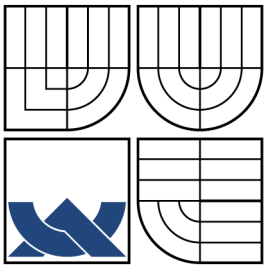
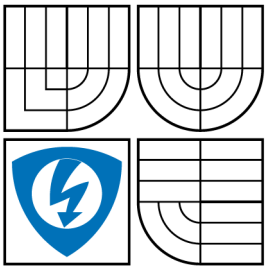


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATION

EFEKTIVNÍ NÁSTROJ PRO KOMPRESI OBRAZU V JAZYCE JAVA

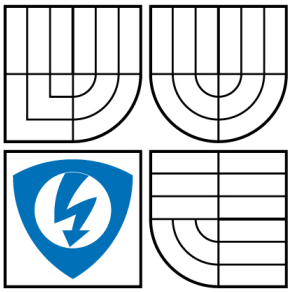
JAVA-BASED EFFECTIVE IMPLEMENTATION OF AN IMAGE COMPRESSION TOOL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — BC. ZDENĚK PRŮŠA
AUTHOR

VEDOUCÍ PRÁCE — ING. JAN MALÝ
SUPERVISOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor

Telekomunikační a informační technika

Student: Průša Zdeněk Bc. ***ID:*** 89287

Ročník: 2 ***Akademický rok:*** 2007/2008

NÁZEV TÉMATU:

Efektivní nástroj pro kompresi obrazu v jazyce Java

POKYNY PRO VYPRACOVÁNÍ:

Vytvořte funkční implementaci nástroje pro ztrátovou kompresi obrazu v jazyce Java. Nástroj bude využívat diskrétní vlnkovou transformaci pro rozklad obrazu do koeficientů a SPIHT kódovací schéma pro samotnou kompresi. Bude efektivně pracovat s barevnou informací a využívat akcelerovaných postupů pro výpočty (fast lifting scheme).

DOPORUČENÁ LITERATURA:

[1] Jansen A., Cour-Harbo A.: Ripples in Mathematics: The Discrete Wavelet Transform, Springer, 2001. ISBN 3-540-41662-5.

[2] Said A., Pearlman W.A.: A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees, IEEE Trans. on Circuits and Systems for Video Technology, vol. 6., pp. 243-250, June 1996, dostupné z: http://www.cipr.rpi.edu/staff/pearlman.html/papers/csvt96_sp.pdf

[3] Kassim A. A., Lee W. S.: Embedded Color Image Coding Using SPIHT With Partially Linked Spatial Orientation Tree, IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, pp. 203-206, 2003.

[4] Lambertová, P.: Implementace SPIHT kodeku v jazyce Java, diplomová práce, 2007, VUT Brno, Fakulta elektrotechniky a komunikačních technologií. Vedoucí práce Ing. Jan Malý.

Termín zadání: 11.2.2008 ***Termín odevzdání:*** 28.5.2008

Vedoucí práce: Ing. Jan Malý

prof. Ing. Kamil Vrba, CSc.

předseda oborové rady

UPOZORNĚNÍ:

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení:	Zdeněk Průša
Bytem:	Dynín 11, 37364
Narozen/a (datum a místo):	24.08.1984, České Budějovice

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií

se sídlem Údolní 244/53, 602 00, Brno

jejímž jménem jedná na základě písemného pověření děkanem fakulty:

Prof. Ing. Kamil Vrba, CSc.

(dále jen „nabyvatel“)

Čl. 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - □ disertační práce
   - □ diplomová práce
   - □ bakalářská práce
   - □ jiná práce, jejíž druh je specifikován jako .....

   (dále jen VŠKP nebo dílo)

Název VŠKP:	Efektivní nástroj pro kompresi obrazu v jazyce Java
Vedoucí/ školitel VŠKP:	Ing. Jan Malý
Ústav:	Ústav telekomunikací
Datum obhajoby VŠKP:	

VŠKP odevzdal autor nabyvateli v*:

- □ tištěné formě – počet exemplářů 2
- □ elektronické formě – počet exemplářů 1

* hodící se zaškrtněte

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - □ ihned po uzavření této smlouvy
   - □ 1 rok po uzavření této smlouvy
   - □ 3 roky po uzavření této smlouvy
   - □ 5 let po uzavření této smlouvy
   - □ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: …….

…….. ……

Nabyvatel Autor

ABSTRAKT

Tato diplomová práce se zabývá problematikou ztrátové komprese digitálních obrazových dat. Ztrátová komprese obecně zavádí určité zkreslení do výsledné reprezentace obrazu. Toto zkreslení by mělo být nerušivé nebo v lepším případě nepozorovatelné. K analýze obrazových dat se používá proces transformace a k vybrání relevantních údajů proces kódování. Hodnocení kvality rekonstruovaného obrazu může být podle prostředků prováděno objektivně nebo subjektivně. V této práci je představen a realizován kodér obrazu založený na dvojrozměrné vlnkové transformaci a SPIHT algoritmu kódování koeficientů. Bylo použito akcelerovaných postupů výpočtu vlnkové transformace pomocí lifting schématu. Kodér efektivně pracuje s barevnou informací obrazů pomocí modifikace původního SPIHT algoritmu. K vlastní realizaci byl použit programovací jazyk JAVA. Návrh byl proveden podle zásad objektového programování a je proto snadno modifikovatelný. Na demonstrovaných příkladech je možno sledovat účinnost a charakteristický způsob zkreslení navrženého kodéru při vysokých kompresních poměrech.

KLÍČOVÁ SLOVA

ztrátová komprese, vlnková transformace, lifting schéma, JAVA, metoda SPIHT, CSPIHT

ABSTRACT

This diploma thesis deals with digital image lossy compression. Lossy compression in general inserts some kind of distorsion to the resulting image. The distorsion shouldn't be interupting or even noticable in the better case. For image analysis there is used process called transformation and for choosing relevant coefficients process called coding.

Evaluation of image quallity can be done by objective or subjective method. There is encoder introduced and realized in this work. Encoder utilizes two-dimension wavelet transform and SPIHT algortihm for coefficient coding. It was made use of accelerated method of wavelet transform computation by lifting scheme. Coder can proccess color information of images using modificated original SPIHT algorithm. For implementation the JAVA programming language was employed. The object-oriented design principes was made use of and thus the program is easy to extended. At demonstaration pictures there are shown effectiveness and characteristic way of distorsion of the proposed coder at high compression rates.

KEYWORDS

lossy image compression, wavelet transform, lifting scheme, JAVA, SPIHT,CSPIHT method

PRŮŠA, Z. *Efektivní nástroj pro kompresi obrazu v jazyce Java*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 68 s. Vedoucí diplomové práce Ing. Jan Malý.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Efektivní nástroj pro kompresi obrazu v jazyce Java“ jsem vypracoval samostatně pod vedením vedoucího diplomové 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 autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Tímto bych rád poděkoval vedoucímu diplomové práce panu Ing. Janu Malému za ochotu, vstřícné jednání a za užitečnou metodickou pomoc.

V Brně dne ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

1 ÚVOD

Komprese statických obrazů je proces, ve kterém je množství dat potřebných pro reprezentaci obrazu podle možností zmenšeno. Kvalita rekonstruovaného obrazu musí vyhovovat dané aplikaci a zároveň musí být aplikace schopna zvládnout výpočetní složitost komprese.

Potřeba vyjádření obrazu menším objemem dat je naprosto logická. Ušetří se paměť, popř. prostředky potřebné k transportu dat. Jako příklad by mohla stačit jednoduchá úvaha: Vezměme si obraz s rozlišením 2816x2112 [1] bodů a každý bod bude vyjádřen třemi barevnými složkami po jednom byte. Celý obrázek pak bude zabírat 2816x2112x3 = 17842176 bytů. Pokud ovšem zavedeme kompresi, výsledná reprezentace obrazu může být 5x, 10x, 20x ... menší.

Způsoby komprese, které vznikly, by se v základu daly rozdělit na bezeztrátové a ztrátové.

Tato práce se zabývá ztrátovou kompresní metodou založenou na reprezentaci obrazu v kmitočtově-prostorové rovině (podobně jako JPEG2000). Dále jsou koeficienty této reprezentace velice úsporně kódovány algoritmem SPIHT do výsledné reprezentace požadované délky.

Cílem této diplomové práce je návrh kompresoru, který by řešil pokročilejší implementační problémy. Přesněji práci s barevnou informací a segmenty obrazu. Uvažování segmentů s sebou také přináší potřebu ošetření případu, kdy rozměry obrazu nejsou celočíselným násobkem rozměrů segmentu.

V první části je předvedena hlavní teorie a předpoklady, druhá pak obsahuje popis a testování implementace ztrátového kodéru v jazyce JAVA.

[1] standardní šesti-megapixelová fotografie

2 TEORETICKÝ ÚVOD

2.1 Komprese obrazu

Po digitalizaci analogového obrazu je každý pixel reprezentován pevným počtem bitů. K redukci datového objemu potřebného k digitální reprezentaci obrazu se používá komprese. Jak již bylo zmíněno v úvodu, způsoby komprese se dělí na *bezeztrátové* a *ztrátové*.

Bezeztrátové kompresní metody jsou například: RLE, LZW nebo Huffmanovo kódování. Mohou být použity samostatně nebo současně s některou ztrátovou kompresní metodou. Protože je tato práce zaměřena na ztrátovou kompresi, další text bude zaměřen na ni. Více o bezeztrátové kompresi lze nalézt v [8, 10].

Ztrátové kompresní metody jdou dále a analýzou obrazu se snaží odstranit *redundantní* a *irelevantní* informaci a tím snížit výsledný objem dat. Jako redundantní se označí ta informace, jejíž odstraněním z celkové reprezentace nedojde ke zkreslení obrazu. Redundance v obrazu může být *prostorová* nebo *kódová* [8].

Prostorovou redundancí rozumíme statistickou korelaci mezi pixely v obrazu. Z tohoto poznatku je tedy jasné, že obsah jednoho pixelu může být uhodnut z obsahu sousedního pixelu. Je tedy možné využít diferenční kódování.

Kódová redundance vychází z teorie informace. Zjednodušeně řečeno, symboly se vyskytují každý s určitou pravděpodobností (četností). Předpoklad je, že ne všechny se stejnou. Principem aritmetického a Huffmanova kódování je přiřadit symbolům s nejčastějším výskytem nejkratší bitovou reprezentaci a naopak. Potom řekneme, že kódová redundance byla odstraněna.

Odstraňováním irelevantní informace již zasahujeme do kvality obrazu. Kvalita je relativní pojem a více o ní dále v kapitole 2.1.2. Všechny metody jsou založeny na nedokonalosti lidského oka. (viz. jasové, texturové a frekvenční maskování v [8]). Otázkou zůstává jak analyzovat obraz, aby se ukázalo, které složky jsou nejméně důležité.

Odpovědí jsou transformace obrazu do frekvenční, nebo jiné oblasti. Transformace zpravidla úplně odstraní potřebu znát prostorovou závislot pixelů a umožní rozlišit škálu od nejvyšších detailů po stejnosměrnou složku. Pro obrazy se používají 2D transformace a to Fourierova, diskrétní kosinová a vlnková transformace. Ke ztrátové kompresi dochází právě při kódování těchto koeficientů. Výsledná kvalita je pak ovlivněna tím, kolik a jak moc zkreslených koeficientů je uvažováno ve výsledné reprezentaci.

Na tomto místě lze také uvést současné standardy pracující se ztrátovou kompresí. Jsou to JPEG a JPEG2000. Blíže o nich v [8, 10].

2.1.1 Barevné modely

Barevný model je abstraktní matematický model, který popisuje způsob, jak může být barva vyjádřena jako uspořádaná n-tice (nejčastěji trojice nebo čtveřice) složek barvy. Když se k modelu přidá přiřazovací funkce, určující jak mají být složky interpretovány, vznikne barevný prostor [10].

Existuje několik barevných modelů, z nichž každý má svůj účel. V této práci je třeba popsat dva a to RGB a YCbCr [8].

RGB Je nejvíce známý model. Využívá 3 složky: červenou, modrou a zelenou. Je to tzv. *aditivní* model, tzn. čím „více“ barev sečteme, tím světlejší je výsledek. Lidské oko obsahuje tyčinky a tři druhy čípků, které jsou citlivé (každý druh jinak) právě na základní barvy RGB. Přirozenou vlastností oka ale také je, že jas je detekován tyčinkami, které jsou oproti čípkům desetkrát citlivější. Proto je vhodné jako první krok separovat jasovou a barevné složky, tzn. převod do např. YCbCr.

YCbCr Tento model využívá jasovou složku Y a barevné složky Cb a Cr. Umožní tedy nakládat s každou složkou jinak. Model je použit v nejrozšířenějších kompresních standardech JPEG a MPEG. Pro převod z RGB do YCbCr slouží následující vztah [8]:

$$\begin{pmatrix} Y \\ Cb \\ Cr \end{pmatrix} = \begin{pmatrix} 0,299 & 0,587 & 0,114 \\ -0,16875 & -0,33126 & 0,5 \\ 0,5 & -0,41869 & -0.08131 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{2.1}$$

A v opačném směru:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1,402 \\ 1 & -0,34413 & -0,71414 \\ 1 & 1,772 & 0 \end{pmatrix} \begin{pmatrix} Y \\ Cb \\ Cr \end{pmatrix} \tag{2.2}$$

2.1.2 Hodnocení kvality obrazu

Kvalita obrazu je důležitý faktor ztrátové komprese. Například při porovnávání dvou kompresních metod se sleduje výsledná kvalita obrazu v závislosti na prostředcích nebo nákladech ke kompresní metodě potřebných.

Prakticky lze kvalitu hodnotit *objektivně (využitím měření)* nebo *subjektivně (využitím pozorovatelů).*

Subjektivní hodnocení Je přirozené, že vizuální kvalita by měla být posuzována lidskými pozorovateli. Při hodnocení jsou obvykle k dispozici celá pole obrazů současně dostupných k posouzení. Tyto obrazy jsou generovány s jedním parametrem fixním a měnícím se druhým parametrem (nebo množinou parametrů). Výsledné hodnocení může být použito ke klasifikaci kompresní metody.

Další metodou zjišťování účinku kompresní metody je metoda vybrání obrazů se stejnou subjektivní kvalitou z pole obrazů. Z výsledků lze vyvodit trendy kompresní metody.

Je jasné, že subjektivní metody jsou nákladné a sledování trvá dlouhou dobu, protože lidské oči jsou lehce unavitelné.

