese, jako je EBCOT, SPIHT, nebo EZBC. Jedním z východisek je i navrhnutí modifikací těchto algoritmů, pro lepší parametry komprese.

Abstract

Target of this thesis is to describe and compare encodings that take discrete wavelet transformed image as input, as it is in JPEG 2000 standard. Its main concern is description of such encoding algorithms as well as comparison between some of them, namely EBCOT, SPIHT and EZBC. One of possible improvements of this work is implementing more effective modification of said algrotihms.

Klíčová slova

DWT, vlnka, vlnková transformace, EZBC, SPIHT, EBCOT, komprese JPEG 2000, CDF

Keywords

DWT, wavelet, wavelet transform, EZBC, SPIHT, EBCOT, JPEG 2000 compression, CDF

Citace

ANTOŠ, Jiří. *Kódování koeficientů diskrétní vlnkové transformace*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Bařina David.

Kódování koeficientů diskrétní vlnkové transformace

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Davida Bařiny, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jiří Antoš
18. května 2017

Poděkování

Děkuji vedoucímu práce panu Ing. Davidu Bařinovi, Ph.D. za odbornou pomoc, kterou mi poskytl při vypracování této práce.

Obsah

Kapitola 1

Úvod

V dnešní době je naprosto běžné uchovávat si vzpomínky na důležité události, které člověk prožije. Vzpomínky představují informace, které lze zaznamenat v mnoha podobách, od psaného textu, přes mluvené slovo až k fotografiím. Jako prostředek, který dokáže uchovat nejvíce informací na malém prostoru, se nejčastěji používá právě zachycení okamžiků do obrazové formy, ve které lze uchovat to, co náš zrak vidí po poměrně dlouhou dobu. Fotografie lze pořizovat více způsoby, v minulosti například na film jako analogový signál. Doba se však změnila, klasické filmy nahradily digitální formáty pro uchování snímaného obrazu.

Jelikož nekomprimovaná data zabírají mnoho paměti, časem dosáhl velké popularity komprimovaný formát JPEG, který dokáže zmenšit velikost nutné paměti při na první pohled nezměněné kvalitě. Jednou z jeho nevýhod, obzvlášť při velké kompresi výsledného obrazu, je vznik blokových artefaktů, díky rozdělení obrazu na menší bloky. Jedním z řešení je používat při velké kompresi obrazových dat nový formát JPEG 2000, který si s těmito artefakty umí poradit lépe. Využívá diskrétní vlnkovou transformaci místo původní diskrétní kosinové transformace a efektivnější algoritmy pro uchování obrazových dat.

Tato práce se zabývá algoritmy pro ztrátovou kompresi, klade si za cíl srovnání vybraných zástupců – SPIHT (*Set Partitioning In Hierarchical Trees*), EBCOT (*Embedded Block Coding with Optimized Truncation*) a EZBC (*Embedded Zero Block Coding*). Předmětem porovnání je mimo jiné účinnost a rychlost kódování.

Nejprve je nutné uvést teorii týkající se převodu do formátu JPEG 2000, totiž diskrétní vlnkovou transformaci a související pojmy, jako jsou vlnky, jejich typy, obecná vlnková transformace a jakým způsobem se aplikují tyto poznatky na uchování reálných dat. Toto je rozebráno v teoretické části práce spolu s postupem převodu rastrového obrazu v RGB modelu do formátu odpovídajícím standardu JPEG 2000, který umožňuje efektivnější uložení dat.

V následující části jsou rozděleny do rodin a poté podrobně popsány některé kódovací algoritmy, zejména EBCOT, SPIHT a EZBC, které byly vybrány pro implementaci, dále pak MRWD, SLCCA, EZW a zmíněni jsou někteří další v současné době používaní zástupci. Na konci této kapitoly je umístěn popis entropického kódování, speciálně pak jeho varianta aritmetické kódování.

Práce pokračuje praktickou částí složenou ze dvou kapitol - implementace a testování. Obsahem první z nich je způsob návrhu aplikací pro testování a představení způsobu fungování programů realizujících EBCOT, SPIHT a EZBC. V druhé jsou pak diskutovány výsledky testování provedeného nad těmito programy na testovacích obrázcích. Testy byly zaměřeny na zkoumání vlastností algoritmů z hlediska kvality a komprese výsledných dat. Jako srovnávací metriky kvality obrazu jsou použity objektivní přístupy pomocí SSIM (*Structure*

Similarity Index) a PSNR (*Peak Signal to Noise Ratio*), jejichž popis je v této kapitole také uveden. Je také provedeno srovnání kompresních poměrů u jednotlivých algoritmů při zachování podobné kvality obrazu, podobné výsledné velikosti toku a doby, která uplyne od začátku do konce komprese.

Poslední kapitolu tvoří závěr, který shrnuje dosažené poznatky na základě výsledků provedených testů a uvádí možnosti pokračování této práce, například doplnění dalších algoritmů nebo vytvoření nového postupu kódování koeficientů.

Kapitola 2

Teoretická část práce

V této části práce jsou popsány fáze postupu převodu obrazových dat do JPEG 2000 bitového toku. Zmíním se také o základní teorii o diskrétní vlnkové transformaci, která je součástí komprese JPEG 2000. Výsledkem této transformace jsou koeficienty, které mohou být zakódovány algoritmy představenými v následující kapitole.

2.1 Komprese obrazu JPEG 2000

Předpokládejme, že jsou obrazová data umístěna ve 2D poli pixelů, které odpovídá matici o M řádcích a N sloupcích, přičemž $M \times N$ udává rozlišení obrázku. Enkodér JPEG 2000 provádí tyto kroky [1]:

1. Transformace barevného prostoru obrazových dat
   - do $\mathrm{YC_bC_r}$ barevného prostoru v případě nereverzibilní transformace
   - do modifikovaného YUV barevného prostoru při reverzibilní transformaci
2. DC level shifting – jedná se o vycentrování hodnot reprezentujících barvy odečtením určité hodnoty od všech pixelů
3. Rozdělení obrazu na jednu nebo více dlaždic (tiling) – obrázek se rozdělí bloky pixelů o stejné velikosti, u pravého a spodního kraje může dojít ke prodloužení obrázku, aby dlaždice pokrývaly celou plochu obrázku
4. Následují jedno- nebo víceúrovňové transformace jednotlivých dlaždic, vznikne několik bloků koeficientů
5. Dále se jednotlivé koeficienty kvantifikují v rámci bloku – zde může docházet ke snížení počtu bitů na pixel a ztrátě kvality
6. A poté se zakódují pomocí kódovacího algoritmu, často se používá algoritmus EBCOT.

2.1.1 Barevný prostor YC_bC_r

Barevný model YC_bC_r využívá vlastnosti lidského oka, které je více citlivé na jas a mnohem méně vnímá chrominanci. Tyto složky jsou od sebe odděleny, výsledkem jsou dvě chrominanční složky s menším počtem uchovávaných informací a jasová složka, která je pro lidský zrak důležitější. Lze tak dosáhnout lepších kompresních poměrů než u prostorů RGB a mYUV. [25]. Jednotlivé složky představují:

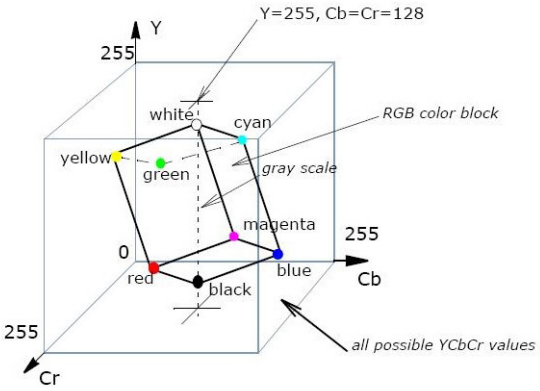
- Y – jasová složka,
- C_b, C_r – modrá a červená chrominanční složka.

Konverze 24-bitového RGB prostoru do 24-bitového YC_bC_r prostoru lze uskutečnit pomocí následujících vztahů (při 8 bitech pro jednotlivé kanály) [24]:

$$Y = 0,299.R + 0,587.G + 0,114.B \tag{2.1}$$

$$C_b = 128 - 0,168736.R - 0,331264.G + 0,5.B \tag{2.2}$$

$$128 + 0,5.R - 0,418688.G - 0,081312.B \tag{2.3}$$

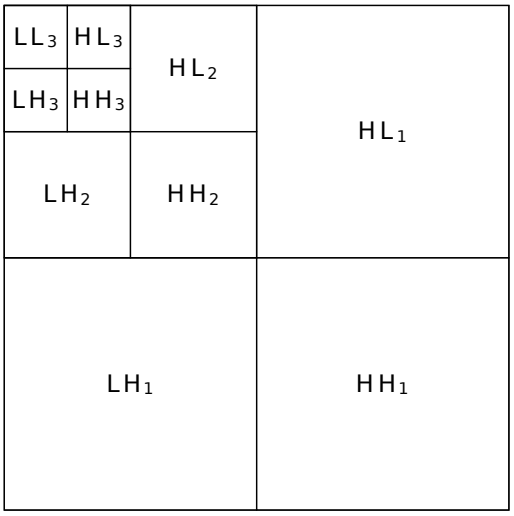


Obrázek 2.1: Vztah barevných prostorů YC_bC_r a RGB

Zdroj: [12]

2.1.2 Transformace obrazu

V rámci jedné úrovně dyadické diskrétní vlnkové transformace, která se provádí aplikací příslušných obrazových filtrů, vzniknou z 2D pole pixelů 4 subpásma koeficientů – subpásma HH, HL, LH a LL, každé má velikost $\frac{M}{2} \times \frac{N}{2}$. HL subpásmo obsahuje detaily obrazu ve vertikálním směru, LH detaily obrazu v horizontálním směru, HH detaily obrazu v diagonálním směru a subpásmo LL obsahuje podvzorkovanou verzi původního obrazu . Nad LL subpásmem lze opět provést dyadickou vlnkovou transformaci, tímto způsobem lze docílit dvou- a víceúrovňové transformace bloku původních obrazových dat. Příklad je na uveden na obrázku 2.2. [2].



Obrázek 2.2: Subpásma po tříúrovňové dekompozici obrazu

Zdroj: [21]

2.2 Vlnková transformace

V následujících odstavcích je popsána definice a vlastnosti vlnky, její vhodné vlastnosti pro kompresi a jednotlivé typy vlnek. Z kategorií vlnkových transformací je dále rozebrána spojitá vlnková transformace. Informace byly převzaty z [2] a [11].

2.2.1 Vlnka – Wavelet

Vlnky (z anglického *wavelet*) nazýváme funkce, které vycházejí z jedné základní vlnky, pojmenované matečná funkce, nebo také prototypová funkce. Obvykle se tato mateřská vlnka označuje $\psi(t)$, jde o funkci nad reálnými čísly v L^2 prostoru , která musí splňovat následující dvě vlastnosti:

1. Energie E vlnky musí být konečná

$$\int_{-\infty}^{+\infty} |\psi(t)|^2 \, \mathrm{d}t < \infty \tag{2.4}$$

2. Vlnka má nulovou průměrnou hodnotu

$$\int_{-\infty}^{+\infty} |\psi(t)| \, \mathrm{d}t = 0 \tag{2.5}$$

Z této funkce lze modifikací získat odvozené vlnky, pomocí translace (posunu) a dilatace (roztažení, stažení). Výsledný tvar vlnky lze vyjádřit takto:

$$\psi(t) = \psi(t)_{a,b} = \frac{1}{\sqrt{a}} \psi \left(\frac{t-b}{a} \right) \tag{2.6}$$

ψ – matečná funkce

b – parametr vyjadřující translaci – translační proměnná

a – parametr vyjadřující dilataci – dilatační proměnná

2.2.2 Vlnková transformace

Na rozdíl od Fourierovy transformace, která je schopná poměrně přesně analyzovat signál pouze v rámci frekvencí, vlnková transformace umožňuje provést přiměřeně přesnou jak časovou, tak frekvenční analýzu signálu. Existuje více různých vlnkových transformací, které se odlišují zvoleným typem mateřské vlnky. Pomocí mateřské vlnky lze vlnkovou transformaci matematicky vyjádřit následovně:

$$V_{(a,b)} = w(a) \int_{-\infty}^{+\infty} f(t)\psi^* \left(\frac{t-b}{a} \right) \mathrm{d}t \tag{2.7}$$

$f(t)\psi^*$ – komplexně sdružená vlnka k matečné funkci

$w(a)$ – váhová funkce, nejčastěji $\frac{1}{\sqrt{a}}$

$f(t)$ – analyzovaný signál

2.2.3 Typy vlnkové transformace

Na aplikaci vlnkové transformace na signál $f(t)$ lze nahlížet jako na konvoluci mezi signálem a vlnkami odvozenými od matečné funkce. Rozlišují se dvě základní kategorie vlnkových transformací – spojitá a diskrétní:

- Spojitá vlnková transformace (CWT) – parametry a, b jsou spojité proměnné, $f(t)$ je spojitý signál. Výsledkem těchto transformací je nekonečné množství koeficientů , proto se pro praktické využití příliš nehodí a nebudeme se jimi zabývat.
- Diskrétní vlnková transformace (DWT) – parametry a, b jsou diskrétní proměnné.

2.2.4 Diskrétní vlnková transformace

Spojitou vlnkovou transformaci lze převést na diskrétní diskretizací parametrů a a b:

$$F_{(DWT)}(m,n) = V_{(a,b)} = \frac{1}{\sqrt{a_0^m}} \int_{-\infty}^{+\infty} f(t)\psi^* \left(\frac{t}{a_0^m} - nb_0 \right) \mathrm{d}t \tag{2.8}$$

$$a = a_0^m, b = nb_0a_0^m$$

$$m, n \in \mathbb{Z}, a_0 > 1, b_0 > 0$$

Z důvodu efektivity výpočtu se nejčastěji používá dyadické vzorkování, kdy se hodnoty dilatační a translační proměnné volí jako mocniny 2, např. $a_0 = 2, b_0 = 1$. Potom lze dyadickou DWT vyjádřit následujícím způsobem:

$$F_{(DWT)}(m,n) = \frac{1}{\sqrt{2^m}} \int_{-\infty}^{+\infty} f(t)\psi^* \left(\frac{t}{2^m} - n \right) \mathrm{d}t \tag{2.9}$$

2.2.5 Mateřské vlnky a jejich vlastnosti pro kompresi

Mateřské vlnky lze rozdělit do několika rodin, které sdružují vlnky podobných vlastností. Některé z vlastností jsou podstatné pro vhodnost použití při DWT nebo CWT. Jde například o to, zda má vlnka kompaktní nosič, pokud ano, je její energie lokalizována na konečném časovém úseku. Na této schopnosti závisí efektivita výpočtu – při kratší vlnce s méně nenulovými koeficienty je počet operací pro konvolucí signálu s FIR filtrem nižší.

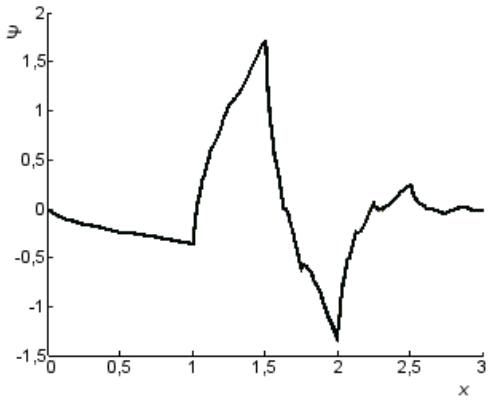
Pokud má vlnka velký počet nulových momentů, bude výsledkem vlnkové transformace velký počet nulových koeficientů, které lze pak efektivně zakódovat.

Další podstatnou vlastností je hladkost vlnky. Lze vypozorovat, že hladké vlnky dosahují lepších výsledků při ztrátové kompresi, než vlnky, které nejsou hladké. Toto je dáno tím, že při nízkých detailech obrazu se začínají v obrazu objevovat artefakty podobné použitým vlnkám, u hladkých vlnek však nejsou tolik výrazné.

Dále může být vlnka symetrická, taková vlnka lépe potlačuje nežádoucí efekt ringing při kompresi na hranách. Kromě toho symetrie zamezuje fázovému posunu mezi špičkou, singularitou, oscilací v signálu a příslušným projevem ve vlnkových koeficientech.

2.2.6 Vlnky Daubechies

Tyto vlnky představují rodinu vlnek různého řádu N, $N \geq 1$. Kromě řádu $n = 1$ jsou asymetrické a nemají explicitní vyjádření vlnky. Jsou spojité, konstruovány rekurzivně a označují se db$N$ podle řádu N. Jde o vlnky ortogonální, vhodné pro DWT i CWT, s N počtem nulových momentů. Na obrázku 2.3 je znázorněna vlnka Daubechies 2. řádu.



