

# Neztrátové komprimační algoritmy v počítačové grafice

Lossless Compression Algorithms in Computer Graphics

Bc. Tomáš Vogeltanz

---

Diplomová práce  
2012



Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky  
akademický rok: 2011/2012

## ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Tomáš VOGELTANZ**  
Osobní číslo: **A10434**  
Studijní program: **N 3902 Inženýrská informatika**  
Studijní obor: **Informační technologie**

Téma práce: **Neztrátové komprimační algoritmy v počítačové grafice**

Zásady pro vypracování:

1. Vytvořte literární rešerši na téma kompresní algoritmy v rastrové počítačové grafice. Zaměřte pozornost zejména na neztrátové algoritmy.
2. Seznamte se s nejčastěji používanými rastrovými formáty využívající neztrátový kompresní algoritmus a stručně popište jejich strukturu.
3. Navrhnete a vytvořte program, ve kterém budou implementovány neztrátové kompresní algoritmy pro rastrovou grafiku používané v současnosti.
4. Popište ovládání tohoto vytvořeného programu a jeho zdrojový kód.
5. Otestujte naprogramované kompresní algoritmy z hlediska kompresního poměru, rychlosti komprese a rychlosti dekomprese pro různé typy obrazových dat. Zjištěné výsledky porovnejte a vyhodnoťte.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. MURRAY, James D. a William VANRYPER. Encyklopedie grafických formátů. 2. vydání. Praha: Computer Press, 1997. ISBN 80-7226-033-2.
2. MORKES, David. Komprimační a archivační programy. Praha: Computer Press, 1998. ISBN 80-7226-089-8.
3. VEČERKA, Arnošt. Komprese dat [online]. Olomouc: Univerzita Palackého, 2008, 30.4.2008 [cit. 2011-12-18]. Dostupné z: <http://phoenix.inf.upol.cz/esf/ucebni/komprese.pdf>
4. TIŠNOVSKÝ, Pavel. Seriál Grafické formáty. Root.cz [online]. Internet Info, 7.9.2006 [cit. 2012-02-01]. Dostupné z: <http://www.root.cz/serialy/graficke-formaty/>
5. STRACHOTA, Pavel. Ukládání a komprese obrazu [online]. Praha: FJFI ČVUT, 2010, 15.9.2010 [cit. 2012-01-08]. Dostupné z: [http://saint-paul.fjfi.cvut.cz/base/public-filesystem/admin-upload/POGR/POGR1/07.ukladani\\_a\\_komprese\\_obrazu.pdf](http://saint-paul.fjfi.cvut.cz/base/public-filesystem/admin-upload/POGR/POGR1/07.ukladani_a_komprese_obrazu.pdf)

Vedoucí diplomové práce:

**Ing. Pavel Pokorný, Ph.D.**

Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce:

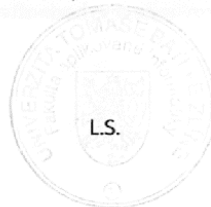
**24. února 2012**

Termín odevzdání diplomové práce:

**21. května 2012**

Ve Zlíně dne 24. února 2012

prof. Ing. Vladimír Vašek, CSc.  
*děkan*



doc. Mgr. Roman Jašek, Ph.D.  
*ředitel ústavu*

## ABSTRAKT

Tato práce je zaměřena na neztrátové kompresní algoritmy v počítačové grafice. Nejprve jsou vysvětleny způsoby reprezentace obrazu a základní pojmy komprese. Dále jsou popsány neztrátové kompresní algoritmy a grafické formáty, které využívají neztrátovou kompresi. V rámci diplomové práce byla vytvořena aplikace, která umožňuje tyto formáty testovat. Popis této aplikace je zahrnut do úvodu praktické části. Nakonec jsou uvedeny výsledky testů neztrátových kompresních algoritmů i s jejich vyhodnocením.

Klíčová slova:

Neztrátová komprese, Testy neztrátových kompresních algoritmů, Kompresní poměr, Doba komprese, Doba dekomprese, Časová efektivita komprese, BMP, GIF, HD Photo, JPEG 2000, JPEG-LS, Lossless JPEG, OpenEXR, PCX, PNG, TGA, TIFF, WebP, JBIG, RLE, LZ77, LZW, Vlnková transformace, Prediktivní metody, PPM, Huffmanovo kódování, Shannon-Fannovo kódování, Aritmetické kódování.

## ABSTRACT

This thesis is focused on lossless compression algorithms in computer graphics. First, ways of image representation and basic compression terms are explained. Next, the lossless compression algorithms and graphic formats, that use lossless compression are described. An application, that allows to test these formats was created within diploma thesis. The description of this application is included into introduction of the practical part. Finally, the results of tests of lossless compression algorithms and their evaluating are presented.

Keywords:

Lossless Compression, Tests of Lossless Compression Algorithms, Compression Ratio, Compression Time, Decompression Time, Time Efficiency of Compression, BMP, GIF, HD Photo, JPEG 2000, JPEG-LS, Lossless JPEG, OpenEXR, PCX, PNG, TGA, TIFF, WebP, JBIG, RLE, LZ77, LZW, Wavelet Transformation, Prediction Methods, PPM, Huffman Coding, Shannon-Fano Coding, Arithmetic Coding.

Na tomto místě bych rád poděkoval panu Ing. Pavlu Pokornému, Ph.D. za jeho věcné připomínky a čas při konzultaci této práce a také všem, kteří mě při tvorbě této práce podporovali.

**Prohlašuji, že**

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

**Prohlašuji,**

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....  
podpis diplomanta

**OBSAH**

<b>ÚVOD.....</b>	<b>10</b>
<b>I TEORETICKÁ ČÁST .....</b>	<b>11</b>
<b>1 REPREZENTACE OBRAZU .....</b>	<b>12</b>
1.1 BAREVNÉ PROSTORY .....	12
1.1.1 RGB.....	12
1.1.2 RGBA.....	13
1.2 REPREZENTACE RASTROVÉHO OBRAZU.....	14
1.2.1 Paleta barev .....	14
<b>2 ZÁKLADNÍ POJMY KOMPRESI.....</b>	<b>15</b>
2.1 TYPY KOMPRESI.....	15
2.2 HLAVNÍ PARAMETRY VÝKONU KOMPRESNÍCH ALGORITMŮ.....	17
<b>3 NEZTRÁTOVÉ KOMPRESNÍ ALGORITMY .....</b>	<b>19</b>
3.1 PEXELOVÉ ZHUŠŤOVÁNÍ .....	19
3.2 RLE .....	19
3.2.1 Bitová úroveň kódování .....	20
3.2.2 Bytová úroveň kódování .....	20
3.2.3 Pixelová úroveň kódování.....	21
3.2.4 Tříbytové kódování .....	21
3.2.5 Vertikální replikační pakety .....	22
3.3 LZ77 .....	22
3.4 LZW .....	23
3.5 HUFFMANOVO KÓDOVÁNÍ .....	24
3.5.1 CCITT kódování .....	25
3.6 SHANNON-FANOVO KÓDOVÁNÍ .....	26
3.7 ARITMETICKÉ KÓDOVÁNÍ .....	27
3.8 JBIG.....	28
3.9 JBIG 2.....	29
3.10 VLNKOVÁ TRANSFORMACE .....	30
3.10.1 Diskrétní vlnková transformace .....	31
3.10.2 Celočíselná vlnková transformace .....	32
3.11 PREDIKTIVNÍ METODY .....	33
3.11.1 PPM.....	34
<b>4 FORMÁTY VYUŽÍVAJÍCÍ NEZTRÁTOVOU KOMPRESI.....</b>	<b>36</b>

4.1	BMP .....	36
4.2	GIF .....	37
4.3	PCX .....	38
4.4	PNG .....	39
4.5	TGA .....	41
4.6	TIFF .....	42
4.7	JPEG 2000 .....	44
4.8	HD PHOTO .....	45
4.9	JPEG-LS .....	47
4.9.1	Lossless JPEG .....	48
4.10	OPENEXR .....	49
4.11	WEBPLL .....	50
<b>II</b>	<b>PRAKTICKÁ ČÁST .....</b>	<b>52</b>
<b>5</b>	<b>APLIKACE PRO TESTOVÁNÍ KOMPRESNÍCH ALGORITMŮ .....</b>	<b>53</b>
5.1	MOŽNOSTI APLIKACE .....	53
5.1.1	Menu Soubor .....	54
5.1.2	Menu Obrázek .....	59
5.1.3	Menu Nápověda .....	61
5.2	POPIS ZDROJOVÝCH KÓDŮ APLIKACE .....	62
5.2.1	Třída Obrazek .....	63
5.2.2	Třída FreeImageRozhraniClass .....	65
5.2.3	Třída MainFrame .....	66
<b>6</b>	<b>TESTOVÁNÍ KOMPRESNÍCH ALGORITMŮ .....</b>	<b>69</b>
6.1	FOTOGRAFIE .....	70
6.1.1	Fotografie formátu JPEG .....	70
6.1.2	Fotografie formátu RAW .....	71
6.1.3	Fotografie v odstínech šedi .....	72
6.1.4	Fotografie s vysokou dynamikou barev .....	74
6.1.5	Fotografie s vysokou dynamikou barev v odstínech šedi .....	75
6.2	OBRÁZKY .....	76
6.2.1	Obrázky ve 24 bitové hloubce .....	76
6.2.2	Obrázky v 8 bitové hloubce .....	77
6.2.3	Obrázky ve 4 bitové hloubce .....	79
6.2.4	Obrázky v 1 bitové hloubce .....	80
6.2.5	Obrázky v odstínech šedi .....	82
6.3	TEXT .....	83
6.3.1	Text v 1 bitové hloubce .....	83
6.3.2	Text v odstínech šedi .....	84
	<b>ZÁVĚR .....</b>	<b>86</b>
	<b>ZÁVĚR V ANGLIČTINĚ .....</b>	<b>88</b>

<b>SEZNAM POUŽITÉ LITERATURY.....</b>	<b>90</b>
<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK .....</b>	<b>97</b>
<b>SEZNAM OBRÁZKŮ .....</b>	<b>102</b>
<b>SEZNAM TABULEK.....</b>	<b>104</b>
<b>SEZNAM PŘÍLOH.....</b>	<b>105</b>

## ÚVOD

Tato práce je zaměřena na neztrátové kompresní algoritmy v počítačové grafice - konkrétně na rastrové obrázky ve 2D. Na úvod jsou popsány možnosti reprezentace obrazu včetně typů barevných prostorů a palety barev.

V rámci druhé kapitole jsou vysvětleny pojmy používané v souvislosti s kompresí. Zde jsou uvedeny základní typy kompresí a parametry, které se vyskytují při vyhodnocení kvality kompresních algoritmů.

Ve třetí kapitole jsou charakterizovány neztrátové kompresní algoritmy. Zastoupeny zde jsou jak starší komprese jako Huffmanovo kódování, RLE, LZ77, LZW, tak komprese novější, ke kterým patří JBIG, vlnková transformace a prediktivní metody.

Čtvrtá kapitola objasňuje problematiku formátů, které využívají neztrátové kompresní algoritmy. Jsou zde popsány struktury těchto formátů a zmíněny neztrátové kompresní algoritmy, které tyto formáty využívají. Tato kapitola obsahuje formáty, jejichž použití je v současnosti již méně časté (např. PCX, TGA), ale také ty, které se hojně využívají (např. GIF, PNG), a ty novější, které si stále hledají cestu do podvědomí běžného uživatele (např. JPEG 2000, JPEG-LS, HD Photo).

Dále je v práci popsána vytvořená aplikace pro testování neztrátových kompresních algoritmů. V páté kapitole je znázorněn výčet možností této aplikace a je upřesněna funkcionality nejdůležitějších tříd.

Na konci práce jsou uvedeny výsledky testů neztrátových kompresních algoritmů pro dané typy obrázků. Tyto výsledky jsou vyhodnoceny a na jejich základě jsou kompresní algoritmy vzájemně porovnány.

Na CD jsou umístěny dvě verze této práce - standardní (určená pro tisk) a rozšířená. Rozšířená verze detailněji popisuje reprezentaci obrazu, neztrátové kompresní algoritmy, formáty obrazových souborů a obrázky použité při testech algoritmů. Dále lze na CD najít zdrojové kódy aplikace pro testování kompresních algoritmů spolu se spustitelným souborem a použitými externími komponenty.

## **I. TEORETICKÁ ČÁST**

## 1 REPREZENTACE OBRAZU

Reprezentace obrazu se v počítačové grafice dělí na rastrovou a vektorovou [1].

Rastrová reprezentace obrazu popisuje obraz různými způsoby, především pomocí matice pixelů. U matice pixelů vzniká problém konečného rozlišení a vysoké paměťové náročnosti (resp. větší velikosti souboru). Kvůli velké velikosti souboru s rastrovým obrázkem je také snaha o kompresi rastrových dat. [2] Rastrovou grafikou lze však zobrazit i složité předlohy a práce s těmito daty je rychlejší [3].

Další možností reprezentace rastrového obrazu je pomocí kvadrantového stromu. Zde se využívá koherence ve vodorovném a svislém směru. Pomocí této metody se úsporně kódují větší souvislé plochy jedné barvy (resp. menšího počtu barev). Princip této metody je adaptivní - přispůsobuje se datům. Pokud je barva všech pixelů obrazu stejná uloží se informace o této barvě, pokud ne celý obraz je rozdělen na kvadranty. Následně když je barva pixelů v kvadrantu stejná, tak dojde k zápisu této barvy, když ne dojde opět k rozdělení tohoto kvadrantu na menší kvadranty a proces se rekurzivně opakuje. [4]

Vektorová reprezentace obrazu je popis geometrických objektů a jejich vlastností (polohy, barvy, překrývání atd.). Díky tomu, že obsahuje pouze parametry objektů, nezávisí na rozlišení zobrazovacího zařízení a obrázky lze libovolně škálovat při zachování kvality. [2] U vektorového formátu se komprese příliš nepoužívá, neboť zde má malý efekt. Pokud je na soubor ve vektorovém formátu komprese použita, tak dochází ke kompresi celého souboru a komprese musí být neztrátová. [3]

### 1.1 Barevné prostory

Barevné prostory určují, jakým způsobem je definována cílová barva [3]. Jsou zde zmíněny jen dva nejpoužívanější barevné prostory - další jsou uvedeny v rozšířené verzi této práce.

#### 1.1.1 RGB

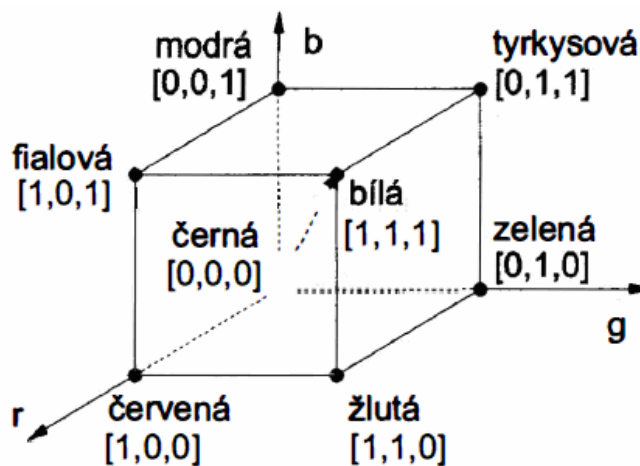
Tento barevný prostor se v grafických formátech používá nejčastěji [3].

Základní vlastností tohoto barevného prostoru je součtové, aditivní skládání barev - čím více barev je zkombinováno (sečteno), tím světlejší je výsledek [1]. Aditivní barevné prostředí nepotřebuje žádné vnější světlo [3]. Barvy bývají uváděny v celočíselném rozsahu 0-255, což odpovídá kódování každé ze složek RGB (Red - červená, Green - zelená, Blue -

modrá) v jednom Bytu. Hodnota 0 znamená, že složka není zastoupena, maximální hodnota 255 indikuje, že složka nabývá své největší intenzity. [1]

Kombinací červené, zelené a modré barvy zobrazovanou displejem v plné intenzitě, lze získat barvu bílou. [1] Naopak při nulové intenzitě všech těchto barev je možné obdržet černou barvu [3]. Šedá barva se vytvoří kombinací všech tří barev se shodnou intenzitou. Lidské oko vnímá různým způsobem intenzitu jednotlivých barevných složek, a proto se používá pro výpočet jasu empirický vztah: [1]

$$I = 0,299 R + 0,587 G + 0,114 B$$



Obr.1 Geometrická reprezentace prostoru RGB [1]

### 1.1.2 RGBA

Zkratka RGBA (resp. RGBA) je používána pro vyjádření skutečnosti, že barevný obraz zapsaný v prostoru RGB je doplněn informací o průhlednosti. Každý barevný bod takového obrazu s sebou nese skalární údaj (např. v rozmezí 0-1), který určuje, v jakém rozsahu pokrývá barva plochu obrazového bodu. Hodnota 0,0 znamená neprůhledný barevný bod, maximální hodnota 1,0 zcela průhledný. Při zobrazení samotného obrazu nemá složka  $\alpha$  ( $\alpha$ -kanál) význam. Pojem RGBA neznámá změnu barevného prostoru, nýbrž přidání další informace. Složka  $\alpha$  se ukládá do rozsahu jednoho i více Bytů. Používá se zejména při kombinování více obrazů do jednoho celku. [1]

## 1.2 Reprezentace Rastrového obrazu

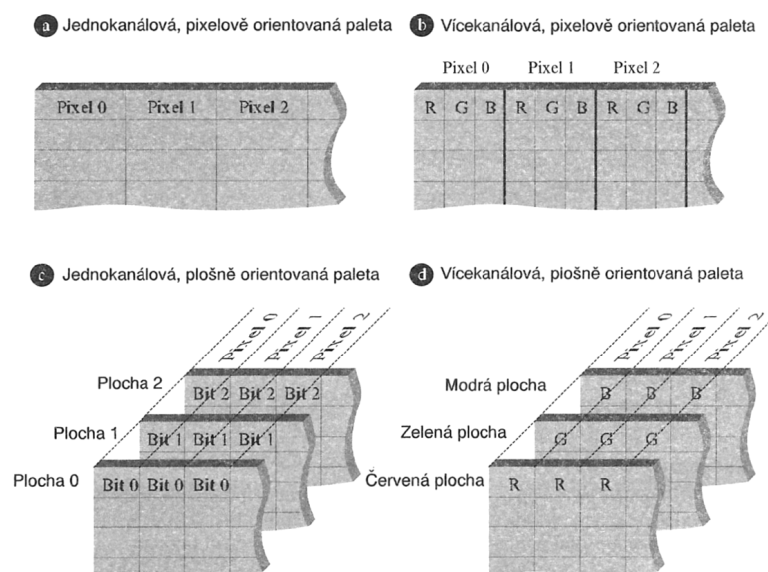
Vyjádření barevných složek pomocí 3 Bytů je v současnosti nejběžnější. Používají se ale i jiná kódování, např. 12 nebo 16 bitů na barevný kanál. [1]

Indexový mód je spojen s používáním tzv. barevné palety. U takového obrazu nereprezentuje hodnota pixelu přímo barvu, ale je ukazatelem do tabulky - barevné palety. [1] Indexovým módem lze ušetřit místo na disku, protože tři hodnoty barev o celkové velikosti 3 Byty jsou nahrazeny jedním indexem o velikosti 1 Byte. [3]

Další možností je reprezentace obrazu v odstínech šedi. Každý bod v obraze reprezentuje buď přímo odstín šedi (static grey), nebo je opět odkazem do palety (greyscale). [1]

### 1.2.1 Paleta barev

Barevná paleta je převodní tabulka, která určuje konkrétní barvu pixelu daným indexem [1]. Například osmibitová hodnota pixelu může reprezentovat 256 různých barev a je tedy doprovázena 256 prvkovou paletou. Pokud má předloha méně barev než je jejich maximální počet v paletě, potom jsou všechny nepoužité prvky v ní nastaveny na nulu. Nevyužité prvky v paletě většinou nejsou uspořádány ve shlcích podle nějakého systému, také většinou nezačínají na nulové pozici. Používané typy palet lze vidět na *Obr.2*. [3]



Obr.2 Typy palet [3]

## 2 ZÁKLADNÍ POJMY KOMPRESSE

Kompresse (resp. komprimace) je proces, který se využívá pro zredukování fyzické velikosti bloku informací [3]. Jinými slovy: Soubory jsou zakódovány do takové podoby, kdy je jejich velikost menší než velikost před kompresí [5]. Kódování znamená způsob reprezentace dat při jejich uložení v souboru, paměti apod. [6]

Každý kompresní algoritmus je navržen tak, že hledá a využívá pro kompresi dat určitý řád v uložených datech. Tímto řádem může být opakování sekvencí znaků, frekvence výskytu jednotlivých znaků, identifikace dlouhých bloků stejných dat a další. [5]

Data různého charakteru vyžadují rozdílný přístup k jejich kompresi. Je zřejmé, že grafická data se svým charakterem budou lišit např. od textových, a že rozdílnost těchto dat způsobí rozdílné možnosti aplikovatelnosti některých kompresních algoritmů. Např. RLE lze více využít u grafických formátů než u textových - v textu se málokdy objevují stejné znaky za sebou. [5]

### 2.1 Typy komprese

Kompresní algoritmy se v základu dělí na neztrátové a ztrátové.

Neztrátová komprese je způsob komprese, při které nedochází ke ztrátě informace - při dekompresi dostáváme stejná data, jako před kompresí. Místo termínu bezztrátová komprese se také používají označení přesná komprese nebo vratná komprese. [6] Všeobecně tyto metody nedosahují tak dobrých kompresních poměrů jako metody ztrátové, ale používají se i jako pomocné algoritmy ke kódování obrazu a zvuku [7].

Ztrátová komprese je způsob komprese, při které jsou výchozí hodnoty poněkud pozměněny nebo některé méně významné hodnoty jsou zanedbány, aby se dosáhlo vyššího kompresního poměru. Dekompresí dostáváme v tomto případě jiné hodnoty, než před kompresí. Ztrátové metody mají uplatnění v kompresi obrazu a zvuku. [6]

Další možnosti dělení kompresních algoritmů podle [3]:

- Fyzická × Logická
- Symetrická × Asymetrická
- Neadaptivní × Adaptivní × Semiadaptivní

Logická komprese používá logické substituce sekvence znaků jinou, úspornější řadou. Konkrétním příkladem jsou zkratkovaná slova jako Čedok (nahrazující někdejší plný název Československá dopravní kancelář) nebo Svazarm (Svaz pro spolupráci s armádou). [5]

Fyzická komprese probíhá bez ohledu na logiku dat, se kterými se manipuluje. Vytváří se nová sekvence znaků (Bytů, bitů apod.), jejíž vztah k původním datům lze rozpoznat výhradně s použitím dekompresního algoritmu. Bez znalosti tohoto dekompresního algoritmu je informační hodnota komprimovaných dat nulová. [5] Algoritmy v počítačové grafice jsou založeny výhradně na fyzické kompresi [3].

O symetrickou kompresi se jedná, pokud je doba (a tím většinou i počet a druh operací) potřebná pro kompresi i dekompresi dat přibližně stejná [5].

U asymetrické komprese není čas potřebný pro kompresi a dekompresi stejný. Většina kompresních algoritmů provede větší množství operací při kompresi dat. Asymetrické algoritmy, jejichž práce je delší při dekompresi, nejsou tolik rozšířené. [5]

Neadaptivní algoritmy jsou určeny výhradně pro kompresi specifického druhu dat. Většinou obsahují předdefinované slovníky nebo řetězce znaků, o kterých je známo, že jejich pravděpodobnost výskytu v souborech dat je vysoká. Použití neadaptivního algoritmu na vhodný druh dat je velice účinné co do dosaženého kompresního poměru, ale i času potřebného pro kompresi a dekompresi dat. Konkrétním příkladem neadaptivního kompresního algoritmu je tzv. Huffmanovo (resp. CCITT) kódování. [5]

Adaptivní algoritmus je naproti tomu schopen dosáhnout určité nezávislosti na komprimovaných datech. Takové algoritmy neobsahují žádné statické slovníky řetězců. Algoritmy si budují tyto slovníky pro každý komprimovaný soubor dat dynamicky v průběhu kódování. Obecně lze říci, že adaptivní algoritmy platí za svou přizpůsobivost a větší šíři použití menší rychlostí ve srovnání se specializovanými neadaptivními algoritmy. Příkladem adaptivního kompresního algoritmu je LZW algoritmus. [5]

Semiadaptivní algoritmus je kombinací adaptivního a neadaptivního algoritmu. Tato metoda provede úvodní průchod dat kvůli vybudování slovníku a poté provede druhý průchod, při kterém se uskuteční vlastní kódování. Při použití této metody dojde k vybudování optimálního slovníku ještě před tím, než dojde ke kódování. [3]

## 2.2 Hlavní parametry výkonu kompresních algoritmů

Hlavními parametry výkonu kompresních algoritmů jsou podle [5]:

- Kompresní poměr nebo podle [8] a [9] je možné také udávat Faktor komprese
- Doba komprese
- Doba dekomprese
- Poměr mezi dobou komprese a kompresním poměrem

Kompresní poměr bývá udáván jako:

- Poměr mezi velikostí komprimovaných a nekomprimovaných dat. Podle tohoto schématu pokud komprimační program zkomprimuje soubor o původní délce 200 kB na 50 kB, je udáván kompresní poměr 1:4, popř. 25 %. Čím menší číselné vyjádření, tím lepší výsledek komprese. [5]
- Počet bitů na Byte, tj. kolik je při kompresi v průměru zapotřebí bitů pro uložení jednoho Bytu výchozího souboru. Předcházející příklad komprese souboru o původní délce 200 kB na 50 kB v komprimovaném stavu by byl tedy charakterizován kompresním poměrem 2 bpB - podle vzorce:

$$\frac{l_c}{l_u} \cdot 8$$

$l_c$  - velikost komprimovaného souboru

$l_u$  - velikost nekomprimovaného souboru

8 - počet bitů v Bytu

Samotný výpočet tedy vypadá následovně:  $\frac{l_c}{l_u} \cdot 8 = \frac{50}{200} \cdot 8 = \frac{1}{4} \cdot 8 = 2$  bpB. [6]

Faktor komprese bývá udáván jako poměr mezi velikostí nekomprimovaných a komprimovaných dat [8] [9]. Předcházející příklad komprese souboru o původní délce 200 kB na 50 kB v komprimovaném stavu by byl charakterizován kompresním poměrem 4:1 nebo 75 %. Čím vyšší číselné vyjádření, tím lepší výsledek komprese. [5]

Některé publikace, jako např. [5] zaměřují pojmy kompresní poměr a faktor komprese - resp. faktor komprese je v této publikaci vnímán jako další možnost vyjádření kompresního poměru a tento samotný výraz (tj. „Faktor komprese“) se zde nevyskytuje.

Doba dekomprese, tedy čas potřebný k rozbalení komprimovaného souboru do původní podoby, je ze všech zmíněných parametrů asi nejméně důležitý. Doba potřebná pro dekompresi bývá téměř u všech komprimačních programů kratší než doba potřebná ke kompresi, a proto tento parametr nebývá tím omezujícím nebo kritickým faktorem při výběru vhodného kompresního algoritmu.

Poměr mezi dobou komprese a kompresním poměrem často nebývá v testech kompresních algoritmů vůbec zmiňován. Vychází z požadavku dosažení co nejlepšího kompresního poměru za „přiměřený“ čas.

Při testování rychlosti algoritmu je nutné brát v úvahu parametry PC, a to především:

- Informace o procesoru (typ, taktování, počet jader, technologie výroby, materiál apod.)
- Velikost a přístupovou dobu paměti RAM
- Přístupovou dobu, rychlost čtení a zápisu pevného disku
- Stav obsazení pevného disku a jeho fragmentaci
- Použití nebo zakázání vyrovnávací paměti CACHE
- Typ operačního systému, pod kterým byly kompresní algoritmy testovány

Pokud má být test kompresních algoritmů opravdu použitelný, měl by kromě názvu těchto testovaných algoritmů obsahovat minimálně následující údaje:

- Typy souborů, na kterých bylo testování prováděno
- Nekomprimovanou velikost původního souboru
- Komprimovanou velikost souboru
- Typ hardware (především procesoru), na kterém byly testy prováděny
- Operační systém, ve kterém byl test proveden

### 3 NEZTRÁTOVÉ KOMPRESNÍ ALGORITMY

Rastrové obrazy se vyznačují vysokou paměťovou náročností, která roste kvadraticky s jejich rozlišením. Na rozdíl od komprese obecných souborů lze vycházet z vlastností a charakteristických rysů konkrétního rastrového obrazu. [1]

#### 3.1 Pixelové zhušťování

Pixelové zhušťování není typická metoda datové komprese - v podstatě ji ani nelze za kompresi označit. Je to účinný způsob ukládání dat v soustředných Bytech v paměti.

Pokud jsou v obrázkových datech obsaženy čtyři bity na pixel, tak je velmi pohodlné ukládat každý pixel do Bytu, protože Byte je nejmenší adresovatelná část paměti na většině počítačových systémů. Použitím této metody je však polovina Bytu úplně nevyužitá. Data předlohy, která obsahuje 4096 čtyřbitových pixelů tak požadují 4096 Bytů paměti. [3]

Paměť lze ušetřit tím, že namísto ukládání jednoho 4-bitového pixelu na jeden Byte, se do jednoho Bytu uloží dva takové 4-bitové pixely. Velikost paměti, do které se má uložit 4096 4-bitových pixelu potom bude mít velikost 2048 Bytů, tedy přesnou polovinu. [3]

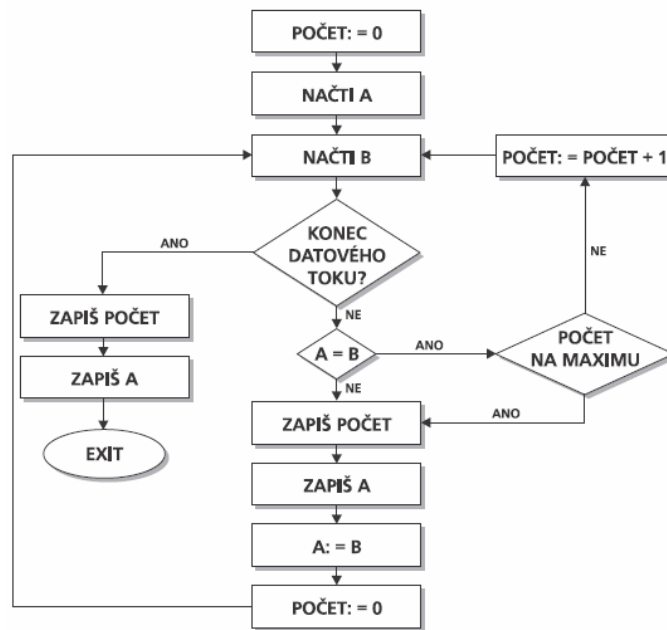
#### 3.2 RLE

Algoritmus Run-Length Encoding (RLE) je do češtiny překládaný jako proudové kódování [5]. Základním principem komprese je to, že se zapíše nejprve počet opakujících se totožných hodnot a poté hodnota samotná [1]. Řetězec opakujících se znaků se nazývá proud. Tento proud znaků je vždy zkomprimován do formy jednoho paketu RLE. [5]

Výhodou tohoto řešení je jednoduchost algoritmu, snadné použití a vysoká kompresní i dekompresní rychlost. Nevýhodou úzká oblast dat, na kterých tato metoda dosahuje dobré kompresní poměry. [5]

Pokud se u sousedních pixelů ve směru ukládání (obvykle po řádcích) často neopakují stejné hodnoty, dochází k záporné kompresi, kdy se zvýší velikost komprimovaných dat oproti jejich původní formě. [2] [5]

Základní struktura algoritmu RLE je zobrazena na *Obr.3*. [5]



Obr.3 Základní schéma algoritmu RLE [5]

### 3.2.1 Bitová úroveň kódování

Bitová úroveň kódování metodou RLE rozeznává pouze dva proudy znaků - proud jedniček a proud nul [5] - a ignoruje zarovnání na Byte nebo na slovo [3].