Objektivní hodnocení zavádí pro posouzení zkreslení poměr signál-šum (signal-to-noise ratio SNR). Další text bude zaměřen na objasnění SNR.

Uvažujme vstupní obraz jako funkci dvou proměnných *f(x,y)*, kde x a y jsou souřadnice bodů obrazu. Podobně definujme výstupní (rekonstruovanou) funkci *g(x,y)*. Dále určíme chybovou funci *e(x,y)* podle [8]:

$$e(x,y) = f(x,y) - g(x,y) \tag{2.3}$$

MSE(mean square error) je definována jako:

$$MSE = \frac{1}{MN} \sum_{x=0,y=0}^{M-1,N-1} e(x,y)^2 \tag{2.4}$$

kde M a N představují horizontální a vertikální rozměry obrazu. Někdy se také uvádí *RMSE*, který je definován jako:

$$RMSE = \sqrt[2]{MSE} \tag{2.5}$$

Pokud známe *MSE*, můžeme dále definovat:

$$SNR_{ms} = 10 \log_{10} \left(\frac{\sum_{x=0,y=0}^{M-1,N-1} g(x,y)^2}{MN \cdot MSE} \right) \tag{2.6}$$

$$SNR_{rms} = \sqrt[2]{SNR_{ms}} \tag{2.7}$$

V obrazové kompresi se také často pracuje s *PSNR* (peak signal-to-noise ratio):

$$PSNR = 10log_{10}\left(\frac{255^2}{MSE}\right) \qquad (dB) \tag{2.8}$$

Čím větší *SNR*, tím lepší je kvalita zpracovávaného obrazu ve vztahu k originálnímu obrazu.

Tento způsob se zdá být správný, ale jeho vypovídací hodnota není zcela směrodatná. Například v objektivním měření se neprojeví jinak velmi rušivý blokový artefakt u standardu JPEG. Dále je také třeba si uvědomit, že obrazový vjem v lidském mozku závisí na podnětu nelineárně.

Na druhou stranu je pořízení takového výsledku málo nákladné a snadno opakovatelné. V nejlepším případě jsou k hodnocení kvality obrazu použity jak subjektivní, tak objektivní metody měření.

2.2 Vlnková transformace

Jak již víme z předchozího textu, ke každé ztrátové kompresi je třeba transformace pro zjištění nepodstatných složek obrazu. Pro účely této práce je třeba popsat vlnkovou transformaci. Nejlépe se její vlastnosti popisují v souvislosti s Fourierovou transformací (FT), přesněji řečeno s krátkodobou Fourierovou transformací (STFT).

2.2.1 Důvod použití vlnkové transformace

U nestacionárních signálů si již nevystačíme popisem pomocí FT, protože nezjistíme pro který čas je spektrum platné. Pro dopřednou FT platí [9]:

$$S(\omega) = \int_{-\infty}^{\infty} s(t)e^{-j\omega t}dt \tag{2.9}$$

Pro zpětnou:

$$s(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} S(\omega)e^{j\omega t}d\omega \tag{2.10}$$

Jedna z důležitých vlastností je vyjádřena následujícím vzorcem:

$$s(at) \Leftrightarrow \frac{1}{a}S\left(\frac{\omega}{a}\right), a \neq 0. \tag{2.11}$$

Ukazuje, že roztažení signálu v čase způsobí smrštění spektra a naopak. Existuje však *mez* dosažitelného rozlišení v časové a v kmitočtové oblasti, která je dána

Heisenbergovým principem neurčitosti [9]. Ten říká, že je nemožné znát přesnou frekvenci přesně v té době, kdy se vyskytla.

Tento jev by se také dal vysvětlit tak, že pro vybrání konkrétního momentu signálu by se použil součin s posunutým Dirakovým impulzem. Z teorie signálů víme, že součin dvou signálů v časové oblasti odpovídá konvoluci jejich spekter. Spektrum Dirakova impulzu obsahuje současně všechny frekvence a konvolucí se spektrem původního signálu dojde k jeho úplnému znehodnocení ve frekvenční oblasti. Dostali bychom vlastně jen přesné časové rozlišení, ale zato žádné frekvenční.

STFT umožňuje sledovat kmitočtové vlastnosti nestacionárního signálu měnící se v čase. Pro STFT je třeba zavést funkci okna k časovému omezení signálu. Nejprve uvažujme vztah (2.9) ve tvaru skalárního součinu [9].

$$< s(t), \psi(t) >= \int_{-\infty}^{\infty} s(t)\psi^*(t)dt \tag{2.12}$$

* značí komplexní sdružení. U STFT se jako bázová funkce $\psi(t)$ bere:

$$\psi(t) = w(t-\tau)e^{j\omega t} \tag{2.13}$$

Kde $w(t-\tau)$ značí právě funkci obecně posunutého okna. STFT pak můžeme psát jako:

$$S_{STFT}(\omega, \tau) = \int_{-\infty}^{\infty} s(t)w(t-\tau)e^{-j\omega t}dt \tag{2.14}$$

Pro $\tau = konst.$ získáme spektrum signálu $s(t)w(t-\tau)$ (okno se neposunulo). Posouváním okna získáme spektrum dalších částí signálu a tedy popis signálu v časové i ve frekvenční rovině. Grafické znázornění se nazývá spektrogram.

Násobení signálu funkcí okna s sebou přináší nechtěné zkreslení spektra (viz. výše). Různými typy oken a také využitím přesahů lze tyto zkreslení odstranit.

Stále nám ale zůstává omezená časová a kmitočtová rozlišitelnost a také fakt, že bázová fce bude vždy komplexní harmonický signál. Některé nestacionární náhodné signály by byly často lépe (= méně koeficienty) popsány jinými bázovými funkcemi. Pro obejití těchto nedostatků je použita právě vlnková transformace.

2.2.2 Vlnková transformace obecně

Ve vlnkové transformaci je na místě bázové funkce použita plně modulovatelná vlnka. Protože to obecně není harmonický signál, nehovoří se už o frekvenci ale o měřítku [1]. Vlnková transformace je tedy vícerozměrná analýza signálu a to času a měřítka (graficky scalegram).

[1] malé měřítko znamená detaily, velké měřítko aproximační hodnoty

2.2.3 Spojitá vlnková transformace

Spojitá vlnková transformace (CWT) je definována vztahem [12]:

$$S_{WT}(p,\tau) = \int_{-\infty}^{\infty} s(t)\psi^*_{p,\tau}(t)dt \tag{2.15}$$

Kde * značí komplexní sdružení a proměnné p, τ měřítko a posunutí. Rovnice ukazuje jak je signál *s(t)* rozložen na množinu bázových funkcí $\psi_{p,\tau}$. Bázové funkce jsou generovány ze základní tzv. mateřské vlnky $\psi(t)$.

$$\psi_{p,\tau} = \frac{1}{\sqrt{p}}\psi\left(\frac{t-\tau}{p}\right) \tag{2.16}$$

Důležité je, že bázové vlnky nejsou nijak předepisovány, jak je tomu u Fourierovy transformace. Je tedy možné vytvořit si vlastní vlnku přímo podle typu analyzovaného signálu.

2.2.4 Vlastnosti mateřských vlnek

Každá mateřská vlnka musí splňovat podmínku *přijatelnosti* a *pravidelnosti*.

Podmínka přijatelnosti říká: vlnka splňující rovnici:

$$\int_{-\infty}^{\infty} \frac{\mid \Psi(\omega) \mid^2}{\mid \omega \mid} d\omega < +\infty \tag{2.17}$$

může být použita k analýze a rekonstrukci signálu bez ztráty informace. $\Psi(\omega)$ je Fourierova transformace mateřské vlnky $\psi(t)$. Z této podmínky také vyplývá, že spektrum vlnky se směrem od vyšších kmitočtů snižuje, až v nule musí být nulové. Spektrum je tedy *pásmově* omezené a stejnosměrná složka je v časové oblasti nulová [2].

Podmínka pravidelnosti říká, že vlnka by měla mít hladký průběh a určitá maxima v časové i frekvenční oblasti. Důkaz je dosti složitý proces [12], ale výsledkem je, že koeficienty vlnkové transformace musí rychle klesat s klesajícím měřítkem.

[2] musí to být „vlnka“

2.2.5 Diskrétní vlnková transformace

Pokud se podíváme na (2.15), vlnková transformace je vlastně korelace mezi signálem a spojitě se posouvající a spojitě expandující[3] vlnkou. Pro zpracování na počítači bychom ale potřebovali konečný počet transformačních koeficientů.

Hledáme tedy popis signálu co nejmenším počtem koeficientů. K tomu byly zavedeny tzv. diskrétní vlnky. Vlnkové funkce zůstávají spojité, ale posun a změny měřítka jsou diskrétní. Obvykle se změny volí tak, aby byly násobky dvou. V tomto případě už se jedná o dyadickou diskrétní vlnkovou transformaci, která se vyznačuje dyadickým vzorkováním časové i frekvenční osy. Definiční vzorec pro generování bázových vlnek se změní na:

$$\psi_{j,k}(t) = \frac{1}{\sqrt{2^j}}\psi\left(\frac{t - k2^j}{2^j}\right) \tag{2.18}$$

kde k, j jsou celá čísla.

Dalším krokem je zajistit, aby diskrétní vlnky byly ortogonální ke svým vlastním protažením a posunutím. To znamená, aby byl nulový skalární součin dvou různě posunutých a roztažených vlnek.

I po takovémto zjednodušení je možné, aby byl jakýkoli signál zrekonstruován z bázových (různě posunutých a roztažených mateřských) vlnek, vážených vlnkovými transformačními koeficienty.

2.2.6 Souvislost vlnkové transformace s filtry

Dokonce i s použitím diskrétních vlnek potřebujeme nekonečné množství posunutí a roztažení k popisu obecně nestacionárního signálu.

Pokud budeme uvažovat práci s časově omezeným signálem, množství posunutí již bude konečné. Dále nějak musíme omezit množství protažení.

Ze vzorce (2.11) již víme, že roztažení signálu v čase dvakrát způsobí smrštění šířky spektra dvakrát a zároveň posunutí všech frekvenčních složek na poloviční kmitočet[4]. Takovýmto postupem můžeme pokrýt analyzovaný signál v kmitočtu stejně jako jsme to udělali v časové oblasti posouváním vlnek. Z výše uvedeného se nabízí velmi výhodná realizace pomocí banky filtrů.

Zavedení měřítkové funkce Předchozí úvahou bychom k pokrytí celého spektra potřebovali zase nekonečné množství protažení, protože každým roztažením vlnky dojde k pokrytí vždy jen poloviny zbývající části spektra. Řešení spočívá v tom,

[3] často se neuvažuje komprese vlnky

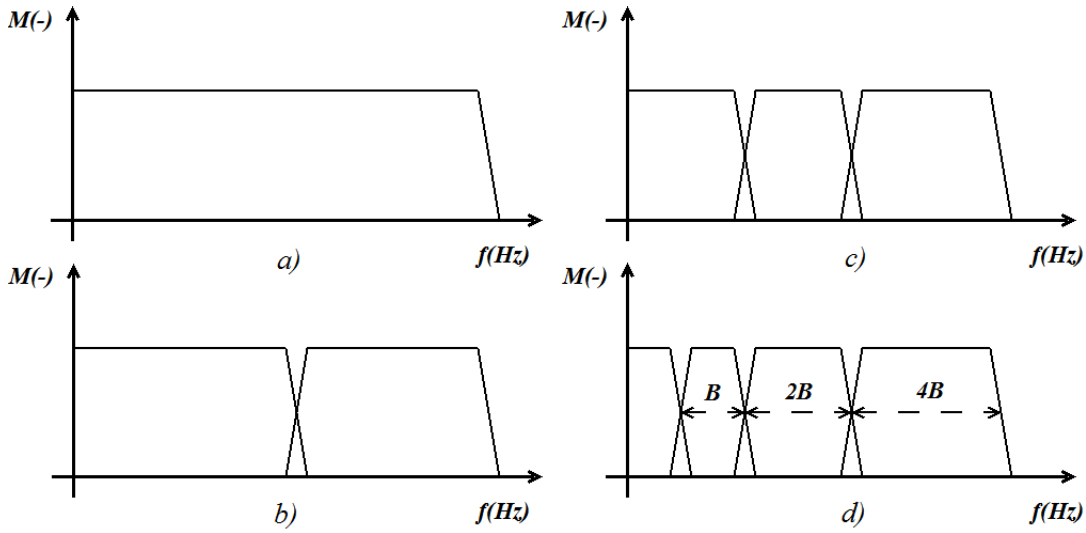
[4] Spektrum bázové vlnky je pásmově omezené!

nesnažit se pokrýt celé spektrum pomocí vlnek, ale v určitém kroku zastavit a zbývající část spektra uchovat jako *měřítkovou funkci*. Zavedením této funkce můžeme rozhodnout o „přesnosti“ vlnkové transformace, neboli o počtu změn měřítek, pro které chceme znát transformační koeficienty.

Pokud tedy vlnku vezmeme jako pásmovou propust a měřítkovou funkci jako dolní propust, můžeme nahlížet na sérii roztažení vlnek a měřítkovou funkci jako na banku filtrů.

Subpásmové kódování Předchozími úpravami jsme převedli vlnkovou transformaci na filtrování vstupního signálu bankou filtrů. Výstupy jednotlivých úrovní jsou vlnkové a měřítkové funkce. Proces filtrování bankou filtrů je znám jako subpásmové kódování [9]. Pro celou banku stačí dva typy filtrů, PP pro vlnkovou funkci a DP pro měřítkovou funkci. Výstup z DP pak může být znovu filtrován stejnými filtry pro další úroveň dekompozice (další protažení vlnky). Dva filtry, dělící pásmo na dvě subpásma, jsou označovány jako kvadraturní zrcadlové filtry. Velice důležitá vlastnost je jejich schopnost perfektní rekonstrukce. Podrobnosti v [9].

Na obr. 2.1 a) je znázorněno obecné kmitočtové spektrum digitálního signálu s maximální možnou šířkou pásma. $M(-)$ je modul přenosu. Obr. 2.1 b) znázorňuje rozdělení spektra na poloviny zrcadlovými filtry. Obr. 2.1 c) ukazuje rozdělení dolní poloviny spektra na další dvě pásma poloviční šířky. Obr. 2.1 d) ukazuje poměr šířek pásem B několika stupňů banky zrcadlových filtrů.