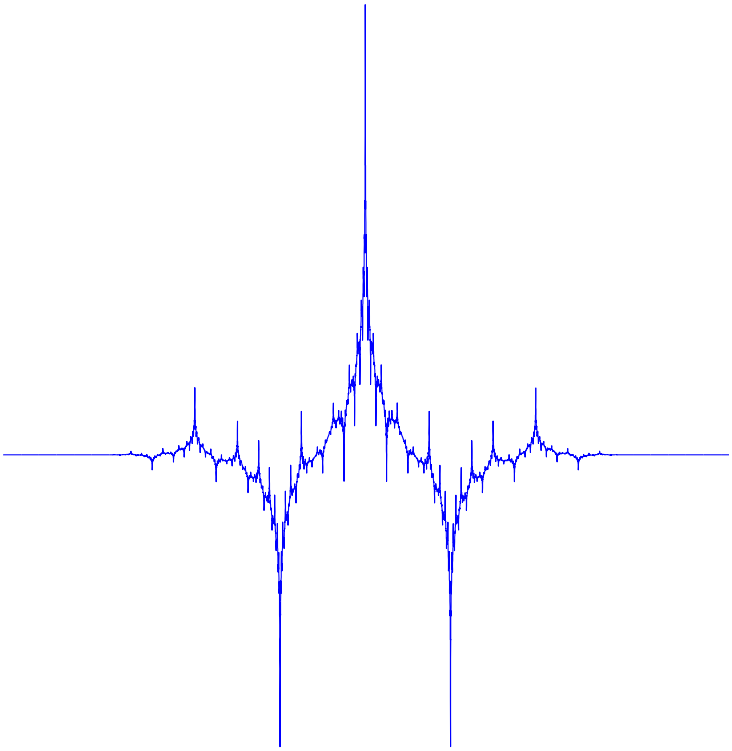
Obrázek 2.3: Průběh Daubechies vlnky 2. řádu v čase

Zdroj: [14]

2.2.7 Vlnky CDF

Vlnky typu biortogonálního splinu, někdy nazývané *Cohen-Daubechies-Feauveau*, mají velký počet nulových momentů a jsou symetrické. Mají kompaktní nosič, jsou vhodné jak pro CWT, tak i pro DWT, přestože nevytváří ortogonální bázi. Pro rozklad a rekonstrukci signálu se používají jiné filtry, označují se podle jejich velikosti, první se zmiňuje filtr pro

rozklad. Např. vlnka CDF 9/7, která se používá pro nereverzibilní transformaci obrazu ve standardu JPEG 2000. Tato vlnka se dá pojmenovat i pomocí hladkosti dolních filtrů, jako CDF 4.4. Druhou vlnkou požívanou při kompresi JPEG 2000 je Le-Gallova vlnka, CDF 5/3 nebo označována také CDF 2.2, která zabezpečuje reverzibilní transformaci. [6] [1].



Obrázek 2.4: Průběh vlnky CDF 5/3 v čase

Zdroj: [3]

2.2.8 Deslauriers-Dubuc 13/7

Jedná se o symetrickou vlnku s kompaktním nosičem, která však není ortogonální. Platí pro ni, že disponuje interpolačními vlastnostmi, někdy je označována jako *interpolet* místo *wavelet*. Má větší počet nulových momentů než vlnka Daubechies o stejném řádu, dokáže přesně reprezentovat polynomy stupně o 1 nižší, než je její vlastní řád. Je spojitá, a tedy derivovatelná. [4].

2.3 Kvantizace

Při kvantizaci se koeficienty z transformace vynásobí kvantizační konstantou q, která se nachází v intervalu $(0; 1\rangle$:

$$a_q(x, y) = a(x, y)q \tag{2.10}$$

Pro získání původních hodnot koeficientů lze použít tento vztah:

$$a(x, y) = \frac{a_q(x, y)}{q} \tag{2.11}$$

Cílem kvantizace je zanedbat některé obrazové informace a tím snížit počet bitů potřebných k zakódování. Např. při $q = 0.5$ všechny koeficienty stanou polovičními a k jejich uchování je potřeba o jeden bit méně.

Pokud se někteří z původně nenulových koeficientů sníží pod úroveň, která odděluje nulové kódované hodnoty od nenulových, dojde ke ztrátě informací a zvýšení poměru koeficientů s hodnotou 0. Tímto lze za cenu ztráty detailů dosáhnout efektivnější komprese.

Kapitola 3

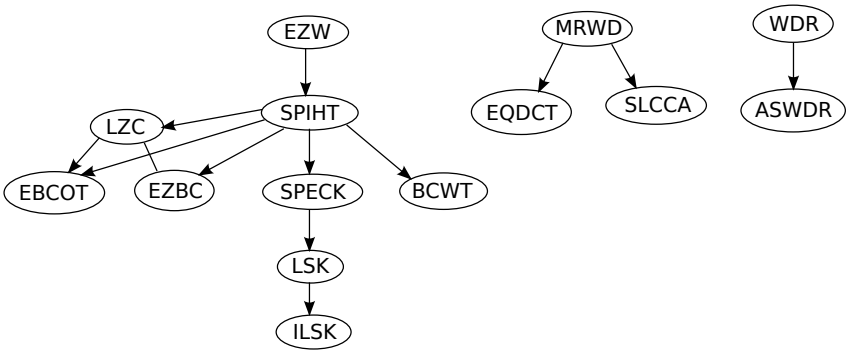
Kódování koeficientů

Podobně jako u DCT lze i na výstup DWT aplikovat kvantizaci, která však už nemusí být hlavním krokem ztrátové komprese. Místo toho je snížení velikosti výstupního bitového toku vynecháním nepodstatných obrazových informací ponecháno na kódování samotné, pokud toto umožňuje.

Existují různé principy, na kterých staví algoritmy kódování koeficientů DWT, jednotlivé algoritmy lze pak seskupit do rodin. Významnou vlastností koeficientů DWT je závislost mezi jednotlivými úrovněmi rozkladu, mezi rodičem z vyšší úrovně a 4^n potomky z úrovní nižších. Konkrétně jde o korelaci významnosti či nevýznamnosti koeficientů, kterou lze popsat pomocí stromových struktur zerotree a quadtree, které budou popsány dále. Mezi takové algoritmy patří EZW, SPIHT, LZC, BCWT, STW, SPECK, EZBC a EQDCT. SLCCA taktéž využívá závislost mezi subpásmy, ale nepoužívá stromové struktury.

Některé algoritmy zpracovávají koeficienty v pořadí daném jejich informační hodnotou - od nejvíce důležitých pro uchování obrazu po nejméně důležité. Takové algoritmy lze kdykoliv při kódování a dekódování přerušit, a výsledkem je obraz o určité kvalitě, která je v daném okamžiku požadována. Označují se embedded, typicky se vyznačují zpřesňovacím průchodem a postupným snižováním hodnoty pro posuzování významnosti koeficientů. Mezi zástupce patří WDR, ASWDR, BCWT, EZW, SPIHT, SPECK, EBCOT a EQDCT.

Mimo závislost mezi úrovněmi rozkladu lze využít při kompresi i závislost v rámci nejbližšího okolí koeficientu, jak je to u MRWD a SLCCA, které nepatří mezi embedded algoritmy. Seskupení koeficientů v blízkém okolí využívá i algoritmus SPECK a jeho odvozeniny LSK a ILSK, dále EBCOT a EQDCT.

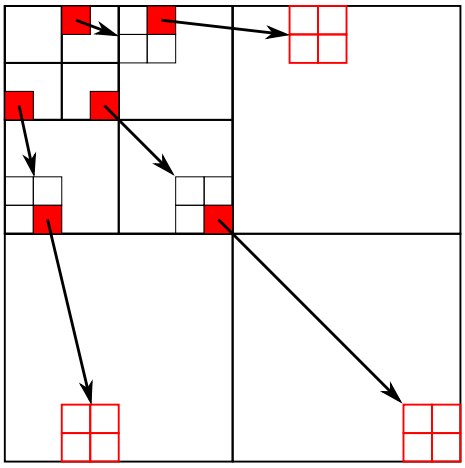


Obrázek 3.1: Strom vývoje algoritmů

3.1 EZW

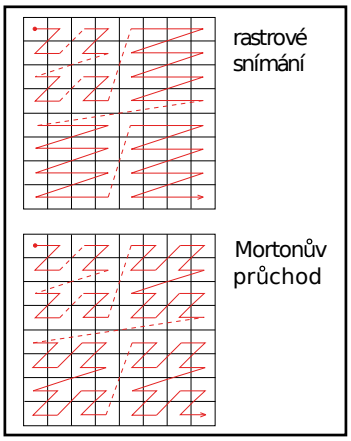
Algoritmus *Embedded Zerotrees of Wavelet transforms* (EZW) představuje jeden ze základních kompresních postupů kódování koeficientů z transformací. Jeho modifikací vznikly samostatné algoritmy, které dosahují kratší doby komprese, nebo větší efektivity, např. SPIHT. Patří do rodiny progresivních kódování, u které se na výstupu nejdříve objeví symboly reprezentující nejvíce významné koeficienty, s každým dalším krokem se kódují koeficienty nesoucí méně informací než předchozí. Lze tedy omezit velikost výsledného výstupu, nebo stanovit maximální počet průchodů pro dosažení určité kvality. Zbylé nezpracované koeficienty představují méně důležité obrazové informace.

Důležitým pojmem je stromová struktura *zerotree*, která je modifikací *quadtree*. Při víceúrovňové vlnkové transformaci dochází k vytváření závislostí koeficientů v podpásmech z jiných úrovní transformace, jak je zobrazeno na obrázku . Předpokládá se vztah mezi čtveřicí koeficientů v daném pásmu a jedním z koeficientů v podpásmu. Tato čtveřice je reprezentována ve stromu *quadtree* čtyřmi poduzly uzlu zastupujícího odpovídající koeficient ze subpásma. *Zerotree* je takový *quadtree*, jehož uzly jsou stejné nebo menší než kořen. [8].



Obrázek 3.2: Závislosti mezi koeficienty v podpásmech

Zdroj: [18], upraveno



Obrázek 3.3: Pořadí zpracovávaných koeficientů

Zdroj: [10]

Na začátku kódování se nalezne největší koeficient, podle kterého se nastaví hodnota prahu P_0. Následují iterace složené ze dvou částí - z dominantní (signifigance pass) a vedlejší (refinement pass). Podle použitého protokolu priorit (např. rastrové snímání, Mortonův průchod, viz obrázek 3.3) se určí pořadí koeficientů obrazu daného subpásma. Stejným způsobem se určí pořadí zpracování subpásem v dané úrovni rozkladu.

Jednou z prováděných akcí v obou průchodech je stanovení významnosti koeficientů porovnáním s hodnotou prahu P_0. Významné koeficienty mají vyšší absolutní hodnotu, než je hodnota prahu, nevýznamné koeficienty mají hodnoty nižší. Postupným snižováním prahu v následujících iteracích významné koeficienty nesou méně informací o obrazu než v předchozích krocích, což je typické pro rodinu progresivních kódování.

Dominantní průchod prochází koeficienty ve stanoveném pořadí, a na výstupu objeví jeden ze symbolů POS (*Signifigant Positive*), NEG (*Significant Negative*), ZTR (*Zero Tree Root*), IZ (*Isolated Zero*) podle hodnoty koeficientu a jeho potomků. Pokud je absolutní

hodnota koeficientu větší než práh P_0, pak se na výstup zapíše POS pro kladnou hodnotu, nebo NEG pro zápornou hodnotu. Pokud je absolutní hodnota menší než práh P_0, výstupem bude jeden ze symbolů ZTR, IZ. ZTR se označuje kořen zerotree, jehož všichni potomci jsou také nevýznamní, IZ značí, že aspoň jeden z následníků je významný a aktuální koeficient nevýznamným.

Dále se sníží hodnota prahu P_0 na $P_0^{'} = P\frac{3}{4}$ a provede se vedlejší průchod pouze nad významnými koeficienty z předchozí části. U těchto koeficientů se porovná jejich hodnota se současnou hodnotou prahu $P_0^{'} = P\frac{3}{4}$, pokud je hodnota koeficientu menší než práh, na výstupu se objeví 0, pokud větší než práh, objeví se 1.

Pro další iteraci se stanoví hodnota prahu $P_1 = \frac{P_0}{2}$ a opět se provedou dominantní a vedlejší průchod. Takto se pokračuje do přerušení kódování, nebo dokud se nedosáhne minimální hodnoty prahu P_{min}. Poté se výstup zakóduje vhodným aritmetickým kódováním. [18].

3.2 SPIHT

Modifikací algoritmu EZW byl vytvořen algoritmus SPIHT (*Set Partitioning in Hierarchical Trees*), a podobně jako EZW patří mezi progresivní kódování. Jednou z jeho specifických vlastností je velká zhuštěnost výstupu, který pak není nutné podrobovat aritmetickému kódování, jako u ostatních kódovacích algoritmů (EZW i EBCOT).

Stejně jako EZW jsou vstupem SPIHT koeficienty z několika úrovní transformace, mezi kterými je určitá závislost. Tento vztah je vyjádřen pomocí prostorových stromů (*spatial orientation trees*), v nichž každý uzel má buď čtyři přímé potomky, nebo je bez potomků. Přímé poduzly opět odpovídají jako u EZW čtyřem koeficientům z úrovně o 1 nižší, jak je znázorněno na obrázku 3.2.

Samotný algoritmus pracuje se čtyřmi množinami souřadnic (indexů) koeficientů:

- $\mathcal{O}_{(i,j)}$: množina souřadnic všech přímých potomků uzlu/koeficientu (i, j)
- $\mathcal{D}_{(i,j)}$: množina souřadnic všech potomků uzlu/koeficientu (i, j)
- $\mathcal{H}$: množina souřadnic všech kořenů prostorových stromů – koeficientů v nejvyšší úrovni
- $\mathcal{L}_{(i,j)}$: množina v všech nepřímých potomků uzlu/koeficientu (i, j)
  $\mathcal{L}_{(i,j)} = \mathcal{D}_{(i,j)} - \mathcal{O}_{(i,j)}$

Prostorovou závislost mezi koeficienty v subpásmech (kromě nejvyšší a nejnižší úrovně) lze zapsat matematicky následovně:

$$\mathcal{O}_{(i,j)} = (2i, 2j), (2i, 2j + 1), (2i + 1, 2j), (2i + 1, 2j + 1) \tag{3.1}$$

Podstatou kódování je rozdělování množin podle významnosti, cílem je získat významné koeficienty jako jednoprvkové množiny a nevýznamné koeficienty sdružit do množin s velkým počtem prvků, které se pak dají zakódovat několika málo bity. Dosažení tohoto stavu představuje konec kódovacího algoritmu. Množina je významná tehdy, je-li aspoň jeden její prvek významný, jinak je množina nevýznamná. Test významnosti koeficientu c spočívá v porovnání jeho absolutní hodnoty s daným prahem P, pokud platí $|c| > P$, pak je koeficient významný, jinak je nevýznamný.

V praxi se algoritmus realizuje nad třemi seznamy, které uchovávají pořadí množin koeficientů:

- LIS (List of Insignifigant Sets) – seznam nevýznamných množin
- LSP (List of Signifigant Points) – seznam významných bodů
- LIP (List of Insignifigant Points) – seznam nevýznamných bodů

Prvky seznamů LSP a LIP jsou souřadnice (i, j) koeficientů, seznam LIS obsahuje prvky dvojího typu – buď se jedná množiny $\mathcal{D}_{(i,j)}$, nebo o množiny $\mathcal{D}_{(i,j)}$. Tyto dva typy prvků lze odlišit značením, například A pro $\mathcal{D}_{(i,j)}$ a B pro $\mathcal{L}_{(i,j)}$.

Postupuje se iterativně, v každé iteraci se provádí dva průchody – řadící a upřesňovací. V řadícím průběhu se testuje významnost jednotlivých prvků v LIP (seznam dříve nevýznamných bodů/koeficientů), ty, které jsou významné, se přesunou do LSP. Obdobně se postupuje pro množiny, které jsou v LIS. Nevýznamné množiny obou typů zůstávají v seznamu LIS, dokud se snížením prahu nestanou významnými. Je-li množina A shledána významnou, rozdělí se na čtyři přímé potomky uzlu na daných souřadnicích a na množinu typu B, která obsahuje zbylé prvky z množiny A. Získaní přímí potomci se otestují, významní putují do seznamu LSP, nevýznamní do seznamu LIP. Pokud je významná množina B, rozdělí se na čtyři podmnožiny typu A, jejichž souřadnice odpovídají přímým potomkům koeficientu se souřadnicemi dané množiny B. [16].