Jedná se o jediný Byte logicky rozdělený do dvou částí. Nejvýznamnější bit každého Bytu určuje proudovou hodnotu (tj. 1 nebo 0) a sedm méně významných bitů představuje proudové číslo udávající počet opakování proudové hodnoty snížený o 1. Je zřejmé, že délka proudu musí být vždy v rozsahu 1-128 znaků, neboť do sedmi bitů lze zapsat číslo 0-127. Pokud původní nekomprimovaná data obsahují proud delší než 128 znaků, musí být tento proud při kódování rozdělen na dva nebo více proudů. [5]

### 3.2.2 Bytová úroveň kódování

Tato modifikace kóduje proudy opakujících se Bytových hodnot a naopak si nevšimá dělení na bity nebo hranic 16 bitových (případně 32 bitových) slov. [5]

Paket RLE se v tomto případě skládá ze dvou Bytů. První Byte udává proudové číslo v rozsahu 0-255 a druhý Byte proudovou hodnotu. Její rozsah je opět 0-255. [5]

1	čítač	hodnota	hodnota se opakuje $(1 + \text{čítač}) \times$
0	hodnota		přímý zápis jediné 7-bitové neopakující se hodnoty
10000000		hodnota	zápis neopakující se hodnoty větší než binárně 10000000

Obr.4 RLE paket na Bytové úrovni [1]

Při kódování pixelů definovaných jedním Bytem rozlišíme příznak opakování hodnotou nejvyššího bitu (viz *Obr.4*) [1]. Při nastavení nejvýznamnějšího bitu prvního Bytu na hodnotu 0 následuje za proudovým číslem přesný proud znaků, který se čte v té podobě, v jaké je zapsán. [5]

Efektivita komprese klesá při zpracování samostatně se vyskytujících hodnot s nenulovým nejvyšším bitem. Pokud zapisujeme obrázky definované pomocí palety, je vhodné uspořádat paletu tak, aby méně používané odstíny byly umístěny na konci palety. [1]

### 3.2.3 Pixelová úroveň kódování

Modifikace RLE na pixelové úrovni se používá u složitějších obrázků, kdy je jediný pixel reprezentován více než jedním Bytem dat. Velikost paketu RLE se v tomto případě liší podle počtu Bytů na pixel. Pokud např. obrázek podporuje schéma dat 3 Byty na pixel, bude mít paket RLE čtyři Byty - první Byte jako proudové číslo a následující tři Byty proudové hodnoty. Kódovací metoda zůstává stejná jako u úrovně s jedním Bytem. [5]

### 3.2.4 Tříbytové kódování

Základní myšlenkou další metody je použití tří Bytů jako jediného paketu RLE. První Byte v tomto případě obsahuje příznakovou hodnotu, následující dva Byty tvoří klasický paket RLE. Druhý Byte opět představuje proudové číslo a třetí proudovou hodnotu. Kódování dat předlohy probíhá tak, že pokud komprimační program zjistí jedno-, dvou- nebo tří- Bytový proud, zapíše se tyto hodnoty do komprimovaného toku dat přímo. [5]

Při dekompresi se nejprve posoudí, zda první Byte paketu RLE obsahuje příznakovou hodnotu nebo nikoli. Pokud se jedná o příznak, expanduje se paket podle údajů proudového čísla a proudové hodnoty. Nejedná-li se o příznak, přidá se tento Byte přímo do nekomprimované podoby souboru. [5]

Problém nastává, pokud nekódovaný tok dat obsahuje hodnotu, která se shoduje s hodnotou vyhrazenou pro příznak. Každý takový znak musí být kódován do paketu se třemi Byty s proudovým číslem 0 (odpovídající jedinému znaku). [5]

S odkazem na předchozí odstavec podle [5] str.28: „Jako příznakovou hodnotu je tudíž nutné zvolit znak s nejmenší pravděpodobností výskytu v datech předlohy, aby v důsledku jeho častého výskytu nedocházelo ke zhoršení kompresního poměru“. Ovšem toto není zcela přesné - znak musí mít nejmenší pravděpodobnost výskytu samostatně a zároveň nejmenší pravděpodobnost výskytu přesně 2× za sebou. Když totiž data předlohy obsahují např. 5 hodnot čísla 2, které se vždy vyskytují samostatně, číslo 8 opakující se 10× za sebou a dále všechny ostatní možné hodnoty, které jsou samostatně obsaženy častěji než hodnota 2, bylo by podle výše uvedené citované věty vhodné použít právě číslo 2 pro příznak opakování. To ovšem zapříčiní, že se těchto 5 hodnot čísla 2 uloží do 15 (5× ve formátu 2 0 2) a ne do pouhých 5 Bytů - pro opakování čísla 8 by se použily 3 B (ve formátu 2 9 8). Pokud se však jako příznak použije číslo 8, tak se číslo 2 zakóduje do 5 Bytů a pro zápis opakování čísla 8 se použijí 3 B (ve formátu 8 9 8) - výsledek tedy bude o 10 Bytů lepší. Podobně nepřesná formulace jako výše citovaná věta se vyskytuje i v [3] str.148: „RLE algoritmus musí proto používat takovou hodnotu příznaku, která se jen výjimečně vyskytuje v nekomprimovaném toku dat“.

### 3.2.5 Vertikální replikační pakety

Tato modifikace komprese RLE představuje forma tzv. vertikálních replikačních paketů neboli paketů s opakovanými vzorkovými řádky. Zlepšeného kompresního poměru se dosahuje vyjádřením toho, zda se opakuje celý předchozí řádek. Je zřejmé, že se tento druh kódování hodí pouze na ty druhy dat, kde se opakování celých řádků předpokládá. [5]

## 3.3 LZ77

Kompresní část algoritmu LZ77 funguje tak, že se pokouší vyhledat co nejdelší opakující se posloupnosti znaků. Pokud takovou opakující se posloupnost nalezne, zapíše na výstup pouze odkaz na předcházející výskyt řetězce. [5]

Například vstupní řetězec „leze po železe“ se zakóduje do podoby „leze po že[10,4]“. Znaky [10,4] je třeba považovat za schématicky zapsaný offset udávající, že dekodér má z předcházejících deseti znaků vybrat první čtyři. [5]

Podstatou tohoto typu komprese je tzv. posuvné okno (sliding window), které obsahuje koncovou část již přečteného (a zkomprimovaného) textu. V tomto okně se kompresní algoritmus snaží nalézt co nejdelší podřetězec odpovídající řetězci na vstupu. Pokud se to podaří, zakóduje jej v podobě odkazu na tento výskyt. Odkaz musí obsahovat ukazatel na začátek podřetězce a jeho délku. [5] [9]

Posuvné okno obsahuje dvě části: prohlížecké okno a aktuální okno. Na začátku algoritmus nastaví posuvné okno tak, aby začátek vstupu obsahovalo aktuální okno. [5]

Potom pokaždé v posuvném okně najdeme co nejdelší počáteční podřetězec (předponu) řetězce z aktuálního okna začínající v prohlížeckém okně. Tento podřetězec se pak zakóduje v podobě ukazatele na počátek podřetězce v prohlížeckém okně a jeho délku. [5]

Existuje několik způsobů jak rozeznat dvojici znaků nesoucí informaci o dvojici (ukazatel, délka) od jednotlivých znaků na výstupu. Nejpoužívanějším je zápis LZSS (Storer, Szymanski): <1, ukazatel, délka> nebo <0, jednotlivý znak>. Nula na počátku signalizuje jednotlivý znak, jednička informační dvojici znaků. [5]

Dekomprese souboru zkomprimovaného metodou LZ77 je velice jednoduchá a rychlá. Vždy, když dekompresní algoritmus narazí na offset udávající ukazatel a délku řetězce, tak tento řetězec zkomprimuje na výstup. [5]

### 3.4 LZW

Tato komprese má velmi dobrý kompresní poměr, rychlou kompresi i dekompresi, malé nároky na paměť [5] a pracuje s celými čísly [3]. Další výhodou je adaptivní metoda vytvářející dynamický substituční slovník. [5] LZW také komprimuje data do Bytů a ne do slov, a proto může být kódovaný výstup jak v systému velký endián, tak v systému malý endián. Na problémy bitového a výplňového uspořádání však lze narazit stejně. [3]

Za nevýhodu se dá považovat rychlý nárůst slovníkových kódů odkazujících na řetězce původního souboru, což vede v některých případech k zaplnění paměti určené pro slovníkové kódy. Z toho pak vyplývá smazání aktuálního slovníku a jeho nové vytváření bez možnosti použití smazaných odkazů. V případě větších slovníků zase narůstá doba vyhledání řetězce ve slovníku. [5]

Základním principem je vyhledávání stejných posloupností Bytů v originálním souboru. Pomocí odkazů na tyto posloupnosti dat algoritmus buduje datový slovník. Každý další

výskyt takové posloupnosti se zakóduje buď jako odkaz na předcházející výskyt posloupnosti nebo slovníkový odkaz spojený s příslušným řetězcem. [5] Datové vzorky (podřetězce) jsou definovány jako toky dat a shodují se se vstupy do slovníku [3].

Běh algoritmu LZW začíná s prázdným slovníkem a řetězcem S (slovo - word) obsahujícím první znak zdrojového souboru. Vždy po přečtení dalšího znaku Z zjistí, jestli se řetězec S+Z vyskytuje ve slovníku. Pokud ano, pouze prodlouží řetězec S o znak Z, jinak zapíše nový odkaz na řetězec do slovníku. Pokud řetězec S obsahuje jediný znak, bude do slovníku zanesen pouze jediný znak. [5] [9]

Pokud se slovník zaplní dříve, než je přečten celý soubor, je celý slovník smazán a začíná se plnit znovu. Někdy se pro zlepšení účinnosti místo smazání slovníku používá algoritmus LRU (last-recently-used) pro odstranění nepoužívaných řetězců ze slovníku. [5]

Dekompresní část algoritmu postupně čte kódy komprimovaného souboru, zapisuje příslušející řetězce na výstup a přidává nové řetězce do slovníku. Do slovníku je vždy přidán řetězec reprezentovaný předcházejícím kódem a první znak z řetězce s aktuálním kódem. Pro případ, že by byl přečten kód, kterému ještě nebyl přiřazen řetězec, je do slovníku přidán řetězec skládající se z předcházejícího řetězce a jeho posledního znaku. Takto vytvořený slovník bude totožný s tím, který vytvořil kompresní algoritmus. [5] [9]

### 3.5 Huffmanovo kódování

Huffmanovo kódování patří do skupiny algoritmů, které pracují na základě různých pravděpodobností znaků kódovaných dat. [5] Huffmanova konstrukce minimálního kódu je vždy optimální, ale není jednoznačná [9].

Při kompresi se postupuje tak, že nejprve komprimační algoritmus zjistí pravděpodobnosti výskytů jednotlivých znaků a každému znaku přiřadí jedinečný kód. Takovéto kódy se liší svou bitovou délkou. Tato část algoritmu je nejdůležitější a musí být navržena tak, aby přiřazení kódů znakům respektovalo požadavek na přiřazení bitově nejkratších kódů znakům s častějším výskytem a bitově delších kódů znakům s méně častým výskytem. Pak algoritmus postupně načítá znaky vstupního souboru, nachází odpovídající předem přiřazené kódy a tyto kódy zapisuje na výstup. [5]

Binární strom se podle zásad Huffmanova algoritmu tvoří od koncových „listů“ stromu, tj. od znaků s nejmenší pravděpodobností. Nejprve dojde k výběru znaků s nejmenší

pravděpodobností výskytu a z nich se vytvoří dva koncové listy stromu. Potom se vybere znak s nejnižší pravděpodobností z dosud nezpracovaných znaků, a tak se postupuje až ke kořeni stromu. [5]

Kompresce na základě vytvořeného binárního stromu probíhá tak, že kompresní algoritmus čte sekvenčně znaky ze vstupního souboru a do výstupního pak zapisuje jim příslušné sekvence bitů [5].

Dekomprese vyžaduje znalost původního binárního stromu. Pokud jsou kódy ve zkomprimované podobě souboru uloženy bezprostředně za sebou, bude již dekompresní algoritmus schopen rozpoznat znak příslušející ke kódu. [5]

Slabinou tohoto algoritmu je mimo jiné i to, že binární strom lze bez problémů sestavit pouze v případě, že pravděpodobnosti výskytu všech znaků vstupního souboru jsou mocninou čísla  $1/2$ . Pouze v tom případě každé vyšší patro vytvářeného stromu obsahuje uzel nebo list stromu s pravděpodobností výskytu dvojnásobnou oproti listům či uzlům ležícím o patro níž. V praxi však s něčím takovým nelze počítat, a proto je zapotřebí zaokrouhlovat pravděpodobnosti výskytu jednotlivých znaků. [5]

Zaokrouhlování ovšem musí být prováděno při dodržení dvou základních zásad: [5]

- součet všech procentuálních hodnot musí dávat hodnotu 100 %;
- je nutné dbát na to, že v každém patře vytvářeného stromu může být maximálně určitý počet listů a uzlů. Tento počet je dále omezen již vytvořenými patry stromu.

### 3.5.1 CCITT kódování

CCITT (International Telegraph and Telephone Consultative Committee) je standardizační organizace, která vyvinula sérii komunikačních protokolů pro přenos černobílých předloh přes telefonní linky a datové sítě. Tyto protokoly jsou obecně známy jako CCITT standardy T.4 a T.6, ale častěji se nazývají CCITT Group 3 a Group 4 komprese. Toto kódování je speciálním typem Huffmanova kódování. [3]

CCITT algoritmy jsou algoritmy neadaptivní - tzn. nepřizpůsobují se každé bitmapě tak, aby byla kódována s optimální účinností, ale používají pevnou tabulku kódových hodnot, které byly vybrány podle referenčního vzorku dokumentů obsahujících text i grafiku. [3]

Protože byl CCITT algoritmus optimalizován pro strojově a ručně psané dokumenty, je zřejmé, že předlohy, které se radikálně liší ve své kompozici nebudou velmi dobře komprimovány. U mnohých z nich dojde k záporné kompresi. [3]

CCITT definuje 3 algoritmy, které se používají pro kódování dvourozměrných dat: [3]

- Group 3 jednoúrovňové (G31D)
- Group 3 dvourozměrné (G32D)
- Group 4 dvourozměrné (G42D)

U G31D se provádí komprese RLE a hodnoty čítače opakování jsou dále kódovány Huffmanovým kódováním. Používají se zde krátké kódy pro typické úseky bílých resp. černých pixelů. [2]

Ve standardu G32D se provádí kódování pozic změny barvy, relativně vůči předchozí pozici a uvažuje se i změna oproti předchozímu řádku (ve vertikálním směru). Každý K-tý řádek se kvůli spolehlivosti kóduje pomocí G31D. [2] Tento standard obsahuje kontrolní kódy pro detekci a odstranění případných chyb. [3]

Kódování G42D je kromě několika modifikací stejné jako kódování G32D. Neobsahuje však žádné kontrolní kódy. [3]

### 3.6 Shannon-Fanovo kódování

Shannon-Fanovo kódování je velice podobné Huffmanovu. Rozdíl mezi oběma algoritmy spočívá v konstrukci binárního stromu. Tvorba binárního stromu v Shannon-Fanově modifikaci je poněkud jednodušší. Lze ji shrnout do dvou následujících kroků: [5]

- Rozdělení souboru symbolů na dvě skupiny se stejnou nebo co nejpodobnější celkovou pravděpodobností znaků obsažených v obou skupinách.
- Opakování prvního kroku na všechny dosud vytvořené skupiny, dokud každá skupina nebude obsahovat jediný znak.

Rozdíl mezi způsoby vytváření binárních stromů v Huffmanově a Shannon-Fanově variantě je v tom, že Huffmanovo kódování vytváří strom od koncových listů směrem ke kořenu, zatímco Shannon-Fanova metoda postupuje obráceně - od kořene k listům. [5]

Zatímco Huffmanova varianta při správném použití generuje vždy optimální kódy pro jednotlivé znaky, jednodušší Shannon-Fanova metoda může v některých případech použít několik bitů navíc. [5] [9]

### 3.7 Aritmetické kódování

Základní myšlenku použitou v této kompresní metodě lze popsat ve stručnosti takto: aritmetické kódování reprezentuje celou zprávu jako číslo z intervalu  $<0,1$ ). Na začátku kódování se uvažuje celý tento interval. Jak se zpráva prodlužuje, postupně se tento interval zužuje. Na konec stačí zapsat libovolné číslo z výsledného intervalu - to samo o sobě reprezentuje celou zprávu. [5]

Algoritmus komprese lze nastínit jako následující sekvenci kroků: [5]

- Zjištění pravděpodobností výskytu jednotlivých znaků ve zdrojovém souboru.
- Rozdělení intervalu  $<0,1$ ) na podintervaly, jejichž vzájemný poměr velikostí odpovídá poměru pravděpodobností jednotlivých znaků (seřazených podle abecedy).
- Uložení tohoto základního rozdělení intervalu  $<0,1$ ).
- Vlastní komprese víceznakové zprávy. Komprese víceznakové zprávy probíhá tak, že se nejprve vybere první znak vstupního souboru. Ten zúží interval  $<0,1$ ) na podinterval příslušející tomuto znaku tak, jak mu byl přidělen v druhém bodě celkového algoritmu aritmetického kódování. Tento podinterval je následně rozdělen stejným způsobem jako dříve celý interval  $<0,1$ ). Po načtení dalšího znaku je podinterval dále zúžen podle načteného znaku. Tak to půjde dále až do načtení posledního znaku zprávy. Čím je kódovaný znak pravděpodobnější, tím se interval zúží méně.
- Posledním bodem je vybrání kteréhokoli zlomku náležejícího do výsledného nejjemnějšího podintervalu a jeho převedení do binární formy [5]. Vznikne tak číslo, které přesně definuje daný interval [8].

Kromě kódové hodnoty je pro potřeby dekomprese zakódované zprávy nezbytné uložit pravděpodobnosti jednotlivých znaků a počet znaků původní zprávy. [5]

Při dekompresi je možné postupovat podle následujícího návodu: [5]

- výběr podintervalu  $\langle K(i), K(i+1) \rangle$  tak, aby kód náležel do tohoto intervalu
- zjištění a zápis znaku, který náleží svou pravděpodobností do tohoto intervalu.

### 3.8 JBIG

JBIG (Joint Bi-level Image Experts Group) je metoda pro bezztrátovou kompresi obrazů, které obsahují ideálně pouze černou a bílou a využívá se především u faxů [5]. Pomocí metody JBIG však lze komprimovat i předlohy ve stupních šedi o hloubce do 256 bitů na pixel [3]. Pro obrázky s odstíny šedi používající více než 8 bitů na pixel by již tato metoda nebyla tak účinná jako metody jiné [5]. Tyto předlohy s větším počtem bitů na pixel jsou komprimovány po bitových plochách, nikoli po pixelech - např. 8 bitový obrázek komprimovaný metodou JBIG bude kódován do osmi oddělených bitových ploch. [3]

JBIG byl zatížen celou řadou patentovaných postupů [3]. Dne 26.2.2011 vypršely ve všech zemích kromě USA veškeré patenty na algoritmy použité v JBIG. Celosvětově není komprese JBIG zatížena žádnými patenty od 4.4.2012. [10]

Před samotným kódováním je obraz rozdělen do bitových rovin, kde je v každé rovině obraz redukován na hloubku šedi jediného bitu na pixel. Pokud má již originální obrázek pouze 1 bit na pixel, fáze rozdělení na bitové roviny odpadá. [5]

Další rozdělení obrazu zajišťuje, že obraz bude k příjemci posílán po částech tzv. progresivní metodou. Obraz je rozdělen na několik horizontálních pruhů, které jsou kódovány postupně. Každý pruh se navíc vysílá ve vrstvách podle rozlišení. Přijímací zařízení bude obraz rekonstruovat tak, jak jej bude přijímat od hrubších (nejvyšších rozlišení) k jemnějším detailům (nejmenším rozlišení). Tato metoda používá postupné zdvojování úrovně rozlišení. Předloha se třemi vrstvami má tedy dvě zdvojení. Počet kódovatelných zdvojení není nijak omezen. [3] [5]

Používá se také sekvenční kódování, při kterém se naopak obraz kóduje shora dolů bez jakéhokoliv prokládání a jako jeden obrázek. Sekvenční předloha JBIG se dekóduje jedním průchodem a dosahuje přinejmenším stejně dobrého kompresního poměru jako Group 4, ovšem obecně horšího než při použití progresivní metody. [3] [5]

Pro kompresi obrazu se používá adaptivní aritmetický kodér (Q CODER), konkrétně se jedná o neztrátový adaptivní algoritmus podobný Huffmanovu kódování. V průběhu komprese se postupně vytváří adaptivní rozdělení pravděpodobností výskytu bílých a černých pixelů. Pro pravděpodobnější symboly se potom v kompresi použije menší počet bitů než pro symboly méně pravděpodobné. [3] [5]

U běžných dvouúrovňových kompresních algoritmů probíhá kódování po jednom řádku, a to metodou proudové délky. Algoritmy tohoto druhu bývají označovány jako kódovací metody 1D. U metod 2D dochází k zakódování pixelových proudů popisem rozdílu mezi hodnotami pixelu v aktuálním a předchozím řádku. [3]

Nadbytečná obrazová data jsou u metody JBIG kódována porovnáním pixelu ve vzorkované řádce s množinou pixelu, které už kodér vzorkoval. Těmto dodatečným pixelům se říká šablona, a vytvářejí jednoduchou mapu vzoru složeného z pixelu, které obklopují právě kódovaný pixel. Z hodnot těchto pixelů se určí nadbytečné vzory v obrazových datech. Tyto vzory jsou následně komprimovány adaptivním aritmetickým kompresním kóděrem. [3]

### 3.9 JBIG 2

Navrhovaným cílem pro JBIG2 bylo dosáhnout lepší bezztrátové komprese, než kterého dosahují jiné již existující standardy a zároveň umožnit při ztrátové kompresi mnohem vyšší kompresní poměr při téměř nezatelném snížení kvality [11]. Obrázky mohou být dekomprimovány s rychlostí dekódování až 250 milionů pixelů za sekundu. [12]

JBIG2 umožňuje využít Huffmanova kódování se standardními či předdefinovanými tabulkami. Kódová tabulka se skládá z řádku, kde každý řádek říká, jak zakódovat konkrétní hodnotu nebo jak zakódovat hodnotu z určitého rozsahu. [11]

V JBIG2 je možné využít adaptivní verzi aritmetického kódování a její varianty. Jednou z variant je adaptivní binární aritmetické kódování. Jelikož kóduje nuly a jedničky, dělí interval na dva podintervaly. [11]

JBIG2 model zachází s textovými daty a pultónovými obrazy jako se speciálními případy. Proto se předpokládá, že JBIG2 enkodér rozdělí obsah stránky na oblast textovou, oblast pultónovou a generickou. Některé oblasti mohou být prázdné a různé oblasti se mohou na fyzické stránce i překrývat. [11]

Kódování textové oblasti je založeno na zakódování bitmapy jednoho reprezentativního symbolu, namísto kódování každého výskytu jednotlivě. Bitmapové reprezentace symbolů se ukládají do jednoho či více slovníku. Slovník obsahuje indexovaný seznam bitmap, kde každá bitmapa reprezentuje právě jeden symbol. Při použití bezztrátové komprese se berou symboly i s nepatrným rozdílem v jejich bitmapové reprezentaci jako různé. Naproti tomu u ztrátové komprese se odchylky neberou v úvahu. [11]

Půltónová oblast se skládá z množství obrazců umístěných podél pravidelné mřížky. Obrazce obvykle odpovídají hodnotám odstínu šedi. Komprese může být realizována dvěma metodami. Jedna z metod je vysoce podobná kontextově závislému aritmetickému kódování, který přizpůsobuje určení pozice jednotlivých vzorů za účelem získání vztahu mezi přiléhajícími pixely. U druhé metody se půltónové zobrazení převede zpět do odstínu šedi. Hodnoty převedených odstínu šedi jsou pak použity jako indexy slovníku s půltónovými obrazci (malé bitmapové obrazce pevné velikosti). [11]

Pro generickou oblast se používá kontextově závislé aritmetické kódování. [11]

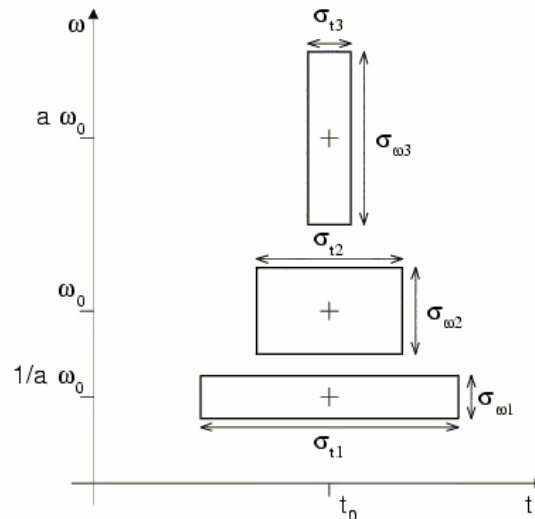
### 3.10 Vlnková transformace

Vznik vlnkové transformace (wavelet transform) je jedním z výsledků snahy získat časově-frekvenční popis signálu [13]. Vlnková transformace se využívá např. při kompresích obrazů nebo při zvýrazňování signálů v šumu. Existuje několik způsobů výpočtu vlnkové transformace s různou výpočetní náročností pro různé typy použitých vlnek. [14]

Historicky starší Fourierova transformace poskytuje informaci o tom, které frekvence se v signálu nacházejí, nevypovídá však o jejich umístění (poloze) v čase, je tedy vhodná jen pro popis stacionárních signálů. [13]

Možným řešením uvedeného problému je použití okna, které v čase ohraničí krátký úsek signálu a umožní z něj určovat spektrum v daném časovém intervalu. Z obdoby Heisenbergova principu neurčitosti vyplývá, že nelze současně určit přesně frekvenci a polohu jejího výskytu v čase. Proto má uvedené řešení pro časově konstantně široké okno pro všechny kmitočty velkou rozlišitelnost ve frekvenci a malou v čase a naopak pro časově úzké okno velkou rozlišitelnost v čase a malou ve frekvenci. Ideou vlnkové transformace je vhodnou změnou šířky okna v čase a jeho tvarem dosáhnout optimálního

poměru rozlišitelnosti v čase a frekvenci. Pro nízké frekvence je okno širší, pro vysoké užší - viz *Obr.5.* [13]

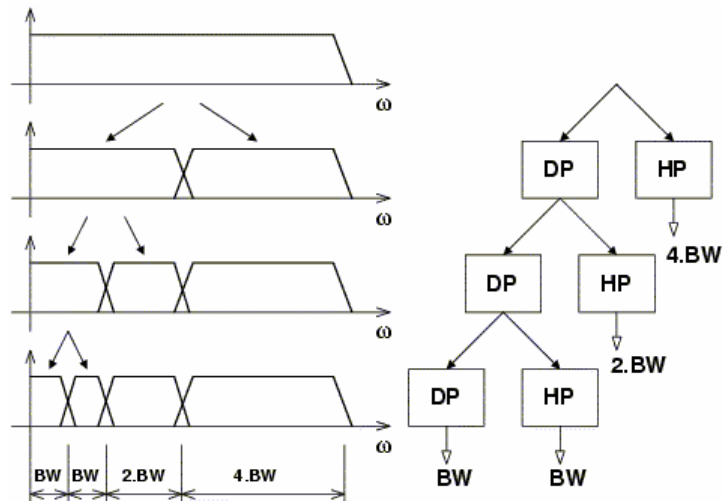


*Obr.5* Časově-kmitočtové rozlišení vlnkové transformace [13]

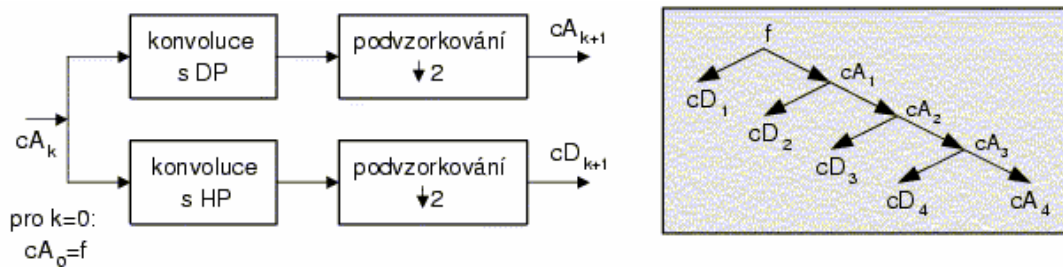
### 3.10.1 Diskrétní vlnková transformace

Základem diskrétní vlnkové transformace (Discrete Wavelet Transform - DWT) je ortonormalita. Díky ortonormalitě umožňuje zvolená vlnka neredundantní dekompozici signálu, tzv. analýzu s mnoha rozlišeními (multiresolution analysis, decomposition). Vlnková funkce se chová jako pásmová propust filtrující vstupní signál kolem centrálního kmitočtu, který je závislý na měřítku mocninou dvou. V následujícím měřítku je filtrována horní polovina pásma předchozí dolnofrekvenční části signálu - viz *Obr.6.* S rostoucím kmitočtem roste šířka pásma tohoto filtru, činitel jakosti  $Q$  je tak konstantní pro celou množinu měřítkem odvozených filtrů. Pro zvolené minimální měřítko však zůstává nepokryto pásmo od nižších kmitočtů do nuly. Proto je od vlnky  $\psi$  odvozena měřítková funkce  $\phi$  (scaling function), která má charakter dolní propusti. [13]

DWT lze počítat rychlým algoritmem, tvořeným filtrací FIR filtry a podvzorkováním. Jeden krok DWT je vidět na *Obr.7-vlevo* a rozklad na aproximace a detaily na *Obr.7-vpravo.* [13]



Obr.6 Frekvenční pohled na diskrétní vlnkovou transformaci [13]



Obr.7 Jeden krok DWT a rozklad na aproximace a detaily [13]

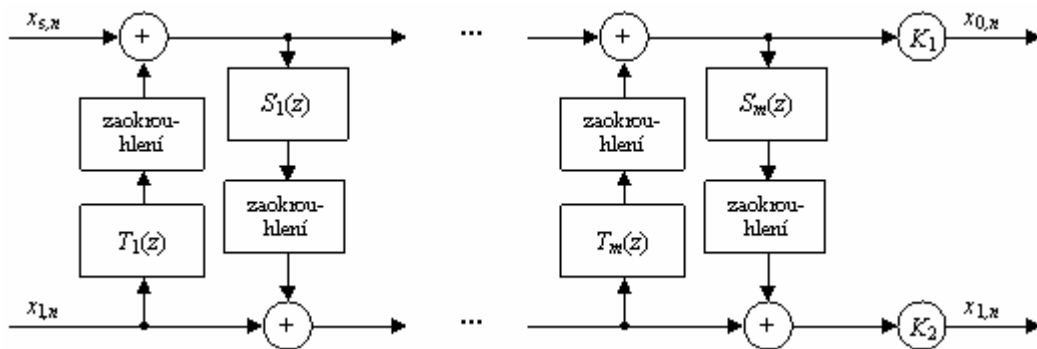
### 3.10.2 Celočíselná vlnková transformace

Celočíselná vlnková transformace (Integer Wavelet Transform - IWT) je reverzibilní, tzn. obraz může být z celočíselných koeficientů plně rekonstruován. Lze ji tedy použít pro neztrátovou kompresi. [8]

Celočíselná vlnková transformace může být použita k rozložení obrazu kterýmkoliv způsobem jako vlnková transformace s reálnými čísly - např. pyramidovým rozkladem, rozkladem na linie atd. [8]