Obr. 2.1: Dělení pásem u banky filtrů

2.2.7 Diskrétní vlnková transformace s diskrétním časem

Při počítačovém zpracování máme samozřejmě všechny signály v diskrétní podobě. Dosud máme ale stále spojité vlnky (i když s diskrétním posunutím). V [12] je dokázáno, že pro zpracování diskrétního signálu stačí dosud uvažované filtry implementovat jako filtry diskrétní. Dostaneme se tedy k diskrétní vlnkové transformaci s diskrétním časem (DTWT) (názvosloví přejato z [9]). U bank diskrétních filtrů navíc po každé filtraci dochází k podvzorkování dvěma. To nám zaprvé zajistí stejnou délku transformačních koeficientů po jakémkoli počtu dekompozičních kroků a za druhé nám omezí maximální počet dekompozičních kroků vlastní délkou analyzovaného signálu. Navzorkovaný diskrétní signál je podle Nyquistova kritéria shora omezen frekvencí $f_{vz}/2$, kde f_{vz} je vzorkovací kmitočet. Dosud uvažovaná PP tedy přejde na HP.

Tímto jsme se dostali do stádia, kdy pomocí banky filtrů dokážeme provést vlnkovou transformaci libovolného (s konečnou energií) jednorozměrného náhodného diskrétního signálu do libovolné úrovně dekompozice (omezeno vlastní délkou vstupní posloupnosti). Tento postup je tak běžný, že nové vlnky se už definují jako impulzní charakteristiky filtrů.

2D Diskrétní vlnková transformace (s diskrétním časem). Pro naše účely je třeba řešit transformaci 2D signálu (obrazu). V obraze se s kmitočtem ani časem v pravém slova smyslu nepracuje. Místo času se bere prostorová informace (pořadí pixelu v horizontálním nebo vertikálním směru) a zavádí se pojem prostorový kmitočet. Je to normovaná veličina a představuje počet period harm. signálu ve všech směrech obrazu. Nejvyšší kmitočet v obraze se tedy může rovnat šířka/2 popř. výška/2. Odpovídá to situaci, kdy se v řádku nebo ve sloupci střídají černé a bílé pixely (nejjemnější detaily).

Transformace se může díky vlastnosti seperability provést nejprve po řádcích a pak po sloupcích. Tuto operaci budeme označovat jako jeden dekompoziční krok (případně jako pyramidový krok).

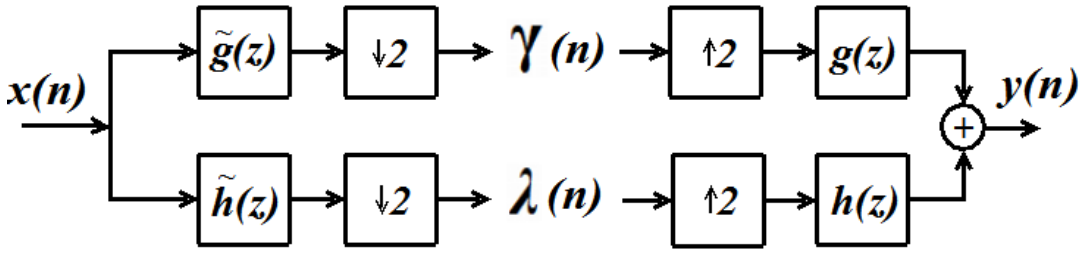
Na tomto místě by již bylo možné vybrat si vlnku (resp. jí odpovídající imp. charakteristiky) a klasickou konvolucí počítat výstupy filtrů. V této práci bude ale ještě představen rychlý způsob výpočtu vlnkové transformace a to pomocí *lifting schématu*.

2.2.8 Lifting schéma

Oproti klasické konvoluci nabízí lifting schéma menší výpočetní náročnost a také minimální vliv okrajových podmínek, se kterými se musí jinak počítat. Vychází se

z impuzlních charakteristik filtrů a přejde se ke struktuře, která se skládá jen z jednoduchých operací. Přechodu se říká *faktorizace* a výsledkem jsou lifting koeficienty.

V souladu s [13] definujme $\tilde{h}$ (pro DP) a $\tilde{g}$ (pro HP) jako imp. charakteristiky analyzujících filtrů a h a g syntetizujících filtrů [5]. $\tilde{\mathbf{P}}$ označuje *polyfázovou matici* a λ, γ značí měřítkové a vlnkové koeficienty. Obrázek 2.2 znázorňuje jeden krok vlnkové transformace. Analyzující filtry jsou následovány bloky podvzorkování dvěma.



Obr. 2.2: Jednoúrovňová banka filtrů

Pravá část znázorňuje rekonstrukci nadvzorkováním a syntetizujícími filtry. Následující vztah udává podmínku perfektní rekonstrukce:

$$h(z)\tilde{h}(z^{-1}) + g(z)\tilde{g}(z^{-1}) = 2 \qquad h(z)\tilde{h}(-z^{-1}) + g(z)\tilde{g}(-z^{-1}) = 0 \tag{2.19}$$

Polyfázová reprezentace Zaměřme se na část podvzorkování. Ve své podstatě je to zahození každého druhého vzorku, což je nehospodárné. Lepší by bylo zavést podvzorkování ještě před filtrací. K tomu se zavede rozdělení na sudé a liché vzorky a ty jsou filtrovány jen sudými a lichými vzorky impulzní charakteristiky. Vzorcem je to vyjádřeno takto:

$$y_s(z) = h_s(z)x_s(z) + z^{-1}h_l(z)x_l(z) \tag{2.20}$$

z^{-1} znázorňuje zpoždění mezi sudými a lichými vzorky.

Pokud předchozí aplikujeme na celý jeden krok vlnkové transformace, výsledek zapíšeme ve vektorovém tvaru:

$$\begin{pmatrix} \lambda(z) \\ \gamma(z) \end{pmatrix} = \tilde{\mathbf{P}}(z) \begin{pmatrix} x_s(z) \\ z^{-1}x_l(z) \end{pmatrix} \tag{2.21}$$

kde $\tilde{\mathbf{P}}$(z) je *polyfázová matice.*

[5] uvažuje všechny filtry s konečnou impulzní charakteristikou

$$\tilde{\mathbf{P}}(z)=\begin{pmatrix} \tilde{h}_s(z) & \tilde{h}_l(z) \\ \tilde{g}_s(z) & \tilde{g}_l(z) \end{pmatrix} \tag{2.22}$$

Polyfázová matice teď provádí vlnkovou transformaci. V dalším textu se budou provádět úpravy právě na ní. Teď zatím uvažujme, že je to jednotková matice a vše co dělá je, že rozděluje vstupní tok dat na liché a sudé vzorky.

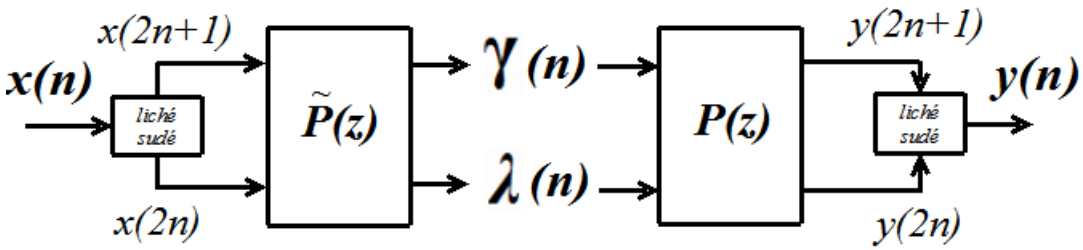
Podobným postupem dojdeme při rekonstrukci k následujícím vzorcům:

$$\begin{pmatrix} y_s(z) \\ zy_l(z) \end{pmatrix}=\mathbf{P}(\mathbf{z})\begin{pmatrix} \lambda_s(z) \\ \gamma_s(z) \end{pmatrix} \tag{2.23}$$

kde $\mathbf{P}$(z) je *duální polyfázová matice.*

$$\mathbf{P}(\mathbf{z})=\begin{pmatrix} \tilde{h}_s(z) & \tilde{g}_s(z) \\ \tilde{h}_l(z) & \tilde{g}_l(z) \end{pmatrix} \tag{2.24}$$

Na Obr. 2.3 je znázorněna polyfázová struktura. Pokud tedy matice $\tilde{\mathbf{P}}(z)$ provádí



Obr. 2.3: Polyfázové vyjádření jednoúrovňové banky filtrů

transformaci a matice $\mathbf{P}(z)$ rekonstrukci, k perfektní rekonstrukci je třeba aby:

$$\tilde{\mathbf{P}}(z^{-1})\mathbf{P}(z) = \mathbf{I} \tag{2.25}$$

To znamená, že $\mathbf{P}(z)^{-1} = \tilde{\mathbf{P}}(z^{-1})$. Po určitých odvozeních se zjistí vztahy mezi analyzujícími a syntetizujícími filtry:

$$\begin{aligned} \tilde{h}_s(z) &= g_l(z^{-1}) \\ \tilde{h}_l(z) &= -g_s(z^{-1}) \\ \tilde{g}_s(z) &= -h_l(z^{-1}) \\ \tilde{g}_l(z) &= h_s(z^{-1}) \end{aligned} \tag{2.26}$$

A tedy:

$$\tilde{h}(z) = -z^{-1}g(-z^{-1}) \qquad \tilde{g}(z) = z^{-1}(z)h(-z^{-1}) \tag{2.27}$$

Ve speciálním případě kdy $h = \tilde{h}$ a $g = \tilde{g}$ je vlnková transformace *ortogonální*, jinak je *biortogogonální*. Podle [13] je výhodné aby $det\,(\mathbf{P}(z)) = 1$), protože pak budou filtry h, g komplementární. (Tím pádem i filtry $\tilde{h}, \tilde{g}$)

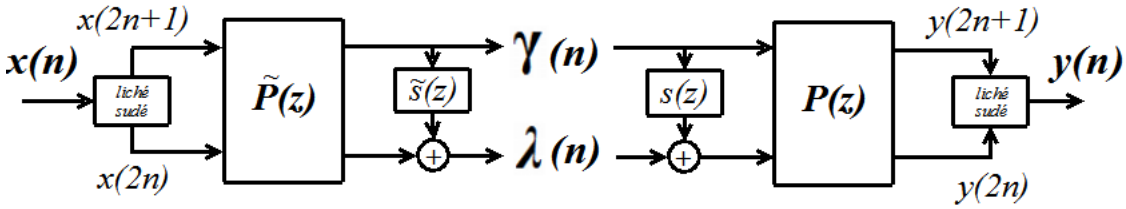
Lifting *Polyfázová matice* je matice Laurentových polynomů. Více o nich v příloze A.1. A pokud se uvažují filtry $\tilde{h}, \tilde{g}$ komplementární, tak každý nový FIR filtr $\tilde{h}^{new}$ komplementární k $\tilde{g}$ lze zapsat jako [13]:

$$\tilde{h}^{new}(z) = \tilde{h}(z) + \tilde{g}(z)\tilde{s}(z^2) \tag{2.28}$$

kde $s(z^2)$ je Laurentův polynom. Dosazením $\tilde{h}^{new}(z)$ za $\tilde{h}(z)$ do polyfázového vyjádření (2.22) vznikne:

$$\tilde{\mathbf{P}}^{new}(z) = \begin{pmatrix} \tilde{h}_s(z) + \tilde{g}_s(z)\tilde{s}(z) & \tilde{h}_l(z) + \tilde{g}_l(z)\tilde{s}(z) \\ \tilde{g}_s(z) & \tilde{g}_l(z) \end{pmatrix} = \begin{pmatrix} 1 & \tilde{s}(z) \\ 0 & 1 \end{pmatrix} \tilde{\mathbf{P}}(z) \tag{2.29}$$

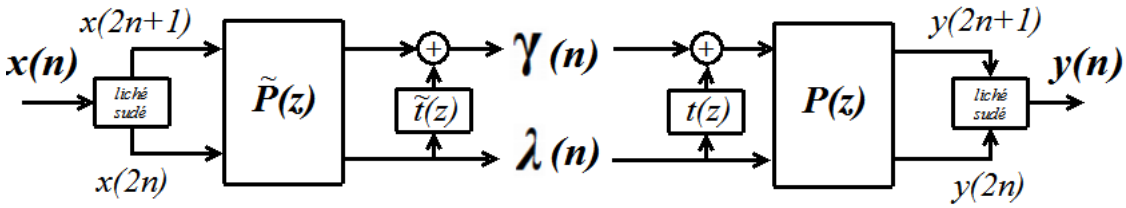
Tato operace se nazývá *primární lifting* nebo také *aktualizační krok*. Tímto došlo k „vyzvednutí“ nízkofrekvenčního pásma pomocí vysokofrekvenčního. Znázorněno na obr. 2.4.



Obr. 2.4: Znázornění aktualizačního kroku

„Vyzvednutí“ vysokofrekvenčního pásma za pomoci nízkofrekvenčního se nazývá *duální lifting* nebo také *predikční krok*. Podobnou úvahou jako v předchozím případě se dojde postupně k těmto rovnicím:

$$\tilde{g}^{new}(z) = \tilde{g}(z) + \tilde{h}(z)\tilde{t}(z^2) \tag{2.30}$$



Obr. 2.5: Znázornění predikčního kroku

$$\tilde{\mathbf{P}}^{new}(z) = \begin{pmatrix} \tilde{h}_s(z) & \tilde{h}_l(z) \\ \tilde{g}_s(z) + \tilde{h}_s(z)\tilde{t}(z) & \tilde{g}_l(z) + \tilde{h}_l(z)\tilde{t}(z) \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \tilde{t}(z) & 1 \end{pmatrix} \tilde{\mathbf{P}}(z) \quad (2.31)$$

Na obr. 2.5 jsou tyto vztahy znázorněny. Podobné vztahy by se daly odvodit pro syntetizující filtry $g^{new}(z)$ (v aktualizačním kroku) a $h^{new}(z)$ v predikčním kroku.

Zde se dostáváme k samotnému významu názvu této techniky. *Lifting* znamená vyzvednutí vlnkové transformace na novou úroveň složitosti [6]. Aplikací aktualizačních a predikčních kroků lze vytvořit i velice složité vlnkové transformace.

Rekonstrukce probíhá velice jednoduše tak, že stačí změnit znaménka všech *lifting* Laurentových polynomů a obrátit směr zpracování.