Algoritmus 1 SPIHT

Inicializace

- $n = \lfloor \log_2(\max_{(k,i,j)}|c_k(i,j)|) \rfloor$

pošli n na výstup
LSP= $\emptyset$
for all $(i, j) \in \mathcal{H}$ **do**
  přidej (i, j) do LIP
  if $\mathcal{D}(i, j) \neq \emptyset$ **then**
    přidej (i, j) do LIS jako záznam typu A
  end if
end for

Kódování

1: Sorting Pass()
2: Refinement Pass()
3: $n = n - 1$
4: jdi na krok 1

Sorting Pass

5: **for all** (i, j) in LIP **do**
6: pošli na výstup $S_n(i, j)$
7: **if** $S_n(i, j) = 1$ **then**
8: přesuň (i, j) do LSP
9: pošli na výstup znaménko $c(i, j)$
10: **end if**
11: **end for**
12: **for all** záznam (i, j) in LIS **do**
13: **if** záznam (i, j) je typu A **then**
14: pošli na výstup $S_n(\mathcal{D}(i, j))$
15: **if** $S_n(\mathcal{D}(i, j)) = 1$ **then**
16: **for all** $(k, l) \in \mathcal{O}(i, j)$ **do**
17: pošli na výstup $S_n(k, l))$
18: **if** $S_n(k, l))$=1 **then**
19: přidej (k, l) do LSP
20: pošli na výstup znaménko $c(k, l)$
21: **end if**
22: **if** $S_n(k, l))$=1 **then**
23: přidej (k, l) na konec LIP
24: **end if**
25: **end for**
26: **end if**
27: **if** $\mathcal{L}(i, j) \neq \emptyset$ **then**
28: přesuň (i, j) na konec LIS jako záznam typu B
29: jdi na krok 30
30: **else**
31: odstraň (i, j) z LIS
32: **end if**
33: **end if**
34: **if** záznam (i, j) je typu B **then**
35: pošli na výstup $S_n(\mathcal{L}(i, j))$
36: **if** $S_n(\mathcal{L}(i, j)) = 1$ **then**
37: přidej $\forall(k, l) \in \mathcal{O}(i, j)$ na konec LIS jako záznam typu A
38: odstraň (i, j) z LIS
39: **end if**
40: **end if**
41: **end for**

Refinement Pass

42: **for all** (i, j) v LSP **do**
43: **if** nebylo (i, j) přidáno v posledním Sorting Pass **then**
44: pošli na výstup n-tý nejvíce významný bit $|c(i, j)|$
45: **end if**
46: **end for**

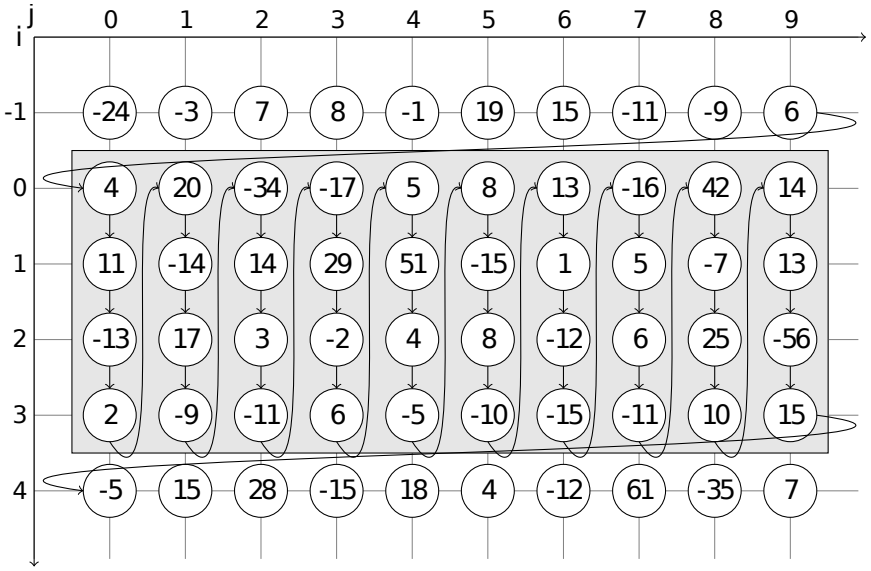
3.3 EBCOT

Obsah této kapitoly je tvořen informacemi převzatými ze zdrojů [7] a [20].

Jedním z algoritmů pro zakódování koeficientů podle standardu JPEG 2000 je EBCOT (*Embedded Block Coding with Optimized Truncation*). Samotné kódování se skládá ze dvou fází – označují se Tier 1 a Tier 2. Tier 1 je tvořena sekvencí průchodů – Signifigance Propagation, Magnitude Refinement a Cleanup Pass, které budou vysvětleny dále na příkladě. Tyto průchody kódují koeficienty v jednom ze čtyř kontextů – RL (*Run-length*), ZC (*Zero Coding*), MR (*Magnitude Refinement*) a SC (*Sign Coding*).

Koeficienty z daného subpásma se nejdříve převedou do reprezentace pomocí bitových rovin, které mají různou úroveň významnosti – od bitové roviny 0 (LSB), 1,2 až po $n-1$ (MSB) bitovou rovinu, kde n je celkový počet rovin. MSB představuje nejvíce významnou bitovou rovinu, LSB naopak nejméně významnou. Pokud mají nejvíce významné roviny pouze nulové bity koeficientů, tj. neobsahují informace pro zakódování, zaznamená se jejich počet, přeskočí se a kódování začíná nejvíce významnou rovinou s nenulovými bity koeficientů.

Subpásmo se dále rozdělí na bloky – matice, nejčastěji o rozměrech 64×64, nebo 32×32 koeficientů. Každý takovýto blok je možné zpracovávat nezávisle na ostatních blocích, je tedy možné výpočty urychlit paralelním zpracováním.



Obrázek 3.4: Blok koeficientů

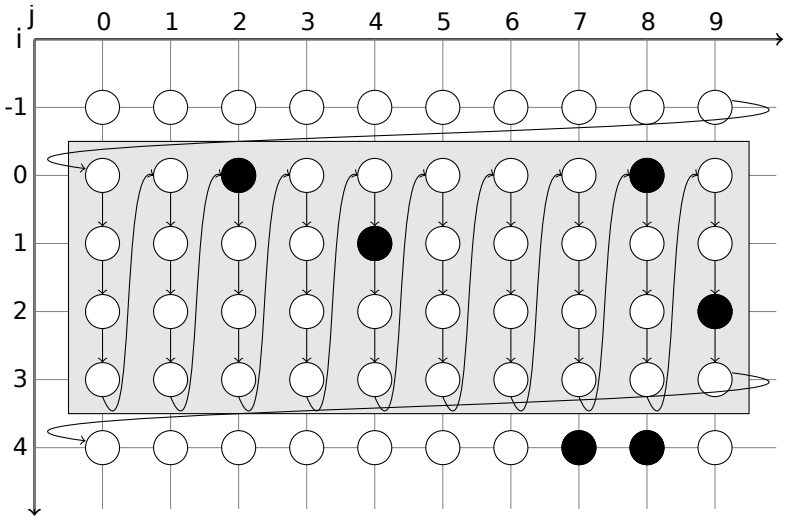
Zdroj: [7]

Uvažujme koeficienty z LH subpásma po kvantizaci. Vezme se první blok koeficientů, který je zobrazen na obrázku 3.4. Blok se dále dělí na skupiny čtyř řádků koeficientů, po kterých se provádí zakódování. Čísla v kroužcích představují hodnoty koeficientů, i označuje index řádku, j index sloupce. V rámci skupiny se jednotlivé koeficienty zpracovávají po jednotlivých sloupcích, jak ukazují šipky na obrázku 3.4.

Koeficient s nejvyšším řádem je na pozici (4,7) s hodnotou 61. Tato hodnota se dá vyjádřit pomocí 6 bitových rovin, očíslovaných 0-5. Začíná se první skupinou čtyř řádků z nejvíce významné roviny 5.

3.3.1 Kódování první bitové roviny

Průchody Signifigance Propagation a Magnitude Refinement se provádějí nad významnými koeficienty. Protože při startu kódování není žádný významný koeficient znám, tak se nad nejvíce významnou bitovou rovinou provede pouze třetí průchod Cleanup Pass. Jedná se o čistící průchod (z angl. cleanup), v tomto průchodu se zakódují všechny koeficienty, které se v dané skupině koeficientů nezakódovaly v průchodech Signifigance Propagation a Magnitude Refinement.



Obrázek 3.5: Bitová rovina 5

Zdroj: [7]

3.3.2 Průchod Cleanup Pass

Bitová rovina 5, nad kterou se provede tento průchod, je zobrazena na obrázku 3.6. Průchod začíná ve sloupci s indexem $j = 0$. Protože v tomto sloupci nejsou žádné významné koeficienty, a také nejsou známy v okolních sloupcích, pro kódování se použije kontext Run-length. Projde se celý sloupec, a jelikož neobsahuje významné koeficienty, do výstupního řetězce se zapíše symbol 0 v kontextu RL – RL(0). Pokračuje se druhým sloupcem $j = 1$. V tomto sloupci ani v sousedních se opět nenachází významné koeficienty, zapíše se tedy RL(0).

Další v pořadí je sloupec $j = 2$, ve kterém je významný koeficient na pozici (0,2). Kódování pokračuje v RL kontextu do té doby, než se objeví významný koeficient, v ten okamžik se zapíše symbol 1 – RL(1). Skončí se zápis kontextem RL a zaznamená se pozice významného koeficientu v kontextu UNIFORM – pro pozici (0,2) je to záznam UNI(00). (V kontextu UNIFORM jsou jednotlivé symboly považovány za stejně pravděpodobné). Kromě pozice se také ihned zaznamená znaménko významného koeficientu v kontextu Sign Coding.

$\bar{\chi}^h$	$\bar{\chi}^v$	κ^{SC}	$\hat{\chi}$
1	1	SC4	1
1	0	SC3	1
1	-1	SC2	1
0	1	SC1	1
0	0	SC0	1
0	-1	SC1	-1
-1	1	SC2	-1
-1	0	SC3	-1
-1	-1	SC4	-1

Tabulka 3.1: Predikce znaménka v kontextu SC

Symbol tohoto kontextu je určen predikcí na základě znamének známých sousedních koeficientů ve čtyřokolí, podle tabulky 3.1. $\bar{\chi}^h$ reprezentuje znaménko dvou horizontálních sousedů, $\bar{\chi}^v$ pak znaménko dvou vertikálních sousedů aktuálního koeficientu. V tabulce 3.1 jsou uvedeny hodnoty pro subpásma LL, LH, HH, pro pásmo HL je třeba hodnoty invertovat. $\bar{\chi}$ je rovno 1, pokud mají oba dva sousedé kladná znaménka, nebo jeden má kladné znaménko a druhý zatím znaménko nemá. $\bar{\chi}$ je rovno 0, pokud mají oba dva sousedé opačná znaménka, nebo oba dva znaménka zatím nemají. $\bar{\chi}$ je rovno -1, pokud mají oba dva sousedé záporná znaménka, nebo jeden má záporné a druhý zatím znaménko nemá. $\hat{\chi}$ je hodnota predikce znaménka pro kódovaný koeficient. Pokud je predikce správná, v kontextu SC se na výstup zapíše symbol 0, jinak se zapíše symbol 1. Pro koeficient na pozici (0,2) platí, že jeho sousedé nemají znaménka, predikované znaménko je tedy záporné. Znaménko však záporné není, proto se do výstupního řetězce zapíše SC(1).

subpásma LL, LH			**subpásmo HL**			**subpásmo HH**		κ^{ZC}
κ^h	κ^v	κ^d	κ^h	κ^v	κ^d	κ^d	$\kappa^h + \kappa^v$	
0	0	0	0	0	0	0	0	ZC0
0	0	1	0	0	1	0	1	ZC1
0	0	≥ 2	0	0	≥ 2	0	≥ 2	ZC2
0	1	x	1	0	x	1	0	ZC3
0	2	x	2	0	x	1	1	ZC4
1	0	0	0	1	0	1	≥ 2	ZC5
1	0	≥ 1	0	1	≥ 1	2	0	ZC6
1	≥ 1	x	≥ 1	1	x	2	≥ 1	ZC7
2	x	x	x	2	x	≥ 3	x	ZC8

Tabulka 3.2: κ^{ZC} představuje ZC kontext, x jakoukoli hodnotu

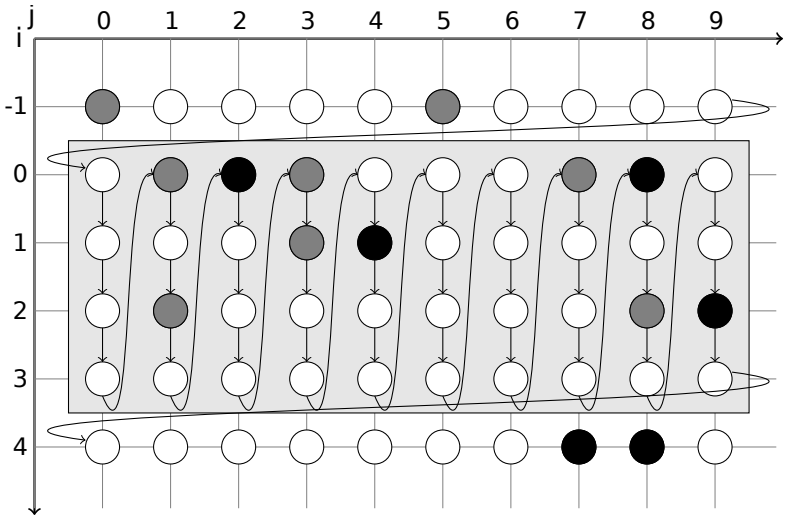
Ze sloupce $j = 2$ zbývají koeficienty na pozicích (1,2), (2,2), (3,2). Jelikož RL kontext byl už ukončen, po kontextu SC se pokračuje kontextem ZC (Zero Coding). Pro určení 1 z 9 symbolů na výstupu se používá predikce na základě dosud známé významnosti sousedů v osmiokolí koeficientu, hodnoty predikce v konkrétních případech jsou uvedeny v tabulce 3.2. κh, κv, κd značí sousední koeficienty v horizontálním, vertikálním a diagonálním směru. V okolí koeficientu (1,2) se nachází pouze jeden významný koeficient – a to na pozici (0,2), ve vertikálním směru. Aktuální koeficient (1,2) významný není, výsledkem predikce pro tuto

pozici je symbol 0 v kontextu ZC0. Následující 2 koeficienty ve sloupci $j = 2$ nemají ve svém okolí významné koeficienty a samy o sobě jsou také nevýznamné, výstupem v kontextu ZC budou 2 symboly 0 – ZC0(0), ZC0(0).

Pokračuje se sloupcem $j = 3$. Protože v sousedství prvního koeficientu (0,3) je významný koeficient (0,2), nerealizuje se kódování v RL kontextu. Místo něj se použije kontext ZC, na výstup se zapíše následující kontext koeficientů: ZC5(0), ZC1(0), ZC0(0), ZC0(0). Na pozici (0,4) ani v jejím okolí se nenachází známý významný koeficient, pro sloupec $j = 4$ se kodér přepne do RL kontextu. Hned druhý koeficient na pozici (1,4) v pořadí je významný, tím pádem se ukončuje RL kontext, zapíše se RL(1). V kontextu UNIFORM se zaznamená pořadí prvního významného koeficientu – výstupem je UNI (01). Po UNIFORM následuje SC kontext, kde se zakóduje znaménko aktuálního koeficientu. Příslušný kontext je SC0, predikuje se pozitivní znaménko. Aktuální koeficient je pozitivní, na základě shody predikce a skutečnosti se vygeneruje symbol 0 v kontextu SC0. Po SC následuje kódování zbylých dvou koeficientů ve sloupci, výsledkem je ZC3(0) a ZC0(0). Výstupy při kódování zbylých sloupců jsou uveden v tabulce 3.3.

	Zakódované symboly a kontexty	Komentář
$j = 0$	RL(0)	Run-Length kontext.
$j = 1$	RL(0)	Run-Length kontext.
$j = 2$	RL(1) UNI(00) SC0(1) ZC3(0) ZC0(0) ZC0(0)	Run-Length do koeficientu (0,2). Pozice prvního významného koeficientu. Predikce znaménka v SC0 kontextu je nesprávná. Zbývající koeficienty sloupce kódovány ZC kontextem.
$j = 3$	ZC5(0) ZC1(0) ZC0(0) ZC0(0)	RL se neaktivuje, koeficient (0,2) je významný.
$j = 4$	RL(1) UNI(01) SC0(0) ZC3(0) ZC0(0)	Run-Length do koeficientu (1,4) Pozice prvního významného koeficientu. Predikce znaménka v SC0 kontextu je správná. Zbývající koeficienty sloupce kódovány ZC kontextem.
$j = 5$	ZC1(0) ZC5(0) ZC1(0) ZC0(0)	RL se neaktivuje, koeficient (1,4) je významný.
$j = 6$	RL(0)	Run-Length kontext.
$j = 7$	RL(0)	Run-Length kontext.
$j = 8$	RL(1) UNI(00) SC0(0) ZC3(0) ZC0(0) ZC0(0)	Run-Length do koeficientu (0,8). Pozice prvního významného koeficientu. Predikce znaménka v SC0 kontextu je správná. Zbývající koeficienty sloupce kódovány ZC kontextem.
$j = 9$	ZC5(0) ZC1(0) ZC0(1) SC0(1) ZC3(0)	RL se neaktivuje, koeficient (0,8) je významný. Významný koeficient zakódován v kontextu ZC0. Predikce znaménka v SC0 kontextu je nesprávná.