Je několik způsobů, jak modifikovat DWT tak, že produkuje celočíselné koeficienty - jedním z nich je např. modifikovaný algoritmus lifting [8]. Při výpočtu vlnkové transformace pomocí algoritmu lifting je možné tento algoritmus upravit tak, aby jeho výstupní hodnoty byly celočíselné. Pokud se upraví i rekonstrukční část vlnkové transformace je výsledná transformace reverzibilní. Vlastní úprava spočívá v zavedení zaokrouhlování do jednotlivých stupňů liftingu. [14] [15]

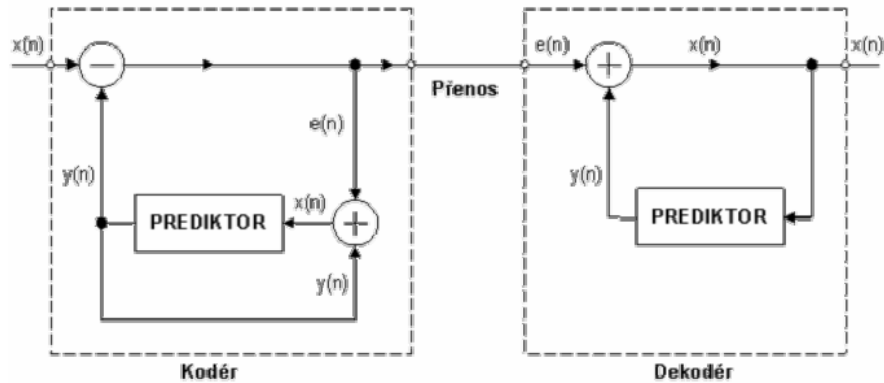
Blokové schéma celočíselného vlnkového rozkladu je na *Obr.8*. Přenosové funkce  $S(z)$  a  $T(z)$  mohou být libovolné a nezáleží ani na počtu kroků liftingu. V blokovém schématu se vyskytuje násobení konstantami (blok  $K_1$  a  $K_2$ ), které mohou způsobit vznik čísel s desetinou čárkou, pokud jejich hodnota není celé číslo. [14] [15]



*Obr.8* Blokové schéma několikastupňového celočíselného liftingu [15]

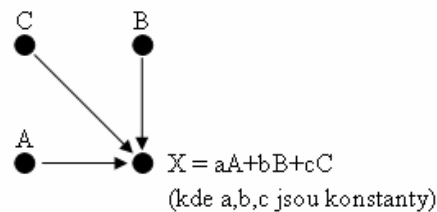
### 3.11 Prediktivní metody

Hodnoty sousedních bodů digitálního obrazu jsou si velmi často podobné (existuje mezi nimi tzv. prostorová korelace). Díky této závislosti lze odhadnout (predikovat) hodnotu následujícího bodu v obraze z hodnot sousedních bodů. Při úspěšné predikci se odhadovaná hodnota  $y(n)$  jen málo liší od skutečné  $x(n)$ . Tento rozdíl je takzvaná chyba predikce  $e(n)$ . Tato chyba dosahuje menších absolutních hodnot než skutečná hodnota pixelu a k jejímu zakódování tedy postačuje kratší kódové slovo. Blokové schéma predikčního kodéru využívajícího principů rozdílové pulsně kódové modulace (DPCM - Differential Pulse Code Modulation) je na *Obr.9*. [16]



Obr.9 Blokové schéma predikčního kodéru a dekodéru [16]

Predikce současné hodnoty může být určena např. váhovanou lineární kombinací předchozích vzorků nebo pomocí polynomiálních funkcí či dvojrozměrných prediktorů. Princip dvojrozměrného prediktoru je zobrazen na Obr.10. [16]



Obr.10 Princip dvourozměrné predikce [16]

### 3.11.1 PPM

PPM (Prediction by Partial Matching) je adaptivní statistická technika datové komprese založená na kontextovém modelování a predikci. PPM modely používají množinu předchozích symbolů pro predikci následujícího symbolu. [17]

Cestou, jak při kódování přesněji určit frekvenci výskytu znaku, je zjištění kontextu, ve kterém se znak vyskytuje - proto se metoda PPM také nazývá metoda konečného kontextu. Např. při kompresi běžného textu v českém jazyce se písmeno ‚e‘ vyskytuje v mnohem větší míře po písmenu ‚s‘ než třeba po písmenu ‚w‘. Kontextová metoda pro kódování výskytu ‚e‘ po ‚s‘ použije menší počet bitů než pro kódování výskytu ‚e‘ po ‚w‘. Vzhledem

k tomu, že tyto metody používají adaptivní model, kde kontext musí mít k dispozici i dekompresní algoritmus, připadá v úvahu jen levý kontext kódovaného znaku. [6]

Problémem kontextového kódování je celkový počet frekvencí, který velmi narůstá s délkou kontextu. U bezkontextového kódování máme 256 různých znaků a tím i 256 frekvencí. Pokud bychom vzali jako kontext jeden znak, máme již  $256^2$  možných frekvencí, neboť každý z 256 znaků může mít v textu před sebou kterýkoliv z 256 znaků. Protože zvětšení kontextu o 1 způsobí zvýšení možných frekvencí o mocninu 256, tak kontextové metody používají jen omezenou délku kontextu - typicky nebývá délka kontextu větší než 4. [6]

Dalším problémem kódování v kontextu je situace, kdy se znak v daném kontextu vyskytuje poprvé. Frekvence jeho dosavadního výskytu a tedy i pravděpodobnost je v tomto případě nula. Nulovou pravděpodobnost ale statisticky kódovat nelze. Proto je nutné pro tyto případy vyčlenit nějakou nenulovou hodnotu pravděpodobnosti. Protože není k dispozici způsob, jak výpočet této pravděpodobnosti matematicky odvodit, bylo intuitivně navrženo několik možných přístupů řešení uvedeného problému. Experimentálně bylo ověřeno, že nejlepší výsledky dává metoda označená jako C. [6]

## 4 FORMÁTY VYUŽÍVAJÍCÍ NEZTRÁTOVOU KOMPRESI

Existence mnoha formátů má několik příčin: [1]

- Historické důvody - formáty odrážejí technický vývoj, zejména postupně se zvyšující barevné možnosti zařízení jak pro snímání obrazu, tak pro jeho zobrazení.
- Vazba na program - podle druhu aplikace vznikaly specializované formáty, například pro úschovu skic a kreseb, černobílých dokumentů či pro přenos barevných fotografií a jejich prezentaci na WWW.
- Technické důvody - mnoho formátů bere ohled na rozlišení skenerů, pomocí kterých jsou obrazy zaznamenávány, na obrazová rozlišení v různých osách, na odlišné architektury obrazové paměti v grafických kartách apod.
- Metoda komprese - vzhledem k velkému paměťovému objemu barevných obrazů je žádoucí uchovávat obraz v komprimované podobě. Volba vhodné kompresní metody je často závislá na charakteru obrazu a na jeho dalším použití.

Uvedený přehled nezaznamenává zcela všechny důvody existence široké škály obrazových formátů. Existuje celá řada dalších aplikačních požadavků, jakými je například uložení více obrázků v jednom souboru, zápis sekvence obrazů pro animaci apod. [1]

### 4.1 BMP

BMP je velmi rozšířený formát, který nejčastěji obsahuje nekomprimovaná data. Je zde však možné použít komprimaci RLE. Komprimaci lze využít pouze u bitmap se čtyřmi a osmi bity na pixel. U 1 a 24 bitových bitmap jsou obrázky uloženy v nekomprimované podobě. Formát pracuje s uspořádáním Bytů malý endian. [18]

Každý korektní soubor typu BMP obsahuje sekce popsané v *Tab.1*, ve které jsou sekce uvedeny podle pořadí výskytu v souboru. Barevnou paletu neobsahují obrázky bitové hloubky 24 bitů. [18]

Hlavička souboru formátu BMP (BITMAPFILEHEADER) má délku 14 Bytů. Hlavička s informacemi o obrázku (BITMAPINFOHEADER) obsahuje základní metainformace o rastrovém obraze. Hlavička verze 3 je nejčastěji používána, i když je možné se setkat i s novějšími hlavičkami verze 4 a 5 [19].

Tab.1 Struktura formátu BMP [18]

Název	Význam
BITMAPFILEHEADER	hlavička souboru BMP
BITMAPINFOHEADER	informační hlavička o obrázku
RGBQUAD[]	tabulka barev (barevná paleta)
BITS	pole bitů obsahujících vlastní rastrová data (pixely)

V barevné paletě udává první trojice Bytů hodnotu tří barevných složek posloupnosti BGR a čtvrtý Byte je nastavený na nulovou hodnotu. Čtvrtý Byte není možné (podle platné specifikace) použít pro jiné účely - tedy ani pro alfa kanál. Podpora alfa kanálu je zavedena až od Windows XP, ale mnoho programů ji neimplementuje. [18]

Ukládání obrazových řádků do souborů se provádí zleva doprava a zespodu nahoru. Všechny obrazové řádky musí mít Bytovou velikost dělitelnou čtyřmi. Mohou zde být uloženy indexy, pixely ve formátu BGR a nebo komprimovaná data. [18]

U RLE jsou obrazová data organizována do skupiny po dvou Bytech. První Byte určuje počet po sobě jdoucích pixelů, které budou mít index barvy určený v druhém Bytu. První Byte ve skupině může být nastavený na nulu, což znamená tzv. únik: např. konec řádku, konec bitmapy nebo delta. Typ úniku je zapsán ve druhém Bytu skupiny. [19]

## 4.2 GIF

V grafickém formátu GIF se celý obrázek, který je zde nazýván logická obrazovka, skládá z několika tzv. rámců, a není tedy v souboru uložen jako jeden celek. Rámce jsou obdélníkové oblasti umístěné uvnitř logické obrazovky. Není nutné, aby rámce pokryly celou logickou obrazovku a mohou se navzájem překrývat. [20]

Minimálně musí být vždy přítomen jeden rámeček, jejich maximální množství však není omezeno. Každý rámeček je možné chápat jako rastrový obrázek, který je celou svou plochou umístěn v logické obrazovce - podle specifikace nesmí žádný pixel z rámce padnout mimo logickou obrazovku. Pozice rámce v logické obrazovce je určena souřadnicí jeho horního levého rohu a velikostí (šířkou, výškou) zadanou v pixelech. [20]

V grafickém formátu GIF rozlišujeme dvě barevné palety: globální a lokální. Globální barevná paleta může v souboru formátu GIF existovat maximálně jednou a její velikost je

zadána v hlavičce popisující logickou obrazovku. Lokální barevná paleta může být přiřazena ke každému rámcí, opět se však nejedná o povinnou součást rámce. [20]

I když se v literaturách uvádí (např. v [3]), že v obrázku formátu GIF je možné zobrazit pouze 256 různých barev, není to podle [20] pravda. Pravdivé je pouze tvrzení, že v jednom rámcí může být zobrazeno maximálně 256 barev. Pro zvýšení celkového počtu barev lze využít možnosti lokálních barevných palet. [20]

Celý soubor je rozdělen do bloků, z nichž některé jsou povinné, některé se musí vyskytovat na určitém místě souboru, další se mohou opakovat apod. Povolené bloky, které se mohou ve formátu GIF vyskytovat, jsou popsány v *Tab.2* spolu s informací, zda se jedná o bloky povinné a zda je možné bloky v rámci jednoho souboru opakovat. Dále je u každého bloku uvedeno, zda se jeho definice objevila už ve specifikaci GIF87a nebo až v novější specifikaci GIF89a. [21]

*Tab.2 Bloky formátu GIF [21]*

Název bloku	Povinná položka	Lze opakovat	Verze
signatura	ano	ne	87a
verze souboru	ano	ne	87a
popis logické obrazovky	ano	ne	87a
globální barevná paleta	ne	ne	87a
rozšiřující grafický blok	ne	ano	89a
popis rámce	ano	ano	87a
lokální barevná paleta	ne	ano	87a
data rámce (pixely)	ano (pro rámeček)	ano	87a
rozšiřující informace (textové, binární)	ne	ano	89a
ukončovací znak GIF souboru	ano	ne	87a

### 4.3 PCX

Jedná se o formát využívající jednoduchou bezeztrátovou kompresi založenou na algoritmu RLE. U PCX je jasně patrná nekonceptnost návrhu a závislost prvních verzí tohoto formátu na použitých grafických kartách, resp. jejich videorežimech. [23]

Grafický soubor typu PCX obsahuje hlavičku, která má vždy délku 128 Bytů. Využito je však pouze 70 Bytů, zbývajících 58 Bytů může využít aplikace, ovšem s tím, že se jejich obsah může kdykoliv, například po konverzi a/nebo editaci přepsat. [23]

Celkový počet bitů na pixel se vypočte tak, že se vynásobí počet bitů na pixel v jedné obrazové rovině (offset 3) a počet obrazových rovin (offset 65). [23]

U osmi bitových obrázků je problém s barevnou paletou, protože v hlavičce souboru je rezervováno místo pouze pro 16 barev palety. Musel se tedy najít jiný způsob uložení. Ten spočívá v tom, že je barevná paleta uložena na konci souboru. Test, zda je zde grafická paleta skutečně uložena, spočívá v načtení 769 Bytů od konce souboru. Pokud tento Byte obsahuje hodnotu C0h, je barevná paleta přítomna a zbylých 768 Bytů obsahuje 256 trojic hodnot RGB. [23]

True Color (24 bitové) obrázky jsou ukládány do tří obrazových rovin. V každé obrazové rovině je rezervováno osm bitů na pixel. Obrazové roviny jsou ukládány prokládaně. [23]

RLE použitý u PCX pracuje s proudem Bytů, přičemž maximální délka tohoto proudu odpovídá délce obrazového řádku (tato hodnota je uložena v hlavičce). Obrazový řádek se postupně načítá a zjišťuje se, kolik Bytů (nikoli pixelů!) má stejnou hodnotu. Blok za sebou jdoucích Bytů se stejnou hodnotou se zapíše jako dvojice Bytů: první Byte udává počet znaků v bloku, druhý Byte hodnotu těchto Bytů. Počítadlo Bytů je inicializováno na hodnotu C0h (nejvyšší dva bity jsou 1), to znamená, že pokud dekomprimační program narazí na Byte větší než C0h, tak ví, že se jedná o komprimovaný blok se dvěma Byty. [24]

Jednotlivé Byty, které nejsou součástí bloku a mají hodnotu menší než C0h, jsou do komprimovaného souboru zapsány ve své původní podobě. Horší je to s Byty, které mají hodnotu větší nebo rovno C0h. Aby nenastala kolize s počítadlem, musí se tyto Byty uložit jako dvojice Bytů C1h C0h (popř. jiná hodnota v intervalu <C0h ; FFh. V tomto případě tedy nenastává komprese, ale naopak prodloužení (expanze) výstupního souboru. [24]

#### 4.4 PNG

Jako kompresní algoritmus je v tomto formátu využita komprese LZ77 a huffmanovo kódování. Tento způsob komprese se také nazývá Deflate. [3]

Grafický formát PNG obsahuje jednu velmi prospěšnou funkci - na každý obrazový řádek je možné ještě před jeho kompresí aplikovat jednoduchý konvoluční filtr, jehož úkolem je připravit data tak, aby se komprimovala lépe. Většinou se počítá s tím, že se po aplikaci filtru na data objeví větší množství pixelů s nulovou hodnotou, nebo hodnotou blízkou

nule, čímž se statisticky zvýší pravděpodobnost toho, že komprimační program v datech nalezne delší shodné posloupnosti a tím zmenší výslednou délku souboru. [25]

Filtrů, které je na jednotlivé obrazové řádky možné aplikovat, existuje v PNG celkem 5. Každý filtr musí provádět operaci, ke které lze nalézt operaci inverzní. Filtry jsou aplikovány na Byty, nikoli pixely. Tyto Byty reprezentují buď odstín šedi či index do barevné palety nebo hodnotu uloženou v jednom barevném kanálu. [26] [27]

Hlavička PNG má délku pouhých osmi Bytů, které mají vždy konstantní hodnoty, a to 89h 50h 4Eh 47h 0Dh 0Ah 1Ah 0Ah. V osmi Bytech se tvůrcům PNG podařilo vytvořit sekvenci Bytů, na kterých je možné otestovat, zda přenos proběhl v pořádku. [28]

Veškeré informace i metainformace o zpracovávaném obrázku jsou uloženy v blocích dat nazvaných chunky (přibližný překlad je blok). Každý chunk se skládá ze čtyř částí. Všechny vícebytové položky v chunkcích (včetně jejich délky) jsou uloženy v pořadí big endian, tj. Byte s nejvyšší vahou je v souboru na prvním místě. [28]

Každý obrázek typu PNG obsahuje tři povinné chunky v následujícím pořadí: [28]

- IHDR - hlavička obrázku
- IDAT - datová část (pixmap)
- IEND - ukončující značka

U obrázků obsahujících barevnou paletu přibývá ještě jeden typ chunku s názvem PLTE a je umístěn mezi chunk IHDR a IDAT. [28] Barevná paleta se nachází ještě před datovým chunkem IDAT. [29]

V chunku typu IHDR je uložena hlavička obrázku. Ta je odlišná od hlavičky celého souboru uvedené v předchozím textu. Hlavička obrázku obsahuje základní metainformace o uloženém obrázku, zejména jeho rozlišení, bitovou hloubku, typ kódování barev a použitou filtraci. Vzhledem k tomu, že datová část hlavičky obrázku má vždy délku 13 Bytů, začíná chunk posloupností 00h 00h 00h 0Dh. Hlavička obrázku musí být uvedena ihned za hlavičkou PNG. [28]

Chunk IDAT obsahuje 4 Byty s délkou chunku a dále následují hodnoty 49h 44h 41h 54h, které dávají řetězec „IDAT“. Poté následují zkomprimovaná data. Jeden komprimovaný datastream je u PNG uložen v „zlib“ formátu. [26]

Zkomprimované data uvnitř datastreamu jsou uložena jako série bloků, z nich každý může představovat nekomprimovaná data, data komprimovaná LZ77 s fixním Huffmanovým kódováním, nebo data komprimovaná LZ77 s Huffmanovým kódováním přizpůsobeným datům. Kompletní filtrovaný obrázek formátu PNG je reprezentován jedním zlib datastreamem, který je uložen v několika IDAT chuncích. [26]

## 4.5 TGA

TGA je možné komprimovat metodou RLE [3]. Podle [30] může být kódování RLE kombinované s Huffmanovým kódováním. Specifikace formátu TGA [31] ovšem uvádí pouze kompresi pomocí RLE, proto také není divu, že [30] dodává: „Mnoho převodních či prohlížečích programů podporují pouze variantu RLE bez Huffmanova kódování“.

Všechny informace jsou v souborech typu TGA rozděleny do čtyř sekcí, přičemž pouze první sekce je povinná. Sekce nemusí obsahovat identifikační hlavičku - pozice sekcí v souboru je možné zjistit již po načtení informační hlavičky souboru. [30]

V první sekci umístěné na začátku souboru je uložena informační hlavička, jejíž velikost je vždy rovna 18 Bytům. V této hlavičce jsou specifikovány všechny důležité informace o rastrovém obraze a způsobu jeho uložení v souboru. [30]

Podporovaný počet bitů na pixel tohoto formátu je 8, 16, 24, 32 [3]. Podle [30] je podporována také 1 bitová hloubka. Ovšem podle specifikace formátu TGA [31] toto není pravda. Na webové stránce [32] je zmíněno, že se jedná o ideové rozšíření původních typů obrázků TGA, které ovšem nebylo touto firmou oficiálně schváleno, a proto je vhodnější pro uložení obrázku s touto barevnou hloubkou použít jiný formát.

Za informační hlavičkou může následovat identifikační pole obrázku, což je textový řetězec o maximální délce 255 znaků. Do tohoto pole lze ukládat libovolné údaje. Tato sekce je nepovinná a v obrazových souborech se moc často nevyskytuje. [30]

Ve třetí sekci může být uložena barevná paleta. Tato sekce je nepovinná a používá se pouze u některých obrázků s formátem 8 bitů na pixel. Paleta obsahuje hodnoty barevných složek ve formátu RGB pro každou položku uloženou v paletě. [30]

V sekci čtvrté jsou uložena vlastní rastrová data, tj. barvy jednotlivých pixelů. Posloupnost rastrových dat (zejména orientaci vertikální osy) lze ve formátu TGA specifikovat přímo v hlavičce, je například možné obrázky ukládat od prvního řádku do řádku posledního či

naopak. Jak již bylo zmíněno, rastrová data mohou být komprimována jednoduchým RLE algoritmem. [30]

Při použití RLE kódování jsou posloupnosti po sobě následujících pixelů s barvami se stejnou hodnotou zakódovány dvojicí hodnot: počtem opakování a hodnotou opakování. V režimu kódování RLE je nejvyšší bit (tj. hodnota 80h) každého Bytu na začátku datového paketu rezervován jako logický příznak, zda se bude jednat o opakování (RLE paket) či nikoliv. Pokud je tento příznak nastaven, tak udávají hodnoty nižších sedmi bitů počet opakování barvy, která je uložena v následujících Bytech - 1 Byte pro osmibitové obrázky, 2 Byte pro šestnáctibitové obrázky atd. [30]

V případě, že je příznak pro opakování 0 (tj. v prvním Byte posloupnosti je uložena hodnota menší než 80h), udávají hodnoty nižších sedmi bitů nekomprimovaná data [30].

## 4.6 TIFF

Formát TIFF (Tag Image File Format) je asi nejvšestrannější a nejrozmanitější existující bitmapový formát. Jeho schopnost rozšíření a podpora mnoha datových kompresních schémat dovoluje vývojářům přizpůsobit si formát TIFF svým potřebám. [3]

Jeden z častých problémů je spojen s kompresí obrazových dat. Některé programy nemají implementovány všechny kompresní algoritmy a v dalších mohou být vytvořeny nestandardní kompresní algoritmy, které se původně ve specifikaci nevyskytují. [3] [33]

Soubory TIFF jsou organizovány do tří sekcí: [3]

- Hlavička souboru předlohy (Image File Header - IFH)
- Adresář souboru předlohy (Image File Directory - IFD)
- Bitmapová data.

Hlavička souboru předlohy o velikosti 8 Bytů se skládá ze tří informačních polí. [3]

Identifikátor obsahuje buď hodnotu 4949h („II“) nebo 4D4Dh („MM“). Tyto hodnoty vyjadřují, zda jsou data v souboru TIFF zapsána v uspořádání malý endian (4949h [34]) nebo v uspořádání velký endian (4D4Dh [34]). Všechna data za těmito dvěma Byty jsou v určeném Bytovém uspořádání. [3]

Jestli je daný soubor skutečně ve formátu TIFF, lze zjistit podle prvních čtyř Bytů. Pokud to jsou 49h 49h 2Ah 00h nebo 4Dh 4Dh 00h 2Ah jde o soubor TIFF. [3]

IFD je blok informací podobný hlavičce u ostatních formátů. Jeho úkolem je popisovat data, se kterými souvisí. Obsahuje informace o výšce, šířce a hloubce předlohy, počet barevných ploch a typ datové komprese použité v bitmapových datech. Na rozdíl od většiny pevně daných hlaviček je IFD dynamická struktura, kde se nemusí měnit pouze její velikost, ale i její pozice v souboru. [3]

IFD může mít různou velikost, protože může obsahovat různý počet datových záznamů - tagů. Každý tag obsahuje jedinečný blok informací. Může nastat situace, kdy dvě IFD obsahují stejné tagy, ale tagy nejsou ve stejném pořadí. [3]

Hodnoty, které mohou být uloženy v tagu Compression (komprese) jsou vidět v *Tab.3* [34]. Další hodnoty tagu, které se používají v knihovně libtiff znázorňuje *Tab.4*. K této tabulce je nutné dodat, že Deflate komprese je považována za experimentální a není podporována všemi aplikacemi, proto je pro přenos obrázků vhodné použít jinou kompresi. [33]

*Tab.3 Hodnoty tagu Compression dle specifikace [34]*

Hodnota	Typ komprese
1	bez komprese
2	CCITT G31D bez instrukcí EOL a RTC
3	CCITT Group 3 (v tagu T4Options se dá nastavit typ 1D nebo 2D)
4	CCITT Group 4
5	LZW
6	TIFF JPEG 6.0 (zastaralá verze [33])
32773	PackBits

*Tab.4 Doplnující hodnoty tagu Compression dle dokumentace knihovny libtiff [33]*

Hodnota	Typ komprese
7	JPEG (nová verze)
32766	2-bitové kódovací schéma používané NeXTem
32771	CCITT
32809	4-bitové RLE schéma z ThunderScanu
32909	kompresní schéma Pixaru pro barevné obrázky s vysokým rozlišením
34676 a 34677	kompresní schéma SGI pro barevné obrázky s vysokým rozlišením
32946	Deflate (ZIP)

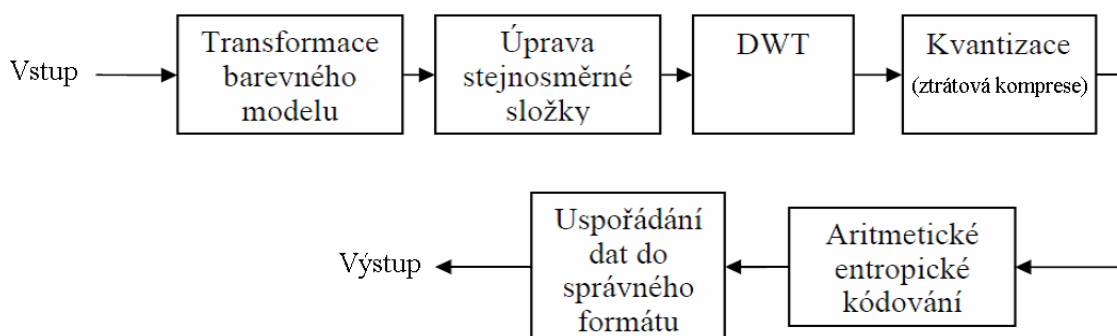
## 4.7 JPEG 2000

Formát JPEG 2000 nabízí široké možnosti pro progresivní přenos s ohledem na různá kritéria - z dat, která postupně přicházejí po přenosové lince, je možné rekonstruovat obraz v nízké kvalitě a s přibývajícimi daty kvalitu zvyšovat. [35]

Kromě ztrátové komprese nabízí JPEG 2000 rovněž kompresi bezztrátovou. Pro kompresi se využívá Diskrétní vlnková transformace (DWT). [35]

Obraz je před DWT rozdělen do stejně velkých obdélníkových oblastí (tzv. tile - dlaždice). Velikost těchto oblastí může uživatel nastavit až na velikost celého obrazu. Rozdělení na části snižuje paměťové nároky, ale snižuje také kvalitu obrazu. [35]

Řetězec operací, které jsou prováděny během komprese JPEG 2000 je naznačen na *Obr.11*. Při dekompresi jsou použity inverzní transformace v přesně opačném pořadí. [35]



*Obr.11 Postup komprese formátu JPEG 2000 [35]*

Prvním krokem je transformace barevného modelu. Nejčastěji je obraz uložen v modelu RGB. JPEG 2000 může pracovat i s jinými modely, které mají např. více než tři složky - pak se tento krok neprovádí. Pro kompresi se model transformuje na YUV, který má míru korelace nižší než RGB. Složky YUV se vypočtou jako lineární kombinace složek RGB. Pokud se provádí bezztrátová komprese, použije se celočíselná aproximace těchto vztahů, kde není třeba zaokrouhlovat a nevzniká tak chyba. [35]

Úprava stejnosměrné složky je velmi jednoduchá operace, která v podstatě znamená přechod od neznaménkového vyjádření složek na znaménkové vyjádření. [35]

Následuje diskretní vlnková transformace - pro ztrátovou transformaci se používá Cohen-Daubechies-Feauveau 9/7, pro bezztrátovou kompresi se používá celočíselná varianta Cohen-Daubechies-Feauveau 5/3. V JPEG 2000 se provádí dvojrozměrná DWT (2D DWT). Nejprve se aplikuje jednorozměrná DWT na všechny řádky a pak na všechny sloupce výsledku předešlého kroku. Vzniknou tak čtyři pásma - LL (nízké kmitočty horizontálně i vertikálně), LH (nízké kmitočty horizontálně, vysoké vertikálně), HL (vysoké kmitočty horizontálně, nízké vertikálně), HH (vysoké kmitočty horizontálně, vysoké vertikálně). Na pásmo LL se může použít 2D DWT opakovaně a dostat tak další úroveň dekompozice. Úroveň dekompozice také ovlivňuje kvalitu komprese a je nastavitelná uživatelem. [35]

Při ztrátové kompresi, se provede tzv. kvantizace. Provádí se vydělením koeficientů DWT kvantizačním krokem a zaokrouhlením. [35]

Následuje aritmetické entropické kódování. Metoda, která je použita v JPEG 2000, se nazývá EBCOT (Embedded Block Coding with Optimal Truncation). Do tohoto bloku vstupují bloky koeficientů DWT (např. 64×64 koeficientů pro jeden blok) a výstupem je komprimovaný bitový stream. [35]

Poslední blok je zodpovědný za to, aby data byla uspořádána ve správném pořadí a byly k nim přidány správné hlavičky [35].

## 4.8 HD Photo

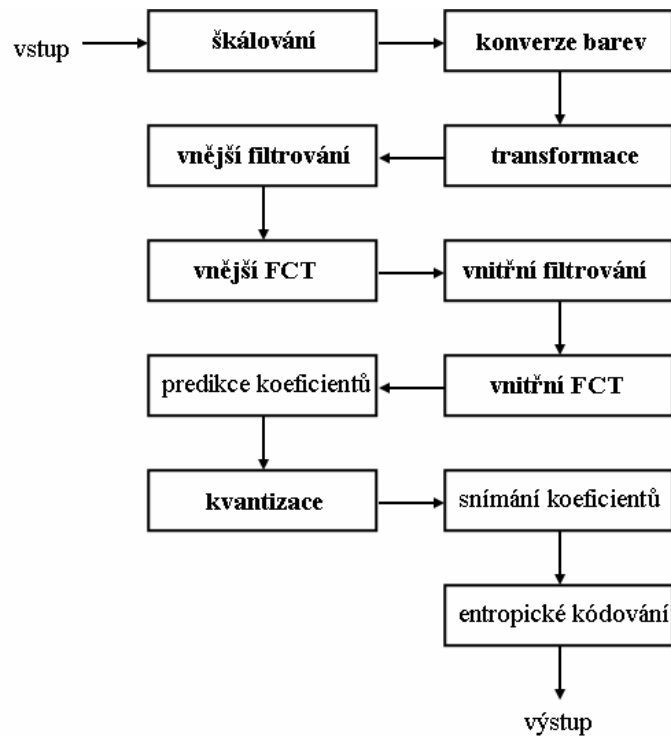
Tento formát je známý také pod názvem Windows Media Photo. HD Photo bylo navrženo k dosažení co nejvyšší obrazové kvality a optimálního výkonu. Soubor může obsahovat více obrázků a metadata. Maximální velikost souboru je  $2^{32}-1$  Bytů. [37]

HD Photo nabízí kódování s vysokým obrazovým rozsahem (HDR) a podporuje širokou škálu barevné reprezentace pixelů - monochromatickou, RGB, CMYK nebo n-kanálovou.