Faktorizace filtrů V předchozím jsme vyzvedli vlnkovou transformaci na vyšší úroveň složitosti. Samozřejmě je třeba mít operaci opačnou, kterou se převede existující vlnková transformace (impulzní charakteristiky filtrů) na kroky „liftingu“. Vztah (2.30) je tedy možné napsat opačně jako:

$$\tilde{g}(z) = \tilde{h}(z)\tilde{t}(z^2) + \tilde{g}^{new}(z) \quad (2.32)$$

Tato rovnice je identická k dlouhému dělení dvou Laurentových polynomů se zbytkem, kdy $\tilde{g}^{new}(z)$ je zbytek. Podobně lze zapsat opačně polyfázové vyjádření (2.31) jako:

$$\begin{aligned} \tilde{\mathbf{P}}(z) &= \begin{pmatrix} \tilde{h}_s(z) & \tilde{h}_l(z) \\ \tilde{h}_s(z)\tilde{t}(z) + \tilde{g}_s^{new}(z) & \tilde{h}_l(z)\tilde{t}(z) + \tilde{g}_l^{new}(z) \end{pmatrix} = \\ &= \tilde{\mathbf{P}}^{new}(z) \begin{pmatrix} 1 & 0 \\ \tilde{t}(z) & 1 \end{pmatrix} \end{aligned} \quad (2.33)$$

[6] Pro připomenutí, začínalo se s $\tilde{\mathbf{P}}(z) = I$

Pro jednu polyfázovou matici (jeden lifting krok) je třeba provést dvě dělení Laurentových polynomů A.2. Stejně můžeme pokračovat s dalšími lifting kroky (aktualizačními a predikčními), až dostaneme jednotkovou matici (nebo matici s konstatními koeficienty na hlavní diagonále). Toho se dosáhne vždy, pokud budou uvažované filtry $\tilde{h}, \tilde{g}$ komplementární [13]. Celý proces *faktorizace* je popsán následujícím vztahem:

$$\tilde{\mathbf{P}}(z) = \begin{pmatrix} K_1 & 0 \\ 0 & K_2 \end{pmatrix} \prod_{i=1}^{n} \left\{ \begin{pmatrix} 1 & s_i(z) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ t_i(z) & 1 \end{pmatrix} \right\} \tag{2.34}$$

2.2.9 Použití lifting schématu

Jak vyplývá z předchozího textu, vstupní posloupnost je omezená a pro zpracování je rozdělena na liché a sudé vzorky. Samotná realizace výpočtu ale rozdělení provede jen logicky. Posloupnost je upravována jednotlivými (predikčními a aktualizačními) kroky liftingu do výsledné podoby, kdy sudé prvky posloupnosti představují měřítkové (nízkofrekvenční) koeficienty a liché vlnkové (vysokofrekvenční) koeficienty nebo obráceně.

Pokud uvážíme posloupnost o sudé délce, můžeme ji znázornit následovně:

$$LHLHLH...LH$$

nebo

$$HLHLHL...HL$$

kde L, H jsou nízkofrekvenční, respektive vysokofrekvenční koeficienty.

Při výpočtu však dochází ke zkreslení na okrajích posloupnosti, tzv. blokový artefakt. Podle [14] má posloupnost začínající nízkofrekvenční složkou toto zkreslení na pravém konci, v případě 2D transformace pak na pravém a spodním okraji bloku. U posloupnosti začínající vysokofrekvenční složkou se toto zkreslení objeví na levém konci a podobně při 2D transformaci na levém a horním okraji bloku. Tato vlastnost byla ověřena, jak je vidět na obr. 3.5 (vpravo). Autoři [14] doporučují použít posloupností lichých délek pro omezení blokového artefaktu. Kódovací postup SPIHT ale vyžaduje rozměry sudé, proto zmiňovaného závěru v této práci není využito. Místo toho je zaveden překryv bloků, který by měl uvedený typ zkreslení také odstranit.

Na obr. B.6 je znázorněno lifting schéma pro transformaci pomocí biortogoální vlnky leGall5/3. Význam parametrů je následující: $x(n)$ je vstupní posloupnost,

a je lifting koeficient predikčního kroku, b aktualizačního a s je koeficient upravující energii pásem. $\lambda(n)$ představuje nízkofrekvenční koeficienty (L z předchozího odstavce) a podobně $\gamma(n)$ představuje vysokofrekvenční koeficienty (H z předchozího odstavce). Číslování vstupní posloupnosti uvažujme od 0. Ze schématu je vidět, že sudé vzorky $x_s(n)$ budou tvořit $\lambda(n)$ koeficienty a $x_l(n)$ tvoří $\gamma(n)$ koeficienty. Uspořádání posloupnosti tedy označme $LHLHLH...LH$.

Výpočet pomocí matic Vyjdeme z rovnice 2.21, do které dosadíme polyfázovou matici pro vlnku leGall5/3 (B.2) [7]:

$$\begin{pmatrix} \lambda(z) \\ \gamma(z) \end{pmatrix} = \begin{pmatrix} s & 0 \\ 0 & 1/s \end{pmatrix} \begin{pmatrix} 1 & b(1+z^{-1}) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ a(1+z) & 1 \end{pmatrix} \begin{pmatrix} x_s(z) \\ z^{-1}x_l(z) \end{pmatrix} \quad (2.35)$$

Matice jsou násobeny odzadu, v pořadí jednotlivých lifting kroků.

Programový výpočet Tentokrát ponechme indexaci v celém schématu na obr. B.6 shodnou. tzn. $x_s(0) = x(0)$, $x_l(0) = x(1)$, $x_s(2) = x(2)$ atd.. (při programové realizaci nedojde ke skutečnému rozdělení posloupnosti). Před prvním lifting krokem tedy platí $x_s(n) = \lambda(n) = x(n)$ pro n sudé a $x_l(n) = \gamma(n) = x(n)$ pro n liché. Označení lichých a sudých prvků ale ponechme pro názornější odlišení lifting kroků. Následuje naznačení výpočtu transformačních koeficientů (N je délka vstupní posloupnosti): Predikční krok pro $n = 1..N-3, n = n+2$:

$$x_l(n) = a(x_s(n-1) + x_s(n+1)) \quad (2.36)$$

Pro $n = N-1$ se řeší okrajová podmínka:

$$x_l(N-1) = 2 \cdot a(x_s(N-2)) \quad (2.37)$$

Aktualizační krok $n = 2..N-2, n = n+2$:

$$x_s(n) = b(x_l(n+1) + x_l(n-1)) \quad (2.38)$$

Pro $n = 0$ se řeší okrajová podmínka:

$$x_s(0) = 2 \cdot b(x_l(n+1)) \quad (2.39)$$

Po provedení výpočtu pro celou posloupnost následuje přeskládání koeficientů do tvaru $LLLL..LHHHH..H$ [8]

Opačná transformace se provede otočením směru zpracování a inverzí znamének a, b koeficientů a převrácením hodnot s.

[7] Z důvodu maticového zápisu uvažujme v obr. B.6 index n platný v rámci dané posloupnosti. Tzn. $x_s(0) = x(0)$, $x_l(0) = x(1)$, $x_s(1) = x(2)$ atd..

[8] L je totožné s x_s, H je totožné s x_l

2.2.10 Psychovizuální analýza dekompozice

Dekompoziční obrazec, jak je znázorněn např. na obr.3.3, znázorňuje množství informace v jednotlivých subpásmech. Podle [11] lze analýzou subpásem rozhodnout, do které ze skupin vnímání daný blok patří. Skupiny značí, jaký charakter obrazové informace daný blok obsahuje. Jsou to: silné hrany, hladké oblasti a detailní oblasti. K rozdělení do těchto skupin slouží parametr *aktivita* bloku. V této práci je ale tento parametr použit k odhadu délky výstupní bitové posloupnosti kompresního postupu SPIHT.

Výpočet aktivity se provádí nad každým HL a LH subpásmem. Nejprve se vypočte střední hodnota a směrodatná odchylka pro subpásmo. Aktivita v subpásmu odpovídá počtu koeficientů, jejichž absolutní hodnota je větší než směrodatná odchylka subpásma. Aktivita bloku je pak součet aktivit subpásem.

2.3 Metoda SPIHT

SPIHT [5] je kompresní algoritmus pracující s koeficienty vlnkové transformace. Nejprve vyhodnotí nejdůležitější koeficienty a ty pak začne od nejvyššího bitu kódovat. Algoritmus může být zastaven v libovolném okamžiku, nebo se může nechat provést do konce, kdy komprimovaná data reprezentují *téměř* bezeztrátový obraz. Úplně bezeztrátová komprese by byla jen v případě celočíselné vlnkové transformace. Jinak vzniknou chyby při zaokrouhlování. Velkou výhodou je také možnost progresivního dekódování tzn. kvalita se postupně zlepšuje.

Řadící algoritmus k rozdělení množin Jak již bylo naznačeno, koeficienty vlnkové transformace jsou seřazeny podle důležitosti. Řadící algoritmus je odvozen přímo z dat a je jasné, že musí být stejný na straně kodéru i na straně dekodéru. Algoritmus je také postaven tak, že není třeba řadit všechny koeficienty, ale jen ty, které jsou v daném *řadícím kroku významné.* Pokud označíme $c_{i,j}$ jako koeficient vlnkové transformace, tak v každém kroku se uvažují jen koeficienty $2^n \leq \mid c_{i,j} \mid < 2^{n+1}$ (n se snižuje o jedna v každém kroku). Pokud tedy platí $\mid c_{i,j} \mid \geq 2^n$ řekneme že koeficient byl *významný*, jinak že je *nevýznamný.*

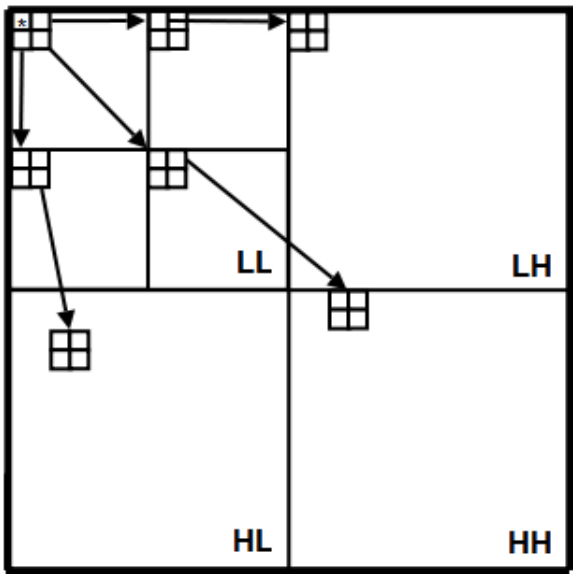
V dalším kroku jsou uvažovány koeficienty jako prostorové stromy (množiny) (viz. dále). Řadící algoritmus přeskupí množiny koeficientů do „rozdělujících podmnožin“ $\mathcal{T}_m$ a provede test významnosti celých podmnožin, tzn. pokud je alespoň jeden prvek podmnožiny významný, je významná celá podmnožina. Pokud je, určitým pravidlem se množina rozdělí na nové podmnožiny $\mathcal{T}_{m,l}$ a na nich je proveden nový test významnosti. Tento proces se opakuje, dokud některá z podmnožin bude významná. Snahou tohoto postupu je vytvořit množiny, které obsahují významné

koeficienty, co nejmenší a obráceně (nevýznamné množiny pak budou kódovány jedním reprezentantem). Definujeme funkci významnosti [5]:

$$S_n(\mathcal{T}) = \begin{cases} 1, & max\{|\ c_{i,j}\ |\} \geq 2^n \quad kde(i,j) \in \mathcal{T} \\ 0, & jinak \end{cases} \tag{2.40}$$

k určení významnosti podmnožiny $\mathcal{T}$. Zjednodušený zápis pro jednoprvkovou množinu bude $S_n(i,j)$.

Prostorově orientované stromy Vlastností vlnkové transformace je koncentrace energie v nízkofrekvenčních složkách. Navíc se zjistila prostorová podobnost mezi subpásmy, kdy odpovídající si prvky subpásem popisují stejnou oblast v obraze. Vztahy mezi těmito prvky lze vyjádřit stromovou strukturou, ve které jsou kořenové prvky v nejnižších subpásmech. Tyto orientované stromy určují prostorové



Obr. 2.6: Prostorově orientované stromy

závislosti koeficientů v hierarchické pyramidě obr.2.6. Stromy jsou uspořádány tak, že každý prvek má buď žádné nebo čtyři přímé následovníky, které tvoří vždy 2x2 sousední koeficienty. Šipky v obr.2.6 vedou vždy od rodiče k přímým potomkům. Byly definovány následující množiny koeficientů:

- $\mathcal{O}\ (i,j)$: množina souřadnic všech přímých potomků uzlu (i,j)
- $\mathcal{D}\ (i,j)$: množina souřadnic všech potomků uzlu (i,j)
- $\mathcal{H}$: množina všech kořenů prostorově orientovaných stromů (koeficienty v nejvyšším dekompozičním stupni)

- $\mathcal{L}\ (i,j) = \mathcal{D}\ (i,j) - \mathcal{O}\ (i,j)$

Prostorový vztah lze zapsat pomocí rovnice (neplatí pro nejvyšší a nejnižší stupně dekompozice):

$$\mathcal{O}\ (i,j) = \{(2i,2j),(2i,2j+1),(2i+1,2j),(2i+1,2j+1)\} \tag{2.41}$$

Úvodní rozdělení do stromů je následující:

1. vytvoří se $\mathcal{D}\ (i,j)$ pro každý $(i,j) \in \mathcal{H}$
2. pokud je množina $\mathcal{D}\ (i,j)$ významná ($S_n(\mathcal{D}\ (i,j)) = 1$), dojde k jejímu rozdělení na $\mathcal{L}\ (i,j)$ a na čtyři jednoprvkové množiny z $(k,l) \in \mathcal{O}\ (i,j)$ (tzn. čtyři přímí potomci).
3. pokud je množina $\mathcal{L}\ (i,j)$ významná ($S_n(\mathcal{L}\ (i,j)) = 1$), dojde k jejímu rozdělení na čtyři množiny $\mathcal{D}\ (k,l), (k,l) \in \mathcal{O}\ (i,j)$ (tzn. vzniknou čtyři množiny s přímými potomky jako novými kořeny)

Kódovací algoritmus V praktické implementaci je informace o pořadí testovaných podmnožin uchována ve třech dynamických seznamech.

- seznam nevýznamných množin - LIS (list of insignificant sets)
- seznam nevýznamných pixelů (koeficientů) - LIP (list of insignificant pixels [9])
- seznam významných pixelů (koeficientů) - LSP (list of significant pixels)

Prvek každého seznamu je identifikován souřadnicemi (i,j). V LIP a LSP je každý prvek přímo pixel, v LIS každý prvek reprezentuje buď množinu $\mathcal{D}\ (i,j)$, nebo $\mathcal{L}\ (i,j)$. Záznam v LIS tedy bude buď typu A pro $\mathcal{D}\ (i,j)$ nebo B pro $\mathcal{L}\ (i,j)$.