Tabulka 3.3: Postup kodóvaní bitové roviny $n = 5$ průchodem Cleanup pass



Obrázek 3.6: Bitová rovina 4

Zdroj: [7]

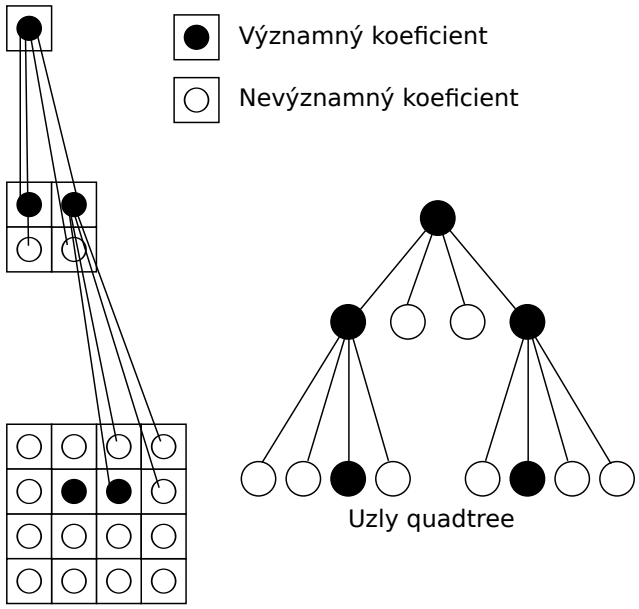
3.3.3 Pokračování a ukončení kódování

Po zpracování bitové roviny $n = 4$ se stejným způsobem zakódují méně významné roviny, sekvencí průchodů Signifigance Propagation-Magnitude Refinement-Cleanup Pass až do roviny $n = 0$. Poté přichází na řadu vhodný aritmetický kodér, který výstup překóduje na základě pravděpodobností jednotlivých symbolů, přičemž tyto pravděpodobnosti jsou aktualizované po každém symbolu. Tímto končí první část (Tier 1) algoritmu EBCOT. V druhé části (Tier 2) se výstup z aritmetického kodéru optimalizuje jako kompromis vůči zkreslení, procesem zvaným Post-Compression Rate-Distortion OPTimization (PCRD-opt).

3.4 EZBC

Algoritmus Embedded Zero Block Coding kombinuje přístupy používané dvěma typy algoritmů – kontextové kódování koeficientů z LZC, a stromové struktury quadtree z algoritmů EZW, SPIHT.

Listy quadtree (tzv. pixel level) tvoří u EZBC řády koeficientů z podpásma. Přímý předek 4 uzlů pak obsahuje maximální hodnotu ze svých (přímých) potomků. Tímto rekurzivním seskupením spolu se vzdáleností od listu roste velikost oblasti koeficientů pokryté jedním uzlem stromu, jak je zobrazeno na obrázku 3.7. Tímto lze zakódovat jednu oblast koeficientů několika málo bity. Získání hodnoty určitého koeficientu nebo oblasti koeficientů pak probíhá opačnou cestou, struktura se rekurzivně dělí na quadtree až do hledané úrovně. Jednotlivá dělení jsou kódována, aby bylo možné proces zreprodukovat při rekonstrukci původních koeficientů během dekódování.



Obrázek 3.7: Pokrytí koeficientů uzly

Zdroj: [9]

Kódování probíhá nad reprezentací koeficientů v podobě bitových rovin, od nejvíce významně roviny po nejméně významnou. V první fázi se provede test významnosti nad každým uzlem quadtree dané bitové roviny, a to porovnáním s určitým prahem. Pokud je uzel shledán významným, rozdělí se na čtyři přímé potomky, u kterých se opět testuje významnost. Takto se rekurzivně provádí testování a dělení, dokud se nedojde na listovou úroveň, pixel-level. Listy, které představují pixely, se opět otestují na významnost, významné listové koeficienty se zakódují spolu se znaménkem. Poté se provede zpřesňovací průchod nad dosavadními zakódovanými symboly.

Podobně jako u SPIHT se informace o rozdělení udržují v seznamech, u EZBC to jsou seznamy LIN – List of Insignifigant Nodes a LSP – List os signifigant pixels, pro každé subpásmo jedna jejich dvojice. Výstupem je sekvence symbolů – symboly vyznačující významnost uzlů v LIS, významnost potomků, znaménko kódovaného koeficientu a zpřesnění koeficientu. [9].

Algoritmus 2 EZBC

Definice

- $c(i,j)$, $m(i,j)$ - kvantifikovaný koeficient subpásma a jeho MSB na pozici (i,j)
- QT - quadtree reprezentující koeficienty daného subpásma
- $QT_k[l](i,j)$ - quadtree uzel subpásma k, úrovně l, na pozici (i,j)

$$\begin{aligned} QT_k[0](i,j) &\equiv |c_k(i,j)| \\ QT_k[l](i,j) &\equiv \max\{QT_k[l-1](2i,2j), \\ &\qquad QT_k[l-1](2i,2j+1), \\ &\qquad QT_k[l-1](2i+1,2j), \\ &\qquad QT_k[l-1](2i+1,2j+1)\} \end{aligned} \tag{3.2}$$

- D_k, D_{max} – hloubka quadtree subpásma k a maximální hloubka quadtree
- $\mathrm{LIN}_k[l]$ – seznam nevýznamných uzlů (List of insignifigant nodes) úrovně l quadtree subpásma k
- LSP_k – seznam významných koeficientů (List of signifigant pixels) subpásma k

Test významnosti uzlu (i,j) **v bitové rovině** n

- $S_n(i,j) \equiv \begin{cases} 1 & \text{if } n \leq m(i,j) \\ 0 & \text{jinak} \end{cases}$

Inicializace

- $\mathrm{LIN}_k[l] \equiv \begin{cases} (0,0) & l = D_k \\ 0 & \emptyset\text{jinak} \end{cases}$
- $LSP_k = \emptyset \forall k$
- $n = \lfloor \log_2(\max_{(k,i,j)}|c_k(i,j)|) \rfloor$
- pošli na výstup n

Kódování

for $l = 0 : D_{max}$ **do**
  for $k = 0 : K-1$ **do**
    CodeLIN(k,l)
  end for
end for
for $k = 0 : K-1$ **do**
  CodeLSP(k)
end for
$n = n - 1$, jdi na první krok kódování

CodeLIN(k,l)

for all (i,j) in $\mathrm{LIN_k[l]}$ **do**
  zakóduj $S_n(i,j)$
  if $S_n(i,j) = 0$ **then**
    uzel (i,j) zůstává v $\mathrm{LIN_k[l]}$
  else
    if $l = 0$ **then**
      zakóduj znaménkový bit $c(i,j)$ a přidej (i,j) do $\mathrm{LSP_k}$
    else
      CodeDescendants(k,l,i,j)
    end if
  end if
end for

CodeDescendants(k, l, i, j)
for all uzel (x, y) v seznamu $(2i, 2j), (2i, 2j + 1), (2i + 1, 2j), (2i + 1, 2j + 1)$ z quadtree uzlu úrovně $l - 1$ subpásma k **do**
zakóduj $S_n(x, y)$
if $S_n(i, j) = 0$ **then**
přidej (x, y) do $\mathrm{LIN_k}$[l-1]
if $l = 1$ **then**
zakóduj znaménkový bit $c_k(x, y)$ a přidej (y, y) do $\mathrm{LSP_k}$
else
CodeDescendants$(k, l - 1, x, y)$
end if
end if
end for
for all koeficient (i, j) in $\mathrm{LSP_k}$ **do**
zakóduj bit $nc_k(i, j)$
end for

3.5 MRWD

Morphological Representation of Wavelet Data (MRWD) je jedním z algoritmů kódování koeficientů založených na hledání morfologií v obrazu. Podobně jako algoritmy EZW a SPIHT, využívá závislosti mezi úrovněmi dekompozice obrazu, jak je naznačeno na obrázku 3.2. Díky těmto závislostem mají významné i nevýznamné koeficienty v jednotlivých subpásmech tendenci objevovat se ve skupinách, které lze zakódovat několika málo bity. Tyto dva algoritmy však pracují se stromovými strukturami *zerotree* a *spatial orientation tree*, které disponují několika omezeními. Předpokládají, že se významné a nevýznamné koeficienty shlukují do čtvercových matic, jejichž rozměry jsou mocninami dvou. Ve skutečnosti toto nebývá často splněno (některé elementy v obrazu, např hrany), a efektivita komprese je pak snížena.

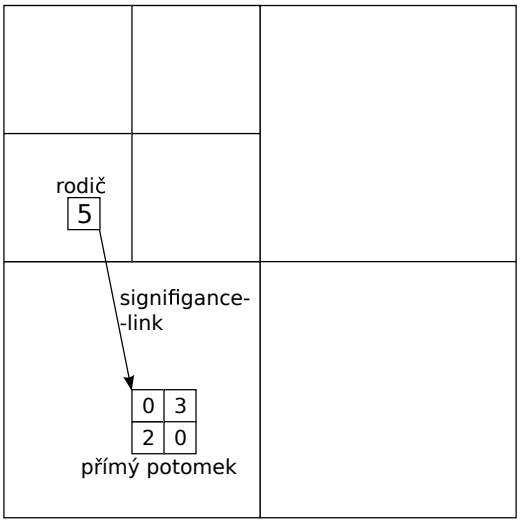
Algoritmus MRWD umožňuje efektivně kódovat i oblasti jiného tvaru, než je čtverec, využitím speciálních průchodů nad bitovými rovinami. Principiálně se jedná o rozšíření masky významnosti bitů, která mapuje pozice významných koeficientů, z předchozí bitové roviny. Prohledává se osmiokolí hran této masky. V případě, že se nalezne významný koeficient, začne se rekurzivně prohledávat jeho okolí s cílem odhalit případné sousední významné koeficienty. Výsledkem hledání je obecně nepravidelný, amorfní shluk koeficientů, který by se jinými algoritmy zakódoval s menší efektivitou než pomocí MRWD.

Podobně jako u EZW lze na výstup tohoto algoritmu aplikovat entropické (např aritmetické) kódování, které operuje na základě pravděpodobností výstupních symbolů. U algoritmu MRWD je pravděpodobnost symbolů reprezentující shluky koeficientů odlišná od pravděpodobností symbolů shluků u jiných algoritmů, toto umožňuje jejich efektivnější kompresi. [17].

3.6 SLCCA

Kódování pomocí Signifigance Linked Connected Component Analysis rozšiřuje myšlenku hledání morfologií obraze představenou v MRWD. Místo osmiokolí se prohledává rozšířená oblast koeficientů, díky tomu vzniká menší počet propojených komponent souvislosti. Je třeba zvolit optimální velikost okolí, příliš velká sousední oblast s sebou nese větší množství nevýznamných koeficientů, které je třeba zakódovat. Podobně jako u ostatních algoritmů lze na výstup aplikovat aritmetický, nebo jiný entropický kodér na zhuštění výsledného bitového toku.

Jednou z vlastností koeficientů využívaných algoritmem SLCCA je tendence snižování významnosti koeficientů v dané oblasti od nejvyšší úrovně dekompozice k nejnižší, což je vyjádřeno vztahem signifigance-link. Jde o propojení mezi rodičovským komponentem/shlukem koeficientů ležícím ve vyšší úrovni dekompozice a shlukem, který obsahuje jeho přímé potomky. Přímý potomek se vyskytuje ve stejné oblasti subpásma jako rodič o jednu úroveň rozkladu níže. Pokud je rodič a aspoň jeden z jeho přímých potomků významnými koeficienty, pak jsou příslušné komponenty ve vztahu signifigance-link, jak je znázorněno na obrázku 3.8. Tento vztah zachycuje závislost mezi úrovněmi rozkladu.



Obrázek 3.8: Vztah signifigance-link

Zdroj: [5]

Pro kódování shluků koeficientů se používají následující 4 symboly: POS, ZERO a LINK. POS a NEG slouží k reprezentaci znaménka, ZERO reprezentuje nevýznamné koeficienty vymezující hranici daného shluku (clusteru). Symbol LINK popisuje přítomnost vztahu signifigance-link. Koeficienty se zpracovávají v reprezentaci formou bitových rovin, jejich významnost v určité bitové rovině se vyjádří symbolem 1, nevýznamné koeficienty se zakódují symbolem 0.

Samotné kódování využívá tři seznamy – List of scan order (LSO), list of child clusters (LCC) a list of signifigant coeficients (LSC), které jsou typu FIFO. Každý prvek těchto front je určen souřadnicemi [x, y], zápis c[x, y] představuje koeficient na souřadnicích [x, y]. Jako u ostatních algoritmů je vstupem 2D pole hodnot koeficientů vzniklých vlnkovou transformací obrazu, které je převedeno do reprezentace pomocí bitových rovin.

Jako první se v každém subpásmu provede analýza komponent souvislosti nad koeficienty pomocí podmíněné dilatace. Pokud je komponenta souvislosti extrémně malá, zahodí se. Dále se seznam LSO naplní všemi koeficienty z pyramidy subpásem, od nejvyšší úrovně s nejhrubější aproximací obrazu po nejnižší úroveň. V rámci jedné úrovně se jednotlivá subpásma zpracovávají v pořadí LL, LH, HL, HH, koeficienty v subpásmu pak po řádcích shora dolů, zleva doprava. [5].

3.6.1 WDR

Algoritmus *Wavelet Difference Reduction* (WDR) patří do skupiny progresivních kódování koeficientů podobně jako EZW a SPIHT. Jako člen této skupiny hledá významné koeficienty podle srovnání hodnoty s hodnotu prahu, který se s každou iterací (zpřesňováním) snižuje. V rámci jedné této iterace se provádí dva, nebo více průchodů, označované jako dominantní a zpřesňovací. Zpřesňovací průběh je velice podobný ostatním algoritmům z této rodiny, provádění dominantního průchodu se však od běžného způsobu pro tuto rodinu liší.

V dominantním průběhu se prochází koeficienty z daného pásma v předem určeném pořadí – rastrový průchod, Mortonův průchod, zigzag a další. Při objevení významného koeficientu odešle na výstup jeho znaménko a také vzdálenost od posledního nalezeného významného koeficientu. Tímto způsobem se dá určit přesná pozice daného koeficientu v pásmu, odvíjí se od součtu vzdáleností předchozích významných koeficientů a způsobu procházení subpásem.

Výstup WDR je tvořen sekvencí čtyř symbolů, a lze jej podobně jako u ostatních algoritmů podrobit aritmetickému kódování, který dále zhušťuje bitový tok. [22].

