VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

BEZEZTRÁTOVÁ KOMPRESE OBRAZU

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE GERGELY KOMJÁTHY
AUTHOR

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

BEZEZTRÁTOVÁ KOMPRESE OBRAZU

LOSSLESS IMAGE COMPRESSION

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — GERGELY KOMJÁTHY

AUTHOR

VEDOUCÍ PRÁCE — Ing. DAVID BAŘINA

SUPERVISOR

BRNO 2011

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2010/2011

Zadání bakalářské práce

Řešitel: **Komjáthy Gergely**
Obor: Informační technologie
Téma: **Bezeztrátová komprese obrazu**
Lossless Image Compression
Kategorie: Zpracování obrazu

Pokyny:

1. Seznamte se s metodami bezeztrátové komprese obrazu (barevné modely, predikce, kódování rozdílu, entropické kódování).
2. Navrhněte postup komprese obrazu s použitím vybraných metod.
3. Implementujte tento postup formou multiplatformní knihovny v jazyce C nebo C++. Umožněte volbu barevného modelu a prediktoru.
4. Srovnejte dosaženou účinnost komprese při různých nastaveních. Dosaženou účinnost porovnejte také se v současnosti používanými formáty (PNG).
5. Diskutujte výsledky Vaší práce a výhody jednotlivých kompresních metod.

Literatura:

- R. J. van der Vleuten and S. Egner, "Lossless and fine-granularity scalable near-lossless color image compression", 25th Symp. Inform. Theory in the Benelux, (Kerkrade, The Netherlands), pp. 209-216, June 2-4, 2004.
- M. Weinberger, G. Seroussi, G. Sapiro, "The LOCO-I Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS", Hewlett-Packard Laboratories Technical Report No. HPL-98-193R1, November 1998, revised October 1999. IEEE Trans. Image Processing, Vol. 9, August 2000, pp.1309-1324.

Při obhajobě semestrální části projektu je požadováno:

- První dva body zadání. Funkční prototyp.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Bařina David, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2010
Datum odevzdání: 18. května 2011

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato práce se zabývá bezeztrátovou kompresí obrazu. Jsou zde uvedeny některé barevné modely, vhodné pro bezeztrátovou kompresi, a vzorce použité pro převody mezi nimi a RGB modelem. Dále práce pojednává o prediktorech a jejich fungování. Je zde popsána funkčnost aritmetického a PPM kódéru, a stručný popis Huffmanova kódování.

Abstract

This thesis deals with lossless image compression. In this paper are shown some colour models, which can be used for lossless image compression and formulas how to convert them to RGB and vica versa. You can learn predictors, how they work and discription of some of them. There is described the function of arithmetic coder, PPM coder and a brief description of Huffman coding.

Klíčová slova

komprese, obraz, bezeztrátová komprese, barevné modely, predikce, entropické kódování, aritmetické kódování, PPM, RLE, MTF

Keywords

compression, image, lossless compression, color models, prediction, entopy coding, arithmetic coding, PPM, RLE, MTF

Citace

Gergely Komjáthy: Bezeztrátová komprese obrazu, bakalářská práce, Brno, FIT VUT v Brně, 2011

Bezeztrátová komprese obrazu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Davida Bařiny

. . . . . .
Gergely Komjáthy
17. května 2011

Poděkování

Rád bych poděkoval vedoucímu své bakalářskej práce Ing. Davidu Bařinovi za připomínky, rady a odbornou pomoc, kterou mi během práce poskytoval.

Obsah

Kapitola 1

Úvod

Obrázky môžu byť uložené v komprimovanej alebo v nekomprimovanej forme. Málokedy však používame nekomprimované obrázky, hlavne kvôli ich veľkosti. S komprimovanými obrázkami sa stretávame častejšie. Táto bakalárska práca sa zaoberá problematikou bezstratovej kompresie, čo je jedným druhom komprimovania.

Kompresia je dôležitou časťou informačných technológii. Nové technológie, nové formáty dát sa objavia každý deň, ktoré potrebujú stále viac priestoru k uloženiu. Práve preto je dôležitá kompresia, aby sme mohli čím viac dát ukladať, a tým rýchlejšie ich preniesť napríklad cez internet. Kompresia obrazu sa skladá z niekoľkých krokov.

Prvým krokom je prevod farebného modelu. Obrazy sú najčastejšie uložené vo farebnom modely RGB, ktorý nie je vhodný pre kompresiu. Niekoľko vhodnejších modelov a vzťahy prevodu z RGB resp. do RGB sú popísané v kapitole 3.

Ďalším krokom kompresie obrazu je predikcia, teda pokus o uhádnutie nasledujúceho pixela. Keď zvolíme vhodný prediktor, chyba v predikcii bude malá, takže početnosť jednotlivých hodnôt bude veľká okolo hodnoty 0, čo je vhodné pre kompresiu. S prediktormi sa môžeme zoznámiť v kapitole 4.

Posledným krokom kompresie je entropické kódovanie. S entropickým kódovaním, čo je bežne používaná kompresná metóda, sa zaoberá kapitola 5. Popisujeme v krátkosti Huffmanovo kódovanie, a rozsiahlejšie sa zaoberáme s aritmetickým kódovaním, ďalej je popísané PPM kódovanie, RLE a MTF transformácia.

Implementácia knižnice je popísaná v kapitole 6 spolu s návodom k použitiu. V poslednej kapitole – kapitola 7 sú uvedené dosiahnuté výsledky a porovnávanie použitých metód.

Kapitola 2

Kompresia

Kompresia je metóda, ktorá prevedie vstupné dáta na výstupné, ktoré majú menšiu veľkosť. Pred použitím dát po kompresii musíme previesť tieto dáta späť do ich pôvodného stavu. Tento postup sa volá dekompresia. Rozlišujeme 2 druhy kompresie: bezstratovú a stratovú kompresiu.

Na komprimovanie dát existuje viac metód, ale sú spoločné v tom, že všetky skúsia odstrániť redundanciu. **Redundancia** je nadbytočná informácia v dátach.

Kompresia je veľmi často používaná metóda takmer vo všetkých častiach informačných technológií. Hlavným dôvodom použitia kompresie je nedostatok miesta na médiach – síce dnešné média majú veľkú kapacitu, ale aj veľkosť súborov je stále väčšia. Kompresia sa používa aj pri prenose súborov cez internet. Načítanie stránok by trvalo oveľa dlhší čas, keby namiesto komprimovaných obrázkov sa používali obrázky nekomprimované.

Ďalším argumentom pre používanie kompresie môže byť rýchlosť prístupu k dátam. Napriek tomu, že komprimované dáta treba najprv dekomprimovať, vo viacerých prípadoch je prístup k dátam rýchlejší, pretože procesory sú výkonnejšie než disky.

To, že aký stupeň má kompresia, záleží na dvoch faktoroch: druhu dát a použitom algoritme. Najlepšie komprimovateľné sú súbory, v ktorých sa často opakujú rovnaké reťazce. Taký súbor je napríklad čistý text alebo obrázok, ktorý má jednofarebnú schému. Existujú aj dáta, ktoré nie sú vhodné pre kompresiu. Sú to napríklad náhodné dáta alebo už skomprimované dáta.

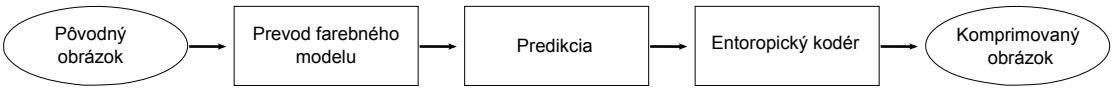
Stupeň kompresie je daný kompresným pomerom. **Kompresný pomer** je pomer komprimovaných dát ku nekomprimovaným. To znamená, že keď je skomprimovaný súbor proti pôvodným polovičný, kompresný pomer je 0,5. Kompresný pomer môže byť vyjadrený aj v "bit na znak"(**bpc** – bits per character). V takých prípadoch sa počíta s tým, že nekomprimované dáta majú ich znaky kódované na 1 byte na znak. 4 bpc odpovedá kompresným pomerom 0,5. Pri kompresii obrázkov kompresný pomer je často vyjadrený aj v **bpp** (bits per pixel) – bitov na pixel, čo udáva, koľko bitov je použitých na ukladanie informácie o jednom pixely. U nekomprimovaných obrázkoch sa bpp zhoduje s farebnou hĺbkou.

Zvyčajne platí, že čím lepšia je kompresia, tým je kódovanie a dekódovanie časovo náročnejšie, a tým viac zdrojov (procesor, pamäť) je potrebných.

2.1 Bezstratová kompresia

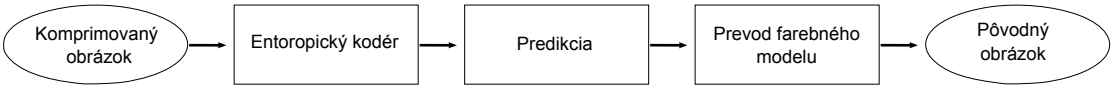
Bezstratová kompresia je metóda, pri ktorej bez zvyšku získame po kompresii a dekompresii rovnaké dáta. Používa sa vtedy, keď nesmie dôjsť k strate dát (spustiteľné programy, texty

a iné dáta, u ktorých potrebujeme zachovať pôvodný obsah).
Kompresia obrazu prebieha nasledovne (obrázok 2.1):



Obrázok 2.1: Postup kompresie obrazu

Pri dekompresii je postup práve opačný (obrázok 2.2):



Obrázok 2.2: Postup dekompresie obrazu

Pre bezstratovú kompresiu obrazu sa používa viac formátov, napríklad PNG, JPEG-LS a JPEG 2000.

2.1.1 PNG

PNG (**P**ortable **N**etwork **G**raphics) [10] bol vyvinutý ako náhrada formátu GIF, ktorý bol patentovo chránený. PNG podporuje 24 bitovú farebnú hĺbku, nie je tak obmedzené na 256 farieb ako GIF a taktiež alfa kanál, čo udáva hodnotu priehľadnosti. Nevýhodou PNG je, že nepodporuje animácie. Podobné formáty vznikli pre podporu animácie, ako napríklad MNG.

PNG používa dvojstupňovú kompresiu: filtrovanie – predikcia a Deflate – kompresia. Na predikciu sa používa jeden z piatich filtrov (bez predikcie, 3 lineárne prediktory alebo prediktor PAETH – kapitola 4.3)

Po predikcii nasleduje kódovanie. PNG používa algoritmus Deflate, čo používa kombináciu algoritmu LZ77 a Huffmanovho kódovania.

2.1.2 JPEG 2000

JPEG 2000 [12] je štandard pre kompresiu obrazu, ktorá umožňuje bezstratovú a stratovú kompresiu. Bol vydaný v roku 2000 ako nástupca formátu JPEG, ale nerozšíril sa. Pri kódovaní obraz môže byť rozdelený na dlaždice, ktoré môžu mať ľubovoľnú veľkosť – až celý obrázok môže byť jedna dlaždica. Používa diskrétnu vlnovú transformáciu (DWT), ktorá môže byť stratová alebo bezstratová. Na kódovanie sa používa aritmetický kodér.

2.1.3 JPEG-LS

JPEG-LS [12] je štandard pre bezstratové kódovanie obrazov, ale tiež podporuje "skoro bezstratové"kódovanie. Pri kódovaní sa používa predikcia, kontextové modelovanie a Golombove kódovanie. JPEG-LS taktiež má "Run-mode", kde počet za sebou nasledujúcich identických pixelov a hodnota pixelu je zakódovaná.

2.2 Stratová kompresia

Stratová kompresia je metóda kompresie dát, ktorá vynechá dáta s malým významom. To znamená, že výsledok dekomprimovania skomprimovaných dát budú odlišné od originálu, ale ešte stále sa dajú použiť. Táto metóda využíva nedokonalosť ľudských zmyslov (zraku a sluchu).

Jej výhodou je, že má vyššiu mieru kompresie ako bezstratová kompresia, ale nie je použiteľná pre všetky typy dát. Využíva sa hlavne pri kompresii zvukových, obrazových a video záznamov. Stratovú kompresiu používajú napríklad metódy JPEG a MPEG. Pri stratovej kompresii si môžeme zvoliť kvalitu výsledku. Čím menšiu kvalitu bude mať výsledok, tým väčšia bude kompresia.

Kapitola 3

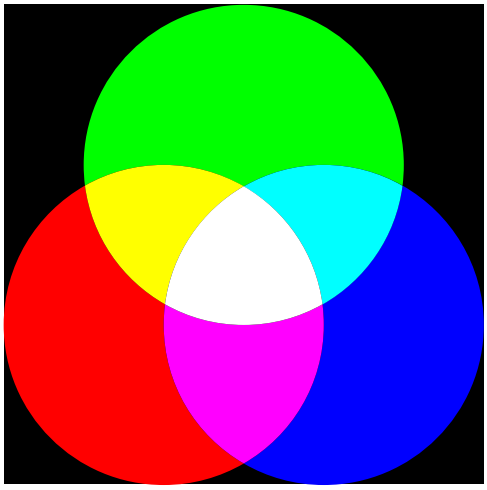
Farebné modely

Svetlo je elektromagnetické žiarenie, ktoré má určitú vlnovú dĺžku a frekvenciu. Svetlo sa šíri od svetelného zdroja, napríklad zo Slnka. Rozlišujeme biele a farebné svetlo. Biele svetlo sa dá rozkladať na jednotlivé farebné zložky: na spektrum. Keď svetlo dopadá na nejaký objekt, časť svetla ten objekt odráža. Túto časť svetla vnímame ako farbu.
Farba [9] je viditeľná oblasť elektromagnetického žiarenia s vlnovými dĺžkami od 400 nm do 700 nm. Ľudia vnímajú farby pomocou fotoreceptorov, ktoré sú umiestnené v sietnici. Tieto fotoreceptory sa nazývajú čapíky. Existujú 3 typy čapíkov, ktoré sú citlivé na rôzne vlnové dĺžky svetla. V sietnici je ešte ďalší typ receptoru: tyčinka. Tyčinky sú oveľa citlivejšie ako čapíky, ale nemajú dôležitú rolu pri vnímaní farieb.
Farba existuje len v očiach a v mozgu. Spektrum viditeľného svetla je uvedené na obrázku 3.1.

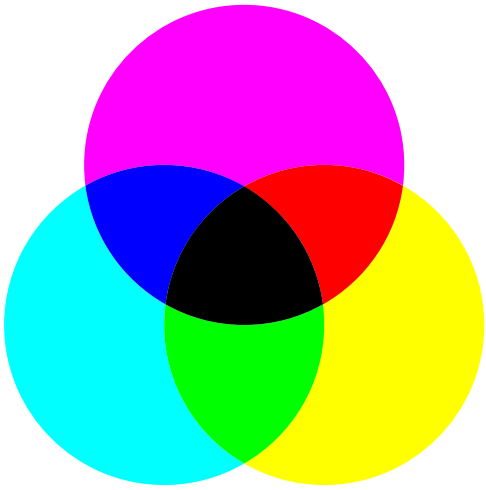
Obrázok 3.1: Spektrum viditeľného svetla [a]

Farebná hĺbka udáva, že koľkými bitmi sú kódované pixely v danom obraze. Čím väčšia je farebná hĺbka, tým viac farieb môže jeden pixel zobrazovať.