Během *řadícího průběhu* jsou testovány prvky v LIP (zde jsou prvky, které byly minulý krok nevýznamné) na významnost a ty které jsou významné, jsou přesunuty do LSP. Podobně jsou procházeny množiny v LIS. Pokud se ukáže, že jsou významné, dojde k jejich rozdělení na nové podmnožiny a původní množina je ze seznamu LIS odstraněna. Podmnožiny s více jak jedním elementem jsou přidány na konec LIS, zatímco jednoprvkové podmnožiny jsou přidány na konec LIP nebo LSP podle toho, zda jsou v daném kroku *významné* či *nevýznamné*. S prvky v LSP jsou pak ještě zpracovány v *upřesňovacím průběhu*.

[9] Podle [13] jsou zřejmě výrazy pixel a koeficient transformace synonyma.

Slovní popis algoritmu Pro snadnější pochopení je do práce také zařazen vývojový diagram C [4],[5].

1. **Inicializace:** $n = \lfloor \log_2 \left(max_{(i,j)}\{| \, c_{i,j} \, |\} \right) \rfloor$, LSP je prázdný, souřadnice prvků $(i,j) \in \mathcal{H}$ do LIP seznamu a do LIS seznamu jako typ A pouze ty, které mají nějaké potomky.

2. **Řadící průběh:**

   (a) pro každý prvek (i,j) v LIP proveď:

      i. na výstup bit $S_n(i,j)$

      ii. pokud je $S_n(i,j) = 1$, přesuň prvek (i,j) do LSP a na výstup znaménkový bit hodnoty koeficientu na souřadnicích (i,j) $c_{i,j}$

   (b) pro každý prvek (i,j) v LIS proveď:

      i. Pokud je prvek (i,j) typu A, potom:

      - na výstup bit $S_n(\mathcal{D}\,(i,j))$
      - pokud je $S_n(\mathcal{D}\,(i,j)) = 1$, potom:
        - pro každý prvek $(k,j) \in \mathcal{O}\,(i,j)$ proveď:
          * na výstup bit $S_n(k,l)$
          * pokud je $S_n(k,l)) = 1$, potom přidej prvek (k,l) na konec LSP a na výstup znaménkový bit $c_{i,j}$
          * pokud je $S_n(k,l)) = 0$, potom přidej prvek (k,l) na konec LIP
        - pokud $\mathcal{L}\,(i,j)$ není prázdný, pak přesuň prvek (i,j) na konec LIS jako typ B, pokud je prázdný odstraň (i,j) z LIS

      ii. pokud je prvek (i,j) typu B, potom:

      - na výstup bit $S_n(\mathcal{L}\,(i,j))$
      - pokud je $S_n(\mathcal{L}\,(i,j)) = 1$, potom:
        - přidej všechny $(k,l) \in \mathcal{O}\,(i,j))$ na konec LIS jako typ A
        - odstraň (i,j) z LIS

3. **Upřesňovací průběh:** Na výstup n-tý MSB pro každý prvek (i,j) v LSP, mimo ty, které byly přidány v posledním řadícím průběhu.

4. **Úprava kvantizačního kroku:** sniž n o jedna a pokračuj na krok 2 dokud bude $n > -1$. Jinak skonči.

Důležité je si uvědomit, že pokud v kroku 2.b mluvíme o přidání prvku na konec seznamu, musí se s nimi pracovat ještě před skončením toho samého řadícího

průběhu. Proces kódování lze kdykoli zastavit a výsledek bude vždy dekódovatelný. Délkou výsledné reprezentace se přímo nastavuje požadovaná kvalita.

Příklad výpočtu pomocí SPIHT algoritmu je možno nalézt v [4].

Dekodér Ke své správné funkci musí mít dekodér k dispozici počáteční n, rozměry kódovaného obrazu a stupeň dekompozice. Poté může podle vstupních dat přesně sledovat cestu dekodéru. Dekódovací algoritmus je možno lehce odvodit z kódovacího. Stačí místo výstupu uvažovat vstup.

Navíc musí dekodér ještě rekonstruovat koeficienty. Koeficient je do výsledného obrazu zapsán až tehdy, když jsou jeho souřadnice přesunuty do LSP. V tomto okamžiku je jasné, že absolutní hodnota koeficientu $c_{i,j}$ leží v intervalu $\langle 2^n, 2^{n+1})$. Současně už je také známé znaménko koeficientu. Rekonstruovaná hodnota se prozatím nastaví na $\hat{c}_{i,j} = \pm 1,5 \times 2^n$ V upřesňovacím průběhu se pak k prvku přičítá nebo od něj odečítá (podle znaménka) 2^{n-1}. Takto se koeficient každým krokem zpřesňuje.

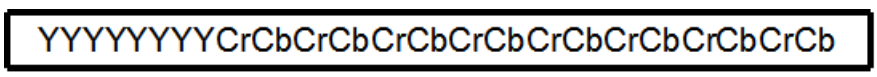
Velice důležitým předpokladem dekódování je naprosto přesná bitová synchronizace dekodéru s kodérem. Jakákoli odchylka znehodnotí celý dekódovací proces.

2.3.1 Zpracování barevné informace

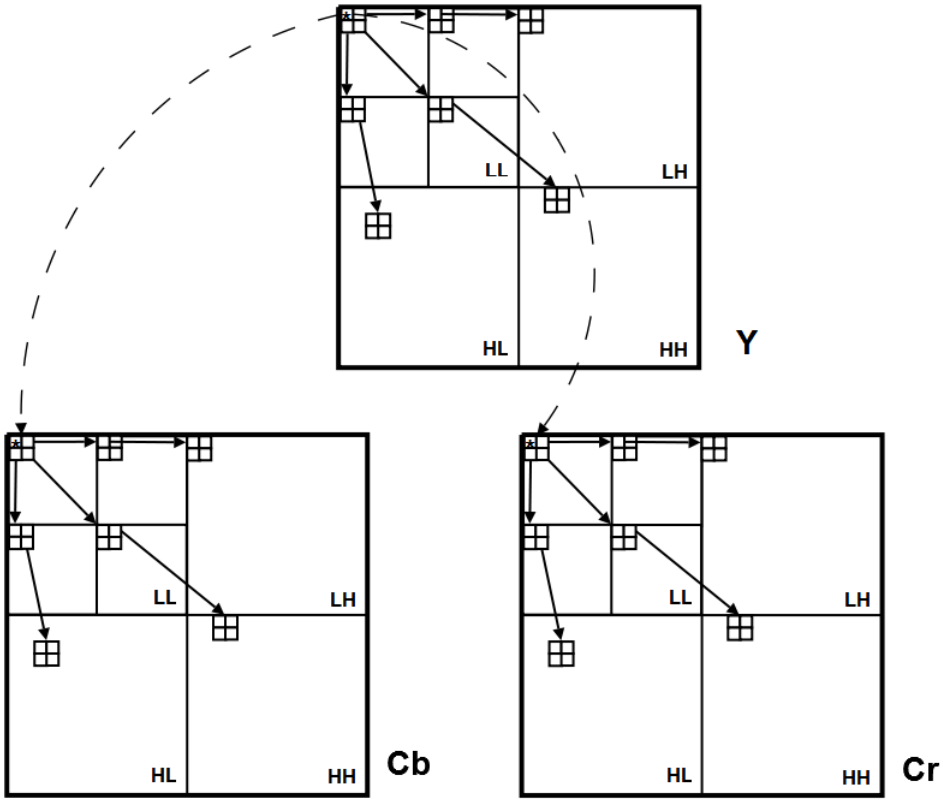
Metoda SPIHT, jak je popsána v [5], uvažuje obrazy obsahující pouze jasovou složku. Barevná informace obrazu je vyjádřena pomocí chrominančních složek modelu YCbCr. První logická úvaha by byla kódovat každou barevnou rovinu obrazu zvlášť. To by ale bylo velice nevýhodné, protože by se těžko synchronizovaly kódovací postupy všech rovin a dále by se zkomplikovala možnost progresivního dekódování. Byly proto navrženy modifikace SPIHT algoritmu umožňující kompresi a rekonstrukci barevných rovin obrazu najednou.

V dalším popisu předpokládejme, že máme k dispozici koeficienty vlnkové transformace provedené zvlášť na každé barevné rovině obrazu.

Metoda kombinace složek[7] spočívá v prostém přidání chrominančních složek do seznamů LIS a LIP podle stejných pravidel jako pro jasovou složku, případně se zavede jejich podvzorkování. Počáteční struktura seznamů je znázorněna na obr. 2.7. Tato metoda vytvoří sdružený bitový tok a umožní přesné nastavení míry komprese.



Obr. 2.7: Úvodní struktura seznamů LIP a LIS



Obr. 2.8: Prostorové stromy pro metodu CSPIHT

Metoda CSPIHT[6] vychází z předpokladu, že hodnoty v jasových rovinách jsou obecně větší než v barevných. Zavádí proto prostorově orientované stromy i mezi subpásmy jasových a chrominančních rovin. Počáteční naplnění seznamů LIS a LIP je provedeno nejprve pro jasovou složku, jak již bylo popsáno. Dále se do seznamu LIS přidají prvky z nejvyšší úrovně dekompozice jasové roviny, které původně nebyly kořeny prostorových stromů. V metodě CSPIHT tyto prvky představují kořeny stromů spojující jasovou rovinu s barevnými, jak je naznačeno na obr. 2.8. Každý tento nový kořen tedy bude mít 8 potomků, 4 z každé roviny. Metoda umožní použít stejný kódovací postup jako dosud a zlepší efektivitu kodéru, zejména při větších kompresních poměrech.

3 REALIZACE KODÉRU

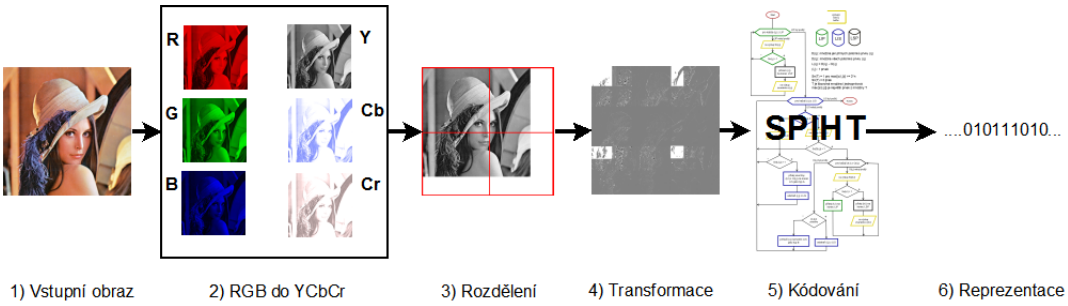
Pro implementaci byl použit programovací jazyk JAVA. Jako jeho výhody lze jmenovat multiplatformnost a objektově orientovaný přístup, jako nevýhody vysokou paměťovou náročnost a pomalý start aplikací. Objektově orientovaný přístup k návrhu softwaru je velice výhodný pro názornost, snadnou modifikovatelnost a rozšiřitelnost.

3.1 Vlastnosti kodéru

Program byl zpracován tak aby vyhovoval zadání, tzn. využívá lifting schéma k výpočtu koeficientů vlnkové transformace a SPIHT algoritmus ke kompresi těchto koeficientů. Dále pracuje s barevnou informací a zavádí segmentaci obrazu na dílčí bloky. Zpracovává i obrazy, jejichž rozměry nejsou celočíselným násobkem rozměrů bloků. Bloky mohou být obecně obdélníkové. Segmentace obrazů na bloky s sebou přináší zkreslení na okrajích bloků. Tento problém je částečně odstraněn zavedením přesahu mezi bloky. Z charakteru SPIHT algoritmu také vyplývá problém, kdy jediný parametr, který určuje výslednou kvalitu obrazu (délka výsledné posloupnosti), přímo nesouvisí s MSE a tedy výsledná kvalita (MSE) různých bloků je různá.

3.1.1 Navržený postup

Celý proces komprese obrazu je znázorněn na obr.3.1. První krok představuje vstupní obraz. V tomto kroku se také uvažuje zadání vstupních parametrů, což jsou: počet kroků dekompozice, velikost bloků, počet bitů na pixel a typ vlnky. Druhý krok pak znázorňuje transformaci barevných prostorů. Třetí krok ukazuje obecné rozdělení obrazu na bloky. Aby byla vlnková transformace realizovatelná, musí být velikost bloků celočíselný násobek 2^n, kde n je počet dekompozičních kroků. Z důvodu použití SPIHT algoritmu musí být rozměry bloků minimálně $2 \cdot 2^n$. Z omezení velikosti bloku také vyplývá, že ve většině případů dojde k určitému přesahu bloku mimo obrázek. Chybějící hodnoty jsou doplněny zrcadlením koeficientů. Ve čtvrtém kroku proběhne nad každým blokem zadaný počet dekompozičních kroků vlnkové transformace. Pátý krok algoritmem SPIHT zakóduje každý blok do bitové posloupnosti. Dekomprese by pak znamenala proces opačný s tím, že výsledkem by byl obraz zkreslený kompresním procesem (stupeň zkreslení je dán zaokrouhlovacími chybami a kompresním poměrem).



Obr. 3.1: Postup zpracování obrazu

3.1.2 Struktura programové realizace

Jako součást řešení byla vytvořena knihovna `SPIHTcodec.jar` a dále grafické uživatelské prostředí `SPIHTgui.jar`. Tento `*.jar` soubor je spustitelný a k jeho správnému běhu je třeba Java Runtime Environment (nejméně verze JRE 1.5). V projektu jsou navíc ještě použity knihovny:

- `jai_imageio.jar` pro práci se všemi běžnými formáty obrazu
- `jfreechart-1.0.9.jar, jcommon-1.0.12.jar` pro vykreslování grafů

dále v SPIHTgui knihovny:

- `appframework-1.0.3.jar, swing-worker-1.1.jar` pro samotné vytvoření GUI a správnou práci s vlákny.