Struktura formátu HD Photo je téměř totožná se strukturou formátu TIFF. Hlavička souboru má délku 8 Bytů, po ní následují adresáře souboru předlohy (Image File Directory - IFD) a poté data. [37]

První 4 Byty mají vždy hodnotu 49h 49h, což značí, že jsou hodnoty vždy ukládány v uspořádání malý endian. Dále je uložen identifikátor formátu HD Photo - BCh. Nakonec obsahuje hlavička verzi formátu - buď hodnota 00h (0. verze) nebo 01h (1. verze). Poslední

4 Byty udávají offset prvního IFD. IFD je stejně jako ve formátu TIFF složen z tagů. V HD Photo jsou přidány nové typy tagů, ale je zde také podpora tagů formátu TIFF. [37]



*Obr.12 Postup komprese formátu HD Photo*

Na *Obr.12* je znázorněn postup komprese formátu HD Photo. Škálování je prováděno na vstupní data, která jsou delší než 24 bitů. Transformace znamená rozdělení obrazu na bloky a makrobloky. FCT znamená hlavní dopřednou transformaci (Forward Core Transformation) a je jádrem celého kompresního schématu. Kvantizace se provádí jen u ztrátové komprese. [38]

HD Photo používá reverzibilní barevnou konverzi, reverzibilní biorthogonální transformaci (druh DWT) a pokročilé kódovací schéma nearitmetické entropie. Kompresní algoritmus je navrhnut pro rychlou kompresi a dekompresi. Obrázek je rozdělen na  $16 \times 16$  makro bloky, což poskytuje minimální paměťové požadavky. [37]

## 4.9 JPEG-LS

Standard JPEG - Lossless (JPEG-LS) je také znám pod názvem LOCO-I (LOW COMPLEXITY LOSSLESS COMPRESSION FOR IMAGES). Formát využívá neztrátovou a mírně ztrátovou (near lossless) kompresi. [39] Příklad struktury formátu JPEG-LS je vidět v Tab.5. [40]

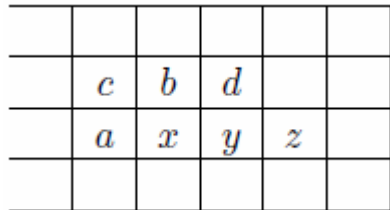
Tab.5 Základní struktura formátu JPEG-LS [40]

Offset	Velikost [Byte]	Význam
0	2	počátek obrázku - vždy FFh D8h
2	2	počátek snímku - vždy FFh F7h
4	2	délka segmentu
6	1	počet bitů na vzorek
7	2	počet řádků
9	2	počet sloupců
11	1	počet složek ve snímku
12	1	ID složky
13	1	informace o podvzorkování složky
14	1	vždy 0
15	2	počátek skenovacího segmentu - vždy FFh DAh
17	2	délka segmentu
19	1	počet složek v tomto skenovacím segmentu
20	1	ID složky ve skenovacím segmentu
21	1	index mapovací tabulky (0-bez mapovací tabulky)
22	1	tolerance při predikci (0-pro neztrátovou kompresi)
23	1	prokládaný mód (0-bez prokládání)
24	1	transformace bodu (0-bez transformace)
25	N	zkomprimovaná data obrázku
25+N	2	konec obrázku - vždy FFh D9h

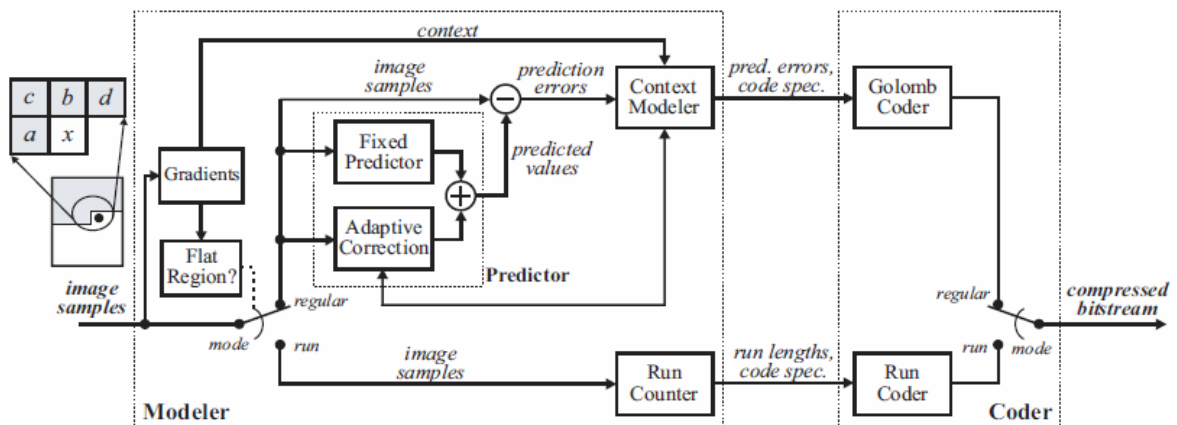
Při kompresi formátu JPEG-LS se prohledává několik z předchozích sousedů aktuálního pixelu a zjišťuje se spojitost (resp. rozdíly) mezi pixely. Tyto spojitosti se používají k predikci pixelu a následně k výběru pravděpodobnostní distribuce. [8] Fixní prediktor zjišťuje testem horizontální a vertikální hrany. [39] Následně se aplikuje adaptivní korekce chyby, způsobené tímto fixním prediktorem [40].

Pravděpodobnostní distribuce se využívá ke kódování chyby predikce pomocí Golombova kódování. Je zde také možnost využití módu Run (RLE) [8].

Na *Obr.13* je vidět aktuální pixel  $x$ , na který je predikce aplikována. Kodér prozkoumá spojitosti mezi pixely a rozhodne, zda kódovat pixel  $x$  v módu Run nebo v módu Regular. Pokud z kontextu vyplývá, že následující pixely  $y$ ,  $z$ , atd. jsou pravděpodobně stejné, kodér zvolí mód Run. Jinak je použit mód Regular. Na *Obr.14* je postup komprese. [8] [39]



*Obr.13* Pozice sousedních hodnot využívaných predikcí ve formátu JPEG-LS [8]



*Obr.14* Postup komprese formátu JPEG-LS [39]

#### 4.9.1 Lossless JPEG

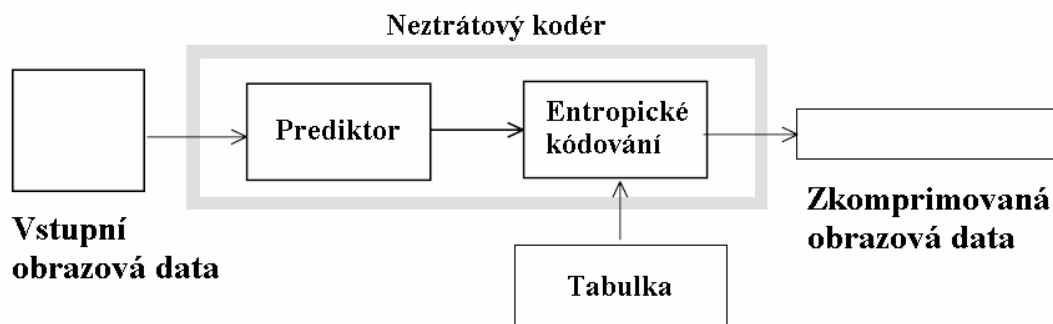
Formátu JPEG-LS předcházela tzv. Lossless JPEG (LJPEG), který je ovšem založen na jiném principu a není tedy možné zaměňovat tyto dva formáty [41].

LJPEG pracuje s předlohami s 2 až 16 bity na vzorek. Vzorek znamená jeden element v dvourozměrném poli (např. jednu hodnotu složky R barevného prostoru RGB). Při

menším počtu bitů na pixel je lepší použít jinou kompresní metodu. JPEG nebyl příliš populární z důvodu, že formát PNG, má u většiny obrazů lepší kompresi. Obecně kvalitnější kompresi vykazuje také JPEG-LS. [3] [40] [41] [42]

LJPEG používá dvourozměrné diferenční pulzně kódované schéma. Základní předpoklad je ten, že hodnota pixelu je kombinována s hodnotami až tří sousedních pixelů tak, aby byla vytvořena predikovaná hodnota. Tato hodnota je poté odečtena od původní hodnoty pixelu. [3] [42]

Až je takto zpracována celá bitmapa, jsou výsledné predikce komprimovány buď Huffmanovou nebo binárně aritmetickou entropní kódovací metodou. Schéma komprese LJPEG je vidět na *Obr.15*. [3] [42]



*Obr.15 Postup komprese formátu LJPEG [42]*

## 4.10 OpenEXR

OpenEXR je open-source formát pro počítačovou grafiku, patřící do kategorie High Dynamic Range (HDR) - obrazy s vysokým dynamickým rozsahem [43]. Jedná se o formát, který ukládá jeden pixel do 16-bitové nebo 32-bitové hodnoty s plovoucí desetinnou čárkou a nebo 32-bitové hodnoty bez znaménka [44].

Na začátku souboru OpenEXR jsou vždy hodnoty 76h 2Fh 31h 01h. Dále následuje hodnota verze, která je datového typu int. Verze může aktuálně nabývat dvou hodnot - 202h (pro dlaždicové uložení pixelů) a 02h (pro uložení pixelů v pruzích). Hodnoty se ukládají v uspořádání malý endian. [45]

Po identifikačním čísle a čísle verze je uložena hlavička. Hlavička je sekvence atributů oddělených nulovým Bytem. Atribut obsahuje jméno, datový typ, velikost a hodnotu. [45]

Po hlavičce je uložena tabulka offsetů, která umožňuje náhodný přístup k jednotlivým pruhům nebo dlaždicím. Poté následují samotné pruhy nebo dlaždice. [45]

Pruhy jsou ukládány po blocích. To kolik pruhů bude uloženo v jednom bloku závisí na použité kompresi. Obrázky uložené pomocí dlaždic umožňují náhodný přístup k obdélníkovému sub-regionu obrázku. [44] [45]

Většina metod datové komprese implementovaných v OpenEXR je neztrátových. OpenEXR také podporuje ztrátovou kompresi a možnost uložení dat bez komprese. Ve formátu OpenEXR jsou implementovány kompresní metody, které znázorňuje *Tab.6*. [44] [45]

*Tab.6 Kompresní schémata formátu OpenEXR* [44] [45]

<b>Kompresce</b>	<b>Neztrátová</b>	<b>Počet pruhů na blok</b>
PIZ	ano	32
ZIP	ano	1 nebo 16
RLE	ano	1
PXR24	ne	16
B44	ne	32
B44A	ne	32

Kompresce PIZ se provádí aplikací vlnkové (wavelet) transformace na data pixelů a výsledek je poté zakódován Huffmanovým kódováním. Kompresce PIZ pracuje dobře jak s obrázkovými daty uloženými v pruzích, tak s daty uloženými ve velkých dlaždicích. Soubory jsou komprimovány a dekomprimovány přibližně stejnou dobu. [44]

#### **4.11 WebPIL**

Nová neztrátová komprese formátu WebP (WebPIL) je v této době na bázi experimentu. Formát WebPIL je pouze dočasný a později splyne s formátem WebP. Kromě vydaných utilit neexistuje žádný program, který tento formát využívá a vzhledem k budoucímu sloučení s formátem WebP se takový nápad jeví jako nevhodný. Formát používá barevný prostor RGBA. [46] [47]

Neztrátová komprese je postavena na transformování obrazu použitím několika rozdílných technik a následného použití entropického kódování na transformované parametry a transformované obrazové data. Transformace aplikovaná na obraz obsahuje prostorovou predikci pixelů, transformaci barevného prostoru, používá lokální palety, zkomprimuje více pixelů do jednoho a nahradí alfu. Pro entropické kódování je zde použita varianta komprese LZ77 a Huffmanovo kódování, které používá 2D hodnoty kódové vzdálenosti. [47]

Při testování balíku obrázků o velikosti 18625893 Bytů došlo k celkové redukci o 28,2 % v porovnání s formátem PNG. 99,4 % obrázků bylo zkomprimováno lépe použitím formátu WebPll než formátu PNG. [47]

Kompresní a dekompresní utility zatím nebyly optimalizovány pro rychlost [47]. Podle [48] došlo při testování po sedmi hodinách provádění komprese na jednom souboru k havárii programu s hlášením o nedostatku paměti. Podobné informace o dlouhé době komprese a nestabilitě lze nalézt také zde: [49]. Větší problém než nestabilita je velká doba komprese, která běžně dosahuje řádů desítek minut až několika hodin. Velikost souboru na použité kvalitě neztrátové komprese příliš nezávisí - rozdíl je v řádu stovek, u větších souborů tisíců kB - ale výrazně ovlivňuje dobu komprese (někdy i v řádu minut). Doba dekomprese je rychlá. [49] [50]

## **II. PRAKTICKÁ ČÁST**

## 5 APLIKACE PRO TESTOVÁNÍ KOMPRESNÍCH ALGORITMŮ

Aplikace má název: „Grafický prohlížeč“. Byla vyvíjena pod operačním systémem Windows 7 Professional a ovládá se pomocí myši. Aplikace je naprogramována v jazyce C++ pomocí vývojového prostředí CodeLite, překladače MinGW a knihovny wxWidgets [51]. GUI je navrženo a generováno v aplikaci wxFormBuilder [52].

K funkčnosti aplikace jsou kromě hlavního spustitelného souboru GrafickyProhlizec.exe zapotřebí také soubory, jejichž seznam je vypsán v *Tab.7* („console\“ znamená název složky, ve které musí být uloženy dané konzolové aplikace).

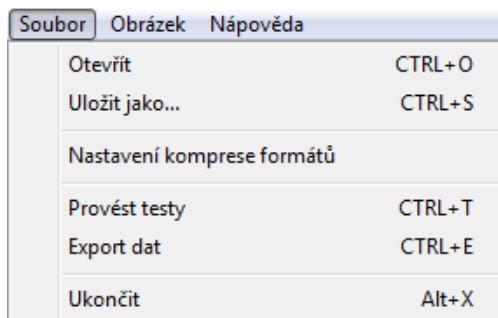
*Tab.7 Seznam souborů nutných pro spuštění aplikace*

Název	Použití	Zdroj
mingwm10.dll	knihovna MinGW (základ aplikace)	[51]
wxmsw28u_gcc_custom.dll	multiplatformní GUI knihovna wxWidgets 2.8.12 (základ aplikace)	[51]
FreeImage.dll	Knihovna pro práci s obrázky běžných formátů (BMP, PNG, TIFF atd.)	[53]
console\JBIGtoPNM.exe console\PNMtoJBIG.exe	konzolové aplikace pro dekódování a kódování formátu JBIG	[54]
console\WMPDecApp.exe console\WMPEncApp.exe	konzolové aplikace pro dekódování a kódování formátu HD Photo	[55]
console\locod.exe console\locoe.exe	konzolové aplikace pro dekódování a kódování formátu JPEG-LS	[56]

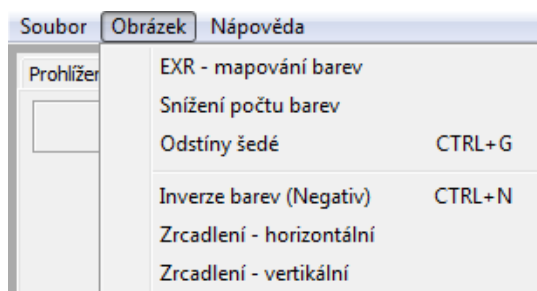
Vzhledem k tomu, že knihovna FreeImage ani knihovna wxWidgets plně nepodporuje formáty PCX, je v aplikaci využito vlastní řešení načítání a ukládání tohoto formátu. Výjimkou je ukládání 24 bitových obrázků PCX, které je provedeno s využitím wxWidgets.

### 5.1 Možnosti aplikace

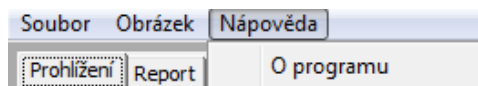
Aplikace má tři hlavní záložky v menu - Soubor ; Obrázek ; Nápověda. Obsah těchto záložek je vidět na *Obr.16*, *Obr.17* a *Obr.18*.



Obr.16 Záložka v menu - Soubor



Obr.17 Záložka v menu - Obrázek



Obr.18 Záložka v menu - Nápověda

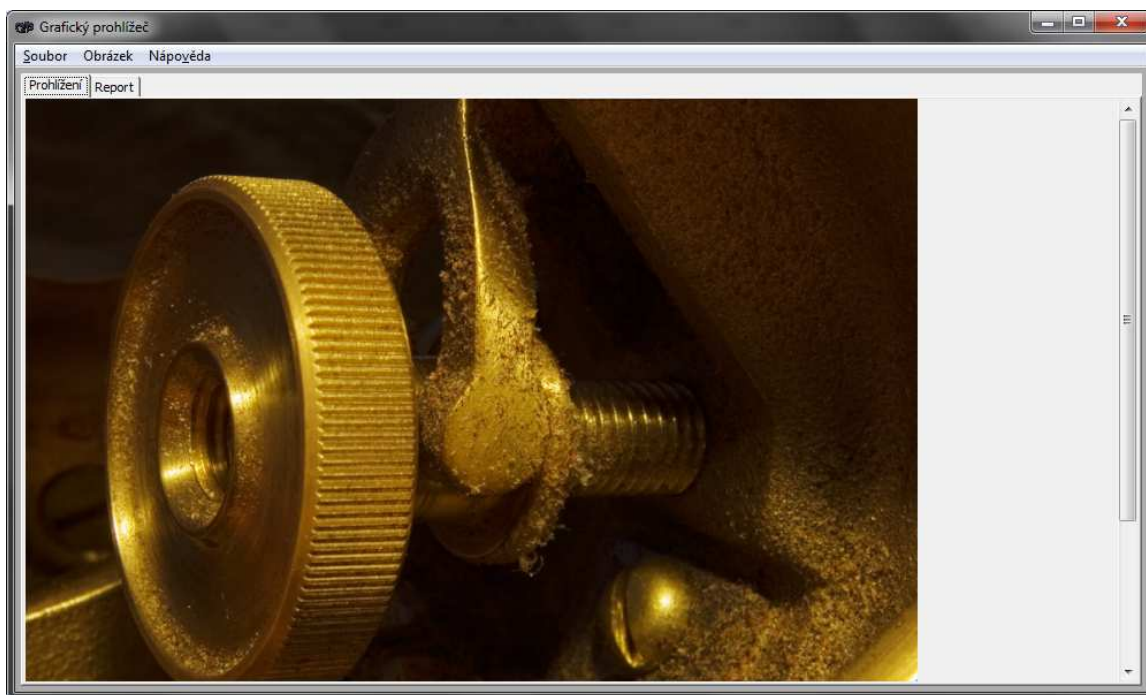
### 5.1.1 Menu Soubor

Pro otevření souboru s obrazovými daty slouží položka „Otevřít“, pro jeho uložení do příslušného formátu lze použít položku „Uložit jako...“.

Aplikace podporuje formáty BMP (i s kompresí RLE), GIF, HD Photo (1, 8 (odstíny šedi), 24, 32 bitů, Gray FLOAT, RGBAF (HD Photo nepodporuje RGBF, ale jen RGBAF s možností ignorovat alfa kanál)), JBIG (1, 8 (odstíny šedi) bitů), JPEG 2000 (8 (odstíny šedi) 24 a 32 bitů), JPEG-LS (8 (odstíny šedi) 24 bitů), OpenEXR (Gray FLOAT, RGBF), PCX, PNG, TGA (>8 bitů), TIFF (podporuje i Gray FLOAT a RGBF).

Díky knihovně FreeImage lze otevřít i formáty jako JPEG, PBM, PGM, PPM, formáty fotoaparátů (RAW) jako BMQ, DNG, NEF a další (kompletní soupis formátů je možné

nalézt v dokumentaci knihovny FreeImage [57]). Prioritně však nejsou aplikací podporovány, a tak lze tyto soubory otevřít jen zápisem jejich názvu v dialogu pro otevření. V záložce prohlížení je možné zhlédnout aktuálně načtený obrázek (viz *Obr.19*).



*Obr.19 Prohlížení obrázků v aplikaci*

Při otevření/uložení souboru se měří doba, za kterou byl soubor otevřen/uložen. Při ukládání se dále vypočte kompresní poměr a veličina, která byla nazvána jako „Časová efektivita komprese“. Tuto veličinu lze popsat následujícím vzorcem:

$$\text{ČEK} = \frac{1 - KP}{DK} \cdot 100 \left[ \frac{\%}{\text{ms}} \right]$$

ČEK - časová efektivita komprese

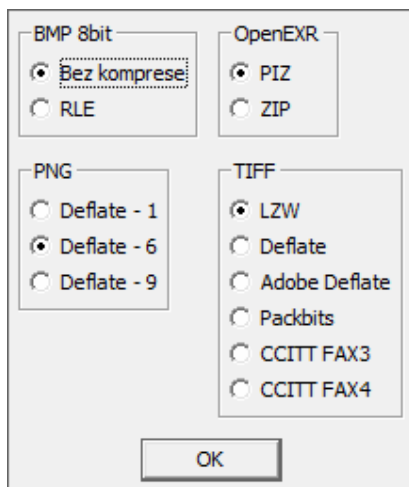
KP - kompresní poměr

DK - doba komprese

Časová efektivita komprese vyjadřuje o kolik procent se zmenší velikost souboru za jednu milisekundu. Při záporné kompresi se tato hodnota pohybuje v záporných číslech - interval možných výsledků je tedy  $(-\infty; 100)$ .

Výše zmíněná veličina je inspirována parametrem popsáním v kapitole 2.2 - konkrétně se jedná o poměr mezi dobou komprese a kompresním poměrem. Tento parametr určuje dobu komprese na jednotku kompresního poměru, přičemž čím vyšší číslo tím lépe, a pokud je výsledek menší než doba komprese došlo k záporné kompresi. Parametr je však na první pohled méně srozumitelný a vyžaduje větší přemýšlení k pochopení jeho účelu, a proto byla použita výše zmíněná modifikace, která vyznívá jasněji.

V položce „Nastavení komprese formátů“ je možné nastavit typ výstupní komprese u formátů BMP, EXR, PNG a TIFF. Konkrétní možnosti nastavení komprese jsou znázorněny na Obr.20. U formátu BMP může být RLE komprese využita jen u 8 bitových obrázků. Komprese „Deflate - 1“ označuje nejnižší úroveň komprese Deflate a „Deflate - 9“ naopak úroveň nejvyšší. Při zvolení komprese CCITT FAX3 a FAX4 budou obrázky převedeny na dvoubarevné - převod barev se uskuteční pomocí zvoleného algoritmu v položce „Snížení počtu barev“.



*Obr.20 Dialog s nastavením komprese formátů*

V aplikaci lze automaticky provést testy kompresních algoritmů na větším počtu souborů. Toto se provede pomocí položky „Provést testy“. Před zahájením je potřeba vybrat složku se vstupními obrázkovými daty, přičemž dojde k prohledání i případných podsložek. Dále se zvolí složka, do které se uloží obrázky v různých formátech a popř. i s různými typy kompresí. Obrázky se stejnou příponou a různým typem komprese (jedná se o formáty

uvedené v dialogu „Nastavení komprese formátů“) jsou v názvu souboru doplněny o tvar „\_<typ komprese>“ - např. jedním z možných výstupů obrázku s názvem „obrázek.bmp“ bude „obrázek\_deflate1.png“.

Jako vstupní obrázky v automatickém testu jsou podporovány formáty s příponou BMP, CR2, EXR, GIF, HDR, J2C, J2K, JP2, JPC, JPEG, JPG, NEF, ORF, PFM, PNG, RAF, TGA, TIF a TIFF. Po načtení obrázku se provede až 16 testů kompresních algoritmů - volba algoritmů je na uživateli. Testovat lze formáty a komprese uvedené v *Tab.8*.

*Tab.8 Volitelné formáty a komprese pro provádění testů*

<b>Formát</b>	<b>Komprese</b>
BMP	RLE
EXR	PIZ
EXR	ZIP
GIF	LZW
HD Photo (WDP)	LOSSLESS WAVELET
JBIG	JBIG
JPEG 2000	LOSSLESS WAVELET
JPEG-LS	LOCO-I
PCX	RLE
PNG	DEFLATE 1
PNG	DEFLATE 9
TGA	RLE
TIFF	LZW
TIFF	PACKBITS
TIFF	CCITT FAX 3
TIFF	CCITT FAX 4

Při provádění testů je zobrazen ukazatel průběhu a postupně se do reportu aplikace doplňují parametry jednotlivých souborů (viz *Obr.21*). Průběh testů nelze pozastavit ani zastavit. Doba komprese a dekomprese se měří dvakrát a výsledkem je průměr těchto hodnot.

Název	BH [bpp]	BP	OŠ	Šířka [px]	Výška [px]	DD [ms]	Typ K	DK [ms]	KP	ČEK [% / ms]
08d880fe-a592-483d-8a31-b508716750ea_C_24	24	RGB(A)	N	387	420	93	HDPHOTO - LOSSLESS_WAVE	118	0.222333	0.659040
08d880fe-a592-483d-8a31-b508716750ea_C_24	24	RGB(A)	N	387	420	117	JP2 - LOSSLESS_WAVELET	242	0.142704	0.354254
08d880fe-a592-483d-8a31-b508716750ea_C_24	24	RGB(A)	N	387	420	30	JLS - LOCO-1	30	0.155354	2.815486
08d880fe-a592-483d-8a31-b508716750ea_C_24	24	RGB(A)	N	387	420	16	PCX - RLE	31	0.401864	1.929471
08d880fe-a592-483d-8a31-b508716750ea_C_24	24	RGB(A)	N	387	420	1	PNG - DEFLATE_1	39	0.166280	2.137743
08d880fe-a592-483d-8a31-b508716750ea_C_24	24	RGB(A)	N	387	420	1	TGA - RLE	8	0.340649	8.241884
08d880fe-a592-483d-8a31-b508716750ea_C_24	24	RGB(A)	N	387	420	16	TIFF - LZW	62	0.228505	1.244346
08d880fe-a592-483d-8a31-b508716750ea_C_24	24	RGB(A)	N	387	420	1	TIFF - PACKBITS	78	0.448515	0.707033
101.wdp	24	RGB(A)	N	987	831	515	HDPHOTO - LOSSLESS_WAVE	609	1.230283	-0.037813
101.jp2	24	RGB(A)	N	987	831	804	JP2 - LOSSLESS_WAVELET	1404	0.799196	0.014302
101.jls	24	RGB(A)	N	987	831	116	JLS - LOCO-1	148	0.814968	0.125022
101.pcx	4	RGB(A)	N	301	429	1	PCX - RLE	8	0.499555	6.255565
101_deflate1.png	8	RGB(A)	N	987	831	16	PNG - DEFLATE_1	16	0.144730	5.345441
101.tga	8	RGB(A)	N	987	831	16	TGA - RLE	31	0.449033	1.777312
101_LZW.tiff	8	RGB(A)	N	987	831	47	TIFF - LZW	70	0.346172	0.934040
101_Packbits.tiff	8	RGB(A)	N	987	831	16	TIFF - PACKBITS	78	2.015721	-1.302206
160.wdp	24	RGB(A)	N	301	429	86	HDPHOTO - LOSSLESS_WAVE	110	2.163610	-1.057827
160.jp2	24	RGB(A)	N	301	429	109	JP2 - LOSSLESS_WAVELET	203	1.347044	-0.170958
160.jls	24	RGB(A)	N	301	429	22	JLS - LOCO-1	22	1.449010	-2.040953
160.pcx	4	RGB(A)	N	301	429	1	PCX - RLE	1	0.475584	52.441601
160_deflate1.png	4	RGB(A)	N	301	429	1	PNG - DEFLATE_1	1	0.325230	67.476962
160.tga	8	RGB(A)	N	301	429	1	TGA - RLE	8	0.613783	4.827710
160_LZW.tiff	4	RGB(A)	N	301	429	1	TIFF - LZW	24	0.316842	2.846493
160_Packbits.tiff	4	RGB(A)	N	301	429	1	TIFF - PACKBITS	16	0.416343	3.647858
2007-08-18-TajemstvíKamenu.wdp	24	RGB(A)	N	397	496	148	HDPHOTO - LOSSLESS_WAVE	195	0.644483	0.182316
2007-08-18-TajemstvíKamenu.jp2	24	RGB(A)	N	397	496	328	JP2 - LOSSLESS_WAVELET	554	0.610918	0.070231
2007-08-18-TajemstvíKamenu.jls	24	RGB(A)	N	397	496	116	JLS - LOCO-1	92	0.564957	0.472873

Obr.21 Průběh provádění testů

Po provedení testů se zobrazí dialog s dotazem, zda se má uložit report do souboru formátu CSV (hodnoty oddělené středníkem). Pokud je zvolena možnost „Ano“, dojde k uložení výsledků testů. V průběhu testu se do složky pro dočasné soubory (Temp) ukládají průběžně získané informace do souboru formátu CSV. Toto řešení je využito kvůli případným pádům aplikace při testování - soubor lze kdykoliv ze složky Temp přesunout, a tak tedy nedojde ke ztrátě výsledků před havárií. Pojmenování tohoto souboru je ve formátu „GrafickyProhlizec\_<rok>-<měsíc>-<den>\_<hodina>-<minuta>-<sekunda>.csv“ (např. GrafickyProhlizec\_2012-3-23\_19-28-40).

Výsledný report, který je vidět na Obr.22 lze exportovat položkou „Export dat“. Rozdíl mezi tímto exportem a exportem po provedení testů je ten, že v tomto reportu může být uloženo i více reportů z předešlých testů, a nebo výsledky jednotlivých souborů, které se v kompletním testu neobjevily.

Prohlázení	Report	Název	BH [bpp]	BP	OŠ	Šířka [pix]	Výška [pix]	DD [ms]	Typ K	DK [ms]	KP	ČEK [% / ms]
77		Fog_LZW.tif	32	FLOAT	A	1000	672	52	TIFF - LZW	107	0.001091	0.26071
80		Fog_Packbits.tiff	32	FLOAT	A	1000	672	8	TIFF - PACKBITS	71	1.009061	-0.012762
81		jilek_vytrvaly.wdp	24	RGB(A)	N	363	629	124	HDPHOTO - LOSSLESS_WAVE	172	0.258665	0.431008
82		jilek_vytrvaly.jp2	24	RGB(A)	N	363	629	172	JP2 - LOSSLESS_WAVELET	312	0.165929	0.267331
83		jilek_vytrvaly.js	24	RGB(A)	N	363	629	38	JLS - LOCO-I	38	0.208284	2.083463
84		jilek_vytrvaly.pcx	24	RGB(A)	N	363	629	32	PCX - RLE	31	0.327765	2.168500
85		jilek_vytrvaly_deflate1.png	24	RGB(A)	N	363	629	31	PNG - DEFLATE_1	47	0.210196	1.680435
86		jilek_vytrvaly.tga	24	RGB(A)	N	363	629	8	TGA - RLE	8	0.276158	9.048025
87		jilek_vytrvaly_LZW.tiff	24	RGB(A)	N	363	629	16	TIFF - LZW	47	0.261905	1.570416
88		jilek_vytrvaly_Packbits.tiff	24	RGB(A)	N	363	629	15	TIFF - PACKBITS	8	0.278011	9.024660
89		PIA12941.wdp	8	RGB(A)	A	1000	1000	296	HDPHOTO - LOSSLESS_WAVE	320	0.589563	0.128261
90		PIA12941.jp2	8	RGB(A)	A	1000	1000	577	JP2 - LOSSLESS_WAVELET	936	0.551463	0.047921
91		PIA12941.js	8	RGB(A)	A	1000	1000	132	JLS - LOCO-I	132	0.495987	0.381828
92		PIA12941.pcx	8	RGB(A)	A	1000	1000	39	PCX - RLE	16	0.842349	0.985319
93		PIA12941_deflate1.png	8	RGB(A)	A	1000	1000	47	PNG - DEFLATE_1	204	0.629309	0.181711
94		PIA12941.tga	8	RGB(A)	A	1000	1000	24	TGA - RLE	39	0.899046	0.258857
95		PIA12941_LZW.tiff	8	RGB(A)	A	1000	1000	31	TIFF - LZW	94	0.738517	0.278174
96		PIA12941_Packbits.tiff	8	RGB(A)	A	1000	1000	1	TIFF - PACKBITS	31	0.865999	0.432260
97		PIA_01.wdp	24	RGB(A)	N	150	729	78	HDPHOTO - LOSSLESS_WAVE	86	0.443719	0.646838
98		PIA_01.jp2	24	RGB(A)	N	150	729	94	JP2 - LOSSLESS_WAVELET	171	0.177378	0.481066
99		PIA_01.js	24	RGB(A)	N	150	729	22	JLS - LOCO-I	14	0.179208	5.862803
100		PIA_01.pcx	24	RGB(A)	N	150	729	1	PCX - RLE	8	0.079836	11.502047
101		PIA_01_deflate1.png	24	RGB(A)	N	150	729	8	PNG - DEFLATE_1	8	0.046838	11.914526
102		PIA_01.tga	24	RGB(A)	N	150	729	1	TGA - RLE	8	0.183489	10.206388
103		PIA_01_LZW.tiff	24	RGB(A)	N	150	729	16	TIFF - LZW	24	0.078984	3.837568
104		PIA_01_Packbits.tiff	24	RGB(A)	N	150	729	1	TIFF - PACKBITS	16	0.134554	5.409035
105		sandra.wdp	8	RGB(A)	A	150	179	15	HDPHOTO - LOSSLESS_WAVE	15	0.690695	2.062033
106		sandra.jp2	8	RGB(A)	A	150	179	15	JP2 - LOSSLESS_WAVELET	40	0.604928	0.987679

Obr.22 Report ve vytvořené aplikaci

Výsledný soubor lze následně spustit v tabulkovém editoru např. v programu Microsoft Excel nebo LibreOffice. Výsledek pak může vypadat jako na Obr.23.