3.7 Seznam dalších algoritmů

- Adaptively Scannned Wavelet Difference Reduction (ASWDR)
- Backward Coding of Wavelet Trees (BWCT)
- Listless Zerotree Coding (LZC)
- Spatial Orientation Tree Wavelet (STW)
- Set Partitioning Embedded bloCK (SPECK)
- Listless SPECK (LSK)
- Improved LSK (ILSK)
- Subband-Block Hierarchical Partitioning (SBHP)
- Signifigance Tree Qauntization (STQ)
- Embedded Quad Tree in DCT Domain (EQDCT)
- Wavelet Block Truncation Coding (WBTC)

3.8 Entropické kódování

Po zakódování koeficientů jedním z algoritmů pro kompresi obrazu je vhodné minimalizovat redundanci výstupních symbolů na minimum. Nejmenší počet bitů pro uchování bitového toku, kterého lze teoreticky dosáhnout bez dalších ztrát informace, je určen entropií. Entropii jednoho symbolu ve výstupním řetězci lze vyjádřit takto (jednotkou je *bit*):

$$E = \log_2(c^l).p_i = \log_2(c^l).\frac{s_i}{l} \tag{3.3}$$

c – počet různých symbolů

p_i – pravděpodobnost výskytu symbolu ve výstupním řetězci
s_i – počet výskytů symbolu ve výstupním řetězci
l – délka výstupního řetězce, počet symbolů ve výstupním toku

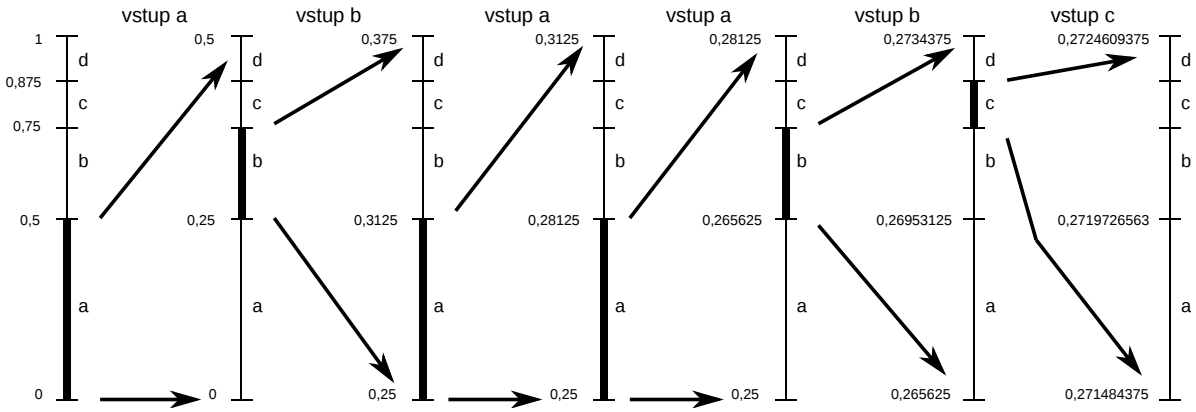
Entropii pro celý výstup lze získat součtem entropií jednotlivých symbolů. Cílem bezeztrátové komprese je přiblížit se této hodnotě a minimalizovat délku finálního bitového toku. Metod entropického kódování, jak tohoto dosáhnout, je více, např. RLE (*Run-Length Encoding*), BWT(*Burrows-Wheelerova transformace*), *Huffmanovo kódování*, nebo pomocí *aritmetického kódování*. V principu jde o vyjádření původní sekvence symbolů co nejmenším výsledným počtem bitů, kdy lze každému symbolu přiřadit různý počet bitů. Některé přístupy lze kombinovat pro lepší výsledky – lze použít Run-Length Encoding na zmenšení délky výstupního toku a poté jej zakódovat jinou metodou.

3.8.1 Aritmetické kódování

Jednou z možností, jak provést entropické zakódování symbolů reprezentujících koeficienty je aritmetické kódování. Oproti Huffmanovu kódování, jehož výsledkem je sekvence bitů pro každý symbol, je výstupem aritmetického algoritmu pro sekvenci symbolů jedno reálné číslo z intervalu $< 0, 1)$. Výhodou je schopnost přiřazovat symbolu neceločíselný počet bitů, díky tomu lze dosáhnout kratší výsledné sekvence bitů než u jiných metod, které tolik neodpovídají vstupům z reálného světa. Využívá se stanovené četnosti symbolů určených k zakódování, podle způsobu zjištění frekvence symbolů se dají rozlišit tři přístupy ke svázání pravděpodobnosti symbolů s vlastním mechanismem kódování:

- statický model – před začátkem kódování se stanoví pravděpodobnosti symbolů na základě celého vstupu, během kódování se pravděpodobnosti nemění
- order-n model – pravděpodobnosti symbolů se stanovují na základě n posledních kódovaných symbolů, v každém kroku se mění
- adaptivní model – pravděpodobnosti symbolů nejsou předem známy, přepočítávají se v každém kroku

Nejdříve se podle typu modelu stanoví pravděpodobnosti jednotlivých symbolů, uvažujme pro jednoduchost statický s konstantními pravděpodobnostmi. Algoritmus operuje nad intervaly, výsledkem je reálné číslo z vypočteného rozsahu mezí. Symboly se seřadí, a každému se přidělí část intervalu $< 0, 1)$ tak, aby se jednotlivé přidělené intervaly nepřekrývaly a aby rozsah odpovídal velikosti pravděpodobnosti. Neboli jeden interval začíná u konce druhého a rozdíl mezí je roven pravděpodobnosti symbolu. Vezme se první interval $< 0, 1)$. V prvním kroku se podle prvního symbolu určí interval, se kterým se bude dále pracovat. Je to interval přidělený tomuto symbolu v dřívějším kroku, ten se dále opět zmenší podle stejného principu na základě druhého symbolu. Toto se opakuje do posledního symbolu ve výstupním toku, výsledkem bývá velice malý interval reprezentovatelný reálným číslem. Postup kódování pro sekvenci symbolů abaabc při možných symbolech a,b,c,d je naznačen na obrázku 3.9.



Obrázek 3.9: Aritmetické kódování sekvence *abaabc* při symbolech a,b,c,d

Zdroj: [13]

Pro dekódování je zapotřebí znát reálné číslo z kodéru, rozdělení intervalu $< 0, 1)$ na menší podle symbolů a symboly samotné. Pak stačí zopakovat dělení intervalů, tak aby se reálné číslo z kodéru v každém kroku nacházelo uvnitř intervalu, a dekódování proběhne úspěšně.

Protože výpočty nad reálnými čísly probíhají pomaleji než nad celými a při reprezentaci v plovoucí řádové čárce můžou vznikat odchylky obzvlášť u malých čísel, je více než vhodná modifikace algoritmu tak, aby intervaly byly určené pomocí celých čísel. Tímto způsobem se kódování podstatně urychlí a riziko zavlečení chyb se zmenší.

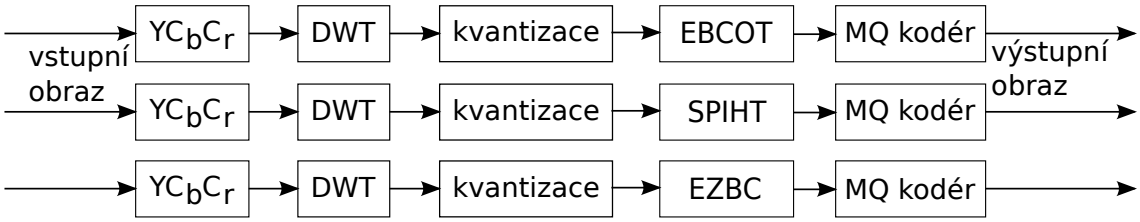
Kapitola 4

Implementace

4.1 Návrh

Jedním z cílů praktické části je navrhnout a implementovat v jazyce C/C++ programy pro provádění JPEG 2000 komprese dvěma různými kódovacími postupy, a to EBCOT, SPIHT a EZBC. Jelikož jde o porovnání aspektů těchto algoritmů, tak pro zachování objektivity musí mít stejné podmínky, tj. zbývající části komprese JPEG2000 budou společné - převod barev do YC_bC_r, vlnková transformace, kvantizace a nakonec MQ kodér. Části programu realizující jednotlivé algoritmy musí být mezi sebou zaměnitelné, je třeba aby měly stejné rozhraní a typy vstupu a výstupů, jsou tedy reprezentované vyměnitelnými funkčními bloky. Pro co nejlepší výpovědní hodnotu testování každému algoritmu odpovídá 1 spustitelný soubor vytvořený ze společných části komprese a 1 typu kódování.

Výsledná podoba aplikací je znázorněna na obrázku 4.1, stejnojmenné části představují jeden a ten samý kód.



Obrázek 4.1: Stuktura aplikací

Zdroj: vlastní zpracování

4.2 Průběh aplikace

Prvním společným krokem aplikací je načtení vstupního obrazu z podporovaných formátů na základě vstupního parametru. Pokud je obraz v barevném prostoru RGB, převede se do pro kompresi vhodnějšího YC_bC_r. Poté se provede diskrétní vlnková transformace nad 2D obrazovými daty, jejímž výstupem je pyramidový rozklad koeficientů na jednotlivá subpásma. Hodnoty koeficientů lze kvantifikovat, nebo ponechat nezměněné. Tímto končí společná část aplikací.

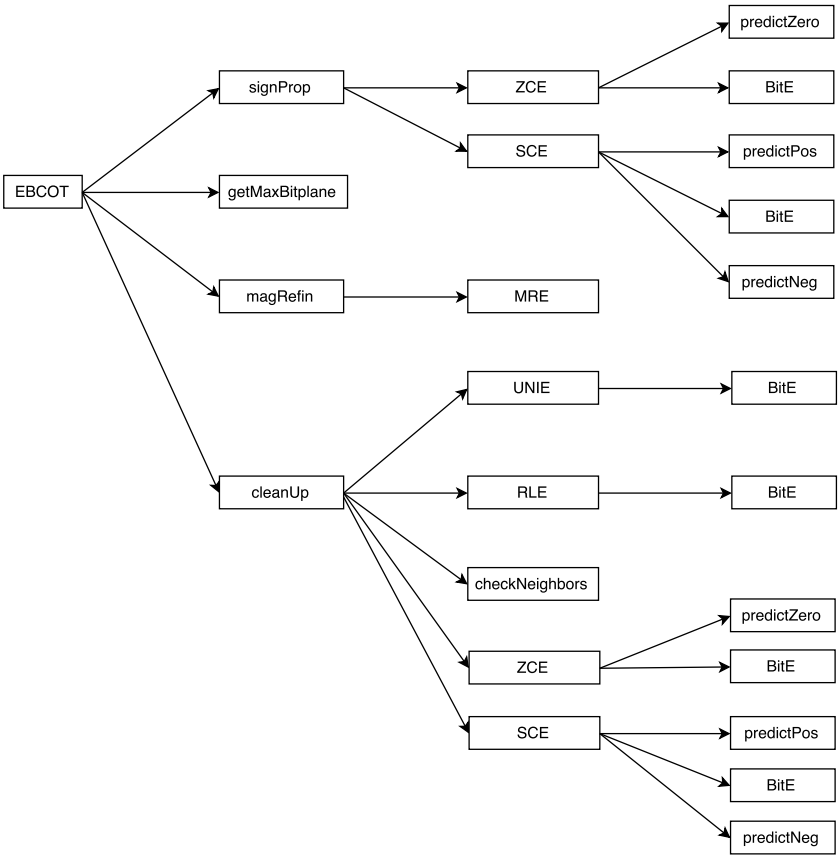
Nad posledním výstupem jsou pak prováděny kódovací algoritmy, součástí je i měření rychlosti zakódování koeficientů. Výstup z algoritmů je podroben aritmetickému MQ kodéru, který je sdílený napříč aplikacemi. Výstupní bitový tok je předmětem srovnávací části práce.

4.3 EBCOT

Kódování podle algoritmu EBCOT je definováno zdrojovým kódem v jazyce C v souboru ebcot.c. Při programování jsem vycházel z výše popsané varianty algoritmu v kapitole 2. Nezbytnou součásti tohoto kódování jsou tabulky kontextů jednotlivých čtyř kódovacích primitiv ve formě složek struktury EBCOT_tables - SC, RL, ZC a UNI s odpovídajícími vstupními parametry tabulek a konkrétního symbolu a kontextu na výstupu. Dále pak datová strukutra code_block s blokem zdrojových koeficientů pro zpracování a s bezprostředním okolí zpracovávaných řádků.

Nejprve se zavolá funkce getMaxBitplane pro zjištění počáteční bitové roviny - rovina s nejvíce významnými bity koeficientů. Pro průchody rovinami jsou rezervované tři procedury - magRefin, signProp a cleanUp. Jejich vstupem je skupina 4 řádků ze zdrojového bloku spolu s nejbližším okolím, a to jak v současné bitové rovině, tak i v souhrnu dříve zpracovaných bitových rovin. Zakódování kontextů se řídí kontextovými tabulkami, realizace probíhá procedurami ZCE, SCE, RLE, UNIE, zpřesňování koeficientů probíhá ve funkci MRE. Pro predikci kovaného znaménka je třeba zjistit znaménka sousedů, toto se realizuje pomocí funkce checkNeighbors.

Výstupem jsou symboly reprezentující znaménka a hodnoty bitů zpracovaných při jednotlivých průchodech bitovými rovinami. U znamének se jedná o predikci funkcemi predictNeg, predictPos a predictZero, bit je poslán na výstup procedurou BitE. Graf volání funkcí je na obrázku 4.2.



Obrázek 4.2: Neúplný graf volání funkcí implementace EBCOTu

Zdroj: vlastní zpracování

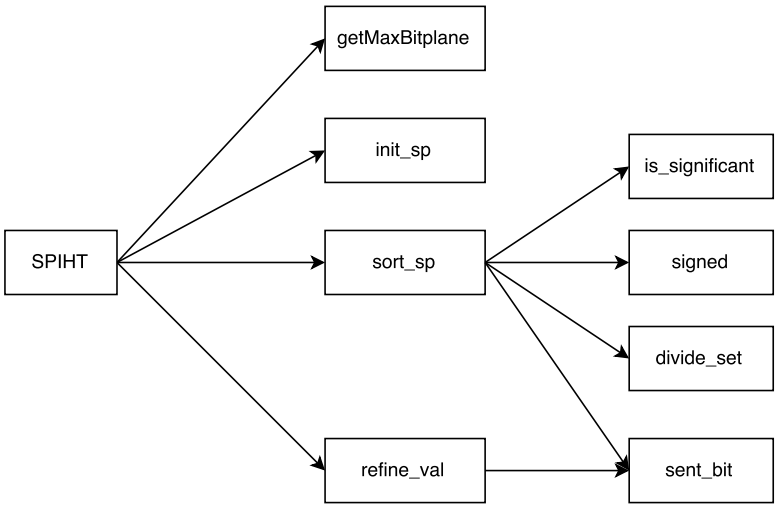
4.4 SPIHT

Program realizující kódování koeficientů algoritmem SPIHT je v souboru spiht2.c. Implementace vychází z původního článku, odpovídá popisu v předchozí kapitole. Základem jsou tři seznamy LIP, LSP, a LIS ve formě dynamicky alokovaných polí. K nim přísluší seznam bodů (souřadnic), který určuje pořadí zpracování koeficientů, také je stanoveno pořadí zpracování přímých potomků v uzlu. Důležitá je rovněž pomocná struktura spiht_header nesoucí informace o rozměrech matice koeficientů ke zpracování, dále pak specifikuje výstupní řetězec pro ukládání komprimovaných koeficientů.

Inicializační fázi kódování provádí funkce init_sp, nastaví počáteční práh a počáteční obsahy seznamů. Řadící průchod je realizován procedurou sort_sp, která vykonává operace stanovení významnosti a zakódování znaménka při postupném průchodu seznamy LIP a LIS. Významnost je určena funkcí is_signifigant, znaménko posílá na výstup funkce signed. Dělení množin provádí funkce divide_set. Pořadí zpracování přímých potomků dělené množiny je určeno staticky kompilací podle popisu v předchozí kapitole, lze jej změnit modifikací zdrojového kódu a opětovným přeložením.

Zpřesňovací průchod je doménou funkce refine_sp, která prochází nezpracované koeficienty a posílá jejich n-tý bit na výstup skrze funkci sent_bit. Třetí fází kódování je

snížení prahu, který je uložen v proměnné refine_val na polovinu. Poté se znovu předá řízení sort_sp, a takto se postupuje buď do dosažení požadované délky, nebo do nejnižší nastavené úrovně prahu. Obě tyto nastavení se nacházejí v pomocné struktuře through.



Obrázek 4.3: Graf volání funkcí implementace algoritmu SPIHT

Zdroj: vlastní zpracování

4.5 EZBC

Implementace kódování EZBC navazuje na původní článek, ve kterém byl algoritmus představen, jehož popis je podán v předchozí kapitole. Program, napsaný v jazyce C, je umístěn v souboru ezbc.c. Je složen z funkcí, které odpovídají původnímu článku, jmenovitě is_-significant, code_LIN, code_descendants, code_LSP, init_enc a inicializačních funkcí pro datové struktury. Podobně jako u SPIHT je kódování koeficientů realizováno pomocí seznamů, a to LIN a LSP, které jsou definované pro každé subpásmo zvlášť. Dále je třeba pracovat se strukturou reprezentující quadtree strom, jehož listy obsahují hodnoty koeficientů daného subpásma. Pro běh programu je také nezbytná správa paměti pro strukturu EZBC kodéru.

Seznam LIN představuje seznam nevýznamných uzlů určité úrovně quadtree stromu, LSP pak seznam významných koeficientů z daného subpásma. Quadtree je reprezentován rekurzivní strukturou složenou z jednotlivých uzlů stromu, jejichž obsah je maximum ze čtyřech přímých potomků, nebo absolutní hodnoty koeficientů u listových uzlů. Ve struktuře reprezentující EZBC kodér je umístěna hodnota postupně se snižujícího prahu pro testování významnosti uzlů, spolu s tabulkou výstupních kontextů a odkazem na zdrojová a výstupní data.