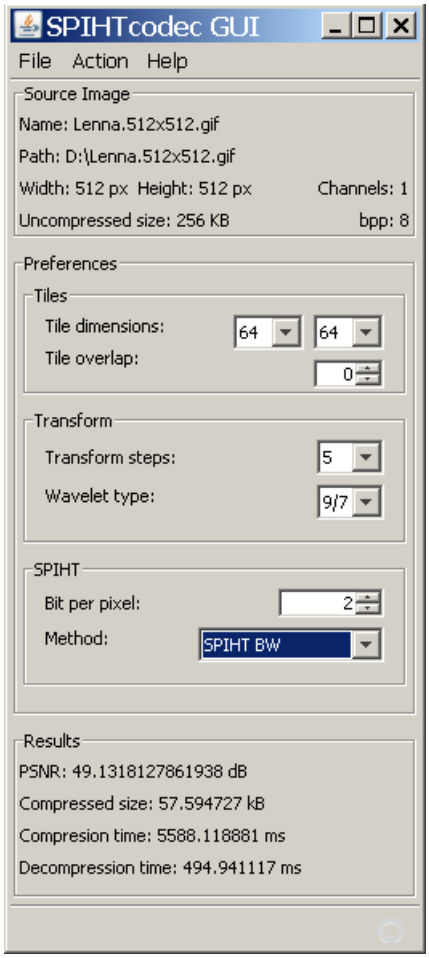
A samozřejmě

- Java Develoment Kit (JDK 1.6)

Celý programový balík je navržen s přihlédnutím k jeho dalšímu možnému rozšíření. Byly aplikovány zásady objektového návrhu aplikace a byly také použity některé návrhové vzory. Vytvořený program je tedy modulární, jednotlivé části snadno upravitelné a výpočetní postupy lehce vyměnitelné. Za tímto účelem bylo vytvořeno množství tříd a vztahy mezi nimi, které by nemusely být na první pohled zcela jasné. Kód je okomentován a z komentářů byla vytvořena HTML dokumentace pomocí nástroje Javadoc. Pro snadnější orientaci ve vztazích mezi třídami byl také vytvořen diagram tříd, který je pro svůj značný rozsah umístěn na přiloženém CD.

Grafické prostředí (na obr.3.2) je podle zásad objektového programování odděleno od aplikačního prostředí. Pomocí GUI se vybírá obraz určený ke zpracování a parametry komprese. Vstupní obraz může být díky knihovně `jai_imageio.jar` v téměř libovolném formátu. V části *Source image* jsou vypsány základní informace

o načteném obrazu. V další části nazvané *Preferences* se nastavují rozměry bloků na které bude obraz dělen, jejich přesah, dále počet kroků a typ vlnkové transformace a nakonec požadovaný počet bitů na pixel a typ kompresní metody SPIHT. V části *Results* jsou pak zobrazeny informace o kompresi. V menu Action je pak možné nastavit, zda se mají vykreslit grafy MSE, zda se má zobrazit tabulka s předchozími výsledky a dále je zde možnost nastavení výpočtu *Memory Preservation*, kdy při kompresi nebude načítán celý obraz, ale vždy jen jeden blok. Při této volbě ale nebude počítán odhad aktivity bloků ani dohad pomocí komprese „na zkoušku“. Kompresní a dekompresní proces je spouštěn z menu *Action* položkou *Show results*. Vlastní výpočet je pak proveden v odděleném vlákně a probíhající výpočet indikován v GUI.



Obr. 3.2: Grafické prostředí

Uživatel knihovny `SPIHTcodec.jar`(v našem případě GUI) musí volat statickou metodu `run(HashMap params, HashMap output, FWT wmethod, SPIHTcode`

smethod, BufferedImage bi) třídy Compress, ve které předává vstupní parametry param ve formě mapy parametr-hodnota. Podobně je definována mapa výstupních parametrů output. Wmethod je objekt definující výpočet dopředné vlnkové transformace. Smethod podobně definuje způsob výpočtu SPIHT algoritmu. Bi je vstupní obraz reprezentovaný objektem třídy BufferedImage ze standardního balíku java.awt.

V metodě dojde k rozdělení obrazu bi na bloky, z nichž každý je reprezentován objektem třídy BaseTierMain. Rozdělení provádí statická metoda třídy Util imageToTiles(Map params,BufferedImage bi,LinkedList tilesIn), kde tilesIn je seznam, ve kterém budou organizovány jednotlivé bloky BaseTierMain. Případně rozdělení na bloky provádí objekt třídy TileIterator, který však nebude pracovat se seznamem tilesIn a bloky umožní zpracovávat po jednom. Při rozdělování obrazu se také provede převod mezi barevnými prostory metodou Util.RGBtoYCbCr(float[] RGB). Vlnková transformace se provede voláním metody transform(WT method, int steps) objektu BaseTierMain, kde method je dříve zmíněná wmethod a steps je počet kroků transformace. SPIHT komprese se provede metodou code(SPIHT method, int length). metod je smethod dříve zmíněná a length udává délku výsledné posloupnosti. Každý zakódovaný blok je reprezentován objektem typu OutBitBuffer. Všechny zakódované bloky jsou pak uloženy v objektu třídy BitReprezentation. Tento objekt představuje celý zkomprimovaný vstupní obraz a dále obsahuje parametry potřebné k rekonstrukci obrazu.

Při rekonstrukci obrazu je vytvořena jeho nová reprezentace instancí třídy BufferedImage bf. Z každého OutBitBuffer uloženého v BitReprezentation se vytvoří nový objekt BaseTierMain. Tomu se zavolají metody code(SPIHT method, int length), kde dekompresní objekt method se získá z kompresního objektu smethod metodou smethod.getDecodeMethod(), délka length se nastaví na 0 pro dekódování celé vstupní posloupnosti. Podobně se provádí inverzní transformace voláním metody transform(WT method, int steps), kde method se získá z objektu definující dopřednou transformaci wmethod voláním wmethod.getInvTrans(), počet kroků transformace steps musí být stejný jako u dopředné. Výstupní obraz bf je pak po blocích naplňován metodou makeImage(BufferedImage bf, int col, int row) objektu typu BaseTierMain. bf je výstupní obraz, col a row udávají pozici bloku v obraze. Ve zbytku metody run dochází k výpočtu PSNR a případnému vykreslení grafů MSE po řádcích a po sloupcích.

Obsah knihovny SPIHTcodec.jar:

- Balíček GUI

Obsahuje třídy starající se o vykreslování obrazů a grafů. Vzhledem k obecně známému postupu bude popis vynechán.

- Balíček `Main`

  Obsahuje třídu Compress s jedinou statickou metodou `run`, která byla popsána v předchozím textu.

- Balíček `SPIHT`

  Obsahuje třídy realizující SPIHT algoritmus, navržené tak, aby bylo možné lehce přidat nový postup a minimalizovala se možnost duplikace kódu.

- Balíček `Structures`

  Třídy reprezentují výsledné bitové posloupnosti kompresního algoritmu SPIHT.

- Balíček `SubbandStructures`

  Obsahuje třídy reprezentující bloky obrazu, subpásma a vztahy mezi nimi. Dále jsou zde třídy definující pyramidové transformační kroky, ale vlastní algoritmus výpočtu je z důvodu snadné rozšiřitelnosti definován jinde.

- Balíček `Util`

  Definuje pomocné třídy.

- Balíček `Wavelet`

  Zde jsou definovány třídy realizující výpočet jednoho kroku jednorozměrné transformace pro různé vlnky.

Detailnější popis je v HTML formátu na přiloženém CD ve složce Javadoc, případně vztahy mezi třídami jsou naznačeny v diagramu tříd.

3.2 Implementační záležitosti

V této části jsou popsány pokročilé implementační problémy definované v zadání práce, nebo objevené v průběhu realizace.

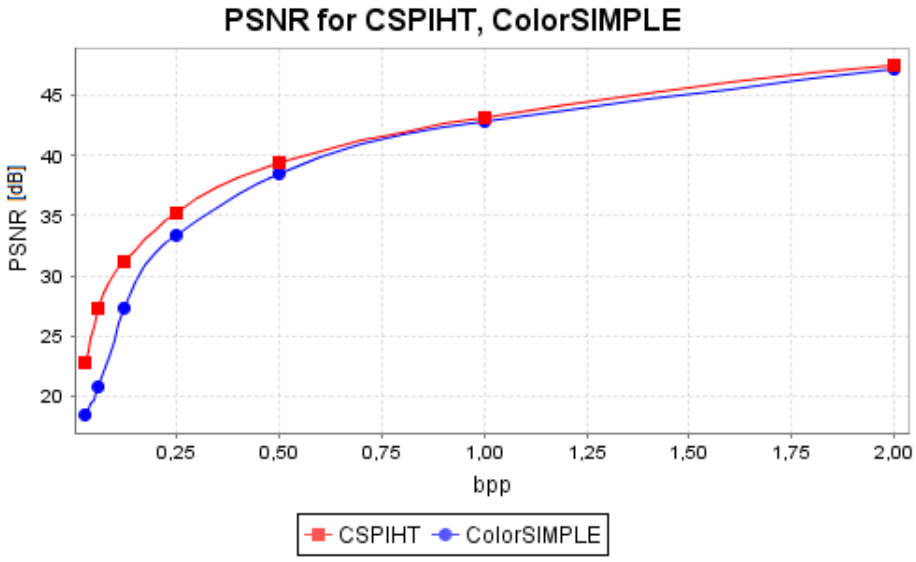
3.2.1 Fast lifting metoda

Způsob výpočtu pro vlnku leGall 5/3 byl již naznačen v kapitole 2.2.9. Výpočet transformace pomocí vlnky CDF 9/7 obsahuje navíc ještě jeden aktualizační a jeden predikční krok liftingu, jak je vidět na schématu obr.B.3 (koeficienty c a d). Dekompoziční obrazec pro 3 dekompoziční stupně a vlnku leGall 5/3 je na obr.3.3

(jednotlivá pásma jsou normalizována do rozsahu 0-255 a ekvalizována, hodnoty koeficientů ve skutečnosti s hloubkou dekompozice velice rychle rostou).

Obr. 3.3: Originál a dekompoziční obrazec obrázku Lenna.512x512.gif

V knihovně `SPIHTcodec.jar` je realizován výpočet pro vlnky CDF 9/7 (třídy `CDF97forw.java, CDF97inv.java` v balíčku `Wavelet`) a leGall 5/3 (třídy `LeGall53forw.java`, `LeGall53inv.java` v balíčku `Wavelet`).



Obr. 3.4: PSNR v závislosti na bpp pro obrázek Lenna512.png

3.2.2 Práce s barevnou informací obrazu

Zpracování barevných rovin obrazu bylo dosaženo použitím dvou modifikací SPIHT jak jsou popsány v kapitole 2.3.1 (dále označovány jako ColorSIMPLE a CSPIHT). Díky modulárnímu návrhu programu postačí změnit jen inicializační část kódovacího postupu SPIHT. Stačí tedy implementovat metodu `inicialization(SubbandLL[] tops, BitBuffer out)` abstraktní třídy `SPIHTcodeStepsDefault` pro kódování a třídy `SPIHTdecodeStepsDefault` pro dekódování. Metoda provádí úvodní naplnění seznamů LIP a LIS. Na obr.3.4 je sledován $PSNR$ v závislosti na *bpp* pro obě modifikace algoritmu SPIHT. Lépe vychází metoda CSPIHT, která jasovou rovinu kóduje přednostně a na barevné roviny je kódovací algoritmus SPIHT rozšířen pomocí modifikovaných prostorových stromů.

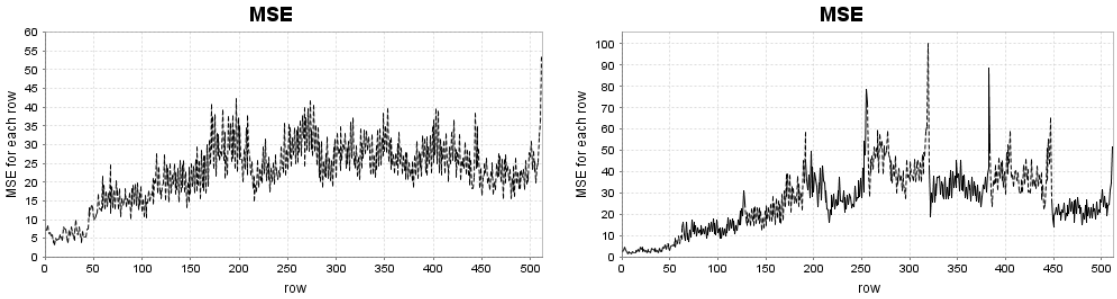
3.2.3 Segmentace obrazu

Rozdělením obrazu na bloky dochází při vyšších kompresních poměrech k blokovému artefaktu, který snižuje PSNR a sám o sobě působí velice rušivě. Toto zkreslení způsobuje vlnková transformace prováděná zvlášť nad každým blokem a při vyšších kompresních poměrech i charakter SPIHT algoritmu.

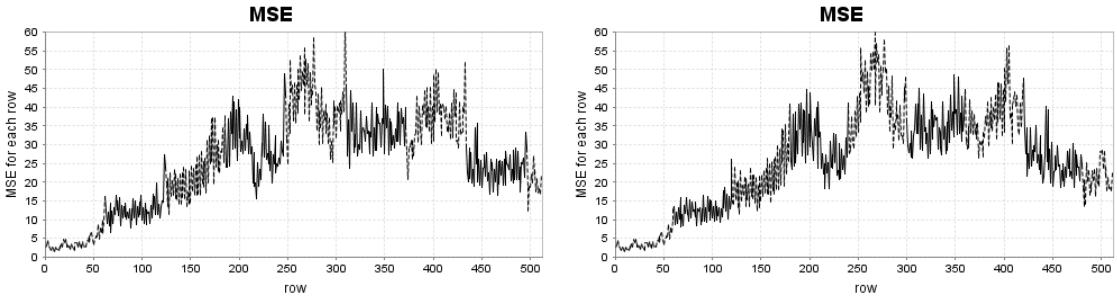
V programu tuto operaci provádí statická metoda `imageToTiles(Map params, BufferedImage bi, LinkedList tilesIn)` najednou pro celý obraz, případně objekt třídy `TileIterator`, který umožní načíst vždy jen aktuální blok obrazu a šetřit tak paměť.

Problém transformace Implementace transformace pomocí lifting schématu přináší výhodu v daleko menším vlivu okrajových podmínek než při výpočtu pomocí klasické konvoluce (již naznačeno v 2.2.9). Při použití lifting schématu se řeší vždy jen přesah jednoho prvku a to tak, že chybějící prvek je nahrazen zrcadleným posledním prvkem. Při tomto postupu však dochází k již zmiňovanému zkreslení na okrajích. Existují algoritmy schopné toto zkreslení odstranit, ale v této práci je tento problém řešen jednodušeji.