Farebný model je popis, pomocou ktorého špecifikujeme farby, umožňuje vytvárať farby a pracovať s nimi. Farebné modely majú niekoľko základných farieb (najčastejšie 3 alebo 4). Ostatné farby sú vytvorené kombináciami – miešaním tých základných farieb. Rozlišujeme dva typy miešania farieb:
Aditívne miešanie farieb sa používa hlavne pri práci so svetlom. Nové farby vznikajú skladaním troch základných farieb. Čím viac komponentov pridáme, tým svetlejšia bude farba. Zložením základných farieb maximálnej intenzity vzniká biela farba (obrázok 3.2).
Subtraktívne miešanie farieb sa hlavne používa v tlačových zariadeniach. Zložením základných farieb maximálnej intenzity vzniká čierna farba (obrázok 3.3).



Obrázok 3.2: Additívne miešanie farieb [b]



Obrázok 3.3: Subtraktívne miešanie farieb [c]

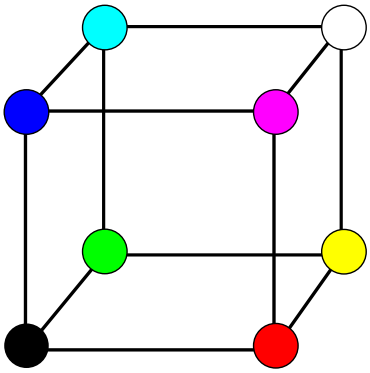
V nasledujúcich podkapitolách sa budeme zaoberať s niektorými farebnými modelmi: RGB, ktorý je najpoužívanejším modelom, YUV, ktorý je často používaný v televíznom vysielaní, LYUV, ktorý je modifikáciou modelu YUV, LOCO-I, ktorý je použitý v štandardu JPEG-LS a ORCT používaný štandardom JPEG 2000.

3.1 Model RGB

Základné farby pre model RGB sú červená, zelená a modrá. Aj názov modelu prichádza z počiatočných písmen anglických názvov základných farieb: **R**ed **G**reen **B**lue.

RGB je najčastejšie používaný farebný model v počítačovej technológii. Obrazy sú najčastejšie reprezentované v tomto farebnom modely a používajú sa u väčšiny obrazoviek (CRT, LCD). Používa sa adaptívne miešanie jeho základných komponentov.

Pre RGB je najčastejšie použitá farebná hĺbka 24 bitová, takže 8 bitov na každú farebnú zložku. Takto reprezentované farby sa označujú ako true-color. V rámci true-color môžeme zobraziť 2^{24} farieb.
Existuje viac variantov modelu RGB. Najrozšírenejšie sú Adobe RGB a sRGB.



Obrázok 3.4: Jednotková kocka pre model RGB [d]

3.2 Model YUV

Väčšina obrazov je špecifikovaná v RGB farebnom modely napriek tomu, že tento farebný model nie je vhodný pre kompresiu dát. Dôvodom nevhodnosti je, že každá zložka obsahuje informácie o farbe, o jasu a o korelácii medzi susednými pixelmi. Pri kompresii teda musíme prevádzať RGB model na iný, vhodnejší pre kompresiu, aby sme dosiahli lepšie výsledky. Takým modelom je model YUV [13]. V modely YUV je samostatne uchovávaná jasová zložka (Y) a farebné zložky (U a V). Ľudské oko lepšie rozpoznáva zmeny v jase ako zmeny vo farbe. To sa dá využiť u tohto modelu – na uchovanie farebnej zložky potrebujeme menší počet bitov.
Na prevod medzi RGB a YUV modelmi môžeme používať nasledujúce vzorce:

$$Y = 0,587 \cdot G + 0,299 \cdot R + 0,114 \cdot B \tag{3.1}$$
$$U = B - Y \tag{3.2}$$
$$V = R - Y \tag{3.3}$$

3.3 Model LYUV

Model YUV je vhodný na kompresiu, ale používa sa na stratovú kompresiu, lebo prevod medzi RGB a YUV je stratový. Preto bolo nutné vytvoriť nové farebné modely, medzi ktorými a modelom RGB je prevod možný bez straty informácie. Jeden z tých modelov je model LYUV [13], ktorý je modifikáciou modelu YUV. Na prevod slúžia nasledujúce vzorce:

$$Y_L = G + \left[\frac{0,299}{0,587} \cdot R + \frac{0,114}{0,587} \cdot B\right] \tag{3.4}$$
$$U_L = [B - 0,587 \cdot Y_L] \tag{3.5}$$
$$V_L = [R - 0,587 \cdot Y_L] \tag{3.6}$$

kde [ ] je zaokrúhlenie na najbližšie celé číslo.
Na prevod späť do modelu RGB môžeme používať vzorce:

$$R = V_L + [0,587 \cdot Y_L] \tag{3.7}$$
$$B = U_L + [0,587 \cdot Y_L] \tag{3.8}$$
$$G = Y_L - \left[\frac{0,299}{0,587} \cdot R + \frac{0,114}{0,587} \cdot B\right] \tag{3.9}$$

3.4 Model LOCO-I

Algoritmus LOCO-I (**LO**w **CO**mplexity **LO**ssless **CO**mpression for Images) [14] sa používa pre bezstratovú kompresiu u JPEG-LS. Prevod z RGB:

$$L_1 = R - G \tag{3.10}$$
$$L_2 = G \tag{3.11}$$
$$L_3 = B - G \tag{3.12}$$

Z LOCO-I späť na RGB :

$$G = L_2 \tag{3.13}$$
$$R = L_1 + G \tag{3.14}$$
$$B = L_3 + G \tag{3.15}$$

3.5 Model ORCT

Model ORCT – **O**riginal **R**eversible **C**olor **T**ransform (CREW – **C**ompression with **R**eversible **E**mbedded **W**avelets) [6] je používaný napríklad štandardom JPEG 2000.
Vzorce na prevody medzi RGB a ORCT

$$Y = \left\lfloor \frac{R + 2 \cdot G + B}{4} \right\rfloor \tag{3.16}$$
$$C_R = R - G \tag{3.17}$$
$$C_B = B - G \tag{3.18}$$

$$G = Y - \left\lfloor \frac{C_R + C_B}{4} \right\rfloor \tag{3.19}$$
$$R = C_R + G \tag{3.20}$$
$$B = C_B + G \tag{3.21}$$

kde $\lfloor\rfloor$ je zaokrúhlenie dolu na najbližšie celé číslo.

Kapitola 4

Prediktory

Cieľom predikcie je uhádnuť hodnotu nasledujúceho pixelu z predchádzajúcich hodnôt. Pretože vo väčšine prípadov v obrazoch sú susedné pixely veľmi často podobné, môžeme odhadnúť hodnotu nasledujúceho pixelu. Keď naša predikcia je dobrá, odhadovaná hodnota sa bude málo líšiť od skutočnej hodnoty, alebo bude so skutočnou hodnotou zhodná. Rozdiel medzi predikčnou a skutočnou hodnotou je chyba predikcie. Výhodou predikcie je, že chyby predikcie je možné lepšie zakódovať, pretože početnosť jednotlivých chýb bude väčšia ako početnosť hodnôt pixelov. Pretože je ľahko implementovateľná, používa sa často pred kódovaním.

4.1 Lineárne prediktory

Lineárne prediktory predikujú nasledujúci pixel pomocou predchádzajúcich pixelov tak, že predikovaná hodnota je lineárnou kombináciou niektorých už spracovaných pixelov. Počty pixelov použitých na predikciu podáva rád prediktora. To znamená, že keď pre predikciu používame len 1 pixel (pixel pred alebo nad skúmaným pixelom), tak hovoríme o prediktore prvého rádu.

4.2 Práca s prediktormi

Predikovanú hodnotu inicializujeme na nulu. Potom skúsime uhádnuť nasledujúci pixel. Chyba predikcie bude daná rozdielom medzi hodnotou pixelu a predikovanej hodnoty. Potom sa prepočíta predikovaná hodnota a predikuje ďalší pixel. V krajných bodoch sa môže stať, že nie je možné hneď použiť prediktor zvoleného rádu. Vtedy môžeme používať prediktor nižšieho rádu alebo prediktor iný. Funkčnosť predikcie prvého rádu je zobrazená v tabuľke 4.1.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Tabuľka 4.1: Originálne hodnoty, hodnoty predikcie a predikovaná hodnota

Z príkladu sa dá vidieť, že aký účinný môže byť aj najjednoduchší prediktor. Pri kódovaní teraz musíme zakódovať len jeden typ symbolu, namiesto pôvodných 25, čím môžeme dosiahnuť oveľa lepší kompresný pomer.

Po dekódovaní dostaneme chyby predikcie, potom z týchto hodnôt musíme vypočítať pôvodné hodnoty. To vypočítame podobnou metódou, len namiesto odčítania prediktoru, sčítame – opačné sekvencie: tabuľka 4.1.

Samozrejme na reálnych dátach vo väčšine prípadov nie sú prediktory tak efektívne, ale chyby predikcie sú obvykle nízke čísla s hustým rozdelením okolo nuly, takže sú lepším vstupom entropických kodérov, ako pôvodné dáta. Účinnosť prediktoru sa dá vizualizovať pomocou histogramov (obrázok 7.5 až 7.8). Ako aj z histogramov vyplýva, početnosť jednotlivých symbolov sa výrazne mení po použití prediktoru. Zvyšuje sa početnosť znaku okolo hodnoty nula, čím môžeme dosiahnuť lepší kompresný pomer.

4.3 Prediktor PAETH

Prediktor PAETH (Paethov filter) sa používa vo formáte PNG. Pri predikovaní hodnoty sa používajú 3 pixely (tabuľka 4.2): pixel nad aktuálnym pixelom (b), ľavý pixel (a) a pixel diagonálne vľavo hore (c). Pre výpočet predikcií sa používajú nasledujúce vzorce [5] :

c	b
a	X

Tabuľka 4.2: Ukážka umiestnenia pixelov

$$p = a + b - c \tag{4.1}$$

$$pa = |p - a| \tag{4.2a}$$

$$pb = |p - b| \tag{4.2b}$$

$$pc = |p - c| \tag{4.2c}$$

$$P_X = \begin{cases} a & \text{min (pa,pb,pc) = pa} \\ b & \text{min (pa,pb,pc) = pb} \\ c & \text{ostatný} \end{cases} \tag{4.3}$$

4.4 Prediktor MED

Podobne ako prediktor PAETH, aj prediktor MED (**M**edian **E**dge **D**etector) používa 3 pixely na predikciu. Predikovaná hodnota sa vypočíta pomocou nasledujúcich vzorcov [2]:

$$P_X = \begin{cases} min(a,b) & \text{c} \geq max(a,b) \\ min(a,b) & \text{c} \leq min(a,b) \\ a+b-c & \text{ostatný} \end{cases} \tag{4.4}$$

Umiestnenie pixelov a, b, c je podobná ako u PAETH prediktoru (tabuľka 4.2). Prediktor MED sa používa napríklad vo formáte JPEG-LS.

4.5 Prediktor GAP

Prediktor GAP [7] (**G**radient **A**djusted **P**rediction) na predikciu nasledujúceho pixelu používa 7 predchádzajúcich pixelov – tabuľka 4.3. Výpočet predikovanej hodnoty bude závisieť od rozdielu hodnôt VG a HG (Rovnice 4.5 a 4.6). V tabuľke 4.4 sú uvedené rovnice, podľa ktorých sa vypočíta predikovaná hodnota podľa hodnoty rozdielu VG a HG. Hodnota x sa vypočíta podľa rovnice 4.7.

		NN	NNE
	NW	N	NE
WW	W	X	

Tabuľka 4.3: Pozícia pixelov

$$VG = |NW - W| + |NN - N| + |NNE - NE| \tag{4.5}$$

$$HG = |WW - W| + |NW - N| + |N - NE| \tag{4.6}$$

VG-HG	<-80	<-32	<-8	[-8 8]	>8	>32	>80
Prediction	N	$\frac{x+N}{2}$	$\frac{3 \cdot x+N}{4}$	x	$\frac{3 \cdot x+W}{4}$	$\frac{x+W}{2}$	W

Tabuľka 4.4: Predikované hodnoty

$$x = \frac{W+N}{2} + \frac{NE-NW}{4} \tag{4.7}$$

Kapitola 5

Entropické kódovanie

Informačná entropia (Shannonova entropia) je stredná hodnota informácie jedného symbolu. Je zadaná obvykle v bitoch. Kodéry sa snažia dosiahnuť práve entropiu. Entropiu symbolu môžeme vypočítať pomocou vzorca:

$$H_i = -\log_2 p_i \tag{5.1}$$

kde p_i je pravdepodobnosť výskytu symbolu.

Entropické kódovanie je veľmi často používaná metóda pri bezstratovej kompresie. Entropický kodér komprimuje dáta takým spôsobom, že nahradí vstupné symboly zodpovedajúcou sekvenciou, ktoré majú premennú dĺžku. Dĺžka sekvencie závisí od pravdepodobnosti výskytu pôvodných symbolov v dátach tak, že najbežnejšie symboly budú reprezentované s najkratšími sekvenciami.

Existuje mnoho typov entropických kodérov, ale najčastejšie použité sú Huffmanovo kódovanie a aritmetické kódovanie.

5.1 MTF

Move **t**o **F**ront [4] je transformačná metóda, ktorá napomáha ďalšej kompresii. Pracuje s poľom, ktoré obsahuje usporiadané vstupné symboly. Zo vstupu číta symbol a dá na výstup index symbolu v poli. Symbol potom presunie na začiatok poľa. Výhodou metódy je, že transformuje opakujúce sa symboly na nulu a často sa opakujúce znaky budú nahradené s nízkou hodnotou. Tieto hodnoty budú mať väčšiu početnosť a tým je vstup vhodnejší na kódovanie pre entropický kodér.

Dekódovanie prebieha podobne ako kódovanie, len tento krát sú k dispozícii indexy symbolov. Na dekódovanie sa používa rovnaké pole ako pri kódovaní. Na výstup dá symbol, ktorý sa nachádza na daný index (vstup) v poli. Symbol potom presúva na začiatok poľa.