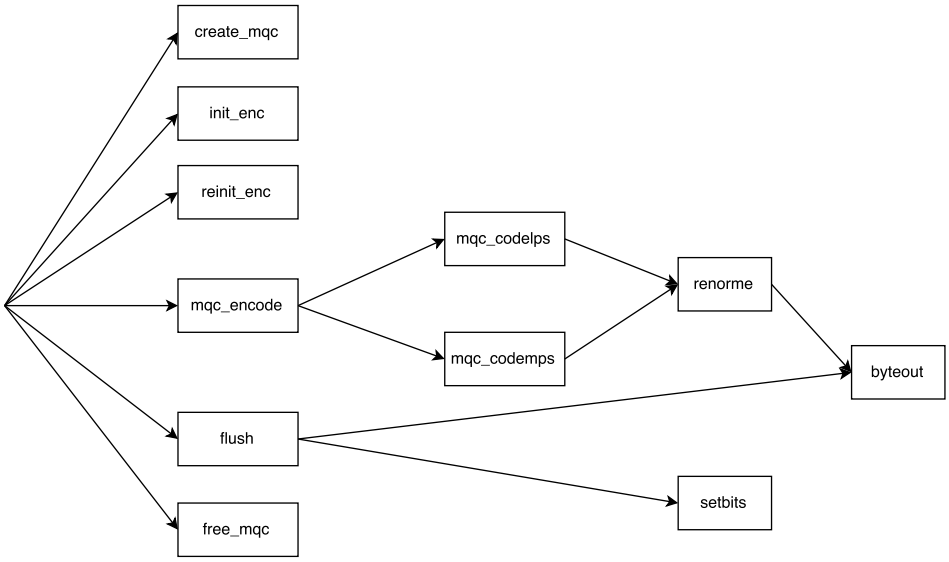
Funkce init_enc nastaví počáteční obsahy seznamů a vytvoří quadtrees všech subpásem. Následuje průchod těmito stromovými strukturami, který zakóduje nevýznamně koeficienty pomocí procedury code_LIN, postupující od listové úrovně všech quadtrees na vyšší úrovně. V rámci této funkce dochází k rozdělení nelistových uzlů na přímé potomky, které jsou zakódovány funkcí code_descendants. V druhém průchodu se pro každé subpásmo kódují významné koeficienty ze seznamu LSP skrze provádění funkce code_LSP. Test významnosti

v závislosti na hodnotě prahu je realizován funkcí is_significant. Třetí fází je snížení hodnoty prahu na polovinu, poté pokračuje provádění obou průchodů s nově sníženým prahem, dokud není dosaženo minimální hodnoty prahu.

4.6 MQ kodér

Implementace aritmetického MQ-kodéru je umístěna v souboru mq.c. Algoritmus byl naprogramován v jazyce C podle standardu JPEG 2000, skládá se z funkcí mqc_encode, code_mps, code_lps, flush, renorme a bit_encode, které odpovídají stejně pojmenovaným částem pseudokódu ze standardu. Jde o celočíselnou variantu, u které stačí celočíselná aritmetika, kterou lze provést rychleji než aritmetiku nad reálnými čísly. Procedury CODE0 a CODE1 ze standardu chybí, jsou nahrazeny jednoduchým větvením na základě porovnání hodnoty DC a MPS. Ve zdrojovém souboru se dále nacházejí funkce pro přidělení a uvolnění paměti pro mq kodér.

Součástí zdrojového kódu je struktura mqc reprezentující MQ-kodér obsahující tabulku MPS a kontextů, současný kontext, ukazatel na současná data bp, registry A, C a CT. MPS představuje více významný symbol, LPS méně významný. Kontext je uchováván jako index do tabulky obsahující pravděpodobnost více významného symbolu spolu s následujícím přiřazeným kontextem. Registry A, C a CT slouží k uchování pravděpodobnosti více významného symbolu. Vstupem aritmetického kódování je DC – decision a CX context – spolu se strukturou mqc je toto předáno jako trojice parametrů proceduře mqc_encode. Ta na základě porovnání hodnoty DC a současného MPS zavolá code_mps pro zakódování více významného symbolu, nebo code_lps pro poslání méně významného symbolu na výstup. V tělech funkcí code_mps a code_lps může docházet k zavolání funkce renorme, která normalizuje interval pravděpodobnosti symbolu MPS. Přitom dojde k úpravě hodnot v registrech A, C, CT případně se zavolá funkce flush, která pošle na výstup dříve zakódované symboly – skrze ukazatel na současná data bp. Registr A určuje velikost pravděpodobnosti, registr C spodní mez intervalu a registr CT reprezentuje počítadlo volných bitů do okamžiku přesunutí symbolů na výstup. Graf volání funkcí je na obrázku 4.4.



Obrázek 4.4: Graf volání funkcí implementovaného AC kodéru

Zdroj: vlastní zpracování

Výstupem MQ kodéru je sekvence bitů, určených kombinacemi vstupních symbolů DC a CX, při počátečních nastaveních pravděpodobností a MPS symbolů jednotlivých kontextů označených čísly 0-18.

Kapitola 5

Srovnání

V této kapitole představím základní objektivní metody vyhodnocení kvality obrazu, na kterých bude postaveno srovnání algoritmů komprese koeficientů z hlediska velikosti výstupního bitového toku. Následuje samotné vyhodnocení a vyvození výsledků z provedeného porovnání.

5.1 MSE

MSE (Mean Square Error) - kvadratická odchylka je základní metodou pro posouzení kvality při analýze a zpracování signálů. Oproti ostatním metodám je jednoduchá, nepotřebuje parametry, každý vzorek může být vyhodnocen nezávisle, nemá velké paměťové nároky a výpočet je rychlý. Nevýhodou je pak v některých případech výsledná hodnota neodpovídající kvalitě obrazu, např. při náhodné výměně 2 pixelů. Podobných hodnot MSE pak můžou dosahovat kvalitativně odlišné obrazy.

Uvažujme původní obraz $x \equiv [n1, n2]$ a komprimovaný obraz $\hat{x} \equiv [n1, n2]$ o stejných rozměrech N_1, N_2. Potom lze MSE vyjádřit podle vzorce:

$$\mathrm{MSE} = \frac{1}{N_1 N_2} \sum_{n_1=0}^{N_1-1} \sum_{n2=0}^{N_2-1} (x[n1, n2] - \hat{x}[n1, n2])^2 \tag{5.1}$$

Výsledná hodnota popisuje podobnost 2 zdrojových obrazů, pro identické obrazy je výsledkem nula, se zvyšující se MSE roste i odlišnost porovnávaných obrazů. Pro obrazy v RGB prostoru se 3 výsledné hodnoty zprůměrují, v prostoru YC_bC_r pak existují 3 hodnoty MSE, 1 pro každý kanál.

5.2 PSNR

Špičkový poměr signálu k šumu (Peak Signal to Noise Ratio) vyjadřuje poměr mezi maximální možnou energií signálu a energií šumu. Vstupem je MSE, pouze upravuje měřítko do jiného vyjádření. Protože s každým dalším zvětšením kanálu o 1 bit se dynamický rozsah zdvojnásobí, výsledek je vyjádřen logaritmickými hodnotami v dB. Při 8 bitech na kanál jsou hodnoty obvykle mezi 30 a 50 dB, čím je hodnota vyšší, tím je obraz kvalitnější. Pokud jsou obrazy shodné, není PSNR definované, jelikož MSE je nulové. PSNR se dá získat následovně:

$$\text{PSNR} = 10\log_{10}\frac{(2^B-1)^2}{MSE} \tag{5.2}$$

B zde představuje maximální možnou hodnotu pixelu, která je u 24-bitového obrazu se 3 kanály $2^{24/3} = 255$. U RGB modelu lze celkové PSNR vypočítat z průměrného MSE u 3 kanálů, u YCbCr se počítá pro každý kanál samostatně. Pokud jsou na vstupu dva identické obrazy a hodnota MSE je nulová, není výsledek definovaný. [19].

5.3 SSIM – Structural Similarity

Předchozí metody neuvažují psychovizuální model lidského vidění, tím se zabývá až metoda SSIM. Obecný předpoklad je, že obrazy jsou vysoce stukturované a mezi pixely existuje závislost, obzvláště mezi sousedními, a tato závislost nese informace o struktuře obrazu. Běžně se dělí strukturní informace na jas (luminanci), kontrast a vlastní strukturu.

Nejprve se vyhodnotí hodnoty jasu obou signálů. Pokud se jedná o diskrétní signály, můžeme je odhadnout pomocí střední intenzity:

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{5.3}$$

Funkce porovnání jasu $l(x, y)$ je potom funkcí μ_x a μ_y. Jelikož známe hodnotu jasu, můžeme jej ze signálu odstranit. Pomocí směrodatné odchylky odhadneme kontrast následujícím způsobem:

$$\sigma_x = \left(\frac{1}{N-1}\sum_{i=1}^{N}(x_i-\mu_x)^2\right)^{\frac{1}{2}} \tag{5.4}$$

Poté je signál normalizován o svou vlastní směrodatnou odchylku, a tak získáme informace o struktuře obrazů. Porovnání struktury $s(x, y)$ se provede na normalizovaných signálech $\frac{(x-\mu_x)}{\sigma_x}$ a $\frac{(y-\mu_y)}{\sigma_y}$. Nakonec kombinací všech 3 složek získáme celkové srovnání:

$$S(x, y) = f(l(x, y), c(x, y), s(x, y)) \tag{5.5}$$

Ze vztahů pro porovnání jasu, kontrastu a strukturních informací odvodíme vzorec pro výpočet SSIM:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{5.6}$$

Protože počítat SSIM pro celý obraz je výpočetně náročné, provádí se výpočet nad oknem zpravidla o rozměrech 8×8 pixelů, které se posouvá po obrazu po jednom pixelu. Celkový SSIM index, MSSIM (meanSSIM – průměrný SSIM), se pak určí jako průměr z dílčích výsledků:

$$\text{MSSIM}(x, y) = \frac{1}{M}\sum_{j=1}^{M}\text{SSIM}(x_j, y_j) \tag{5.7}$$

kde X a Y představují referenční, respektive komprimovaný obraz, x_j a y_j obsah j-tého okna obrazu. [23].

(a) lena (b) baboon (c) plane

(d) aiguille (e) lake (f) comet

(g) blackhole (h) castle

Obrázek 5.1: Testovací obrázky

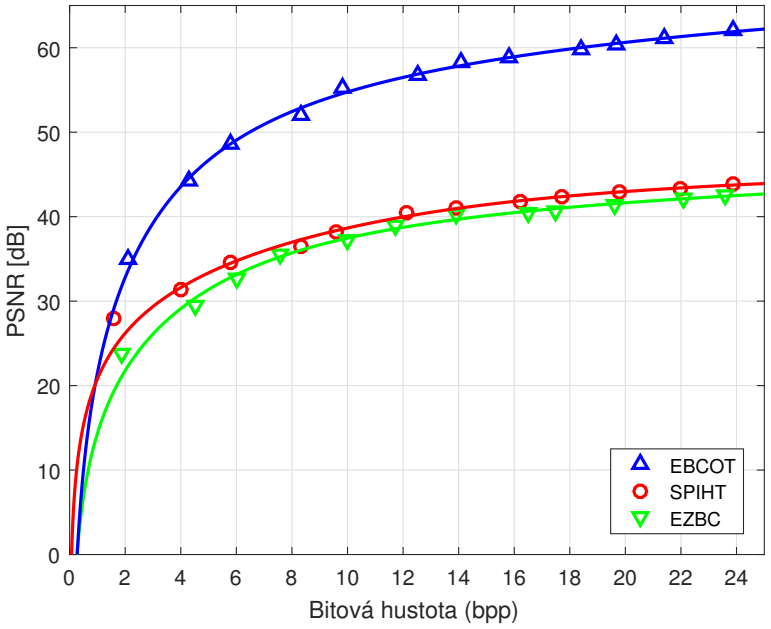
5.4 Testovací vzorky

Pro možnost srovnání s odbornými pracemi v oboru komprese obrazu jsou součástí testovací množiny běžně používané testovací obrazy *lena*, *baboon* a *plane* o rozměrech 512x512 bodů. K ním bylo přidáno dalších 5 obrazů - *aiguille* pro otestování hran,*lake* a *comet* s velkými plochami bez vysokých frekvencí, *blackhole* jako ilustrace s mnoha bílými body na pozadí a *castle* jako příklad s atypickými rozměry obrazu. Obraz *castle* má rozměry 579x377, zbývající 4 obrazy pak shodně 1200x800 bodů.

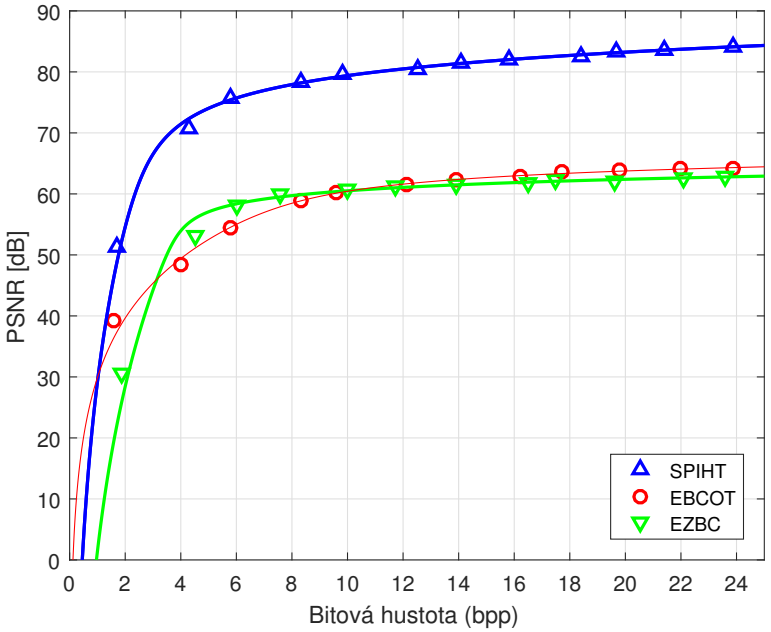
5.5 Kvalita obrazu při stejné hodnotě bpp

Jedním z provedených srovnání algoritmů pro zakódování koeficientů je zkoumání, jak si poradí se zadanou bitovou hloubkou obrazových dat.

Z výsledků je patrné, nejlepší kvality dosahuje algoritmus SPIHT, čemuž odpovídá i lepší efektivita komprese. Rozdíl mezi SPIHT a EBCOT je 10-20 dB u PSNR a u SSIM 20-30%, přičemž hodnoty PSNR jsou u všech testovacích obrázků o jednotky dB vyšší u EBCOT než u EZBC. U některých obrázků (aiguille, comet, lena) byly hodnoty SSIM vyšší u EZBC než u EBCOT, rozdíl u všech obrázků je v jednotkách %.



Obrázek 5.2: Hodnoty PSNR při stejném bpp



Obrázek 5.3: Hodnoty SSIM při stejném bpp

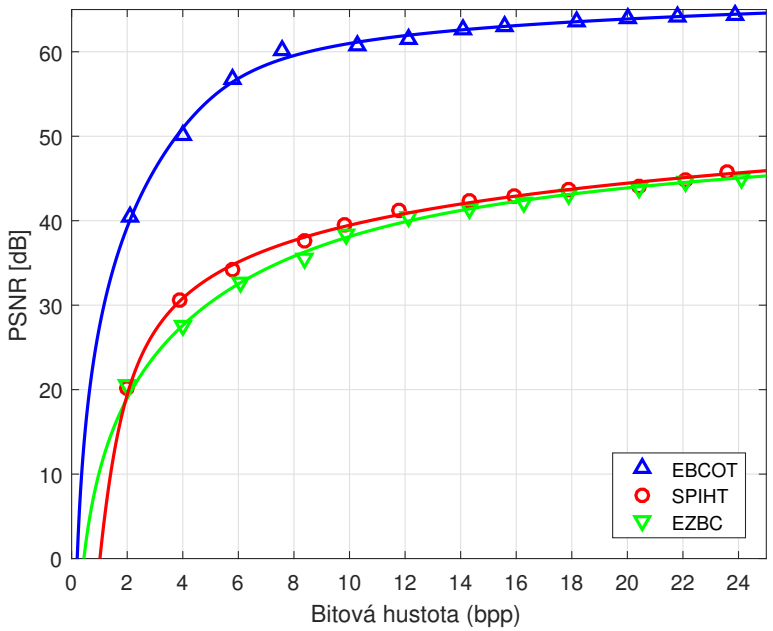
5.6 Kvalita obrazu při stejných velikostech bitového toku

V této podkapitole budou prodiskutovány výsledky porovnání algoritmů při co nejvíce podobném počtu bitů výsledného bitového toku. Jelikož je předmětem srovnání kódování koeficientů stavějící na různých principech, není možné zaručit naprosto stejnou velikost u

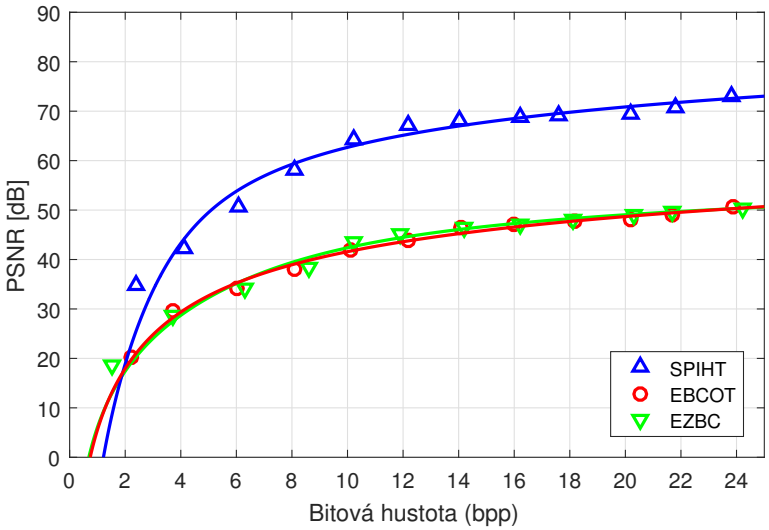
obou algoritmů. Je nutné stanovit hodnotu o kolik se můžou velikosti bitových toků lišit - pro prováděné experimenty byla stanovena 2% odchylka. Velikosti byly určeny jako 1-, 2-, 5-násobky řádů kB.