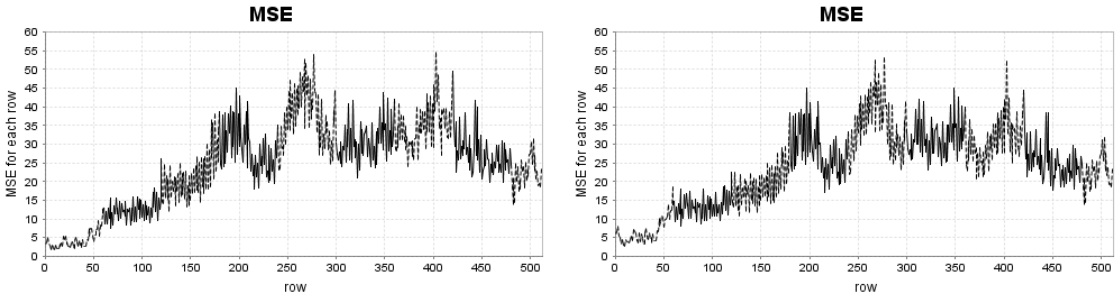
Blokový artefakt způsobený vlnkovou transformací je vidět na obr. 3.5 vpravo (vlevo je „ideální“ průběh bez blokového artefaktu). Je zde zobrazen průběh MSE v jednotlivých řádcích obrazu. Bloky byly záměrně zvoleny tak, aby v horizontálním směru obsáhly celý obraz. A to z důvodu odstranění vlivu blokového artefaktu v horizontálním směru na výpočet MSE řádků. Použit byl obraz *Lenna.512x512.gif*, 5 kroků transformace a velikost bloků 512x64. Obrazy odpovídající grafům je pro porovnání možné nalézt na přiloženém CD. Vysoké špičky průběhu obr. 3.5 (vpravo) na koncích bloků indikují zkreslení blokovým artefaktem. Jelikož se toto zkreslení vyskytuje vždy na konci bloku, je možné ho odstranit překrytím zkreslených pixelů



Obr. 3.5: MSE v řádcích, blok 512x512 (vlevo), 512x64 (vpravo)



Obr. 3.6: MSE v řádcích, přesah 2 pixely (vlevo), 4 pixely (vpravo)



Obr. 3.7: MSE v řádcích, přesah 4 pixely, odhad pomocí aktivity (vlevo), pomocí komprese „na zkoušku“ (vpravo)

blokem následujícím. To ovšem zavádí určitou redundanci. Na obr. 3.6 (vlevo) je zobrazen průběh MSE pro přesah 2 pixely, vpravo pak pro přesah 4 pixely. Je vidět, jak se vysoké špičky zavedením přesahu ztrácí.

Problém SPIHT algoritmu Ze vstupních parametrů, konkrétně z rozměrů obrazu a bitů na pixel vychází počet bitů, do kterých se má zakódovat celý obraz. Při rozdělení obrazu na bloky narazíme na problém, jaký počet bitů jednotlivým blokům přiřadit. Poměrové rozdělení podle velikosti bloků je nejjednodušší možný způsob. Délky výstupních bitových posloupností jsou tedy pro každý blok stejné. To nám ale nezaručí stejné zkreslení bloku, protože kvalita rekonstruovaného bloku je

závislá na zpracovávaných datech. Nakonec tedy vznikne mezi bloky přechod velmi podobný blokovému artefaktu (tentokrát ale z jiných příčin).

Experimentálně byl zkoumán vliv změn délek výstupních posloupností jednotlivých bloků. Univerzální pravidlo, platné pro všechny typy obrazů, však nalezeno nebylo. Vyžadovalo by to velice důladnou analýzu obrazových dat nebo rozsáhlé modifikace SPIHT algoritmu. Počet bitů na blok resp. poměry počtu bitů mezi bloky byly odhadovány pomocí *aktivity* bloků a kompresí „na zkoušku".

Výpočet aktivity bloku je popsán v kapitole 2.2.10. Čím větší hodnota aktivity, tím větší vnímaná úroveň detailů v obraze a tím delší posloupnost musí být bloku přiřazena. Bity jsou tedy rozděleny v poměru aktivit bloků.

Při kompresi „na zkoušku" se všechny bloky zakódují do stejného počtu bitů. Po dekompresi se u každého bloku spočítá MSE a v jejich poměru se pak rozdělí blokům jejich bitové délky.

Tyto metody sice omezí, ale úplně neodstraní blokový artefakt způsobený SPIHT algoritmem, zvlášť při vyšších kompresních poměrech. Dosažené výsledky je možné porovnat na obr. 3.7, kde je patrný určitý posun MSE řádků směrem k nižším hodnotám MSE a k průběhu ideálnímu, tzn. bez zavedení segmentace (obr. 3.5 vlevo).

3.3 Vlastnosti kodéru

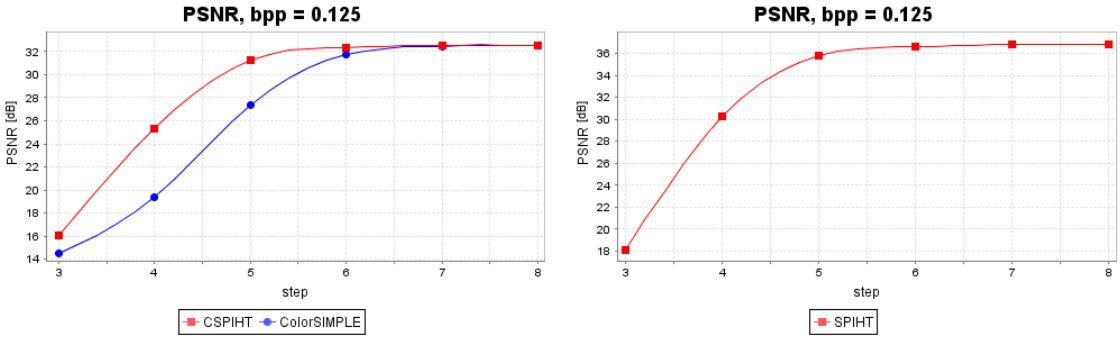
V této části je porovnáván vliv různých parametrů na výsledek komprese. Testovací obrázky jsou na obr.D.3.

3.3.1 Optimální počet transformačních kroků

Grafy na obr.3.8 zobrazují závislost PSNR na počtu transformačních kroků pro SPIHT algoritmus a jeho modifikace při $bpp = 0.125$ bez rozdělení na bloky. Je vidět, že od 5-ti úrovní dekompozice je nárůst PSNR už jen pozvolný. Větší počet kroků znamená větší výpočetní náročnost. Jako optimální tedy označme 5-6 transformačních kroků. To také znamená, že rozměry bloků d musí být větší než $d_{min} = 2 \cdot 2^5 = 64$ resp. $d_{min} = 2 \cdot 2^6 = 128$.

3.3.2 Optimální velikost bloku

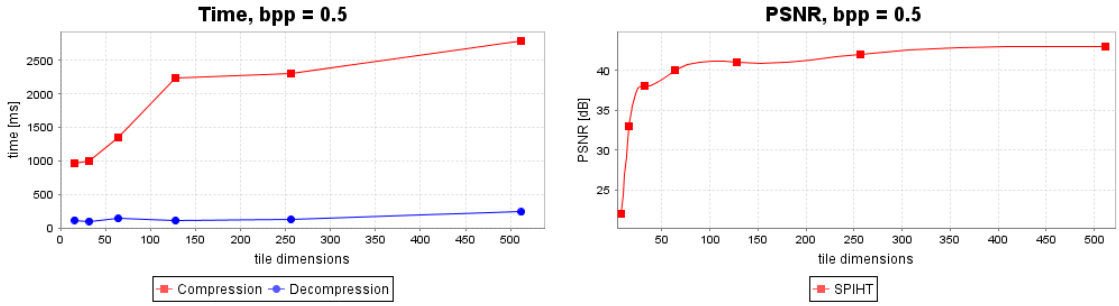
Vliv velikosti bloků na paměťovou náročnost postupu je zřejmý. Zbývá ještě zjistit, jak se mění ostatní parametry komprese v závislosti na zvolených rozměrech bloků. Na obr.3.9 jsou znázorněny doby komprese a dekomprese a dosažené PSNR. *Bpp* bylo ve všech případech rovno 0.5, aby se blokový artefakt co nejméně promítl do



Obr. 3.8: Závislost PSNR na transformačních krocích pro obrázek Lenna512.png (Lenna.512x512.gif)

výsledného PSNR. Počet kroků transformace je volen tak, aby v nejvyšším subpásmu LL vždy zbyly alespoň 2x2 koeficienty.

Z obr.3.9 vyplývá, že optimální rozměry bloků z hlediska výpočetní náročnosti a dosaženého PSNR jsou 32 a 64 pixelů. Z hlediska pouze dosaženého PSNR platí, že čím větší blok, tím lepší. To si je možné vysvětlit použitím více dekompozičních kroků a vlastním charakterem algoritmu SPIHT.



Obr. 3.9: Kompresní a dekompresní čas (vlevo) PSNR (vpravo) v závislosti na velikosti bloků pro obrázek Lenna.512x512.gif

3.4 Hodnocení

Následující text obsahuje shrnutí výsledků realizovaného kodéru, porovnání se zadáním a zhodnocení zjištěných výstupů.[1]

[1]Při testování byl použit notebook ASUS A6J T2300@1.66GHz,1.00GB RAM, Windows XP Professional SP2, JRE 1.6.0(build 1.6.0_02-b06)

3.4.1 Výpočetní a paměťová náročnost

Náročnost celého kódovacího procesu je velmi důležitým faktorem, který ukazuje jeho využitelnost na stávajících platformách a zařízeních. Jedná se o nároky na *paměť* a na *výpočetní výkonnost*, se kterými se váže další parametr: *spotřeba* - důležitý zejména u mobilních zařízení.

Při práci s obrazem je tedy nutné mít v paměti uložen vstupní a výstupní obraz pro výpočet PSNR, MSE. Dále je třeba mít v paměti uložen vždy alespoň jeden zpracovávaný blok. Poslední položkou uloženou v paměti je vlastní výstupní bitová posloupnost. Při dekompresi je výstupní obraz rekonstruován po blocích a to zase vždy po jednom. K těmto nárokům se ještě přidá vlastní paměťová náročnost výpočtu vlnkové transformace a SPIHT algoritmu.

Časovou náročnost kompresního procesu omezme na problém hledání náročnosti výpočtu transformace a SPIHT kódování.

Časovou a paměťovou náročnost hodnoťme absolutně (f) a poté asymptoticky, pro nejhorší případ pomocí notace velké-$\mathcal{O}$[2].

Složitost transformace je dána její implementací a možností využití vlastnosti separability transformace. Vlastní výpočet je urychlen programovou implementací lifting schématu.

Uvažujme obrazový blok pro jednoduchost o rozměrech $N * N$ dále n počet dekompozičních kroků a k počet lifting kroků (aktualizačních a predikčních).

Při výpočtu jednoho kroku transformace nad jedním řádkem nebo sloupcem délky N je zapotřebí $2N$ paměťového prostoru (vstupní a výstupní posloupnost). Další krok transformace je prováděn nad vektorem délky $N/2$, další nad $N/4$. Můžeme tedy říci, že paměťová náročnost je lineární $\mathcal{O}(\mathcal{N})$

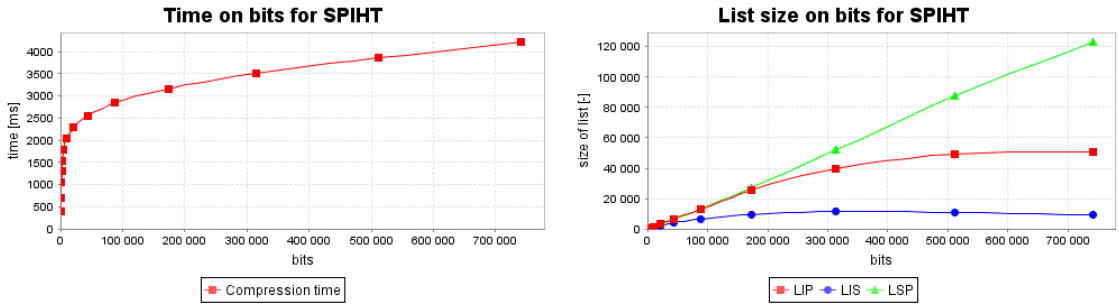
Programová realizace pro jeden řádek (nebo sloupec) používá k cyklů délky $N/2$ a jeden délky N. Výpočet se provádí v cyklu pro N řádků a pak v cyklu pro N sloupců ($2N * [N(k/2+1)]$). Pro n dekompozičních kroků vychází:

$$f(N) = \sum_{i=0}^{N-1} \left(\frac{N}{2^{i-1}}\right)^2 * (\frac{k}{2} + 1) \tag{3.1}$$

kde $f(N)$ je absolutní složitost výpočtu v závislosti na N. Asymptotická složitost je tedy kvadratická $\mathcal{O}(N^2)$. Složitost pro jeden řádek nebo sloupec je ale lineární.

Složitost SPIHT algoritmu a její vyčíslení je poněkud složitější problém, protože náročnost SPIHT algoritmu je velice závislá na zpracovávaných datech. Pro SPIHT je charakteristická asymetrická časová náročnost kódování a dekódování. Dekódování je náročné mnohem méně, jak je vidět např. z obr. 3.9 vlevo.

Složitost kompresního algoritmu určeme pro zjednodušení pomocí experimentálního měření na testovacím obrazu (Lenna.512x512.gif, 512x512, 6 kroků transformace). Délka kódování je přímo dána požadovaným počtem bitů po kompresi. Aby proběhl celý algoritmus, je pro testovací obrázek třeba 740962 bitů (což odpovídá 2,82 bpp pro téměř bezeztrátovou kompresi). Pro tuto hodnotu sledujme časovou a paměťovou náročnost v průběhu komprese. Paměťovou náročnost představuje vstupní blok transformačních koeficientů rozměrů 512x512, dynamické seznamy LIP, LIS a LSP a počet bitů výstupní posloupnosti. Naměřené hodnoty jsou vyneseny v grafech na obr.3.10. Vlevo je znázorněna závislost mezi bitovou délkou výstupu a dobou komprese, vpravo pak velikost seznamů v průběhu komprese. Jak je vidět,velikost seznamů LIP a LIS se od určité délky výstupní posloupnosti nezvětšují, zatímco seznam LSP roste lineárně.