5.2 RLE

RLE (**R**un-**l**ength **E**ncoding) patrí medzi metódy na bezstratovú kompresiu dát. Vychádza z predpokladu, že susedné pixely sú často rovnaké. Výstupom kodéru sú dvojice číslic: počet výskytu symbolov a symbol. Príkladom kompresie:

Vstup	1 1 1 1 8 8 8
Výstup	4 1 3 8

Tabuľka 5.1: Príklad kodéru RLE

Nevýhodou tejto metódy je, že kóduje jeden symbol dvomi znakmi, môže tak pri nevhodnom vstupe až zdvojnásobiť jeho veľkosť. Preto vznikli rôzne varianty [1], ktoré riešia tento problém. Je možné napríklad kódovať len behy s minimálnou dĺžkou 4, ostatné nekomprimovať, a označiť, ktoré čísla označujú opakovanie. Takýmto spôsobom kódujeme behy s minimálnou dĺžkou 4 s tromi znakmi. Označiť opakovanie môžeme napríklad so špeciálnym znakom, ktorý nie je v našej abecede. Príkladom kompresie:
Berme abecedu A={1, 2, 3, 4, 5} a zvoľme symbol pre označenie opakovania napríklad s={0}.

Vstup	1 1 1 1 1 2 3 5 5 5 5 4 4 4
Výstup	0 5 1 2 3 0 4 5 4 4 4

Tabuľka 5.2: Príklad kodéru RLE

5.3 Huffmanovo kódovanie

Huffmanovo kódovanie je pomenované podľa svojho objaviteľa D. A. Huffmana. Je to metóda pre bezstratovú kompresiu dát. Reprezentuje symboly binárnym prefixovým kódom, ktoré nie sú prefixmi kódov iných symbolov. Čím je symbol početnejší, tým bude prefixový kód kratší. Niektoré kódy môžu byť aj dlhšie, ako pôvodný symbol.

Kompresia sa skladá z dvoch častí: prvá fáza je vytvorenie štatistiky podľa početnosti vstupných symbolov. V druhej fáze pomocou tejto štatistiky sa vytvori binárny strom. Listy stromu sú jednotlivé symboly, cesta k listu, ktorá sa skladá z núl a jednotiek, je prefixovým kódom. Pomocou binárneho stromu sa potom zakódujú dáta. Dekompresia pomocou toho istého binárneho stromu dekóduje pôvodné symboly.

Výhodou tejto metódy je rýchla kompresia, dekompresia a jednoduchosť implementácie. Nevýhodou je, že binárny strom musí byť uložený do výstupného súboru.

Huffmanovo kódovanie sa používa hlavne pre kompresiu multimediálnych dát. Je súčasťou kodéra u MP3 a JPEG.

5.4 Aritmetické kódovanie

Podobne ako Huffmanovo kódovanie, aj aritmetické kódovanie [3] je metódou pre bezstratovú kompresiu. Líši sa od Huffmanovho kódovania však s tým, že aritmetický kodér nepriradí kódy vstupným symbolom, ale priradí jeden kód celému vstupnému toku, napríklad jedno reálne číslo z intervalu [0; 1). A tým prekonáva problém prideľovania celočíselných kódov k symbolu.

Interval sa rozdelí podľa pravdepodobnosti výskytu jednotlivých symbolov, a tieto intervaly priradí symbolom (*low*, *high*). Potom sčíta každý znak po jednom, a znižuje sa interval podľa pravdepodobnosti symbolu – čím väčší je interval symbolu, tým menej sa

znižuje interval. Znížený interval zaberie stále viac bitov. Výsledný kód bude jedno číslo z výsledného intervalu.

5.4.1 Príklad kódovania

Berme reťazec "abacb". Pravdepodobnosť symbolu 'a' je 0,4; symbolu 'b' je 0,4 a symbolu 'c' je 0,2. Interval vtedy rozdeľujeme na tri časti:

Symbol	Pravdepodobnosť	Interval
a	0,4	[0; 0,4)
b	0,4	[0,4; 0,8)
c	0,2	[0,8; 1)

Tabuľka 5.3: Symboly, ich pravdepodobnosť a interval

Pre výpočet nových intervalov použijeme nasledujúce vzorce:

$$Range = High - Low \tag{5.2}$$
$$Hight = Low + Range \cdot HV \tag{5.3}$$
$$Low = low \cdot LV \tag{5.4}$$

kde *Low* je začiatok intervalu a *High* je koniec intervalu, *LV* a *HV* sú začiatok a koniec intervalu symbolu.

Teraz už môžeme čítať znaky po jednom, a zmenšiť interval. Prvým znakom je 'a'. To znamená, že pôvodný interval [0; 1) znižujeme na [0; 0,4). Odteraz už pracujeme s novým intervalom. Ďalším znakom je 'b'. interval [0; 0,4) teda znižujeme na [0,16; 0,32). Pre ostatné znaky vypočítame takým istým spôsobom – tabuľka 5.4 a obrázok 5.1

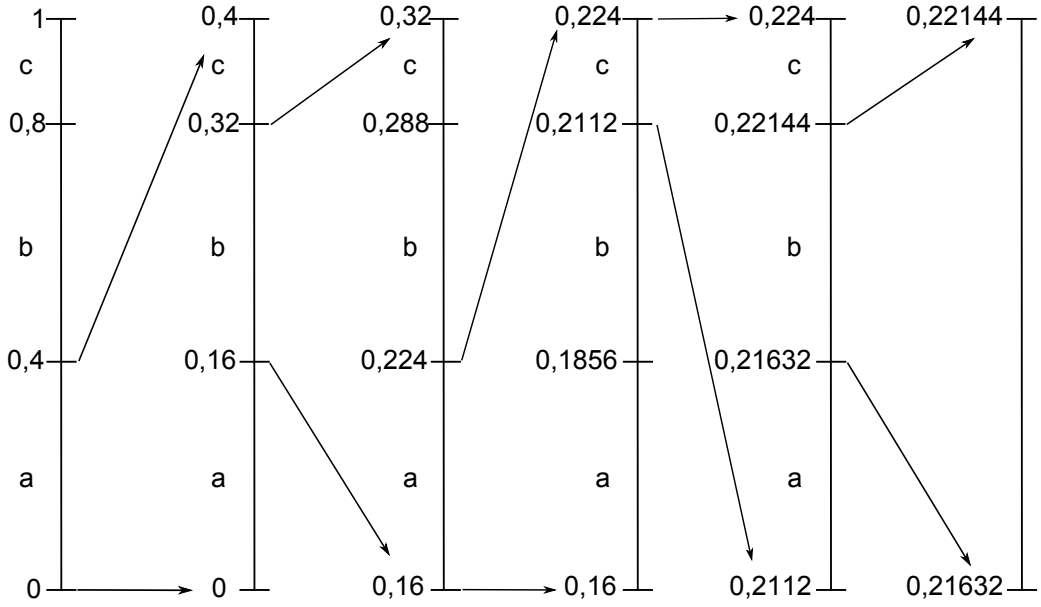
Znak	Range	Low	High	Interval
	1	0	1	[0; 1)
a	1	0	0,4	[0; 0,4)
b	0,4	0,16	0,32	[0,16; 0,32)
a	0,16	0,16	0,224	[0,16; 0,224)
c	0,064	0,2112	0,224	[0,2112; 0,224)
b	0,0128	0,21632	0,22144	[0,21632; 0,22144)

Tabuľka 5.4: Kódovanie

Výsledný kód bude jedno ľubovoľné číslo z intervalu [0,21632; 0,22144).

5.4.2 Príklad dekódovania

Pri dekódovaní potrebujeme poznať symboly a ich intervaly, čo je v tabuľke 5.3. Dekódovanie bude prebiehať nasledovne: berieme výsledný kód. Určíme do ktorého intervalu patrí. Podľa intervalu určíme symbol. Prepočítame kód.



Obrázok 5.1: Postup kódovania

Pre prepočítanie kódu použijeme nasledujúci vzorec:

$$kod = \frac{kod - LV}{HV - LV} \tag{5.5}$$

kde *kod* je výsledný kód, *LV* a *HV* sú začiatok a koniec intervalu symbolov. Tieto kroky opakujeme, kým nedekódujeme všetky symboly.
Dekódovanie predchádzajúceho príkladu je nasledovné:

Kód	Interval	Symbol	Nový kód
0,22	[0; 0,4)	a	(0,22-0)/0,4=0,55
0,55	[0,4; 0,8)	b	(0,55-4)/0,4=0,375
0,375	[0; 0,4)	a	(0,375-0)/0,4=0,9375
0,9375	[0,8; 1)	c	(0,9375-0,8)/0,2=0,6875
0,6875	[0,4; 0,8)	b	

Tabuľka 5.5: Dekódovanie

Veľkou nevýhodou tohto výpočtu je, že čím viac znakov sme zakódovali, čísla, s ktorými pracujeme, sú stále väčšie. Pretože počítače pracujú len s reálnymi číslami s obmedzenou presnosťou, keď máme veľký počet vstupných symbolov, tak nám táto presnosť nebude stačiť. Preto sa používa takzvané škálovanie (sekcie 5.4.4 a 5.4.5).

5.4.3 Celočíselná implementácia

Práca s reálnymi číslami je pre procesory oveľa náročnejšia a aj dlhšia, ako práca s celými číslami, preto vo väčšine prípadov v aritmetickom kódovaní sa namiesto reálnych čísiel používajú celé čísla.

Kódovanie

Výsledkom kódovania teda nie je jedno reálne číslo, ale sekvencia bitu. Ďalší rozdiel medzi celočíselnou implementáciou a implementáciou pomocou reálnych čísiel je, že u celočíselnej implementácii sa nepoužíva pravdepodobnosť symbolov, ale symboly majú priradený jeden interval, ktorý reprezentuje ich početnosť. Dolná hranica intervalu (*LowCount*) je súčtom početnosti všetkých predchádzajúcich symbolov a horná hranica (*HighCount*) je *LowCount* zvýšená o početnosti symbolu.

$$LowCount = \sum_{i=0}^{n-1} Count(i) \tag{5.6}$$

$$HighCount = LowCount + Count(n) \tag{5.7}$$

Kódovanie prebieha nasledovne:

Inicializujeme *Low* na nulu a *High* na samé jednotky (binárne). Nesmieme však používať všetky bity, lebo dôjde k pretečeniu. Počas kódovania sa však tieto hodnoty menia. Na výpočty môžeme používať nasledujúce vzorce:

$$Range = High - Low + 1 \tag{5.8}$$

$$High = Low + Range \cdot \left[\frac{HighCount(n)}{N}\right] - 1 \tag{5.9}$$

$$Low = Low + Range \cdot \left[\frac{Lowcount(n)}{N}\right] \tag{5.10}$$

kde n je vstupný symbol a N je počet všetkých symbolov, $\left[\frac{a}{b}\right]$ je celočíselné delenie.

Výslednú sekvenciu bitov hlavne dostaneme z E1, E2 a E3 škálovania (sekcie 5.4.4 a 5.4.5). Posledné bity výstupu sú nastavené tak, aby bolo možné z toho zistiť interval na konci kódovania. Na to nám stačia 2 bity. Nastavíme 0 a 1, keď *Low* leží v prvej štvrtine a keď ešte máme neriešené E3 škálovanie, tak musíme ešte pridať vhodné množstvo jednotkových bitov. V opačnom prípade nastavíme 1 (nuly doplní dekodér).

Dekódovanie

Pri dekódovaní nepoužívame celý kód naraz, ale čítame ho po bitoch. Preto je možné začínať dekódovanie ešte pred ukončením zakódovania. Potrebujeme naraz toľko bitov z kódu, koľko bitov používame pri dekódovaní/kódovaní pre premenné *High* a *Low*. Môžeme to využiť napríklad pri posielaní komprimovaných dát cez internet. Dekódovanie symbolov prebieha nasledovne: pred začatím dekódovania čítame do jednej premennej (*Buffer*) príslušný počet bitov z kódu a inicializujeme premenné *Low* na nulu, *High* na samé jednotky (binárny). Zistíme jeden symbol:

```
n = prvý symbol;
while ( HighRange(n) <= (Buffer-Low)/(Range/N) )
    n=ďalší symbol;
```

Po zistení symbolu, symbol pošleme na výstup, upravíme *Low* a *High* podľa vzorcov 5.9 a 5.10, a prevedieme škálovanie na *High*, *Low* a *Buffer*. Pri škálovaní posledný bit *Bufferu* bude ďalší bit z kódu. Keď už nie sú v kóde ďalšie bity (koniec kódu), tak pridáme nuly. Keď sme skončili so škálovaním, dekódujeme ďalší symbol. Pri dekódovaní sa stáva problémom, že dekodér nevie, koľko symbolov má dekódovať. Riešenie tohto problému je napríklad to, že použijeme jeden znak na EOF, alebo uložíme počet znakov, a pri dekódovaní dekódujeme presný počet symbolov.

5.4.4 E1 a E2 škálovanie

Keď sa pri kódovaní dostaneme do nejakého intervalu, nikdy z neho nevyjdeme. Túto vlastnosť využijeme u E1 a E2 škálovania. Keď sa zhodujú najvýznamnejšie bity (MSB) na začiatku a konci intervalu (hodnôt *Low* a *High*), už sa nebudú meniť, takže je možné zapísať na výstup a môžeme upraviť interval tak, že každý bit posunujeme o jednu pozíciu vľavo, a posledný bit nastavíme na 0 pre začiatok intervalu (*Low*) a na 1 pri konci intervalu (*High*). S tým zväčšíme interval.

E1– prvý bit (MSB) *Low* a *High* je 0:
Low: 0xxxxxxx ⇒ xxxxxxx0,
High: 0xxxxxxx ⇒ xxxxxxx1
a na výstup pošleme nulu.

E2 – prvý bit (MSB) *Low* a *High* je 1:
Low: 1xxxxxxx ⇒ xxxxxxx0,
High: 1xxxxxxx ⇒ xxxxxxx1
a na výstup pošleme jednotku.

5.4.5 E3 škálovanie

Avšak E1 a E2 škálovania nie sú postačujúce, pretože nevyriešia problém, keď *Low* a *High* majú rôzne prvé bity (MSB) a interval je veľmi malý:

Low: 01111111; *High*: 10000000 .

Keď *Low* a *High* majú rôzne MSB (*Low* ma nulu, *High* má jednotku) a druhý najvýznamnejší bit u *Low* je 1 a u *High* je 0, musíme prevádzať E3 škálovanie. E3 škálovanie pracuje podobne ako E1 a E2, ale pri E3 škálovaní najvýznamnejšie bity sa nemenia, len nasledujúci bit sa odstráni, ostatné bity sa posunú doľava, a posledný bit sa nastaví na 0 u *Low* a na 1 u *High*.

E3: – prvý bit (MSB) *Low* je 0, *High* je 1, druhý bit *Low* je 1, *High* je 0

Low: 01xxxxxx ⇒ 0xxxxxx0,
High: 10xxxxxx ⇒ 1xxxxxx1

Na výstup nepošle žiadny znak, len zapamätá, koľkokrát za sebou bolo použité E3 škálovanie, a pri nasledujúcom E1 alebo E2 škálovaní pošle správny počet (koľkokrát bolo prevedené E3 škálovanie) bitov, opačné ako bit poslané výsledkom E1 alebo E2 škálovania, na výstup (jednotky keď po E3 nasleduje E1 a nuly keď E2).