1	Výsledky testů kompresních algoritmů	Grafický prohlížeč - Diplomová práce - Bc. Tomáš Vogetanz
2	27.3.2012 15:52	
3		
4	Název	BH [bpp] BP OŠ Šířka [pix] Výška [pix] DD [ms] Typ K DK [ms] KP ČEK [% / ms]
5	amonamrath (2)_rle.bmp	8 RGB(A) N 1680 1050 52 BMP - RLE 19 0.720098 1.410327
6	amonamrath (2)_piz.exr	96 RGBF N 1680 1050 150 EXR - PIZ 299 0.500277 0.940333
7	amonamrath (2)_zip.exr	96 RGBF N 1680 1050 227 EXR - ZIP 1723 0.904492 0.005543
8	amonamrath (2)_gif	8 RGB(A) N 1680 1050 129 GIF - LZW 171 0.54169 0.268053
9	amonamrath (2)_wdp	24 RGB(A) N 1680 1050 654 HDPHOTO - LOSSLESS_WAVELET 759 0.407115 0.080226
10	amonamrath (2)_jp2	24 RGB(A) N 1680 1050 1141 JP2 - LOSSLESS_WAVELET 1874 0.341179 0.035049
11	amonamrath (2)_js	24 RGB(A) N 1680 1050 387 JLS - LOCO-I 329 0.375655 0.18977
12	amonamrath (2)_pcx	24 RGB(A) N 1680 1050 113 PCX - RLE 205 0.915328 0.041304
13	amonamrath (2)_deflate1.png	24 RGB(A) N 1680 1050 126 PNG - DEFLATE_1 446 0.032086 0.121709
14	amonamrath (2)_deflate9.png	24 RGB(A) N 1680 1050 117 PNG - DEFLATE_9 3830 0.400145 0.015662
15	amonamrath (2)_tga	24 RGB(A) N 1680 1050 18 TGA - RLE 51 0.814888 0.362965
16	amonamrath (2)_LZW.tiff	24 RGB(A) N 1680 1050 89 TIFF - LZW 155 0.53775 0.298226
17	amonamrath (2)_Packbits.tiff	24 RGB(A) N 1680 1050 19 TIFF - PACKBITS 64 0.899462 0.153778
18	amonamrath (2)_CCITTFAX3.tiff	1 RGB(A) A 1680 1050 15 TIFF - CCITTFAX3 40 1.637933 1.594884
19	amonamrath (2)_CCITTFAX4.tiff	1 RGB(A) A 1680 1050 26 TIFF - CCITTFAX4 83 1.700928 0.844493
20	ccitt1_rle.bmp	8 RGB(A) A 1728 2376 22 BMP - RLE 13 0.29793 5.402079
21	ccitt1_piz.exr	32 FLOAT A 1728 2376 124 EXR - PIZ 217 0.200759 0.354498
22	ccitt1_zip.exr	32 FLOAT A 1728 2376 136 EXR - ZIP 242 0.113441 0.366347
23	ccitt1_gif	1 RGB(A) A 1728 2376 46 GIF - LZW 94 0.082978 0.975555
24	ccitt1_wdp	1 RGB(A) A 1728 2376 263 HDPHOTO - LOSSLESS_WAVELET 300 0.163134 0.278855
25	ccitt1.jp2	8 RGB(A) A 1728 2376 499 JP2 - LOSSLESS_WAVELET 836 0.608033 0.046886
26	ccitt1.js	8 RGB(A) A 1728 2376 17 JLS - LOCO-I 16 0.075353 5.779044
27	ccitt1.pcx	1 RGB(A) A 1728 2376 9 PCX - RLE 2 0.150059 42.497048
28	ccitt1_deflate1.png	1 RGB(A) A 1728 2376 7 PNG - DEFLATE_1 13 0.577463 7.056434
29	ccitt1_deflate9.png	1 RGB(A) A 1728 2376 4 PNG - DEFLATE_9 77 0.059993 1.220789
30	ccitt1.tga	8 RGB(A) A 1728 2376 26 TGA - RLE 73 0.294589 0.966317
31	ccitt1_LZW.tiff	1 RGB(A) A 1728 2376 7 TIFF - LZW 97 0.083561 0.944783
32	ccitt1_Packbits.tiff	1 RGB(A) A 1728 2376 3 TIFF - PACKBITS 16 0.184253 5.390308

Obr.23 Výsledky testů otevřené v aplikaci Microsoft Excel

### 5.1.2 Menu Obrázek

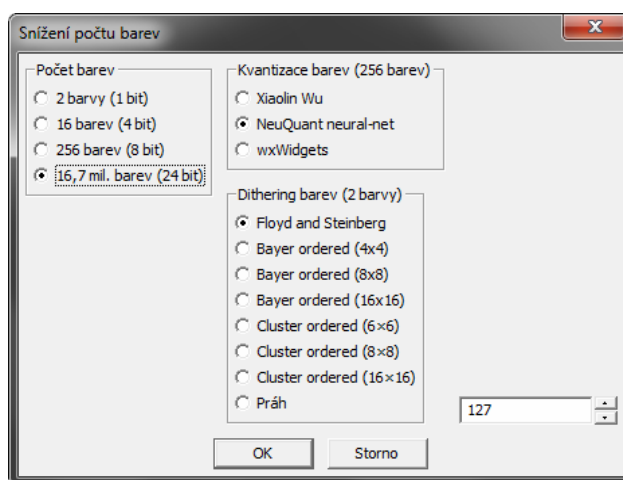
Práce s obrázky je převážně soustředěna na barevné prostory RGB, RGBA, Gray FLOAT, RGBF a RGBAF, ale je možné pracovat i s obrázky RGB16 a RGBA16.

V menu Obrázek se vyskytuje funkce horizontálního a vertikálního zrcadlení, inverze barev (nelze provést pro obrázky s barevným prostorem Gray FLOAT, RGBF a RGBAF), převod do odstínu šedi, snížení počtu barev obrázku a mapování barev (resp. tónování barev).

Výstupní bitová hloubka obrázku při převodu do odstínu šedi závisí na barevném prostoru vstupního obrázku - u barevného prostoru RGBF, RGBAF, RGB16 a RGBA16 dojde k převodu na barevný prostor Gray FLOAT (32 bitů), u barevných prostorů RGB a RGBA dochází k převodu na 8 bitový obrázek.

Po kliknutí na položku „Snížení počtu barev“ se objeví dialog zobrazený na *Obr.24*. Tento dialog umožňuje výběr počtu barev, na který se bude obrázek převádět a algoritmy, kterými se převod provede. Algoritmy se volí pro 2 a 256 barev, při převodu na 16 barev je vždy použita knihovna wxWidgets. Knihovna FreeImage sice také umožňuje kvantizaci na 16 barev, ale výsledná bitová hloubka obrázku je 8 bitů, a nikoliv 4 bity, jak by bylo vhodné - použitím knihovny wxWidgets byl tento problém vyřešen. Přebod na 16,7 mil. barev se provádí pro barevné prostory RGB16, RGBA16, Gray FLOAT, RGBF a RGBAF. Při snížení barev na 24 bitů je využito vybraného algoritmu v dialogu „EXR - mapování barev“.

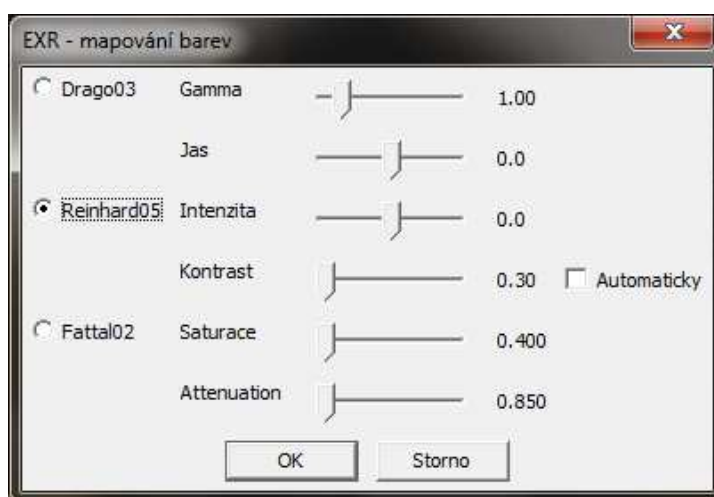
Převod se neprovede, pokud je zvolená bitová hloubka větší nebo rovna bitové hloubce vstupního obrázku, popř. je bitová hloubka rovna 1. Při provádění testů se u případných převodů barev (např. při ukládání do formátu GIF nebo CCITT FAX3) použijí aktuálně zvolené algoritmy v tomto dialogu.



*Obr.24 Dialog pro snížení počtu barev*

Položka „EXR - mapování barev“ slouží pro určení algoritmu, který provede převod barev z formátu obsahující vysoký dynamický rozsah barev (převážně určeno pro formát EXR, ale také pro TIFF a popř. i HDR a PFM). Mapování barev se využívá při zobrazování tohoto typu obrázku - na výchozí obrázek s vysokou dynamikou se aplikuje zvolená metoda a dojde k převodu na 24 bitovou hloubku, díky které lze obrázek vykreslit běžnými programovacími technikami.

Zobrazený dialog je vidět na *Obr.25*, ve kterém jsou vyplněny výchozí hodnoty, které se pro mapování barev používají. Každému typu algoritmu přísluší dva parametry, např. při volbě algoritmu Drago03 lze nastavit parametry Gamma a Jas, ostatní parametry v tu chvíli nemají na funkčnost vliv. Při mapování barev většinou dochází ke snížení počtu barev v obrázku.



*Obr.25 Dialog pro mapování barev formátu s vysokým dynamickým rozsahem*

### 5.1.3 Menu Nápověda

V menu „Nápověda“ je pouze položka s názvem „O aplikaci“. Tato položka obsahuje informace o podporovaných formátech, použitých knihovnách, konzolových aplikacích a také vysvětlivky ke zkratkám v hlavních sloupcích reportu. Význam těchto zkratek je také popsán v *Tab.9*.

Tab.9 Význam zkratek v reportu

Zkratka	Význam
Název	název souboru s obrázkem
BH [bpp]	bitová hloubka v bitech na pixel
BP	barevný prostor
OŠ	odstíny šedi (Ano/Ne)
Šířka [pix]	šířka obrázku v pixelech
Výška [pix]	výška obrázku v pixelech
DD [ms]	doba dekomprese v ms
Typ K	typ (výstupní) komprese
DK [ms]	doba komprese v ms
KP	kompresní poměr
ČEK [% / ms]	časová efektivita komprese (viz kapitola 5.1.1)

## 5.2 Popis zdrojových kódů aplikace

Projekt obsahuje následující soubory, které byly vytvořeny v aplikaci wxFormBuilder:

- `exr_mapovanibarev.fbp` - dialog pro položku „EXR - mapování barev“
- `gui.fbp` - grafické rozhraní hlavní aplikace
- `nastavenikompreseformatu.fbp` - dialog pro položku „Nastavení komprese formátů“
- `oaplikaci.fbp` - dialog pro položku „O aplikaci“
- `snizenipoctubarev.fbp` - dialog pro položku „Snížení počtu barev“
- `ukazatelstavu.fbp` - dialog s ukazatelem průběhu
- `vyberformatutestu.fbp` - dialog s výběrem formátů, které se budou testovat

Po vygenerování zdrojového kódu v jazyce C++ přes aplikaci wxFormBuilder vznikly stejnojmenné soubory s příponou `.h` (hlavička) a `.cpp` (zdrojový kód).

Dalšími soubory jsou `exr_mapovanibarev_main.h`, `exr_mapovanibarev_main.cpp`. Tyto dva zmíněné soubory implementují dialog pro položku „EXR - mapování barev“ a obslužné metody pro událost změny pozice posuvníků, ovlivňujících hodnoty parametrů algoritmů.

Konkrétní intervaly pro jednotlivé parametry jsou:

- Gamma -  $\langle 0,05 ; 8 \rangle$
- Jas -  $\langle -8 ; 8 \rangle$
- Intenzita -  $\langle -8 ; 8 \rangle$
- Kontrast -  $\langle 0,3 ; 1 \rangle$
- Saturace -  $\langle 0,4 ; 0,6 \rangle$
- Attenuation -  $\langle 0,8 ; 0,9 \rangle$

Soubor FreeImage.h umístěný ve složce h\_pomocne slouží jen jako pomocná knihovna, v aplikaci nemá žádné využití. Z této knihovny byly použity struktury, výčtové typy, informace o funkcích a některá makra pro implementaci třídy FreeImageRozhraniClass v souboru freeimagerozhrani.h.

Posledními čtyřmi soubory jsou main.h, main.cpp, obrazek.h a obrazek.cpp. Soubory main obsahují implementaci hlavního grafického rozhraní a jsou jádrem celé aplikace. V souborech obrazek je vytvořena třída Obrazek, která umožňuje načtení a uložení souborů formátu BMP (1, 4, 8, 24 bitů) a PCX (1, 4, 8, 24 bitů - u 24 bitů pouze načtení).

### 5.2.1 Třída Obrazek

Diagram této třídy a její asociace zobrazuje Příloha P I. Třída Obrazek je implementována v souborech obrazek.h a obrazek.cpp. Mimo ní jsou zde vytvořeny struktury hlaviček formátu BMP a PCX a výčtový typ pro určení formátu, který byl načten.

Třída obsahuje atributy s informacemi o formátu obrázku, bitové hloubce, šířce, výšce apod. a samozřejmě také samotná obrazová data. Obrazová data jsou uložena ve formátu BMP. Když se tedy načte formát PCX, tak jsou poté data dekodována do formátu BMP.

Metoda pro načtení (NacistDataZeSouboru) a dekodování (Dekodovat) obrázku jsou ve třídě odděleny kvůli eliminaci vlivu času načítání souboru z pevného disku při měření doby dekomprese. Kvůli stejnému důvodu je také odděleno kódování (Kodovat) a uložení souboru (UlozitDataDoSouboru). Kvůli většímu komfortu zde je však také implementována metoda, která provede obě operace najednou - tedy načtení i dekodování (NacistADekodovatZeSouboru) resp. kódování i uložení (KodovatAUlozitDoSouboru).

Dekódování je provedeno způsobem, který umožňuje použít stejný algoritmus pro obrázky s jednou bitovou rovinou a bitovou hloubkou 1, 4, 8, a také se třemi bitovými rovinami a bitovou hloubkou 8 (24 bitů) a navíc není nutné provést po dekodování vertikální zrcadlení nebo převod na barevný prostor BGR. Toto je možné díky následujícímu výpočtu indexu pole, na který se má aktuálně dekodovaná hodnota vložit:

$$I = (V - R - 1) \cdot (BnR - DBR) \cdot BR + UBR \cdot BR + BR - 1 - AR + PDB \cdot (V - R - 1)$$

I - index, na který se hodnota uloží (prvním indexem je 0)

V - výška obrázku v pixelech (vypočítaná z hlavičky PCX)

R - aktuálně zpracovávaný řádek obrázku

BnR - počet Byte na řádek (z hlavičky PCX)

DBR - doplnění Bytu na řádek pro PCX

BR - počet bitových rovin (z hlavičky PCX)

UBR - počet již zpracovaných Bytů z aktuální dekodované roviny

AR - aktuálně dekodovaná rovina (1. rovina je označena číslem 0)

PDB - počet doplňujících Byte pro BMP (aby byla velikost řádku dělitelná čtyřmi)

Proměnná DBR se určí následujícím vzorcem:

$$DBR = BnR - \frac{S \cdot BH}{8}$$

BnR - počet Byte na řádek (z hlavičky PCX)

S - šířka obrázku v pixelech

BH - bitová hloubka obrázku (počet bitů na pixel z hlavičky PCX)

Proměnnou PDB lze vypočítat takto:

$$PDB = \text{ceil}\left(\frac{S \cdot BH}{8 \cdot 4}\right) \cdot 4 - \frac{S \cdot BH}{8}$$

S - šířka obrázku v pixelech

BH - bitová hloubka obrázku (počet bitů na pixel pro BMP)

ceil - funkce pro zaokrouhlení desetinného čísla vždy na vyšší hodnotu

Kódování bylo naprogramováno jen pro bitovou hloubku 1, 4 a 8 bitů. PCX s 8 bity na pixel a 3 bitovými rovinami se v aplikaci ukládá pomocí knihovny wxWidgets.

### 5.2.2 Třída FreeImageRozhraniClass

Diagram této třídy a její asociace zobrazuje Příloha P II. Třída FreeImageRozhraniClass je implementována v souboru freeimagerozhrani.h. Jedná se o rozhraní mezi knihovnou FreeImage a aplikací. Mimo ní jsou zde ze souboru FreeImage.h převedeny struktury, výčtové typy, makra a definovány datové typy ukazatelů na konkrétní typ funkce pro načtení těchto funkcí z dynamické knihovny FreeImage.dll.

V této třídě patří mezi hlavní atributy ukazatel na aktuálně načtený obrázek, jeho formát, handle DLL knihovny a informace o typu kvantizačního, mapovacího a dithering algoritmu, které jsou využívány metodou pro snížení počtu barev v obrázku.

Zjednodušený příklad načtení dynamické knihovny a funkce v této knihovně je ukázán v následujícím zdrojovém kódu:

```
1 #include <stdio.h>
2 #include <windows.h>
3 //definování datového typu - ukazatel na funkci Load
4 typedef FIBITMAP* (__stdcall * FREEIMAGE_LOAD)(FREE_IMAGE_FORMAT
   fif, const char *filename, int flags);
5 //ukazatel na funkci Load
6 FREEIMAGE_LOAD FreeImage_Load;
7 //handle na dll knihovnu
8 HINSTANCE m_hDll;
9 //pokud se zdaří načtení dll knihovny
10 if((m_hDll = LoadLibrary(_("FreeImage.dll"))) != NULL)
11 {
12     //dojde k pokusu o načtení funkce s daným názvem (přesný název
   funkce byl zjištěn otevřením souboru FreeImage.dll v hex-editoru)
13     FreeImage_Load = (FREEIMAGE_LOAD)GetProcAddress(m_hDll,
   "_FreeImage_Load@12");
14     //pokud se funkci nepodařilo načíst vypíše se chyba
15     if(FreeImage_Load == NULL)
16     {
17         printf("chyba pri nacistani funkce FreeImage_Load\n");
18     }
}
```

Třída `FreeImageRozhraniClass` obsahuje metody pro práci s obrázky, které využívají načtených funkcí z dynamické knihovny. K nejdůležitějším patří metody načtení (`NacistObrazek`) a uložení (`UlozitObrazek`) obrázku, konverze barevného prostoru (`PrevodBarevnehoProstoru`), dále zjištění bitové hloubky, šířky, výšky, barevného prostoru, převod na odstíny šedé, snížení počtu barev, zrcadlení, inverze barev a nakonec i převod bitmapy využívané knihovnou `FreeImage` (struktura `FIBITMAP`) na objekt třídy `wxImage` (`FIBITMAP_na_WxImage`). Poslední metoda je naprogramována kvůli zobrazení obrázku v prohlížečím okně aplikace pomocí knihovny `wxWidgets`.

Metodu pro převod barevného prostoru (resp. převod bitové hloubky) je dobré použít před uložením samotného souboru. Dochází zde totiž ke snížení bitové hloubky v závislosti na zvoleném formátu, který má být pro uložení použit. Pokud se tato metoda před uložením neprovede, může dojít k vytvoření souboru s nulovou délkou - jinými slovy uložení souboru se nezdaří.

### 5.2.3 Třída `MainFrame`

Diagram této třídy a její asociace zobrazuje Příloha P III. Ve třídě `MainFrame` (soubor `main.h` a `main.cpp`) je implementováno základní grafické rozhraní a obslužné metody událostí. Základem jsou tedy obslužné metody pro kliknutí na položky v menu - např. `OnSouborOtevritClick`, `OnSouborUlozitJakoClick`, `OnSouborNastaveniKompreseClick`, `OnSouborProvestTestyClick` apod. Dále je zde také obslužná metoda pro vykreslování obrázků do okna s možností skrolování (`OnScrolledWindowProhlizeniPaint`). Aby se obrázek zbytečně nevykresloval celý, byly naprogramovány metody pro výpočet oblasti, kterou je nutné vykreslit (`VypocetPocatkuAKonceBitmapy`, `VypocetPosunuPozice`). Před vykreslením pak dojde k "vystřížení" této vypočtené oblasti a následně se zobrazuje jen tato část bitmapy - tím se sníží zátěž procesoru při skrolování u obrázků s vysokým rozlišením.

Při načítání obrázků je nejdříve využita knihovna `FreeImage`. Pokud se knihovně nepovede načíst vstupní soubor, dojde k pokusu o načtení souboru `PCX` (pomocí objektu třídy `Obrazek`). Pokud i tento pokus selže testuje se přípona souboru, a pokud je totožná s příponami pro formát `HD Photo`, `JBIG` nebo `JPEG-LS`, tak se vytvoří proces a spustí se příslušná konzolová aplikace (provede se metoda `ProcesSpustit`). U uložení se jedná o něco podobného, jen je zde specifikován výstupní formát, takže je vždy možné určit, která

komponenta se o uložení postará. Po uložení obrázku je určen kompresní poměr a časová efektivita komprese.

Pro spuštění konzolových aplikací se vytvoří synchronní proces a je zjištěno jeho PID. Synchronní znamená, že aplikace čeká na dokončení tohoto procesu. Poté se z výchozí zprávy aplikace zjistí informace o době komprese, popř. dekomprese. Soubory potřebné pro převod formátů pomocí této aplikace jsou ukládány do složky dočasných souborů (Temp). Ve chvíli, kdy jsou soubory nepotřebné dojde k jejich smazání. Názvy těchto souborů se vytvářejí z aktuálního data a času.

Pro výpočet kompresního poměru se používá metoda `ZjistitKompresniPomer` a pro výpočet časové efektivity metoda `ZjistitCasovouEfektivitu`. Pro výpočet kompresního poměru se stanoví velikost uloženého souboru a dále se podle výšky šířky a bitové hloubky vypočte velikost, kterou by měl soubor BMP v nekomprimované podobě - počítá se zde i s hlavičkou souboru a informační hlavičkou, zarovnáním na řádky dělitelné čtyřmi, a případnou paletou barev. Vzorec pro výpočet velikosti souboru BMP vypadá takto:

$$VBMP = \text{ceil}\left(\frac{S \cdot V \cdot BH}{8}\right) + PDB \cdot V + VH + VP$$

VBMP - velikost souboru BMP

S - šířka obrázku v pixelech

V - výška obrázku v pixelech

BH - bitová hloubka obrázku

PDB - počet doplňujících Byte pro BMP (aby byla velikost řádku dělitelná čtyřmi)

VH - velikost hlavičky a informační hlavičky BMP V3 (tzn. 54 Bytů)

VP - velikost palety (u obrázků s bitovou hloubkou větší než 8 je velikost palety 0)

ceil - funkce pro zaokrouhlení desetinného čísla vždy na vyšší hodnotu

Pro uložení exportovaných údajů se využívá formátu CSV s oddělením hodnot pomocí středníku (tzn. „;“) a oddělením řádků pomocí znaků s hodnotou v ASCII tabulce 0Dh a 0Ah (Aplikace Microsoft Excel načte i soubor bez znaku 0Dh). V programovém kódu se tyto znaky do textu zapíší jako „\r\n“.

Formát CSV byl využit díky jednoduchosti, dostatečným možnostem pro zvolený účel (včetně možnosti jednoduchého načtení v tabulkové aplikaci) a jen malým omezením - v textu nelze použít znaky „;“ „\r“ „\n“.

## 6 TESTOVÁNÍ KOMPRESNÍCH ALGORITMŮ

Testování bylo prováděno na zařízení s následujícími parametry:

- Intel® Core™ i5-430M 2,26 GHz (Turbo 2,53 GHz); 2 jádra; 4 vlákna; 3 MB Cache; 1066 MHz; 32 nm
- 4 GB DDR3 SDRAM; 1066 MHz
- HDD SATA 5400 ot/min; vyrovnávací paměť 8192 kB; doba přístupu 18,4 ms; průměrná přenosová rychlost 65 MB/s
- Windows® 7 Professional 64bit

Testované soubory dat byly rozděleny na 12 kategorií, které jsou hodnoceny odděleně. Výsledky testů jsou uvedeny v tabulce a grafu pro každou kategorii a jsou seřazeny vzestupně podle průměru hodnot kompresního poměru. Vysvětlivky pro zkratky v tabulkách jsou v *Tab.10*.

*Tab.10 Význam zkratek v tabulkách s výsledky*

<b>Zkratka</b>	<b>Význam</b>
Typ K	typ komprese
DD [ms]	doba dekomprese v ms
DK [ms]	doba komprese v ms
$KP_{\phi}$	průměr kompresních poměrů
$KP_{med}$	medián kompresních poměrů
$KP_{min}$	minimální kompresní poměr
$KP_{max}$	maximální kompresní poměr
$\sigma_{KP}$	směrodatná odchylka kompresních poměrů
ČEK [%/ms]	časová efektivita komprese

V rozšířené verzi práce je u každé kategorie vybrán průměrný testovaný obrázek, a to z hlediska průměru kompresních poměrů a průměru počtu pixelů testovaných obrázků.

Obrázky využitě v testech byly vyhledány na webech: [58] [59] [60] [61] [62] [63] [64] [65]. Celkově bylo pro testování využito 10 842 obrázků. Příloha P IV zobrazuje přehled nejlepších výsledků v testech.

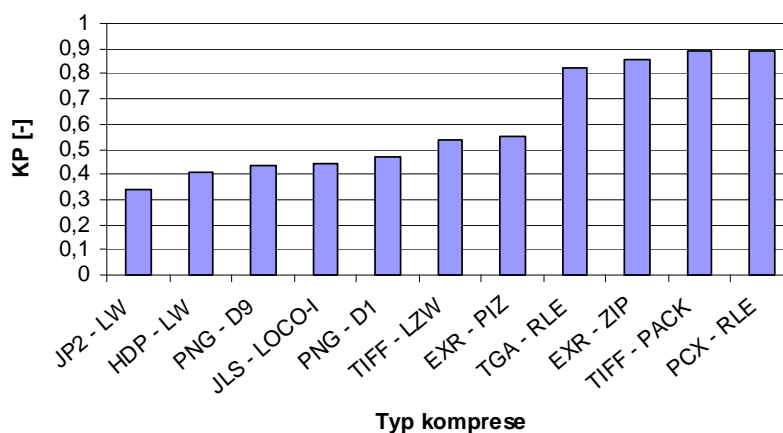
## 6.1 Fotografie

### 6.1.1 Fotografie formátu JPEG

V této kategorii jsou testovány fotografie, které byly původně uloženy ve formátu JPEG - před testem na ně tedy byla aplikována ztrátová komprese. Celkem bylo testováno 2839 fotografií a průměrný počet pixelů jedné fotografie se pohybuje okolo 1 893 965 pixelů. Výsledky testů jsou uvedeny v *Tab.11* a grafické zobrazení průměru kompresních poměrů je na *Obr.26*.

*Tab.11 Výsledky testů pro fotografie formátu JPEG*

Typ K	DD [ms]	DK [ms]	KP <sub>φ</sub>	KP <sub>med</sub>	KP <sub>min</sub>	KP <sub>max</sub>	σ <sub>KP</sub>	ČEK [%/ms]
JP2 - LW	1423	2009	0,342	0,354	0,037	0,826	0,139	0,033
HDP - LW	717	817	0,407	0,414	0,073	0,850	0,127	0,073
PNG - D9	131	3464	0,438	0,451	0,043	0,919	0,165	0,016
JLS - LOCO-I	453	383	0,444	0,458	0,030	0,927	0,159	0,145
PNG - D1	142	402	0,469	0,499	0,060	0,882	0,161	0,132
TIFF - LZW	85	147	0,534	0,549	0,078	1,213	0,194	0,317
EXR - PIZ	158	336	0,550	0,572	0,067	1,019	0,167	0,134
TGA - RLE	20	54	0,822	0,941	0,080	1,270	0,238	0,329
EXR - ZIP	249	2101	0,857	0,927	0,081	1,566	0,315	0,007
TIFF - PACK	21	58	0,888	1,006	0,082	1,329	0,252	0,194
PCX - RLE	125	228	0,889	0,990	0,064	1,640	0,274	0,049



*Obr.26 Graf výsledku testů pro fotografie formátu JPEG*

Na fotografie původně uložené ve formátu JPEG se nejlépe hodí použít formát JPEG 2000 s vlnkovou kompresí. Následující vlnková komprese formátu HD Photo dosahovala o asi 6 procent horší výsledky. JPEG 2000 exceluje nejenom z hlediska průměru kompresních poměrů, ale i mediánu, maximálního a minimálního kompresního poměru - u mediánu a maxima dosáhl nejlepších hodnot, u minima byl tento formát horší jen o 0,7 % než formát JPEG-LS. Jediným negativem JPEG 2000 je dlouhá doba komprese. Pokud je kladen důraz spíše na tento parametr a následně pak na kompresní poměr, tak je vhodnější použít HD Photo, popř. i JPEG-LS nebo PNG-DEFLATE\_1.

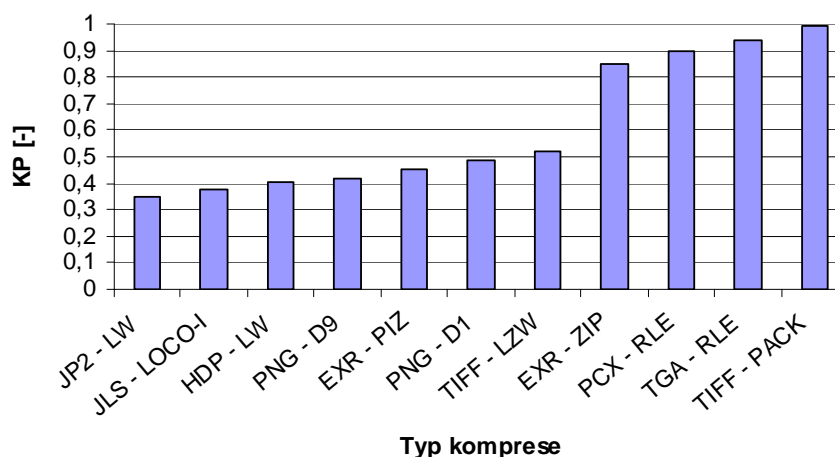
Pouze prvních 5 kompresí uvedených v *Tab.11* nedosáhly záporné komprese - konkrétně JPEG 2000, HD Photo, PNG-DEFLATE\_9, JPEG-LS a PNG-DEFLATE\_1. V porovnání s těmito formáty nejsou ostatní formáty vhodné pro ukládání fotografií původně uložených ve formátu JPEG.