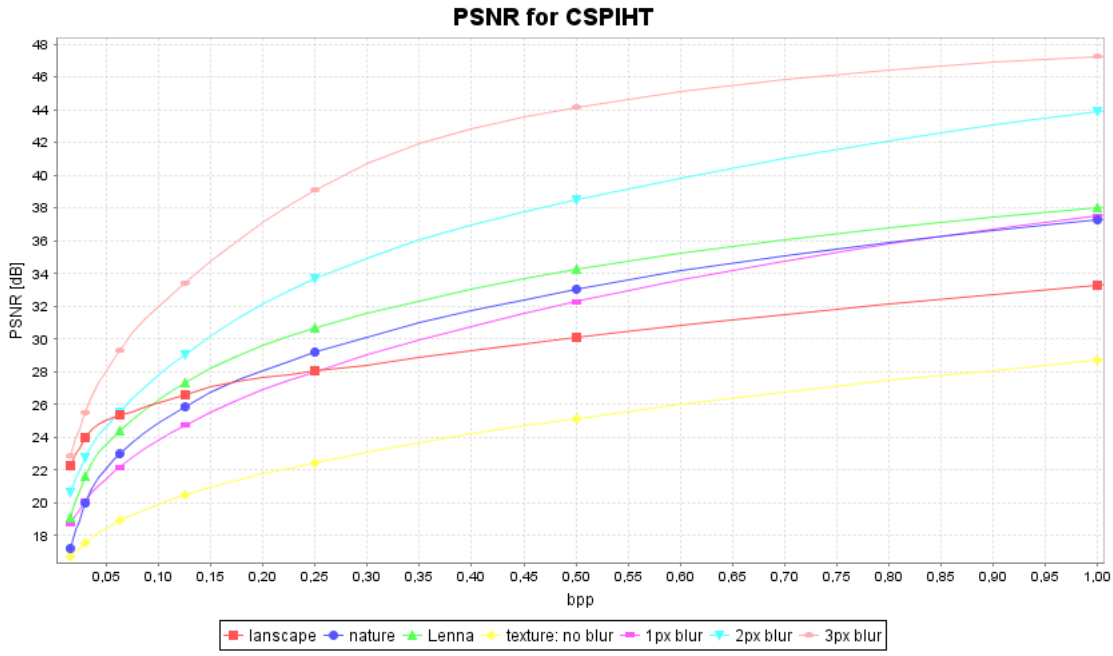


Obr. 3.10: Časová a prostorová závislost na počtu bitů (Lenna.512x512.gif, 6 kroků transformace)

3.4.2 Testování

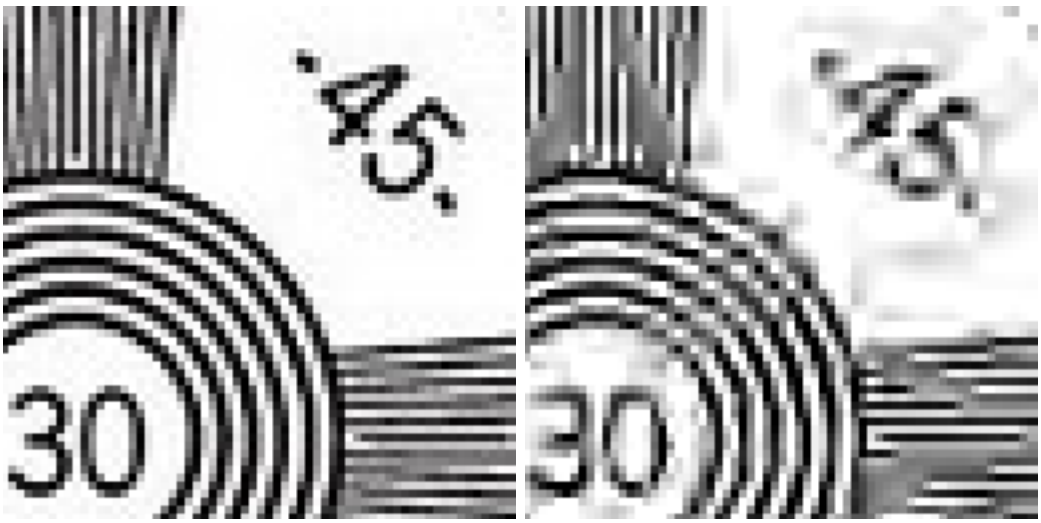
Realizovaný kodér vykazuje velice dobré výsledky a to obzvláště u barevných obrazů, kdy lze dosáhnout velice nízkých bpp při zachování obrazové informace. Pro testování účinnosti kodéru byly použity testovací obrázky ukázané v příloze D. Obrázky *nature.png*, *landscape.png* a *Lenna512.png* jsou zástupci detailních přirozených obrazů. Obrázkem *geometry.png* se testuje schopnost SPIHT algoritmu kódovat ostré přechody. Obrázek *texture.png* je nejdetailnější ze všech testovaných. Vliv množství detailů na účinnost komprese je sledován za použití posledního jmenovaného a jeho úprav pomocí gausovského vyhlazení. V následujících testech, pokud není uvedeno jinak, se pracovalo s bloky 512x512, šesti dekompozičními kroky, vlnkou CDF9/7 a kompresním algoritmem CSPIHT.

Objektivní hodnocení PSNR se bere jako objektivní parametr účinnosti komprese. Dosažené výsledky pro všechny obrazy jsou porovnány v grafu na obr. 3.11.



Obr. 3.11: Porovnání dosaženého PSNR pro obrazy různého charakteru

Z grafu je patrné, že s klesajícím bpp klesá PSNR téměř logaritmicky. Zároveň ale PSNR závisí na úrovni detailů v obraze, jak je vidět z dosažených výsledků obrazu *texture.png* a jeho vyhlazených variant.



Obr. 3.12: Zkreslení způsobené kompresí u náhlých přechodů (vlevo originál, vpravo bpp=0.25, 3x zvětšeno)

Geometrické tvary obsažené v obraze *geometry.png* slouží k testování jiných vlastností než předchozí obrazy. Tentokrát nebude počítáno PSNR, ale bude se sle-

dovat charakteristické zkreslení hran. Na obr. 3.12 je porovnání originálu a rekonstruovaného obrazu s bpp=0.25. Je vidět, že při této úrovni komprese dochází u SPIHT algoritmu k rozmazání hran a vzniku artefaktů v jejich okolí.

Subjektivní hodnocení V příloze E je možné vizuálně porovnat hodnoty v grafech se skutečným zkreslením. Demonstrační obrazy jsou také umístěny na přiloženém CD ve složce *images*. Porovnáním těchto obrazů se dojde k závěru, že zkreslení metodou SPIHT se vyznačuje postupnou ztrátou detailů a hran a postupnému „rozmazání“ těchto hran do jejich okolí. Při bližším zkoumání je patrné, že toto zkreslení má „vlnkový“ charakter, velmi podobný použité vlnce.

Naproti tomu homogenní nebo málo měnící se části jsou s klesajícím bpp zkresleny o poznání méně. Jedná se například o případ obrázku *landscape.png*, kdy les je zkreslen více než obloha. Jako pozitivní lze hodnotit nepřítomnost blokového artefaktu, pokud je možnost zpracovat obraz jako celek.

V příloze E na obr.E.3 a obr.E.4 je vidět blokový artefakt, způsobený charakterem SPIHT kodéru.

4 ZÁVĚR

Tato diplomová práce se zabývá realizací kodéru statických obrazů založeném na metodě SPIHT a vlnkové transformaci. Obsahuje také nezbytný teoretický popis použitých postupů a algoritmů. Při vypracování se postupovalo podle zadání, které bylo splněno ve všech bodech.

Teoretická část se zabývá především vlnkovou transformací, metodou SPIHT a jejími modifikacemi pro zpracování barevné informace obrazu. Vlnková transformace je moderní metoda reprezentace obrazu v prostorově-kmitočtové oblasti a její kvality, v souvislosti s kompresí obrazu již byly prověřeny u standardu JPEG2000. Ten používá biortogonální vlnky CDF 9/7 a leGall 5/3, které jsou uvažovány také v této práci. Výpočet transformace za použití těchto vlnek je proveden pomocí optimalizovaného *lifting* schématu. Tyto principiální schémata, zobrazená pomocí blokových diagramů pro obě vlnky, se nachází v příloze B na obr.B.3 a obr.B.6. Programová realizace se ale od těchto diagramů liší a tento rozdíl mezi výpočty je popsán v 2.2.9. Výpočet transformace bloku obrazových dat je pak dosažen aplikací předchozího postupu postupně na všechny řádky a pak na všechny sloupce. Vlastností separability vlnkové transformace se tedy rozšíří aplikace transformace na dvojrozměrné obrazové bloky.

Při segmentaci obrazu a aplikaci vlnkové transformace dochází na okrajích bloků k blokovému artefaktu. Ten je v této práci řešen přesahem bloků. Dosažené výsledky jsou znázorněny na grafech 3.6.

Samotný kódovací proces je realizován podle SPIHT algoritmu popsaném v [5]. K tomuto účelu byly také vytvořeny vývojové diagramy v příloze C na obrázcích. Modifikace tohoto algoritmu umožní zpracovávat i barevné roviny obrazů.

Při zpracování obrazu po segmentech se obtížně odhaduje délka výstupní posloupnosti pro bloky. Tento parametr určuje zkreslení po dekompresi, ale závislost mezi těmito parametry je nedefinovaná. Při malých hodnotách bitů na pixel dochází u bloků k různému zkreslení. Při realizaci bylo zkoumáno využití statistické analýzy zpracovávaných dat a také komprese „na zkoušku“. Žádná z těchto metod ale úplně neodstraní rozdílnost zkreslení bloků tak, aby výsledek byl srovnatelný se zpracováním nevyužívající segmentace. Dosažené výsledky jsou porovnány v grafech na obr. 3.5, 3.6 a 3.7.

Práce obsahuje popis a výstupy realizovaného kodéru. Ten byl navržen a realizován v programovacím jazyce JAVA podle zásad objektového programování. Tím je dosaženo snadné modifikovatelnosti programu a omezila se případná duplicita kódu. Byly také využity některé návrhové vzory objektového programování pro zvýšení univerzálnosti kódu.

Realizovaný kodér se na testovaných obrázcích prokázal jako velice účinný, o čemž

je možné se přesvědčit z demonstrovaných obrazů v kapitole 3.4.2 a z obrazů na přiloženém CD.

Díky modulárnímu návrhu není vyloučena možnost dalšího rozšíření kodéru a to zejména v oblasti optimalizace výpočtu a řešení nedostatků SPIHT algoritmu.

LITERATURA

[1] BARUA, S., KOTTERI K. A., BELL A. E., CARLETTA J. E. *Optimal Quantized Lifting Coefficients for the 9/7 Wavelet,* [online]. Dostupné z URL: <http://www.ece.vt.edu/fac_support/dspcl/docs /ICASSP04_Lift.pdf>.

[2] BURGET, R. *Přednášky a poznámky z předmětu Teoretická informatika,* Brno: elektronický text

[3] DARWIN, I.F. *JAVA kuchařka programátora,* Brno: Computer Press, a.s., 2006. 798 s. ISBN 80-251-0944-5

[4] Fowler, M. *SPIHT Charts,* [online]. [cit. 15.10.2007]. Dostupné z URL: <http://www.ws.binghamton.edu/fowler/fowler%20personal%20page /EE523_files/SPIHT_Charts.pdf>.

[5] SAID, A., PEARLMAN W.A. *A New Fast and Efficient Image Codec Besed on Set Partitioning in Hierarchical Trees,* [online]. IEEE Trans. on Circuits and Systems for Video Technology, vol. 6., pp. 243-250, June 1996, Dostupné z URL: <http://www.cipr.rpi.edu/staff/˜pearlman.html/papers/csvt96_sp.pdf>.

[6] BOURIDANE, A.,KHELIFI, F., AMIRA, A., KURUGOLLU, F., BOUSSAKTA, S. *A very low bit-rate embedded color image coding with SPIHT,* IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04)., vol. 3., pp. iii- 689-92 vol.3, 17-21 May 2004,

[7] KIM, B., PEARLMAN W.A. *Fast Color-Embedded Video Coding with SPIHT,* [online].
Dostupné z URL: <http://www.cipr.rpi.edu/pearlman/papers/dcc98_kp.pdf>.

[8] SHI, Y. Q., SUN, H. *IMAGE and VIDEO COMPRESSION for MULTIMEDIA ENGINEERING: fundamentals, algorithms, and standards,* USA: CRC Press LCC, 2000. 462 s. ISBN 0-8493-3491-8

[9] SMÉKAL, Z. *Číslicové zpracování signálů,* Brno: elektronický text, poslední aktualizace 13.02.2007. 151 s.

[10] SYSEL, P. *Přednášky a poznámky z předmětu Grafické a multimediální procesory,* Brno: elektronický text

[11] RAMOS, M. G., HEMAMI, S.S., TAMBURRO, M.A. *Psychovisually-based multiresolution image segmentation,* [online].
Dostupné z URL: <http://www.cipr.rpi.edu/pearlman/papers/dcc98_kp.pdf>.

[12] VALENS, C. *A Really Friendly Guide to Wavelets,* [online]. 1999-2004, poslední aktualizace 26.02.2004 [cit. 20.11.2007]. Dostupné z URL: <http://pagesperso-orange.fr/polyvalens/clemens/wavelets/wavelets.html>.

[13] VALENS, C. *The Fast Lifting Wavelet Transform,* [online]. 1999-2004, poslední aktualizace 02.12.2004 [cit. 20.11.2007]. Dostupné z URL: <http://pagesperso-orange.fr/polyvalens/clemens/lifting/lifting.html>.

[14] WEI, J., PICKERING, M., FRATER, M., ARNOLD, J., BOMAN, J., ZENG, W., *Boundary artefact reduction using odd tile length and the low pass firts convention (OCLPF),* [online]. 2001, poslední aktualizace 03.08.2001 [cit. 15.03.2008]. Dostupné z URL: <www.cs.missouri.edu/˜zengw/spie-paper4472-33_preprint.pdf>.

[15] *Wavelet forum* [online].
Dostupné z URL: <http://www.wavelet.org/phpBB2/viewforum.php?f=165>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

bpp (bit per pixel) – bitů na pixel

CWT (continuous wavelet transform) – spojitá vlnková transformace

DP dolní propust

DWT (discrete wavelet transform) – diskrétní vlnková transformace

DTWT (discrete time wavelet transform) – diskrétní vlnková transformace s diskrétním časem

DWT (Discrete Wavelet Transform) – Diskrétní vlnková transformace

GUI (Graphical User Interface) – grafické uživatelské prostředí

HP horní propust

FT Fourierova Transformace

LZW (Lempel Ziv Welch) – slovníková metoda

MSE (mean square error) – průměrná kvadratická chyba

pixel PIcture ELement

PP pásmová propust

PSNR (peak signal-noise ratio) – špičkový odstup signál-šum

RGB barevný model Red Green Blue

RLE (Run Lenght Encoding) – kódování délkou sledu

RMSE (root mean square error) – odmocněná průměrná kvadratická chyba

SNR (signal-noise ratio) – odstup signál-šum

SPIHT (Set Partitioning in Hierarchical Trees) – metoda kódování koeficientů

STFT Short Time Fourier Transform – krátkodobá Fourierova transformace

WT (wavelet transform) – vlnková transformace

YCbCr barevný model Luminance + Chrominační složky