5.5 PPM kodér

Prediction by **P**artial **M**atching [11] [8] je štatistická metóda kompresie dát. Pri kódovaní sa využíva aj kontext symbolu. Pravdepodobnosť symbolu teda závisí aj na kontexte, nie len na symbole. Rád kontextu je daný počtom predchádzajúcich symbolov, ktoré využíva pre predikciu. Na kódovanie sa používa aritmetický kodér, ktorému PPM kodér pošle pravdepodobnosť symbolu. Na určenie pravdepodobnosti symbolov môže použiť statický alebo adaptívny model.

Statický model používa vždy rovnaké pravdepodobnosti na daný symbol v závislosti na jeho kontext. Tieto pravdepodobnosti sú určené z štatistiky rozsiahlych dát. Nevýhodou statického modelu je, že vstupné dáta môžu byť štatisticky odlišné od dát použitých na určenie pravdepodobnosti. Môže sa stáť, že niektorý symbol ma nulovú pravdepodobnosť – symbol s daným kontextom sa neobjavil v dátach použitých na vytvorenie pravdepodobností. Tento problém sa dá riešiť napríklad pridelením malých pravdepodobností symbolov, ktoré sa nevyskytli.

Pri adaptívnom modely sa pravdepodobnosti symbolov zmenia počas kódovania podľa načítaných symbolov. S adaptívnym modelom sa dajú dosiahnuť lepšie výsledky, pretože pravdepodobnosti symbolov zodpovedajú aktuálnym pravdepodobnostiam, ale jeho implementácia je zložitejšia a kódovanie pomalejšie ako pri statickom modely.

Hlavným problémom realizácie PPM kodéra je udržovanie dátovej štruktúry. Potrebujeme takú štruktúru, do ktorej sa dá uložiť všetky symboly s kontextmi a dá sa v nej rýchlo hľadať. Takou štruktúrou je napríklad trie.

5.5.1 Trie

Slovo trie sa pochádza z anglického slova retrieval, čo znamená získavanie, vyhľadávanie. Trie je strom, ktorý sa vetví s každým symbolom kontextu. Uzly reprezentujú jednotlivé symboly, cesta k symbolu udáva kontext symbolu. Každý uzol obsahuje 2 údaje: symbol (kód symbolu) a početnosť symbolu (v danom kontexte). Trie rastie do šírky, ale do hĺbky nie. Maximálna hĺbka trie je n+1, kde n je rád kontextu. Úroveň 1 v trie obsahuje jeden uzol pre každý prečítaný symbol. Úroveň v trie udáva rád kontextov.

5.5.2 Kódovanie

Kodér číta jeden symbol zo vstupu. Potom kodér vyhľadá symbol s jeho kontextom rádu n, a zistí jeho pravdepodobnosť. Keď symbol v danom kontexte sa ešte neobjavil, kodér prepína na kratší kontext. Rád kontextu klesá až do rádu -1, kým symbol nenájde. Keď symbol nenašiel, to znamená, že sa symbol vyskytol prvý krát. Vtedy je zakódovaný s pevnou pravdepodobnosťou: 1/(*Veľkosť abecedy*). Keď je už k dispozícii pravdepodobnosť symbolu, PPM kodér zavolá algoritmus aritmetického kodéra so získanou pravdepodobnosťou, ktorý symbol zakóduje.

5.5.3 Dekódovanie

Dekodér na rozdiel od kodéra nemá k dispozícii nasledujúci symbol. Preto nevie, kedy má prepnúť na kratší kontext. Tento problém je riešený tak, že je vyhradený jeden symbol abecedy, ktorý označuje prepnutie na kratší kontext. Tento symbol sa volá escape symbol. Keď kodér prepína na kratší kontext, zakóduje symbol escape, aby dekodér vedel, kedy má prepínať na kratší kontext. Pravdepodobnosť escape symbolu je treba zvoliť tak, aby mal vysokú prioritu, keď sa používa často, a čo najnižšiu, keď sa používa málokedy.

5.5.4 Vylúčenie

Vylúčenie sa používa, keď sa kodér prepína na kratší kontext. Vtedy môžeme využiť informácie kontextov vyšších rádov pre vylúčenie niektorých symbolov z nižších rádov – symboly, ktoré sa vyskytli po kontextoch vyšších rádov. Tým zvyšujeme pravdepodobnosti symbolov nižších rádov a zlepšujeme kompresiu.

5.5.5 Varianty PPM

Existuje viac variantov PPM kodéru, ktoré sa väčšinou líšia len v spôsobe prideľovania pravdepodobnosti escape symbolu či symbolov.

- U PPMC pravdepodobnosť symbolu escape je $s/(s+n)$, kde s je počet symbolov a n je súčet početnosti symbolov v danom kontexte. Keď sa po danom kontexte vyskytol vždy rovnaký symbol, to naznačuje, že i v budúcnosti bude za tým kontextom rovnaký symbol. Preto escape symbol môže mať nízku prioritu, pretože sa používa málokedy. V opačnom prípade, keď sa po kontexte vyskytol vždy iný symbol, symbol escape má mať vysokú prioritu, pretože sa používa často.

- PPMA prideľuje escape symbolu pravdepodobnosť $1/(n+1)$, ako by mal escape symbol vždy početnosť 1. Ostatným symbolom prideľuje pravdepodobnosť p/n, kde p je početnosť symbolu. Súčet pravdepodobnosti symbolov bez symbolu escape sa rovná 1.

- PPMB je podobný metóde PPMC, ale symbolom prideľuje pravdepodobnosť až po druhom výskyte v danom kontexte.

- PPMP berie výskyt symbolov ako oddelené Poissonovské procesy.

- PPMX používa pre pravdepodobnosť symbolu escape približnú hodnotu t/n (prvý výraz súčtu). Keď t má hodnotu 0 alebo n, PPMX sa modifikuje na PPMXC a používa rovnakú pravdepodobnosť ako PPMC.

- PPM* sa snaží zvyšovať rád kontextu až do nekonečna. S touto metódou vieme dosiahnuť až o 6% lepšiu kompresiu, ako s PPMC, ale potrebuje inú dátovú štruktúru a viac počítačových zdrojov.

- PPMZ sa zameriava na odstránenie chýb a nedostatkov PPM kodéra a dosahuje veľmi dobré výsledky.

Kapitola 6

Implementácia knižnice

Knižnica je implementovaná v jazyku C++. Pre práce s obrazmi (načítanie, a zápis) sa používa knižnica OpenCV. Sú implementované tri entropické kodéry: aritmetické kodéry – jeden statický, jeden adaptívny, a PPM kodér (PPMC). Je možnosť vybrať si z viacerých prediktorov a farebných modelov, a voľne ich kombinovať. Ďalej je implementovaný RLE kodér a MTF transformácia, ktoré taktiež môžu byť súčasťou postupu kompresie.
Kompresia obrazu prebieha nasledovne (obrázok 6.1):

Obrázok 6.1: Postup kompresie obrazu

Pri dekompresii je postup práve opačný (obrázok 6.2):

Obrázok 6.2: Postup dekompresie obrazu

6.1 Načítanie a zápis obrazov

Na načítanie a zápis obrázkových súborov sa používa knižnica OpenCV. Funkcie a štruktúry OpenCV sa používajú iba funkcie *read_pic()* a *save_pic()*. Obrázok sa načítava pomocou funkcie *imread()*. Pixely obrazu a ďalšie informácie sa ukladajú do štruktúry *pict*. Komponenty pixelov sú uložené do dvojdimenzionálneho vektoru – každý kanál zvlášť. Program potom pracuje s touto štruktúrou. Pri zápise do obrazu (dekódovanie) zo štuktúry *pict* sa vytvorí štruktúra, ktorú pomocou funkcie *imwrite()* uloží ako obrázok.

6.2 Prevod farebného modelu

Keď nie je zadaný prevod farebného modelu ako parameter, pre kompresiu (a dekompresiu) sa používa farebný model RGB. Okrem farebného modelu RGB program podporuje tri ďalšie farebné modely – LYUV, LOCO-I a ORCT. Pred kódovaním (dekódovaním) však musí program previesť obraz z modelu RGB do vybraného farebného modelu. Pre každý model sú implementované dve funkcie – jedna na prevod z RGB a jedna zo zvoleného modelu späť do RGB tak, aby nedošlo k strate dát.

6.3 Predikcia

Na predikciu program používa 3 funkcie. Funkcia *toPred()* je volaná pri kompresii a funkcia *fromPred()* pri dekompresii. Na výpočet predikovanej hodnoty obidve funkcie volajú funkciu *getPred()*, ktorá vráti predikovanú hodnotu podľa zvoleného prediktora. Implementované prediktory sú GAP, PAETH, MED a 3 lineárne prediktory. Keď nie je možné použiť zvolený prediktor (krajné body), je použitý iný prediktor.

6.4 MTF transformácia

Na začiatku transformácie vytvorime jeden vektor, ktorý obsahuje všetky možné hodnoty symbolov od najnižšej po najvyššiu. Potom každý symbol sa vyhľadáva vo vektore, symbol bude nahradený jeho pozíciou a vo vektore nájdený symbol sa posúva na začiatok vektora. Pri dekódovaní namiesto hodnôt symbolu máme k dispozícii pozície vo vektore. Podľa tých vyhľadáva hodnoty symbolov, a podobne ako u kódovania posúva nájdený symbol na začiatok vektora.

6.5 RLE kódovanie

RLE kodér číta symboly po jednom, a porovná s predchádzajúcim symbolom. Keď sa symboly zhodujú, tak zvyšuje premennú, ktorá počíta za seba nasledujúce zhodné symboly. V opačnom prípade, keď počet zhodných symbolov je väčší ako 3, tak zapíše namiesto symbolov špeciálny symbol, ktorý má hodnotu väčšiu, ako najvyšší symbol, počet opakovaní a symbol. Keď počet opakovaní je menší, tak zapíše symboly nezmenene.

Dekodér keď nájde špeciálny symbol, tak číta počet opakovaní a symbol a zapíše správny počet symbolov. Keď nasledujúci symbol nie je špeciálnym symbolom, tak zapíše bez zmeny.

6.6 Triedy arithmetic_encoder a arithmetic_decoder

Všetky tri implementované kodéry používajú na kódovanie a dekódovanie tie dve triedy.

Úlohou triedy *arithmetic_encoder* je zakódovať vstupné symboly po jednom a výstup vypísať do výstupného súboru. Má metódu *writes()*, ktorá vypíše ďalšie dáta do výstupného súboru. Kódovanie jedného symbolu je implementované v metóde *encode()*. Ako parameter čaká štruktúru, v čom je *LowCount* a *HighCount* symbolu a *Sum* – počet symbolov.

Trieda *arithmetic_decoder* sa používa na dekódovanie súboru po symboloch. Obsahuje metódu *reads()*, čo číta dáta zo súboru uložené metódou kodéru *writes()*. Metóda *decode()*,

ktorá dekóduje jeden symbol, čaká ako parameter štruktúru obsahujúci *LowCount* a *HighCount* symbolu a *Sum* – počet symbolov. Keď tento symbol bol zakódovaný, vráti 1, inak vráti -1.

6.7 Statický aritmetický kodér

Statický aritmetický kodér kóduje spolu všetky farebné kanály. Na začiatku zistí najnižšie a najvyššie číslo (symbol) z vektora pomocou funkcie *getminm()*. Zavolaním funkcie *hgram()* dostáva početnosť jednotlivých symbolov. Z tých početnostiach potom vypočíta *LowCount* a *HighCount* symbolu a *Sum*. Pomocou metódy enkódera zapíše do výstupného súboru potrebné dáta – šírku, výšku a počet farebných kanálov obrazu, najnižší a najvyšší symbol, keď bola použitá MTF transformácia, tak ešte aj originálne najnižšie symboly pre každý kanál a naposledy početnosť jednotlivých symbolov. Potom kóduje všetky symboly pomocou metódy *encode()* triedy *arithmetic_encoder*.

Pri dekódovaní najprv číta uložené dáta. Pomocou metódy *decode()* triedy *arithmetic_decoder* sa dekódujú jednotlivé pixely po jednom.

6.8 Adaptívny aritmetický kodér

Adaptívny aritmetický kodér funguje podobne ako statický, tiež používa metódy *arithmetic_encoder* pre uloženie dát a pre kódovanie. Kóduje však každý farebný kanál zvlášť. Na začiatku kódovania každý symbol má rovnakú pravdepodobnosť – 1/(*Veľkosť abecedy*). Kodér číta symboly po jednom, zakóduje ich, potom zvyšuje početnosť symbolu o 1. Dekodér funguje podobne – zvyšuje pravdepodobnosť dekódovaných symbolov.

6.9 PPM kodér

PPM kodér podobne ako adaptívny aritmetický kodér kóduje farebné kanály obrazu zvlášť, a tiež používa triedu *arithmetic_encoder*. Na udržovanie dátových štruktúr používa triedu *trie*.

V triedu *trie* každý symbol je reprezentovaný s jedným uzlom. Každý uzol obsahuje ukazovateľ na nasledujúci uzol, ukazovateľ na jedného potomka a ukazovateľ na ten istý symbol s kratším kontextom, hodnotu symbolu a počet výskytov. Úroveň v trie udáva rád kontextu.

Pri kódovaní najprv vyhľadá v trie symbol s kontextom. Pomocou metódy *encode() arithmetic_codera* zakóduje vrátené dáta. Keď symbol nenašiel, všetky symboly v danom kontexte sú uložené do množiny. Tieto symboly sú ignorované v kontextoch nižších rád. Potom kodér prepína na kratší kontext. Množina je vyprázdnená po zakódovaní symbolu. Vylučovanie sa používa podobne aj pri dekódovaní. Keď sa symbol vôbec nevyskytoval pred tým, tak zakóduje s pravdepodobnosťou 1/(*Veľkosť abecedy*). Po zakódovaní symbolu, symbol je vložený do trie.

Pri dekódovaní sa používa metóda *searchd()* triedy *trie* a metóda *decode()* triedy *arithmetic_decoder*. Pomocou kontextu symbolu zistíme symbol: použijeme metódu *searchd()*, tá vráti *LowCount*, *HighCount* symbolu a súčet počtov výskytov symbolov, čím je potom volaná meóda *decode()*. Podľa vrátenej hodnoty dekódera pokračuje vyhľadávanie, prepína na kratší kontext, alebo pomocou metódy *insert()* vloží symbol do trie a začne dekódovať nasledujúci symbol.

6.9.1 Trieda trie

Trie sa používa na udržovanie datovej štruktúry pre každý kontext. Obsahuje metódy pre prácu s trie.