### 6.1.2 Fotografie formátu RAW

Zdrojem testovaných subjektů jsou fotografie pořízené v nezkomprimovaném formátu příslušným typem fotoaparátu - konkrétně formáty CR2, NEF, ORF, RAF. Celkem bylo testováno 90 fotografií s průměrným počtem pixelů 9 083 009. Výsledky testů jsou uvedeny v *Tab.12* a grafické zobrazení průměru kompresních poměrů je na *Obr.27*.

*Tab.12 Výsledky testů pro fotografie formátů RAW*

Typ K	DD [ms]	DK [ms]	KP <sub>φ</sub>	KP <sub>med</sub>	KP <sub>min</sub>	KP <sub>max</sub>	σ <sub>KP</sub>	ČEK [%/ms]
JP2 - LW	5383	8492	0,347	0,317	0,207	0,598	0,095	0,008
JLS - LOCO-I	2067	1745	0,375	0,358	0,190	0,736	0,118	0,036
HDP - LW	3437	3881	0,404	0,380	0,303	0,634	0,081	0,015
PNG - D9	678	34073	0,419	0,395	0,243	0,773	0,111	0,002
EXR - PIZ	743	1670	0,455	0,427	0,257	0,812	0,120	0,033
PNG - D1	723	1933	0,484	0,461	0,314	0,784	0,105	0,027
TIFF - LZW	362	661	0,523	0,491	0,267	0,967	0,158	0,072
EXR - ZIP	1318	13775	0,847	0,814	0,542	1,300	0,188	0,001
PCX - RLE	619	1017	0,894	0,909	0,589	1,137	0,107	0,010
TGA - RLE	77	278	0,942	0,957	0,746	1,002	0,060	0,021
TIFF - PACK	96	286	0,994	1,005	0,826	1,009	0,030	0,002



Obr.27 Graf výsledku testů pro fotografie formátu RAW

Výsledky těchto testů se převážně u nejlepších typů kompresí velmi podobají výsledkům testů fotografií JPEG. A to jak v pořadí, tak v hodnotách kompresních poměrů. Výrazné zlepšení kompresních poměrů lze pozorovat u formátů JPEG-LS (asi 7 %) a EXR-PIZ (asi 10 %) a výrazné zhoršení u formátů TGA (asi 12 %) a TIFF-PACKBITS (asi 11 %).

Formáty, u kterých nedošlo k záporné kompresi jsou JPEG 2000, JPEG-LS, HD Photo, PNG-DEFLATE\_9, EXR-PIZ, PNG-DEFLATE\_1 a TIFF-LZW. Záporná komprese se u tohoto typu fotografií již neobjevuje v takové míře jako u fotografií JPEG.

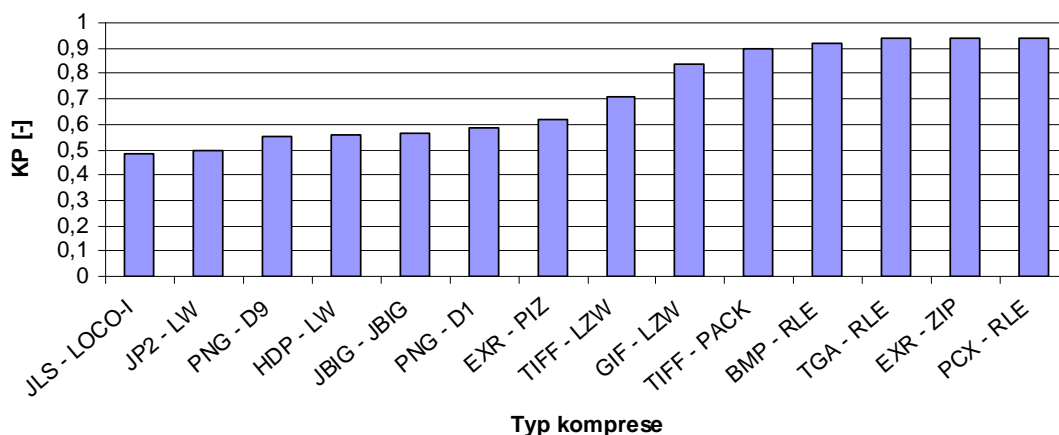
Jako nejlepší se jeví použití formátu JPEG 2000 s vlnkovou kompresí, který dosáhl nejlepšího průměru kompresních poměrů, mediánu a maximálního kompresního poměru. U minimálního kompresního poměru byl lepší o 1 % formát JPEG-LS. Pokud by byla na prvním místě potřeba využití rychlé komprese, je vhodnější použít formát JPEG-LS, který komprimuje obrázky asi 5-krát rychleji.

### 6.1.3 Fotografie v odstínech šedi

Tato sekce se zabývá fotografiemi v odstínech šedi. Celkem bylo testováno 665 fotografií s průměrným počtem pixelů 1 855 096. Výsledky testů jsou uvedeny v Tab.13 a grafické zobrazení průměru kompresních poměrů je na Obr.28.

Tab.13 Výsledky testů pro fotografie v odstínech šedi

Typ K	DD [ms]	DK [ms]	KP <sub>φ</sub>	KP <sub>med</sub>	KP <sub>min</sub>	KP <sub>max</sub>	σ <sub>KP</sub>	ČEK [%/ms]
JLS - LOCO-I	125	110	0,483	0,485	0,007	0,888	0,139	0,469
JP2 - LW	426	683	0,496	0,493	0,008	0,898	0,139	0,074
PNG - D9	38	1170	0,552	0,562	0,004	0,906	0,139	0,038
HDP - LW	257	283	0,558	0,552	0,134	0,920	0,125	0,156
JBIG - JBIG	620	599	0,566	0,581	0,006	0,985	0,159	0,072
PNG - D1	41	152	0,586	0,597	0,011	0,913	0,131	0,272
EXR - PIZ	75	140	0,621	0,630	0,066	0,985	0,151	0,271
TIFF - LZW	29	69	0,711	0,714	0,029	1,267	0,196	0,420
GIF - LZW	158	218	0,837	0,854	0,026	1,306	0,195	0,075
TIFF - PACK	10	33	0,898	0,960	0,025	1,008	0,141	0,310
BMP - RLE	23	18	0,919	0,988	0,020	1,023	0,137	0,450
TGA - RLE	17	50	0,937	1,004	0,026	1,077	0,150	0,127
EXR - ZIP	78	532	0,938	0,959	0,023	1,539	0,252	0,012
PCX - RLE	42	16	0,938	0,972	0,038	1,750	0,163	0,377



Obr.28 Graf výsledku testů pro fotografie v odstínech šedi

Nejlepší kompresí pro fotografie v odstínech šedi je komprese LOCO-I formátu JPEG-LS. Nejlepších výsledků dosáhl ve všech parametrech kromě minimálního kompresního poměru, kde však dosáhl nižší hodnoty jen o 0,3 % než komprese PNG-DEFLATE\_9. Nejlepší volba z hlediska rychlosti komprese je rovněž formát JPEG-LS.

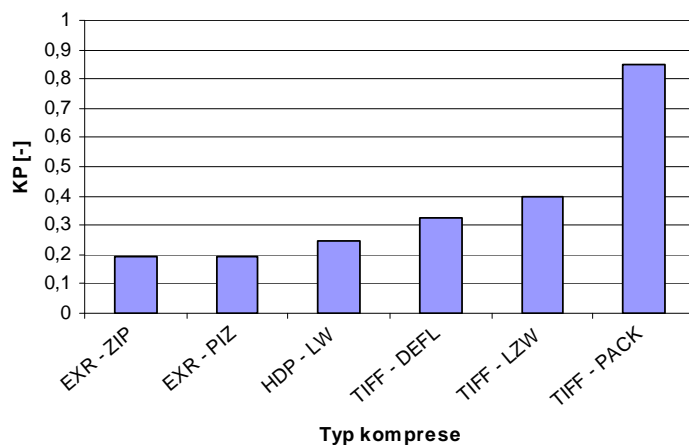
Formáty, u kterých nedošlo k záporné kompresi jsou JPEG-LS, JPEG 2000, PNG-DEFLATE\_9, HD Photo, JBIG, PNG-DEFLATE\_1 a EXR-PIZ.

### 6.1.4 Fotografie s vysokou dynamikou barev

V této kategorii byly testovány fotografie a formáty podporující barevný prostor RGBF, resp. RGBAF (u formátu HD Photo jediná možná varianta) - tedy 96 nebo 128 bitovou hloubku. Celkem bylo testováno 66 fotografií s průměrným počtem pixelů 815 251. Výsledky testů jsou uvedeny v *Tab.14* a grafické zobrazení průměru kompresních poměrů je na *Obr.29*.

*Tab.14* Výsledky testů pro fotografie s vysokou dynamikou barev

Typ K	DD [ms]	DK [ms]	KP <sub>φ</sub>	KP <sub>med</sub>	KP <sub>min</sub>	KP <sub>max</sub>	σ <sub>KP</sub>	ČEK [%/ms]
EXR - ZIP	105	566	0,191	0,186	0,003	0,398	0,109	0,143
EXR - PIZ	94	177	0,194	0,174	0,011	0,398	0,096	0,457
HDP - LW	371	397	0,250	0,252	0,071	0,413	0,082	0,189
TIFF - DEFL	100	1134	0,328	0,319	0,008	0,588	0,172	0,059
TIFF - LZW	104	224	0,400	0,394	0,070	0,675	0,177	0,268
TIFF - PACK	24	99	0,849	0,973	0,317	1,020	0,199	0,152



*Obr.29* Graf výsledku testů pro fotografie s vysokou dynamikou barev

Z výsledků není možné určit jednoznačného vítěze. Kompresní algoritmy ZIP a PIZ formátu EXR si jsou kompresními poměry velmi podobné. S přihlédnutím k rychlosti komprese je lepší použít formát EXR s kompresí PIZ.

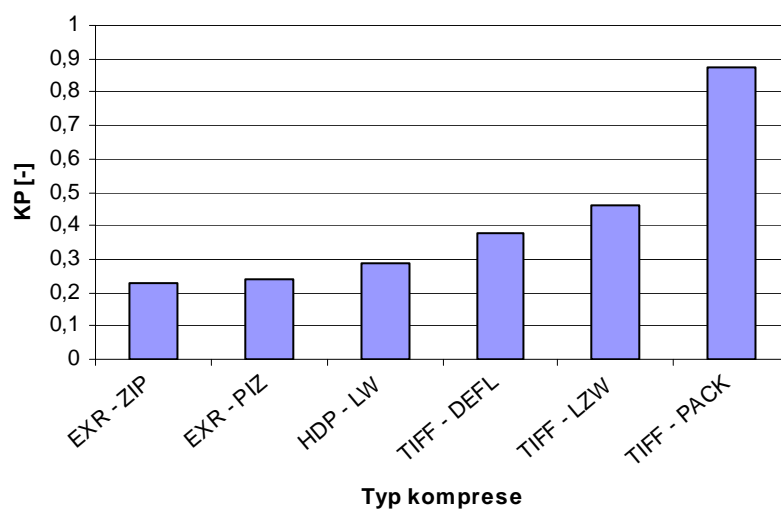
Vzhledem k možnosti dosažení záporné komprese kompresního algoritmu PACKBITS formátu TIFF a také vzhledem ke špatnému kompresnímu poměru není vhodné tuto kompresi používat - všechny ostatní kompresní algoritmy jsou v tomto ohledu minimálně dvojnásobně lepší.

### 6.1.5 Fotografie s vysokou dynamikou barev v odstínech šedi

V této kategorii byly testovány fotografie a formáty podporující barevný prostor FLOAT (někdy též nazýván GRAY FLOAT) - tedy 32 bitovou hloubku. Celkem bylo testováno 72 fotografií s průměrným počtem pixelů 787 691. Výsledky testů jsou uvedeny v *Tab.15* a grafické zobrazení průměru kompresních poměrů je na *Obr.30*.

*Tab.15 Výsledky testů pro fotografie s vysokou dynamikou barev v odstínech šedi*

Typ K	DD [ms]	DK [ms]	KP <sub>φ</sub>	KP <sub>med</sub>	KP <sub>min</sub>	KP <sub>max</sub>	σ <sub>KP</sub>	ČEK [%/ms]
EXR - ZIP	32	153	0,227	0,251	0,007	0,429	0,110	0,505
EXR - PIZ	53	91	0,237	0,250	0,033	0,448	0,101	0,838
HDP - LW	129	139	0,288	0,298	0,101	0,401	0,075	0,512
TIFF - DEFL	37	298	0,379	0,425	0,014	0,589	0,165	0,208
TIFF - LZW	37	101	0,459	0,510	0,052	0,672	0,181	0,535
TIFF - PACK	9	61	0,874	1,008	0,337	1,023	0,191	0,206



*Obr.30 Graf výsledku testů pro fotografie s vysokou dynamikou barev v odstínech šedi*

Nejlepší kompresí pro tuto kategorii je komprese ZIP formátu EXR, kterou ale následuje komprese PIZ stejného formátu - rozdíl je u kompresního poměru pouze 1 %. Při nutnosti využití rychlé komprese je lepší použít kompresi PIZ.

Ze stejného důvodu, který byl zmíněn v kapitole 6.1.4 není vhodné využívat formátu TIFF s kompresí PACKBITS.

## 6.2 Obrázky

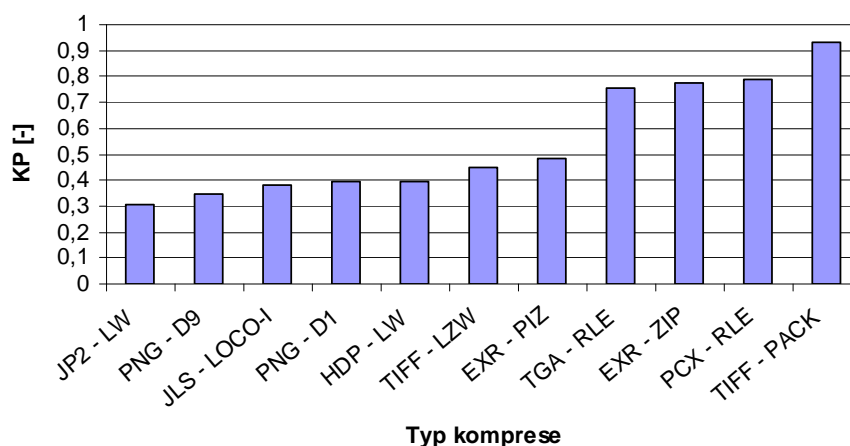
V této sekci byly testovány obrázky vytvořené pomocí PC (např. grafickým editorem).

### 6.2.1 Obrázky ve 24 bitové hloubce

Celkem bylo testováno 1 936 obrázků s průměrným počtem pixelů 1 336 500. Výsledky testů jsou uvedeny v *Tab.16* a grafické zobrazení průměru kompresních poměrů je na *Obr.31*.

*Tab.16 Výsledky testů pro obrázky ve 24 bitové hloubce*

Typ K	DD [ms]	DK [ms]	KP <sub>0</sub>	KP <sub>med</sub>	KP <sub>min</sub>	KP <sub>max</sub>	σ <sub>KP</sub>	ČEK [%/ms]
JP2 - LW	769	1257	0,308	0,284	0,001	0,898	0,122	0,055
PNG - D9	84	3649	0,350	0,333	0,002	0,921	0,162	0,018
JLS - LOCO-I	304	256	0,378	0,361	0,002	0,932	0,145	0,243
PNG - D1	91	251	0,395	0,392	0,006	0,885	0,155	0,241
HDP - LW	494	568	0,397	0,363	0,038	0,987	0,125	0,106
TIFF - LZW	59	108	0,451	0,433	0,022	1,515	0,188	0,509
EXR - PIZ	116	242	0,485	0,477	0,016	1,004	0,155	0,213
TGA - RLE	15	43	0,758	0,814	0,010	1,388	0,224	0,564
EXR - ZIP	174	1467	0,776	0,799	0,011	1,601	0,283	0,015
PCX - RLE	91	151	0,792	0,847	0,013	3,858	0,281	0,137
TIFF - PACK	16	52	0,931	1,003	0,021	1,469	0,182	0,133



Obr.31 Graf výsledku testů pro obrázky ve 24 bitové hloubce

Nejlepším formátem je pro tuto kategorii JPEG 2000, který z hlediska kompresního poměru porazil všechny formáty. Jen u maximálního kompresního poměru byla zaznamenána asi o 1 % lepší hodnota u komprese PNG-DEFLATE\_1. Z pohledu rychlosti komprese je možné využít formát JPEG-LS nebo PNG-DEFLATE\_1.

V Tab.16 a Tab.11 se objevuje jedna zajímavá zvláštnost, a to že komprese DEFLATE s úrovní 1 může u některých souborů dosáhnout lepšího kompresního poměru než komprese DEFLATE s úrovní 9 (viz hodnoty maximálních kompresních poměrů těchto dvou kompresí). Tam, kde komprese DEFLATE\_9 dosáhla výsledku 0,921 (tedy svého nejhoršího kompresního poměru v Tab.16), tam dosáhla komprese DEFLATE\_1 hodnoty 0,847. Zde tedy neplatí vždy pravidlo, čím větší zvolená úroveň tím lepší komprese.

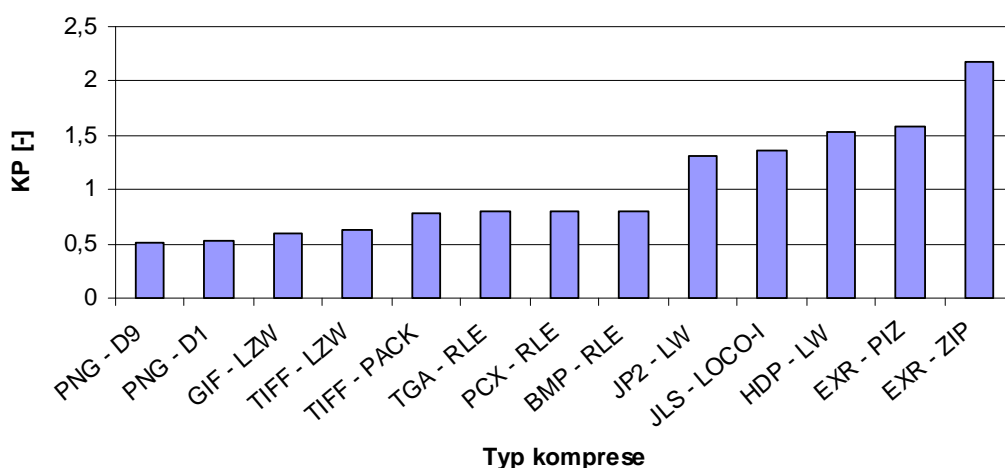
Formáty, u kterých nedošlo k záporným kompresím byly JPEG 2000, PNG-DEFLATE\_9, JPEG-LS, PNG-DEFLATE\_1 a HD Photo. Přestože se komprese TIFF-PACKBITS svým průměrem kompresních poměrů nejvíce blíží trendu záporné komprese, nejhorší maximální kompresní poměr dosáhl formát PCX s kompresí RLE (3,858).

### 6.2.2 Obrázky v 8 bitové hloubce

Celkem bylo testováno 1 512 obrázků s průměrným počtem pixelů 1 215 197. Výsledky testů jsou uvedeny v Tab.17 a grafické zobrazení průměru kompresních poměrů je na Obr.32.

Tab.17 Výsledky testů pro obrázky v 8 bitové hloubce

Typ K	DD [ms]	DK [ms]	KP <sub>φ</sub>	KP <sub>med</sub>	KP <sub>min</sub>	KP <sub>max</sub>	σ <sub>KP</sub>	ČEK [%/ms]
PNG - D9	14	318	0,503	0,531	0,007	0,946	0,212	0,156
PNG - D1	16	54	0,526	0,559	0,015	0,947	0,199	0,879
GIF - LZW	89	123	0,593	0,625	0,008	1,298	0,236	0,331
TIFF - LZW	17	44	0,624	0,646	0,028	1,606	0,236	0,860
TIFF - PACK	8	25	0,782	0,845	0,028	2,017	0,252	0,883
TGA - RLE	13	33	0,792	0,868	0,025	1,075	0,253	0,632
PCX - RLE	29	11	0,794	0,841	0,040	1,299	0,269	1,806
BMP - RLE	23	14	0,797	0,886	0,025	1,044	0,230	1,477
JP2 - LW	834	1370	1,312	1,276	0,014	2,709	0,528	-0,023
JLS - LOCO-I	247	201	1,352	1,339	0,031	2,756	0,573	-0,175
HDP - LW	458	520	1,536	1,570	0,124	2,748	0,468	-0,103
EXR - PIZ	103	209	1,577	1,600	0,069	2,846	0,557	-0,276
EXR - ZIP	150	1094	2,180	2,278	0,043	4,464	1,006	-0,108



Obr.32 Graf výsledku testů pro obrázky v 8 bitové hloubce

Nejlepší kompresí v této kategorii je PNG-DEFLATE\_9. Kompresní poměr vykazuje ve všech případech nejlepší hodnoty. Jen o asi 2 % horší je komprese PNG-DEFLATE\_1, která svou rychlostí komprese překonává PNG-DEFLATE\_9 téměř 6-krát.

Dvě výše zmíněné komprese byly jediné, které nedosáhly záporné komprese. Naopak u novějších typů formátů JPEG 2000, JPEG-LS, HD Photo, EXR-PIZ a EXR-ZIP se záporné komprese objevují nejčastěji. U těchto formátů k tomu však spíše dochází kvůli nutnosti

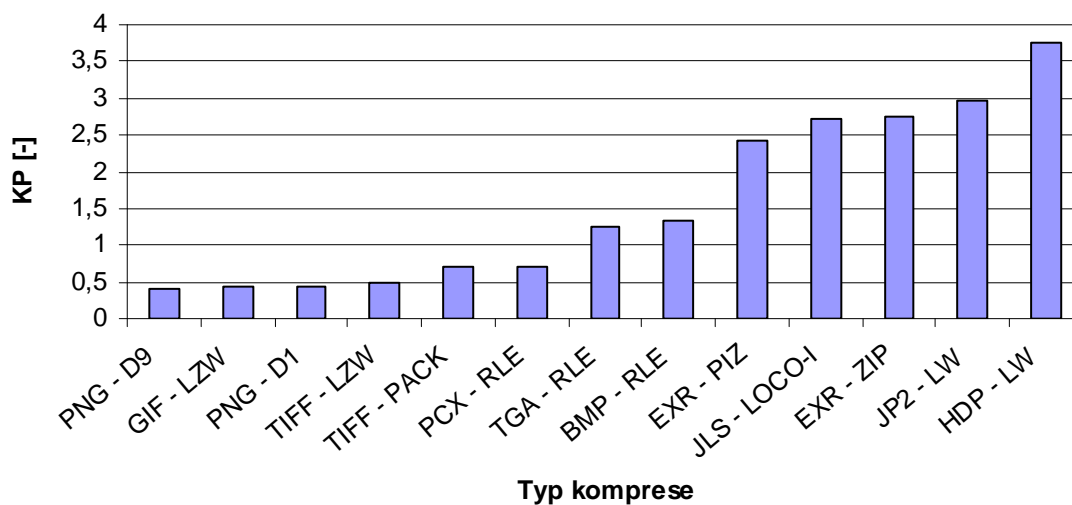
převodu na větší bitovou hloubku než kvůli špatnému kompresnímu algoritmu. Tyto formáty by neměly být využívány pro barevné obrázky v 8 bitové hloubce.

### 6.2.3 Obrázky ve 4 bitové hloubce

Celkem bylo testováno 635 obrázků s průměrným počtem pixelů 774 558. Výsledky testů jsou uvedeny v *Tab.18* a grafické zobrazení průměru kompresních poměrů je na *Obr.33*.

*Tab.18 Výsledky testů pro obrázky ve 4 bitové hloubce*

Typ K	DD [ms]	DK [ms]	KP <sub>φ</sub>	KP <sub>med</sub>	KP <sub>min</sub>	KP <sub>max</sub>	σ <sub>KP</sub>	ČEK [%/ms]
PNG - D9	5	293	0,407	0,416	0,013	1,153	0,179	0,202
GIF - LZW	27	33	0,431	0,450	0,027	1,015	0,178	1,734
PNG - D1	6	13	0,441	0,450	0,024	1,153	0,175	4,233
TIFF - LZW	5	23	0,487	0,502	0,044	2,831	0,247	2,234
TIFF - PACK	4	13	0,695	0,739	0,104	2,723	0,255	2,396
PCX - RLE	10	4	0,700	0,735	0,109	1,226	0,255	8,221
TGA - RLE	8	19	1,240	1,310	0,105	6,385	0,537	-1,257
BMP - RLE	20	11	1,322	1,395	0,106	8,538	0,640	-2,985
EXR - PIZ	65	128	2,409	2,608	0,233	6,550	0,852	-1,103
JLS - LOCO-I	114	91	2,722	2,881	0,126	5,498	1,037	-1,895
EXR - ZIP	86	448	2,760	2,717	0,086	5,811	1,330	-0,393
JP2 - LW	594	984	2,969	3,161	0,203	5,993	1,081	-0,200
HDP - LW	285	326	3,746	3,947	0,452	10,324	1,058	-0,843



*Obr.33 Graf výsledku testů pro obrázky ve 4 bitové hloubce*

V kategorii 4 bitových obrázků je z hlediska průměru kompresních poměrů, mediánu a minimálního kompresního poměru nejlepší komprese PNG-DEFLATE\_9. Problémy jsou však s vyšší hodnotou doby komprese, která výrazně převyšuje ostatní hodnoty v popředí tabulky.

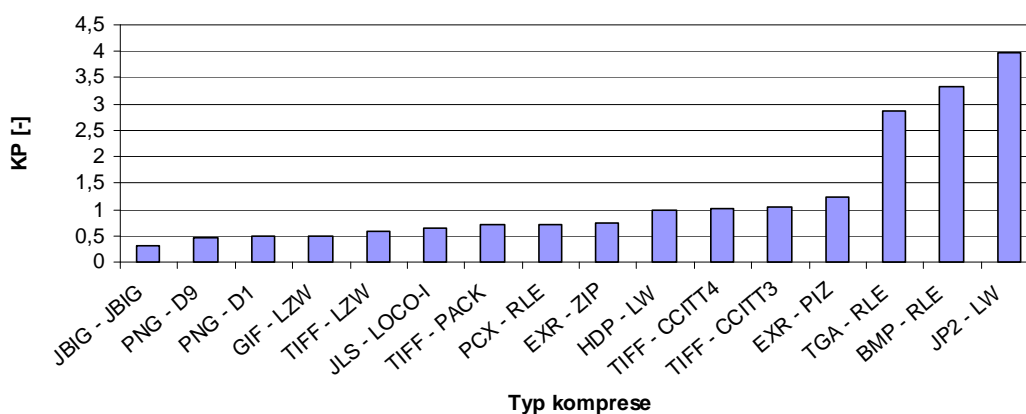
V této kategorii se nepodařilo najít typ komprese, která by někdy nedosahovala záporných hodnot. Nejlépe je v tomto ohledu formát GIF, dosáhl nejnižšího maximálního kompresního poměru. Pokud se k tomu připočte druhý nejlepší kompresní poměr a asi 9-krát rychlejší komprese než u formátu PNG-DEFLATE\_9, jedná se o nejlepší volbu právě v případě upřednostnění rychlosti komprese před nepatrně lepším kompresním poměrem. Další možnou volbou v tomto směru je i komprese PNG-DEFLATE\_1, která je ještě 3-krát rychlejší než GIF a horší jen asi o 1 % v průměru kompresních poměrů.

#### 6.2.4 Obrázky v 1 bitové hloubce

Celkem bylo testováno 1915 obrázků s průměrným počtem pixelů 940 427. Výsledky testů jsou uvedeny v *Tab.19* a grafické zobrazení průměru kompresních poměrů je na *Obr.34*.

*Tab.19 Výsledky testů pro obrázky v 1 bitové hloubce*

Typ K	DD [ms]	DK [ms]	KP <sub>φ</sub>	KP <sub>med</sub>	KP <sub>min</sub>	KP <sub>max</sub>	σ <sub>KP</sub>	ČEK [%/ms]
JBIG - JBIG	29	32	0,317	0,299	0,003	0,985	0,226	2,164
PNG - D9	3	77	0,452	0,435	0,008	1,945	0,281	0,710
PNG - D1	3	5	0,482	0,471	0,012	1,945	0,269	10,821
GIF - LZW	21	27	0,498	0,486	0,025	1,176	0,275	1,876
TIFF - LZW	3	19	0,583	0,521	0,043	3,405	0,402	2,246
JLS - LOCO-I	21	20	0,653	0,543	0,015	1,741	0,456	1,760
TIFF - PACK	2	10	0,705	0,675	0,051	3,243	0,372	2,851
PCX - RLE	3	2	0,717	0,721	0,058	1,811	0,324	18,214
EXR - ZIP	34	100	0,745	0,712	0,046	4,284	0,479	0,255
HDP - LW	86	97	0,978	0,881	0,031	3,063	0,595	0,023
TIFF - CCITT4	7	22	1,011	0,675	0,006	3,529	0,929	-0,051
TIFF - CCITT3	6	20	1,038	0,770	0,018	3,486	0,899	-0,189
EXR - PIZ	42	72	1,230	1,069	0,122	6,892	0,911	-0,320
TGA - RLE	9	21	2,881	2,024	0,137	11,216	2,507	-8,991
BMP - RLE	21	10	3,340	3,015	0,083	15,000	2,419	-23,288
JP2 - LW	266	444	3,985	3,592	0,090	7,998	2,471	-0,673



Obr.34 Graf výsledku testů pro obrázky v 1 bitové hloubce

V této kategorii je jasně nejlepším řešením použití komprese JBIG. Je to jediný případ, jehož kompresní poměr byl při testech vždy pod hodnotou 1. I když z pohledu rychlosti komprese představuje JBIG spíše průměr, příliš se nevyplatí upřednostňovat kompresi PNG-DEFLATE\_1. Ta má sice asi 6-krát rychlejší kompresní algoritmus, ovšem kompresní poměr se pohybuje v hodnotách horších asi o 17 %, což nelze zanedbat.

Milým překvapením jsou formáty JPEG-LS a EXR-ZIP, které i přes nutnost zvýšení bitové hloubky dosáhly v průměru nezáporné komprese. JPEG-LS měla dokonce třetí nejlepší maximální hodnotu kompresního poměru.

Naopak komprese RLE ve formátech TGA a BMP i vlnkové transformace ve formátech JPEG 2000 a EXR dokazují, že při použití těchto typů kompresí si není možné dovolit zvyšování bitové hloubky.

Nemilým překvapením je komprese TIFF-CCITT3 a TIFF-CCITT4, které dosahují v průměru záporné komprese. Je ale nutné přihlédnout k tomu, že tyto komprese byly vytvořeny převážně pro textový obsah v obrázku a navíc pro medián kompresních poměrů dosahují výrazně lepších hodnot.