Výsledky podle očekávání korespondují s efektivitou kódování koeficientů, kdy algoritmus, který pro uchování stejného počtu koeficientů potřebuje méně bitů, dokáže na menší počet bitů zakódovat více obrazových informací.

Hodnoty PSNR i SSIM jsou tedy u kódování koeficientů pomocí SPIHT nejvyšší, a to o několik desítek dB, resp. procent. Následují hodnoty PSNR a SSIM algoritmů EZBC a EBCOT, kdy u většiny obrázků je lepším algoritmem EBCOT, u třech z nich (lake, aiguille, comet) však bylo dosaženo lepší kvality kódování obrazu algortimem EZBC. Kompletní výsledky jsou v grafech 5.4 a 5.5.



Obrázek 5.4: Hodnoty PSNR při stejné velikosti výstupního toku



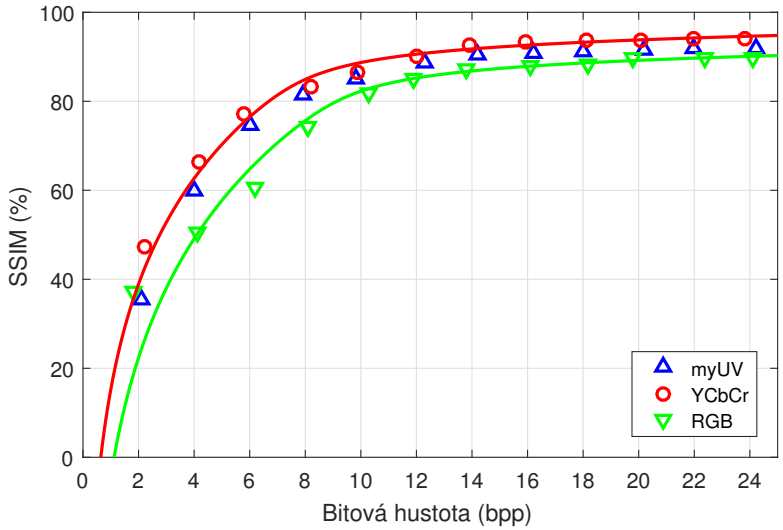
Obrázek 5.5: Hodnoty SSIM při stejné velikosti výstupního toku

Běžný objem ušetřených dat pro uchování obrazu je kolem 10%, v některých případech lze tedy u algoritmu SPIHT aritmetické kódování vynechat, tím se dosáhne větší rychlosti komprese na úkor nepatrného zvětšení výsledného bitového toku. Rozdíl mezi ostatními dvěma algoritmy je minimální, v jendotkách dB a %. EBCOT je jedním z algoritmů, který do značné míry spoléhá na dodatečné použití aritmetického kodéru pro zhuštění informace, a tomu uzpůsobuje svůj výstup při kódování koeficientů.

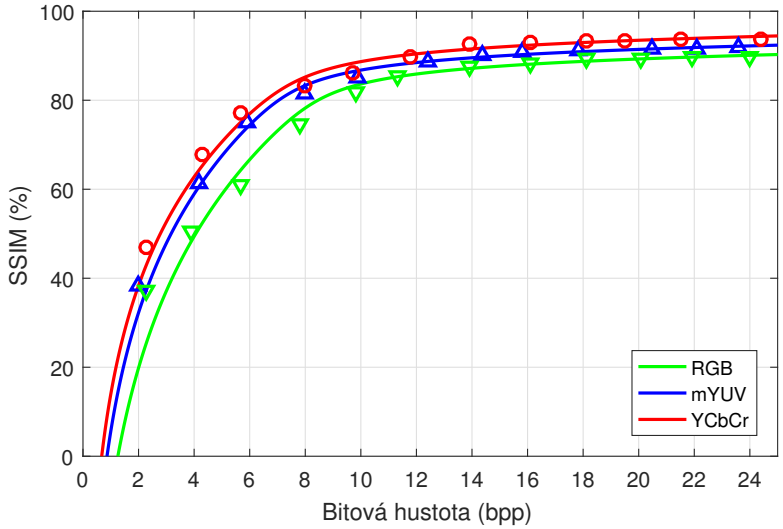
5.7 Barevné prostory

Dále byl proveden test efektivity komprese 3 používaných barevných prostorů - RGB, YC_bC_r a mYUV. Standard JPEG 2000 používá pro reverzibilní transformaci (kterou lze provést celočíselnými operacemi) prostor mYUV, pro nereverzibilní transformaci prostor YC_bC_r.

Podle výsledků srovnání je nejlepší variantou prostor YC_bC_r s nereverzibilní transformací, následuje barevný prostor mYUV pro reverzibilní transformaci a nejhorší efektivity je dosaženo při použití barevného prostoru RGB. Toto pořadí je nezávislé na algoritmu, vyskytuje se jak u kódování EBCOT, tak i u SPIHT a EZBC.



Obrázek 5.6: Hodnoty SSIM algoritmu EBCOT v závislosti na barevném prostoru



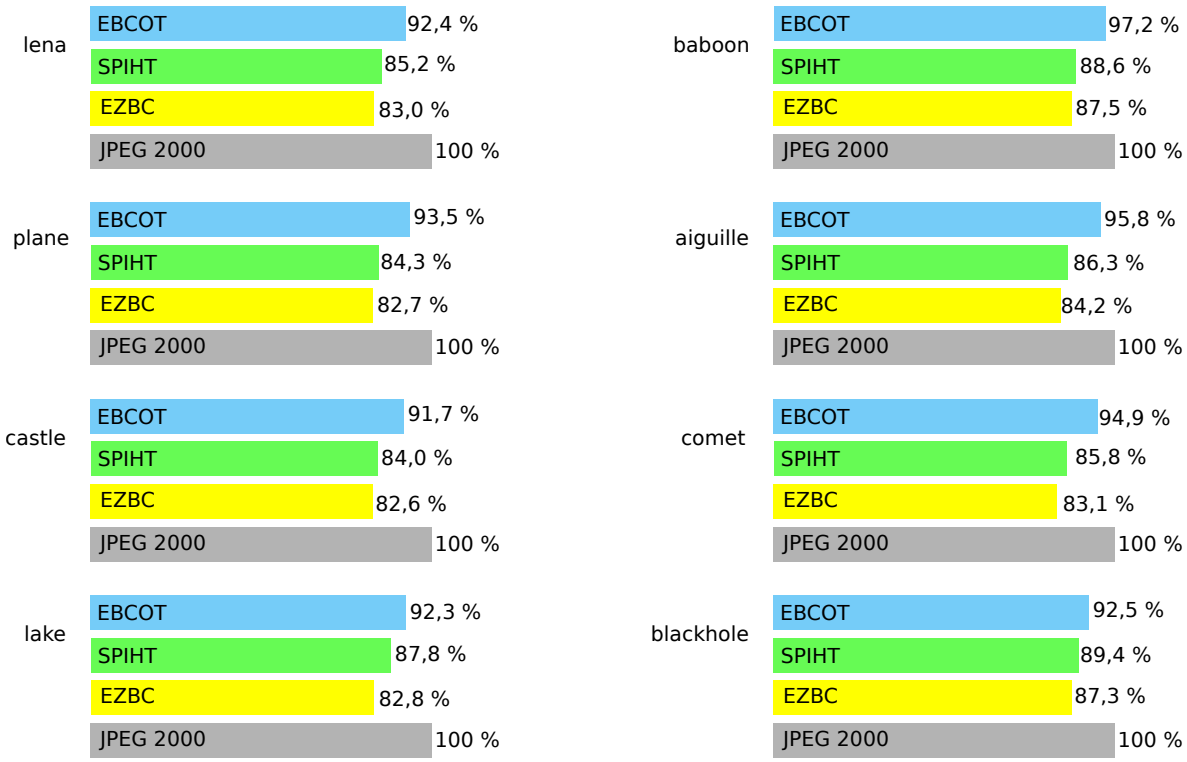
Obrázek 5.7: Hodnoty SSIM algoritmu SPIHT v závislosti na barevném prostoru

Pro nejlepší výsledky byl pro další srovnání zvolen pouze barevný prostor YC_bC_r s nereverzibilní transformací.

5.8 Rychlost komprese

Významnou vlastností algoritmů je rychlost převodu z jiného obrazového formátu do formátu JPEG 2000. Lze měřit čas v různých fázích komprese, seskupit více fází dohromady atp. V rámci objektivity bylo hlavním předmětem zkoumání doba trvání zakódování koeficientů u algoritmů EBCOT, SPIHT a EZBC a také u referenční komprese z knihovny OpenJPEG podle standardu JPEG 2000.

EZBC díky své jednoduchosti je nejrychlejším testovaným algoritmem, pro kompresi potřeboval nejméně času. Ze vzájemného porovnání ostatních dvou testovaných algoritmů vychází nepatrně lépe SPIHT, všechny implementované postupy jsou pak znatelně lepší než referenční algoritmus.



Obrázek 5.8: Doba potřebná pro konverzi

5.9 Efektivita aritmetického kodéru

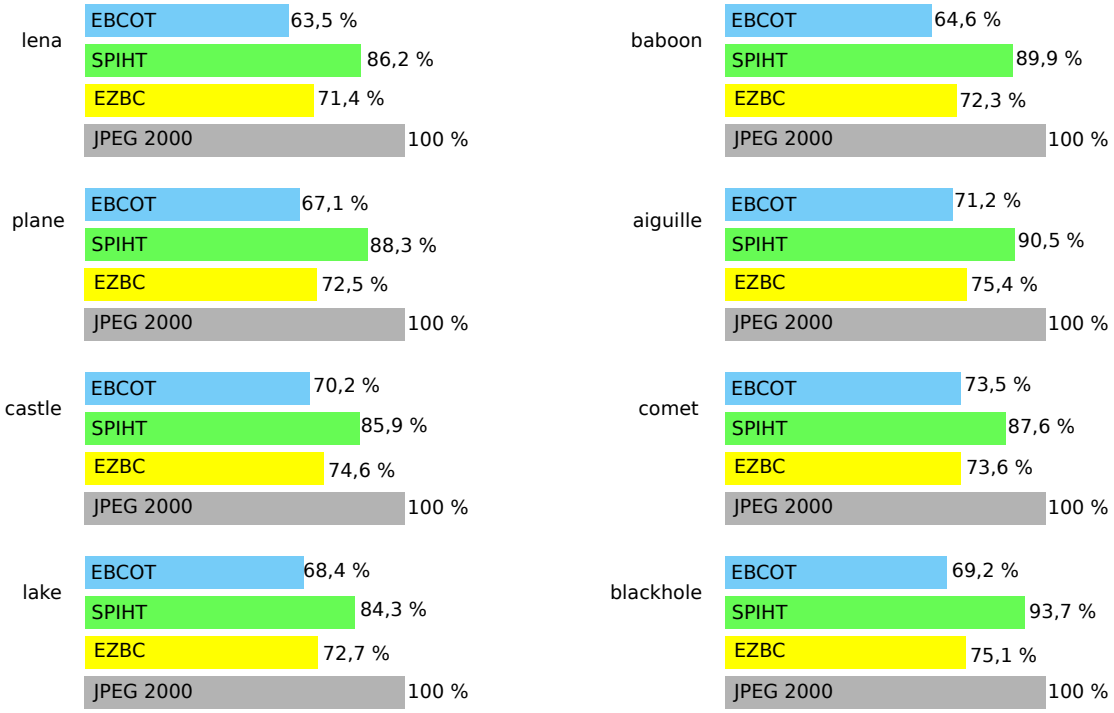
Výstup obou algoritmů je standardně podroben aritmetickému kodéru pro ještě lepší stupeň komprese a maximalizace efektivity uložení informace. Mimo toto klasické srovnání bylo provedeno porovnání výsledných bitových toků bez využití aritmetického kodéru, a to jak mezi jednotlivými algoritmy navzájem, tak i stanovení poměrů toků při nepoužití versus při použití aritmetického kódování u jednoho specifického algoritmu.

EBCOT je jedním z algoritmů, který do značné míry spoléhá na dodatečné použití aritmetického kodéru pro zhuštění informace, a tomu uzpůsobuje svůj výstup při kódování koeficientů. Podle dosažených výsledků v provedených srovnáních může být při praktickém použití až 30-40% bitů potřebných pro uložení obrazových dat oproti běhu kódování EBCOT bez následného použití aritmetického kodéru. Největší úspora dat byla zjištěna u obrázku plane, a to 39%.

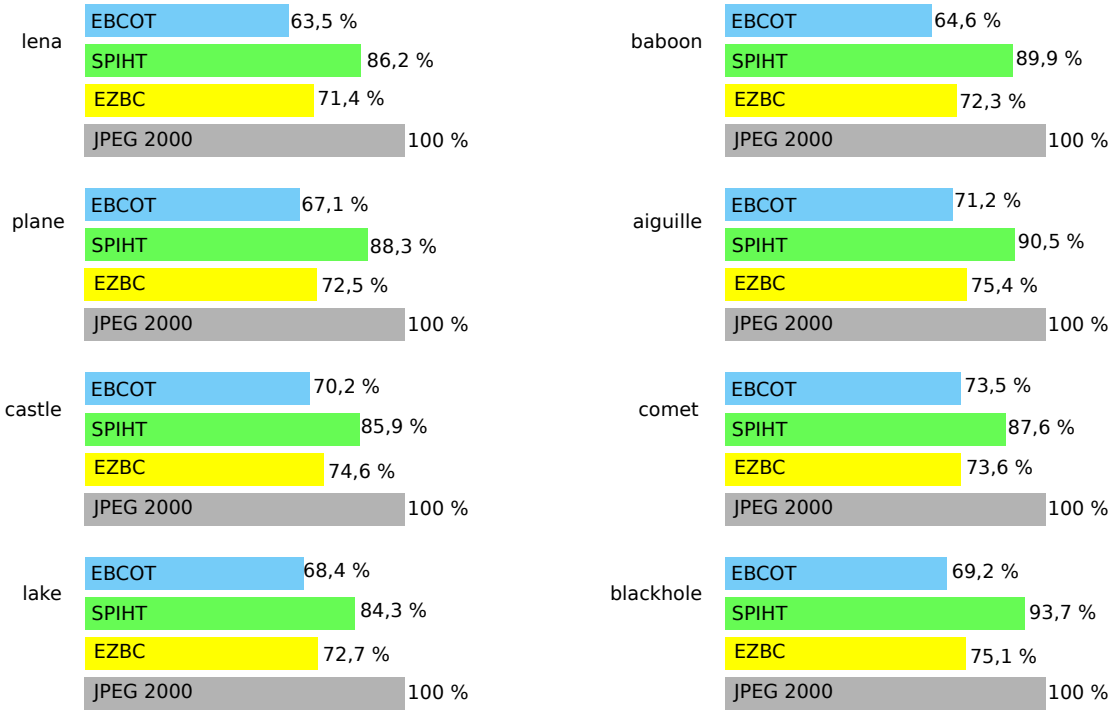
Naproti tomu SPIHT produkuje výstupní bitový tok, který je sám o sobě dostatečně informačně zhuštěný, a jeho následující průchod skrze aritmetický kodér už výrazné zlepšení kompresního poměru nenabízí. Běžný objem ušetřených dat pro uchování obrazu je kolem 10%, v některýh případech lze tedy u algoritmu SPIHT aritmetické kódování vynechat,

tím se dosáhne větší rychlosti komprese na úkor nepatrného zvětšení výsledného bitového toku.