Pre vloženiu do trie sa používa metóda *insert()*, čo vytvorí uzol v trie, keď sa symbol ešte nevyskytol s daným kontextom, alebo upraví počet výskytov pre daný kontext a všetkých jeho kratších kontextov.

Na vyhľadávanie v trie sú implementované 2 metódy – jedna pre kódovanie (*search()*) a jedna pre dekódovanie (*searchd()*).

Metóda *search()* hľadá v trie symbol. Keď je symbol nájdený, metóda vráti *LowCount*, *HighCount* symbolu a súčet počtov výskytov symbolov v danom kontexte. Keď sa symbol nevyskytol ešte po danom kontexte, tak sa vráti *LowCount* a *HighCount* špeciálneho symbolu, ktorý označuje, že symbol sa nenašiel, spolu so súčtom počtov výskytov symbolov.

Metóda *searchd()* sa vráti *LowCount*, *HighCount* nasledujúceho symbolu, symbol a súčet počtov výskytov symbolov po daného kontextu.

6.10 Návod k použitiu

Pri spustení programu užívateľ zadá parametre, ktoré ovplyvňujú beh programu. Podporované parametre sú uvedené v tabuľke 6.1.

Prvé tri parametre sú povinné (v správnom poradí), ostatné sú voliteľné. Keď nie je zadaný farebný model, používa sa model RGB. Je možné aj porovnať dva obrázky – pomocou parametre -compare <file1> <file2>, kde <file1> a <file2> sú obrázky.
Keď niektorý povinný parameter chýba, alebo je zadaný zlý parameter, program vypíše nápoveď a ukončí sa.

Parameter	Možné hodnoty	Význam
-encode=<hodnota>	i	Štatický aritmetický kodér
	a	Adaptívny aritmetický kodér
	p	PPM kodér
-decode=<hodnota>	i	Štatický aritmetický kodér
	a	Adaptívny aritmetický kodér
	p	PPM kodér
-input=<hodnota>	Pri kódovaní	Obraz ktorý bude zakódovaný
	Pri dekódovaní	Zakódovaný obraz
-output=<hodnota>	pri kódovaní	Výstup kódovania
	pri dekódovaní	Výstupný obraz dekodéra
-pred=<hodnota>	1	Lineárny prediktor prvého rádu
	2	Lineárny prediktor druhého rádu
	3	Lineárny prediktor tretieho rádu
	p	Prediktor PAETH
	m	Prediktor MED
	g	Prediktor GAP
-cmodel=<hodnota>	Lyuv	Použitie farebneho modelu LYUV
	LocoI	Použitie farebneho modelu LOCO-I
	Orct	Použitie farebneho modelu ORCT
-context=<hodnota>	Číslo	Udáva veľkosť kontextu u PPM kodéra
-rle		Run-length encoding
-mtf		Move to front transformácia

Tabuľka 6.1: Parametre programu

Kapitola 7

Dosiahnuté výsledky

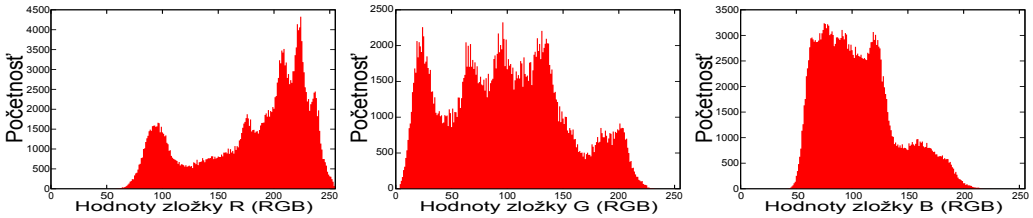
V tejto kapitole budú popísané dosiahnuté výsledky a porovnávanie prediktorov, farebných modelov a kodérov. Porovnávanie farebných modelov a prediktorov je založené na referenčnom obrázku lena.bmp. Výsledky pre iné obrazy môžu byť úplne odlišné.

Vplyv prevodu farebných modelov na početnosti hodnôt pixelov a efektívnosť prediktorov sa dá vizualizovať pomocou histogramov. Histogram je graf, ktorý na ose x má hodnoty (zložku) pixelov a na ose y ich početnosti. Histogram ukazuje rozdelenie pixelov. Pre entropické kódovanie je najvhodnejšia čím menšia abeceda a čím rozdielnejšie početnosti symbolov (jeden symbol s čím väčšou početnosťou). Veľká abeceda s podobnými pravdepodobnosťami symbolov je najmenej priaznivá pre kompresiu (náhodné dáta).

7.1 Porovnávanie farebných modelov

Ako sme spomenuli v sekcii 3.2, farebný model RGB nie je vhodný na kompresiu. V tejto sekcii porovnáme implementované farebné modely pomocou histogramu. Obecne platí, že čím menšiu abecedu máme s čím väčšou početnosťou jednotlivých symbolov, tým lepšie sa dá obraz zakódovať.

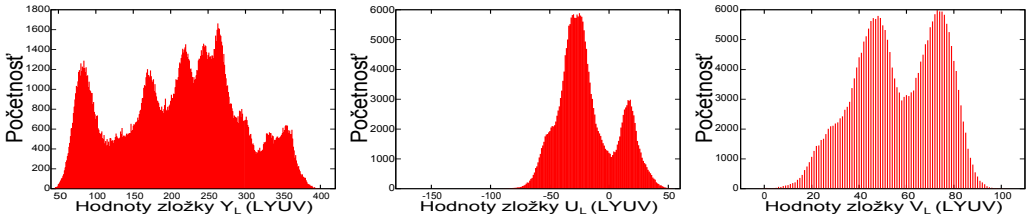
Na nasledujúcich histogramoch 7.1 - 7.4 môžeme vidieť početnosti zložiek pixelov referenčného obrázka lena.bmp v implementovaných farebných modeloch.



(a) Zložka R (b) Zložka G (c) Zložka B

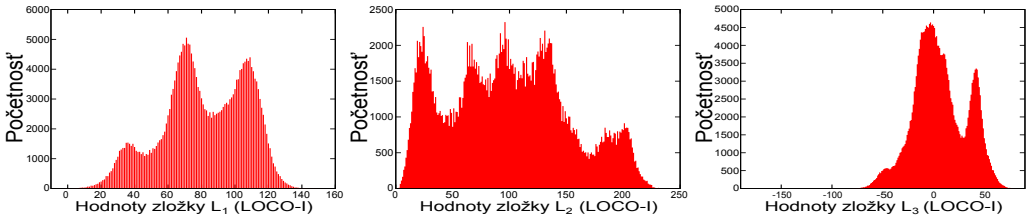
Obrázok 7.1: Histogram početnosti zložiek pixelov (RGB)

Z histogramov 7.1 - 7.4 sa dá vidieť, že vo farebných modeloch LYUV, LOCO-I a ORCT niektoré symboly majú oveľa väčšiu početnosť ako v modely RGB, preto aj keď je abeceda v niektorých prípadoch väčšia, môžeme dosiahnuť s tými modelmi lepší kompresný pomer.



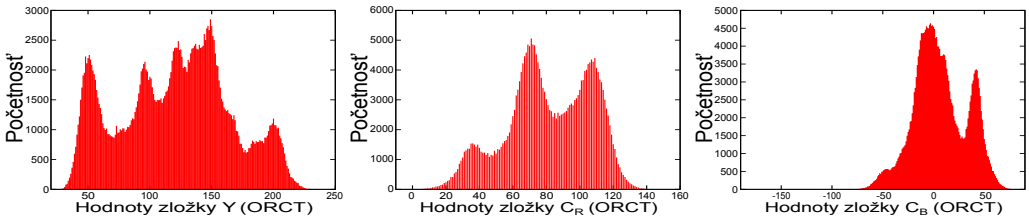
(a) Zložka Y_L (b) Zložka U_L (c) Zložka V_L

Obrázok 7.2: Histogram početnosti zložiek pixelov (LYUV)



(a) Zložka L_1 (b) Zložka L_2 (c) Zložka L_3

Obrázok 7.3: Histogram početnosti zložiek pixelov (LOCO-I)

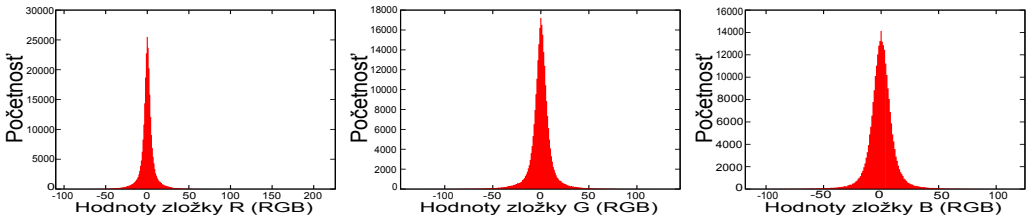


(a) Zložka Y (b) Zložka C_R (c) Zložka C_B

Obrázok 7.4: Histogram početnosti zložiek pixelov (ORCT)

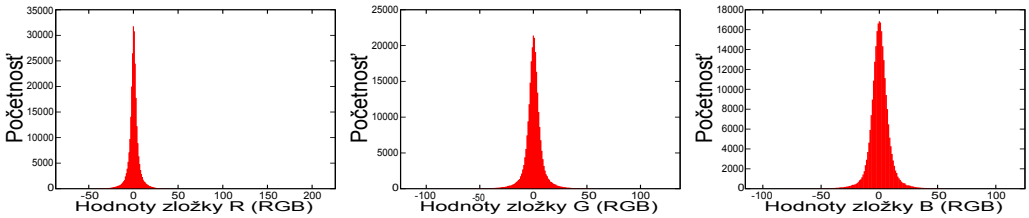
7.2 Porovnávanie prediktorov

V tejto sekcie porovnáme implementované prediktory – lineárne prediktory (číslo označuje rád prediktoru),GAP, PAETH a MED. Ich vplyv na obrázok Lena.bmp v modely RGB je vizualizovaný pomocou histogramov (obrázky 7.5 – 7.10).



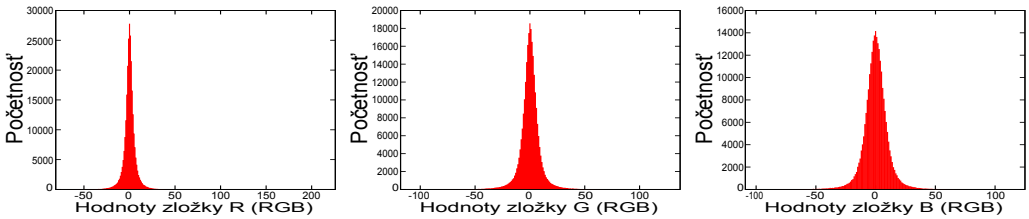
(a) Zložka R (b) Zložka G (c) Zložka B

Obrázok 7.5: Histogram početnosti zložiek pixelov (RGB) po predikcii (lin. pred. 1. rádu)



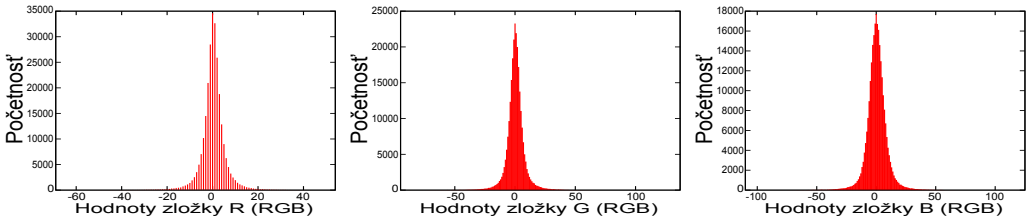
(a) Zložka R (b) Zložka G (c) Zložka B

Obrázok 7.6: Histogram početnosti zložiek pixelov (RGB) po predikcii (lin. pred. 2. rádu)



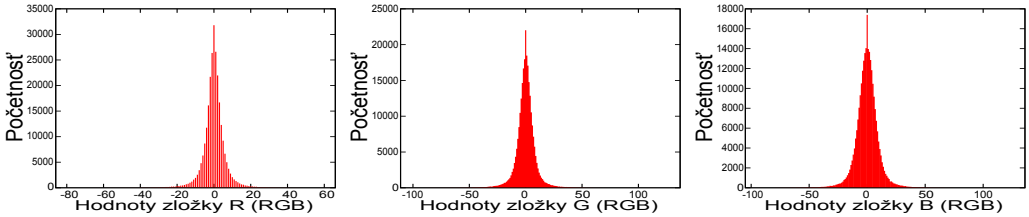
(a) Zložka R (b) Zložka G (c) Zložka B

Obrázok 7.7: Histogram početnosti zložiek pixelov (RGB) po predikcii (lin. pred. 3. rádu)



(a) Zložka R (b) Zložka G (c) Zložka B

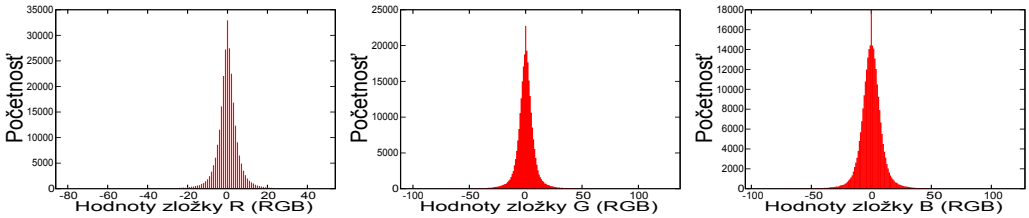
Obrázok 7.8: Histogram početnosti zložiek pixelov (RGB) po predikcii (GAP)



(a) Zložka R (b) Zložka G (c) Zložka B

Obrázok 7.9: Histogram početnosti zložiek pixelov (RGB) po predikcii (PAETH)

V tabuľkách 7.1, 7.2, 7.3 a 7.4 sú porovnávané jednotlivé prediktory a ich vplyv na kompresiu (s adaptívnym aritmetickým kodérom). V tabuľkách sú uvedené rozsahy symbolov jednotlivých farebných kanálov, veľkosť komprimovaného obrázku a kompresný pomer. Pôvodný veľkosť obrázku je 768.1KB. Ako môžeme vidieť, každý prediktor má veľký vplyv



(a) Zložka R (b) Zložka G (c) Zložka B

Obrázok 7.10: Histogram početnosti zložiek pixelov (RGB) po predikcii (MED)

na kompresiu. V ďalších sekciách už používame iba nelineárne prediktory – GAP, PAETH a MED. Na dôkladnejšie porovnávanie slúžia tabuľky A.1 až A.6.