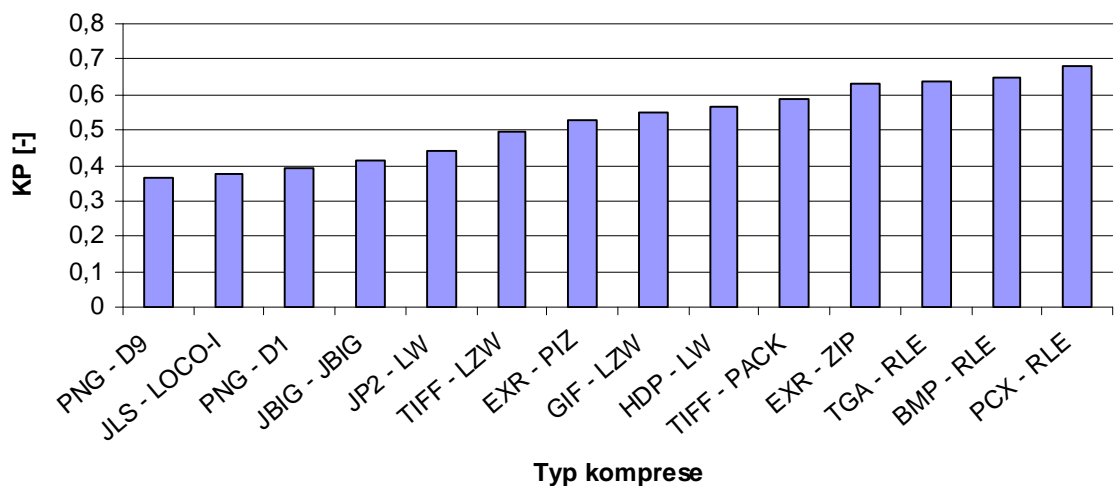
U obrázků, kterým bylo sníženo počet barev pomocí Floyd a Steinbergovi metody bylo dosaženo horších výsledků kompresních poměrů než u obrázků původně vytvořených ve dvou barvách.

### 6.2.5 Obrázky v odstínech šedi

Celkem bylo testováno 436 obrázků s průměrným počtem pixelů 3 614 476. Výsledky testů jsou uvedeny v *Tab.20* a grafické zobrazení průměru kompresních poměrů je na *Obr.35*.

*Tab.20 Výsledky testů pro obrázky v odstínech šedi*

Typ K	DD [ms]	DK [ms]	KP <sub>φ</sub>	KP <sub>med</sub>	KP <sub>min</sub>	KP <sub>max</sub>	σ <sub>KP</sub>	ČEK [%/ms]
PNG - D9	59	1441	0,367	0,249	0,004	0,932	0,254	0,044
JLS - LOCO-I	167	148	0,376	0,356	0,014	0,929	0,220	0,421
PNG - D1	61	223	0,392	0,272	0,009	0,935	0,255	0,272
JBIG - JBIG	971	915	0,413	0,348	0,006	1,036	0,265	0,064
JP2 - LW	787	1291	0,440	0,447	0,014	0,957	0,211	0,043
TIFF - LZW	46	96	0,495	0,352	0,035	1,330	0,347	0,526
EXR - PIZ	117	221	0,529	0,579	0,039	1,114	0,248	0,213
GIF - LZW	195	283	0,548	0,439	0,016	1,332	0,372	0,160
HDP - LW	446	498	0,565	0,580	0,115	0,981	0,198	0,087
TIFF - PACK	14	40	0,588	0,489	0,032	1,008	0,354	1,030
EXR - ZIP	138	755	0,633	0,469	0,023	1,654	0,419	0,049
TGA - RLE	27	83	0,634	0,611	0,032	1,071	0,361	0,439
BMP - RLE	33	25	0,648	0,690	0,027	1,125	0,345	1,401
PCX - RLE	72	25	0,682	0,580	0,047	1,816	0,405	1,276



*Obr.35 Graf výsledku testů pro obrázky v odstínech šedi*

Nejlepší kompresí pro obrázky v odstínech šedi je PNG-DEFLATE a JPEG-LS. Pro kompresi DEFLATE\_9 a DEFLATE\_1 hovoří také medián kompresních poměrů, které dosahují nejlepších výsledků. Na druhou stranu komprese JPEG-LS je rychlejší a dosáhla nejlepšího maximálního kompresního poměru. Navíc nejsou v průměru kompresních poměrů výrazné rozdíly.

Záporných kompresí nedosahovaly formáty PNG-DEFLATE\_9, JPEG-LS, PNG-DEFLATE\_1, JPEG 2000 a HD Photo. Poslední jmenovaný formát však kvůli horšímu průměru kompresních poměrů není pro tento typ obrázků doporučen.

## 6.3 Text

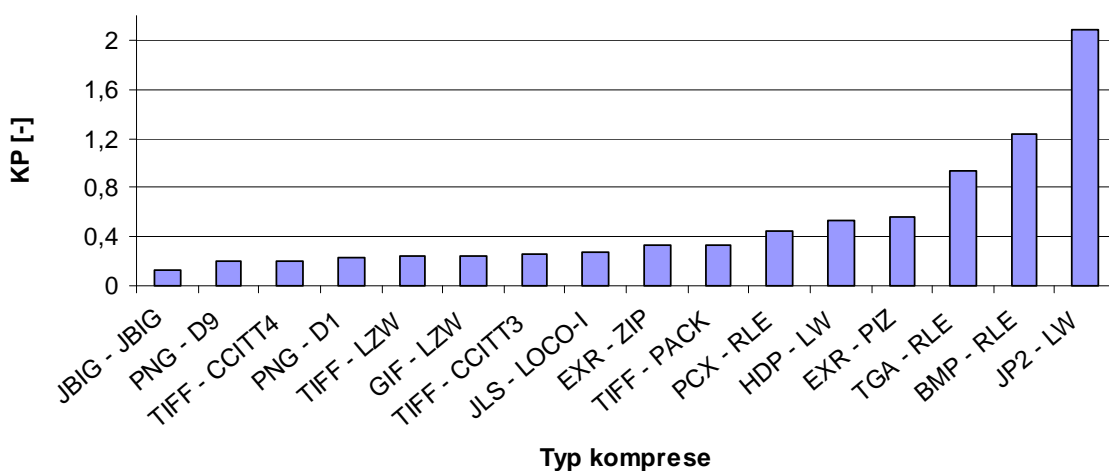
Obrázky pro tyto testy obsahovaly textovou informaci, někdy i v kombinaci s obrázkem.

### 6.3.1 Text v 1 bitové hloubce

Celkem bylo testováno 357 obrázků s průměrným počtem pixelů 2 861 160. Výsledky testů jsou uvedeny v *Tab.21* a grafické zobrazení průměru kompresních poměrů je na *Obr.36*.

*Tab.21 Výsledky testů pro obrázky s textem v 1 bitové hloubce*

Typ K	DD [ms]	DK [ms]	KP <sub>φ</sub>	KP <sub>med</sub>	KP <sub>min</sub>	KP <sub>max</sub>	σ <sub>KP</sub>	ČEK [%/ms]
JBIG - JBIG	58	67	0,126	0,129	0,017	0,552	0,054	1,306
PNG - D9	5	147	0,197	0,199	0,032	0,627	0,072	0,545
TIFF - CCITT4	7	20	0,198	0,189	0,024	1,219	0,112	4,024
PNG - D1	4	9	0,228	0,228	0,041	0,662	0,076	8,945
TIFF - LZW	4	21	0,247	0,247	0,057	0,792	0,085	3,549
GIF - LZW	45	63	0,249	0,250	0,044	0,723	0,084	1,198
TIFF - CCITT3	7	19	0,257	0,257	0,052	1,324	0,123	3,864
JLS - LOCO-I	25	23	0,277	0,277	0,043	0,933	0,104	3,203
EXR - ZIP	94	211	0,328	0,327	0,082	1,002	0,110	0,319
TIFF - PACK	2	11	0,334	0,344	0,076	0,740	0,103	5,919
PCX - RLE	6	2	0,440	0,454	0,097	0,946	0,139	23,229
HDP - LW	220	248	0,538	0,541	0,085	1,707	0,199	0,186
EXR - PIZ	103	177	0,560	0,562	0,182	1,486	0,173	0,248
TGA - RLE	20	55	0,929	0,914	0,211	3,845	0,396	0,129
BMP - RLE	37	15	1,232	1,251	0,166	4,167	0,525	-1,573
JP2 - LW	617	1080	2,083	2,103	0,276	6,846	0,776	-0,100



Obr.36 Graf výsledku testů pro obrázky s textem v 1 bitové hloubce

Pro tuto kategorii je nejlepší využít kompresi JBIG. Tato komprese dosahuje nejlepšího průměru, mediánu, minimálního i maximálního kompresního poměru. Při požadavku na rychlost komprese je vhodnější využití komprese TIFF-CCITT4, která je rychlejší více než 3-krát. Na druhou stranu JBIG nedosahuje záporných kompresí a kompresní poměr je lepší asi o 7 %. Horší komprese však dosahovaly formáty u subjektů s textem a obrázkem - u čistého textu k záporným kompresím u TIFF-CCITT4, TIFF-CCITT3 apod. nedocházelo.

Další možnou variantou je při preferování rychlosti komprese formát PNG-DEFLATE\_1, který je ještě 2-krát rychlejší než TIFF-CCITT4 a nedosáhl záporné komprese. Průměrná hodnota kompresních poměrů je o asi 3 % horší než u TIFF-CCITT4.

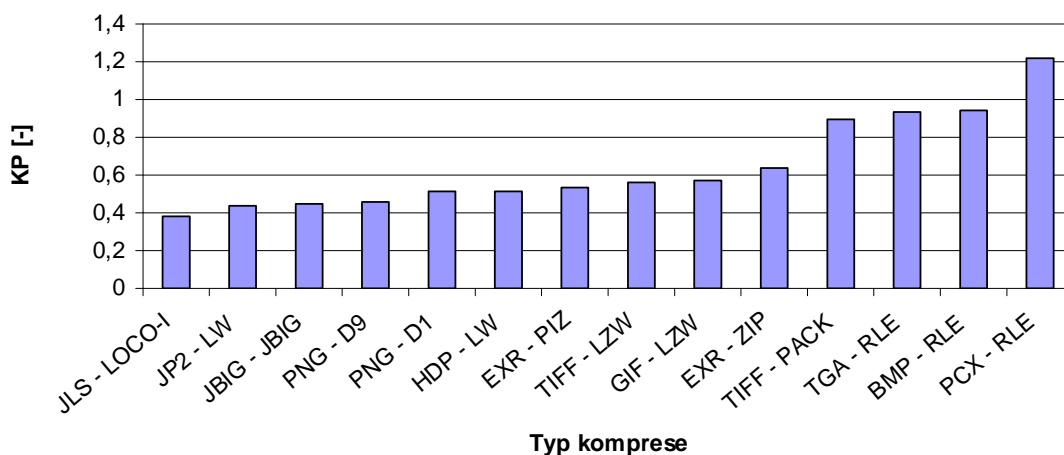
Formáty, u kterých se neprojevila záporná komprese jsou JBIG, PNG-DEFLATE\_9, PNG-DEFLATE\_1, TIFF-LZW, GIF-LZW, JPEG-LS, TIFF-PACKBITS, PCX. Kvůli horším hodnotám průměru kompresních poměrů formátů JPEG-LS, TIF-PACKBITS a PCX však není doporučeno příliš tyto formáty využívat. Naopak jak již bylo zmíněno výše, lepší variantou je komprese TIFF-CCITT4, a to i přesto, že záporné komprese dosáhla.

### 6.3.2 Text v odstínech šedi

Celkem bylo testováno 319 obrázků s průměrným počtem pixelů 2 854 891. Výsledky testů jsou uvedeny v Tab.22 a grafické zobrazení průměru kompresních poměrů je na Obr.37.

Tab.22 Výsledky testů pro obrázky s textem v odstínech šedi

Typ K	DD [ms]	DK [ms]	KP <sub>φ</sub>	KP <sub>med</sub>	KP <sub>min</sub>	KP <sub>max</sub>	σ <sub>KP</sub>	ČEK [%/ms]
JLS - LOCO-I	212	190	0,377	0,365	0,140	0,623	0,081	0,328
JP2 - LW	783	1269	0,433	0,429	0,176	0,622	0,078	0,045
JBIG - JBIG	865	833	0,449	0,442	0,146	0,716	0,092	0,066
PNG - D9	67	3034	0,459	0,448	0,173	0,697	0,085	0,018
PNG - D1	72	249	0,512	0,506	0,200	0,718	0,076	0,196
HDP - LW	405	444	0,514	0,512	0,281	0,702	0,070	0,109
EXR - PIZ	134	232	0,534	0,537	0,220	0,735	0,085	0,201
TIFF - LZW	47	90	0,561	0,552	0,205	0,934	0,121	0,486
GIF - LZW	206	304	0,567	0,555	0,171	0,931	0,123	0,142
EXR - ZIP	115	672	0,638	0,626	0,283	1,027	0,139	0,054
TIFF - PACK	17	39	0,891	0,881	0,284	1,005	0,074	0,277
TGA - RLE	27	77	0,930	0,916	0,298	1,060	0,081	0,091
BMP - RLE	46	27	0,945	0,945	0,303	1,007	0,061	0,204
PCX - RLE	70	28	1,219	1,155	0,398	1,751	0,199	-0,769



Obr.37 Graf výsledku testů pro obrázky s textem v odstínech šedi

V tomto testu výrazně převyšuje komprese LOCO-I formátu JPEG-LS všechny kompresní algoritmy, a to i z hlediska rychlosti komprese. U maximálního kompresního poměru byl sice lepší formát JPEG 2000, ovšem jen o 0,1 %, což je velmi nepatrný rozdíl.

Záporných kompresí dosáhly formáty EXR-ZIP, TIFF-PACKBITS, TGA, BMP a PCX a i z pohledu na hodnoty průměrů kompresních poměrů těchto formátů je zřejmé, že pro obrázky testované v této kategorii není jejich použití vhodné.

## ZÁVĚR

Cílem této diplomové práce bylo vytvořit literární rešerši na téma neztrátové kompresní algoritmy v rastrové počítačové grafice. Dalším bodem bylo seznámit se s formáty, které tyto algoritmy využívají a popsat jejich strukturu. Následovalo naprogramování aplikace s implementací neztrátových algoritmů a popis této aplikace. Nakonec byly tyto algoritmy testovány a vyhodnoceny na dvanácti různých kolekcích obrázků.

Bylo zjištěno, že u fotografií a 24 bitových obrázků je nejlepší volbou využití formátu JPEG 2000 s vlnkovou transformací. Tento formát disponuje výborným kompresním poměrem, jehož hodnoty jsou často nejlepší ze všech testovaných formátů. Jedinou chybou formátu JPEG 2000 je doba komprese, která v testech patří k jedné z nejdelších.

Pokud je dán požadavek na rychlou a zároveň kvalitní kompresi, tak je nejvhodnější použít kompresní algoritmus LOCO-I formátu JPEG-LS. I když tato komprese dosahuje u barevných fotografií a 24 bitových obrázků dobrých výsledků, kvalita této komprese se projevuje hlavně u obrázků v odstínech šedi - bez ohledu na to, zda se jedná o fotografie, obrázky vytvořené v počítači nebo obsahující text.

Přestože komprese JBIG podporuje obrázky v odstínech šedi, nejedná se o prioritu této komprese. Skvělý kompresní poměr dosahuje u obrázků v 1 bitové hloubce - bez ohledu na to, zda jde čistě o obrázky, o obrázky s textem nebo něco mezi.

Výsledky komprese CCITT4 a CCITT3 potvrdily, že jejich primárním využitím jsou 1 bitové obrázky s textem, u kterých obzvlášť komprese CCITT4 dosahovala velmi dobrých hodnot. Pokud se však v obrázku nevyskytuje text, tak často dochází k záporné kompresi. Rychlost komprese je u těchto dvou algoritmů totožná.

U 8 a 4 bitových obrázků je nejlepší možností využití komprese DEFLATE formátu PNG. V úrovních komprese DEFLATE není příliš velký rozdíl, a je tedy na volbě uživatele jakou vybere. Formát PNG je nejuniverzálnějším formátem - ve všech testovaných kategoriích patří kompresní algoritmus tohoto formátu k jedněm z nejlepších z hlediska kompresního poměru, a při snížení úrovně komprese i z hlediska rychlosti komprese.

U fotografií s vysokou dynamikou barev ukazuje svou sílu formát EXR s kompresí PIZ a ZIP. Z pohledu kompresního poměru není mezi těmito dvěma typy komprese větší rozdíl. S přihlédnutím k rychlosti komprese je však PIZ lepší volbou.

Komprese LZW u formátů GIF a TIFF je vhodné použít jen u obrázků s bitovou hloubkou nižší než 8 bitů. U ostatních obrázků dosahuje komprese LZW spíše průměrných hodnot. Obecně se dá říct, že je lepší upřednostňovat kompresi LZW formátu GIF, která dokáže o něco více snížit velikost souboru než LZW formátu TIFF.

Převážně komprese formátu JPEG 2000, ale i HD Photo, EXR a JPEG-LS se nehodí při použití na barevné obrázky s bitovou hloubkou 8 a nižší.

Komprese RLE formátů PCX, TGA, BMP a komprese PACKBITS formátu TIFF dosahují z dnešního pohledu špatných kompresních poměrů ve všech kategoriích a nevyplatí se je tedy příliš využívat.

## ZÁVĚR V ANGLIČTINĚ

The aim of this diploma thesis was to create survey describing about lossless compression algorithms in raster computer graphics. The next item was to become familiar with formats, that use these algorithms and to describe structure of these formats. It is followed by programming of an application with lossless algorithms implementation and by description of the application. At the end, these algorithms were tested and evaluated on twelve different image collections.

It was found, the best choice for photographs and 24 bit depth pictures is JPEG 2000 format with wavelet transformation. This format has an excellent compression ratio, whose values are often the best of all tested formats. The only flaw is the JPEG 2000 compression time, which in tests belong to one of the longest.

If the request is made for both quality and fast compression, so it is the best to use a LOCO-I compression of JPEG-LS format. Although this compression reaches good results for photographs and 24 bit depth images, quality of this compression is manifested mainly in greyscale images - irrespective of whether they are photographs, images created in a computer or images containing the text.

Although JBIG compression supports greyscale images, these are not its priority. The excellent compression ratio is reached with 1 bit depth images - irrespective of whether it is purely images, the images with text or something between these two choices.

CCITT4 and CCITT3 compression results confirmed, that their primary use are 1 bit images with text. Specially CCITT4 compression make very good progress. However, if the text is not in the image, it often produces negative compression. Compression times are identical for these two algorithms.

For 8 and 4 bit images is the best possible to use DEFLATE compression of PNG format. The DEFLATE compression levels is not much difference, so a user themself can make decision about choosing the compression level. PNG is the most universal format of all - the compression algorithm of this format belongs to one of the best in view of compression ratio and of compression time (valid only for small compression level) in all tested categories.

EXR format including PIZ and ZIP compressions shows its strength for photos with high dynamic colours. In view of compression ratio, between these two types of compression are not a big differences. The PIZ is a little bit better than ZIP because of quicker compression.

LZW compression in GIF and TIFF formats should be used only for images with the bit depth less than 8 bits. LZW compression reaches average values of compression ratio for other types of images. Generally we can say, that it is better to prefer GIF-LZW compression, which can reduce the file size more than TIFF-LZW.

Mostly JPEG 2000, as well as HD Photo, EXR, JPEG-LS is not suitable for use on colour images with the bit depth of 8 or less.

RLE compression of PCX, TGA, BMP and PACKBITS compression of TIFF achieve poor compression ratio in all categories and therefore they are not recommended to use.

**SEZNAM POUŽITÉ LITERATURY**

- [1] ŽÁRA, Jiří, Bedřich BENEŠ, Jiří SOCHOR a Petr FELKEL. *Moderní počítačová grafika: Kompletní průvodce metodami 2D a 3D grafiky*. 2. vydání. Brno: Computer Press, 2004. ISBN 80-251-0454-0.
- [2] STRACHOTA, Pavel. *Ukládání a komprese obrazu* [online]. Praha: FJFI ČVUT, 2010, 15.9.2010 [cit. 2012-01-08]. Dostupné z: [http://saint-paul.fjfi.cvut.cz/base/public-filesystem/admin-upload/POGR/POGR1/07.ukladani\\_a\\_kompresse\\_obrazu.pdf](http://saint-paul.fjfi.cvut.cz/base/public-filesystem/admin-upload/POGR/POGR1/07.ukladani_a_kompresse_obrazu.pdf)
- [3] MURRAY, James D. a William VANRYPER. *Encyklopedie grafických formátů*. 2. vydání. Praha: Computer Press, 1997. ISBN 80-7226-033-2.
- [4] PELIKÁN, Josef. *Kódování rastrových obrázků* [online]. Praha: CGG MFF UK, 2011, 16 s. [cit. 2012-03-23]. Dostupné z: <http://cgg.mff.cuni.cz/~pepca/lectures/pdf/pg1-19-imagecoding.pdf>
- [5] MORKEŠ, David. *Komprimační a archivační programy*. Praha: Computer Press, 1998. ISBN 80-7226-089-8.
- [6] VEČERKA, Arnošt. *Kompresse dat* [online]. Olomouc: Univerzita Palackého, 2008, 30.4.2008 [cit. 2011-12-18]. Dostupné z: <http://phoenix.inf.upol.cz/esf/ucebni/kompresse.pdf>
- [7] GIMLI. *Kompresse a kompresní algoritmy. Gimliho stránky* [online]. [2006-2010] [cit. 2012-01-09]. Dostupné z: <http://gimli.mysteria.cz/kompresse/algoritmy.html>
- [8] SALOMON, David. *Data Compression: the complete reference*. 4th edition. London: Springer, 2007, 1092 s. ISBN 1-84628-602-6.
- [9] MATOUŠEK, Radomil. *Metody kódování* [online]. verze 1.9. Brno: VUT, 2006, 96 s. [cit. 2012-03-26]. Dostupné z: <http://www.uai.fme.vutbr.cz/~matousek/TIK/TIKv9.swf>
- [10] KUHN, Markus. *JBIG1 patent information. The Computer Laboratory: Faculty of Computer Science and Technology* [online]. 26.2.2011 [cit. 2012-04-30]. Dostupné z: <http://www.cl.cam.ac.uk/~mgk25/jbigkit/patents/>

- [11] HATLAPATKA, Radim. *JBIG2 komprese* [online]. Brno, 2009 [cit. 2012-03-11]. Dostupné z: <http://www.fi.muni.cz/~xhatlap/bc/bakalarka.pdf>. Bakalářská práce. Masarykova univerzita, Fakulta informatiky. Vedoucí práce doc. RNDr. Petr Sojka, Ph.D.
- [12] JBIG2 Development. JOINT PHOTOGRAPHIC EXPERTS GROUP. *The JPEG committee home page* [online]. 2007 [cit. 2012-03-11]. Dostupné z: <http://www.jpeg.org/jbig/jbigpt2.html>
- [13] ŠMÍD, Radislav. *Úvod do vlnkové transformace* [online]. Praha: ČVUT FEL, 9.8.2001, 9 s. [cit. 2012-03-25]. Dostupné z: <http://measure.feld.cvut.cz/usr/staff/smid/wavelets/Wavelet-intro8859.pdf>
- [14] VRÁNA, Jaroslav, Jan ČERMÁK a Radek ZEZULA. Výpočet vlnkové transformace pomocí algoritmu "lifting". *Elektrorevue* [online]. Ústav telekomunikací FEKT, 8.6.2004 [cit. 2012-03-25]. Dostupné z: <http://www.elektrorevue.cz/clanky/04034/index.html>
- [15] VRÁNA, Jaroslav. Využití adaptivního liftingu při kompresi dat. *Elektrorevue* [online]. Ústav telekomunikací FEKT, 10.2.2006 [cit. 2012-03-25]. Dostupné z: <http://www.elektrorevue.cz/clanky/06010/index.html>
- [16] *DZS - cvičení č. 6 - DPCM, DCT a její použití při kompresi obrazové informace* [online]. 23.5.2007, 5 s. [cit. 2012-03-24]. Dostupné z: [www.comtel.cz/files/download.php?id=3119](http://www.comtel.cz/files/download.php?id=3119)
- [17] BALÍK, Miroslav, Marek HANUŠ, Jan HOLUB a Michal PAULÍČEK. PPMC. *Knihovna vizualizačních appletů pro kompresi dat* [online]. 2006 [cit. 2012-03-24]. Dostupné z: [http://www.stringology.org/DataCompression/ppmc/index\\_cs.html](http://www.stringology.org/DataCompression/ppmc/index_cs.html)
- [18] TIŠNOVSKÝ, Pavel. Grafický formát BMP - používaný a přitom neoblíbený. *Root.cz* [online]. Internet Info, 19.10.2006 [cit. 2012-04-02]. Dostupné z: <http://www.root.cz/clanky/graficky-format-bmp-pouzivany-a-pritom-neoblíbeny/>
- [19] TIŠNOVSKÝ, Pavel. BMP na vícero způsobů. *Root.cz* [online]. Internet Info, 26.10.2006 [cit. 2012-04-02]. Dostupné z: <http://www.root.cz/clanky/bmp-na-vicero-zpusobu/>

- [20] TIŠNOVSKÝ, Pavel. Pravda a mýty o GIFu. *Root.cz* [online]. Internet Info, 17.8.2006 [cit. 2012-03-03]. Dostupné z: <http://www.root.cz/clanky/pravda-a-myty-o-gifu/>
- [21] TIŠNOVSKÝ, Pavel. Anatomie grafického formátu GIF. *Root.cz* [online]. Internet Info, 24.8.2006 [cit. 2012-03-05]. Dostupné z: <http://www.root.cz/clanky/anatomie-grafickeho-formatu-gif/>
- [22] TIŠNOVSKÝ, Pavel. GIF: animace a konkurence. *Root.cz* [online]. Internet Info, 31.8.2006 [cit. 2012-03-05]. Dostupné z: <http://www.root.cz/clanky/gif-animace-a-konkurence/>
- [23] TIŠNOVSKÝ, Pavel. Grafický formát PCX - výlet do historie PC. *Root.cz* [online]. Internet Info, 16.11.2006 [cit. 2012-02-26]. Dostupné z: <http://www.root.cz/clanky/graficky-format-pcx-vylet-do-historie-pc/>
- [24] TIŠNOVSKÝ, Pavel. PCX prakticky - implementace komprimace RLE. *Root.cz* [online]. Internet Info, 23.11.2006 [cit. 2012-02-28]. Dostupné z: <http://www.root.cz/clanky/pcx-prakticky-implementace-komprimace-rle/>
- [25] TIŠNOVSKÝ, Pavel. PNG is Not GIF. *Root.cz* [online]. Internet Info, 7.9.2006 [cit. 2012-03-08]. Dostupné z: <http://www.root.cz/clanky/png-is-not-gif/>
- [26] Portable Network Graphics (PNG) Specification (Second Edition): Information technology — Computer graphics and image processing — Portable Network Graphics (PNG): Functional specification. ISO/IEC 15948:2003 (E). W3C. *World Wide Web Consortium (W3C)* [online]. 10.11.2003 [cit. 2012-03-09]. Dostupné z: <http://www.w3.org/TR/PNG/>
- [27] TIŠNOVSKÝ, Pavel. Řádkové filtry v PNG. *Root.cz* [online]. Internet Info, 29.9.2006 [cit. 2012-03-08]. Dostupné z: <http://www.root.cz/clanky/radkove-filtry-v-png/>
- [28] TIŠNOVSKÝ, Pavel. Anatomie grafického formátu PNG. *Root.cz* [online]. Internet Info, 14.9.2006 [cit. 2012-03-08]. Dostupné z: <http://www.root.cz/clanky/anatomie-grafickeho-formatu-png/>

- [29] TIŠNOVSKÝ, Pavel. PNG - bity, byty, chunky. *Root.cz* [online]. Internet Info, 21.9.2006 [cit. 2012-03-08]. Dostupné z: <http://www.root.cz/clanky/png-bity-byty-chunky/>
- [30] TIŠNOVSKÝ, Pavel. Grafický formát TGA - jednoduchý, oblíbený, používaný. *Root.cz* [online]. Internet Info, 2.11.2006 [cit. 2012-02-28]. Dostupné z: <http://www.root.cz/clanky/graficky-format-tga-jednoduchy-oblibeny-pouzivany/>
- [31] TRUEVISION. *Truevision TGA: FILE FORMAT SPECIFICATION 2.0* [online]. 1991 [cit. 2012-02-28]. Dostupné z: <http://www.dca.fee.unicamp.br/~martino/disciplinas/ea978/tgaffs.pdf>
- [32] TIŠNOVSKÝ, Pavel. Grafický formát TGA prakticky. *Root.cz* [online]. Internet Info, 9.11.2006 [cit. 2012-02-28]. Dostupné z: <http://www.root.cz/clanky/graficky-format-tga-prakticky/>
- [33] WARMERDAM, Frank, Andrey KISELEV, Mike WELLES a Dwight KELLY. Modifying The TIFF Library. *LibTIFF - TIFF Library and Utilities* [online]. 1997, 23.12.2003 [cit. 2012-03-14]. Dostupné z: <http://www.libtiff.org/internals.html>
- [34] ADOBE DEVELOPERS ASSOCIATION. *TIFF: Revision 6.0* [online]. 3.6.1992 [cit. 2012-03-14]. Dostupné z: <http://partners.adobe.com/public/developer/en/tiff/TIFF6.pdf>
- [35] KVÁŠ, Marek. Kompresie JPEG 2000 a akcelerace pomocí DSP. *Elektrorevue* [online]. 9.4.2008, roč. 2008, č. 13, s. 12 [cit. 2012-03-15]. ISSN 1213-1539. Dostupné z: <http://www.elektrorevue.cz/cz/download/kompresie-jpeg-2000-a-akcelerace-pomoci-dsp/>
- [36] ISO/IEC JTC1/SC29 WG1. *INFORMATION TECHNOLOGY: JPEG 2000 IMAGE CODING SYSTEM* [online]. Martin Boliek. 16.3.2000, 11.4.2000 [cit. 2012-03-16]. Dostupné z: <http://www.jpeg.org/public/fcd15444-1.pdf>
- [37] MICROSOFT. *HD Photo: Photographic Still Image File Format* [online]. 16.11.2006, 43 s. [cit. 2012-03-18]. Dostupné z: [http://download.microsoft.com/download/7/4/3/74394b57-2028-4859-a668-c4c0af0726e2/HDPhoto\\_Feature\\_Spec\\_1.0.doc](http://download.microsoft.com/download/7/4/3/74394b57-2028-4859-a668-c4c0af0726e2/HDPhoto_Feature_Spec_1.0.doc)

- [38] ITU-T. *T.832: Information technology – JPEG XR image coding system – Image coding specification* [online]. Březen 2009, 212 s. [cit. 2012-03-18]. Dostupné z: [http://www.itu.int/rec/dologin\\_pub.asp?lang=e&id=T-REC-T.832-200903-S!!PDF-E&type=items](http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-T.832-200903-S!!PDF-E&type=items)
- [39] WEINBERGER, Marcelo, Gadiel SEROUSSI a Guillermo SAPIRO. HEWLETT-PACKARD. *The LOCO-I Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS* [online]. 9. vydání. USA, Srpen 2000, 34 s. [cit. 2012-03-22]. Dostupné z: <http://www.hpl.hp.com/loco/HPL-98-193R1.pdf>
- [40] ITU-T. *T.87: Information technology – Lossless and near-lossless compression of continuous-tone still images – Baseline* [online]. 18.6.1998, 75 s. [cit. 2012-03-22]. Dostupné z: [http://www.itu.int/rec/dologin\\_pub.asp?lang=e&id=T-REC-T.87-199806-I!!ZPF-E&type=items](http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-T.87-199806-I!!ZPF-E&type=items)
- [41] JPEG image compression FAQ, part 1/2: Section - [13] Isn't there a lossless JPEG?. ADVAMEG. *Internet FAQ Archives* [online]. 28.3.1999, 21.11.2011 [cit. 2012-03-22]. Dostupné z: <http://www.faqs.org/faqs/jpeg-faq/part1/section-13.html>
- [42] WALLACE, Gregory K. *The JPEG Still Picture Compression Standard* [online]. Prosinec 1991, 17 s. [cit. 2012-03-22]. Dostupné z: <http://white.stanford.edu/~brian/psy221/reader/Wallace.JPEG.pdf>
- [43] OpenEXR: nový open-source formát pro počítačovou grafiku. KREJČÍ, Richard. *Grafika* [online]. 12.2.2003 [cit. 2012-03-17]. Dostupné z: <http://www.grafika.cz/art/dv/openexr.html>
- [44] KAINZ, Florian a Rod BOGART. INDUSTRIAL LIGHT & MAGIC. *Technical Introduction to OpenEXR* [online]. 18.2.2009, 15 s. [cit. 2012-03-17]. Dostupné z: <http://www.openexr.com/TechnicalIntroduction.pdf>
- [45] INDUSTRIAL LIGHT & MAGIC. *OpenEXR File Layout* [online]. 24.4.2007, 11 s. [cit. 2012-03-17]. Dostupné z: <http://www.openexr.com/openexrfilelayout.pdf>
- [46] MASSIMINO, Pascal. Experimental branch. In: *Groups Google* [online]. 28.11.2011 [cit. 2012-03-22]. Dostupné z: [https://groups.google.com/a/webmproject.org/group/webp-discuss/browse\\_thread/thread/bf368050925aeb8e#](https://groups.google.com/a/webmproject.org/group/webp-discuss/browse_thread/thread/bf368050925aeb8e#)