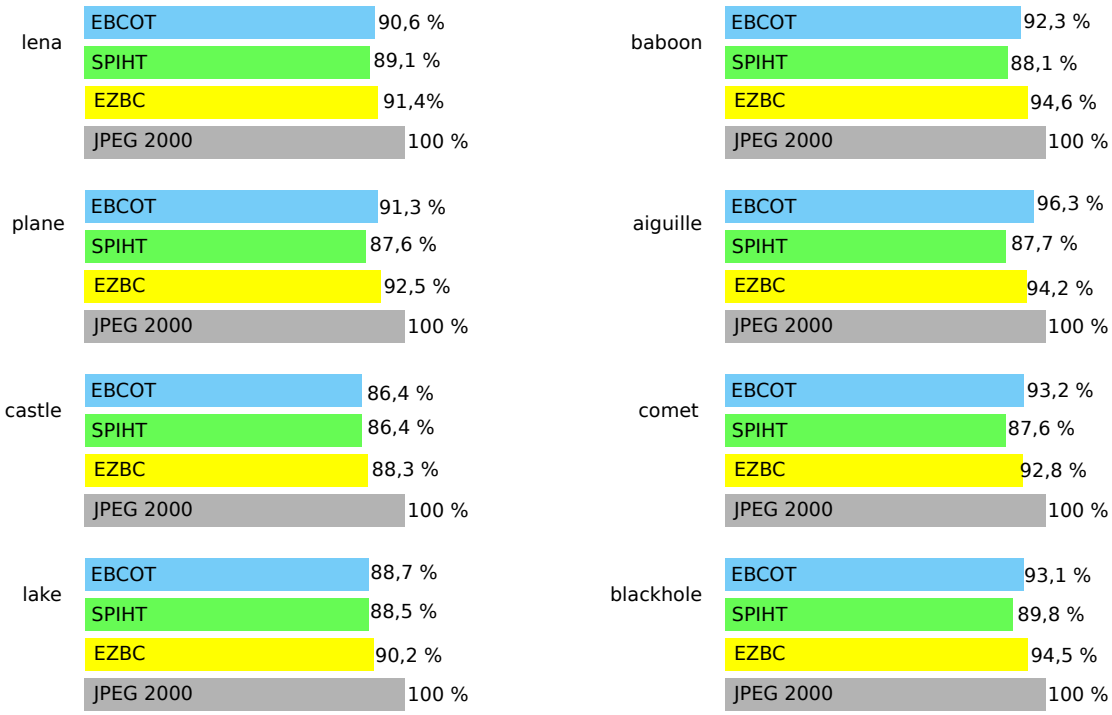
Výsledky algoritmu EZBC jsou podobné výsledkům EBCOT, rozdíl mezi kódováním s a bez použití aritmetického kodéru je však nižší, činí 15-20 %. Komprese koeficientů EZBC byla v některých případech úspěšnější nebo srovnatelná s kompresí pomocí EBCOT, u většiny obrázků však EBCOT dosahoval lepších výsledků v řádu jednotek %.



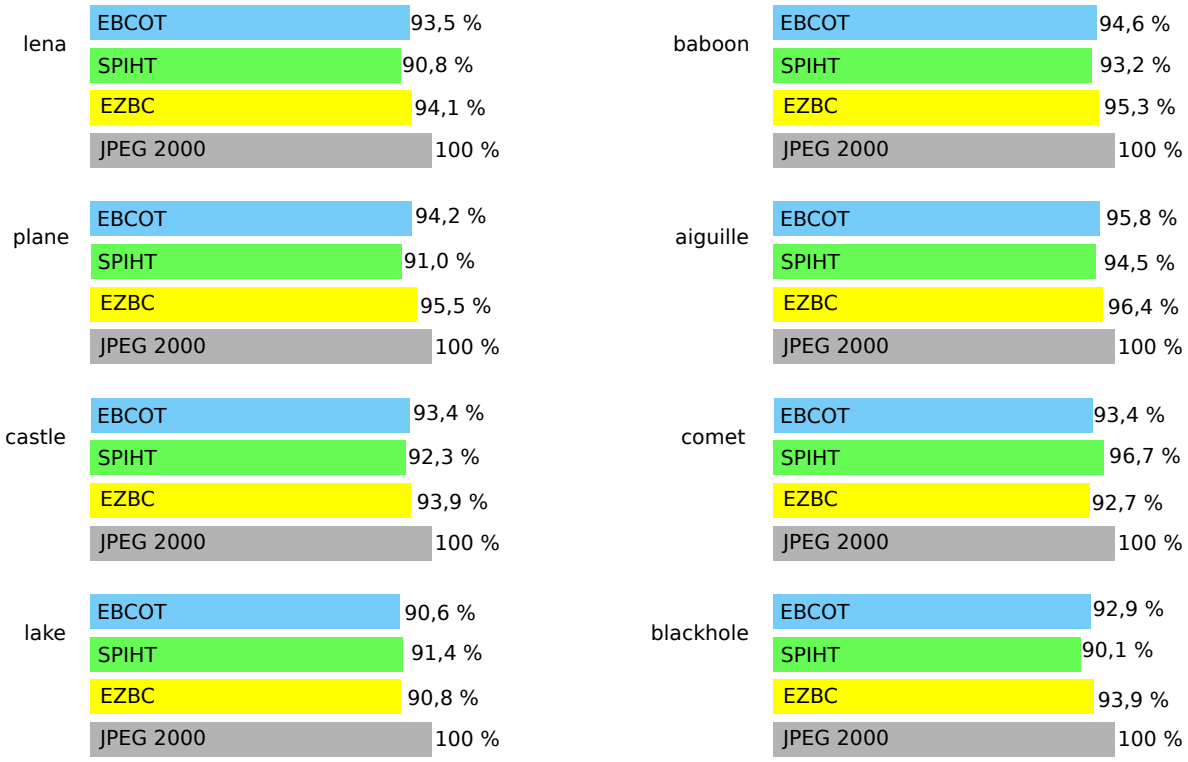
Obrázek 5.9: Velikosti výstupu při použití AC kodéru při stejném PSNR



Obrázek 5.10: Velikosti výstupu při použití AC kodéru při stejném SSIM



Obrázek 5.11: Velikosti výstupu při nepoužití AC kodéru při stejném PSNR



Obrázek 5.12: Velikosti výstupu při nepoužití AC kodéru při stejném SSIM

5.10 Souhrn srovnání

Testem barevných prostorů bylo prokázáno, že nejlepší kompresi umožňuje barevný prostor YC_bC_r, následovaný mYUV a nejhorším byl prostor RGB. Z výsledků vyplývá, že nereverzibilní transformace dosahuje lepších kompresních poměrů než reverzibilní, a to o 1-3 dB u PSNR a 2-3 % u SSIM. Rozdíl mezi prostorem mYUV a RGB je pak ještě větší.

Ze všech provedených srovnání vychází algoritmus SPIHT lépe než EBCOT a EZBC, toto může být v některých případech dáno implementací. Při stejných hodnotách bpp je kvalita obrazu při kódování SPIHT lepší o 12-20 dB u PSNR, 20-26 % u SSIM. Pokud je prioritou co nejvyšší kvalita při dané velikosti, opět je lepším postupem SPIHT, jeho hodnoty PSNR jsou o 11-25 dB vyšší než u EBCOT, podobně i hodnoty SSIM jsou o 19-29 % vyšší.

EZBC dosahoval podobných výsledků jako EBCOT, rozdíl mezi těmto dvěma postupy je minimální. Na většině testovacích obrázků vychází lépe EBCOT, v některých případech však byl horší než EZBC. Pokud porovnáme kvalitu obrazu při stejném bpp, je mezi oběma algoritmy rozdíl do 3 dB u PSNR a do 5 % u SSIM. Při stejné výstupní velikosti jsou pak hodnoty od sebe vzdáleny do 3 dB a do 3 %.

Měřením doby potřebné pro kompresi bylo zjištěno, že všechny testované algoritmy překonávají referenční kódování reprezentované knihovnou OpenJPEG, přičemž nejrychlejší komprese dosahoval EZBC, následovaný algoritmem SPIHT, který potřeboval až o 5 % více času. EBCOT je na třetím místě s rozdílem maximálně 10%, posledním je pak referenční algoritmus s o 3-8 % delší dobou pro zakódování koeficientů.

Jako poslední byl otestován vliv aritmetického kódování na výsledný objem dat. U kódování algoritmem SPIHT je účinek MQ kodéru relativně malý, snižuje objem dat asi o 5%. Lze jej vynechat a snížit dobu potřebnou pro kompresi za cenu zvýšeného objemu dat. Naproti tomu výstupy algortmů EBCOT a EZBC nejsou zhuštěné, jako je tomu u SPIHT, a účinnost aritmetického kódování je vyšší, velikost výsledného bitového toku je až o 40 % menší u kódování EBCOT a o 22 % u EZBC.

Kapitola 6

Závěr

Hlavní obsah této práce je rozmístěn do čtyř kapitol, které se zabývají postupně teoretickými základy komprese JPEG 2000, jednotlivými algoritmy pro kódování koeficientů, jejich implementací a srovnáním. V první kapitole je stručně popsán postup pro kompresi obrazu podle JPEG 2000, poté jsou rozebrány dvě její fáze - diskrétní vlnková transformace a kvantizace. Je zde také popsána teorie vlnek a jejich typy používaných při ztrátové kompresi obrazu.

Ve druhé kapitole jsou podrobně představené vybrané algoritmy pro kompresi koeficientů vzniklých diskrétní vlnkovou transformací, EZW a SPIHT, MRWD, EBCOT, SLCCA a EZBC. Na začátku této části jsou tyto a další vybrané kódovací postupy sdruženy do rodin podle svých vlastností a zařazeny do kontextu pomocí vývojového stromu, v kterém lze vysledovat postupný vývoj zpracovávaných algoritmů. Dále je zde umístěn i popis entropického kódování a aritmetického kódování, které se používá u JPEG 2000.

V následující části je uveden popis implementace algoritmů EBCOT, SPIHT a EZBC, a to jak ze statického pohledu na sémantiku proměnných, tak i dynamicky ve formě grafu volání funkcí jednotlivých algoritmů a vysvětlení prováděných operací.

Obsahem poslední kapitoly je srovnání těchto algoritmů na testovací množině obrázků pomocí metod pro objektivní hodnocení obrazu PSNR a SSIM z hlediska efektivity, vlivu aritmetického kodéru na velikost výstupu, úrovně kvality při požadované velikosti a rychlosti komprese koeficientů do výsledného bitového toku. Kromě velikosti výstupu při vynechání aritmetického kódování, kdy byl nejlepším postupem EBCOT, a rychlosti komprese, kde první místo získal EZBC, byl podle výsledků testů na zhotovené implementaci nejlepším algoritmem SPIHT. Kódování EZBC dosahovalo podobných výsledků jako EBCOT, na některých testovacích vzorcích bylo dokonce lepší. Byl otestován i vliv barevného prostoru zdrojového obrazu na kvalitu obrazu, jako nejvýhodnější se jeví barevný prostor YC_bC_r, který se používá pro nereverzibilní transformaci, následovaný mYUV využívaný při reverzibilní transformaci. Nejhorším z trojice byl podle očekávání shledán barevný prostor RGB.

Jednou z možností. jak rozšířit tuto práci, je provést další srovnání algoritmů kódování koeficientů, ať už z více hledisek, nebo přidáním implementací dalších algoritmů a jejich variant. Jelikož se oblast komprese koeficientů stále vyvíjí, nemusíme se omezovat na stávající postupy kódování a můžeme vymyslet i nové neotřelé, nebo kombinující současné způsoby uchování koeficientů. Prostor pro vylepšení skýtají i samotná kódování, u kterých lze některé jejich části optimalizovat pro vyšší rychlost, nebo přizpůsobit jejich strukturu paralelnímu zpracování oblastí koeficientů. Další cestou je vytvoření programu, který provede požadované testy a jejich výsledky zobrazí přímo do grafického znázornění nebo tabulek.

Seznam obrázků

Literatura

[1] ABOUFADEL, E.: *JPEG 2000: The Next Compression Standard using wavelet technology*. [online], [vid. 02.02.2017].
URL http://faculty.gvsu.edu/aboufade/web/wavelets/student_work/EF/

[2] ADDISON, P. S.: *The Illustrated Wavelet Transform Handbook*. CRC Press, 2002, ISBN 978-0750306928, 368 s.

[3] BAŘINA, D.: *Biortogonální spline vlnka 2.2* . [online], 2010, [vid. 11.02.2017].
URL https://en.wikipedia.org/wiki/File:Wavelet_Bior2.2.svg

[4] BURGOS, R. B.; SANTOS, M. A. C.; e SILVA, R. R.: Deslauriers–Dubuc interpolating wavelet beam finite element. *Finite Elements in Analysis & Design*, ročník 75, 11 2013: s. 71–77, ISSN 0168-874X.

[5] CHAI, B.-B.; VASS, J.; ZHUANG, X.: Significance-linked connected component analysis for wavelet image coding. *IEEE Transactions on Image Processing*, ročník 8, č. 6, Jun 1999: s. 774–784, ISSN 1057-7149, doi:10.1109/83.766856.

[6] COHEN, A.; DAUBECHIES, I.; FEAUVEAU, J.-C.: Biorthogonal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, ročník 45, č. 5, 1992: s. 485–560, ISSN 1097-0312, doi:10.1002/cpa.3160450502.

[7] DELAUNAY, X.: *EBCOT coding passes explained on a detailed example*. [online], 2001, [vid. 02.02.2017].
URL http://d.xav.free.fr/ebcot/EBCOT_example.pdf

[8] FINKEL, R. A.; BENTLEY, J. L.: Quad trees a data structure for retrieval on composite keys. *Acta Informatica*, ročník 4, č. 1, 1974: s. 1–9, ISSN 1432-0525.

[9] HSIANG, S.-T.: Embedded image coding using zeroblocks of subband/wavelet coefficients and context modeling. In *Proceedings DCC 2001. Data Compression Conference*, 2001, ISSN 1068-0314, s. 83–92.

[10] KISKA, T.: *Obrazové kodeky založené na vlnkové transformaci*. Bakalářská práce, Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, Brno, 2013, 72 s. Vedoucí práce Mgr. Pavel Rajmic, Ph.D.

[11] MALLAT, S.: *A wavelet tour of signal processing: the sparse way*. 3rd ed. Boston : Amsterdam : Academic Press ; Elsevier, 2009, ISBN 978-0-12-374370-1, 805 s.

[12] NEETHU, J.; AMITHA, V.; V., S.; aj.: Analysis of Various Color Space Models on Effective Single Image Super Resolution. *Intelligent Systems Technologies and Applications*, ročník 384, 2016: s. 529–540.

[13] PŘIKRYL, L.: *Komprese obrazu pomocí vlnkové transformace*. Bakalářská práce, Vysoké učení technické v Brně, Fakulta informačních technologií, Brno, 2013, 48 s. Vedoucí práce Ing. David Bařina.

[14] RYŠAVÝ, L.: *Komprese obrazu pomocí vlnkové transformace*. Bakalářská práce, Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Brno, 2009, 62 s. Vedoucí práce doc. Ing. Eva Gescheidtová.

[15] SAID, A.: *Example of Application for Image Compression* . [online], 1999, [vid. 11.02.2017].
URL http://www.cipr.rpi.edu/~pearlman/papers/ex_spiht-ezw.pdf

[16] SAID, A.; PEARLMAN, W. A.: A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, ročník 6, č. 3, 6 1996: s. 243–250, ISSN 1051-8215.

[17] SERVETTO, S.; RAMCHANDRAN, K.; ORCHARD, M.: Morphological representation of wavelet data for image coding. In *International Conference on Acoustics, Speech, and Signal Processing*, ročník 4, 1995, s. 2229–2232.

[18] SHAPIRO, J. M.: Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, ročník 41, č. 12, 12 1993: s. 3445–3462, ISSN 1053-587X.

[19] SOLOMON, D.: *Data Compression : the complete reference*. London: Springer, Čtvrté vydání, 2007, ISBN 978-1-84628-602-5, 1092 s.

[20] TAUBMAN, D.: High performance scalable image compression with EBCOT. *IEEE Transactions on Image Processing*, ročník 9, č. 7, 7 2000: s. 1158–1170, ISSN 1057-7149.

[21] URBÁNEK, P.: *Komprese obrazu pomocí vlnkové transformace*. Diplomová práce, Vysoké učení technické v Brně, Fakulta informačních technologií, Brno, 2013, 75 s. Vedoucí práce Ing. David Bařina.

[22] WALKER, J. S.; NGUYEN, T. Q.: Adaptive scanning methods for wavelet difference reduction in lossy image compression. In *International Conference on Image Processing*, ročník 3, 9 2000, ISSN 1522-4880, s. 182–185.

[23] WANG, Z.; BOVIK, A. C.; SHEIKH, H. R.; aj.: Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, ročník 13, č. 4, 2004: s. 600–612, ISSN 1057-7149, doi:10.1109/TIP.2003.819861.

[24] ITU-R T.871: *Digital compression and coding of continuous-tone still images: JPEG File Interchange Format (JFIF)* . ITU, 2011.

[25] ITU-R BT.601-5: *Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios* . ITU, 2007.