Prediktor	Rozsah symbolu			Veľkosť súboru (KB)	Kompresný pomer
	kanál 1	kanál 2	kanál 3		
bez prediktoru	8 – 225	3 – 248	54 – 255	698,6	0,91
1	-118 – 125	-146 – 145	-110 – 226	501,3	0,65
2	-115 – 125	-124 – 197	-87 – 226	461,8	0,60
3	-109 – 125	-118 – 137	-81 – 226	477,9	0,62
GAP	-110 – 125	-95 – 137	-69 – 226	446,7	0,58
PAETH	-105 – 136	-111 – 137	-86 – 226	468,0	0,61
MED	-105 – 125	-111 – 137	-86 – 226	461,9	0,60

Tabuľka 7.1: Porovnávanie prediktoru (RGB)

Prediktor	Rozsah symbolu			Veľkosť súboru (KB)	Kompresný pomer
	kanál 1	kanál 2	kanál 3		
bez prediktoru	42 – 418	-179 – 59	-10 – 109	665,9	0,87
1	-214 – 276	-84 – 105	-53 – 64	484,3	0,63
2	-177 – 276	-102 – 62	-43 – 64	449,1	0,58
3	-167 – 276	-102 – 62	-43 – 64	469,5	0,61
GAP	-167 – 276	-84 – 104	-42 – 64	440,3	0,57
PAETH	-147 – 276	-98 – 97	-39 – 64	461,6	0,60
MED	-160 – 276	-90 – 118	-49 – 64	456,0	0,59

Tabuľka 7.2: Porovnávanie prediktoru (LYUV)

Prediktor	Rozsah symbolu			Veľkosť súboru (KB)	Kompresný pomer
	kanál 1	kanál 2	kanál 3		
bez prediktoru	-10 – 157	3 – 248	-185 – 85	667,1	0,87
1	-78 – 89	-146 – 145	-95 – 119	489,6	0,64
2	-64 – 89	-124 – 137	-114 – 73	455,3	0,59
3	-66 – 89	-118 – 137	-95 – 119	476,9	0,62
GAP	-55 – 89	-95 – 137	-110 – 111	447,9	0,58
PAETH	-68 – 89	-111 – 137	-104 – 132	469,5	0,61
MED	-65 – 89	-111 – 137	-102 – 118	464,0	0,60

Tabuľka 7.3: Porovnávanie prediktoru (LOCO-I)

Prediktor	Rozsah symbolu			Veľkosť súboru (KB)	Kompresný pomer
	kanál 1	kanál 2	kanál 3		
bez prediktoru	27 – 241	-10 – 157	-185 – 85	660,0	0,86
1	-124 – 156	-78 – 89	-95 – 119	480,8	0,62
2	-100 – 156	-64 – 89	-114 – 73	446,0	0,58
3	-92 – 156	-66 – 89	-95 – 119	467,3	0,61
GAP	-78 – 156	-55 – 89	-110 – 111	438,0	0,57
PAETH	-90 – 156	-68 – 89	-104 – 132	459,6	0,60
MED	-90 – 156	-65 – 89	-102 – 118	454,1	0,59

Tabuľka 7.4: Porovnávanie prediktoru (ORCT)

7.3 Porovnávanie kodérov

V tejto sekcii sa venujeme porovnávaniu kodérov. Ako už bolo spomenuté, tri kodéry sú implementované, ktoré majú svoje výhody a nevýhody. Najrýchlejší kodér z troch implementovaných je statický aritmetický kodér. Jeho nevýhodou je, že uloží do súboru aj tabuľku symbolov s početnosťami, a že musí vstupné symboly čítať dvakrát. Raz keď počíta početnosti a raz pri kódovaní. Tieto nevýhody rieši adaptívny aritmetický kodér. Početnosť symbolov nie je konstantná, ale po kódovaní jednotlivých symbolov sa stále mení, preto nie je treba ani uložiť tabuľku do výstupného súboru, ani čítať symboly viackrát. S adaptívnym aritmetickým kodérom boli dosiahnuté trošku lepšie výsledky, ako statickým, ale najlepšie výsledky boli dosiahnuté PPM kodérom. Nevýhodou PPM kodéra však je, že je pomalší ako aritmetické kodéry, a potrebuje viac pamäti (s čím väčším kontextom pracujeme, tým viac pamäti potrebujeme).

7.4 Výsledky kódovania

Dosiahnuté výsledky budú ukázané na 14 obrázkoch. V tabuľke 7.5 sú uvedené testovacie obrázky, ich veľkosť vo formáte BMP a PNG (V. BMP a V. PNG), kompresný pomer PNG (K.p. PNG) a informácie o zakódovanom obrázku, s ktorými boli dosiahnuté najlepšie výsledky na daný obrázok – veľkosť zakódovaneho obrázku pomocou knižnice (V. z. o.) a jeho kompresný pomer (K.p.); použitý farebný model (F.m), použitý prediktor a použitý kodér.

Všetky výsledky kódovania sú uvedené v tabuľkách A.1 - A.6. Z tabuľky 7.5 sa dá odvodiť, že z implementovaných farebných modelov najlepšie výsledky môžeme dosiahnuť s modelom ORCT. Najefektívnejší prediktor pre testovacie obrázky boli GAP (9 krát z 14) a MED (5 krát z 14). Najefektívnejší kodér je PPM kodér s kontextom 1 aj 2, okrem obrázku mandrill.bmp, kde najlepší výsledok dával adaptívny aritmetický kodér. Všetky komprimované testované obrázky majú menšiu veľkosť ako rovnaký obrázok vo formáte PNG, priemerne o 17%.

7.5 RLE a MTF

V tejto sekcie ukážeme, ako sa zmení výsledok kódovania, keď jeho súčasťou bude aj RLE alebo MTF.

V nasledujúcej tabuľke (tabuľka 7.6) sú uvedené dosiahnuté výsledky s používaním RLE a MTF. Použitý farebný model, prediktor a kodér je podobný ako v tabuľke 7.5.

Obraz	V. BMP	V. PNG	K.p. PNG	V. z. o.	K.p.	F.m	Prediktor	Kodér
lena	768,1	463,8	0,60	433,3	0,56	ORCT	GAP	PPM k. 1
mandrill	768,1	612,0	0,80	585,6	0,76	ORCT	GAP	A.A.C.
pepper	768,1	494,5	0,64	475,7	0,62	ORCT	GAP	PPM k. 1
airplane	768,1	414,7	0,54	367,5	0,48	ORCT	GAP	PPM k. 1
BoatsColor	1329,8	712,4	0,54	560,3	0,42	ORCT	MED	PPM k. 2
ZeldaColor	1329,8	688,7	0,52	557,5	0,42	ORCT	MED	PPM k. 2
barbara	1215,1	800,2	0,66	508,8	0,42	ORCT	MED	PPM k. 2
girl	1215,1	676,6	0,56	443,1	0,36	ORCT	MED	PPM k. 2
monarch	1152,1	602,9	0,52	451,8	0,39	ORCT	GAP	PPM k. 2
lenna_full	1002,0	582,2	0,58	446,9	0,45	LOCO-I	MED	PPM k. 1
cablecar	720,1	411,2	0,57	367,2	0,51	ORCT	GAP	PPM k. 1
cornfield	720,1	441,2	0,61	394,9	0,55	ORCT	GAP	PPM k. 1
pens	720,1	409,8	0,57	359,9	0,50	ORCT	GAP	PPM k. 1
soccer	720,1	451,0	0,63	386,5	0,54	ORCT	GAP	PPM k. 1

Tabuľka 7.5: Dosiahnuté výsledky

Obraz	Bez MTF a RLE	K.p.	RLE	K.p.	MTF	K.p	MTF a RLE	K.p.
lena	433,3	0,56	432,9	0,56	475,5	0,62	475,1	0,62
mandrill	585,6	0,76	585,6	0,76	621,4	0,81	621,5	0,81
pepper	475,7	0,62	475,4	0,62	524,4	0,68	524,0	0,68
airplane	367,5	0,48	368,8	0,48	409,7	0,53	410,2	0,53
BoatsColor	560,3	0,42	562,2	0,42	654,3	0,49	655,3	0,49
ZeldaColor	557,5	0,42	558,8	0,42	649,5	0,49	650,1	0,49
barbara	508,8	0,42	510,0	0,42	581,4	0,48	581,0	0,48
girl	443,1	0,36	444,4	0,37	495,7	0,41	495,0	0,41
monarch	451,8	0,39	455,1	0,40	521,9	0,45	524,3	0,46
lenna_full	446,9	0,45	448,9	0,45	516,3	0,52	513,7	0,51
cablecar	367,2	0,51	369,2	0,51	411,9	0,57	412,8	0,57
cornfield	394,9	0,55	395,7	0,55	437,5	0,61	438,4	0,61
pens	359,9	0,50	360,4	0,50	401,7	0,56	401,9	0,56
soccer	386,5	0,54	386,7	0,54	433,6	0,60	433,7	0,60

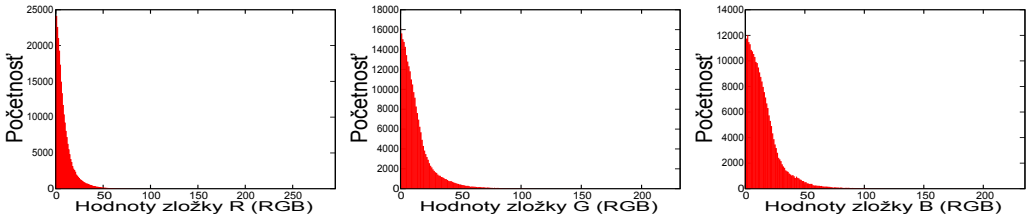
Tabuľka 7.6: Porovnávanie výsledkov použitím MTF a RLE

Ako vidíme z tabuľky 7.6, okrem obrázkov lena.bmp a pepper.bmp použitie RLE kodéra ako časť kódovania neviedlo k lepšej kompresii. To môžeme vysvetliť tým, že obrázky obsahovali len malé behy a aj keď počet symbolov klesol, početnosť nových symbolov je menšia a preto je potrebné na ich zakódovanie viac bitov.

Na testových obrázkoch MTF transformácia významne zvýšila veľkosť zakódovaného súboru. Na obrázku 7.11 sú vizualizované početnosti zložiek pixelu obrázku lena.bmp po predikcii a po MTF transformácii. Keď porovnáme s histogramom bez MTF transformácie (obrázok 7.8) vidíme, že bez MTF transformácie najvyššie početnosti sú oveľa väčšie ako po transformácie. To vedie k nižšej miere kompresie.

7.6 Porovnávanie metód kompresie

Bezstratová kompresia obrazu pozostáva z troch základných krokov: prevod farebného modelu, predikcia a kódovanie, súčasťou kódovania však môžu byť aj ďalšie kroky, ktoré zlepšujú



(a) Zložka R (b) Zložka G (c) Zložka B

Obrázok 7.11: Histogram početnosti zložiek pixelov (RGB) po predikcii (GAP) a po MTF

mieru kompresie. Pretože každý obrázok má iné vlastnosti, je treba nájsť najvhodnejší spôsob na jeho kódovanie. V tejto práci boli vyskúšané rôzne metódy pre kódovania, ktoré dávali rôzne výsledky pre jednotlivý obrazy.

Najlepšie výsledky boli dosiahnuté použitím farebného modelu ORCT. Ostatné farebné modely boli efektívnejší len v niektorých príkladov. Už bolo spomenuté, že model RGB nie je vhodný pri kompresii, a implementované farebné modely sú vhodnejšie, to ale vždy platiť nemusí. Najlepší výsledok u obrázku pepper.bmp zakódovaný aritmetickým kodérom bol napríklad dosiahnutý použitím farebného modelu RGB.

Prediktory PAETH a MED dávali podobné výsledky, ale väčšinou s prediktorom MED bol dosiahnutý lepší kompresný pomer. Z prediktorov najvhodnejší bolo prediktor GAP, ale v niektorých prípadoch s prediktormi MED a PAETH boli dosiahnuté oveľa lepšie výsledky. Voľba prediktoru závisí aj na entropickom kodéru. Z testov sa dá vidieť, že použitím aritmetického kodéra (a PPM kodéra s kontextom 0) u polovice obrázku boli dosiahnuté najlepšie výsledky prediktorom GAP a u druhej polovice s prediktorom MED. U PPM kodéra s kontextom 1 a 2 u 9 obrázkoch z 14 bol lepším prediktorom GAP. Podľa testoch sa dá povedať, že s aritmetickým kodérom sa oplatí použiť prediktor MED, ktorý je rýchlejší a jednoduchšie implementovateľný ako prediktor GAP, ale u PPM kodéru je lepšie použiť prediktor GAP.

Najefektívnejší entropický kodér z implementovaných podľa testov je PPM kodér. Adaptívny aritmetický kodér dáva väčšinou lepšie výsledky ako statický, ale statický aritmetický kodér je rýchlejší, takže v niektorých prípadoch sa môže oplatiť použitie statického aritmetického kodéra. PPM kodér je oveľa pomalší ako aritmetické kodéry, a potrebuje viac pamäti. Vo väčšine prípadov, ale dosiahneme s ním oveľa lepšie výsledky ako aritmetickým kodérom. Boli prevedené testy s PPM kodérom s rôznymi kontextmi, najlepšie výsledky pre testovacie obrázky však boli dosiahnuté s kontextmi 1 a 2. Výsledky kódovania PPM kodérom s kontextom rádu 3 ukázalo horšie výsledky pre každý obrázok (na všetky možné kombinácie farebných modelov a prediktorov). To je možné vysvetliť tým, že kóduje kodér veľa krát escape symbol, čo zhoršuje kompresný pomer.

MTF transformácia a RLE kódovanie na testových obrázkoch neboli veľmi efektívne: okrem obrázkov lena.bmp a pepper.bmp používanie RLE kodéra neviedlo k lepšej kompresii, MTF transformácia zhoršila mieru kompresie pri všetkých obrázkoch. To ale neznamená, že tieto metódy nie sú vhodné na kompresiu. RLE kodér je veľmi efektívny, keď obraz obsahuje veľa dlhých behov, MTF je efektívnejší po použití transformácie, ako je Burrows-Wheelerova transformácia.

Kapitola 8

Záver

Cieľom tejto bakalárskej práce bolo vytvorenie multiplatformovej knižnice (testovaná bola na operačných systémoch Microsoft Windows 7 a Ubuntu 10.10) pre bezstratovú kompresiu obrazu. Knižnica obsahuje funkcie na prevody farebného modelu z RGB do LYUV, ORCT a LOCO-I, ich prevod späť do RGB, funkcie na predikcie – tri lineárne a tri nelineárne prediktory (nelineárne sú GAP, PAETH a MED), je implementovaná MTF transformácia, RLE kodér, a tri entropické kodéry – statický a adaptívny aritmetický kodér a PPM kodér. Knižnica bola vytvorená tak, aby všetky tieto implementované časti kompresie bolo možné voľne kombinovať pomocou parametrov programu.