- [47] Lossless and Transparency Encoding in WebP. GOOGLE. *WebP* [online]. 17.11.2011 [cit. 2012-03-19]. Dostupné z: [http://code.google.com/intl/cs/speed/webp/docs/webp\\_lossless\\_alpha\\_study.html](http://code.google.com/intl/cs/speed/webp/docs/webp_lossless_alpha_study.html)
- [48] Webp lossless – first impressions. *I am an extreme moderate* [online]. 20.11.2011 [cit. 2012-03-19]. Dostupné z: <http://extrememoderate.wordpress.com/2011/11/20/webp-lossless-first-impressions/>
- [49] RATUSHNYAK, Alexander. FLIC - a new fast lossless image compressor. In: *Encode's Forum* [online]. Listopad 2011 [cit. 2012-03-19]. Dostupné z: <http://encode.ru/threads/1222-FLIC-a-new-fast-lossless-image-compressor>
- [50] A web-centric image compression benchmark. *I am an extreme moderate* [online]. 28.11.2011 [cit. 2012-03-19]. Dostupné z: <http://extrememoderate.wordpress.com/2011/11/28/a-web-centric-image-compression-benchmark/>
- [51] IFRAH, ERAN. *CodeLite with MinGW and wxWidgets* [software]. Ver. 2.10.0.4778, Ver. 4.4.1, Ver. 2.8.12. 12.4.2011. [cit. 2012-04-27]. Dostupné z: <http://sourceforge.net/projects/codelite/files/Releases/codelite-2.10/codelite-2.10.0.4778-mingw4.4.1-wx2.8.12.exe/download>
- [52] HURTADO, Jose Antonio, RJP COMPUTING, RJMYST3, Michal BLIŽŇÁK, Jérémie FOUCHÉ. *wxFormBuilder* [software]. Ver. 3.3.0-beta. 2.12.2011 [cit. 2012-04-27]. Dostupné z: [http://sourceforge.net/projects/wxformbuilder/files/wxformbuilder-nightly/3.3.0-beta/wxFormBuilder\\_v3.3.0-beta.exe/download](http://sourceforge.net/projects/wxformbuilder/files/wxformbuilder-nightly/3.3.0-beta/wxFormBuilder_v3.3.0-beta.exe/download)
- [53] DROLON Hervé. *FreeImage* [software]. Ver. 3.15.3. 17.3.2012 [cit. 2012-04-27]. Dostupné z: <http://downloads.sourceforge.net/freeimage/FreeImage3153Win32.zip>
- [54] KUHN Markus. *JBIG-KIT* [software]. Ver. 2.0. 30.8.2008 [cit. 2012-04-30]. Dostupné z: <http://www.cl.cam.ac.uk/~mgk25/download/jbigkit-2.0.tar.gz>
- [55] MICROSOFT. *HD Photo Device Porting Kit 1.0* [software]. Ver. 1.0. 3.4.2010 [cit. 2012-04-27]. Dostupné z:

[http://download.microsoft.com/download/1/7/4/1743e193-c99b-4ffa-9365-b3bb4c2a736a/hdphoto\\_1.0\\_dpk\\_setup.exe](http://download.microsoft.com/download/1/7/4/1743e193-c99b-4ffa-9365-b3bb4c2a736a/hdphoto_1.0_dpk_setup.exe)

- [56] HEWLETT-PACKARD. *Loco executables* [software]. Ver. 1.0.0. Říjen 1999 [cit. 2012-04-27]. Dostupné z: <http://www.hpl.hp.com/loco/jlsrefV100.zip>
- [57] DROLON, Hervé. *FreeImage: a free, open source graphics library* [online]. 3.15.3. 17.3.2012, 129 s. [cit. 2012-04-27]. Dostupné z: <http://downloads.sourceforge.net/freeimage/FreeImage3153.pdf>
- [58] *Ulož.to* [online]. Nodus Technologies, 2012 [cit. 2012-05-11]. Dostupné z: <http://www.uloz.to/>
- [59] *Photojournal: NASA's Image Access Home Page* [online]. JPL, 2012 [cit. 2012-05-11]. Dostupné z: <http://photojournal.jpl.nasa.gov/index.html>
- [60] The New Test Images. *Image Compression* [online]. 2011 [cit. 2012-05-11]. Dostupné z: [http://www.imagecompression.info/test\\_images/](http://www.imagecompression.info/test_images/)
- [61] *PhotographyBlog* [online]. 2012 [cit. 2012-05-11]. Dostupné z: <http://www.photographyblog.com/>
- [62] Recognition Benchmark Images. *The University of Kentucky Center for Visualization & Virtual Environments* [online]. 2009 [cit. 2012-05-11]. Dostupné z: <http://vis.uky.edu/~stewe/ukbench/>
- [63] The USC-SIPI Image Database. *USC Ming Hsieh Department of Electrical Engineering - Signal and Image Processing Institute* [online]. 2012 [cit. 2012-05-11]. Dostupné z: <http://sipi.usc.edu/database/database.php>
- [64] Light Probe Image Gallery. DEBEVEC, Paul. *Paul Debevec Home Page* [online]. 2010 [cit. 2012-05-12]. Dostupné z: <http://www.pauldebevec.com/Probes/>
- [65] OpenEXR Downloads. *OpenEXR* [online]. Industrial Light and Magic, 2012 [cit. 2012-05-12]. Dostupné z: <http://www.openexr.com/downloads.html>

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

2D	2 Dimenze - dvojrozměrný objekt
ASCII	American Standard Code for Information Interchange - typ kódování textové informace
B	Byte - jednotka množství dat v informatice.
BGR	barevný prostor RGB uložený v pořadí barev ve formátu B, G a R
BH	bitová hloubka obrázku
BMQ	NuCore Raw Image File - nezpracované obrazové data firmy NuCore
BMP	Microsoft Windows Bitmap - grafický formát pro ukládání rastrové počítačové grafiky.
BMP - RLE	formát BMP s kompresí RLE
BP	barevný prostor obrázku
bpB	bits per Byte - počet bitů na 1 Byte. Jednotka pro vyjádření jednoho z možných typů kompresního poměru.
CCITT	International Telegraph and Telephone Consultative Committee - standardizační organizace (nyní ITU-T)
CD	Compact Disc - optický disk určený pro ukládání digitálních dat
CMYK	barevný prostor založený na subtraktivním míchání barev
CR2	Canon Digital Camera RAW Image Format version 2.0 - nezpracované obrazové data firmy Canon. Základem tohoto formátu je struktura TIFF
CSV	Comma-Separated Values (hodnoty oddělené čárkami) - souborový formát určený pro výměnu tabulkových dat
ČEK	Časová Efektivita Kompresce
DD	Doba Dekompresce algoritmu
DK	Doba Kompresce algoritmu

---

DNG	Adobe Digital Negative - nezpracované obrazové data firmy Adobe
DPCM	Differential Pulse Code Modulation - rozdílová pulsně kódová modulace
DWT	Discrete Wavelet Transformation - diskrétní vlnková transformace
EBCOT	Embedded Block Coding with Optimal Truncation - metoda aritmetického entropického kódování
EXR - PIZ	grafický formát EXR s kompresí PIZ
EXR - ZIP	grafický formát EXR s kompresí ZIP
FCT	Forward Core Transform - hlavní dopředná transformace
FIR	Finite Impulse Response - filtr s konečnou impulzní odezvou
G31D	CCITT Group 3 jednoúrovňové
G32D	CCITT Group 3 dvourozměrné
G42D	CCITT Group 4 dvourozměrné
GIF	Graphics Interchange Format - grafický formát pro ukládání rastrové počítačové grafiky.
GIF - LZW	formát GIF s kompresí LZW
GUI	Graphical User Interface - grafické uživatelské rozhraní
HD Photo	High Definition Photo - grafický formát firmy Microsoft, speciálně navržený pro fotografie (známý také jako Windows Media Photo)
HDP - LW	formát HD Photo s kompresí Lossless Wavelet
HDR	High Dynamic Range - vysoký dynamický rozsah (barev)
ID	identifikační kód
IDAT	datová část (pixmap) formátu PNG
IEND	ukončující značka formátu PNG
IFD	Image File Directory - adresář souboru předlohy formátu TIFF

IFH	Image File Header - hlavička souboru předlohy formátu TIFF
IHDR	hlavička obrázku formátu PNG
ITU-T	International Telecommunication Union - Telecommunication Standardization Sector - mezinárodní telekomunikační unie (dříve CCITT)
IWT	Integer Wavelet Transformation - Celočíselná vlnková transformace
JBIG	Joint Bi-level Image Experts Group - neztrátová komprese obrazu, popř. formát, který využívá kompresi JBIG
JBIG - JBIG	formát JBIG s kompresí JBIG
JLS - LOCO-I	formát JPEG-LS s kompresí LOCO-I
JP2 - LW	formát JPEG 2000 s kompresí Lossless Wavelet
JPEG	Joint Photographic Experts Group - grafický formát se ztrátovou kompresí
JPEG 2000	Joint Photographic Experts Group 2000 - grafický formát se ztrátovou a neztrátovou kompresí
JPEG-LS	Joint Photographic Experts Group - Lossless - grafický formát s neztrátovou a mírně ztrátovou kompresí
kB	kiloByte - $2^{10}$ Byte.
KP	kompresní poměr algoritmu
$KP_{\phi}$	průměr kompresních poměrů
$KP_{med}$	medián kompresních poměrů
$KP_{min}$	minimální kompresní poměr
$KP_{max}$	maximální kompresní poměr
LJPEG	Lossless Joint Photographic Experts Group - původní implementace JPEG s neztrátovou kompresí
LOCO-I	LOW COMplexity LOSSless COMpression for Images - typ neztrátové komprese používaný ve formátu JPEG-LS

---

LZ77	kompresní slovníková metoda vytvořena Lemplem a Zivem v roce 1977
LZSS	Lempel-Ziv-Storer-Szymanski - modifikace kompresní metody LZ77
LZW	Lempel-Ziv-Welch - neztrátový slovníkový kompresní algoritmus
MB	MegaByte - $2^{20}$ Byte.
NEF	Nikon Digital Camera Raw Image Format - nezpracované obrazové data firmy Nikon
OpenEXR	grafický formát pro obrázky s vysokou dynamikou barev
ORF	Olympus Digital Camera Raw Image Format - nezpracované obrazové data firmy Olympus
OŠ	odstíny šedi
PC	Personal Computer - osobní počítač
PCX	Personal Computer eXchange - formát pro ukládání rastrové počítačové grafiky.
PCX - RLE	grafický formát PCX s kompresí RLE
PFM	Portable Floatmap - formát s podporou barevného prostoru RGBF
PNG	Portable Network Graphics - grafický formát určený pro bezztrátovou kompresi rastrové grafiky.
PNG - D1	formát PNG s kompresí Deflate úrovně 1
PNG - D9	formát PNG s kompresí Deflate úrovně 9
PPM	Prediction by Partial Matching - adaptivní statistická technika komprese
RAF	Fuji Digital Camera Raw Image Format - nezpracované obrazové data firmy Fuji
RAM	Random-Access Memory - operační paměť s přímým přístupem
RAW	označení obrázkových souborů obsahující nezpracovaná data

RGB	barevný prostor s barvami Red (červená), Green (zelená) a Blue (modrá)
RGBA	barevný prostor RGB s alfa kanálem (průhlednost)
RGBAf	barevný prostor RGBF s alfa kanálem (průhlednost)
RGBF	barevný prostor RGB. Každá složka je uložena datovým typem Float.
RLE	Run-Length Encoding - komprese proudového kódování
TGA	Truevision Graphics Adapter - formát pro ukládání rastrové počítačové grafiky.
TGA - RLE	formát TGA s kompresí RLE
TIFF	Tag Image File Format - formát pro ukládání rastrové počítačové grafiky.
TIFF - DEFL	formát TIFF s kompresí DEFLATE
TIFF - CCITT3	formát TIFF s kompresí CCITT3
TIFF - CCITT4	formát TIFF s kompresí CCITT4
TIFF - LZW	formát TIFF s kompresí LZW
TIFF - PACK	formát TIFF s kompresí PACKBITS
Typ K	typ komprese
YUV	barevný model používaný v televizním vysílání v normě PAL
WWW	World Wide Web - celosvětové propojení počítačových sítí
$\sigma_{KP}$	směrodatná odchylka kompresních poměrů

## SEZNAM OBRÁZKŮ

<i>Obr.1 Geometrická reprezentace prostoru RGB [1].....</i>	13
<i>Obr.2 Typy palet [3].....</i>	14
<i>Obr.3 Základní schéma algoritmu RLE [5] .....</i>	20
<i>Obr.4 RLE paket na Bytové úrovni [1] .....</i>	21
<i>Obr.5 Časově-kmitočtové rozlišení vlnkové transformace [13].....</i>	31
<i>Obr.6 Frekvenční pohled na diskretní vlnkovou transformaci [13] .....</i>	32
<i>Obr.7 Jeden krok DWT a rozklad na aproximace a detaily [13].....</i>	32
<i>Obr.8 Blokové schéma několikastupňového celočíselného liftingu [15] .....</i>	33
<i>Obr.9 Blokové schéma predikčního kodéru a dekodéru [16] .....</i>	34
<i>Obr.10 Princip dvourozměrné predikce [16].....</i>	34
<i>Obr.11 Postup komprese formátu JPEG 2000 [35].....</i>	44
<i>Obr.12 Postup komprese formátu HD Photo.....</i>	46
<i>Obr.13 Pozice sousedních hodnot využívaných predikcí ve formátu JPEG-LS [8].....</i>	48
<i>Obr.14 Postup komprese formátu JPEG-LS [39] .....</i>	48
<i>Obr.15 Postup komprese formátu LJPEG [42] .....</i>	49
<i>Obr.16 Záložka v menu - Soubor .....</i>	54
<i>Obr.17 Záložka v menu - Obrázek .....</i>	54
<i>Obr.18 Záložka v menu - Náповěda .....</i>	54
<i>Obr.19 Prohlížení obrázků v aplikaci.....</i>	55
<i>Obr.20 Dialog s nastavením komprese formátů .....</i>	56
<i>Obr.21 Průběh provádění testů .....</i>	58
<i>Obr.22 Report ve vytvořené aplikaci .....</i>	59
<i>Obr.23 Výsledky testů otevřené v aplikaci Microsoft Excel.....</i>	59
<i>Obr.24 Dialog pro snížení počtu barev .....</i>	60
<i>Obr.25 Dialog pro mapování barev formátu s vysokým dynamickým rozsahem.....</i>	61
<i>Obr.26 Graf výsledku testů pro fotografie formátu JPEG.....</i>	70
<i>Obr.27 Graf výsledku testů pro fotografie formátu RAW.....</i>	72
<i>Obr.28 Graf výsledku testů pro fotografie v odstínech šedi .....</i>	73
<i>Obr.29 Graf výsledku testů pro fotografie s vysokou dynamikou barev.....</i>	74
<i>Obr.30 Graf výsledku testů pro fotografie s vysokou dynamikou barev v odstínech šedi.....</i>	75

---

<i>Obr.31 Graf výsledku testů pro obrázky ve 24 bitové hloubce .....</i>	<i>77</i>
<i>Obr.32 Graf výsledku testů pro obrázky v 8 bitové hloubce .....</i>	<i>78</i>
<i>Obr.33 Graf výsledku testů pro obrázky ve 4 bitové hloubce .....</i>	<i>79</i>
<i>Obr.34 Graf výsledku testů pro obrázky v 1 bitové hloubce .....</i>	<i>81</i>
<i>Obr.35 Graf výsledku testů pro obrázky v odstínech šedi .....</i>	<i>82</i>
<i>Obr.36 Graf výsledku testů pro obrázky s textem v 1 bitové hloubce .....</i>	<i>84</i>
<i>Obr.37 Graf výsledku testů pro obrázky s textem v odstínech šedi .....</i>	<i>85</i>

**SEZNAM TABULEK**

<i>Tab.1 Struktura formátu BMP [18]</i> .....	37
<i>Tab.2 Bloky formátu GIF [21]</i> .....	38
<i>Tab.3 Hodnoty tagu Compression dle specifikace [34]</i> .....	43
<i>Tab.4 Doplnující hodnoty tagu Compression dle dokumentace knihovny libtiff [33]</i> .....	43
<i>Tab.5 Základní struktura formátu JPEG-LS [40]</i> .....	47
<i>Tab.6 Kompresní schémata formátu OpenEXR [44] [45]</i> .....	50
<i>Tab.7 Seznam souborů nutných pro spuštění aplikace</i> .....	53
<i>Tab.8 Volitelné formáty a komprese pro provádění testů</i> .....	57
<i>Tab.9 Význam zkratk v reportu</i> .....	62
<i>Tab.10 Význam zkratk v tabulkách s výsledky</i> .....	69
<i>Tab.11 Výsledky testů pro fotografie formátu JPEG</i> .....	70
<i>Tab.12 Výsledky testů pro fotografie formátů RAW</i> .....	71
<i>Tab.13 Výsledky testů pro fotografie v odstínech šedi</i> .....	73
<i>Tab.14 Výsledky testů pro fotografie s vysokou dynamikou barev</i> .....	74
<i>Tab.15 Výsledky testů pro fotografie s vysokou dynamikou barev v odstínech šedi</i> .....	75
<i>Tab.16 Výsledky testů pro obrázky ve 24 bitové hloubce</i> .....	76
<i>Tab.17 Výsledky testů pro obrázky v 8 bitové hloubce</i> .....	78
<i>Tab.18 Výsledky testů pro obrázky ve 4 bitové hloubce</i> .....	79
<i>Tab.19 Výsledky testů pro obrázky v 1 bitové hloubce</i> .....	80
<i>Tab.20 Výsledky testů pro obrázky v odstínech šedi</i> .....	82
<i>Tab.21 Výsledky testů pro obrázky s textem v 1 bitové hloubce</i> .....	83
<i>Tab.22 Výsledky testů pro obrázky s textem v odstínech šedi</i> .....	85

## SEZNAM PŘÍLOH

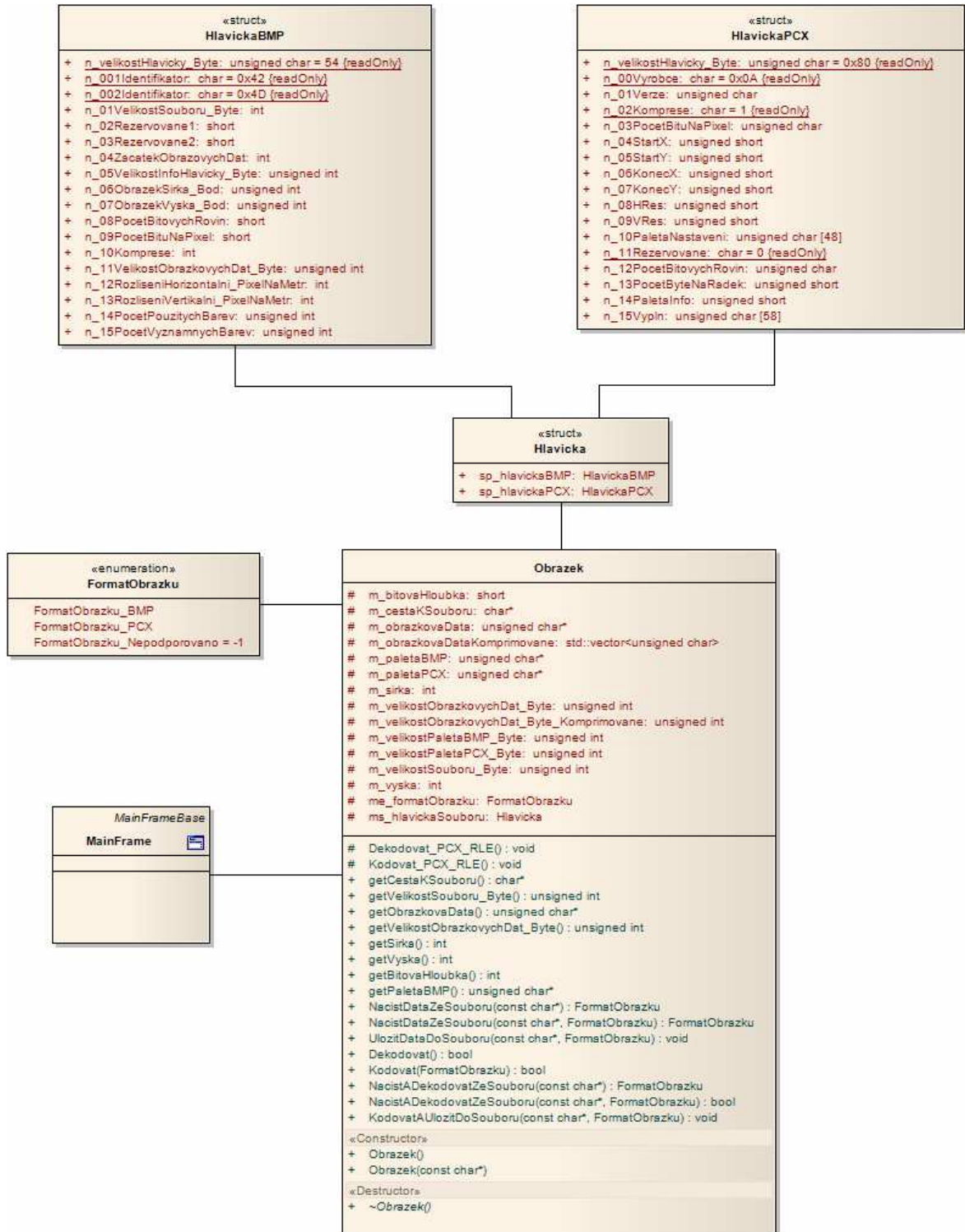
Příloha P I: Diagram třídy Obrazek

Příloha P II: Diagram třídy FreeImageRozhraniClass

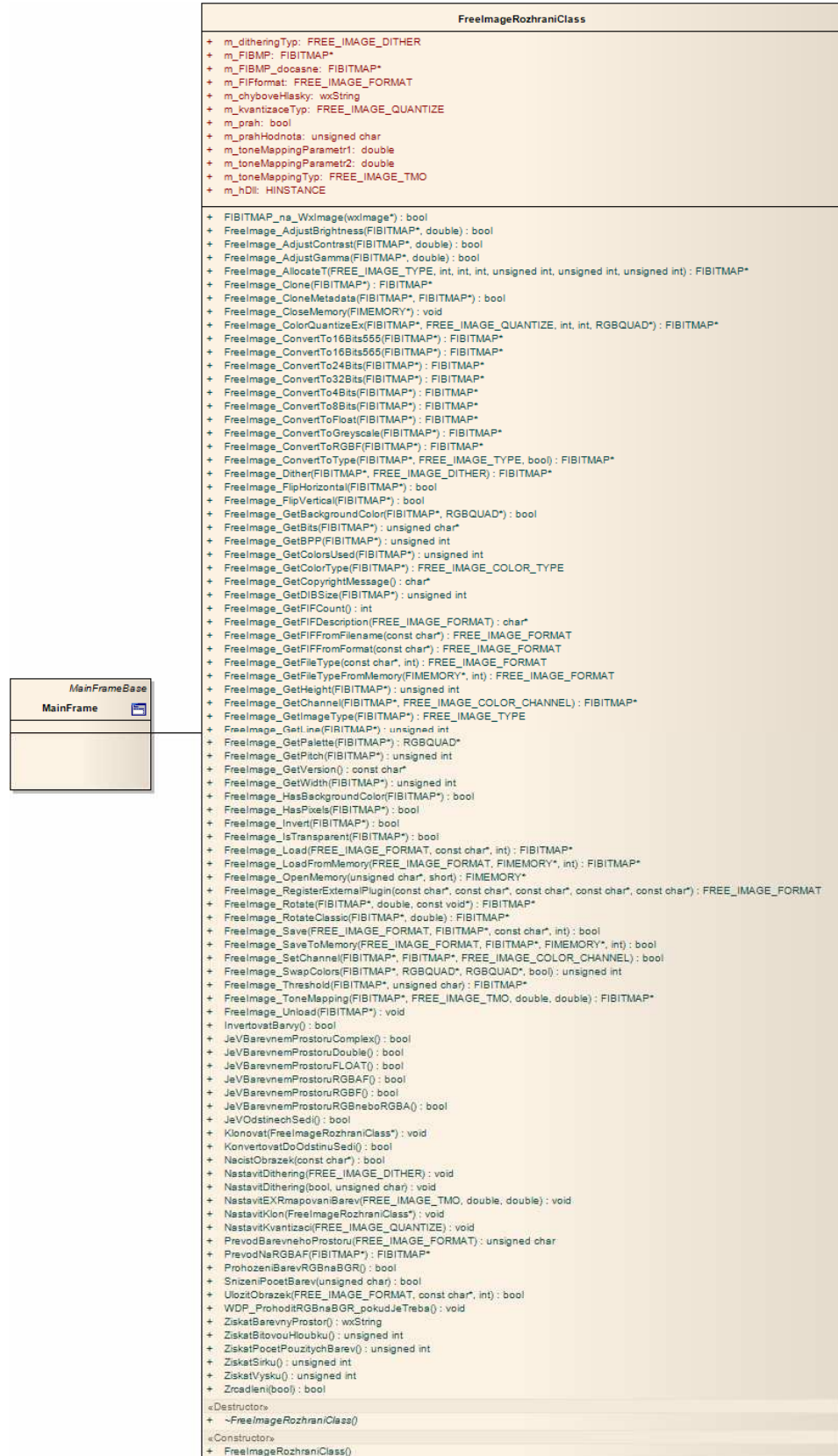
Příloha P III: Diagram třídy MainFrame

Příloha P IV: Sumarizace výsledků testů

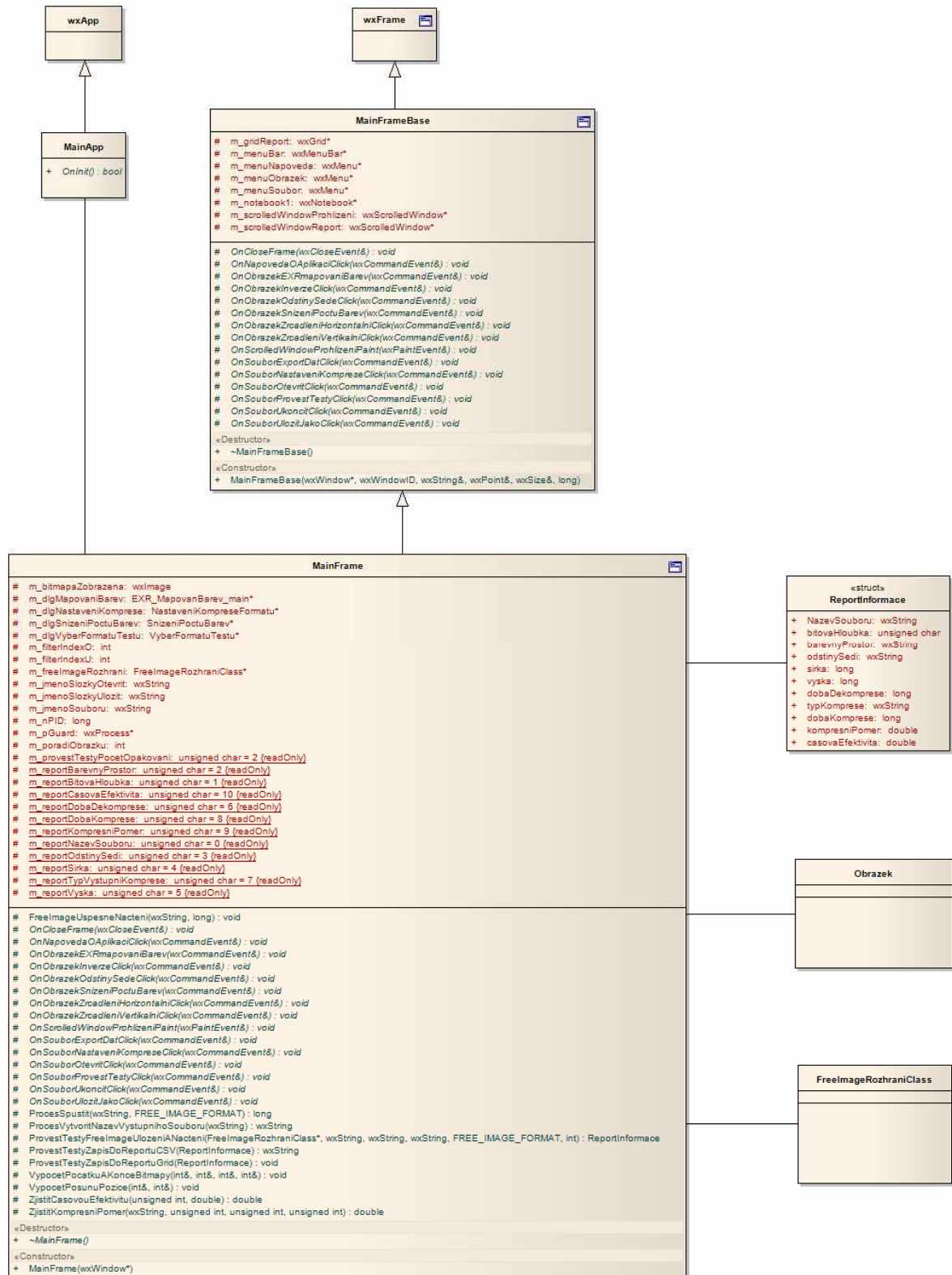
# PŘÍLOHA P I: DIAGRAM TŘÍDY OBRAZEK



# PŘÍLOHA P II: DIAGRAM TRÍDY FREEIMAGEROZHRAICLASS



# PŘÍLOHA P III: DIAGRAM TŘÍDY MAINFRAME



## PŘÍLOHA P IV: SUMARIZACE VÝSLEDKŮ TESTŮ

Testovaná kategorie	Nejlepší kompresní algoritmus	Kapitola
Fotografie formátu JPEG	JP2 - LW	6.1.1
Fotografie formátu RAW	JP2 - LW	6.1.2
Fotografie v odstínech šedi	JLS - LOCO-I	6.1.3
Fotografie s vysokou dynamikou barev	EXR - ZIP	6.1.4
Fotografie s vysokou dynamikou barev v odstínech šedi	EXR - ZIP	6.1.5
Obrázky ve 24 bitové hloubce	JP2 - LW	6.2.1
Obrázky v 8 bitové hloubce	PNG - D9	6.2.2
Obrázky v 4 bitové hloubce	PNG - D9	6.2.3
Obrázky v 1 bitové hloubce	JBIG - JBIG	6.2.4
Obrázky v odstínech šedi	PNG - D9	6.2.5
Text v 1 bitové hloubce	JBIG - JBIG	6.3.1
Text v odstínech šedi	JLS - LOCO-I	6.3.2