Po implementácii knižnica bola otestovaná na sade obrázkov (tabuľka B.1). Dosiahnuté výsledky sú uvedené v tabuľkách A.1 až A.6. Podľa prevedených testov najvhodnejšie z implementovaných farebných modelov pre kompresiu obrazov je model ORCT. Okrem jedného obrázku u všetkých obrázkoch boli dosiahnuté najlepšie výsledky použitím tohto modelu. Z prediktorov najlepšie výsledky boli dosiahnuté prediktorom GAP. Z kodérov najlepšie výsledky dával PPM kodér. Experimenty ale ukázali, že používanie dlhších kontextov sa zhoršuje kompresný pomer. Prevedené testy ukazujú, že veľkosti zakódovaných obrázkov boli priemerne o 17% menšie, ako rovnaké obrázky vo formáte PNG.

V rámci rozšírenia knižnice by bolo možné pridať ďalšie farebné modely, prediktory, pridať ďalšie kroky kompresie, ktoré umožňujú lepšiu kompresiu, ako napríklad Burrows-Wheelerova transformácia. Bolo by dobré vyskúšať iné varianty PPM kodéra. Pretože MTF transformácia nezlepšila kompresný pomer, oplatilo by implementovať iné algoritmy ako náhradu MTF, napríklad IF (Inversion Frequencies) alebo WFC (Weigthed Frequency Count).

Literatura

[1] Abel, J.: Post BWT stages of the Burrows-Wheeler compression algorithm. *Softw. Pract. Exper.*, ročník 40, 2010: s. 751–777, ISSN 0038-0644.
URL
http://www.juergen-abel.info/Preprints/Preprint_Post_BWT_Stages.pdf

[2] Avramovic, A.; Reljin, B.: Gradient edge detection predictor for image lossless compression. In *ELMAR, 2010 PROCEEDINGS*, 2010, ISSN 1334-2630, s. 131 –134.

[3] Bodden, E.; Clasen, M.; Kneis, J.: Arithmetic Coding revealed - A guided tour from theory to praxis. Technická Zpráva SABLE-TR-2007-5, Sable Research Group, School of Computer Science, McGill University, Montréal, Québec, Canada, 2007.

[4] Burrows, M.; Wheeler, D. J.: A block-sorting lossless data compression algorithm. Technická zpráva, 1994.

[5] Hakkennes, E.; Vassiliadis, S.: *Hardwired Paeth Codec for Portable Network Graphics (PNG)*, ročník 2. Los Alamitos, CA, USA: IEEE Computer Society, 1999, ISBN 0-7695-0321-7, 2318 s.,
doi:http://doi.ieeecomputersociety.org/10.1109/EURMIC.1999.794796.

[6] Hao, P.; Shi, Q.: Comparative study of color transforms for image coding and derivation of integer reversible color transform. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, ročník 3, 2000, s. 224 –227 vol.3, doi:10.1109/ICPR.2000.903526.

[7] Hua, Y.; Hu, B.: The realization of fast gradient adjusted prediction method for lossless video compression with low memory requirement. In *Communications, Circuits and Systems, 2004. ICCCAS 2004. 2004 International Conference on*, ročník 2, 2004, s. 1165 – 1169 Vol.2, doi:10.1109/ICCCAS.2004.1346382.

[8] Nosek, A.: *Implementace kompresní metody PPM*. Diplomová práce, ČVUT v Praze, Praha, 2006.

[9] Poynton, C.: A Guided Tour of Color Space.
URL http://www.poynton.com/PDFs/Guided_tour.pdf

[10] Roelofs, G.: *PNG: The Definitive Guide*. O'Reilly, 1999, 321 s.
URL http://www.libpng.org/pub/png/book/

[11] Salomon, D.: *Data Compression: The Complete Reference*. Springer-Verlag, 2007, ISBN 1-84628-602-6, xxv + 1092 s., with contributions by Giovanni Motta and David Bryant.

[12] Santa-Cruz, D.; Ebrahimi, T.; Askelof, J.; aj.: JPEG 2000 still image coding versus other standards. 2000.
URL http://www.jpeg.org/public/wg1n1816.pdf

[13] Vleuten, R.; Egner, S.: Lossless and Fine-Granularity Scalable Near-Lossless Color Image Compression. 2002, ISSN 1068-0314, str. 477, doi:10.1109/DCC.2002.1000020.
URL http://www.win.tue.nl/wic2004/28.pdf

[14] Weinberger, M.; Seroussi, G.; Sapiro, G.: The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS. *Image Processing, IEEE Transactions on*, ročník 9, č. 8, 2000: s. 1309 –1324, ISSN 1057-7149, doi:10.1109/83.855427.

Príloha A

Tabuľky dosiahnutých výsledkov

F. m.	Obraz	lena	mandrill	pepper	airplane	BoatsColor	ZeldaColor	barbara
	BMP	768,1	768,1	768,1	768,1	1329,8	1329,8	1215,1
	PNG	463,8	612,0	494,5	414,7	712,4	688,7	800,2
	PNG	0,60	0,80	0,64	0,54	0,54	0,52	0,66
RGB	Bez pred.	698,6	734,4	701,1	631,9	1187,1	1250,9	1155,9
	K. p.	0,91	0,96	0,91	0,82	0,89	0,94	0,95
	GAP	**446,7**	**613,1**	**484,8**	**412,9**	721,9	**660,6**	**801,0**
	K. p.	**0,58**	**0,80**	**0,63**	**0,54**	0,54	**0,50**	**0,66**
	PAETH	468,0	623,5	505,4	422,8	718,0	686,5	815,7
	K. p.	0,61	0,81	0,66	0,55	0,54	0,52	0,67
	MED	461,9	618,2	500,9	414,7	**702,5**	671,7	806,1
	K. p.	0,60	0,80	0,65	0,54	**0,53**	0,51	0,66
LYUV	Bez pred.	665,9	720,2	710,0	552,6	1071,8	1147,5	1032,5
	K. p.	0,87	0,94	0,92	0,72	0,81	0,86	0,85
	GAP	**440,3**	**589,3**	**488,9**	**385,4**	627,4	604,5	677,4
	K. p.	**0,57**	**0,77**	**0,64**	**0,50**	0,47	0,45	0,56
	PAETH	461,6	604,7	510,7	399,1	608,1	601,9	625,0
	K. p.	0,60	0,79	0,66	0,52	0,46	0,45	0,51
	MED	456,0	599,5	506,5	392,0	**594,7**	**591,1**	**623,6**
	K. p.	0,59	0,78	0,66	0,51	**0,45**	**0,44**	**0,51**
LOCO-I	Bez pred.	667,1	706,2	711,9	549,3	1030,4	1113,1	1025,3
	K. p.	0,87	0,92	0,93	0,72	0,77	0,84	0,84
	GAP	**447,9**	**593,2**	**497,3**	**388,2**	612,5	593,8	662,8
	K. p.	**0,58**	**0,77**	**0,65**	**0,51**	0,46	0,45	0,55
	PAETH	469,5	608,4	518,3	402,7	589,6	586,6	577,1
	K. p.	0,61	0,79	0,67	0,52	0,44	0,44	0,47
	MED	464,0	603,5	514,1	395,7	**579,4**	**576,5**	**573,1**
	K. p.	0,60	0,79	0,67	0,52	**0,44**	**0,43**	**0,47**
ORCT	Bez pred.	660,0	703,6	713,9	544,3	1036,7	1109,0	1026,2
	K. p.	0,86	0,92	0,93	0,71	0,78	0,83	0,84
	GAP	**438,0**	**585,6**	**486,5**	**381,6**	612,0	592,8	662,3
	K. p.	**0,57**	**0,76**	**0,63**	**0,50**	0,46	0,45	0,55
	PAETH	459,6	601,1	508,1	395,6	588,8	585,7	576,6
	K. p.	0,60	0,78	0,66	0,52	0,44	0,44	0,47
	MED	454,1	595,9	503,7	388,5	**578,7**	**575,6**	**572,5**
	K. p.	0,59	0,78	0,66	0,51	**0,44**	**0,43**	**0,47**

Tabuľka A.1: Výsledky adaptívneho aritmetického kodéra časť 1

F. m.	Obraz	girl	monarch	lenna_full	cablecar	cornfield	pens	soccer
	BMP	1215,1	1152,1	1002,0	720,1	720,1	720,1	720,1
	PNG	676,6	602,9	582,2	411,2	441,2	409,8	451,0
	PNG	0,56	0,52	0,58	0,57	0,61	0,57	0,63
RGB	Bez pred.	1100,5	1031,1	896,6	653,2	671,3	674,4	664,6
	K. p.	0,91	0,89	0,89	0,91	0,93	0,94	0,92
	GAP	**653,8**	**593,5**	**601,0**	421,7	464,4	408,6	452,3
	K. p.	**0,54**	**0,52**	**0,60**	0,59	0,64	0,57	0,63
	PAETH	675,4	614,7	613,4	425,9	470,2	416,8	451,1
	K. p.	0,56	0,53	0,61	0,59	0,65	0,58	0,63
	MED	661,6	598,8	604,6	**418,9**	**459,8**	**402,2**	**438,8**
	K. p.	0,54	0,52	0,60	**0,58**	**0,64**	**0,56**	**0,61**
LYUV	Bez pred.	1018,2	976,1	820,8	600,4	606,5	604,4	608,0
	K. p.	0,84	0,85	0,82	0,83	0,84	0,84	0,84
	GAP	603,7	494,8	506,7	**388,4**	**413,9**	**374,3**	408,7
	K. p.	0,50	0,43	0,51	**0,54**	**0,57**	**0,52**	0,57
	PAETH	565,6	503,4	495,4	396,9	424,4	388,5	414,0
	K. p.	0,47	0,44	0,49	0,55	0,59	0,54	0,57
	MED	**560,6**	**485,7**	**490,1**	390,9	416,5	377,9	**404,7**
	K. p.	**0,46**	**0,42**	**0,49**	0,54	0,58	0,52	**0,56**
LOCO-I	Bez pred.	997,1	986,0	807,6	592,5	600,4	587,7	592,7
	K. p.	0,82	0,86	0,80	0,82	0,83	0,82	0,82
	GAP	589,2	496,9	495,2	**391,3**	**416,3**	**373,3**	406,8
	K. p.	0,48	0,43	0,49	**0,54**	**0,58**	**0,52**	0,56
	PAETH	508,6	501,9	479,6	399,3	427,2	387,7	413,3
	K. p.	0,42	0,44	0,48	0,55	0,59	0,54	0,57
	MED	**502,0**	**482,8**	**473,9**	393,7	419,7	377,7	**404,7**
	K. p.	**0,41**	**0,42**	**0,47**	0,55	0,58	0,52	**0,56**
ORCT	Bez pred.	998,3	989,9	812,0	591,8	602,1	588,9	593,8
	K. p.	0,82	0,86	0,81	0,82	0,84	0,82	0,82
	GAP	587,4	494,7	495,8	**389,1**	**414,4**	**371,7**	405,0
	K. p.	0,48	0,43	0,49	**0,54**	**0,58**	**0,52**	0,56
	PAETH	507,2	499,8	480,1	397,1	425,3	385,9	411,3
	K. p.	0,42	0,43	0,48	0,55	0,59	0,54	0,57
	MED	**500,6**	**480,9**	**474,4**	391,4	417,6	375,5	**402,4**
	K. p.	**0,41**	**0,42**	**0,47**	0,54	0,58	0,52	**0,56**

Tabuľka A.2: Výsledky adaptívneho aritmetického kodéra časť 2

F. m.	Obraz	lena	mandrill	pepper	airplane	BoatsColor	ZeldaColor	barbara
	BMP	768,1	768,1	768,1	768,1	1329,8	1329,8	1215,1
	PNG	463,8	612,0	494,5	414,7	712,4	688,7	800,2
	PNG	0,60	0,80	0,64	0,54	0,54	0,52	0,66
LYUV	GAP	**447,3**	**602,7**	**494,0**	**369,2**	656,3	630,5	710,5
	K. p.	**0,58**	**0,78**	**0,64**	**0,48**	0,49	0,47	0,58
	PAETH	468,4	616,4	515,4	408,9	640,4	634,8	668,4
	K. p.	0,61	0,80	0,67	0,53	0,48	0,48	0,55
	MED	462,7	611,2	510,9	401,8	**629,8**	**624,0**	**666,2**
	K. p.	0,60	0,80	0,67	0,52	**0,47**	**0,47**	**0,55**
LOCO-I	GAP	**448,8**	**596,1**	**500,0**	**391,5**	621,3	598,0	675,5
	K. p.	**0,58**	**0,78**	**0,65**	**0,51**	0,47	0,45	0,56
	PAETH	470,4	611,2	520,6	405,1	599,3	594,6	606,0
	K. p.	0,61	0,80	0,68	0,53	0,45	0,45	0,50
	MED	464,9	606,0	516,4	398,0	**588,6**	**583,9**	**601,2**
	K. p.	0,61	0,79	0,67	0,52	**0,44**	**0,44**	**0,49**
ORCT	GAP	**439,8**	**587,6**	**491,3**	**384,3**	620,7	596,9	674,8
	K. p.	**0,57**	**0,77**	**0,64**	**0,50**	0,47	0,45	0,56
	PAETH	461,4	603,0	512,7	397,4	598,2	593,4	605,0
	K. p.	0,60	0,79	0,67	0,52	0,45	0,45	0,50
	MED	455,9	597,6	508,4	390,3	**587,6**	**582,7**	**600,1**
	K. p.	0,59	0,78	0,66	0,51	**0,44**	**0,44**	**0,49**

F. m.	Obraz	girl	monarch	lenna_full	cablecar	cornfield	pens	soccer
	BMP	1215,1	1152,1	1002,0	720,1	720,1	720,1	720,1
	PNG	676,6	602,9	582,2	411,2	441,2	409,8	451,0
	PNG	0,56	0,52	0,58	0,57	0,61	0,57	0,63
LYUV	GAP	625,5	524,7	530,6	**398,1**	**430,0**	**388,2**	426,2
	K. p.	0,51	0,46	0,53	**0,55**	**0,60**	**0,54**	0,59
	PAETH	598,6	535,2	521,1	405,7	439,0	400,6	428,8
	K. p.	0,49	0,46	0,52	0,56	0,61	0,56	0,60
	MED	**593,5**	**519,4**	**515,6**	399,4	430,6	388,9	**418,8**
	K. p.	**0,49**	**0,45**	**0,51**	0,55	0,60	0,54	**0,58**
LOCO-I	GAP	592,6	505,6	504,4	**393,4**	**421,7**	**375,7**	410,7
	K. p.	0,49	0,44	0,50	**0,55**	**0,59**	**0,52**	0,57
	PAETH	531,4	511,9	493,0	400,8	431,8	389,5	416,1
	K. p.	0,44	0,44	0,49	0,56	0,60	0,54	0,58
	MED	**523,8**	**493,2**	**486,6**	395,2	423,9	379,2	**407,2**
	K. p.	**0,43**	**0,43**	**0,49**	0,55	0,59	0,52	**0,57**
ORCT	GAP	590,6	502,9	505,1	**391,0**	**419,8**	**374,0**	408,6
	K. p.	0,49	0,44	0,50	**0,54**	**0,58**	**0,52**	0,57
	PAETH	529,4	509,4	493,4	398,5	429,9	387,6	413,8
	K. p.	0,44	0,44	0,49	0,55	0,60	0,54	0,57
	MED	**522,0**	**491,0**	**487,0**	392,7	421,7	377,0	**404,5**
	K. p.	**0,43**	**0,43**	**0,49**	0,55	0,59	0,52	**0,56**

Tabuľka A.3: Výsledky štatického aritmetického kodéra

F. m.	Obraz	lena	mandrill	pepper	airplane	BoatsColor	ZeldaColor	barbara
	BMP	768,1	768,1	768,1	768,1	1329,8	1329,8	1215,1
	PNG	463,8	612,0	494,5	414,7	712,4	688,7	800,2
	PNG	0,60	0,80	0,64	0,54	0,54	0,52	0,66
LYUV	GAP	**440,0**	**589,5**	**488,8**	**385,1**	627,2	604,4	677,4
	K. p.	**0,57**	**0,77**	**0,64**	**0,50**	0,47	0,45	0,56
	PAETH	461,3	604,9	510,5	398,5	607,9	601,6	625,1
	K. p.	0,60	0,79	0,66	0,52	0,46	0,45	0,51
	MED	455,7	599,7	506,3	391,5	**597,2**	**590,9**	**623,7**
	K. p.	0,59	0,78	0,66	0,51	**0,45**	**0,44**	**0,51**
LOCO-I	GAP	**447,7**	**593,4**	**497,1**	**387,6**	612,4	593,8	662,8
	K. p.	**0,58**	**0,77**	**0,65**	**0,50**	0,46	0,45	0,55
	PAETH	469,3	608,7	518,2	401,9	589,4	586,4	577,2
	K. p.	0,61	0,79	0,67	0,52	0,44	0,44	0,48
	MED	463,8	603,7	513,9	394,9	**579,3**	**576,3**	**573,1**
	K. p.	0,60	0,79	0,67	0,51	**0,44**	**0,43**	**0,47**
ORCT	GAP	**437,7**	**585,8**	**486,4**	**381,0**	611,8	592,8	662,3
	K. p.	**0,57**	**0,76**	**0,63**	**0,50**	0,46	0,45	0,55
	PAETH	459,4	601,3	507,9	394,9	588,7	585,5	576,6
	K. p.	0,60	0,78	0,66	0,51	0,44	0,44	0,47
	MED	453,9	596,1	503,5	387,7	**578,5**	**575,4**	**572,5**
	K. p.	0,59	0,78	0,66	0,50	**0,43**	**0,43**	**0,47**

F. m.	Obraz	girl	monarch	lenna_full	cablecar	cornfield	pens	soccer
	BMP	1215,1	1152,1	1002,0	720,1	720,1	720,1	720,1
	PNG	676,6	602,9	582,2	411,2	441,2	409,8	451,0
	PNG	0,56	0,52	0,58	0,57	0,61	0,57	0,63
LYUV	GAP	603,4	494,6	506,3	**388,5**	**413,6**	**374,2**	408,7
	K. p.	0,50	0,43	0,51	**0,54**	**0,57**	**0,52**	0,57
	PAETH	565,2	503,3	495,1	396,8	424,2	388,3	414,0
	K. p.	0,47	0,44	0,49	0,55	0,59	0,54	0,57
	MED	**560,2**	**485,5**	**489,8**	390,9	416,3	377,8	**404,7**
	K. p.	**0,46**	**0,42**	**0,49**	0,54	0,58	0,52	**0,56**
LOCO-I	GAP	589,0	496,8	494,9	**391,4**	**416,1**	**373,3**	406,8
	K. p.	0,48	0,43	0,49	**0,54**	**0,58**	**0,52**	0,56
	PAETH	508,4	501,8	479,4	399,3	427,0	387,6	413,3
	K. p.	0,42	0,44	0,48	0,55	0,59	0,54	0,57
	MED	**501,7**	**482,6**	**473,7**	393,7	419,5	377,6	**404,7**
	K. p.	**0,41**	**0,42**	**0,47**	0,55	0,58	0,52	**0,56**
ORCT	GAP	587,2	494,5	495,6	**389,1**	**414,2**	**371,7**	405,1
	K. p.	0,48	0,43	0,49	**0,54**	**0,58**	**0,52**	0,56
	PAETH	507,0	499,7	479,9	397,1	425,1	385,8	411,3
	K. p.	0,42	0,43	0,48	0,55	0,59	3,54	0,57
	MED	**500,3**	**480,8**	**474,2**	391,4	417,4	375,4	**402,4**
	K. p.	**0,41**	**0,42**	**0,47**	0,54	0,58	0,52	**0,56**

Tabuľka A.4: Výsledky PPM kodéra s kontextom 0

F. m.	Obraz	lena	mandrill	pepper	airplane	BoatsColor	ZeldaColor	barbara
	BMP	768,1	768,1	768,1	768,1	1329,8	1329,8	1215,1
	PNG	463,8	612,0	494,5	414,7	712,4	688,7	800,2
	PNG	0,60	0,80	0,64	0,54	0,54	0,52	0,66
LYUV	GAP	**435,6**	**596,4**	**478,0**	**373,0**	591,1	**582,2**	653,7
	K. p.	**0,57**	**0,78**	**0,62**	**0,49**	0,44	**0,44**	0,54
	PAETH	454,7	612,3	494,9	389,1	599,2	594,9	585,5
	K. p.	0,59	0,80	0,64	0,51	0,45	0,45	0,48
	MED	448,7	606,8	489,8	381,9	**588,4**	584,3	**589,0**
	K. p.	0,58	0,79	0,64	0,50	**0,44**	0,44	**0,48**
LOCO-I	GAP	**444,0**	**599,8**	**485,0**	**374,5**	572,4	568,1	634,5
	K. p.	**0,58**	**0,78**	**0,63**	**0,49**	0,43	0,43	0,52
	PAETH	463,2	615,3	501,6	391,2	577,4	576,6	518,8
	K. p.	0,60	0,80	0,65	0,51	0,43	0,43	0,43
	MED	457,4	610,1	496,6	384,0	**567,2**	**566,5**	**516,7**
	K. p.	0,60	0,79	0,65	0,50	**0,43**	**0,43**	**0,43**
ORCT	GAP	**433,3**	**590,9**	**475,7**	**367,5**	571,6	567,1	634,0
	K. p.	**0,56**	**0,77**	**0,62**	**0,48**	0,43	0,43	0,52
	PAETH	452,6	607,3	493,0	384,1	576,6	575,7	518,3
	K. p.	0,59	0,79	0,64	0,50	0,43	0,43	0,43
	MED	446,8	601,7	487,7	376,9	**566,3**	**565,6**	**516,1**
	K. p.	0,58	0,78	0,63	0,49	**0,43**	**0,43**	**0,42**

F. m.	Obraz	girl	monarch	lenna_full	cablecar	cornfield	pens	soccer
	BMP	1215,1	1152,1	1002,0	720,1	720,1	720,1	720,1
	PNG	676,6	602,9	582,2	411,2	441,2	409,8	451,0
	PNG	0,56	0,52	0,58	0,57	0,61	0,57	0,63
LYUV	GAP	590,6	**463,6**	478,0	**368,8**	**397,4**	**363,4**	**390,8**
	K. p.	0,49	**0,40**	0,48	**0,51**	**0,55**	**0,50**	**0,54**
	PAETH	536,6	483,8	471,1	385,2	414,7	385,1	409,8
	K. p.	0,44	0,42	0,47	0,53	0,58	0,53	0,57
	MED	**536,2**	467,7	**465,9**	379,2	406,4	374,0	399,4
	K. p.	**0,44**	0,41	**0,46**	0,53	0,56	0,52	0,55
LOCO-I	GAP	571,8	**457,1**	466,1	**370,2**	**397,9**	**363,0**	**389,9**
	K. p.	0,47	**0,40**	0,47	**0,51**	**0,55**	**0,50**	**0,54**
	PAETH	457,5	476,4	452,6	386,2	416,2	384,3	408,8
	K. p.	0,38	0,41	0,45	0,54	0,58	0,53	0,57
	MED	**451,9**	458,6	**446,9**	380,4	408,3	374,0	399,4
	K. p.	**0,37**	0,40	**0,45**	0,53	0,56	0,52	0,55
ORCT	GAP	570,3	**455,4**	466,4	**367,2**	**394,9**	**359,9**	**386,5**
	K. p.	0,47	**0,40**	0,47	**0,51**	**0,55**	**0,50**	**0,54**
	PAETH	456,1	474,9	452,8	383,6	413,5	381,8	406,0
	K. p.	0,38	0,41	0,45	0,53	0,57	0,53	0,56
	MED	**450,5**	457,0	**447,1**	377,6	405,3	371,0	396,0
	K. p.	**0,37**	0,40	**0,45**	0,52	0,56	0,52	0,55

Tabuľka A.5: Výsledky PPM kodéra s kontextom 1

F. m.	Obraz	lena	mandrill	pepper	airplane	BoatsColor	ZeldaColor	barbara
	BMP	768,1	768,1	768,1	768,1	1329,8	1329,8	1215,1
	PNG	463,8	612,0	494,5	414,7	712,4	688,7	800,2
	PNG	0,60	0,80	0,64	0,54	0,54	0,52	0,66
LYUV	GAP	**447,0**	**624,2**	**494,3**	**377,6**	594,7	585,7	655,2
	K. p.	**0,58**	**0,81**	**0,64**	**0,49**	0,45	0,44	0,47
	PAETH	466,7	640,3	513,0	394,1	599,1	595,2	**585,9**
	K. p.	0,61	0,83	0,67	0,51	0,45	0,45	**0,48**
	MED	460,0	634,6	507,6	386,4	**588,2**	**583,9**	589,0
	K. p.	0,60	0,83	0,66	0,50	**0,44**	**0,44**	0,48
LOCO-I	GAP	**456,4**	**627,5**	**500,2**	**378,5**	572,8	567,6	627,8
	K. p.	**0,59**	**0,82**	**0,65**	**0,49**	0,43	0,43	0,52
	PAETH	476,6	643,6	518,4	395,7	572,3	570,3	512,5
	K. p.	0,62	0,84	0,67	0,52	0,43	0,43	0,42
	MED	470,0	637,9	513,0	387,9	**561,2**	**558,7**	**509,3**
	K. p.	0,61	0,83	0,67	0,51	**0,42**	**0,42**	**0,42**
ORCT	GAP	**444,6**	**618,1**	**490,8**	**371,1**	572,0	566,3	627,3
	K. p.	**0,58**	**0,80**	**0,64**	**0,48**	0,43	0,43	0,52
	PAETH	464,8	635,1	510,0	388,1	571,4	569,1	512,1
	K. p.	0,61	0,83	0,66	0,51	0,43	0,43	0,42
	MED	458,2	628,9	504,3	380,5	**560,3**	**557,5**	**508,8**
	K. p.	0,60	0,82	0,66	0,50	**0,42**	**0,42**	**0,42**

F. m.	Obraz	girl	monarch	lenna_full	cablecar	cornfield	pens	soccer
	BMP	1215,1	1152,1	1002,0	720,1	720,1	720,1	720,1
	PNG	676,6	602,9	582,2	411,2	441,2	409,8	451,0
	PNG	0,56	0,52	0,58	0,57	0,61	0,57	0,63
LYUV	GAP	588,1	**463,0**	482,1	**371,6**	**403,3**	**369,7**	**401,9**
	K. p.	0,48	**0,40**	0,48	**0,52**	**0,56**	**0,51**	**0,56**
	PAETH	535,8	486,1	475,7	387,2	420,0	390,5	420,5
	K. p.	0,44	0,42	0,47	0,54	0,58	0,54	0,58
	MED	**535,4**	469,7	**469,6**	380,6	410,6	378,1	409,2
	K. p.	**0,44**	0,41	**0,47**	0,53	0,57	0,53	0,57
LOCO-I	GAP	561,7	**453,5**	468,0	**372,6**	**403,0**	**368,5**	**400,3**
	K. p.	0,46	**0,39**	0,47	**0,52**	**0,56**	**0,51**	**0,56**
	PAETH	450,9	477,5	454,7	388,3	420,8	389,3	419,0
	K. p.	0,37	0,41	0,45	0,54	0,58	0,54	0,58
	MED	**444,5**	458,8	**447,9**	381,7	411,8	377,7	408,6
	K. p.	**0,37**	0,40	**0,45**	0,53	0,57	0,52	0,57
ORCT	GAP	560,1	**451,8**	468,3	**369,2**	**399,7**	**365,1**	**396,3**
	K. p.	0,46	**0,39**	0,47	**0,51**	**0,56**	**0,51**	**0,55**
	PAETH	449,4	475,5	454,8	385,3	417,6	386,6	415,8
	K. p.	0,37	0,41	0,45	0,54	0,58	0,54	0,58
	MED	**443,1**	457,0	**448,1**	378,7	408,5	374,5	404,6
	K. p.	**0,36**	0,40	**0,45**	0,53	0,57	0,52	0,56

Tabuľka A.6: Výsledky PPM kodéra s kontextom 2

Príloha B

Použité obrázky

[a] Spectrum.svg:
http://upload.wikimedia.org/wikipedia/commons/c/cc/Spectrum.svg

[b] Additive_color.svg:
http://upload.wikimedia.org/wikipedia/commons/2/2b/Additive_color.svg

[c] Substractive_color.svg:
http://upload.wikimedia.org/wikipedia/commons/a/a2/Substractive_color.svg

[d] Avl3119color4a.svg:
http://upload.wikimedia.org/wikipedia/commons/a/a9/Avl3119color4a.svg

B.1 Testové obrázky

Obrázok	Zdroj
lena.bmp	http://www.bilsen.com
mandrill.bmp	http://www.bilsen.com
pepper.bmp	http://www.hlevkin.com
airplane.bmp	http://www.hlevkin.com
BoatsColor.bmp	http://www.hlevkin.com
ZeldaColor.bmp	http://www.hlevkin.com
barbara.bmp	http://www.hlevkin.com
girl.bmp	http://www.hlevkin.com
monarch.bmp	http://www.hlevkin.com
Lenna_full.bmp	http://www.hlevkin.com
cablecar.bmp	http://www.hlevkin.com
cornfield.bmp	http://www.hlevkin.com
pens.bmp	http://www.hlevkin.com
soccer.bmp	http://www.hlevkin.com

Tabuľka B.1: Testové obrázky

Príloha C

Obsah CD

Priložený CD obsahuje:

- Zdrojové súbory knižnice
- Programovú dokumentáciu knižnice (Doxygen)
- Technickú správu vo formátu PDF
- Zdrojové súbory technickej správy pre LaTeX
- Sadu testovacích obrázkov